Arduino documentation

Erik Verberne
erik@verberne.net
Version: 1.23

Download: http://bit.ly/eve_arduino
Sample sketches: http://bit.ly/eve_arduinosketches

Foreword

I work in the Netherlands at a school as a teacher ICT Management. At my department we also teach Application Developers. In December 2013 I decided to investigate whether it would be interesting for our Application Developers to take a minor in the subject Embedded Software and specifically Arduino. So, I bought some Arduino boards and a couple of sensors and actuators to experiment with. I soon found out that I needed to document my findings. This document is the result of experimenting in my free time. Since Arduino is open source, I decided to donate this document to other Arduino fans in the world. As long as it is clear that I'm the author of this document, you can use this document for any non-commercial or educational projects.

I derived and simplified most sketches from samples that came with the corresponding libraries. Most schematics and most photographs are my own work.

Have fun with it. I certainly enjoyed writing it.

Erik Verberne

January 29th, 2014

Table of contents

Foreword **2**
Table of contents **3**
Introduction **11**
Version history **13**
Arduino Boards **23**
1. Arduino 25
2. Arduino UNO R3 28
3. Yourduino RoboRed 30
4. Seeeduino V4.2 32
5. Seeeduino Lotus v1.1 34
6. Arduino Mega 2560 R3 37
7. Arduino Mega 2560 R3 CH340 39
8. Arduino Mega 2560 Pro Mini 41
9. Arduino Nano 43
10. Sparkfun Pro Micro - 5V/16MHz 45
11. Arduino Pro Mini ATmega168U 5v, 16MHz 47
12. Arduino Pro Mini ATmega328p 3.3V, 8MHz 49
13. Teensy 3.1 & 3.2 51
14. LoRa: Nexus 56
15. The Things Uno 58
16. LoRa: Sodaq One 60
18. Boarduino 72
19. AVR Development board as Arduino 75
20. Arduino on a breadboard 77
21. Attiny45/Attiny85 79
22. Attiny85 Digispark board 83
23. ESP8266 MCU 87

Arduino IDE **89**
24. Arduino IDE 91
25. Serial Monitor 93
26. Libraries 95
27. Board management 97

Misc. software **99**
28. Processing 101
29. Fritzing 104

Programming/ Programmers **105**
30. Programming an Arduino Board through USB 107
31. USBasp v2.0 programmer 108
32. AVR ISP to ICSP Adapter 110
33. Uploading the bootloader by using an Arduino as ISP 112
34. AVR Atmega8/168/328 Development board 113
35. Self-made Attiny 45/85 ISP adapter 115
36. ST Link V2 116

Sound **119**
37. Buzzer 121
38. Piezo Speaker 123
39. Mini MP3 Player MP3-TF-16P 125

LED displays **131**

40. Onboard LED D13 133
41. LED 134
42. RGB LED board 137
43. 10 segment LED Bar Graph F2510BH 139
44. 8x8 DOT Matrix 1088AS 141
45. 8x8 DOT Matrix 1388ASR 143
46. 8x8 DOT Matrix HL-M1588BR 145
47. 8x8 DOT matrix on PCB with MAX7219 chip 146
48. Single Digit 7-Segment Display 150
49. 4 Digits 7-Segment Display Common Cathode (HS420561K-32) 154
50. 4 Digits 7-Segment Display Common Cathode (5641AS) 158
51. 4 Digits 7-Segment Display Common Anode 159
52. 8 Digits 7-Segment Display with TM1638 chip 161
53. MAX7219 LED driver 164
54. WS2812B RGB LED breakout-board 174
55. WS2812B RGB LED strip 177
56. WS2812B RGB LED ring 179

LCD/TFT/OLED & e-Paper displays 183

57. Nokia 5110/3310 LCD 185
58. 16x2 Display 1602A 189
59. 16x2 Display with LCM1602 chip (I2C) 192
60. Adafruit 2.8" TFT Resistive Touch Shield v2 195
61. 2.4” TFT LCD Shield Touch Board 199
62. 0.96 inch 128x64 OLED display (I2C) 204
63. 0.91 inch 128x32 OLED display (I2C) 207
64. Waveshare 2.7 e-Paper HAT with an Arduino 211

Input sensors 217

65. Switches 219
66. Photo-interrupter ITR8102 230
67. Photo-interrupter RPI-352 232
68. 4x4 Keypad 233
69. Potentiometer 236
70. Joystick 238
71. Rotary encoder 241
72. Nunchuk with connection board 244
73. Nunchuk with connection board on A2..A5 248
74. Nunchuk without connection board 254
75. Wii u.Draw Game Tablet 255
76. PlayStation controller (PSX/PS2) 260
77. PS/2 keyboard 263
78. RGB and Gesture sensor APDS-9960 266

Sensors 269

79. Temperature Sensor LM35 271
80. Temperature and Humidity sensor board DHT11 273
81. DS18B20 temperature sensor 276
82. DS18B20 temperature sensor 279
83. Barometer BMP180 281
84. Barometer BMP280-5V 284
85. Barometer BME280-5V 287

86. Water sensor 290
87. Capacitive Soil Moisture sensor v1.2 292
88. Touch Sensor TTP223B 294
89. Distance Sensor HC-SR04 296
90. PIR/Motion Sensor HC-SR501 300
91. Photo resistor (LDR) 302
92. Pindiode - Photodiode BPW34 304
93. Laser Sensor module 306
94. Flame Sensor (IR photo transistor) 308
95. IR proximity sensor board (line finder) 310
96. Sound detection FC-04 312
97. Sound detection with digital and analog output 314
98. 6DOF MPU-6050 3 Axis Gyro & Accelerometer 316
99. GPS XM37-1612 (GY-NEO6Mv2?) 324
100. NEO6 GPS moduel GPS GY-NEO6MV2 327
101. GPS GY-GPS6MV2 328
102. MQ-3 alcohol gas sensor board 331
103. MQ-6 LPG, iso-butane and propane gas sensor 333
104. Load cells 336
105. Load cell amplifier HX711 338
Storage 341
106. SD Card 343
107. Mifare RFID RC522 347
108. Adafruit PN532 NFC/RFID controller breakout board 350
Real Time Clock 357
109. RTC module with DS1302 chip 359
110. Tiny RTC I²C module with DS1307 chip 362
Servo's, Motors & Steppers 365
111. Standard Servo 367
112. Tinkerkit Braccio Robot 370
113. DC/Stepper Motor Driver board L298n 377
114. Stepper Motor 28BYJ-48 5V with ULN2003 Interface board 381
115. Adafruit TB6612 Stepper/motor driver 384
116. Stepper motor NEMA-17 399
117. Stepper motor from a DVD player 401
118. Floppy disc drive 404
119. Vibration motor 406
Camera triggers 409
120. Self-made trigger cable for Canon DSLR camera's 411
121. Self-made trigger cable for Sony ILCE system camera 412
122. Self-made Canon CHDK/SDM trigger cable 414
Wired communication 423
123. SoftEasyTransfer communication between 2 Arduino's 425
Wireless communication 429
124. IR sensor (receive) 431
125. IR sensor board (receive) 434
126. Keyes IR sensor board (receive) 435
127. Various IR remote controls 436
128. IR LED (send) 442

129. 315/330/433 MHz RF Receiver XY-MK-5V 444
130. 433 MHz RF Receiver KT-JSMK-7B 447
131. 433 MHz RF transceiver Aurel RTX-MID-5V 449
132. 433 MHz RF Wireless Socket Flamingo FA500S 452
133. 433 MHz RF Transmitter FS1000A 453
134. Silvercrest Wireless Socket set 455
135. 433 MHz RF Wireless Socket set Flamingo SF-500S/3 461
136. 433 MHz RF 4 channel Wireless kit XY-DJM-5V 465
137. NRF24L01 2.4GHZ Wireless Transceiver 467
138. GSM/GPRS SIM800L module 474
139. GSM/GPRS Neoway 590 DIY kit 479
140. SIMCOM SIM7020E (GSM) LTE NBIoT breakout board 483
141. LoRa 485
142. Grove Lora Radio 868 486
143. Bluetooth Keyes BT_Board v2.0 491
144. Bluetooth JY-MCU BT_Board v1.06 495
145. Bluetooth 4.0 BLE CC41A (CC2541) module 499
146. Keyes Bluetooth 4.0 BLE 504
147. Adafruit Bluefruit LE UART Friend (BLE) 509

R/C transmitters & receivers 521

148. R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx) 523
149. R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx) 526
150. R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx) 529
151. R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx) 533
152. R/C: 9ch Turnigy TGY 9x (Tx) firmware upgrade 537
153. Decoding a PPM pulse train to separate channel signals 544
154. Decoding a PPM pulse train from a trainer connector 546

Shields 551

155. Ethershield 553
156. USB Host shield 557
157. Arduino Sensor Shield v5.0 560
158. Proto type shield 561
159. Nano sensor shield 561
160. Arduino Mega Sensor Shield v 2.0 562
161. Adafruit 2.8" TFT Resistive Touch Shield v2 563
162. 2.4” TFT LCD Shield Touch Board 563
163. LoRa: Shield for HopeRF board 563
164. LoRa: Dragino-shield 563

Grove 564

165. Grove Base Shield for Arduino 566
166. Grove Mega Shield v1.2 568
167. Grove Shield for micro:bit v2.0 570
168. Grove cable 571
169. Grove to 4 pin Male cable (self-made) 573
170. Grove LED Socket Kit v1.4 (=LED) 574
171. Grove Chainable RGB LED v2.0 576
172. Grove WS2812 Waterproof LED strip 579
173. Grove Buzzer v1.2 (=piezo speaker) 581
174. Grove Speaker v1.1 583

175. Grove Button v1.1 (=switch) 585
176. Grove Tilt v1.1 587
177. Grove Touch v1.1 589
178. Grove Rotary Angle Sensor v1.2 (=potentiometer) 591
179. Grove Light Sensor v1.1 (=LDR) 593
180. Grove Light Sensor v1.2 (Linear Light Sensor) 595
181. Grove Line Finder v1.1 596
182. Grove Temperature Sensor v1.2 598
183. Grove Temperature & Humidity Sensor v1.2 600
184. Grove Sound Sensor v1.6 603
185. Grove Ultrasonic Ranger v2.0 (=distance sensor) 605
186. Grove Relay v1.2 607
187. Grove 4 Digit Display v1.0 609
188. Grove 16x2 LCD (White on Blue) 612
189. Grove LCD 16x2 RGB Backlight v4.0 614
190. Grove mini Servo 616
191. Grove Gesture v1.0 618

Isolation from higher voltages 622

192. Relay 5V board 624
193. 4x Relay 5V Board 627
194. 8x Relay 5V Board HL-58S V1.2 630
195. Optocoupler MOC3023 633
196. Optocoupler 817C 635

Logical Level shifters 637

197. 8-bit Logic Level Shifter 74LVC245 639
198. Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor 640
199. 4 channel bi-directional Logic Level Convertor 641

Power supplies 643

200. Black Wings breadboard power regulator 645
201. YuRobot breadboard power regulator 646
202. ATX Power Supply (PSU) 647
203. DC Step-Down Adjustable Power module 648
204. DC to DC Step-Up Boost Power Supply with USB 649
205. DC to DC Step-Up Boost Power Supply with USB 650

USB to Serial adapters 651

206. PL203HX USB to TTL Serial adapter 653
207. P203HX USB to TTL Serial cable 654
208. CP2102 USB to TTL Serial adapter 655
209. FTDI friend by Adafruit's (USB to TTL Serial adapter) 656
210. FTDI FT232L USB to TTL Serial adapter 658
211. Counterfeit FTDI FT232RL chips 660

Miscellaneous 669

212. Adafruit Thermal Printer CSN-A2-T 671
213. Resistor 674
214. Inverter 7404 679
215. Shift register 74HC595 680
216. Solderless breadboard 683

ESP8266 WiFi modules 685

217. Common ESP8266 687

218. Connection schemes for ESP8266 modules 689
219. Using AT commands 691
220. Using Lua scripts 694
221. ESPlorer to use AT commands or LUA scripts 696
222. ESP8266 Firmware 697
223. Programming ESP8266 with Arduino IDE 702
224. ESP8266 as WiFi to Serial 704
225. ESP-module: ESP8266 ESP-01 710
226. ESP-module: ESP8266 ESP-01 mod 713
227. Module: USB to ESP-01 714
228. Module: USB to ESP-01 flash mod v1 715
229. Module: USB to ESP-01 flash mod v2 716
230. Module: USB to ESP-01 with flash switch 717
231. Module: ESP-01 5V-3.3V adapter 718
232. Module: Open Smart ESP-01 to DIP 719
233. Module: Open Smart ESP-01 to DIP Mod 720
234. ESP-module: Adafruit HUZZAH ESP8266 breakout 721
235. ESP-module: ESP8266 ESP-07 AI-Thinker 723
236. ESP-module: ESP8266 ESP-07 with IO Adapter Plate 724
237. NodeMCU ESP-12E Doit Devkit 726
238. Sonoff T1 UK 1 gang 728
ESP32 WiFi modules 735
239. Common ESP32 736
240. TTGO ESP32 WiFi & Bluetooth Battery OLED 738
241. ESP32-CAM 740
242. Programming ESP32 with Arduino IDE 743
STM32 modules 745
243. Common STM32 747
244. Programming STM32 with Arduino IDE 748
245. Programming with System Workbench for STM32 749
246. STM32F103C8T6 Minimum Development Board (aka Blue Pill) 762
247. STM32F030F4P6 Minimum Development Board 771
248. STM32L4R9I-Discovery kit 773
249. STM32L4R9I-Discovery kit fan-out expansion board 779
Raspberry Pi 781
250. Raspberry Pi 783
251. Raspberry Pi GPIO 792
252. Sample scripts Raspberry Pi 795
253. HAT: Dragino Lora/GPS 801
254. HAT: GrovePi+ 805
255. HAT: Raspberry pi to Arduino Shields Connection Bridge v1 808
256. Waveshare 2.7 e-Paper HAT 810
257. Waveshare 7.5 e-Paper + HAT 812
258. Add-ons for Raspberry Pi 814
Mindstorms NXT 819
259. Mindstorms NXT-Brick 821
260. Programming with NXC/NBC 823
261. Mindstorms and Arduino 827
262. Mindstorms sensors with Arduino 831

Non Arduino MCU Boards **833**
263. BBC Micro:bit 835
264. Adafruit Playground Express 842
The Things Network **851**
265. Basics The Things Network 853
266. Single Channel Gateway on Raspberry Pi (J.F. Telkamp) 854
267. Single Channel Gateway on ESP8266 (Jaap Braam) 860
268. Configuration of applications and devices at TTN 864
The Things Network Nodes **869**
269. LoRa: RFM95W module 871
270. LoRa: HopeRF Adapter board for RFM95W 874
271. LoRa: Shield for HopeRF board 876
272. LoRa: Dragino-shield 882
273. LoRa: RN2483 module 886
274. LoRa: RN2483 Enschede Nano breakout board 892
275. LoRa: Nexus as a TTN node 894
276. Lora: The Things Uno as TTN node 898
277. LoRa: Sodaq One as a TTN node 901
278. LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI as a TTN Node 907
279. Grove Lora Radio 868 as a TTN Node (failure) 913
The Things Network Data Handling **917**
280. Data handling 919
281. Data handling at the TTN Console 920
282. Data handling with TTNCTL 925
283. Data handling with MQTT 927
284. Data handling with Node-red 929
Projects **937**
285. High Speed Photography 939
Links **943**
286. Web shops 945
287. Reference and tutorials 948
288. Overview Libraries ..... **Fout! Bladwijzer niet gedefinieerd.**
To Do ..... Fout! Bladwijzer niet gedefinieerd.
289. Template Xx ..... **Fout! Bladwijzer niet gedefinieerd.**

Introduction

In this document I've described the basic use of some Arduino Boards, Software, Programmer/Programming and several components to be used as input or output devices.

Most chapters consist of the following paragraphs:

- Specifications
  Technical information needed when using this component.
- Datasheet
  Links to datasheet(s) related to the component or parts of the component.
- Connections
  Names, description and ports to be used on your Arduino
- Libraries needed
  Download links to 3rd party libraries and names of the standard libraries to be used with this component
- Library use explanation
  Description how to use the library to get you started. No in-depth explanation!
- Sample
  Sample sketch and needed connections to test the component.

For every component, the description is composed of all the information you need to get things started. Use the sample sketch and your own creativity to build more complex sketches/projects.

Version history

This chapter describes changes made to this document:

Ver.	Changes	Date
1.0	First version shared	*May 8th 2014*
1.1	• **Added** this Version History. • **Added** *"33* Uploading the bootloader by using an Arduino as ISP*"*. • **Added** *"*60 "Adafruit 2.8" TFT Resistive Touch Shield v2" • **Corrected** some small typo's.	*Sep 14th 2015*
1.11	• **Added** 61 "2.4" TFT LCD Shield Touch Board " • **Changed** the orientation of the "60 "Adafruit 2.8" TFT Resistive Touch Shield v2**"**, thus simplifying the use of the library and the sample sketch. • **Corrected** some small typos. • **Added** links to datasheets for several components. • **Added** specifications for several components. • **Correction** made to both Bluetooth modules. A voltage divider has been added to the RxD port of the Bluetooth module, creating an input voltage of 3.3V instead of 5V. • **Added** an extra Sample to the Keyes Bluetooth module, Using D10 and D11 through SoftwareSerial.	*Sep 22nd 2015*
1.12	• **Correction** made to "21.4 First time preparation of Arduino IDE for Attiny45/85" for Arduino IDE 1.6.5. • **Added** "22 Attiny85 Digispark board". • **Added** a sample of PWM output (LED fade sample) in: "41.4 Sample LED". • **Added "**54 WS2812B RGB LED breakout-board**"** • **Added** "73 Nunchuk with connection board on A2..A5" • **Added** a Processing sample to Nunchuk in "73.3 Sample Nunchuk with connection board on A2..A5". • **Added** "128 SoftEasyTransfer" • **Added** "132.6 IR Remote Control 4**"** • **Added** "142 NRF24L01 2.4GHZ Wireless Transceiver**"** • **Moved** Sections Wired and Wireless communication down, just before section Shields • **Moved** Section Isolation from higher voltages down, just before section Shields. • **Added** Section ESP8266 Wi-Fi. • **Added** Chapters for using the ESP8266. • **Added** Section "Raspberry Pi. • **Added** Section Mindstorms NXT in To-do. • **Corrected** a truckload of small typo's	*Nov 1st 2015*

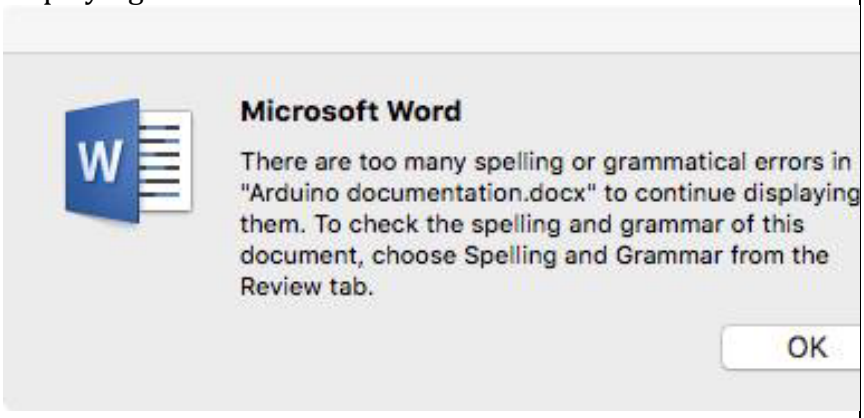
Ver.	Changes	Date
1.13	• **Finished** Mindstorms NXT connection to Arduino (description of sample program and special connector). • **Added** description how to program the Boarduino. • **Changed** errors in pin diagram and sample connections for "49 4 Digits 7-Segment Display Common Cathode" • **Changed** errors in sample connections for "142 NRF24L01 2.4GHZ Wireless Transceiver" • **Replaced** RCSwitch library in "134 315/330/433 MHz RF Receiver XY-MK-5V" with Virtual Wire, so it is possible to Transmit text strings instead of codes. • **Updated** libraries and made changes to samples where needed. Most libraries were installed through Library Manager, some were replaced, some were updated to newer version. • **Added** a section about Arduino IDE • **Added** a chapter about Libraries • **Added** a chapter about board management • **Added** "**Fout! Verwijzingsbron niet gevonden. Fout! Verwijzingsbron niet gevonden.**", with a table showing all libraries used in this document. This way it is easier to maintain accessibility and to prevent 404-errors when following the links. FAILED • **Added** a chapter about the Serial Monitor. Sample sketches for both Output (from Arduino to computer) and Input (from computer to Arduino). • **Added** "242 ESP-module: ESP8266 ESP-07 with IO Adapter Plate". • **Added** "243 NodeMCU ESP-12E Doit Devkit". • **Added** description of the tool esptool-ck ESP8266 section. • **Documented** a problem with the USB-Serial cable with the PL203HX chip in combination with flashing firmware and uploading sketches to the ESP8266 modules. • **Added** remarks about maximum current per port Arduino.	*Published at Nov. 29th 2015*
1.14	• **Add**ed "213 P203HX USB to TTL Serial cable". • **Added** "214 CP2102 USB to TTL Serial adapter". • **Added** "216 FTDI FT232L USB to TTL Serial adapter" • **Added** "217 Counterfeit FTDI FT232RL chips", describing the problem about bricking it and a solution to de-brick it. • **Added** "55 WS2812B RGB LED ".	*Dec. 30th 2015*

Ver.	Changes	Date
1.15	• **Re-added** "**Fout! Verwijzingsbron niet gevonden. Fout! Verwijzingsbron niet gevonden.**", somehow this chapter got missing in version 1.14. • **Added** pin diagram in Raspberry Pi Section • **Added** missing photo's • **Added** "278 LoRa: HopeRF Adapter board for RFM95W" • **Added** "280 LoRa: Dragino-shield" • **Added "**279 LoRa: Shield for HopeRF board" • **Added** "261 HAT: Dragino Lora/GPS" • **Added "**14 LoRa: Nexus**"** • **Added** "13 Teensy 3.1" • **Added** "58 16x2 Display 1602A" • **Added "**170 Grove Base Shield for Arduino" • **Added** "262 HAT: GrovePi+" • **Added** "174 Grove cable" • **Added** "175 Grove to 4 pin Male cable (self-made)" • **Added** "176 Grove LED Socket Kit v1.4 (=LED)" • **Added** "179 Grove Buzzer v1.2 (=piezo speaker)" • **Added** "181 Grove Button v1.1 (=switch)" • **Added** "183 Grove Touch" • **Added** "184 Grove Rotary Angle Sensor v1.2 (=potentiometer)" • **Added** "185 Grove Light Sensor v1.1 (=LDR)" • **Added** "188 Grove Temperature Sensor" • **Added** "189 Grove Temperature & Humidity Sensor" • **Added** "190 Grove Sound Sensor" • **Added** "191 Grove Ultrasonic Ranger v2.0 (=distance sensor)" • **Added** "192 Grove Relay v1.2" • **Added** "195 Grove LCD 16x2 RGB Backlight" • **Added** "196 Grove mini Servo" • **Added** "197 Grove Gesture v1.0" • **Added** "161 USB Host shield" • **Added** "76 PlayStation controller" • **Added** "152 Adafruit Bluefruit LE UART Friend (BLE)" • **Added** A section for non-Arduino MCU boards. • **Added** "271 BBC Micro:bit" • **Added** a section about the Internet Of Things (IOT) for The Things Network, with instructions on how to configure an application at the TTN, build your own Single Channel Gateway and building a node. • **Changed** page layout so it can be printed double sided. Section starting at odd pages and a blank page break after the start of every section.	*Nov. 12th, 2016*

Ver.	Changes	Date

1.17	• **Added** 6DOF MPU-6050 3 Axis Gyro With Accelerometer Sensor Module
	• **Added** separate sketches for working with switches:
	  ○ **External pulldown resistor**
	  ○ **External pullup resistor**
	  ○ **Intern pullup resistor**
	• **Added** Logic Level Convertor bidirectional
	• **Added** Adafruit SPI/I2C save 5-3.3V level shifter
	• **Added** 74LVC245 5-3.3 level shifter
	• **Added** "6 Arduino Mega 2560 R3"
	• **Added** "16 LoRa: Sodaq One" in the Arduino Boards section
	• **Added** "90 PIR/Motion Sensor HC-SR501"
	• **Added** "99 GPS XM37-1612 (GY-NEO6Mv2?)"
	• **Added** "102 MQ-3 alcohol gas sensor board"
	• **Added** "103 MQ-6 LPG, iso-butane and propane gas sensor"
	• **Changed** stepper motor wire colors used in sample at "116 Adafruit TB6612 Stepper/motor driver"
	• **Added** "116 Adafruit TB6612 Stepper/motor driver"
	• **Added "**120 Stepper motor NEMA-17"
	• **Added** "150 Bluetooth 4.0 BLE CC41A (CC2541) module"
	• **Added** "151 Keyes Bluetooth 4.0 BLE"
	• **Added** "143 GSM/GPRS SIM800L module"
	• **Adding** "231 ESP-module: ESP8266 ESP-01"
	• **Added** "275 Single Channel Gateway on ESP8266"
	• **Added** "285 LoRa: Sodaq One as a TTN node" in The Things Network Nodes section.
	• **Rearranged** IOT section in three new sections
	  ○ The Things Network
	  ○ The Things Network Nodes
	  ○ The Things Network Data Handling
		Nov 1st 2017

Ver.	Changes	Date
1.18	• **Added** "10 Sparkfun Pro Micro - 5V/16MHz" • **Added** "11 Arduino Pro Mini ATmega168U 5v, 16MHz" • **Added** "12 Arduino Pro Mini ATmega328p 3.3V, 8MHz" • **Added** "78 RGB and Gesture sensor APDS-9960 " • **Added** "165 Arduino Mega Sensor Shield v 2.0" • **Added "**207 YuRobot breadboard power regulator**"** • **Added** "233 Module: USB to ESP-01" • **Added** "234 Module: USB to ESP-01 flash mod" • **Added** "235 Module: USB to ESP-01 flash mod v2" • **Added** "236 Module: USB to ESP-01 with flash switch" • **Added "**237 Module: ESP-01 5V-3.3V adapter**"** • **Added** "238 Module: Open Smart ESP-01 to DIP**"** • **Added "**239 Module: Open Smart ESP-01 to DIP Mod**"** • **Replaced** Photo from shield with Hope RF board, because orientation was not correct: "279 LoRa: Shield for HopeRF board" • **Corrected** Breakout pin-numbers in "282 LoRa: RN2483 Enschede Nano breakout board" • **Changed** order of the Section about ESP8266. Moved background chapters in front of description of ESP Modules.	*Nov 22nd 2017*
1.19	• **Added** "104 Load cells" • **Added** "105 Load cell amplifier HX711" • **Added** "139 Silvercrest Wireless Socket set" a 433 MHz set from 'Lidl' • **Added "**140 433 MHz RF Wireless Socket set Flamingo SF-500S/3" a 433 MHz Click-On/Click-Off set from 'Action' • **Added** "244 Sonoff T1 UK 1 gang" • **Changed** "271.3 How to program the " • **Added** "272 Adafruit Playground Express" • **Added** "263 HAT: Raspberry pi to Arduino Shields Connection Bridge v1" • **Added** "218 Adafruit Thermal Printer CSN-A2-T" • **Added** section for ESP32 modules	*Jun 2018*
1.20	• **Adding** section for STM32 modules • **Added** chapter for "254 STM32L4R9I-Discovery kit" • **Added** chapter for "252 STM32F103C8T6 Minimum Development Board (aka Blue Pill)" • **Added** chapter for "253 STM32F030F4P6 Minimum Development Board" • **Added** chapter for "36 ST Link V2" • **Added** chapter for "62 0.96 inch 128x64 OLED display (I2C)" • **Added** chapter for "63 0.91 inch 128x32 OLED display (I2C)" • **Added** chapter for "145 SIMCOM SIM7020E (GSM) LTE NBIoT breakout board" • **Added** several missing Photo's	*Nov 21st 2018*

Ver.	Changes	Date
1.21	• **Added** a section for R/C transmitters and receivers. • **Added** "153 R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)". • **Added** "154 R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)" only the 2 sticks were covered (CH 1..4). • **Added** "155 R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)" only the 2 sticks were covered (CH 1..4). • **Added** "156 R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)" only the 2 sticks were covered (CH 1..4). • **Added** "158 Decoding a PPM pulse train to separate channel signals". • **Added** "159 Decoding a PPM pulse train from a trainer connector". • **Added** "87 Capacitive Soil Moisture sensor v1.2". • **Added "**88 Touch Sensor TTP223B". • **Added "**39 Mini MP3 Player MP3-TF-16P". • **Replaced** libraries and altered explanation and sample sketches for a few modules (DS1302 RTC for example) • **Saved** all sample sketches to the shared folder: http://bit.ly/eve_arduinosketches. I've just compiled them before saving (only a few off the sketches still have errors), but I've not yet tested the working yet. The names of these sketches are displayed above the sample sketch in the text. • **Added** "43 10 segment LED Bar Graph F2510BH". • **Reworked** the 8x8 DOT Matrix chapter (lots of mistakes). • **Added** "46 8x8 DOT Matrix HL-M1588BR". • **Added** "53 MAX7219 LED driver" with samples for an LED Array/Bar Graph, multiple Seven Segment displays and a 8x8 DOT Matrix. • **Performed** an extensive spelling check! After reaching 100.000 words and 1400 spelling-errors, Word stopped displaying them! 	*Dec 23rd 2018*

1.22	• **Added** "3 Yourduino RoboRed" • **Added** "4 Seeeduino V4.2" • **Added** "5 Seeeduino Lotus v1.1" • **Added** "8 Arduino Mega 2560 Pro Mini" • **Changed** link to json file for the "16 LoRa: Sodaq One" in "16.5 First time preparation of Arduino IDE for Sodaq One" • **Added** "51 4 Digits 7-Segment Display Common Anode" • **Added** "56 WS2812B RGB LED ring" • **Added** "64 Waveshare 2.7 e-Paper HAT with an Arduino" • **Corrected** the compilation errors on the Nunchuk working directly on A2..A5 "73.3 Sample Nunchuk with connection board on A2..A5" • **Added** "75 Wii u.Draw Game Tablet" • **Changed** library and sample sketch for "80 Temperature and Humidity sensor board". • **Added** "93 Laser Sensor module", although DS18B20 (temperature sensor) is printed on it, this is incorrect. It is often sold with a Laser sensor attached to it instead of a temperature sensor. • **Added** "100 NEO6 GPS moduel GPS GY-NEO6MV2" • **Added** "101 GPS GY-GPS6MV2" • **Added** "109 Adafruit PN532 NFC/RFID controller breakout board" • **Added** "113 Tinkerkit Braccio Robot" • **Added "**135 433 MHz RF Receiver KT-JSMK-7B" • **Added "**136 433 MHz RF transceiver Aurel RTX-MID-5V" • **Added "**137 433 MHz RF Wireless Socket Flamingo FA500S" • **Added** Sample sketch for "143 GSM/GPRS SIM800L module" • **Added** "144 GSM/GPRS Neoway 590 DIY kit" • **Added** "171 Grove Mega Shield v1.2" • **Added** "173 Grove Shield for micro:bit v2.0" • **Added** "177 Grove Chainable RGB LED v2.0" • **Added** "178 Grove WS2812 Waterproof LED strip" • **Added** "180 Grove Speaker v1.1" • **Added** "186 Grove Light Sensor v1.2 (Linear Light Sensor)" • **Added** "187 Grove Line Finder v1.1" • **Added** "193 Grove 4 Digit Display v1.0" • **Added** "194 Grove 16x2 LCD (White on Blue)" • **Added** "200 8x Relay 5V Board HL-58S V1.2" • **Added** "258.2 Raspberry Pi Zero W" • **Added** "264 Waveshare 2.7 e-Paper HAT" • **Added** "265 Waveshare 7.5 e-Paper + HAT" • **Added** "266.3 Raspberry Pi Camera v2.1" • **Added** "266.1 7" Touch Display v1.1" • **Added** "271.4 Edge Connector Breakout board for micro:bit" • **Reworked** all the Bluetooth chapters	*Oct. 13 2019*

1.23	• **Ruined the Table of contents, new chapters can't be added any more. It is only possible to update the pagenumbers of chapters that already were in the Table of Contents. (28 MB is probably way toolarge for Word?)** • **Tested/corrected** all of my sample sketches in : http://bit.ly/eve_arduinosketches and listed the results in the file _List of sketches.xlsx. • **Added** several missing photo's • **Added** "15 The Things Uno" • **Added** "81 DS18B20 temperature sensor" • **Added** "82 DS18B20 temperature sensor" • **Added** "83 Barometer BMP180" • **Added** "84 Barometer BMP280-5V" • **Added** "85 Barometer BME280-5V" • **Added** "92 Pindiode - Photodiode BPW34" • **Added** "107 Micro-SD (Secure Digital)" • **Changed** sketch in "110.5 Sample RTC module with DS1302 chip" • **Added** sample sketches for both motor and NEMA17 stepper motor "116 Adafruit TB6612 Stepper/motor driver" and "114 DC/Stepper Motor Driver board L298n" • **Added** "117 Pololu A4988 Stepper driver" • **Added** "118 Yongfukang HR4988 Stepper driver" • **Added** "119 CNC Shield v3.0" • **Added** a sample sketch for: "120 Stepper motor NEMA-17" • **Added** "121 Stepper motor from a DVD player" • **Added** "122 Stepper motor from a Floppy disc drive" • **Addes** "124 Vibration motor" • **Changed** "127 Self-made Canon CHDK/SDM trigger cable**"** and the corresponding CHDK Lua scripts, since some functions were deprecated/replaced. • **Rechecked** the sample sketches for "128 SoftEasyTransfer communication between 2 Arduino's", they should be OK now. • **Added** "147 Grove Lora Radio 868" • **Added** "172 Grove Shield NodeMCU" • **Added** "202 Optocoupler 817C" • **Added** "247 ESP32-CAM" • **Added** "266.4 Raspberry Pi Camera NoIR v2.1" • **Changed** the Arduino sketch that connects to the Lego NXT Mindstorms "269.2 Sample Mindstorms and Arduino" • **Added** "284 Lora: The Things Uno as TTN node" • **Added** "286 LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI" • **Added** "287 Grove Lora Radio 868 as a TTN Node (failure)" • **Changed** the link to an I2C scanner on several places. • **Removed** Chapter about Libraries. I haven't updated it for a long while and I'm not plannig to do so in the future.	*Apr. 27 2020*

1.24 draft.	• **Add** Optocouplers: 4N25
	• **Add** LED ring
	• **Add** "Waveshare 7.5 e-Paper + HAT on Arduino"
	• **Add** voltage divider 5V => 3.3V
	• **Add** RFlink selfmade for Domoticz
	• **Check** working of all sample sketches
	• **Add** extra switches to Robbe Futaba F-14 Tx
	• **Add** the extra channels on the R/C transmitters F14, Dx6i
	• **Add projects** Simon Says, Holidays Time Switch, Fake TV, Word Clock, Light globe, Light Sword, POV stick
	• **Add** Lora The Things Node
	• **Add** LoRa The Things Gateway
	• **Add missing photo's**
	  ○ Diskette stepper
	  ○ LED strip
	  ○ RPI Camera NoIR
	  ○ Optical encoders
	  ○ Micro-SD reader/writer
	  ○ Female Dupont <=> Micro USB cable (LoRa GPS tracker)
	• **Add** a sample sketch to self-made Canon and Sony trigger cables chapters
	• **Add** Flat piezo as speaker or as sound detector
	• **Add** chapter about Serial.print/Serial.read etc…
	• **Add** sample sketch for "145 SIMCOM SIM7020E (GSM) LTE NBIoT breakout board"
	• **Add** photo, library description, datasheet, specs for IR send
	• **Add** HEF4050BP Hex non-inverting buffer as a logic level converter from up to 15 V to standard TTL levels
	• **Add** chapter for using MOSFET, transistors & Darlington to drive larger currents/higher voltages
	• **Add** LED strip with MOSFET/Transistor
	• **Add** chapter for capacitors.
		Some future release

Arduino Boards

This section describes the Arduino boards I've been using, like the Arduino UNO, the NANO. I even added some boards that aren't real Arduino, like the Attiny45/85. All these boards can be programmed with the Arduino IDE. For each board you will find specifications, connections and protocols.

1. Arduino

"*ARDUINO IS AN OPEN-SOURCE ELECTRONICS PROTOTYPING PLATFORM BASED ON FLEXIBLE, EASY-TO-USE HARDWARE AND SOFTWARE. IT'S INTENDED FOR ARTISTS, DESIGNERS, HOBBYISTS AND ANYONE INTERESTED IN CREATING INTERACTIVE OBJECTS OR ENVIRONMENTS*[1]"

Since Arduino is Open Source, the CAD and PCB design is freely available. Everyone can buy a pre-assembled original Arduino board[2] or a cloned board from another company. You can also build an Arduino for yourself or for selling. Although it is allowed to build and sell cloned Arduino boards, it's not allowed to use the name Arduino and the corresponding logo. Most boards are designed around the Atmel Atmega328.

1.1. Popular Arduino boards

There are several different Arduino boards on the market (both original and cloned).

- Arduino UNO
  - Most popular board. Ideal for starters.
  - Standard USB for data and power and programming.
  - Power Input connector.
  - female headers.
  - 14 digital I/O ports (of which 6 PWM).
  - 6 analog input ports.
  - 1 hardware serial port (UART).
- Arduino Nano
  - Much smaller than the UNO (only 18x43 mm).
  - Mini USB for data and power and programming.
  - Input 6-20 V on Vin (6-12 recommended).
  - Male headers at the bottom side, so ideal to use on a solder less breadboard.
  - 14 digital I/O ports (of which 6 PWM).
  - 8 analog input ports.
  - 1 hardware serial port (UART).
- Arduino Mini
  - Smallest Arduino board used in small sized projects.
  - No USB connector, you need a USB to TTL Serial convertor to program it.
  - No separate power connector (you must use +9V header pins)
  - Input 7-9 V.
  - Male headers at the bottom side, so ideal to use on a solder less breadboard.
  - 14 digital I/O ports (of which 6 PWM).
  - 8 analog input ports (4 of them are not connected to header pins).
  - No hardware serial port (UART).

- Arduino Mega
  - Largest Arduino board for large number of I/O ports.

[1] Quote from the arduino.cc website.

[2] Most original Arduino boards are made by SmartProjects in Italy. The Arduino Pro, LilyPad and Pro Mini are made and designed by Sparkfun in the US. The original Arduino Nano is built by Gravitech.

- Normal size USB for data and power and programming.
- Power input connector.
- Input 6-20 V (7-12 recommended).
- Female headers at the top side.
- 54 digital I/O ports (of which 15 PWM).
- 16 analog input ports.
- 4 serial ports.

1.2. Connections

The following connections are available on most Arduino boards. Differences can be found in the number of Digital I/O and Analog Inputs.

Common connections

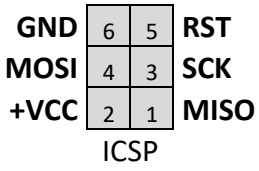
Name	Description
GND	Ground
5V	Regulated 5V output Regulated 5V input (not recommended)
3.3V	Regulated 3.3V output (from FTDI)
VIN	Non-regulated input (6-12 V)
RESET	
IOREF	
AREF	
Tx	Digital Input/Output. Two values: LOW, HIGH
Dx~	Digital Input/Output PWM. Values: 0..255 through PWM (Pulse Width Modulation)
Ax	Analog Input. Values: 0..1023

Shared connections

Name	Shared with	Description
Rx	D0	TTL Serial Receive
Tx	D1	TTL Serial Transmit
SCK	D13	SPI Serial Clock
MISO	D12	SPI Master In Slave Out
MOSI	D11	SPI Master Out Slave In
SS	D10	SPI Slave Select
SDA	UNO/NANO: A4 MEGA: D20	I^2C / TWI Data
SCL	UNO/NANO: A5 MEGA: D21	I^2C / TWI Clock

ICSP header

All SPI headers are also available on a separate double row of header pins.

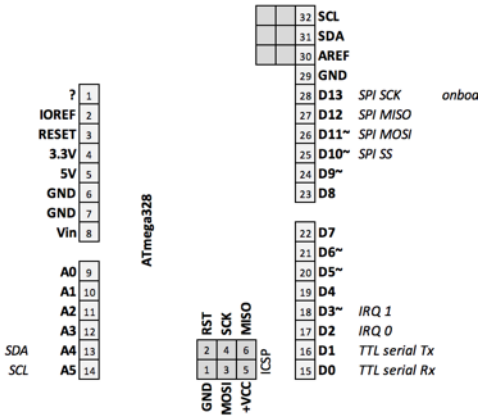


2. Arduino UNO R3

2.1. Specifications

Microcontroller	Atmega328
Input Voltage	7-12 V recommended, 6-20 V limits
Operating Voltage	5V
Current at 3.3V pin	50mA
Current at 5V pins	500mA
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL Serial converter	Atmega16U2
UART	1
3V	Available
I2C	1

2.2. Layout/connections Arduino UNO R3



The following remarks were made by user '*68ths*' on the Arduino Forum:

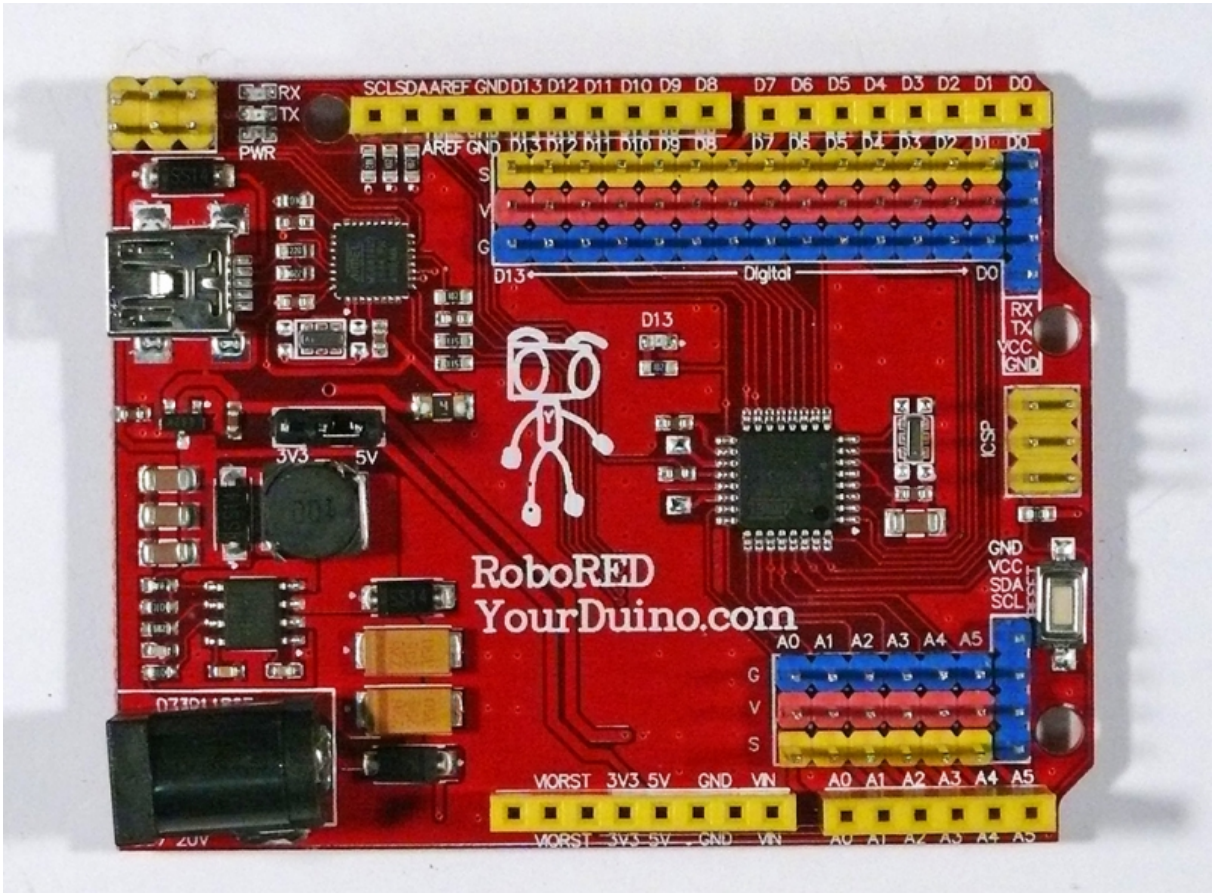
> One regret: you indicate Imax = 40 mA per pin
>
> This is what is indicated on the Arduino website, but it is absolutely not in line with Atmel's datasheet .
> Datasheet :
> 40 mA is "Absolute Maximum Rating".
>
> Quote
>
> *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*
>
> The value indicated by Atmel in continuous operation only 20 mA per pin.
> And simultaneously:
> - max 200 mA on VCC --> max current in the bondings
> - max 200 mA on GND --> max current in the bondings
> - max 150 mA per port
> So it is unthinkable to use more than three or four outputs simultaneously at 20 mA.
>
> It is not finished :
> With a current of 20 mA and @ 25 ° C, high level is no longer 5V but 4.5V (ohm law apply to Rdon) and the low level is not 0V but + 0,5V.
> These results are degraded when the chip temperature increases.
>
> I think it is necessary to advise beginners on this subject.

2.3. Programming a sketch on the Arduino UNO

Follow the following steps to program an Arduino UNO:

- Connect your Arduino UNO to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO UNO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

3. Yourduino RoboRed

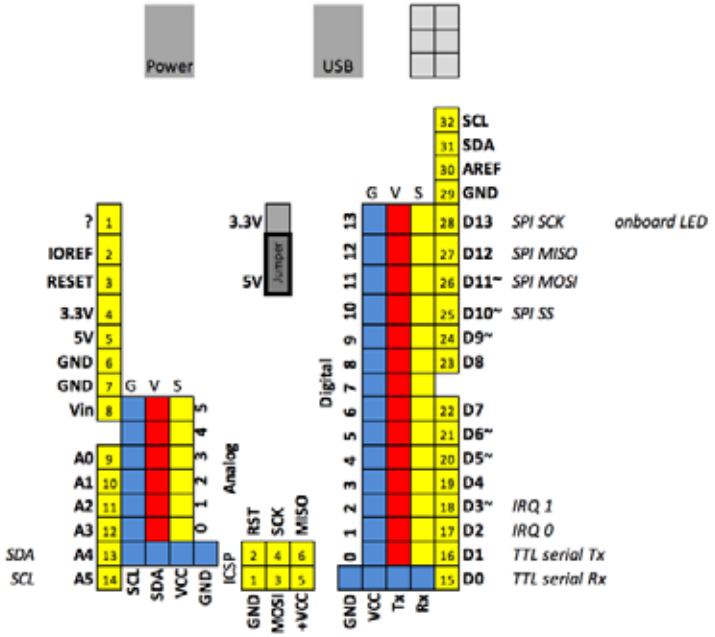


The Yourduino is a custom made Arduino UNO with some nice extra specs. It is designed by the owners of yourduino.com. The RoboRed can be switched between 3.3V and 5V, so very interesting when you plan to switch between 3.3V sensors and 5V sensors. Another interesting feature is the current it can deliver on the 3.3V and 5V pins. Beside this, it is totally compatible with the original Arduino UNO R3.

3.1. Specifications

Microcontroller	Atmega328
Input Voltage	7-20 V
Operating Voltage	3.3 or 5V (by switch or jumper)
Current at 3.3V pin	500mA
Current at 5V pins	2000mA!!
Digital I/O pins	14 (of which 6 PWM) all with 3 pin connector
Analog input pins	6 all with 3 pin connector
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL Serial converter	Atmega16U2
UART	1 with 4 pin connector (Rx, Tx, Vcc, GND)
3V	Available
I2C	1 with 4 pin connector (SCL, SDA, VCC, GND)

3.2. Layout/connections Yourduino RoboRed

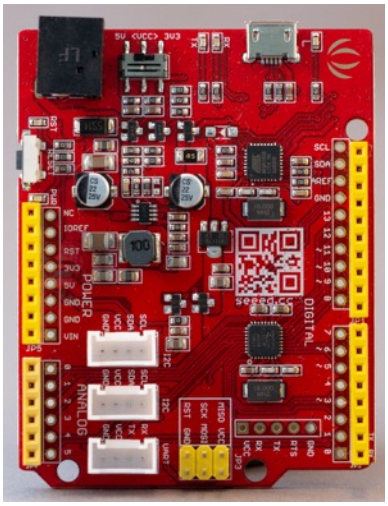


3.3. Programming a sketch on the Yourduino RoboRed

Follow the following steps to program an Arduino UNO:

- Connect your Arduino UNO to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO UNO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

4. Seeeduino V4.2




This Arduino clone is made by Seeed Studio famous for its large collection of sensors with Grove connectors.

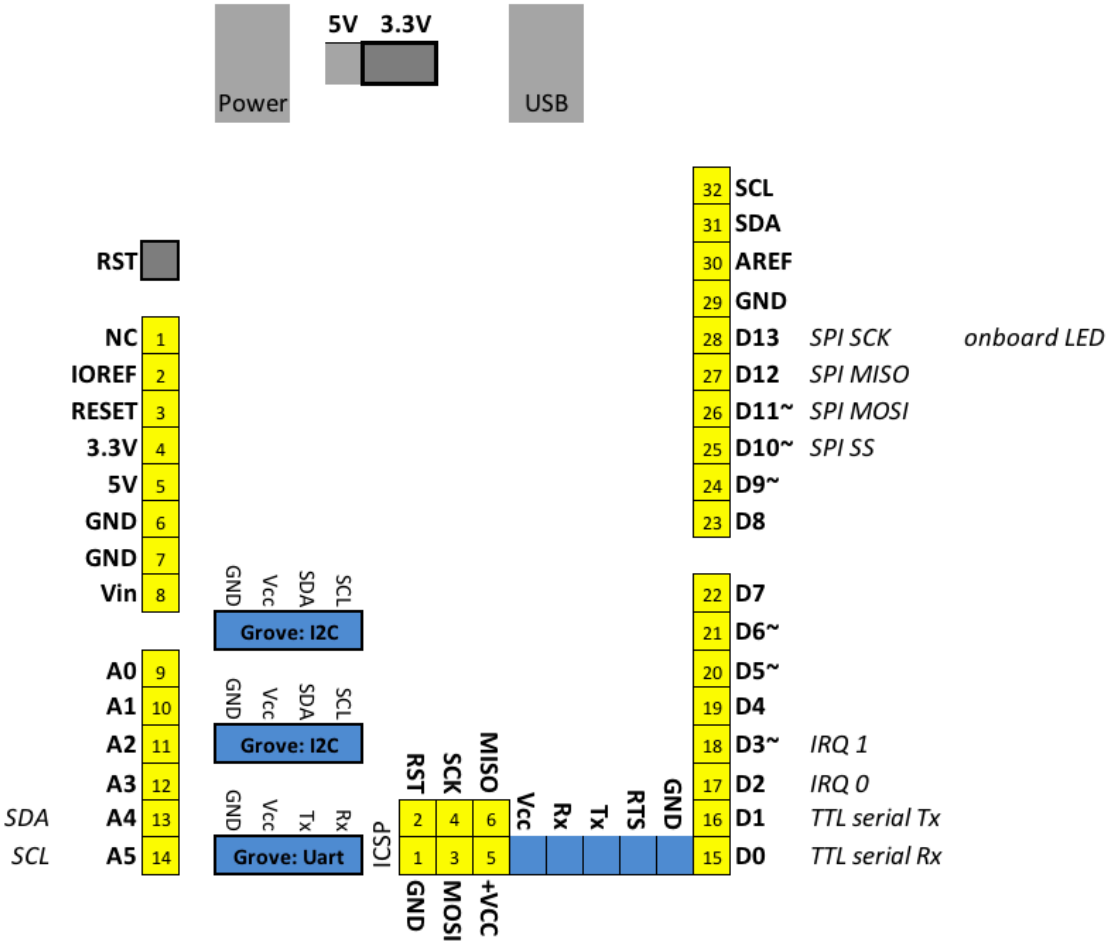
4.1. Specifications Seeeduino v4.2

Microcontroller	Atmega328
Bootloader	Arduino Uno
Input Voltage	7-15 V
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
Operating Voltage	3.3 or 5V (by switch)
USB to TTL Serial converter	Atmega16U2 on D0 & D1 and also on a Grove connector
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
I2C	On A5 & A4 and also on 2 Grove connectors
Shield	• Normal UNO shield • Next to all pins there are through-hole pads in a 0.1"grid. This makes it easy to solder additional pin headers to connect to 0.1"grid breadboards or PCB's
USB port	micro-USB
Current at 3.3V pin	500mA
Current at 5V pins	500mA (2000mA when powered by DC jack)

4.2. Documentation Seeeduino v4.2

http://wiki.seeedstudio.com/Seeeduino_v4.2/

4.3. Layout/connections Seeeduino v4.2

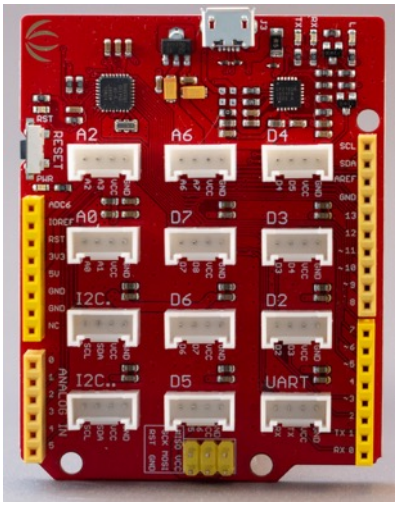


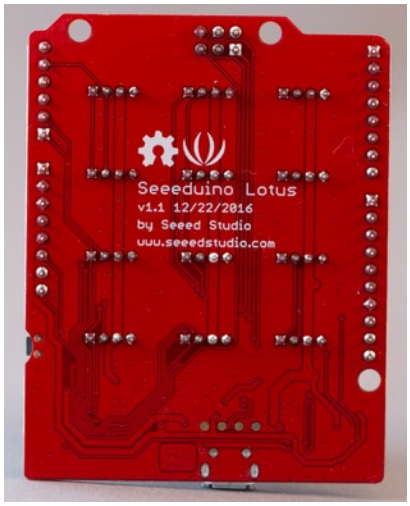
4.4. Programming a sketch on the Seeeduino v4.2

Follow the following steps to program the Seeeduino v4.2:

- Connect your Seeeduino v4.2to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO UNO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

5. Seeeduino Lotus v1.1





This Arduino clone is made by Seeed Studio famous for its large collection of sensors with Grove connectors. This board is fully equipt with those Grove connectors.

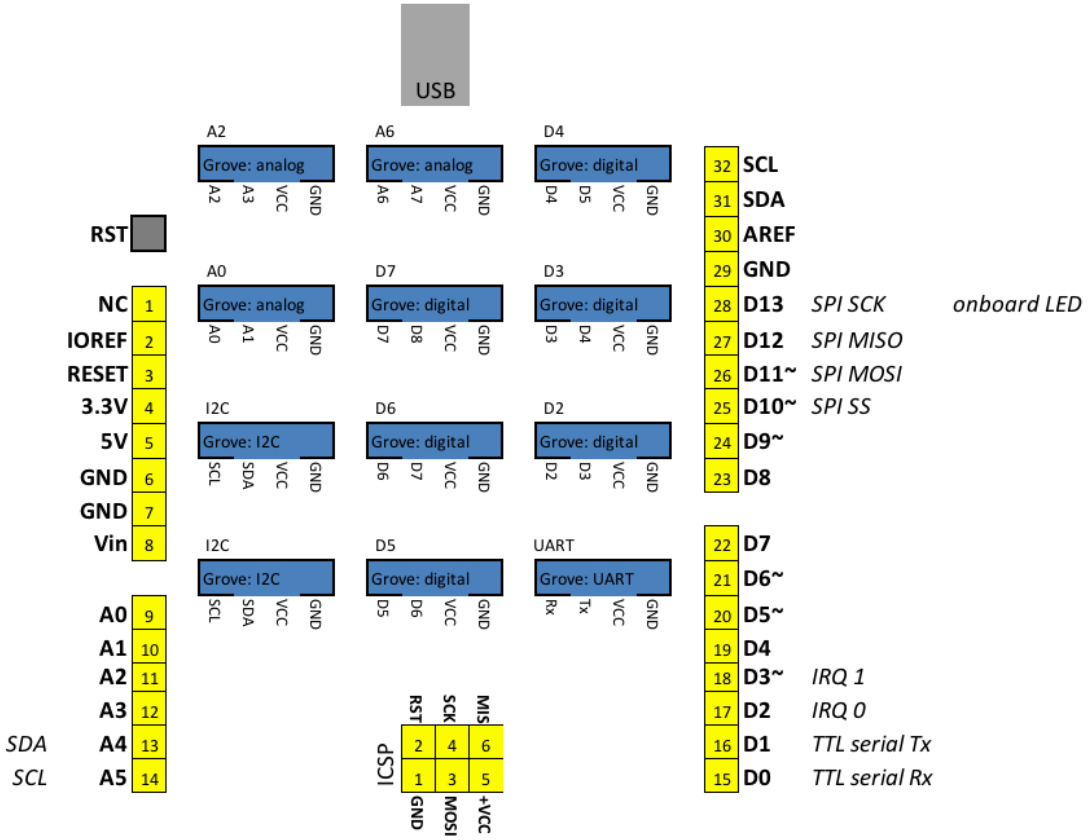
5.1. Specifications Seeeduino Lotus v1.1

Microcontroller	Atmega328
Bootloader	Arduino Uno
Input Voltage	5V only on USB
Digital I/O pins	14 (of which 6 PWM), 6 of them also on Grove connectors
Analog input pins	7 all of them also on Grove connectors (except A4 & A5)
Operating Voltage	5V (by switch)
USB to TTL Serial converter	CP2102N on D0 & D1 and also on a Grove connector
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
I2C	On A5 & A4 and also on 2 Grove connectors
USB port	micro-USB
DC current per I/O pin	40 mA

5.2. Documentation Seeeduino Lotus v1.1

http://wiki.seeedstudio.com/Seeeduino_Lotus/

5.3. Layout/connections Seeeduino Lotus v1.1

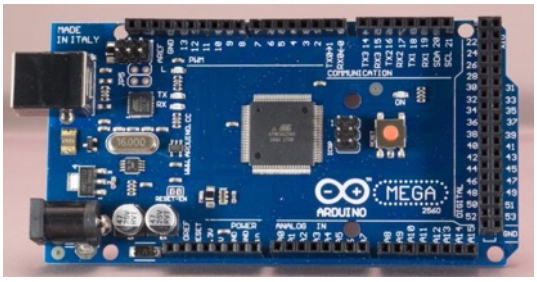


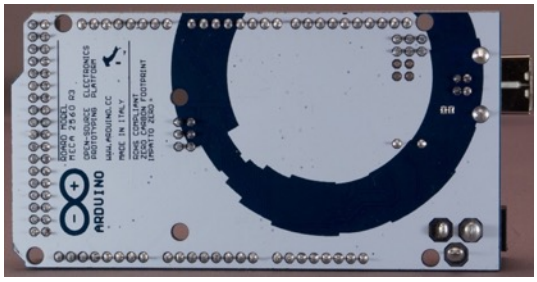
5.4. Programming a sketch on the Seeeduino Lotus v1.1

Follow the following steps to program the Seeeduino Lotus v1.1:

- Connect your Seeeduino Lotus v1.1 to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO UNO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

6. Arduino Mega 2560 R3

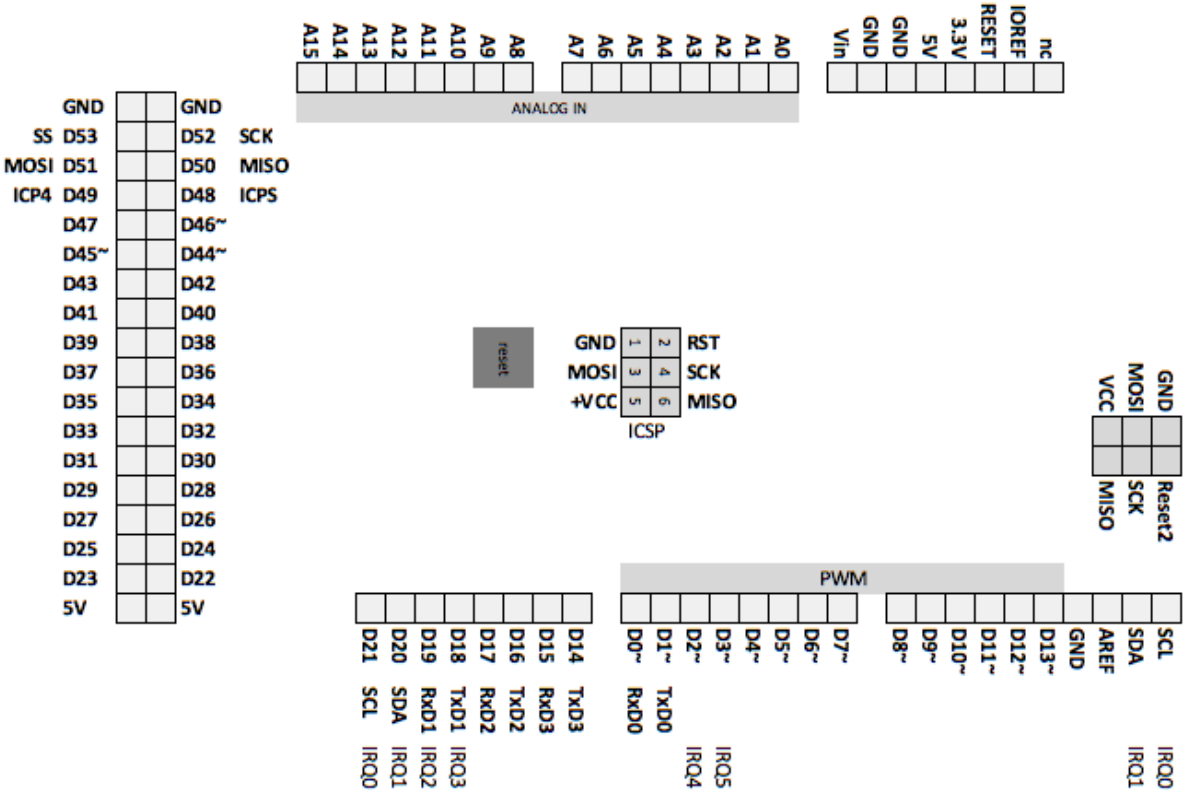




6.1. Specifications Arduino Mega 2560 R3

Microcontroller	Atmega2560
Clock Speed	16 MHz
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	54 (of which 15 PWM)
Analog input pins	16
DC current per I/O pin	20 mA
DC current for 3.3V pin	50 mA
Flash memory	256 KB
SRAM	8 KB
EEPROM	4 KB
USB to TTL Serial converter	Atmega16U2
UART	4
3V	Available
External Interrupts	5

6.2. Layout/connections Arduino Mega 2560R3

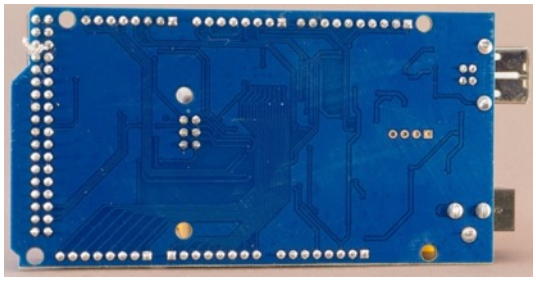


6.3. Programming a sketch on the Arduino Mega 2560 R3

Follow the following steps to program an Arduino Mega:

- Connect your Arduino Mega to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO Mega.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

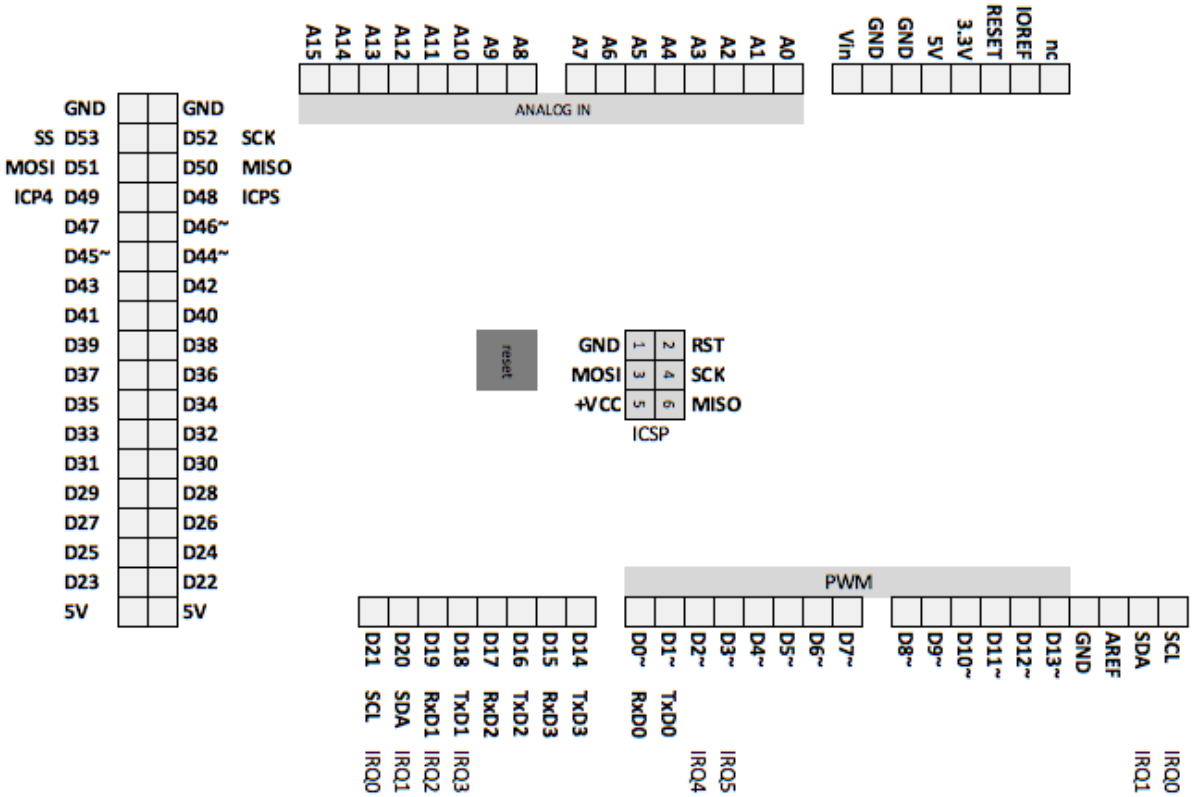
7. Arduino Mega 2560 R3 CH340



7.1. Specifications Arduino Mega 2560 R3 CH340

Microcontroller	Atmega2560
Clock Speed	16 MHz
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	54 (of which 15 PWM)
Analog input pins	16
DC current per I/O pin	20 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	8 KB
EEPROM	4 KB
USB to TTL Serial converter	CH340
UART	4
3V	Available
External Interrupts	5

7.2. Layout/connections Arduino Mega 2560R3 CH340

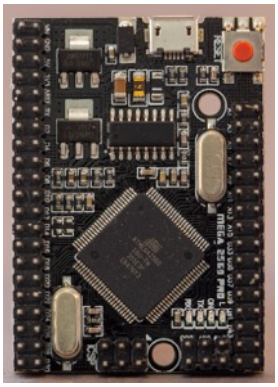
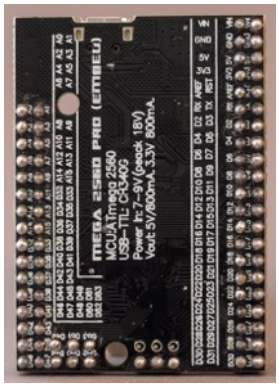


7.3. Programming a sketch on the Arduino Mega 2560 R3 CH340

Follow the following steps to program an Arduino Mega:

- Connect your Arduino Mega to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO Mega.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

8. Arduino Mega 2560 Pro Mini

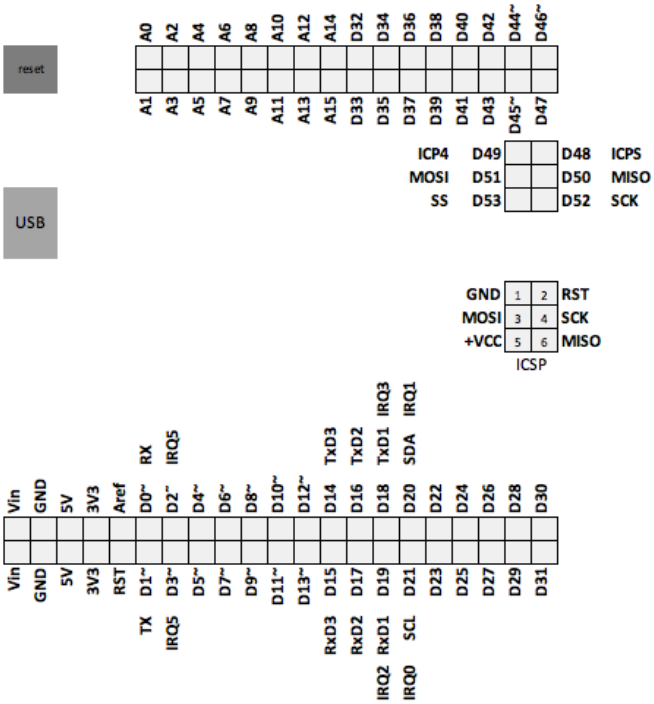



This is the smaller version of the Arduino Mega 2560 R3, same hardware and specs, but much smaller .

8.1. Specifications Arduino Mega 2560 Pro Mini

Microcontroller	Atmega2560
Clock Speed	16 MHz
Operating Voltage	7-9 V (peak 18V)
Vout: 5V	800 mA
Vout: 3.3V	800 mA
Digital I/O pins	54 (of which 15 PWM)
Analog input pins	16
DC current per I/O pin	20 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	8 KB
EEPROM	4 KB
USB to TTL Serial converter	CH340
UART	4
3V	Available
External Interrupts	5

8.2. Layout/connections Arduino Mega 2560 Pro Mini

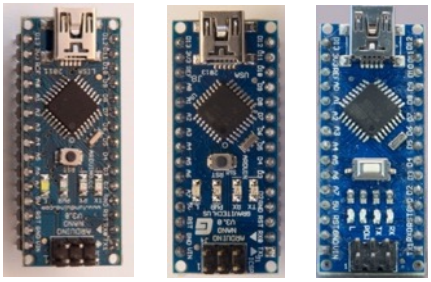


8.3. Programming a sketch on the Arduino Mega 2560 Pro Mini

Follow the following steps to program an Arduino Mega:

- Connect your Arduino Mega to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO Mega.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

9. Arduino Nano

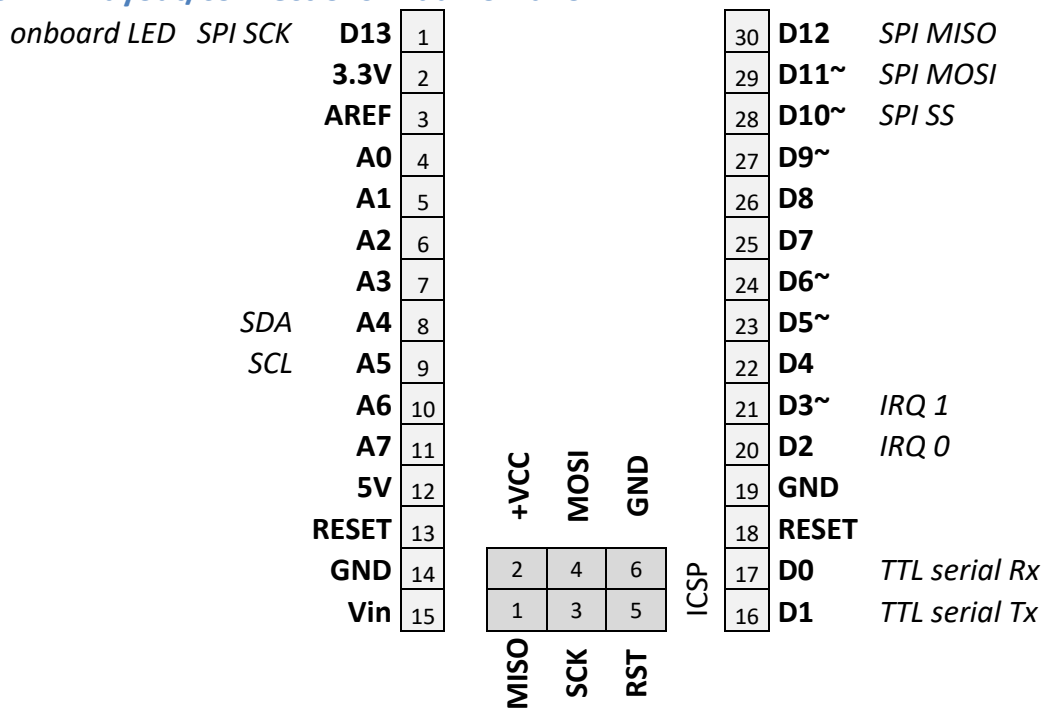


9.1. Specifications Arduino Nano

Be careful not to buy an adapter with a counterfeit FTDI chip[1]

Microcontroller	Atmega328
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	8
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL serial convertor	FTDI FT232RL
UART	1
3.3V	***Only available through FTDI, so only when powered through USB!***

9.2. Layout/connections Arduino Nano



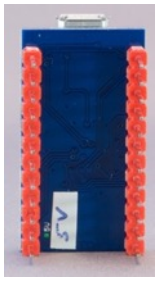
[1] FTDI is fighting a war against counterfeit chips, in chapter "217 Counterfeit FTDI FT232RL chips" you'll find information about this problem and possible solutions.

9.3. Programming a sketch on the Arduino Nano

Follow the following steps to program an Arduino Nano:

- Connect your Arduino Nano to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO NANO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

10. Sparkfun Pro Micro - 5V/16MHz

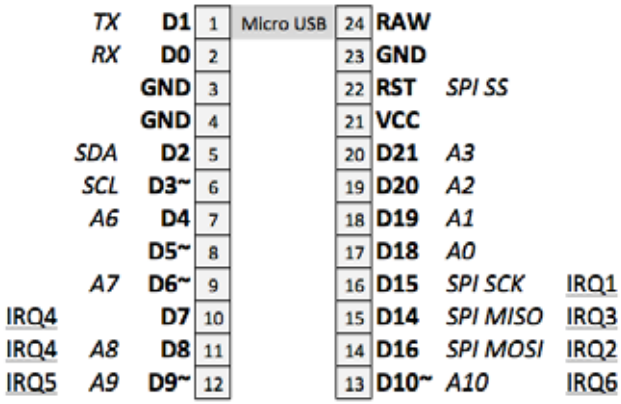



This is a very small Arduino and it even comes with real USB port than can be programmed to act as a HID (like a keyboard, mouse, joystick or MIDI device).

10.1. Specifications Sparkfun Pro Micro - 5V/16MHz

Microcontroller	Atmega32U4 16 MHz
Operating Voltage	5-12 V recommended
Digital I/O pins	12 (of which 5 PWM)
Analog input pins	9 these pins are also digital I/O
Flash memory	32 KB
SRAM	2.5 KB
EEPROM	1 KB
USB to TTL serial convertor	ATmega32U4
UART	1
Onboard user LED	none
Hardware interrupts	5
DC current I/O pins	10mA (5 recommended) 150mA entire package

10.2. Layout/connections Sparkfun Pro Micro - 5V/16MHz



10.3. First time preparation of Arduino IDE for Sparkfun Pro Micro - 5V/16MHz

This section describes how to add support for the Attiny45/85 to the Arduino IDE. More details can be found at: https://learn.sparkfun.com/tutorials/pro-micro--fio-v3-hookup-guide.

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and paste the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator. https://raw.githubusercontent.com/sparkfun/Arduino_Boards/master/IDE_Board_Manager/package_sparkfun_index.json

- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for Pro Micro (this could take a few seconds) and select the SPARKFUN AVR BOARDS BY SPARKFUN ELECTRONICS.
- Click on the INSTALL button
- Press the CLOSE button.
- Sparkfun Pro Micro is now listed at TOOLS, BOARD

10.4. Programming a sketch on the Sparkfun Pro Micro - 5V/16MHz

Follow the following steps to program a Pro Micro:

- Connect your Pro Micro to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the SPARKFUN PRO MICRO.
- Choose TOOLS, PROCESSOR, then select ATMEGA32U4 (5V, 16MHz) don't choose the default 3.3V, 8MHz, or you will damage the bootloader.[1]
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

[1] After damaging the bootloader, the USB port was not recognized any more. By using an USB to ISP programmer I was able to upload the bootloader again with the Arduino IDE.

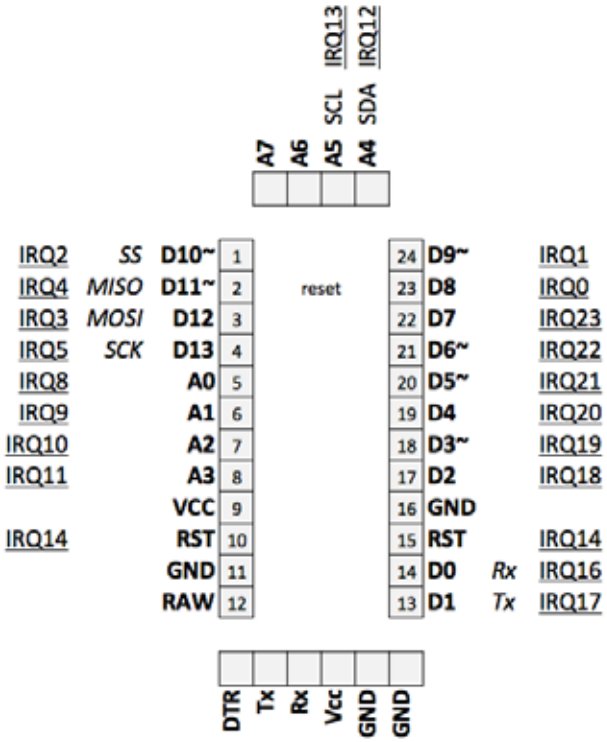
11. Arduino Pro Mini ATmega168U 5v, 16MHz


This board is about the same size as the Sparkfun Pro Micro, but it lacks an USB to TTL Serial adapter.

11.1. Specifications Arduino Pro Mini ATmega168U 5v, 16MHz

Microcontroller	Atmega168U 16 MHz
Operating Voltage	5,5-12 V recommended
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	8
Flash memory	16 KB
SRAM	1 KB
EEPROM	512 B
USB to TTL serial convertor	none
UART	1
Onboard user LED	D13
Hardware interrupts	22
DC current I/O pins	40mA (20 recommended) 200mA entire package (150mA when RAW powered)

11.2. Layout/connections Arduino Pro Mini ATmega168U 5v, 16MHz

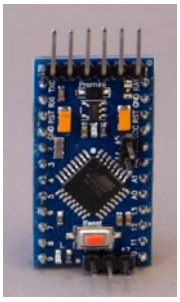


11.3. Programming a sketch on the Arduino Pro Mini ATmega168U 5v, 16MHz

Follow the following steps to program a Pro Micro:

- Make the following connections between an USB to TTL Connect your Pro Micro to your computer using an USB to TTL Serial adapter (any will do).
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the Arduino Pro or Pro Mini.
- Choose TOOLS, PROCESSOR, then select ATMEGA168U (5V, 16MHz).
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.
- Press and release the reset button during compilation, but before the sketch gets uploaded.

12. Arduino Pro Mini ATmega328p 3.3V, 8MHz

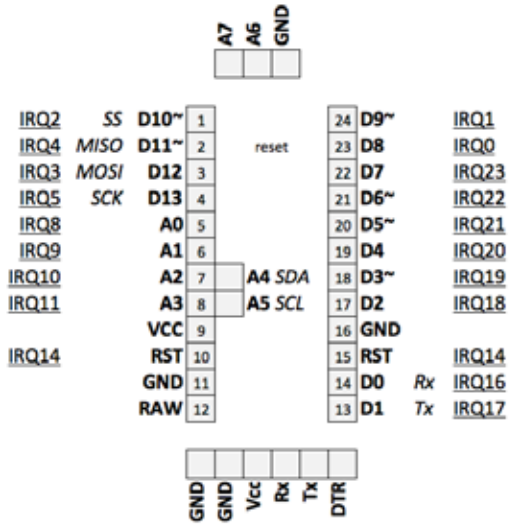



This board is about the same size as the Sparkfun Pro Micro, but it lacks an USB to TTL Serial adapter.

12.1. Specifications Arduino Pro Mini ATmega328p 3.3V, 8MHz

Microcontroller	Atmega328p 8 MHz
Operating Voltage	3,5-12 V recommended
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	8
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL serial convertor	none
UART	1
Onboard user LED	D13
Hardware interrupts	22
DC current I/O pins	40mA (20 recommended) 200mA entire package (150mA when RAW powered)

12.2. Layout/connections Arduino Pro Mini ATmega328p 3.3V, 8MHz



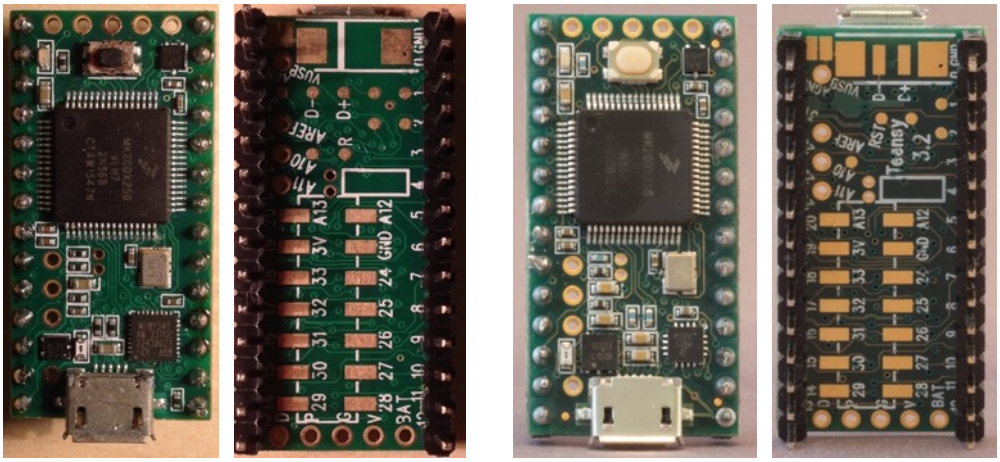
12.3. Programming a sketch on the Arduino Pro Mini ATmega328p 3.3V, 8MHz

Follow the following steps to program a Pro Micro:

- Make the following connections between an USB to TTL Connect your Pro Micro to your computer using an USB to TTL Serial adapter (any will do).
- Load the sketch you want to compile and upload.

- Choose TOOLS, BOARD, then select the Arduino Pro or Pro Mini.
- Choose TOOLS, PROCESSOR, then select ATMEGA328 (3.3V, 8MHz).
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.
- Press and release the reset button during compilation, but before the sketch gets uploaded.

13. Teensy 3.1 & 3.2

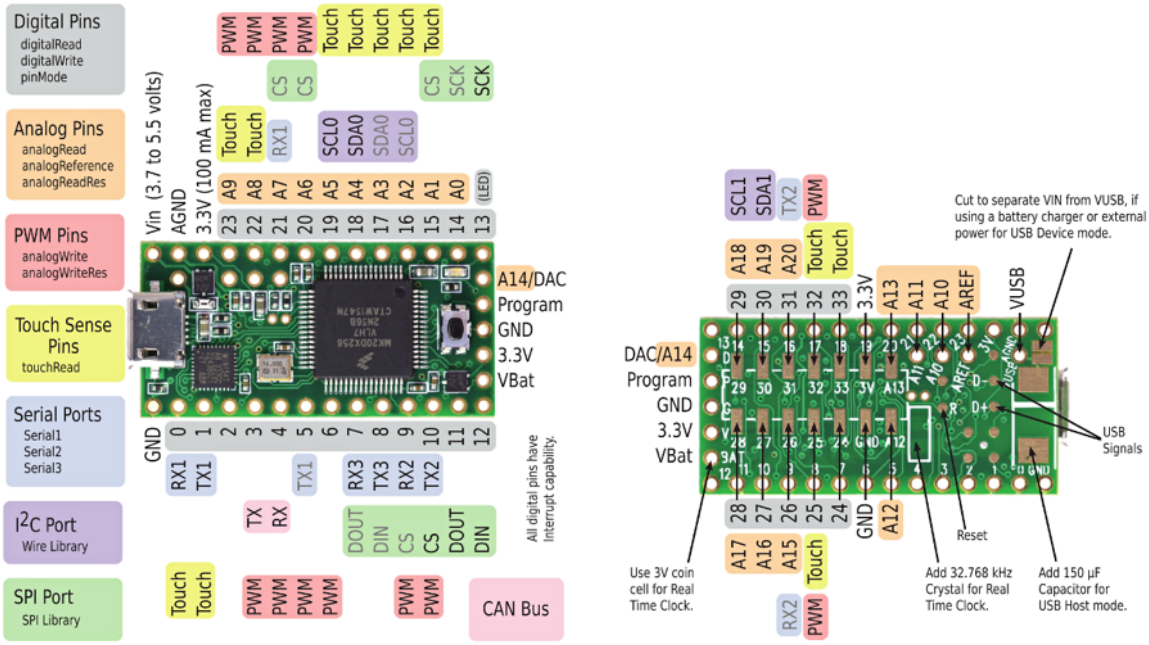


Teensy is not an Arduino but can be programmed through the Teenduino add-on for Arduino IDE. Although the Teensy is much smaller than most Arduino's, it is much more powerful. It is faster, has more memory (FLASH and RAM), more pins, more interrupts and it even comes with an USB port than can be programmed to act as a HID (like a keyboard, mouse, joystick or MIDI device).

13.1. Specifications Teensy 3.1 & 3.2

Microcontroller	MK20DX256VLH7, Cortex-M4, 72 MHz (overclockable: 96 MHz)
Operating Voltage	3.3 V **(5V tolerant)**
Vin	Teensy 3.1: 3.7V to 5.5V Teensy 3.2: 3.6V to 6.0V
Flash memory	256 KB
RAM	64 KB
EEPROM	2 KB
DMA	16 channels
Digital IO	34 pins (12 PWM)
Analog Input	21 pins/16 bits (12 touch sensing)
Analog output	1 pin / 12 bits
USB (HID)	1
Serial (UART)	3
SPI	1
I^2C	2
CAN bus	1
Onboard LED	port 13
Power regulator	Teensy 3.1: 100mA Teensy 3.2: 250 mA (powerful enough to feed an ESP8266)

Connections Teensy 3.1 & 3.2



This picture is for the Teensy 3.1. There are 2 differences between version 3.1 & 3.2, that is the better power regulator for the latter, capable of delivering more current and working with a broader input voltage range.

13.2. First time preparation Arduino IDE

Before you can use your Teensy with the Arduino IDE, you first need to install the Teensyduino add-on to the Arduino IDE. You can find a tutorial a t the following URL:

http://www.pjrc.com/teensy/td_download.html

At this moment Teensyduino only has beta support for Arduino IDE 1.6.12. This will soon be fixed I hope.

During the install of Teensyduino you can select which modified libraries you want to install to be used with the Teensy.

13.3. Programming a sketch on the Teensy

Since the Teensy can be programmed with HID functionality, programming can be difficult sometimes. For example, if your Teensy was programmed to move your mouse in circles, it can be very difficult to move the mouse pointer to the Upload button. Therefore, follow the following steps when you need to program your Teensy.

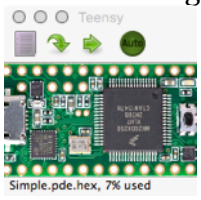
1. Move the cursor to an empty line below the last line of code you are working on.
2. Select the correct Teensy board in the Arduino IDE, through TOOLS, BOARD.
3. If you are using HID functionality (like keyboard, mouse etc..) choose the correct USB type through TOOLS, USB TYPE. Your sketch will not compile correctly, if you omit this step and you will see the following message:

```
sketch_oct17b:7: error: 'Keyboard' was not declared in this
scope
To make a USB Keyboard, use the Tools > USB Type menu
   Keyboard.println("Lorem ipsum dolor sit amet, consectetur
adipescent elit.");
   ^
'Keyboard' was not declared in this scope
```

4. You don't need to select a port, since the Arduino IDE is calling for Teensyduino and not a USB to serial device. Teensyduino is using other techniques to upload the code to your Teensy.
5. Connect your Teensy and ***immediately*** press the program button on your Teensy. This is to prevent the HID functionality (that could've been programmed in the Teensy) to come in action.

6. Press the UPLOAD button. After compiling, Arduino IDE will call Teensyduino to upload the compiled sketch to the Teensy, reboot the Teensy, and the sketch will start running. *That's why it is nice to have your cursor somewhere below the last line of code.*
7. To stop this code from running, perform the following steps:
   a. Click on the AUTO button in Teensyduino so it will be grayed out.

   b. Press the Program button on your Teensy.



8. After modifying your code, repeat from step 6.

Summary:
1. Cursor on empty line.
2. TOOLS, BOARD, Teensy ...
3. TOOLS, USB Type, ...
4. Leave Port as is.
5. Connect Teensy and ***immediately*** press program button
6. Press Upload
7. Stop code from running:
   a. Click on AUTO button in Teensyduino
   b. Press Program button on Teensy
8. Repeat from step 6 after modifying your sketch.

13.4. Compatibility Arduino Libraries

Use the following link to check a list of compatible Arduino libraries.
https://www.pjrc.com/teensy/td_libs.html

13.5. HID Samples

You can program the Teensy with almost the same sketches as a real Arduino. The main difference between a Teensy and an Arduino is the HID functionality. You can program the Teensy to act as a keyboard, mouse, joystick etc...

This paragraph contains two samples in which the Teensy is programmed as a HID.

Keyboard sample

The following code will type the text " Lorem ipsum dolor sit amet." every 5 seconds.

- Copy the code below.
- Make sure to select a USB type that includes keyboard functionality (TOOLS, USB TYPE, ...).
- Upload sketch (see previous paragraph about programming the Teensy).

More about using the Teensy as a keyboard device can be found at:
https://www.pjrc.com/teensy/td_keyboard.html

021_TeensyKeyboard.ino

```
void setup()
{
}

void loop()
{
  Keyboard.println("Lorem ipsum dolor sit amet.");
  delay(5000);
}
```

Mouse sample

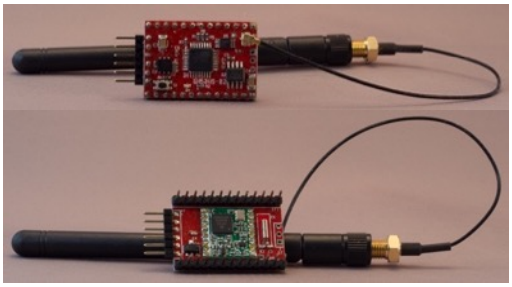
- The following code moves your mouse in a square and repeats this every 5 seconds.
- Make sure to select a USB type that includes mouse functionality (TOOLS, USB TYPE, ...).
- Upload sketch (see previous paragraph about programming the Teensy).

More about using the Teensy as a mouse device can be found at:
https://www.pjrc.com/teensy/td_mouse.html

022_TeensyMouse.ino

```
void setup()
{
}

void loop()
{
    Mouse.move(0, 127);
    delay(1000);

    Mouse.move(127, 0);
    delay(1000);

    Mouse.move(0, -127);
    delay(1000);

    Mouse.move(-127, 0);

    delay(5000);
}
```

14. LoRa: Nexus

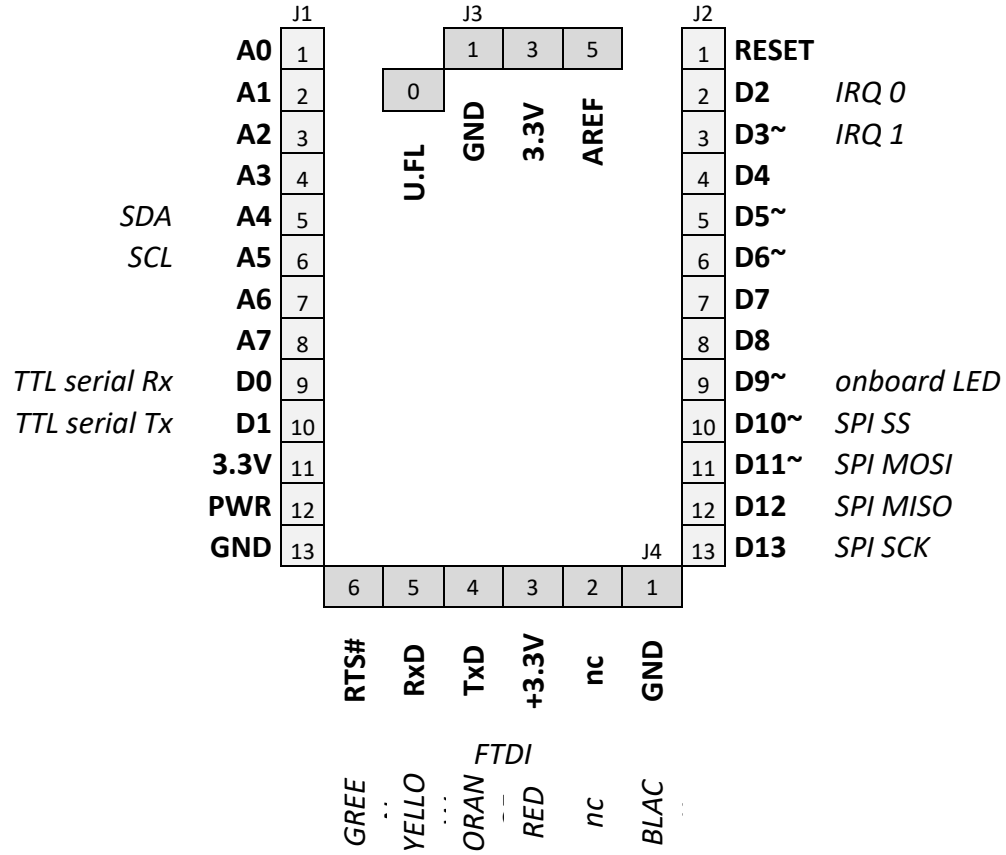


This specialized board was developed by Ideetron in The Netherlands and is based on an Arduino Mini and a RFM95W or RFM98W LoRa module.

14.1. Specifications LoRa Nexus

Microcontroller	Atmega328P-AU
Operating Voltage	
Digital I/O pins	14
Analog input pins	8
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL serial convertor	none
UART	1
Special connectors	U.Fl antenna connector
LoRa module	RFM95W or RFM98W

14.2. Layout/connections LoRa Nexus



14.3. Datasheets

Nexus

http://webshop.ideetron.nl/Files/3/1000/1211/Attachments/Product/4778I8m8OV005G7UF6f3815000Fu1Z48.pdf

Datasheet RFM95W

http://webshop.ideetron.nl/Files/3/1000/1211/Attachments/Product/38C8M4K5j578I7Go79331xub2L8Dr2Al.pdf

14.4. Connections between RFM95W and Nexus

The RFM95W is presoldered on the back of the Nexus board and according to the datasheet of the Nexus, the following Nexus pins are used.

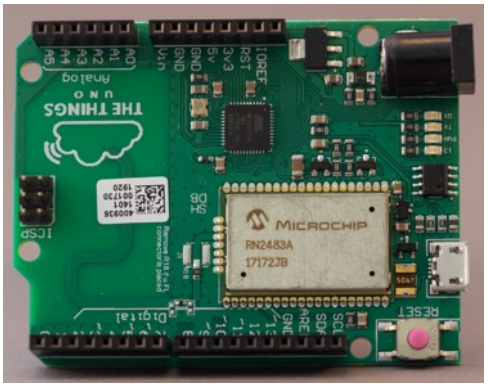
Pin nr	Name	Description	Nexus pin
1	GND	Ground	GND
2	MISO	SPI Data output	D12
3	MOSI	SPI Data input	D11
4	SCK	SPI Clock input	D13
5	nSS	SPI Chip Select input	D10
6	RST	Reset trigger input	RST
7	DIO5	Digital I/O software	D6
8	GND	Ground	GND
9	ANT	Antenna	n.a.
10	GND	Ground	GND
11	DIO3	Digital I/O software configured	n.c.
12	DIO4	Digital I/O software configured	n.c.
13	3.3V	Power Supply	3.3V
14	DIO0	Digital I/O software configured	D4
15	DIO1	Digital I/O software configured	D5
16	DIO2	Digital I/O software configured	D7

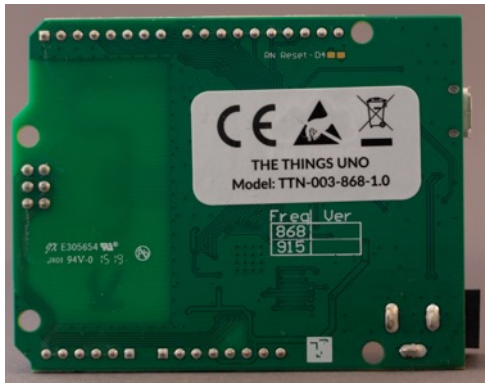
14.5. Programming a sketch on the Nexus

Follow the following steps to program a Nexus:

- Connect your Nexus to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO PRO OR PRO MINI.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

15. The Things Uno





The Things Uno is based on the Arduino Leonardo (not the Arduino Uno), with an added Microchip LoRaWAN module (RN2483A). Compared to the Arduino Uno is has more digital I/O and anolog Input pins. It also contains 25% more SRAM and it can serve as a USB HID.

15.1. Specifications The Things Uno

Microcontroller	ATmega32U4 16MHz
Bootloader	Arduino Leonardo
Input Voltage	7-12V (6-20V limits)
Digital I/O pins	20 (of which 7 PWM)[1]
Analog input pins	12
Operating Voltage	5V
USB to TTL Serial converter	included in the ATmega32U4 on D0 & D1 and also on a Grove connector
Flash memory	32 KB
SRAM	2,5 KB
EEPROM	1 KB
I2C	On A5 & A4 and also on 2 separate pins
SPI	Only on the ICSP header, not connected to any of the digital I/O pins.
USB port	micro-USB
Current at 3.3V pin	50mA
Current per I/O pin	40 mA

15.2. Documentation The Things Uno

Quickstart The Things Uno on The Things Network:

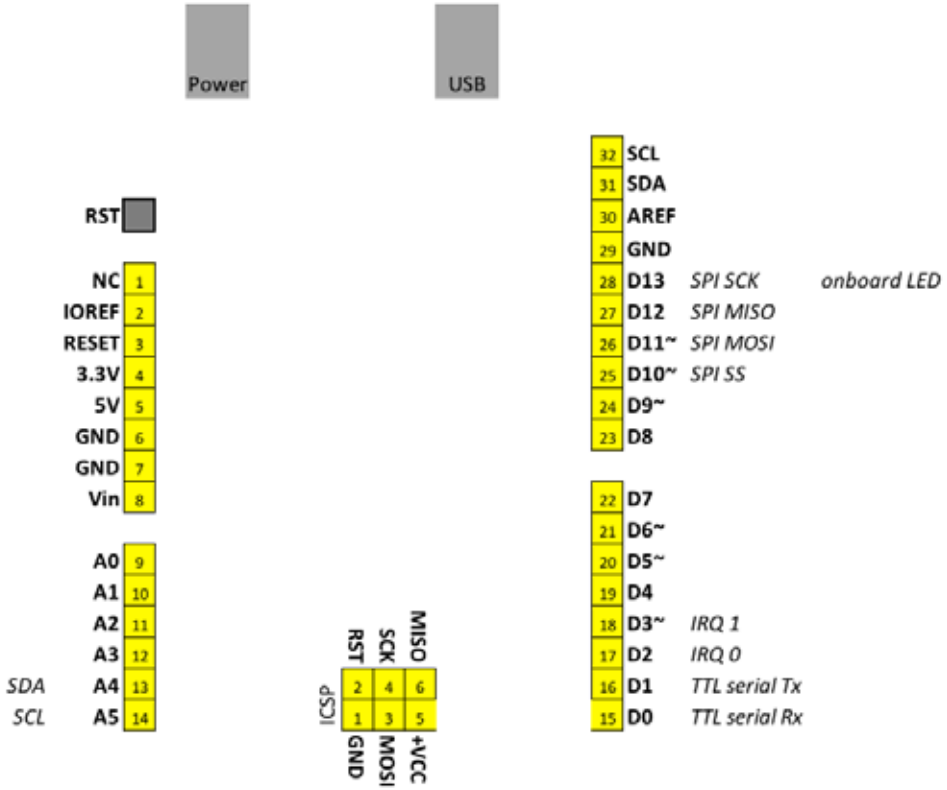
- https://www.thethingsnetwork.org/docs/devices/uno/quick-start.html

Documentation Arduino Leonardo:

- https://store.arduino.cc/arduino-leonardo-with-headers

[1] The following thread discusses how to use (address) the extra 6 digital and the 7 extra analoge pins (compared to the Arduino Uno).

15.3. Layout/connections The Things Uno

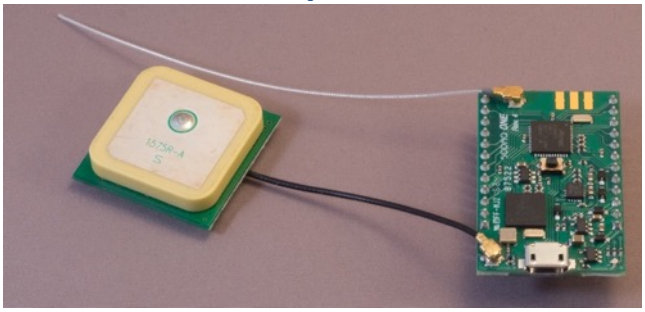
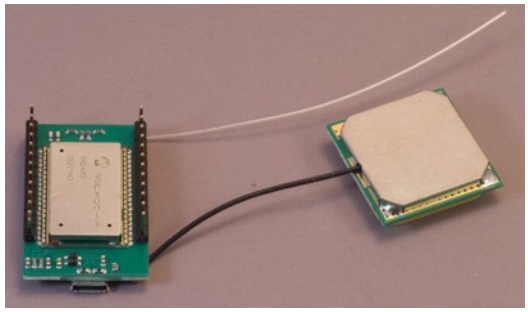


15.4. Programming a sketch on the The Things Uno

Follow the following steps to program The Things Uno:

- Connect your The Things Uno to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the ARDUINO LEONARDO.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

16. LoRa: Sodaq One

This board is based on Sodaq's Autonomo 32 bits Arduino compatible platform. It comes with a presoldered LoRa RN2483 module, a solar charge controller a GPS module and even an Accelerometer/Magnetometer. It runs on either a LiPo battery, or on USB power.

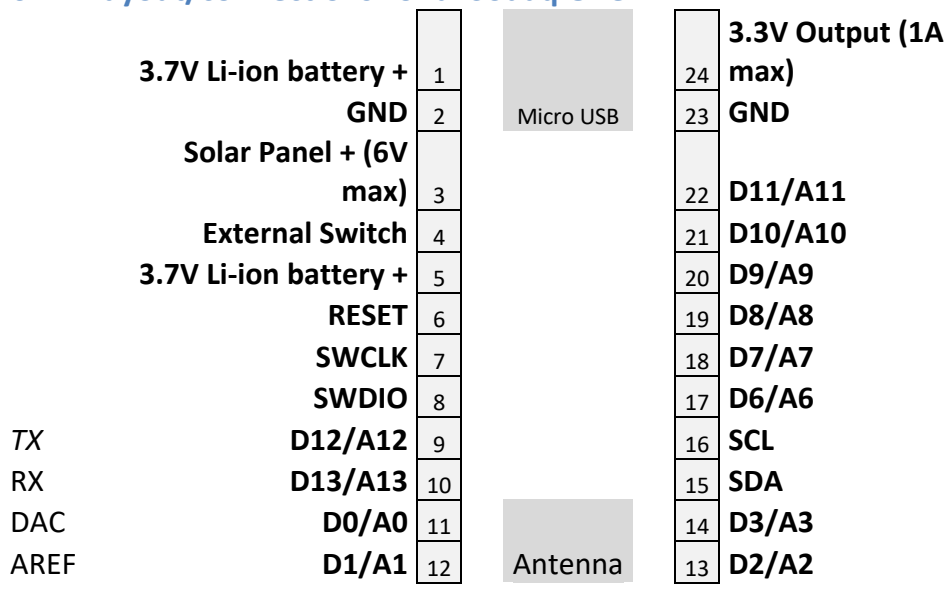
More information about the Sodaq One, can be found at the following URL:

- http://support.sodaq.com/sodaq-one/

16.1. Specification LoRa: Sodaq One

Description	Sodaq ONE v1	Sodaq ONE v2
Microcontroller	ATSAMD21G18, 32-Bit ARM Cortex M0+ Arduino MO compatible	*same as v1*
Clock Speed	48 MHz	*same as v1*
Operating Voltage	3.3V	*same as v1*
Power Supply	USB (5V) or LiPo (3.7V)	*same as v1*
Charging	Solar Panel Charging controller, up to 500 mA charge Current	
Digital I/O pins	14, all can be used for digital and analog with PWM, UART, SPI and TWI (I2C)	*same as v1*
Internal I/O ports	8	*same as v1*
External Interrupts	All 14 pins	*same as v1*
Analog Input	10-bit DAC	*same as v1*
DC current per I/O pin	7mA	*same as v1*
Flash memory	256 KB	*same as v1*
SRAM	32 KB	*same as v1*
EEPROM	Up to 16 KB by emulation	*same as v1*
LED	RGB LED	*same as v1*
Lora Reset line	Not connect	Connected
Antenna connector	U.Fl or SMA	*same as v1*
LoRa module	RN2483 868/433 or 915 MHz	RN2483 868 or 915 MHz
GPS module	uBlox EVA-7M	uBlox EVA-8M
Other modules	LSM303D Accelerometer/Magnetometer	LIS3DE Accelerometer

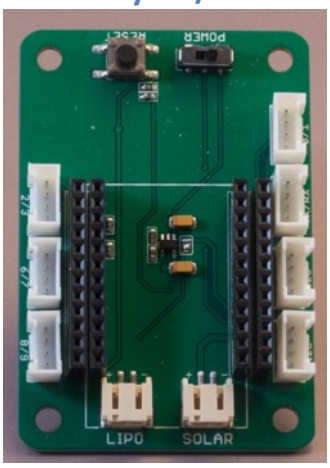
16.2. Layout/connections LoRa: Sodaq One

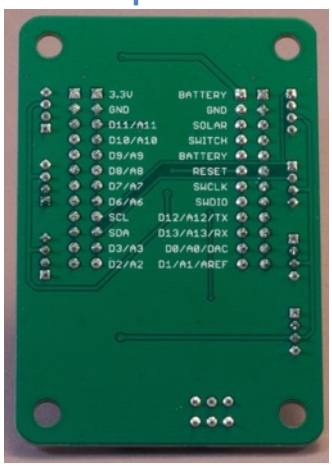


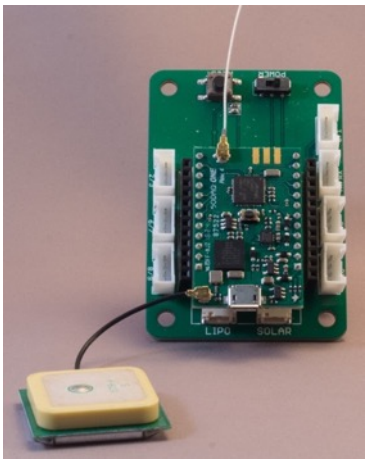
Next to these external pins, the Sodaq One also has 8 extra internal I/O ports

Name	Internal Pin/Port	Definition in Arduino IDE	Description
Red LED	D14 *OUTPUT*	LED_RED	Set this pin LOW to turn ON the RED LED
Green LED	D15 *OUTPUT*	LED_GREEN	Set this pin LOW to turn ON the Green LED
Blue LED	D16 *OUTPUT*	LED_BLUE	Set this pin LOW to turn ON the Blue LED
GPS Time pulse	D17 *INPUT*	GPS_TIMEPULSE	
GPS Switch	D18 *OUTPUT*	GPS_ENABLE	Set this pin high turn on the GPS module
User Button	D19 *INPUT*	BUTTON	High when not pressed, low when pressed
Power Enable	D22 *OUTPUT*	ENABLE_PIN_IO	
External Switch	D23 *INPUT*	SWITCH_SENSE	Used to sense the position of an external switch connected between the External Switch pin and the battery.

16.3. Layout/connections Sodaq One Base Board

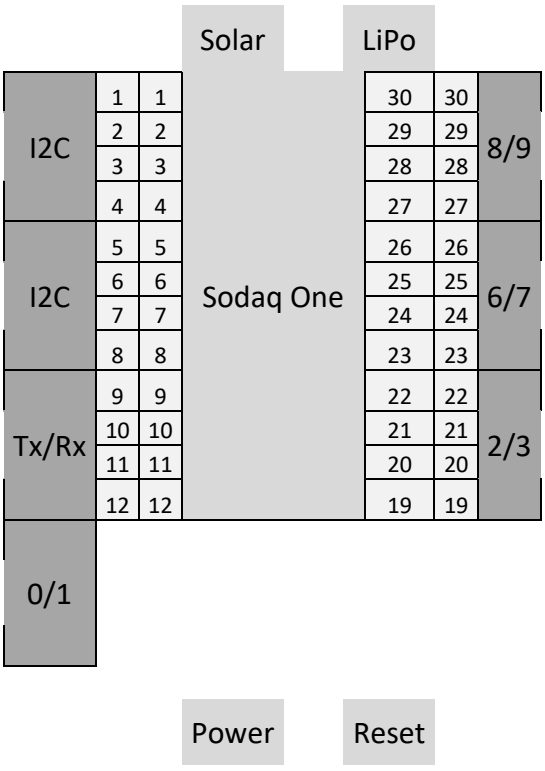






You can use the Sodaq One directly on a breadboard, or use the Sodaq One Base Board.

- 7 Grove compatible connectors exposing all 14 I/O pins
  - 2x I2C
  - 4x digital/analog port couples (0/1, 2/3, 6/7 and 8/9)
  - 1x Tx/Rx
- Solar panel connector
- LiPo connector (3.7V)
- Power switch
- Reset switch



16.4. Datasheets

Sodaq One schematics

- http://support.sodaq.com/sodaq-one/schema-sodaq-one/

RN2483 LoRa module

- The datasheet for the RN2483 can be found at: http://www.farnell.com/datasheets/1947946.pdf?_ga=1.80085719.1148387527.1482060915
- Command reference guide: http://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf
- Several other interesting documents can be found at: http://www.microchip.com/wwwproducts/en/RN2483

Sodaq One v1 module

- LSM303D Accelerometer/ magnetometer
  - https://www.pololu.com/file/download/LSM303D.pdf?file_id=0J703
- uBlox EVA-7M
  - https://www.u-blox.com/sites/default/files/products/documents/EVA-7M_DataSheet_(UBX-13000581).pdf

Sodaq One v2 modules

- LIS3DE Accelerometer (Soda
  - http://www.st.com/content/ccc/resource/technical/document/datasheet/group3/10/aa/64/5a/f1/cc/49/7f/DM00066267/files/DM00066267.pdf/jcr:content/translations/en.DM00066267.pdf
- uBlox EVA-8M
  - https://www.u-blox.com/sites/default/files/EVA-8M_DataSheet_(UBX-16009928).pdf

16.5. First time preparation of Arduino IDE for Sodaq One

This section describes how to add support for the Sodaq One to the Arduino IDE. More details can be found at: http://support.sodaq.com/sodaq-one/#one-vs-one-v2 in the section "Setting up your Arduino IDE".

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and past the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator.)
  http://downloads.sodaq.net/package_sodaq_samd_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for Sodaq (this could take a few seconds) and select the SODAQ SAMD BOARDS BY SODAQ.
- Click on the INSTALL button
- Press the CLOSE button.
- Sodaq One is now listed at TOOLS, BOARD.

16.6. Programming a sketch on the Sodaq One

Follow the following steps to program a Sodaq One:

- Connect your Sodaq One to your computer using an USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the SODAQ ONE.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

16.7. Sample Sodaq One: RGB LED[1]

The following sketch shows how to use the onboard RGB LED. These LED's are connected to D14, D15 and D16 and can be addressed with the predefined variables LED_RED, LED_GREEN and LED_BLUE.

The LED's are active when the corresponding lines are LOW.

023_SodaqRGB.ino

```
void setup()
{
  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
}

void loop()
{
  digitalWrite(LED_RED, LOW);
  digitalWrite(LED_GREEN, HIGH);
  digitalWrite(LED_BLUE, HIGH);
  delay(500);

  digitalWrite(LED_RED, HIGH);
  digitalWrite(LED_GREEN, LOW);
  digitalWrite(LED_BLUE, HIGH);
  delay(500);

  digitalWrite(LED_RED, HIGH);
  digitalWrite(LED_GREEN, HIGH);
  digitalWrite(LED_BLUE, LOW);
  delay(500);
}
```

[1] The original script was copied from the Sodaq website and can be found at: http://support.sodaq.com/sodaq-one/ at SODAQ ONE, TUTORIALS.

16.8. Sample Sodaq One: Serials

The Sodaq One has 3 hardware serials:

- SerialUSB is connected to your computer through the USB cable. Opening the serial monitor will not reset the sketch. You could use the following code to wait for the serial monitor to open.

```
while ((!SerialUSB) && (millis() < 30000)) {}
```

- Serial is connected to pin D12/A12/TX and D13/A13/RX.
- Serial1 is connected to the RN2483 LoRaWAN module and is used to send commands to the RN2483.

The following sketch prints the text "Hello, World" to the serial monitor.

024_SodaqSerialUSB.ino

```
void setup()
{
  SerialUSB.begin(9600);
}

void loop()
{
  SerialUSB.println("Hello, world!");
  delay(1000);
}
```

Most sketches found on the Internet, use Serial instead of SerialUSB for writing to the serial monitor. By adding the line `#define Serial SerialUSB` to those sketches, you can use them with your Sodaq One.

025_SodaqSerial.ino

```
#define Serial SerialUSB

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println("Hello, world!");
  delay(1000);
}
```

16.9. Sample Sodaq One: Sample Button[1]

On top of the Sodaq One, there is a user button. This button is not a reset button, but it is connected to the internal port D19 and can be addressed with the predefined variable BUTTON. This button is connected to an internal pull-up resistor, so it is LOW when pressed.

026_SodaqButton.ino

```
void setup()
{
  pinMode(BUTTON, INPUT_PULLUP);
  pinMode(LED_GREEN, OUTPUT);
}

void loop()
{
  int sensorVal = digitalRead(BUTTON);
  if (sensorVal == HIGH)
  {
    digitalWrite(LED_GREEN, HIGH);
  }
  else
  {
    digitalWrite(LED_GREEN, LOW);
  }
}
```

[1] The original script was copied from the Sodaq website and can be found at: http://support.sodaq.com/sodaq-one/ at SODAQ ONE, TUTORIALS.

16.10. Sample Sodaq One: Accelerometer/Magnetometer[1]

To use the LSM303D Accelerometer/Magnetometer, you need the LSM303 library by Pololu. This library can be installed through the Library Manager.

The following sketch shows the values for both the Accelerometer and the Magnetometer.

161_Sodaq_AccMagn.ino

```
#include <Wire.h>
#include <LSM303.h>

LSM303 compass;

char report[80];

void setup()
{
  SerialUSB.begin(9600);
  Wire.begin();
  compass.init();
  compass.enableDefault();
}

void loop()
{
  compass.read();
  snprintf(report, sizeof(report), "A: %6d %6d %6d    M: %6d %6d %6d",
           compass.a.x, compass.a.y, compass.a.z,
           compass.m.x, compass.m.y, compass.m.z);
  SerialUSB.println(report);
  delay(100);
}
```

[1] The original script was copied from the Sodaq website and can be found at: http://support.sodaq.com/sodaq-one/ at SODAQ ONE, TUTORIALS.

16.11. Sample Sodaq One: GPS[1]

To use the GPS, you need the Sodaq Ublox GPS library. This library can be found at:

- https://github.com/SodaqMoja/Sodaq_UBlox_GPS

http://downloads.sodaq.net/package_sodaq_index.json
The following sketch will show your GPS coordinates at different (increasing) intervals.

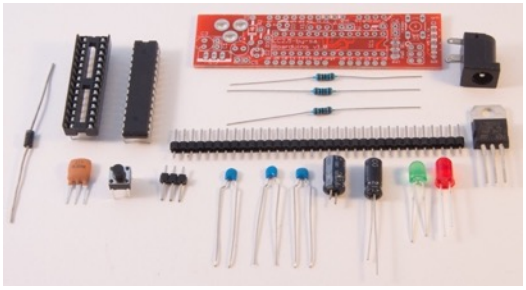
162_Sodaq_GPS.ino

```
#include <Arduino.h>
#include <Sodaq_UBlox_GPS.h>

#define MySerial        SERIAL_PORT_MONITOR
#define ARRAY_DIM(arr)  (sizeof(arr) / sizeof(arr[0]))

uint32_t intervals[] = {
  1UL * 60 * 1000,
  1UL * 60 * 1000,
  1UL * 60 * 1000,

  2UL * 60 * 1000,
  2UL * 60 * 1000,
  5UL * 60 * 1000,
  5UL * 60 * 1000,

  15UL * 60 * 1000,
  30UL * 60 * 1000,
  1UL * 60 * 60 * 1000,
  3UL * 60 * 60 * 1000,
  4UL * 60 * 60 * 1000,
  8UL * 60 * 60 * 1000,
};
size_t interval_ix = 0;

void find_fix(uint32_t delay_until);
void do_flash_led(int pin);

void setup()
{
  delay(3000);
  while (!SerialUSB) {
  }

  MySerial.begin(57600);

  digitalWrite(LED_RED, HIGH);
  pinMode(LED_RED, OUTPUT);
  digitalWrite(LED_GREEN, HIGH);
  pinMode(LED_GREEN, OUTPUT);
  digitalWrite(LED_BLUE, HIGH);
  pinMode(LED_BLUE, OUTPUT);

  do_flash_led(LED_RED);
  do_flash_led(LED_GREEN);
  do_flash_led(LED_BLUE);

  MySerial.println("SODAQ LoRaONE test_gps is starting ...");

  sodaq_gps.init(GPS_ENABLE);
```

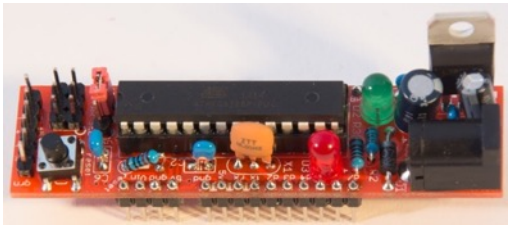
[1] The original script was copied from the Sodaq website and can be found at: http://support.sodaq.com/sodaq-one/ at SODAQ ONE, TUTORIALS.

```
  find_fix(0);
}

void loop()
{
  uint32_t wait_ms = intervals[interval_ix];
  if (++interval_ix > ARRAY_DIM(intervals)) {
    interval_ix = 0;
  }
  find_fix(wait_ms);
}

void find_fix(uint32_t delay_until)
{
  MySerial.println(String("delay ... ") + delay_until + String("ms"));
  delay(delay_until);

  uint32_t start = millis();
  uint32_t timeout = 900L * 1000;
  MySerial.println(String("waiting for fix ..., timeout=") + timeout +
String("ms"));
  if (sodaq_gps.scan(false, timeout))
  {
    MySerial.println(String(" time to find fix: ") + (millis() - start) +
String("ms"));
    MySerial.println(String(" datetime = ") +
sodaq_gps.getDateTimeString());
    MySerial.println(String(" lat = ") + String(sodaq_gps.getLat(), 7));
    MySerial.println(String(" lon = ") + String(sodaq_gps.getLon(), 7));
    MySerial.println(String(" num sats = ") +
String(sodaq_gps.getNumberOfSatellites()));
  } else {
    MySerial.println("No Fix");
  }
}

void do_flash_led(int pin)
{
  for (size_t i = 0; i < 2; ++i)
  {
    delay(100);
    digitalWrite(pin, LOW);
    delay(100);
    digitalWrite(pin, HIGH);
  }
}
```

18. Boarduino



Boarduino is a DYI-kit Arduino. You can buy the PCB and component separately or as a kit. Compared to the prices of the original Arduino boards, a Boarduino is much cheaper and smaller. The Boarduino lacks an UART (no serial port) and a programming chip, so you'll need some kind of programmer.



18.1. Specifications Boarduino

Same specs as the Arduino UNO, but it lacks the USB port and UART (serial port[1]), so you'll need an external programmer to upload your compiled sketches.

Microcontroller	Atmega328
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL Serial	none
UART	none
3.3V	NO AVAILABLE

18.2. Datasheet Boarduino

Atmega328P-PU

- http://www.farnell.com/datasheets/1684409.pdf

[1] To add a serial port to this programmer, see chapter 212

PL203HX USB to TTL Serial.

18.3. Building Instructions Boarduino

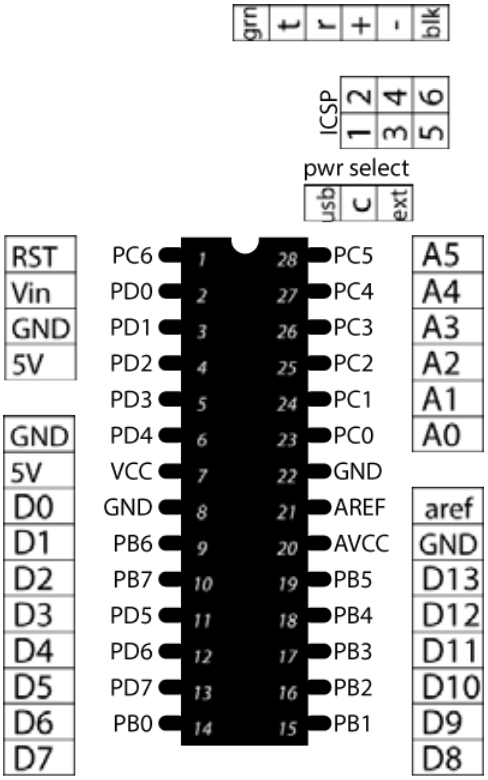
Shopping list:

#	Item	#	Item
1	Atmega328P-20PU	**4**	Ceramic Capacitor 0,1 uF
1	28-pin socket DIL	**1**	Electrolytic capacitor 46 uF / 25 V
1	16 MHZ ceramic resonator	**1**	Electrolytic capacitor 100 uF / 6,3 V
1	2.1 mm Power Jack PCB mount	**1**	LED red
1	1N4001 diode	**1**	LED green
1	5 V regulator 7805 TO-220 package	**1**	6mm tact switch
1	10K ohm, ¼ Watt resistor	**1**	Jumper
2	1K ohm, ¼ W resistor	**1**	Boarduino PCB
43	male header 0.1 "spacing 3x3 pin 2x6 pin 1x4 pin 1x8 pin 1x10 pin		

Instructions:

- http://learn.adafruit.com/boarduino-kits

18.4. Layout/connections Boarduino



18.5. Burning bootloader to a Boarduino

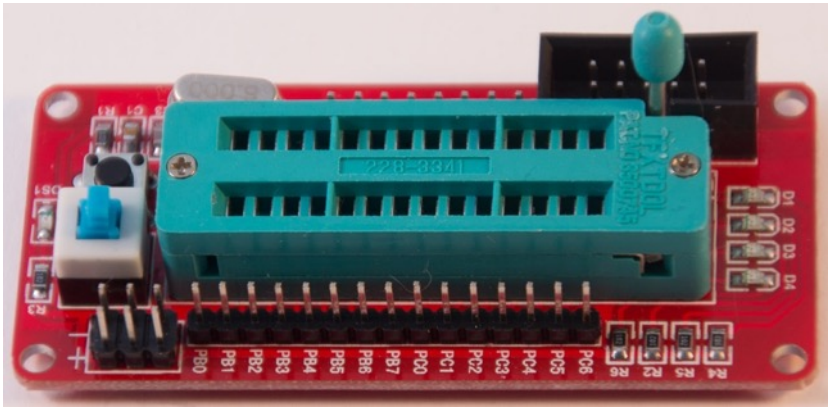
Before you can program a Boarduino with the Arduino IDE, you first have to upload a bootloader. You'll only have to do this once.

- Connect an **USBasp to the ISP headers**.
- Select TOOLS, BOARD, DUEMILANOVE/DIECIMILA.
- Select TOOLS, PROGRAMMER, USBasp
- Select TOOLS, PROCESSOR, ATMEGA328
- You don't need to set a port.
- Choose TOOLS, BURN BOOTLOADER.

18.6. Programming a Boarduino

- Connect an **USBasp to the ISP headers**.
- Select TOOLS, BOARD, DUEMILANOVE/DIECIMILA.
- Select TOOLS, PROGRAMMER, USBasp
- Select TOOLS, PROCESSOR, ATMEGA328
- You don't need to set a port.
- Choose TOOLS, BURN BOOTLOADER.
- Save you sketch before you upload it to your Boarduino, because with this method the sketch is not saved automatically!
- To upload your sketch, don't press the Upload button, but instead choose: SKETCH, UPLOAD USING PROGRAMMER.

19. AVR Development board as Arduino



With this development board you can program Atmega8, Atmega168 and Atmega328 MCU's through the 10 pins ISP connector. You can also use it as an Arduino Board without an UART (i.e. without a serial connection through USB). Use the layout below to match the Atmega328 to Arduino pin assignment.

19.1. Specifications AVR Development board as Arduino

Same specs as the Arduino UNO, but it lacks the USB port and UART (serial port[1]), so you'll need an external programmer to upload your compiled sketches.

Microcontroller	Atmega328
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
USB to TTL Serial	none
UART	none
3.3V	NOT AVAILABLE

19.2. Datasheet AVR Development board as Arduino

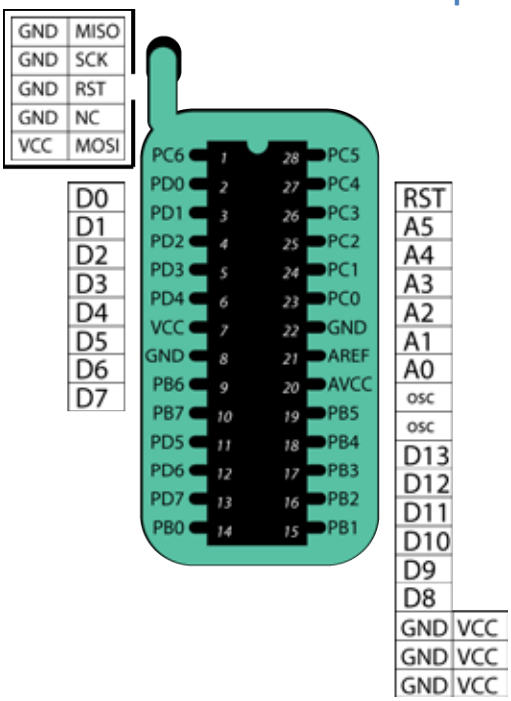
Atmega328P-PU

- http://www.farnell.com/datasheets/1684409.pdf

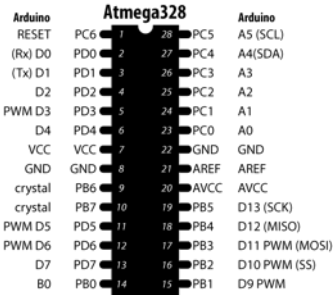
[1] To add a serial port to this programmer, see chapter 212

PL203HX USB to TTL Serial.

19.3. Connections AVR Development board as Arduino



20. Arduino on a breadboard



The most important part of your Arduino is the Atmega328P-PU. This MCU (Micro Controller Unit) and a 16MHz oscillator is the heart of your board. Putting this on a solder less breadboard together with some resistors and capacitors you can build your own Arduino board.

20.1. Specifications Arduino on a breadboard

Same specs as the Arduino UNO, but it lacks the USB port and UART (serial port[1]), so you'll need an external programmer to upload your compiled sketches.

Microcontroller	Atmega328
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
DC current per I/O pin	40 mA
DC current for 3.3V pin	50 mA
Flash memory	32 KB
SRAM	2 KB
EEPROM	1 KB
USB to TTL Serial	none
UART	none
3.3V	NOT AVAILABLE

20.2. Datasheet Arduino on a breadboard

Atmega328P-PU

- http://www.farnell.com/datasheets/1684409.pdf

[1] To add a serial port to this programmer, see chapter 212

PL203HX USB to TTL Serial.

20.3. Connections Atmega328P-PU

Atmega328	Arduino pin
PC0	A0
PC1	A1
PC2	A2
PC3	A3
PC4	A4
PC5	A5
PD0	D0 (Rx)
PD1	D1 (Tx
PD2	D2
PD3	D3 (PWM)
PD4	D4
PD5	D5 (PWM)
PD6	D6 (PWM)
PD7	D7

Atmega328	Arduino pin
PB0	D8
PB1	D9 (PWM)
PB2	D10 (SS, PWM)
PB3	D11 (MOSI, PWM)
PB4	D12 (MISO)
PB5	D13 (SCK)
PC6	Reset
VCC	5V
GND	GND
PB6	crystal
PB7	crystal
GND	GND
AREF	AREF
AVCC	?

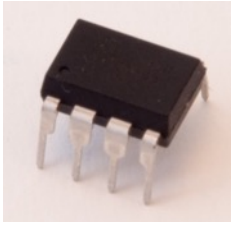
20.4. Sample Arduino on a breadboard

To build your own Arduino on a breadboard, you will need the following components:

- 5V regulated power supply
  - See chapter: 206 Black Wings
    OR
  - See chapter: 209 DC Step-Down Adjustable Power module
    OR
  - Self-made regulator:
    - 2x 7805 Voltage regulator
    - 2x 10 uF Capacitor
    - 1 x LED (optional)
    - 1 x 220 ohm Resistor (optional)
- 1 x Atmega328P-PU with an Arduino bootloader.
- 1 x 16 MHz oscillator
- 2 x 22pF
- 1 x LED (optional)
- 1 x 220 ohm Resistor (optional)
- 1x 10k Ohm Resistor
- 22 AWG wire (for example a piece of solid core UTP cable)
- small momentary NO (normally open) button

Instructions for building this Arduino on a breadboard can be found at:
http://arduino.cc/en/Main/Standalone

21. Attiny45/Attiny85



The Attiny chips are not real Arduino's, but can be used as a permanent replacement for projects with very limited port usage. Besides VCC and GND you don't need any other components, so it is ideal to shrinkify your projects.

Not all libraries and functions are available for the Attiny chips, but at lot of them have been ported.

The following Arduino commands should be supported:

- pinMode()
- digitalWrite()
- digitalRead()
- analogRead()
- analogWrite()
- shiftOut()
- pulseIn()
- millis()
- micros()
- delay()
- delayMicroseconds()
- SoftwareSerial

21.1. Specifications Attiny45/Attiny85

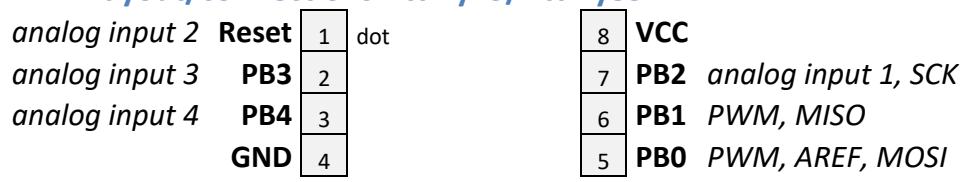
The Attiny45/85 lacks a lot of features that an Arduino board has, like a serial port[1] and a specialized Rx- (Receive data) and a Tx-pin (Transmit data).

Microcontroller	Attiny45/85
Operating Voltage	2.7-5.5 V (0-10 MHz) 4.5-5.5 V (0-20 MHz external clock)
Digital I/O pins	2 (PWM)
Analog input pins	3
Flash memory	4/8 KB
SRAM	256/512 B
EEPROM	256/512 B

[1] To add a serial port to this programmer you'll need to sacrifice 2 ports for Rx and Tx, the Software Serial library and a

PL203HX USB to TTL Serial (see chapter 212).

21.2. Layout/connections Attiny45/Attiny85



21.3. Datasheet Attiny45/Attiny85

http://www.atmel.com/Images/Atmel-2586-AVR-8-bit-Microcontroller-Attiny25-Attiny45-Attiny85_Datasheet.pdf

21.4. First time preparation of Arduino IDE for Attiny45/85

This section describes how to add support for the Attiny45/85 to the Arduino IDE. More details can be found at: http://highlowtech.org/?p=1695

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and past the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator.) https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for Attiny (this could take a few seconds) and select the ATTINY BY DAVID A. MELLIS.
- Click on the INSTALL button
- Press the CLOSE button.
- Attiny is now listed at TOOLS, BOARD.

First time preparation to run Attiny at 8 MHz

To make your Attiny also run at 8 MHz (this is needed for several libraries)[1], you will need to load the correct bootloader to your Attiny.

- Choose TOOLS, BOARD, ATTINY.
- Go to TOOLS, PROCESSOR, choose the appropriate Attiny (Attiny45/85/44/84).
- Choose TOOLS, POCESSOR, 8 MHZ INTERNAL[2]
- Choose which programmer you are going to use:
  - "33 Uploading the bootloader by using an Arduino as ISP"
  - "31 USBasp v2.0 programmer"
  - "35 Self-made Attiny 45/85 ISP adapter"
  - AVR programmer (not yet described in this document)
- Connect your programmer to your Attiny
  - Connect SS or RST to Attiny pin 1.
  - Connect MOSI to Attiny pin 5.

[1] http://42bots.com/tutorials/how-to-program-Attiny85-with-arduino-uno-part-2/

[2] Don't choose the external clock settings unless you have an external clock. After setting an external clock **you won't be able the change it back to internal before you add an external oscillator to your Attiny.**

    - Connect MISO to Attiny pin 6.
    - Connect SCK to Attiny pin 7.
    - Connect GND to Attiny pin 4.
    - Connect 5V to Attiny pin 8.
- Go to TOOLS, BURN BOOTLOADER.
- You should see a message saying : "***Done burning bootloader***

Below you will find a sample to control a servo with the alternative SoftwareServo library.

21.5. Programming the Attiny45/Attiny85[1]

- Choose TOOLS, BOARD, ATTINY.
- Go to TOOLS, PROCESSOR, choose the appropriate Attiny (Attiny45/85/44/84).
- Choose TOOLS, POCESSOR, 8 MHZ INTERNAL[2] or 1 MHZ INTERNAL. As long as you haven't programmed the 8MHz bootloader to your Attiny (see "First time preparation to run Attiny at 8 "), make sure your programmer supports a slow clock speed of 1 MHz that is default to the Attiny. For example, when you use the USBasp as a programmer you should place a jumper on JP3 to slow down the clock of the USBasp.
- Choose which programmer you are going to use:
    - "33 Uploading the bootloader by using an Arduino as ISP"
    - "31 USBasp v2.0 programmer"
    - "35 Self-made Attiny 45/85 ISP adapter"
    - AVR programmer (not yet described in this document)
- Connect your programmer to your Attiny
    - Connect SS or RST to Attiny pin 1.
    - Connect MOSI to Attiny pin 5.
    - Connect MISO to Attiny pin 6.
    - Connect SCK to Attiny pin 7.
    - Connect GND to Attiny pin 4.
    - Connect 5V to Attiny pin 8.
- Go to TOOLS. PROGRAMMER, choose the appropriate programmer (USBASP, ARDUINO AS ISP, ….)
- Press the UPLOAD button.

[1] http://42bots.com/tutorials/how-to-program-Attiny85-with-arduino-uno-part-1/

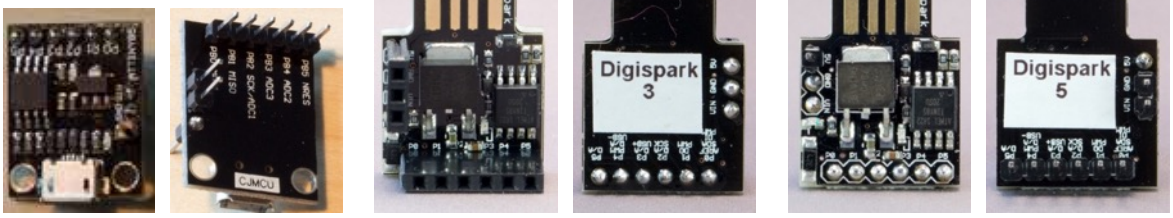
[2] Don't choose the external clock settings unless you have an external clock. After setting an external clock **you won't be able the change it back to internal before you add an external oscillator to your Attiny.**

Sample sketch Servo on the Attiny45/Attiny85

http://playground.arduino.cc/ComponentLib/Servo

```
#include <SoftwareServo.h>

SoftwareServo myservo;
int pos = 0;

void setup()
{
  myservo.attach(1);
}

void loop()
{
  for(pos = 0; pos < 180; pos += 10)
  {
    myservo.write(pos);
    delay(60);
    SoftwareServo::refresh();
  }
  for(pos = 180; pos>=1; pos-=10)
  {
    myservo.write(pos);
    delay(60);
    SoftwareServo::refresh();
  }
}
```

22. Attiny85 Digispark board



The Digispark board is one of the smallest Arduino's based on the Attiny85. It contains a USB A or USB micro connector for uploading sketches and power, a power-on led and an Internal LED connected to port 1 for testing purposes.

Not all libraries and functions are available for the Attiny chips, but at lot of them have been ported.

The following Arduino commands should be supported:

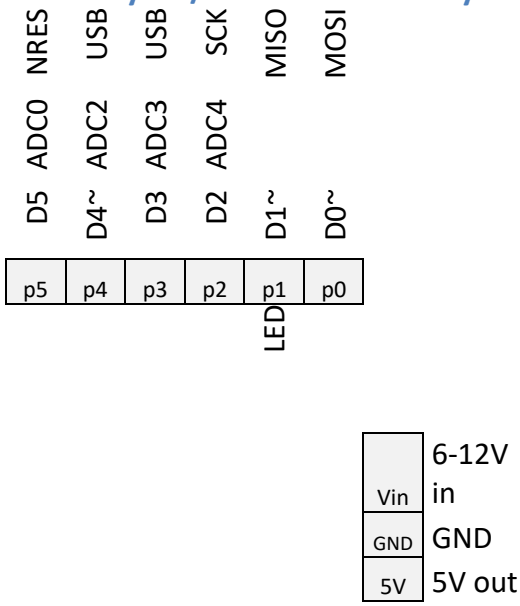
- pinMode()
- digitalWrite()
- digitalRead()
- analogRead()
- analogWrite()
- shiftOut()
- pulseIn()
- millis()
- micros()
- delay()
- delayMicroseconds()
- SoftwareSerial

22.1. Specifications Attiny85 Digispark board

The Attiny45/85 lacks a lot of features that an Arduino board has.

Microcontroller	Attiny85
Operating Voltage	5V or 7-12V
On-board power regulator	5V 500mA
Built-in USB	USB A or micro
I/O pins	6 (if your program actively communicates over USB, 2 of them are used for USB)
ADC	on 4 pins
PWM	on 3 pins (more possible with Software PWM)
Flash memory	8 KB (2KB used by bootloader)
SRAM	256/512 B
EEPROM	256/512 B
Protocols	I2C and SPI (via USI)
LED	Power LED and Test/Status LED
Pin5	has some limitations, output is more like 3.6V, works fine for non-current sourcing use!
GPIO output	20mA max per pin.

22.2. Layout/connections Attiny85 Digispark board



The Digispark is breadboard friendly on small breadboards only (20 holes wide, 4.8 cm wide). Before you solder the headers, you should decide first how you are going to use this board.

Without a breadboard

You are free to use Female or Male headers on top or on the bottom of the board.

With a small breadboard

Solder Male headers at the bottom, for P0..P5 and for Vin and GND. Solder one header (F/M) on TOP for 5V.

When you place the Digispark on your breadboard, Vin will match the VCC bus and GND the GND bus.

With a wide breadboard

Solder Male headers at the BOTTOM for P0..P5. Solder 3 headers on TOP for Vin, GND and 5V (F/M).
You can place the Digispark on your breadboard, but must use separate jumper wires to connect Vin and GND.

22.3. Wiki for Attiny85 Digispark board

http://digistump.com/wiki/digispark

22.4. First time preparation of Arduino IDE for the Attiny85 Digispark board

This section describes how to add support for the Attiny85 Digispark board to the Arduino IDE. More details can be found at: http://digistump.com/wiki/digispark/tutorials/connecting.

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and past the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator.)
  http://digistump.com/package_digistump_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for Digistump (this could take a few seconds) and select the DIGISTUMP AVR BOARDS BY DIGISTUMP.
- Click on the INSTALL button, on a Windows computer you will now be asked to install (or update) the drivers for your Attiny Digispark board.
- Press the CLOSE button.
- The Digispark boards are now listed at TOOLS, BOARD.
- During this process, all necessary libraries and some example sketches are also made available in Arduino IDE.
- If you are using version 2.0a4 of the Digistump board plugin on a Mac, you'll need to change some settings for 2 files. To do this, type the following commands in a Terminal window.

```
cd
cd Library/Arduino15/packages/digistump/tools/micronucleus/2.0a4/
chmod +x launcher
chmod +x micronucleus
```

22.5. Programming the Attiny85 Digispark board

- Choose TOOLS, BOARD, DIGISPARK (DEFAULT 16,5 MHz).
- Choose TOOLS, PROGRAMMER, USBTINYISP.
- **Disconnect your Attiny85 Digispark board from your computer.**
- Press the UPLOAD button to upload your sketch, wait until you see the message:
  Running Digispark Uploader...
  Plug in device now... (will timeout in 60 seconds)
- Now you can connect your Attiny Digispark board to your computer. The board will be detected automatically and the sketch will be uploaded.
  running: 100% complete
  >> Micronucleus done. Thank you!

Sample sketch Attiny85 Digispark board

```
void setup() {
pinMode(1, OUTPUT);
}

void loop() {
  digitalWrite(1, HIGH);
  delay(2000);
  digitalWrite(1, LOW);
  delay(2000);
}
```

23. ESP8266 MCU





The ESP8266 is a tiny, cheap and powerful MCU, it can be programmed with C++, Lua and with Python, it can even be programmed through the Arduino IDE. Beside that it can also be used as a cheap WiFi module on Arduino. More details can be found in the ESP8266 section.

Arduino IDE

This section describes Arduino IDE, the software you need to program your Arduino.

24. Arduino IDE

Arduino IDE is the most popular IDE to edit, compile and upload software projects (sketches) to the Arduino boards.

To edit, compile and upload software projects (sketches) from your computer, you need at least the Arduino IDE. Two other pieces of software are also recommended, i.e. Processing for the connection between your Arduino projects and your computer and Fritzing to create drawings. This chapter describes the Arduino IDE, Processing and Fritzing are described in the following chapters.

24.1. Arduino IDE software

With this Arduino Integrated Development Environment, you can edit, compile and upload Arduino sketches to the Arduino boards. In this document I refer to version 1.6.5, available for Windows, OS X and Linux (and an older version for Raspberry Pi).

Pict. 1 Sample of the sketch "BareMinimum"

24.2. Links for Arduino

- Home page
  http://arduino.cc/
- Tutorials
  http://arduino.cc/en/Guide/HomePage
  http://arduino.cc/en/Tutorial/HomePage
- Reference
  http://arduino.cc/en/Reference/HomePage
- Download IDE:
  https://www.arduino.cc/en/Main/Software
- Drivers:
  http://www.ftdichip.com/Drivers/VCP.htm

24.3. Getting started with Arduino IDE

When you start a new sketch, Arduino IDE loads its default empty sketch, as is copied below.

```
// Put the inclusions of libraries, declaration of constants
// and declaration and initializing of variables here

void setup()
{
     // Put your setup code here, to run once:
}

void loop()
{
     // Put your main code here, to run repeatedly:
     // The loop never ends!. Quiting the loop is performed
     // by turning off the power to the Arduino board.
}

// Put the declaration of functions and procedures here
```

Everything inside the `setup` procedure will be run only once. Code that is typed in the main procedure called `loop`, will be run continuously.

25. Serial Monitor

To upload your sketch to the Arduino board, most of the time you use some sort of USB to TTL Serial adapter or UART. This could be a built-in UART (UNO, NANO, Digispark), or an external UART (needed when uploading to Boarduino, Attiny and most ESP8266).

In case of an external UART (USB to TTL Serial adapter, or FTDI), you must connect the UART's Tx-pin (Transmit) with the Dx pin of the Arduino (D0), and the UART's Rx-pin (Receiver) with the Tx pin of the Arduino (D1). As you can see Transmit and Receive are crossed with each other.

In case of an internal UART (like the UNO, Nano and Digispark), this is automatically done by plugging in the USB cable in the Arduino board.

If other components are connected to D0 and D1, make sure to disconnect these components while uploading sketches.

Another use of these UARTs is to communicate between the Arduino board and your computer while your sketch is running. It's a great way to debug your sketch. You can write to and read from the serial port from within your sketch. To show this on your screen, you must open the Serial Monitor (TOOLS, SERIAL MONITOR)[1]. By opening this Serial Monitor, most Arduino boards will perform a reset, so your sketch will restart.

To open a connection to the serial port from within your sketch, you'll first have to place the following line within the procedure `setup`.

```
Serial.begin(9600);
```

Writing to the serial port can be done with one of the following commands:

```
Serial.print("The value for A1 is:");
```

This will print the message *The value for A1 is:*, leaving the cursor just behind the colon.

```
Serial.println(A1);
```

This will print the value of analog port A1, followed by a newline, so leaving the cursor at the beginning of the next line.

[1] You can also use terminal programs like PuTTY.

Output to Serial Monitor

The following sketch will fill your Serial Monitor with the word Arduino.

001_.SerialMonitorOutput.ino

```
void setup()
{
  Serial.begin(9600);
  delay(1000);
  Serial.println("Hello world, this is:");
}

void loop()
{
  Serial.println("Arduino");
  delay(500);
}
```

This is a great way for debugging. For example, you can display the value of a variable or the value of an input port.

Input from Serial Monitor

You can also use Serial Monitor to input to the Arduino board. The following sketch will ask you to type your name and will store your name in a String.

002_.SerialMonitorInput.ino

```
String name;

void setup()
{
  Serial.begin(9600);
  delay(1000);
  Serial.println("What is your name?");
}

void loop()
{
    if(Serial.available() > 0)
    {
        name = Serial.readStringUntil('\n');
        Serial.print("Welcome ");
        Serial.println(name);
    }
}
```

26. Libraries

Support for additional components or functions can be added by including libraries in your sketches. Support for many libraries is already built-in the Arduino IDE, but it is also possible to add new libraries. It is even possible to add self-made libraries, but that's beyond the scope of this document.

26.1. Built-in libraries

The following libraries are built-in the Arduino IDE:

- Bridge
- EEPROM
- Esplora
- Ethernet
- Firmata
- GSM
- LiquidCrystal
- Robot Control
- Robot IR Remote
- Robot Motor
- SD
- SPI
- Servo
- SoftwareSerial
- SpacebrewYun
- Stepper
- TFT
- Temboo
- WiFi
- Wire

26.2. Library examples

To get started with libraries, sample sketches are included with most libraries. You can find these samples at FILE, EXAMPLES and then grouped under the corresponding library name. To start working with the SD card reader library for example you can go to FILE, EXAMPLES, SD, CARDINFO. Before you compile and upload one of these samples, you must make sure that that sample is using the same pins as you. Off course you can change to sketch to match the pins that you've used in your setup.

Examples are always read-only. Changes you make to these sample sketches will not be saved. If you need to have these changes permanently, you must save that sketch in your sketches folder.

26.3. Adding libraries through the Library Manager

The preferred way to add libraries to Arduino is through the Library manager.

- Go to SKETCH, INCLUDE LIBRARY, MANAGE LIBRARIES….
- Browse through this list, or type keywords to search the available libraries.
- Select the library you want to install and then click on the INSTALL or UPDATE button.

- The library will now be installed and can be used immediately after installation, it is not needed to restart Arduino IDE.

26.4. Adding ZIP libraries

The second method of adding libraries is by adding a ZIP library.

- Download the ZIP file of the wanted library.
- Go to SKETCH, INCLUDE LIBRARY, ADD .ZIP LIBRARY.
- Browse to the downloaded ZIP library and click on CHOOSE.
- The library will now be installed and can be used immediately after installation, it is not needed to restart Arduino IDE.

26.5. Manually adding libraries

The third way of adding libraries is the manual way.

- Download the library files.
- Create a folder for your library inside your libraries folder.
- Place the downloaded library files in this new folder. Make sure that both the .cpp and the .h file is directly in this newly created folder.
- After placing a library manually, you need to restart Arduino IDE, otherwise the library will not be recognized.

26.6. Non device specific libraries

Descriptions	Library	Origin
Graphics	Adafruit_GFX.h	Library Manager
TWI (Two Wire Interface)	Wire.h	Arduino built-in
Print formatting	PrintEx.h	Library Manager

27. Board management

By default Arduino IDE supports several (original) Arduino AVR boards,. Luckily it is possible to add other boards as well. It is even possible to add support for a self-build board, but that's beyond the scope of this document.

27.1. Built-in Arduino AVR boards

Support for the following boards is built-in the Arduino IDE:

- Arduino Yún
- Arduino/Genuino Uno
- Arduino Duemilanova or Diecimila
- Arduino Nano
- Arduino/Genuino Mega or Mega 2560
- Arduino Mega ADK
- Arduino Leonardo
- Arduino Genuino Micro
- Arduino Esplora
- Arduino Mini
- Arduino Ethernet
- Arduino Fio
- Arduino BT
- LilyPad Arduino USB
- LilyPad Arduino
- Arduino Pro or Pro Mini
- Arduino NG or older
- Arduino Robot Control
- Arduino Robot Motor
- Arduino Gemma

27.2. Adding support for other boards

To add support for new boards, follow the steps below.

- Go to TOOLS, BOARD, BOARD MANAGER.
- Browse through this list, or type keywords to search the available boards.
- Select the board you want to install and then click on the INSTALL or UPDATE button.
- The board will now be installed and can be used immediately after installation, it is not needed to restart Arduino IDE.

In case the board you want to install is not in this list follow the following steps.

- Find the .json installation URL for the board. You can find a large collection of boards and corresponding URL's at the following link: https://github.com/arduino/Arduino/wiki/Unofficial-list-of-3rd-party-boards-support-urls
  (or on the website of the manufacturer) .
- Copy this URL.
- Go to FILE, PREFERENCES.
- Click on the Browse button next to ADDITIONAL BOARDS MANAGER URLS.
- Past the URL as a new line in this list and press OK.
- Now your board is listed in the board manager.

Misc. software

This section describes other applications like Processing to communicate with your computer and Fritzing to make electronic drawings.

28. Processing

Processing is a programming language, development environment targeted to visual arts. It can easily connect with your Arduino project through the Serial port (through USB). For example, you can use your Arduino to read the position of a joystick and send this to Processing to make a cube turn around.

28.1. Links for Processing

- Home page
  http://www.processing.org/
- Tutorials
  http://www.processing.org/tutorials/
  http://www.processing.org/examples/
- Reference
  http://www.processing.org/reference/
- Download for Windows 32/64 bit:
  http://download.processing.org/processing-2.1-windows32.zip
  http://download.processing.org/processing-2.1-windows64.zip
- Download for OS X:
  http://download.processing.org/processing-2.1-macosx.zip
- Download for Linux 32/64 bit:
  http://download.processing.org/processing-2.1-linux32.tgz
  http://download.processing.org/processing-2.1-linux64.tgz

28.2. List Serial Ports

When using Processing to communicate with an Arduino board, you need to determine the name of the serial port through which the Arduino is connected to the computer. The following Processing sketch can be used to show all available serial ports and all data coming in through one specific port.

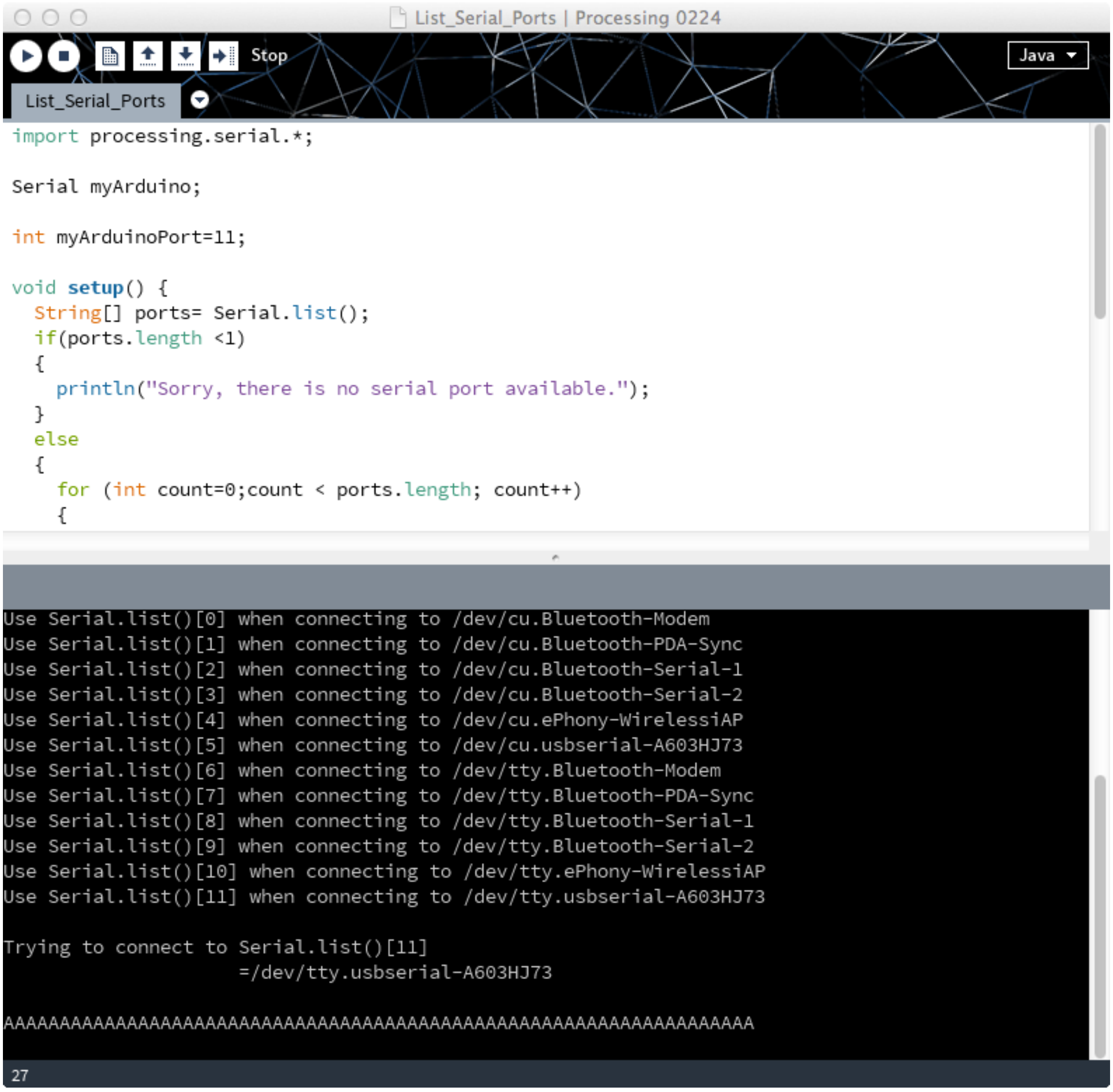
Processing sketch

```
import processing.serial.*;

Serial myArduino;

int myArduinoPort=11;

void setup() {
  String[] ports= Serial.list();
  if(ports.length <1)
  {
    println("Sorry, there is no serial port available.");
  }
  else
  {
    for (int count=0;count < ports.length; count++)
    {
      println("Use Serial.list()[" + count + "] when connecting to " +
ports[count]);
    }
  }
  // Load a test sketch in your Arduino to produce serial data,
  // match the baudrate
  // Put the portnumber you want to test in Serial.list()[..] and
  // watch the monitor below.
  println("\nTrying to connect to Serial.list()[" + myArduinoPort + "]");
  println("                    =" + ports[myArduinoPort] + " \n");
  myArduino = new Serial(this, Serial.list()[myArduinoPort], 9600);
}

void draw() {
  while (myArduino.available() > 0) {
    char inByte = char(myArduino.read());
    print(inByte);
  }
}
```

The following Arduino sketch prints the letter A to the serial port.

```
void setup() {
   Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.print("A");
  delay(500);
}
```

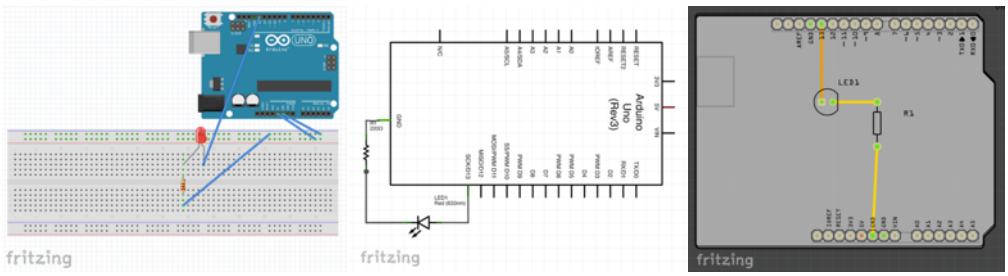
If you load both sketches (in Processing and Arduino) and if you've selected the right serial port, you will see the following output in Processing:



Pict. 2 Sample output from Processing

29. Fritzing

Fritzing is a tool you can use to create drawings of your Arduino projects. In Fritzing you can draw in three different environments: Breadboard, Schematic and PCB. You can switch between these three environments at will. You can use Fritzing to document your projects, check your design, route the connections on a breadboard or create a PCB (Printed Circuit Board) to build your own Arduino shield. Fritzing is not limited to Arduino.



Pict. 3 Blink-example (from left to right, Breadboard, Schematic and PCB design)

All drawings used in the Arduino tutorials at arduino.cc and all drawings in this document have been created by Fritzing.

In this document I refer to 0.8.5, available for Windows, OS X and Linux.

29.1. Links for Fritzing

- Home page:
  http://fritzing.org
- Tutorials:
  http://fritzing.org/learning/
- Reference:
  http://fritzing.org/learning/full_reference
- Download for Windows:
  http://fritzing.org/download/0.8.5b/windows/fritzing.2013.12.17.pc.zip
- Download for OS X:
  http://fritzing.org/download/0.8.5b/mac-os-x-105/fritzing.2013.12.17.cocoa.dmg
- Linux 32/64 bit:
  http://fritzing.org/download/0.8.5b/linux-32bit/fritzing-0.8.5b.linux.i386.tar.bz2
  http://fritzing.org/download/0.8.5b/linux-64bit/fritzing-0.8.5b.linux.AMD64.tar.bz2
- Custom parts:
  https://code.google.com/p/fritzing/issues/detail?id=875

Programming/ Programmers

There are different ways to program an Arduino board or another AVR MCU, using different techniques (programming) or equipment (programmers). In most situations you will program an Arduino board through the onboard USB connector, but some boards lack this connector. This chapter describes different programming techniques and different programmers.

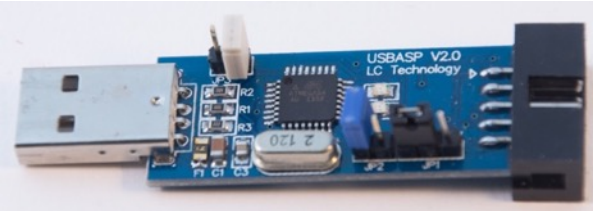
30. Programming an Arduino Board through USB

This is the default way to program a sketch to an Arduino Board.

Programming a sketch sample

- Connect your Arduino to your computer using a USB cable.
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the correct Arduino board.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD.

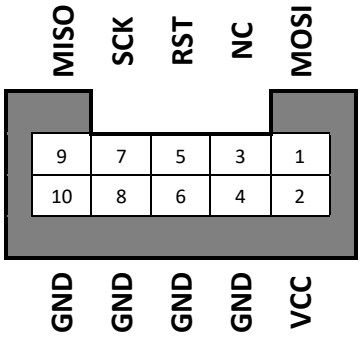
31. USBasp v2.0 programmer



31.1. Links USBasp v2.0 programmer

- Drivers Windows (none needed for Linux nor for Mac OSx):
  http://www.fischl.de/usbasp/usbasp-windriver.2011-05-28.zip
- Software AVRDUDE
  http://download.savannah.gnu.org/releases/avrdude/
- Documentation and firmware:
  http://www.fischl.de/usbasp/
- http://www.fischl.de/usbasp/Readme.txt

31.2. Connections USBasp v2.0 programmer



J1 Power, choose between 3.3 and 5V.
J2 Firmware upgrade (with another programmer).
J3 SCK, either 375 kHz or 8 kHz (in case target clock is lower than 1.5 MHz).

31.3. Sample Setup USBasp v2.0 programmer[1]

- Connect 9 MISO to D12.
- Connect 7 SCK to D13.
- Connect 5 RST to RESET
- Connect 1 MOSI to D11.
- Connect VCC to 5V.
- Connect GND to GND.

[1] This setup can be simplified by using an AVR ISP to ICSP adapter as described in the next chapter.

Programming a new bootloader sample

The following steps can be used to program a new bootloader to an Arduino UNO board. This is needed when the bootloader is corrupt, or when a destroyed Atmega328 has been replaced with a new one.

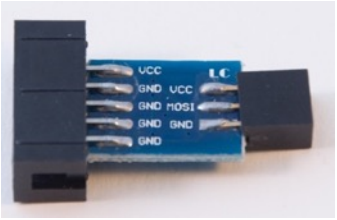
- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose TOOLS, BURN BOOTLOADER.

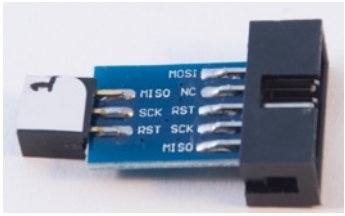
Programming a sketch sample

The USBasp can also be used to program a sketch to an Arduino Board.

- Load the sketch you want to compile and upload.
- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose FILE, UPLOAD USING PROGRAMMER.

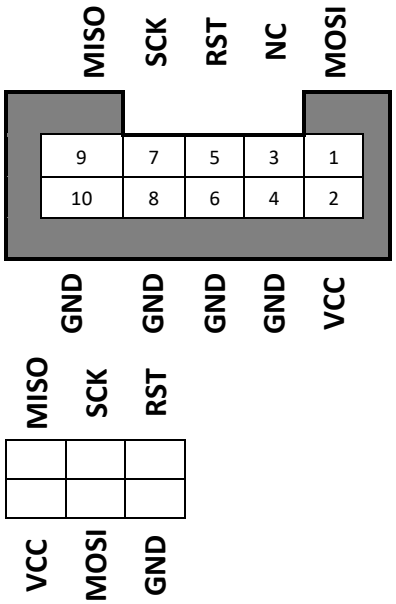
32. AVR ISP to ICSP Adapter





With this AVR ISP to ICSP adapter you can directly connect a USBasp programmer (see previous chapter) with the ICSP connector on an Arduino board.

32.1. Connections AVR ISP to ICSP Adapter



32.2. Sample Setup USBASP v2.0 programmer

The following setup is a simplified version of the setup of the previous chapter.

- Connect USBasp to a USB port.
- Connect AVR ISP to ICSP adapter with a flat cable to the USBasp.
- Connect other end of the AVR ISP to ICSP adapter to the ICSP headers on the Arduino UNO (with the GND pin facing the Atmega328).

Programming a new bootloader sample

The following steps can be used to program a new bootloader to an Arduino UNO board. This is needed when the bootloader is corrupt, or when a destroyed Atmega328 has been replaced with a new one.

- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose TOOLS, BURN BOOTLOADER.

Programming a sketch sample

The USBasp can also be used to program a sketch to an Arduino Board.

- Load the sketch you want to compile and upload.
- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose FILE, UPLOAD USING PROGRAMMER.

33. Uploading the bootloader by using an Arduino as ISP

Programming an Arduino is taken care of by a small routine called the bootloader. The bootloader is located on the ATMEGA328P chip on the Arduino. Sometimes it is necessary to upload a new bootloader to this chip. This chapter describes one way to do this.

You need two Arduino's and a 10 uF capacitor. One of these Arduino's must have a correct working bootloader. This working Arduino will be serving as an AVR ISP (in-system programmer).

More information can be found at https://www.arduino.cc/en/Tutorial/ArduinoISP

33.1. Sample Arduino as ISP

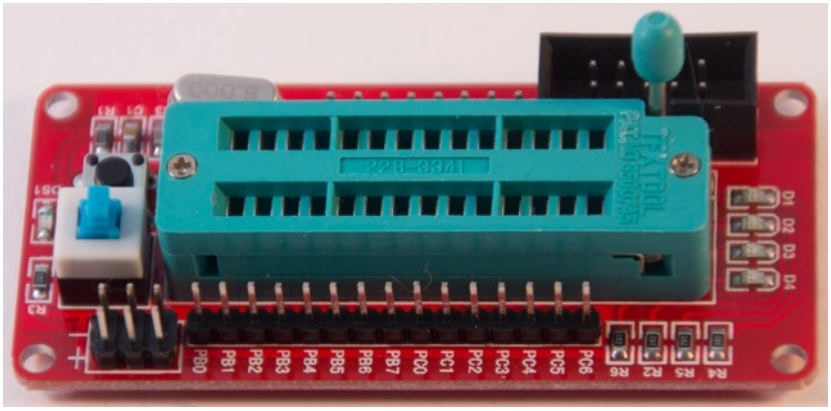
Follow the following steps:

- Connect the correct working Arduino to your computer.
- Open the ArduinoISP sketch (in examples).
- Connect your Arduino to your computer using a USB cable.
- Choose TOOLS, BOARD, then select the correct Arduino board.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port.
- Choose FILE, UPLOAD.
- At this point your correct working Arduino will serve as an AVR ISP.
- Make the following connections between your working and non-working Arduino.

Working Arduino	**Non-Working Arduino**
D13	D13
D12	D12
D11	D11
D10	RESET
5V	5V
GND	GND

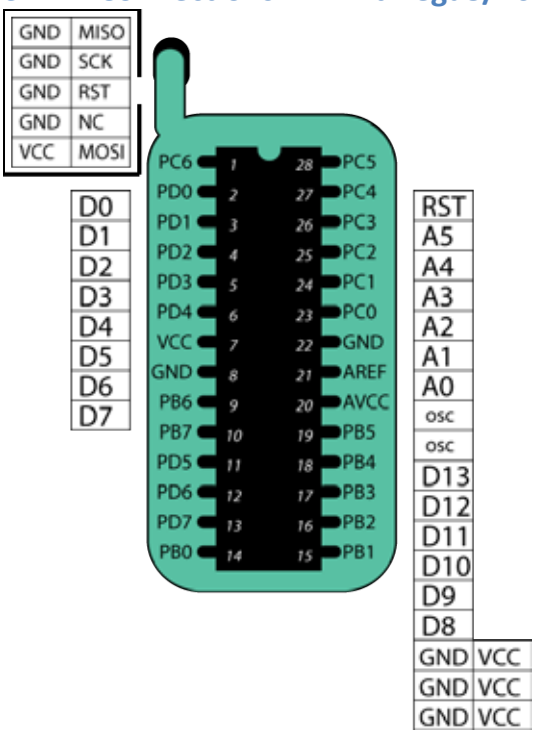
- On the correct working Arduino, connect a 10 uF capacitor between RESET AND GND (you could try it without).
- Keep the working Arduino connected to your computer.
- Choose TOOLS, BOARD, then select the NON-WORKING Arduino board.
- Choose TOOLS, PROGRAMMER, ARDUINO AS ISP
- Choose BURN BOOTLOADER.

34. AVR Atmega8/168/328 Development board



With this development board you can program Atmega8, Atmega168 and Atmega328 MCU's through the 10 pins ISP connector. You can also use it as an Arduino Board without an UART (i.e. without a serial connection through USB). Use the layout below to match the Atmega328 to Arduino pin assignment.

34.1. Connections AVR Atmega8/168/328 Development board



34.2. Sample setup AVR Atmega8/168/328 Development board

- Replace 8 MHz crystal with 16 MHz
- Place Atmega328 in ZIF-socket.
- Connect flat cable between USBasp and Development board.

Programming a new bootloader

The following steps can be used to program a new Arduino UNO bootloader to an Atmega328.

- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose TOOLS, BURN BOOTLOADER.

Programming a sketch sample

- Load the sketch you want to compile and upload.
- Choose TOOLS, PROGRAMMER, USBasp.
- Choose TOOLS, BOARD, ARDUINO UNO.
- Choose FILE, UPLOAD USING PROGRAMMER.

Sometimes it is needed to disconnect and reconnect the USBasp on your USB port, otherwise the Arduino IDE will end up with AVRdude errors.

35. Self-made Attiny 45/85 ISP adapter

A self-made Attiny45/85 – ISP adapter cable on a piece of breadboard.

35.1. Connections Self-made Attiny 45/85 ISP adapter

35.2. Sample Programming an Attiny45

Place the Attiny on a solder less breadboard, place the double row of header pins of the Attiny45/85 adapter on the solder less breadboard with pin 1 aligned with pin 1 of the Attiny. Connect the 10 pins female connector in the ISP connector of a USBasp[1].

- Open the Blink sketch from FILE/EXAMPLES (change pin 13 to 0, that is pin 5 on the Attiny).
- Choose TOOLS/BOARD/Attiny45 internal 8 MHz
- Choose TOOLS/PROGRAMMER/USBasp.
- Upload the Blink sketch to the Attiny, by choosing FILE/UPLOAD USING PROGRAMMER.
  Ignore the errors like this "***avrdude: please define PAGEL and BS2 signals in the configuration file for part Attiny***".
- Connect a LED and current limiting resistor of 220 ohm between pin 5 and GND (see chapter about LED).
  The LED should blink 1 second on, 1 second off.

[1] You might also use the ISP-ICSP adapter to connect the Attiny to the ICSP of an Arduino board (not tested yet). The Arduino should then be loaded with the ArduinoISP sketch.

36. ST Link V2

An ST Link V2 adapter is an in-circuit programmer/debugger for STM8/STM32. With Arduino IDE, you can use the ST Link V2 only for programming purposes. The Arduino IDE does not support the debugging facility.

ST Link V2.2 adapters are recognized as ST Link and as USB-Serial adapter, so you can use Serial Monitor to communication between your STM8/STM32 module and your computer. Too bad, most ST Link V2 adapters from Chinese web shops are V2.0/V2.1 and therefore don't support USB-serial.

Pin		Description	Connection on STM8/STM32
1	RST	Reset	n.c.
2	SWIM	Serial Wire Interface Module	n.c.
3	GND	Ground	Connect only 1 of the two GND pins to the STM32.
4	3.3V	3.3V output	Connect only 1 of the two 3.3V pins to the STM32.
5	5.0V	5.0V output	n.c.
6	5.0V	5.0V output	n.c.
7	3.3V	3.3V output	Connect only 1 of the two 3.3V pins to the STM32.
8	GND	Ground	Connect only 1 of the two GND pins to the STM32.
9	SWDIO	Bi-directional Serial Wire Data signal	(SW)DIO
10	SWCLK	Serial Wire Clock signal	(SW)CLK

36.1. Datasheet

Via the following link, you can download the documentation from ST Microelectronics:

- https://www.st.com/en/development-tools/st-link-v2.html
  The documents listed at this URL are mostly written for ST Microelectronics' own ST-LINK/V2 adapter, but a lot of information is also applicable for ST Link V2 adapters from other manufacturers.

36.2. Software/drivers

STSW-LINK004	STM32 ST-LINK utility	

https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stsw-link004.html		
STSW-LINK007 https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stsw-link007.html	ST-LINK, ST-LINK/V2, ST-LINK/V2-1 firmware upgrade	
STSW-LINK009	ST-LINK, ST-LINK/V2, ST-LINK/V2-1 USB driver signed for Windows7, Windows8, Windows 10	

Sound

This section describes the use of sound as an output device. A piezo speaker to produce some kind of musical notes and a buzzer to produce loud beeps.

37. Buzzer

A buzzer is a tiny speaker that can only produce a loud buzzing sound. Playing notes is not possible (use a piezo speaker instead).

37.1. Specifications Buzzer

- Voltage: 3.5-5.5V
- Frequency: 2300Hz

37.2. Connections Buzzer

Pin nr	Name	Description	Arduino pin
1	+	Signal	Any digital port
2	-	Ground	GND

37.3. Libraries needed for Buzzer

None needed.

37.4. Sample Buzzer

The following sketch plays the mayday signal S.O.S.

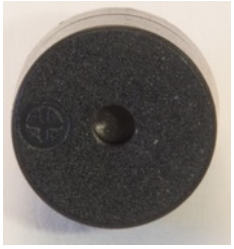
Sample Connections

- Connect + to D8.
- Connect - to GND.

004_Buzzer.ino

```
int speakerPin = 8;
int shortnote = 50;
int longnote = 200;
int interpunction = 200;

void playLetterO()
{
  for (int i = 1; i <=3; i++)
  {
    digitalWrite(speakerPin, HIGH);
    delay(longnote);
    digitalWrite(speakerPin, LOW);
    delay(longnote);
  }
  delay(interpunction);
}

void playLetterS()
{
  for (int i = 1; i <=3; i++)
  {
    digitalWrite(speakerPin, HIGH);
    delay(shortnote);
    digitalWrite(speakerPin, LOW);
    delay(shortnote);
  }
  delay(interpunction);
}

void setup()
{
  pinMode(speakerPin, OUTPUT);
}

void loop()
{
  playLetterS();
  playLetterO();
  playLetterS();
  delay(1000);
}
```

38. Piezo Speaker



A piezo speaker is a tiny speaker that is often used in toys.

38.1. Specifications Piezo Speaker

38.2. Connections Piezo Speaker

Pin nr	Name	Description	Arduino pin
1	+	Signal	Any PWM port
2	-	Ground	GND

38.3. Libraries needed for Piezo Speaker

None needed.

38.4. Sample Piezo Speaker

The following sketch plays a simple melody.

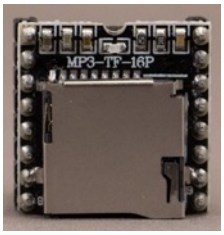
Sample Connections

- Connect + to D9.
- Connect - to GND.

005_PiezoSpeaker.ino

```
int speakerPin = 9;

int length = 15;
char notes[] = "ccggaagffeeddc ";
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration)
{
  for (long i = 0; i < duration * 1000L; i += tone * 2)
  {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration)
{
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
  for (int i = 0; i < 8; i++)
  {
    if (names[i] == note)
    {
      playTone(tones[i], duration);
    }
  }
}

void setup()
{
  pinMode(speakerPin, OUTPUT);
}

void loop()
{
  for (int i = 0; i < length; i++)
  {
    if (notes[i] == ' ')
    {
      delay(beats[i] * tempo);
    }
    else
    {
      playNote(notes[i], beats[i] * tempo);
    }
    delay(tempo / 2);
  }
}
```

39. Mini MP3 Player MP3-TF-16P



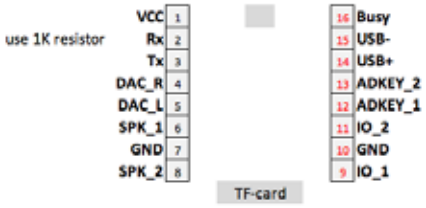
39.1. Specifications Mini MP3 Player

- Sampling rates (KHz): 8/11.025/12/16/22.05/24/32/44.1/48
- 24-bit DAC output
- dynamic range support: 90dB
- SNR support: 85dB
- File system: FAT16, FAT32 file system
- TF-SD card reader: Max 32G TF card
- Control modes:
  - IO control mode
  - serial mode
  - AD key control mode
- Audio data is sorted by folder, supports up to 100 folders, folders can assign every 255 Tracks
- 30 level adjustable volume, adjustable EQ 6

39.2. Datasheet Mini MP3 Player

- https://www.dfrobot.com/wiki/index.php/DFPlayer_Mini_SKU:DFR0299
- http://www.picaxe.com/docs/spe033.pdf
-

39.3. Connections Mini MP3 Player



Pin nr	Name	Description	Arduino pin
1	VCC	Input voltage	5V
2	Rx	Read data	Use a 1 Kohm resistor to connect to any Digital port
3	Tx	Transmit data	Any Digital port
4	DAC_R	• Audio output Right channel • Drive earphone and amplifier	
5	DAC_I	• Audio output Left channel Drive earphone and amplifier	
6	SPK_1	• Speaker – • Drive speaker less than 3W	
7	GND	Ground	
8	SPK_2	• Speaker + • Drive speaker less than 3W	
9	IO_1	• Trigger port 1 • Short press to play previous (long press to decrease volume)	
10	GND	Ground	
11	IO_2	• Trigger port 2 • Short press to play next (long press to increase volume)	
12	ADKEY_1	Trigger play first segment	
13	ADKEY_2	Trigger play fifth segment	
14	USB +	USB+ DP	
15	USB -	USB- DM	
16	Busy	• Playing status • Low: playing • High: not playing	

39.4. Libraries needed for Mini MP3 Player

- Michael C. Miller's (Makuna) DFMiniMP3 library through the library manager.

Library use explanation

```
#include <SoftwareSerial.h>
```

Include SoftwareSerial, so you can give commands to the MP3 module with self-chosen digital ports, leaving the hardware serial ports free for other purposes.

```
#include <DFMiniMp3.h>
```

Include Michael C. Miller's (Makuna) DFMiniMP3 library.

```
class Mp3Notify
{
public:
  static void OnError(uint16_t errorCode)
  {
  }
  static void OnPlayFinished(uint16_t globalTrack)
  {
  }
  static void OnCardOnline(uint16_t code)
  {
  }
  static void OnCardInserted(uint16_t code)
  {
  }
  static void OnCardRemoved(uint16_t code)
  {
  }
};
```

You need to define a Notify class to insert the actions that needs to be performed at the defined events (OnError, OnPlayFinished, OnCardOnline, OnCardInserted and OnCardRemoved).

```
SoftwareSerial MP3Serial(<TX>,<RX>, 11);
```

Creates a Software Serial port to the MP3 module, with <RX> the port to which the MP3 mini player's RX pin is connected (through an 1K ohm resistor) and <TX> the port to which the MP3 mini player's TX pin is connected.

```
DFMiniMp3<SoftwareSerial, Mp3Notify> mp3(MP3Serial);
```

Makes an instance of the DFMiniMp3 class named mp3. The parameters used state the SERIAL_METHOD either SoftwareSerial or HardwareSerial, the name of the Notify class defined above and the name of the Serial port to be used either Serial, Serial1, or the name of the SoftwareSerial.

```
mp3.begin();
```

Initialize mp3.

```
mp3.setVolume(24);
```

Set the speaker volume to 24 (from 0..30).

```
mp3.getTotalTrackCount();
```

Counts the number of available mp3 files in the folder sd:/mp3. The names of the files should be 0001<string>.mp3 to 9999<string>.mp3.

```
void waitMilliseconds(uint16_t msWait)
{
  uint32_t start = millis();
  while ((millis() - start) < msWait)
  {
    mp3.loop();
    delay(1);
  }
}
```

This replacement for delay(<time>) checks performs the function mp3.loop() to check whether one of the events from the Notify class has occurred.

```
mp3.playMp3FolderTrack(1);
```

Plays the first file (0001<string>.mp3) from the folder sd:/mp3.

The API reference for this library can be found at:
https://github.com/Makuna/DFMiniMp3/wiki/API-Reference

39.5. Sample Mini MP3 Player

The following sketch plays the first 2 files (0001<string>.mp3 and 0002<string>.mp3) from the sd:/mp3 folder.

Sample Connections

- Connect VCC to 5V
- Connect GND to GND
- Connect SPK_1 to a speaker (-)
- Connect SPK_2 to a speaker (+)
- Connect Rx to one leg of a 1K ohm resistor
- Connect the other leg of the 1K ohm resistor to D10
- Connect Tx to D11

Sample Sketch

```
#include <SoftwareSerial.h>
#include <DFMiniMp3.h>
//If you are using OSx to copy the mp3-files, use the following command to
remove hidden files before playing:
// dot_clean /Volumes/<SDVolumeName>

class Mp3Notify
{
public:
  static void OnError(uint16_t errorCode)
  {
    Serial.print("Com Error ");
    Serial.println(errorCode);
  }

  static void OnPlayFinished(uint16_t globalTrack)
  {
    Serial.print("Play finished for #");
    Serial.println(globalTrack);
  }

  static void OnCardOnline(uint16_t code)
  {
    Serial.print("Card online ");
    Serial.println(code);
  }

  static void OnCardInserted(uint16_t code)
  {
    Serial.print("Card inserted ");
    Serial.println(code);
  }

  static void OnCardRemoved(uint16_t code)
  {
    Serial.print("Card removed ");
    Serial.println(code);
  }
};

SoftwareSerial MP3Serial(10, 11);
DFMiniMp3<SoftwareSerial, Mp3Notify> mp3(MP3Serial);

void setup()
{
```

```
  Serial.begin(115200);
  Serial.println("initializing...");
  mp3.begin();
  uint16_t volume=24;
  mp3.setVolume(volume);
  uint16_t count = mp3.getTotalTrackCount();
  Serial.print("files ");
  Serial.println(count);
  Serial.println("starting...");
}

void waitMilliseconds(uint16_t msWait)
{
  uint32_t start = millis();
  while ((millis() - start) < msWait)
  {
    mp3.loop();
    delay(1);
  }
}

void loop()
{
  Serial.println("track 1");
  mp3.playMp3FolderTrack(1);
  waitMilliseconds(5000);
  Serial.println("track 2");
  mp3.playMp3FolderTrack(2);
  waitMilliseconds(5000);
}
```

LED displays

In this section you will find several components using LED's, RGB LED's, an 8x8 LED matrix and 7 Segment Digits.

40. Onboard LED D13

40.1. Specifications Onboard LED D13

- Onboard, connected to D13.

40.2. Connections Onboard LED D13

No connections needed, since this led is already soldered on the Arduino board and connected to D13.

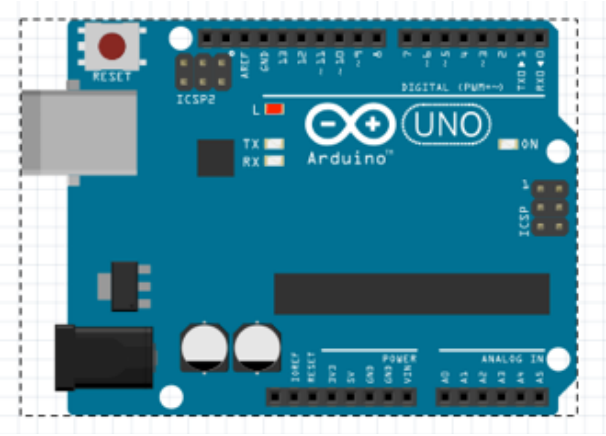
40.3. Libraries needed for Onboard LED D13

None needed.

40.4. Sample Onboard LED D13

Sample Connections

- No connections needed.



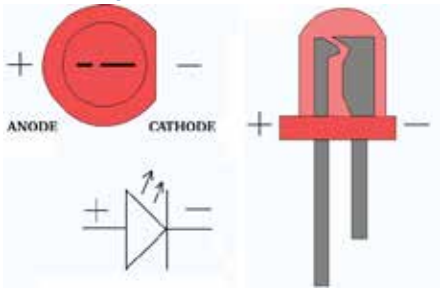
006_LEDOnboard.ino

```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

41. LED

LED comes from Light Emitting Diode. A Diode is a device that will only allow current in one direction. If this current runs through a LED then it will emit light.

41.1. Specifications LED



Most LED have a maximum allowed Voltage of 1.8-2.0 V and a maximum allowed Current of 20 mA. The output Voltage of an Arduino is 5V and this would fry your LED. To prevent your LED's from frying, a limiting resistor is needed. In most case a resistor of 220 ohm should do the trick.

One side of a LED is called the Anode and should be connected to a Digital output through a limiting resistor of 220 ohm[1].

The other side is called the Cathode and should be connected to GND.

There are several ways you can distinguish the Anode from the Cathode:

- The Anode (+) has the longest leg.
- The Cathode (-) has a flat edge.
- Inside the LED the Cathode is connected to something that looks like a cup.
- Last resort, connect the LED and try it out. If the LED won't lid, you have to turn it around!

[1] With an average I_r= 20 mA and V_r= 1.8 V, this value can be calculated with the calculator on the following website:
http://led.linear1.org/1led.wiz

41.2. Connections LED

Pin nr	Name	Description	Arduino pin
1	Cathode	shortest leg flat edge	GND
2	Anode	longest leg rounded edge	Any Digital port through 220 ohm resistor

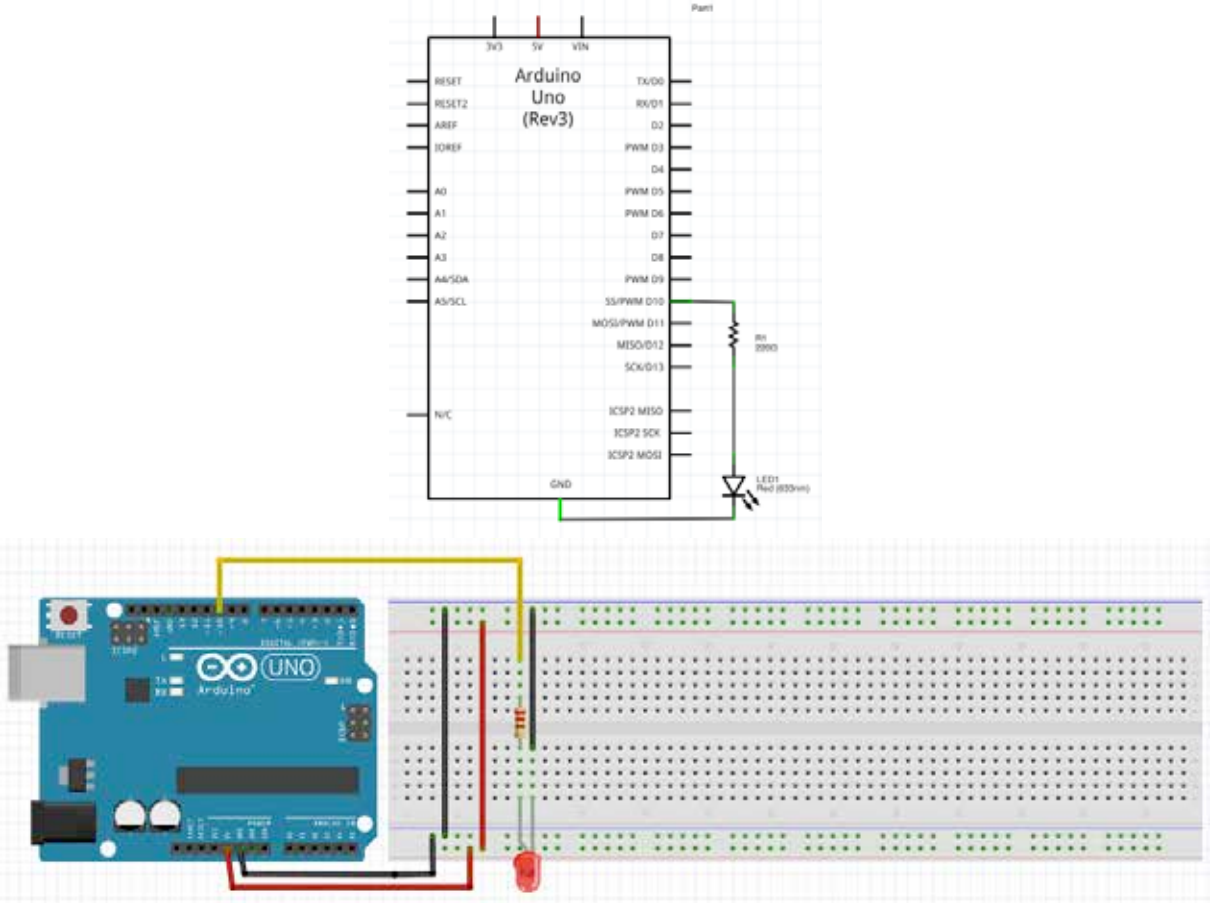
41.3. Libraries needed for LED

None needed.

41.4. Sample LED

Sample Connections

- Connect Cathode with GND.
- Connect Anode with one end of a Resistor of 220 Ohm.
- Connect the other end of the Resistor with D10.



Sample Sketch using digitalWrite()

In this sketch the LED is blinking. The LED is either On or OFF

007_LED.ino

```
int led = 10;

void setup()
{
  pinMode(led, OUTPUT);
}

void loop()
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

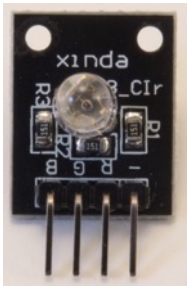
Sample Sketch using analogWrite() (PWM)

In this sketch the LED is fading on and OFF, by using analogWrite() on one of the PWM ports.

008_LEDPWM.ino

```
int led = 10;
int brightness = 0;
int fadeAmount = 5;

void setup()
{
  pinMode(led, OUTPUT);
}

void loop()
{
  analogWrite(led, brightness);
  brightness = brightness + fadeAmount;
  if (brightness == 0 || brightness == 255)
  {
    fadeAmount = -fadeAmount;
  }
  delay(10);
}
```

42. RGB LED board



An RGB LED is like 3 LEDS in one, RED, GREEN and BLUE. By mixing these three colors you can also create colors in between like YELLOW, MAGENTA and CYAN.

42.1. Specifications RGB LED board

This RGB LED board from Xinda contains 1 RGB LED and three current limiting resistors so it can be connected directly to the output ports of your Arduino. It has a common Cathode.

42.2. Connections RGB LED board

I found two variantions of the pins, the order is different, but the pin names are the same (R, G, B and -/GND)

Pin nr	Name	Description	Arduino pin
1	B	Blue	Any PWM port
2	G	Green	Any PWM port
3	R	Red	Any PWM port
4	-/GND	Ground	GND

Pin nr	Name	Description	Arduino pin
1	R	Red	Any PWM port
2	G	Green	Any PWM port
3	B	Blue	Any PWM port
4	-/GND	Ground	GND

42.3. Libraries needed for RGB LED board

None needed.

42.4. Sample RGB LED board

The following sketch changes the color of the RGB led continuously.

Sample Connections

- Connect B to D9.
- Connect G to D10.
- Connect R to D11.
- Connect GND to GND.

009_LEDRGB.ino

```
int RGBred = 11;
int RGBgreen = 10;
int RGBblue = 9;

void setup()
{
  pinMode(RGBred, OUTPUT);
  pinMode(RGBgreen, OUTPUT);
  pinMode(RGBblue, OUTPUT);
}

void loop()
{
  for (int redvalue=0;redvalue <=255;redvalue=redvalue+255)
  {
    for (int greenvalue=0;greenvalue <=255;greenvalue=greenvalue+255)
    {
        for (int bluevalue=0;bluevalue <=255;bluevalue=bluevalue+255)
        {
          RGB(redvalue,greenvalue,bluevalue);
          delay(500);
        }
    }
  }
}

void RGB(int red, int green, int blue)
{
  analogWrite(RGBred, red);
  analogWrite(RGBgreen, green);
  analogWrite(RGBblue, blue);
}
```

43. 10 segment LED Bar Graph F2510BH

43.1. Specifications 10 segment LED Bar Graph F2510BH

- Dimensions: 25.4x10.1x7.9 mm
- Color: Red
- 57 mW
- Peak current: 80 mA
- Voltage: 1.8V
- Forward current: 20mA
- A separate Anode pin per LED
- A separate Cathode pin per LED

43.2. Datasheet 10 segment LED Bar Graph F2510BH

I wasn't able to find the datasheet to this specific LED Bar Graph, but all needed information is listed at the specs in the previous paragraph.

43.3. Connections 10 segment LED Bar Graph F2510BH

The side with the type number F2510DH printed on it, is the anode side. This side can also be recognized by the angled corner. This angled corner indicates pin 1 (all pins on this side are anodes).

Pin nr	Name	Description	Arduino pin
1 – 10	Anode LED 1-10	A separate input pin per LED	Any digital port
11 - 20	Cathode LED 1-10	A separate ground pin per LED	Ground through an individual 220 ohm resistor per LED

43.4. Libraries needed for 10 segment LED Bar Graph F2510BH

There are no libraries needed for this LED Bar Graph

43.5. Sample 10 segment LED Bar Graph F2510BH

This sketch uses 10 digital outputs on your Arduino, 1 for every segment[1].

The following sketch will blink all LED's briefly.

Sample Connections

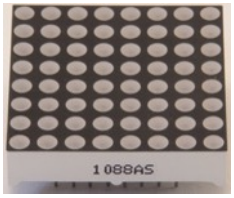
- Connect VCC to 5V.
- Connect GND to GND.
- Connect all Cathodes to GND through an individual 220 ohm resistor.
- Connect Anodes LED1..LED10 to D2..D11

131_BarGraph.ino

```
void setup() {
  for (int LED = 1; LED <= 10; LED++)
  {
    pinMode(LED + 1, OUTPUT);
  }
}

void loop() {
  for (int LED = 1; LED <= 10; LED++)
  {
    digitalWrite(LED + 1, HIGH);
    delay(100);
    digitalWrite(LED + 1, LOW);
    delay(100);
  }
}
```

[1] If you want to drive more LED's individually, you'll need some extra IC's, MAX7219 "53 MAX7219 LED driver" or 2 shift-registers (74HC595) and some transistors to deliver more current.

44. 8x8 DOT Matrix 1088AS



This 8x8 DOT Matrix is also used in chapter ”47 8x8 DOT matrix on PCB with MAX7219 chip”.

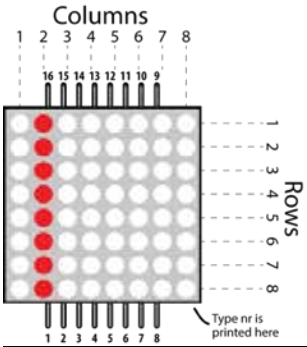
44.1. Specifications 8x8 DOT Matrix 1088AS

- 64 RED LED’s 3mm
- Dimensions: 32x32x8mm
- V_f=2.1, I_f=20 mA
- Row Cathode
- Column Anode

44.2. Datasheet 8x8 DOT Matrix 1088AS

- http://megtestesules.info/hobbielektronika/adatlapok/LED8x8_1088AS.pdf

44.3. Connections 8x8 DOT Matrix 1088AS



Pin nr	Row	Column
1	Cathode Row 5	
2	Cathode Row 7	
3		Anode Row 2
4		Anode Row 3
5	Cathode Row 8	
6		Anode Row 5
7	Cathode Row 6	
8	Cathode Row 3	
9	Cathode Row 1	
10		Anode Row 4
11		Anode Row 6
12	Cathode Row 4	
13		Anode Row 1
14	Cathode Row 2	
15		Anode Row 7
16		Anode Row 8

44.4. Libraries needed for 8x8 DOT Matrix 1088AS

None needed.

44.5. Sample 8x8 DOT Matrix 1088AS

To drive every LED in this matrix individually without using any chips, you will need 16 digital ports. This is only possible with the Arduino Mega. Using only 8 digital ports, this ketch will drive a whole column at a time and cycle through the 8 columns.[1]

Sample Connections

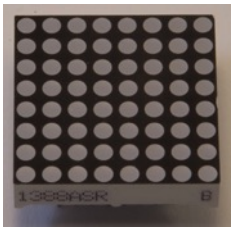
- Connect 1, 2, 5, 7, 8, 9, 12 and 14 to one end of a 220 ohm resistor each.
- Connect the other ends of the 8 resistors to GND.
- Connect 13 (Row 1) to D6
- Connect 3 (Row 2) to D7
- Connect 4 (Row 3) to D8
- Connect 10 (Row 4) to D9
- Connect 6 (Row 5) to D10
- Connect 11 (Row 6) to D11
- Connect 15 (Row 7) to D12
- Connect 16 (Row 8) to D13

010_LED8x8_1088AS.ino

```
void setup()
{
  for (int rownr=6;rownr<=13;rownr++)
  {
    pinMode(rownr,OUTPUT);
    digitalWrite(rownr,LOW);
  }
}

void loop()
{
  for (int rownr=6;rownr<=13;rownr++)
  {
    digitalWrite(rownr,HIGH);
    delay(200);
    digitalWrite(rownr,LOW);
  }
}
```

[1] If you want to drive every LED individually, you'll need some extra IC's, MAX7219 "53 MAX7219 LED driver" or 2 shift-registers (74HC595) and some transistors to deliver more current.

45. 8x8 DOT Matrix 1388ASR



In this DOT matrix the rows are connected to the Anodes and the columns to the Cathodes, so this DOT matrix cannot (easily) replace the DOT matrix on the board that was described in ”47 8x8 DOT matrix on PCB with MAX7219 chip”.

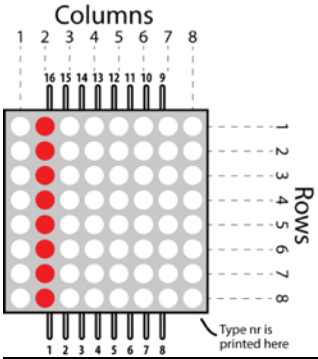
45.1. Specifications 8x8 DOT Matrix 1388ASR

- 64 RED LED’s 3mm.
- Dimensions: 32x32mm
- Row Anode
- Column Cathode
- V_f=1.8, I_f=20 mA

45.2. Datasheet 8x8 DOT Matrix 1388ASR

http://www.alfacomponent.com/r_rayconn/index_2.files/PDF/MATRIX/REC-M1388ASR.pdf

45.3. Connections 8x8 DOT Matrix 1388ASR



Pin nr	Row	Column
1	Anode Row 5	
2	Anode Row 7	
3		Cathode Column 2
4		Cathode Column 3
5	Anode Row 8	
6		Cathode Column 5
7	Anode Row 6	
8	Anode Row 3	
9	Anode Row 1	
10		Cathode Column 4
11		Cathode Column 6
12	Anode Row 4	
13		Cathode Column 1
14	Anode Row 2	
15		Cathode Column 7

Pin nr	Row	Column
16		Cathode Column 8

45.4. Libraries needed for 8x8 DOT Matrix 1388ASR

None needed.

45.5. Sample 8x8 DOT Matrix 1388ASR

To drive every LED in this matrix individually without using any chips, you will need 16 digital ports. This is only possible with the Arduino Mega. Using only 8 digital ports, this ketch will drive a whole column at a time and cycle through the 8 columns.[1]

Sample Connections

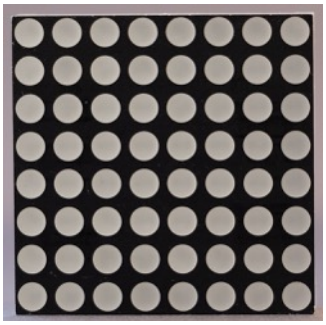
- Connect 3, 4, 6, 10, 11, 13, 15 and 16 to one end of a 220 ohm resistor each.
- Connect the other ends of the 8 resistors to GND.
- Connect 9 (Row 1) to D6
- Connect 14 (Row 2) to D7
- Connect 8 (Row 3) to D8
- Connect 12 (Row 4) to D9
- Connect 1 (Row 5) to D10
- Connect 7 (Row 6) to D11
- Connect 2 (Row 7) to D12
- Connect 5 (Row 8) to D13

011_LED8x8_1388ASR.ino

```
void setup()
{
  for (int rownr=6;rownr<=13;rownr++)
  {
    pinMode(rownr,OUTPUT);
    digitalWrite(rownr,LOW);
  }
}

void loop()
{
  for (int rownr=6;rownr<=13;rownr++)
  {
    digitalWrite(rownr,HIGH);
    delay(200);
    digitalWrite(rownr,LOW);
  }
}
```

[1] If you want to drive every LED individually, you'll need some extra IC's, MAX7219 "53 MAX7219 LED driver" or 2 shift-registers (74HC595) and some transistors to deliver more current.

46. 8x8 DOT Matrix HL-M1588BR



In this DOT matrix the rows are connected to the Anodes and the columns to the Cathodes, so this DOT matrix cannot (easily) replace the DOT matrix on the board that was described in ”47 8x8 DOT matrix on PCB with MAX7219 chip”.

46.1. Specifications 8x8 DOT Matrix HL-M1588BR

- 64 RED LED’s 3.75mm.
- Dimensions: 37.8x37.8mm
- Row Anode
- Column Cathode
- V_f=2
- I_f=30 mA.

46.2. Datasheet 8x8 DOT Matrix HL-M1588BR

- http://www.hlvled.com/images/up_images/HL-M1588BR.pdf

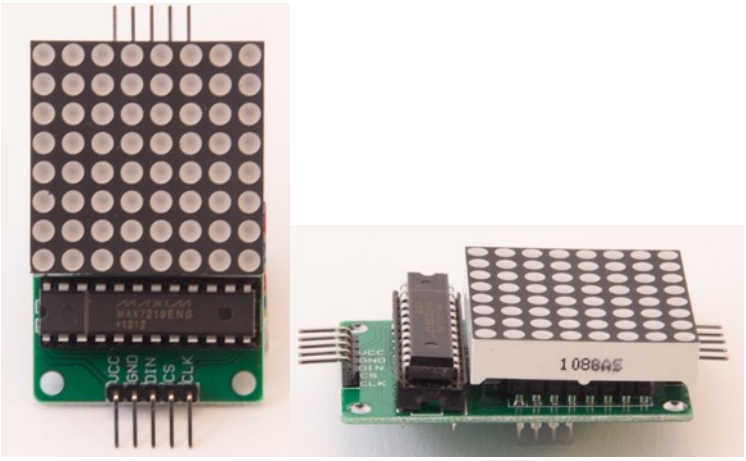
46.3. Connections 8x8 DOT Matrix HL-M1588BR

The connections for this matrix are the same as for the “45 8x8 DOT Matrix 1388ASR”.

46.4. Sample 8x8 DOT Matrix HL-M1588BR

Since this 8x8 matrix is pin compatible with the 1388ASR, you can check “45 8x8 DOT Matrix 1388ASR” for a sample sketch.

47. 8x8 DOT matrix on PCB with MAX7219 chip



This 8x8 DOT matrix can be used as a simple display, or combined with others to build a larger display. The single LED's in these single/multiple DOT Matrix displays can be individually addressed by a serial protocol, thus minimizing the number of output pins needed in your project. Beside the use of the 5 V and GND pin you only need 3 digital output pins, instead of one for every columns or row..

47.1. Specifications 8x8 DOT Matrix on PCB with MAX7219 chip

- MAX7219 LED display driver from Maxim
- 1088AS 8x8 DOT matrix.
- 5 pin header row for input from MCU or previous module.
- 5 pin header row for output to next module,
- Serially addressed through Serial Peripheral Interface: SPI (http://arduino.cc/en/Reference/SPI).

47.2. Datasheet MAX7219: Serially Interfaced, 8-Digit LED Display Drivers

http://pdf1.alldatasheet.com/datasheet-pdf/view/73745/MAXIM/MAX7219.html

47.3. Connections 8x8 DOT Matrix on PCB with MAX7219 chip

After purchase, the components need to be soldered on the labeled side. Printed label on side of the LED Matrix should be placed opposite to the notch on the MAX7219 chip (see picture above). Multiple modules can be daisy chained by connecting the

5 pin header at bottom (input)

Pin nr	Name	Description	Arduino pin
1	VCC	5 V	5 V
2	GND	Ground	GND
3	DIN	Data in	MOSI (D11)
4	CS	Chip Select	Any Digital port
5	CLK	Clock	SCK (D13)

5 pin header at top (output to other DOT matrix modules)

Pin nr	Name	Description	Next module
1	VCC	5 V	VCC
2	GND	Ground	GND
3	DOUT	Data out	DIN
4	CS	Chip Select	DS
5	CLK	Clock	CLK

47.4. Libraries needed for 8x8 DOT Matrix on PCB with MAX7219 chip

I've been using the following 3 libraries[1]

- SPI (Serial Peripheral Interface) library through the Library Manager.
- Max72xxPanel to address the Max7219 LED display driver. Created by Mark Ruys (mark@paracas.nl).
  https://github.com/markruys/arduino-Max72xxPanel
- Adafruit GFX library through the Library Manager. This library is used for displaying graphics on all sorts of displays.
  More information on the use of the Adafruit_GFX library can be find at the following URL: http://learn.adafruit.com/adafruit-gfx-graphics-library?view=all

Library use explanation

```
#include <SPI.h>
```
Include the Serial Peripheral Interface included in the Arduino IDE.

```
#include <Adafruit_GFX.h>
```
Include the display library from Adafruit.

```
#include <Max72xxPanel.h>
```
Include the MAX 72xxPanel library from Mark Ruys (mark@paracas.nl).

```
Max72xxPanel mymatrix = Max72xxPanel(CS, 1, 1);
```
Create 'mymatrix' a new instance of the object type Max72xxPanel. CS is an Integer value corresponding to the Arduino Digital Output to which CS pin is connected.

```
mymatrix.fillScreen(LOW);
```
Turn off all pixel in mymatrix's buffer.

```
mymatrix.write();
```
Showing content of mymatrix's buffer on the DOT matrix. In this case, all LED's are turned off. This function should be called every time a change in mymatrix's buffer needs to be showed in the DOT Matrix.

```
mymatrix.drawPixel(4, 4, HIGH);
```
Turn on pixel 4, 4 in mymatrix's buffer.

```
mymatrix.write();
```
Showing content of mymatrix's buffer on the DOT matrix. In this case, LED 4,4 is turned on.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

[1] An alternative library can be installed through the Library Manager: LedControl by Eberhard Fahle. More freedom in digital ports, but no support for GFX.

47.5. Sample 8x8 DOT Matrix on PCB with MAX7219 chip

This sample sketch shows a dot traveling from row to row and from column to column.

Sample Connections

- Connect VCC with 5V.
- Connect GND with GND.
- Connect DIN with MOSI (D11).
- Connect CS with D10.
- Connect CLK with SCK (D13).

012_LED8x8_MAX7219.ino

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;

Max72xxPanel matrix = Max72xxPanel(pinCS, 1, 1);

int wait = 100; // In milliseconds

void setup()
{
  matrix.fillScreen(LOW);
  matrix.write();
}

void loop()
{
  for (int county=0;county<matrix.height();county++)
  {
    for (int countx=0;countx<matrix.width();countx++)
    {
      matrix.drawPixel(countx, county, HIGH);
      matrix.write();
      delay(wait);
      matrix.drawPixel(countx, county, LOW);
      matrix.write();
      delay(wait);
    }
  }
}
```

48. Single Digit 7-Segment Display

This Single Digit display has 7 segments and 1 decimal point.

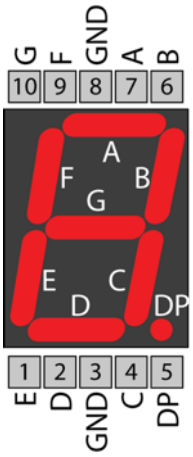
48.1. Specifications Single Digit 7-Segment Display

- 1 Digit with 7 Segments A..G and 1 Decimal Point.
- Common cathode (2 pins).
- 8 Digital pins needed.
- 8 limiting resistors needed (1 per Segment and 1 for the decimal point).

48.2. Datasheet HS120561K Single Digit 7-Segment Display

http://www.aplomb.nl/Niels_skn/7-Segment_QRD%20Niels.pdf, page 13.

48.3. Connections Single Digit 7-Segment Display



Pin nr	Name	Description	Arduino pin
1	E	E-segment	D6 through a 220 ohm resistor
2	D	D-segment	D5 through a 220 ohm resistor
3	Cathode	common cathode	GND
4	C	C-segment	D4 through a 220 ohm resistor
5	DP	Decimal Point	D9 through a 220 ohm resistor
6	B	B-segment	D3 through a 220 ohm resistor
7	A	A-segment	D2 through a 220 ohm resistor
8	Cathode	common cathode	GND
9	F	F-segment	D7 through a 220 ohm resistor
10	G	G-segment	D8 through a 220 ohm resistor

48.4. Libraries needed for Single Digit 7-Segment Display

- None needed.

Explanation

The following table shows which segment needs to be switched on for every number.

Nr	A	B	C	D	E	F	G
0	ON	ON	ON	ON	ON	ON	OFF
1	OFF	ON	ON	OFF	OFF	OFF	OFF
2	ON	ON	OFF	ON	ON	OFF	ON
3	ON	ON	ON	ON	OFF	OFF	ON
4	OFF	ON	ON	OFF	OFF	ON	ON
5	ON	OFF	ON	ON	OFF	ON	ON
6	ON	OFF	ON	ON	ON	ON	ON
7	ON	ON	ON	OFF	OFF	OFF	OFF
8	ON	ON	ON	ON	ON	ON	ON
9	ON	ON	ON	OFF	OFF	ON	ON

http://www.hacktronics.com/Tutorials/arduino-and-7-segment-led.html

48.5. Sample Single Digit 7-Segment Display

This sketch uses 8 digital outputs on your Arduino, 1 for every segment[1].

The following sketch counts down from 9 to 0 repeatedly.

Sample Connections

- Connect one end of a 220 ohm resistor to pin 1 (E) and the other end to D6.
- Connect one end of a 220 ohm resistor to pin 2 (D) and the other end to D5.
- Connect 3 (Cathode) to GND
- Connect one end of a 220 ohm resistor to pin 4 (C) and the other end to D4.
- Connect one end of a 220 ohm resistor to pin 5 (CP) and the other end to D9.
- Connect one end of a 220 ohm resistor to pin 6 (B) and the other end to D3.
- Connect one end of a 220 ohm resistor to pin 7 (A) and the other end to D2.
- Connect 8 (Cathode) to GND
- Connect one end of a 220 ohm resistor to pin 9 (F) and the other end to D7.
- Connect one end of a 220 ohm resistor to pin 10 (G) and the other end to D8.

013_7SegSingle.ino

```
byte seven_seg_digits[10][7] =
{
  { 1,1,1,1,1,1,0 },  // = 0
  { 0,1,1,0,0,0,0 },  // = 1
  { 1,1,0,1,1,0,1 },  // = 2
  { 1,1,1,1,0,0,1 },  // = 3
  { 0,1,1,0,0,1,1 },  // = 4
  { 1,0,1,1,0,1,1 },  // = 5
  { 1,0,1,1,1,1,1 },  // = 6
  { 1,1,1,0,0,0,0 },  // = 7
  { 1,1,1,1,1,1,1 },  // = 8
  { 1,1,1,0,0,1,1 }   // = 9
};

void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  writeDot(1);
}

void writeDot(byte dot)
{
  digitalWrite(9, dot);
}

void sevenSegWrite(byte digit)
{
  byte pin = 2;
```

[1] If you want to drive more LED's individually, you'll need some extra IC's, MAX7219 "53 MAX7219 LED driver" or 2 shift-registers (74HC595) and some transistors to deliver more current.

```
  for (byte segCount = 0;segCount < 7;++segCount)
  {
    digitalWrite(pin, seven_seg_digits[digit][segCount]);
    ++pin;
  }
}

oid loop()
{
  for (byte count = 10;count > 0;--count)
  {
    delay(1000);
    sevenSegWrite(count - 1);
  }
}
```

49. 4 Digits 7-Segment Display Common Cathode (HS420561K-32)

This 4 Digits display has 7 segments and 1 decimal point per digit. It differs with the 4 Digits 7-Segment display from the next chapter in the fact that this display has an common cathode.

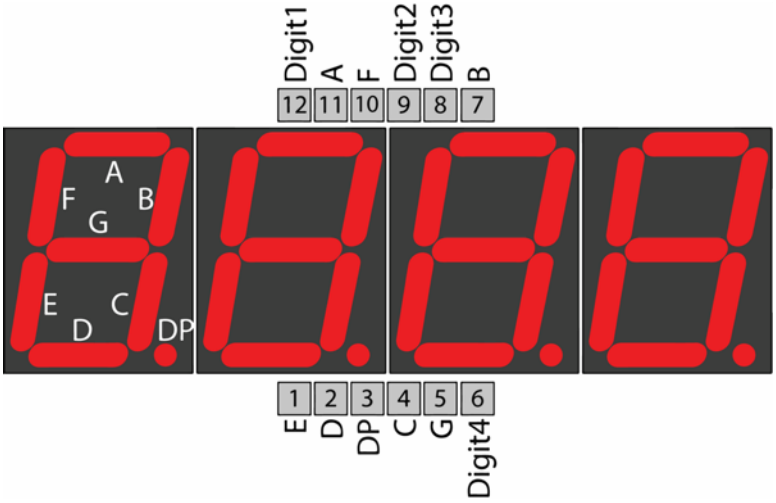
49.1. Specifications 4 Digits 7-Segment Display Common Cathode (HS420561K-32)

- 4 digits, with 7 Segments A..G and 1 Decimal Point per digit.
- Common cathode.
- 1 Select pin per digit.
- 4 limiting resistors needed, 1 for every select pin.
- Total of 12 Digital pins needed.
- HS420561K-

49.2. Datasheet HS420561K Single Digit 7-Segment Display Common Cathode (HS420561K-32)

http://www.aplomb.nl/Niels_skn/7-Segment_QRD%20Niels.pdf, page 35.

49.3. Connections 4 Digits 7-Segment Display Common Cathode (HS420561K-32)



Pin nr	Name	Description	Arduino pin
1	E	Segment E	D10
2	D	Segment D	D9
3	DP	Decimal Point	D13
4	C	Segment C	D8
5	G	Segment G	D12
6	Digit4	Common cathode for Digit 4	D5 through a 220 ohm resistor
7	B	Segment B	D7
8	Digit3	Common cathode for Digit 3	D4 through a 220 ohm resistor
9	Digit2	Common cathode for Digit 2	D3 through a 220 ohm resistor
10	F	Segment F	D11
11	A	Segment A	D6
12	Digit1	Common cathode for Digit 1	D2 through a 220 ohm resistor

Not all 4 digits 7-segment displays have the same pinout. You can check yours as follows:

- *Place a 220 ohm resistor between Digit 1 (pin 12).*
- *Connect a jumper wire to 5V.*
- *Connect the other end of the jumper wire with any of the segment pins (1, 2, 3, 4, 5, 7, 10 and 11) and check if the corresponding segment of Digit 1 will lit.*
- *Repeat this for the other digits by moving the resistor to Digit 2 (pin 9), Digit 3 (pin 8) and Digit 4 (pin 6).*

49.4. Libraries needed for 4 Digits 7-Segment Display Common Cathode (HS420561K-32)

- Seven Segment Display Library from Dean Reading deanreading@hotmail.com. https://github.com/DeanIsMe/SevSeg.

Library use explanation

```
#include "SevSeg.h"
```

Include the Seven Segment library from Dean Reading.

```
SevSeg mysevseg;
```

Create a new instance of the object SevSeg.

```
mysevseg.Begin(CA,D1pin,D2pin,D3pin,D4pin,A,B,C,D,E,F,G,DP);
```

CA is a Boolean: 1 for common Anode, 0 for common Cathode.
D1pin..DP are integer values for the digital pins to which D1pin..DP are connected to (probably 2..13).

```
mysevseg.Brightness(100);
```

Set brightness to 100 (0-100).

```
mysevseg.PrintOutput();
```

.PrintOutput() must be called repeatedly to get the number displayed (refresh??).

```
mysevseg.NewNum(CentSec,(byte) 2);
```

.NewNum displays the number on your display.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

49.5. Sample 4 Digits 7-Segment Display Common Cathode (HS420561K-32)

This sketch uses 12 digital outputs on your Arduino, 8 for the segments and 4 for the digits[1].

The following sketch counts from 000.0 to 999.9 seconds repeatedly.

Sample Connections

- Connect pin 1 (E) to D2.
- Connect pin 2 (D) to D3.
- Connect pin 3 (DP) to D4.
- Connect pin 4 (C) to D5.
- Connect pin 5 (G) to D6.
- Connect one end of a 220 ohm resistor to pin 6 (Digit 4) and the other end to D7.
- Connect pin 7 (B) to D8.
- Connect one end of a 220 ohm resistor to pin 8 (Digit 3) and the other end to D9.
- Connect one end of a 220 ohm resistor to pin 9 (Digit 2) and the other end to D10.
- Connect pin 10 (F) to D11.
- Connect pin 11 (A) to D12.
- Connect one end of a 220 ohm resistor to pin 12 (Digit 1) and the other end to D13.

014_7Seg4DigitsComCathode.ino

```
#include "SevSeg.h"

SevSeg sevseg;

void setup()
{
  byte numDigits = 4;
  byte digitPins[] = {13, 10, 9, 7};
  byte segmentPins[] = {12, 8, 5, 3, 2, 11, 6, 4};
  sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins);
  sevseg.setBrightness(90);
}

void loop() {
  static unsigned long timer = millis();
  static int deciSeconds = 0;

  if (millis() >= timer)
  {
    deciSeconds++;
    timer += 100;
    if (deciSeconds == 10000)
    {
      deciSeconds = 0;
    }
    sevseg.setNumber(deciSeconds, 1);
  }
  sevseg.refreshDisplay();
}
```

http://www.hacktronics.com/Tutorials/arduino-and-7-segment-led.html

[1] You can save a lot of pins by using the "53 MAX7219 LED driver".

50. 4 Digits 7-Segment Display Common Cathode (5641AS)


This 4 Digits display has 7 segments and 1 decimal point per digit. For other information check the previous chapter.

50.1. Specifications 4 Digits 7-Segment Display Common Cathode (5641AS)

- 4 digits, with 7 Segments A..G and 1 Decimal Point per digit.
- Common cathode.
- 1 Select pin per digit.
- 4 limiting resistors needed, 1 for every select pin.
- Total of 12 Digital pins needed.
- 5641AS

50.2. Datasheet HS420561K Single Digit 7-Segment Display Common Cathode (5641AS)

http://mcdelectronics.com/userfiles/1185/files/LED%20Displays/quad_digit_7_segment_display_GNQ-5641Ax-Bx.pdf

51. 4 Digits 7-Segment Display Common Anode


This 4 Digits display has 7 segments and 1 decimal point per digit. It differs with the 4 Digits 7-Segment display from the previous chapter in the fact that this display has an common anode.

51.1. Specifications 4 Digits 7-Segment Display Common Anode

- 4 digits, with 7 Segments A..G and 1 Decimal Point per digit.
- Common cathode.
- 1 Select pin per digit.
- 4 limiting resistors needed, 1 for every select pin.
- Total of 12 Digital pins needed.
- 3641BS

51.2. Datasheet 3641BS Single Digit 7-Segment Display Common Anode

http://www.dipmicro.com/?datasheet=NFD-3641.pdf.

51.3. Connections 4 Digits 7-Segment Display Common Anode

Check the previous chapter for the connections.

51.4. Libraries needed for 4 Digits 7-Segment Display Common Anode

Check the previous chapter for information about the library.

51.5. Sample 4 Digits 7-Segment Display Common Anode

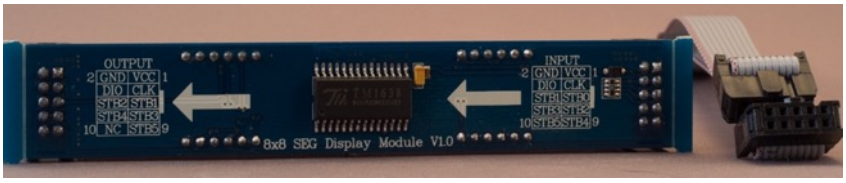
- Check the previous chapter for the connections. The sketch below is almost the same and differs only in the line were it says COMMON_ANODE instead of COMMON_CATHODE in the previous chapter.

136_7Seg4DigitsComAnode.ino

```
#include "SevSeg.h"

SevSeg sevseg;

void setup()
{
  byte numDigits = 4;
  byte digitPins[] = {13, 10, 9, 7};
  byte segmentPins[] = {12, 8, 5, 3, 2, 11, 6, 4};
  sevseg.begin(COMMON_ANODE, numDigits, digitPins, segmentPins);
  sevseg.setBrightness(90);
}

void loop() {
  static unsigned long timer = millis();
  static int deciSeconds = 0;

  if (millis() >= timer)
  {
    deciSeconds++;
    timer += 100;
    if (deciSeconds == 10000)
    {
      deciSeconds = 0;
    }
    sevseg.setNumber(deciSeconds, 1);
  }
  sevseg.refreshDisplay();
}
```

http://www.hacktronics.com/Tutorials/arduino-and-7-segment-led.html

52. 8 Digits 7-Segment Display with TM1638 chip



8 Digits 7 SEG display with a TM1638 chip, so you only need 3 to 8 ports on your Arduino, depended on the number of 8 digits displays you are using.

52.1. Specifications 8 Digits 7-Segment Display with TM1638 chip

- TM1638 controller chip.
- 8 Digits of 8 LEDs.
- Only three digital ports needed for one 8x8 SEG display (1 extra for every additional 8x8 SEG display).

52.2. Datasheet TM1638 chip

- http://dl.dropboxusercontent.com/u/8663580/TM1638English%20version.pdf

52.3. Connections 8 Digits 7-Segment Display with TM1638 chip

Input 2x5 pin header

Pin nr	Name	Description	Arduino pin
1	VCC	5 V	5V
2	GND	Ground	GND
3	CLK	Clock	Any Digital port
4	DIO	Data in	Any Digital port
5	STB0	Strobe 0 (this display module)	Any Digital port
6	STB1	Strobe 1 (next display module)	Any Digital port
7	STB2	Strobe 2 (next display module +1)	Any Digital port
8	STB3	Strobe 3 (next display module +2)	Any Digital port
9	STB4	Strobe 4 (next display module +3)	Any Digital port
10	STB5	Strobe 5 (next display module +4)	Any Digital port

Output 2x5 pin header

Pin nr	Name	Description
1	VCC	5 V
2	GND	Ground
3	CLK	Clock
4	DIO	Data in
5	STB1	Strobe 1 (next display module)
6	STB2	Strobe 2 (next display module +1)
7	STB3	Strobe 3 (next display module +2)
8	STB4	Strobe 4 (next display module +3)
9	STB5	Strobe 5 (next display module +4)
10	NC	Not connected

This 2x5 pin header is used to connect to output your signal to the next 8x8 SEG display through a flat cable (daisy chaining). The connections to the Arduino are made on the first display.

52.4. Libraries needed for 8 Digits 7-Segment Display with TM1638 chip

- TM1638 LED Driver library from Ricardo Batista rjbatista@gmail.com. https://code.google.com/p/tm1638-library/

Library use explanation

```
#include <TM1638.h>
```

Include TM1638 LED Driver library.

```
TM1638 mymodule(DIO, CLK, STR0);
```

Create 'mymodule' a new instance of the object type TM1638.
DIO, CLK and STR0 are Integer values corresponding to the Arduino
Digital Output to which DIO, CLK and STR0 are connected.

```
mymodule.setDisplayToDecNumber(1234);
```

Display 1234 on the 8x8 SEG display.

```
module.clearDisplay();
```

Clear all digits and dots.

```
module.setDisplayToString("  CAFE  ");
```

Display the string CAFE in the middle of the display. Not all
characters are readable on an 8x8 display.

More information about this library, can be found at:
https://github.com/rjbatista/tm1638-library/wiki/Reference

52.5. Sample 8 Digits 7-Segment Display with TM1638 chip

This sketch counts down from 1000-0ms, then it say BOOH. A non-blocking delay has been used.

Sample connections

- Connect VCC with 5V.
- Connect GND with GND.
- Connect CLK with D9
- Connect DIO with D8
- Connect STB0 with D7

015_7Seg8DigitsTM1638.ino

```
#include <TM1638.h>

TM1638 module(8, 9, 7);

long previousMillis = 0;
long interval = 10;
long countdown = 1000; //ms

void setup()
{
}

void loop()
{
  for (int count = countdown; count >= 0; count--)
  {
    //non-blocking delay
    previousMillis = millis();
    while (millis() - previousMillis < interval)
    {
      module.setDisplayToDecNumber(count, 2, false);;
    }
  }
  module.clearDisplay();
  while (1)
  {
    module.setDisplayToString(String("BOOH"), 0, 2);
  }
}
```

53. MAX7219 LED driver

The MAX7219 is a popular LED Driver. It can drive 64 individual bits arranged as an array, as an 8x8 matrix or 8 single digits seven segment displays. It only needs 3 IO ports, 1 current limiting resistor and two decoupling capacitors.

53.1. Specifications MAX7219 LED driver

- Common Cathode LED driver
- Can be used for:
  - 1-8 digits 7 segment displays
  - 8x8 LED matrices
  - Array of 1-64 LED's (or Bar Graphs)
- Multiple MAX7219 modules can be daisy chained
- Compatible with SPI, QSPI and microwire
- 10 MHz serial interface
- Digital and analog brightness control
- 8x8 bit buffer RAM
- Only 1 current limiting resistor needed (9.53K ..47K)
- Display blanked on power-up
- V+, DIN, CLK, LOAD, CS: -0.3 to 6V
- All other pins: -0.3 to V+ + 0.3V

53.2. Datasheet MAX7219 LED driver

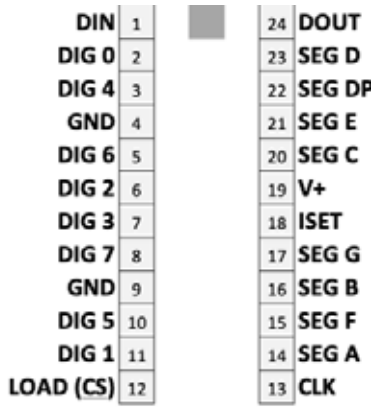
The datasheet can be found at:

- http://pdf1.alldatasheet.com/datasheet-pdf/view/73745/MAXIM/MAX7219.html

More information can be found at the following instructable:

- https://www.instructables.com/id/Controlling-simple-LED-Bar-Graph-with-Arduino/

53.3. Connections MAX7219 LED driver



Pin nr	Name	Description	Arduino pin
1	DIN	Digital IN	MOSI (D11) Only on first MAX7219
2	DIG 0	Common cathode Digit 0	
3	DIG 4	Common cathode Digit 4	
4	GND		GND
5	DIG 6	Common cathode Digit 6	
6	DIG 2	Common cathode Digit 2	
7	DIG 3	Common cathode Digit 3	
8	DIG 7	Common cathode Digit 7	
9	GND		GND
10	DIG 5	Common cathode Digit 5	
11	DIG 1	Common cathode Digit 1	
12	LOAD (CS)		Any digital pin (on all MAX7219's the same pin)
13	CLK		SCK (D13) (on all MAX7219's)
14	SEG A	Anode Segment A	
15	SEG F	Anode Segment F	
16	SEG B	Anode Segment B	
17	SEG G	Anode Segment G	
18	ISET		Current limiting resistor 9.53K..47K to 5V (depending on # LED's)
19	V+		5V
20	SEG C	Anode Segment C	
21	SEG E	Anode Segment E	
22	SEG DP	Anode Segment DP	
23	SEG D	Anode Segment D	
24	DOUT	Digital out to next MAX7219	• n.c. • DIN next MAX7219

53.4. Libraries needed for MAX7219 LED driver (LedControl)

There are several libraries available for the MAX7219. LedControl is a popular library to use with LED Bar Graphs, LED Arrays and 7 segments displays.

- LedControl library by Eberhard Fahle through the Library Manager

Library use explanation

```
#include <LedControl.h>
```

Include Eberhard Fahle's LedControl library.

```
LedControl mydisplay = LedControl(DIN, CLK, CS_LOAD, CHIPS);
```

Create 'mydisplay' a new instance of the object type LedControl.
DIN, CLK and CS_LOAD are the pints to which DIN, CLK and CS_LOAD are connected.
CHIPS is the number of daisy chained MAX7219 modules. The MAX7219 directly connected is chipnr 0.

```
mydisplay.shutdown(0, false);
```

Turn on the display at chip nr 0.

```
mydisplay.setIntensity(0, <brightness>);
```

Sets the light intensity of the display at chipnr 0, to a value from 0 (low) to 15 (maximum)

```
mydisplay.clearDisplay(<chipnr>);
```

Clears display nr 0.

```
mydisplay.setDigit(0, DIGIT, NUMBER, false);
```

Displays NUMBER on the specified DIGIT on chipnr 0.

```
mydisplay.setChar(0, DIGIT, CHARACTER, false);
```

Displays CHARACTER on the specified DIGIT on chipnr 0. Only the characters below can be used. All not-defined characters will display as a space.

setChar('..')	Display
A a	A (upper)
B b	b (lower)
C c	c (lower)
D d	d (lower)
E e	E (Upper)
F f	F (Upper)
H h	H (Upper)
L l	L (Upper)
o (lower)	o (Lower)
n (lower)	n (Lower)
P p	P (Upper)
Space	space
-	- (score)
_	_ (underscore)
. ,	. (dot)

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

53.5. Libraries needed for MAX7219 LED driver (Max72xxPanel)

There are several libraries available for the MAX7219. Max72xxPanel is a popular library to use with 8x8 DOT Matrices.

- SPI (Serial Peripheral Interface) library through the Library Manager.
- Max72xxPanel to address the Max7219 LED display driver. Created by Mark Ruys (mark@paracas.nl).
  https://github.com/markruys/arduino-Max72xxPanel
- Adafruit GFX library through the Library Manager. This library is used for displaying graphics on all sorts of displays.
  More information on the use of the Adafruit_GFX library can be find at the following URL: http://learn.adafruit.com/adafruit-gfx-graphics-library?view=all

Library use explanation

```
#include <SPI.h>
```
Include the Serial Peripheral Interface included in the Arduino IDE.

```
#include <Adafruit_GFX.h>
```
Include the display library from Adafruit.

```
#include <Max72xxPanel.h>
```
Include the MAX 72xxPanel library from Mark Ruys (mark@paracas.nl).

```
Max72xxPanel mymatrix = Max72xxPanel(CS, HOR, VERT);
```
Create 'mymatrix' a new instance of the object type Max72xxPanel.
CS is an Integer value corresponding to the Arduino Digital Output to which CS pin is connected.
HOR is the number of panels horizontal.
VERT is the number of panels vertical.

```
mymatrix.fillScreen(LOW);
```
Turn of all pixel in mymatrix's buffer.

```
mymatrix.write();
```
Showing content of mymatrix's buffer on the DOT matrix. In this case, all LED's are turned off. This function should be called every time a change in mymatrix's buffer needs to be showed in the DOT Matrix.

```
mymatrix.drawPixel(4, 4, HIGH);
```
Turn on pixel 4, 4 in mymatrix's buffer.

```
mymatrix.write();
```
Showing content of mymatrix's buffer on the DOT matrix. In this case, LED 4,4 is turned on.

```
mymatrix.setIntensity(0);
```
Sets the light intensity to a value from 0 (low) to 15 (maximum).

```
mymatrix.width()
```
Results in the width of all panels together.

```
mymatrix.height()
```
Results in the height of all panels together.

```
mymatrix.drawLine(X1, Y1, X2, Y2, HIGH);
```
Draws a line from (X1, Y1) to (X2, Y2).

53.6. Samples MAX7219

This paragraphs describes the connections and sample sketches for three different configurations:

- 10 segment LED Bar Graph of LED Array
- 2 single digit 7 segment displays
- 8x8 LED Matrix

All these configurations have the following connections in common. At first I thought that all configurations needed the same current-limiting resistor. But I'm not sure yet. I don't understand (yet) how to calculate the correct value for the resistor. With trial on error, I noticed that when the resistor value was too low, that the MAX7219 skipped some clock signals. This means that when I looped through the digits 0..9, the MAX7219 sometimes skipped 1 or 2 digits at random. This problem was solved by raising the resistor value from 10K to 15K.

Common Connections MAX7219

- Connect DIN (1) to MOSI (D11)
- Connect all GND's (4,9) to GND
- Connect CS (12) to D10
- Connect CLK (13) to SCK (D13)
- Connect VCC (19) to 5V
- Connect an 0.1 uF capacitor between GND (4/9) and VCC (19) (as close as possible to the MAX7219 module)
- Connect an 10uF electrolytic capacitor between GND (4/9 minus) and VCC (19, (as close as possible to the MAX7219 module)

MAX7219 sample with a 10 segment Bar Graph (LedControl)

For a LED Array or a LED Bar Graph, you'll need to group the cathodes of 8 consecutive LED's to DIG0, the next 8 LED's to DIG 1 etc…
From all groups of 8, you'll need to connect the anodes of the first LED of every group to SEG DP and the other 7 anodes to SEG A..SEG G.
This way, you can drive up to 64 (8x8) individually.

The following sketch drives the LED's on a 10 segment LED Bar Graph.

- Connect a 10K resistor between ISET (18) and VCC (19)
- Connect DIG 0 (2) to the cathodes of the first 8 LED's
- Connect DIG 1 (11) to the cathodes of the last 2 LED's
- Connect SEG DP (22) to the anode of LED1
- Connect SEG A..SEG H (14, 16, 20, 23, 21 15 & 17) to the anodes of LED2..LED8
- Connect SEG DP (22) to the anode of LED 9 (this pin is also connected to LED 1)
- Connect SEG A (14) to the anode of LED 10 (this pin is also connected to LED 2)

133_MAX7219_BarGraph.ino

```
#include <LedControl.h>

#define NRCHIPS 1  //Nr of chips (=displays)
#define NRSEGMENTS 10 //Nr of segments in bar graph
#define DIN 11
#define CLK 13
#define CS_LOAD 10

LedControl mydisplay = LedControl(DIN, CLK, CS_LOAD, NRCHIPS);

void setup()
{
  mydisplay.shutdown(0, false);  // turns on display
  mydisplay.setIntensity(0, 4); // 15 = brightest
  mydisplay.clearDisplay(0);
}

void loop()
{
  for (int segment = 0; segment < NRSEGMENTS; segment++)
  {
    int rownr = segment / 8;
    int columnr = segment % 8;
    mydisplay.setLed(0, rownr, columnr, true);
    delay (100);
    mydisplay.setLed(0, rownr, columnr, false);
    delay(100);
  }
}
```

MAX7219 sample with 2 single digit 7 Segment displays (LedControl)

The following sketch loops through the two digits and per digit through the numbers 0..9 and through all available characters.

- Connect a 15-33K resistor between ISET (18) and VCC (19)
- Connect DIG 0 (2) with both cathodes (GND) of the first digit
- Connect DIG 1 (11) with both cathodes (GND) of the second digit
- Connect SEG DP (22) to the DP's of both digits
- Connect SEG A..SEG H (14, 16, 20, 23, 21 15 & 17) to the A..H of both digits

132_MAX7219_2Single7Segments.ino

```
#include <LedControl.h>

#define NRCHIPS 1  //Nr of chips (=displays)
#define NRDIGITS 1 //Nr of digits per chip
#define DIN 11
#define CLK 13
#define CS_LOAD 10

LedControl mydisplay = LedControl(DIN, CLK, CS_LOAD, NRCHIPS);
char character[15] = {'A', 'B', 'C', 'D', 'E', 'F', 'H', 'L', 'n', 'o',
'P', ' ', '-', '_', '.'};

void setup()
{
  mydisplay.shutdown(0, false);  // turns on display
  mydisplay.setIntensity(0, 15); // 15 = brightest
  mydisplay.clearDisplay(0);
}

void loop()
{
  for (int digit = 0; digit < NRDIGITS; digit++)
  {
    for (int count = 0; count < 10; count++)
    {
      mydisplay.setDigit(0, digit, count, false);
      delay(200);
    }
    for (int count = 0; count < 13; count++)
    {
      mydisplay.setChar(0, digit, character[count], false);
      delay(200);
    }
  }
}
```

MAX7219 with an 8x8 LED matrix

For an 8x8 matrix, you'll need to connect the consecutive cathodes to DIG0..7 and the anodes to SEG DP & SEG A..H. On some matrices the Rows are Cathodes and on other matrices the Columns are Cathodes.

Connections specific for a matrix with Cathode = Row, Anode = Column

For example, the 8x8 DOT Matrices1388ASR & HLM1588BR.

- Connect a 33K resistor between ISET (18) and VCC (19)
- Connect Row 1..8 to DIG 0..7
- Connect Column 1 to SEG DP
- Connect Column 2..8 to SEG A..H

MAX7219		**8x8 Matrix, Cathode = Row, Anode = Column**	
Pin	**Description**	**Pin**	**Description**
2	DG0	9	Cathode Row 1
3	DG4	1	Cathode Row 5
5	DG6	2	Cathode Row 7
6	DG2	8	Cathode Row 3
7	DG3	12	Cathode Row 4
8	DG7	5	Cathode Row 8
10	DG5	7	Cathode Row 6
11	DG1	14	Cathode Row 2
14	SEG A	3	Anode Column 2
15	SEG F	15	Anode Column 7
16	SEG B	4	Anode Column 3
17	SEG G	16	Anode Column 8
20	SEG C	10	Anode Column 4
21	SEG E	11	Anode Column 6
22	SEG DP	13	Anode Column 1
23	SEG D	6	Anode Column 5

Connections specific for a matrix with Cathode = Column, Anode = Row

For example, the 8x8 DOT Matrix1088AS.

- Connect a 33K resistor between ISET (18) and VCC (19)
- Connect Column 1..8 to DIG 0..7
- Connect Row 1 to SEG DP
- Connect Row 2..8 to SEG A..H

MAX7219		8x8 Matrix, Cathode = Column, Anode = Row	
Pin	**Description**	**Pin**	**Description**
2	DG0	13	Cathode Column 1
3	DG4	6	Cathode Column 5
5	DG6	15	Cathode Column 7
6	DG2	4	Cathode Column 3
7	DG3	10	Cathode Column 4
8	DG7	16	Cathode Column 8
10	DG5	11	Cathode Column 6
11	DG1	3	Cathode Column 2
14	SEG A	14	Anode Row 2
15	SEG F	2	Anode Row 7
16	SEG B	8	Anode Row 3
17	SEG G	5	Anode Row 8
20	SEG C	12	Anode Row 4
21	SEG E	7	Anode Row 6
22	SEG DP	9	Anode Row 1
23	SEG D	1	Anode Row 5

134_MAX7219_Matrix1.ino (LedControl)

This sample uses the LedControl library.

```
#include <LedControl.h>

#define NRCHIPS 1  //Nr of chips (=displays)
#define NRSEGMENTS 64 //Nr of segments in bar graph
#define DIN 11
#define CLK 13
#define CS_LOAD 10

LedControl mydisplay = LedControl(DIN, CLK, CS_LOAD, NRCHIPS);

void setup() {
  mydisplay.shutdown(0, false);  // turns on display
  mydisplay.setIntensity(0, 4); // 15 = brightest
  mydisplay.clearDisplay(0);
}

void loop()
{
  for (int segment = 0; segment < NRSEGMENTS; segment++)
  {
    int rownr = segment / 8;
    int columnr = segment % 8;
    mydisplay.setLed(0, rownr, columnr, true);
    delay (100);
    mydisplay.setLed(0, rownr, columnr, false);
    delay(100);
  }
}
```

135_MAX7219_Matrix2.ino (Max72xxPanel)

This sample uses the Max72xxPanel library with Adafruit's GFX library.

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 10;

Max72xxPanel matrix = Max72xxPanel(pinCS, 1, 1);

void setup()
{
  matrix.setIntensity(0);
}

int wait = 50;
int inc = -2;

void loop()
{
  for ( int x = 0; x < matrix.width() - 1; x++ )
  {
    matrix.fillScreen(LOW);
    matrix.drawLine(x, 0, matrix.width() - 1 - x, matrix.height() - 1,
HIGH);
    matrix.write(); // Send bitmap to display
    delay(wait);
  }
  for ( int y = 0; y < matrix.height() - 1; y++ )
  {
    matrix.fillScreen(LOW);
    matrix.drawLine(matrix.width() - 1, y, 0, matrix.height() - 1 - y,
HIGH);
    matrix.write();
    delay(wait);
  }
  wait = wait + inc;
  if ( wait <= 0 ) inc = 2;
  if ( wait >= 50 ) inc = -2;
}
```

54. WS2812B RGB LED breakout-board

The WS2812B REG LED breakout-board is an RGB LED with only 4 unique connectors. Sequentially connected WS2812B modules share only 1 PWM port on the Arduino.

54.1. Specifications WS2812B RGB LED breakout-board

- RGB each controlled with 8 bits (16 million colors)
- Integrated control circuit.
- Voltage 5 V.
- Single line cascading signal port.
- Only 1 Arduino port needed.
- Each LED draws a maximum of 60 mA when turned on at 100% white. In most projects, not all LEDs will be turned on at 100% white simultaneously. So as a rule of thumb, Adafruit advices 20 mA per LED. Since the 5V port of the UNO serves 200 mA max, this means a maximum of 10 WS2812B modules.[1]

54.2. Datasheet WS2812B RGB LED breakout-board

- https://www.adafruit.com/datasheets/WS2812B.pdf

54.3. Connections WS2812B RGB LED breakout-board

Pin nr	Name	Description	Arduino pin
1	G	Ground	GND
2	V	VCC	5V (or separate power)
3	O	Output	I of next WS2812B module
4	G	Ground	G of next WS2812B module
5	V	VCC	V of next WS2812B module
6	I	Input	Any PWM port

Always connect a large capacitor (1000 uF, 6.3 V or higher) with – to ground and + to 5V as closely to the WS2812B-modules as possible.
Also connect a 470 ohm resistor between the Arduino PWM port and the WS2812B modules as a protection of the Arduino port.

54.4. Libraries needed for WS2812B RGB LED breakout-board

- Adafruit's NeoPixel library through the Library Manager.[2]

[1] Adafruit has a very nice article about their NeoPixel (based on WS2812B) and powering: https://learn.adafruit.com/adafruit-neopixel-uberguide/power.
[2] Another nice library is FastLED from Daniel Garcia, also through the Library Manager.

Library use explanation

```
#include <Adafruit_NeoPixel.h>
```

Include the Adafruit NeoPixel library.

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB +
NEO_KHZ800);
```

Create 'strip' a new instance of the object type Adafruit_NeoPixel. PIN is an Integer value corresponding to the Arduino Digital Output to which I is connected. PIXELS is the number of WS2812B-modules used. NEO_GRB and NEO_KHZ800 are constants defined in the NeoPixel library and need to match the specs of the type of WS2812B-modules used.

```
strip.begin();
strip.show();
```

Initialize the strip and subsequently clear all WS2812B-modules in the strip (switch them off).

```
uint32_t color = strip.Color(255 , 0, 0);
```

Fill a 32 bit integer with the value for Red (255,0,0).

```
strip.setPixelColor(0, color);
```

Set the buffer for WS2812B-module 0 (first WS2812B-module) to the color Red.

```
strip.show();
```

Set all the WS2812B-modules to the colors defined in the WS2812B-modules buffers.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

54.5. Sample WS2812B RGB LED breakout-board

The following sketch sets 3 WS2812B-modules to respectively RED, GREEN and BLUE.

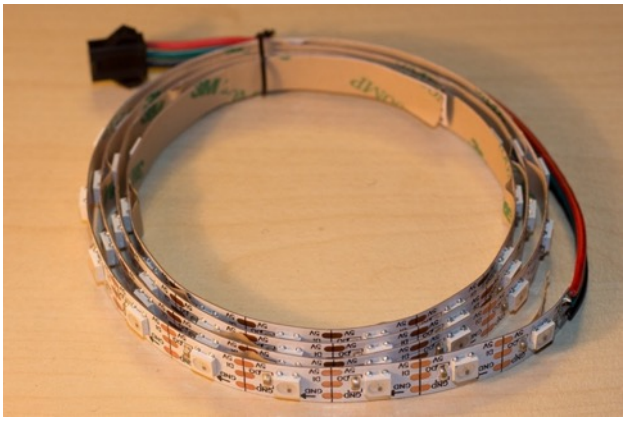
Sample Connections

- First WS2812B-module:
  - Connect VCC to 5V.
  - Connect GND to GND.
  - Connect a 1000 uF 6.3 V capacitor between VCC and GND.
- All WS2812B-modules (including the first):
  - Connect VCC to VCC of the next WS2812B-module (or directly to 5V).
  - Connect GND to GND of the next WS2812B-module (or directly to GND)
- Connect Input to one leg of a 470 ohm resistor.
- Connect the other leg of the 470 ohm resistor to D6.

016_WS2812B.ino

```
#include <Adafruit_NeoPixel.h>

#define PIN 6
#define PIXELS 3
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB +
NEO_KHZ800);

void setup() {
  strip.begin();
  strip.show();
}

void loop() {
  uint32_t color;
  for (int count = 0; count < PIXELS; count++)
  {
    color = strip.Color(255 , 0, 0);
    strip.setPixelColor(count, color);
    strip.show();
    delay(100);
    color = strip.Color(0, 50, 0);
    strip.setPixelColor(count, color);
    strip.show();
    delay(100);
  }
}
```

55. WS2812B RGB LED strip



This WS2812B RGB LED strip consists of multiple LED's per meter. You can cut the strip between any two LED's creating smaller strips.

55.1. Specifications WS2812B RGB LED strip

- RGB each controlled with 8 bits (16 million colors)
- Integrated control circuit.
- Voltage 5 V.
- Single line cascading signal port.
- Only 1 Arduino port needed.
- These strips need a separate power supply wit about 1-1,5 A per strip![1]
- Available in lengths of 1, 2 and 5 meter.
- Available in different densities: 30, 60 and 144 LED's per meter.
- Also available in a waterproof version.
- PCB (background) is available in white or black.
- Self-adhesive strip.
- Input connector and Output connector can be connected to create longer strips (daisy chaining).
- Extra leads for power either at the Input or at the Output connector.

55.2. Datasheet WS2812B RGB LED strip

- https://www.adafruit.com/datasheets/WS2812B.pdf

[1] Adafruit has a very nice article about their NeoPixel (based on WS2812B) and powering: https://learn.adafruit.com/adafruit-neopixel-uberguide/power.

55.3. Connections WS2812B RGB LED strip

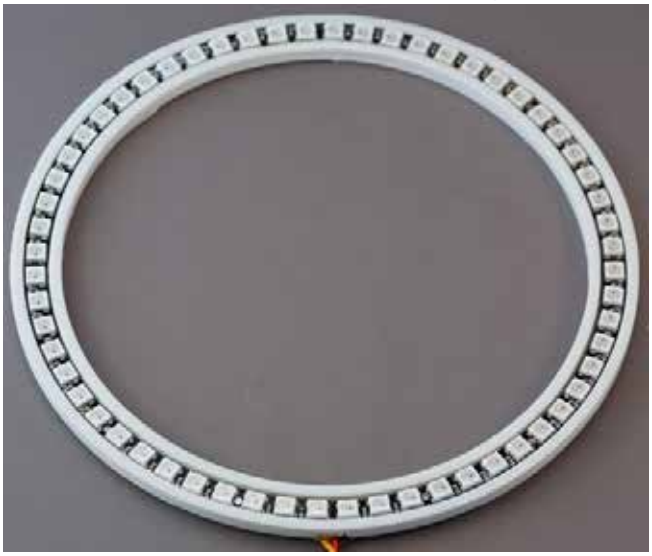
Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	DI	Date In	Any PWM port
3	+5V	5 Volt	separate power supply
1a	GND	Ground	NC or GND of next strip
2a	DO	Data Out	NC or DI of next strip
3a	+5V	5 Volt	NC or +5V of next strip

Always connect a large capacitor (1000 uF, 6.3 V or higher) with – to ground and + to 5V as closely to the WS2812B-modules as possible.
Also connect a 470 ohm resistor between the Arduino PWM port and the WS2812B modules as a protection of the Arduino port.

55.4. Libraries needed for WS2812B RGB LED strip

- See previous chapter for library description.

56. WS2812B RGB LED ring



This WS2812B RGB LED ring consists of 4 segments with each 15 WS2812B RGB LED's. In the picture above they are mounted in a 3D printed ring.

56.1. Specifications WS2812B RGB LED strip

- RGB each controlled with 8 bits (16 million colors)
- Integrated control circuit.
- Voltage 5 V.
- Single line cascading signal port.
- Only 1 Arduino port needed.
- These rings need a separate power supply wit about 1-1,5 A per strip![1]
- All 4 segments are equal and should be soldered together. 3 wires need to be soldered at the input-side of 1 of these segments.

56.2. Datasheet WS2812B RGB LED breakout-board

- https://www.adafruit.com/datasheets/WS2812B.pdf

[1] Adafruit has a very nice article about their NeoPixel (based on WS2812B) and powering: https://learn.adafruit.com/adafruit-neopixel-uberguide/power.

56.3. Connections WS2812B RGB LED breakout-board

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	DI	Date In	Any PWM port
3	+5V	5 Volt	separate power supply
1a	GND	Ground	NC or GND of next strip
2a	DO	Data Out	NC or DI of next strip
3a	+5V	5 Volt	NC or +5V of next strip

Always connect a large capacitor (1000 uF, 6.3 V or higher) with – to ground and + to 5V as closely to the WS2812B-modules as possible.
Also connect a 470 ohm resistor between the Arduino PWM port and the WS2812B modules as a protection of the Arduino port.

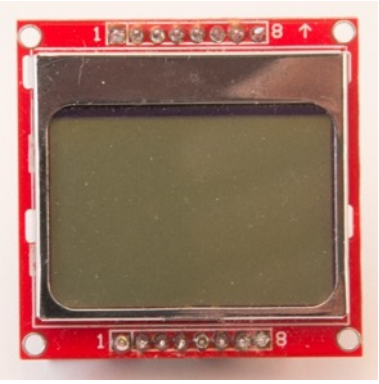
56.4. Libraries needed for WS2812B RGB LED breakout-board

- See previous chapter for library description.

LCD/TFT/OLED & e-Paper displays

In this section you will find a couple of LCD/TFT/OLED and e-Paper displays.

57. Nokia 5110/3310 LCD



This screen has been used in the popular Nokia 5110/3310 mobile phones and was produced in large quantities. They are still available and the prices are very low when bought on e-bay or at free shipping companies like Deal Extreme, Mini in the Box and Banggood (about $ 3,-).

57.1. Specifications Nokia 5110/3310 LCD

- 48x84 pixel monochrome LCD.
- Based on Philips PCD8544 display.
- 8 pin header row.

57.2. Datasheet Philips PCD8544: 48 × 84 pixels matrix LCD controller/driver

https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf

57.3. Connections Nokia 5110/3310 LCD

Be careful, not all PCD8544/5110 displays have the same pin-assignment.

Pin nr	Name	Description	Arduino pin
1	RST/RES	Reset	Any digital port
2	(S)CE	Chip Enable	Any digital port
3	DC	Data Command	Any digital port
4	(S)DIN	(Serial) Data in	Any digital port
5	CLK	Clock	Any digital port
6	VCC	3.3 V	3.3 V
7	Light	Backlight	GND
8	GND	Ground	GND or Digital PWM port

57.4. Libraries needed for Nokia 5110/3310 LCD

There are more PCD8544 libraries available, but in this document the following libraries are recommended.

- Adafruit_PCD8544 library from Adafruit through the Library Manager.
- Adafruit GFX library through the Library Manager. This library is used for displaying graphics on all sorts of displays.
  More information on the use of the Adafruit_GFX library can be find at the following URL: http://learn.adafruit.com/adafruit-gfx-graphics-library?view=all
- SPI (Serial Peripheral Interface) library through the Library Manager.

Library use explanation

```
#include <Adafruit_GFX.h>
```

Include the graphical library from Adafruit.

```
#include <Adafruit_PCD8544.h>
```

Include the display library from Adafruit.

```
Adafruit_PCD8544 mydisplay = Adafruit_PCD8544(CLK, DIN, DC, CE, RST);
```

Create an object called 'mydisplay', CLK, DIN, DC, CE and RST are Integer values corresponding to the correct pin assignment.

```
mydisplay.begin();
```

Initialize mydisplay and put the Adafruit logo in the mydisplay buffer. This buffer consumes 0,5 KB, but makes display very fast. At this point the buffer is not yet printed to the display.

```
mydisplay.setContrast(50);
```

Set contrast of the display at 50%.

```
mydisplay.clearDisplay();
```

Clear mydisplay's buffer (only the Adafruit logo was in there at this point).

```
mydisplay.println("Hello World");
```

Put the line "Hello World" in mydisplay's buffer.

```
mydisplay.display;
```

Write the content of mydisplay to the LCD screen. This comment is needed every time you want the changes to mydisplay's buffer to be seen on your display.

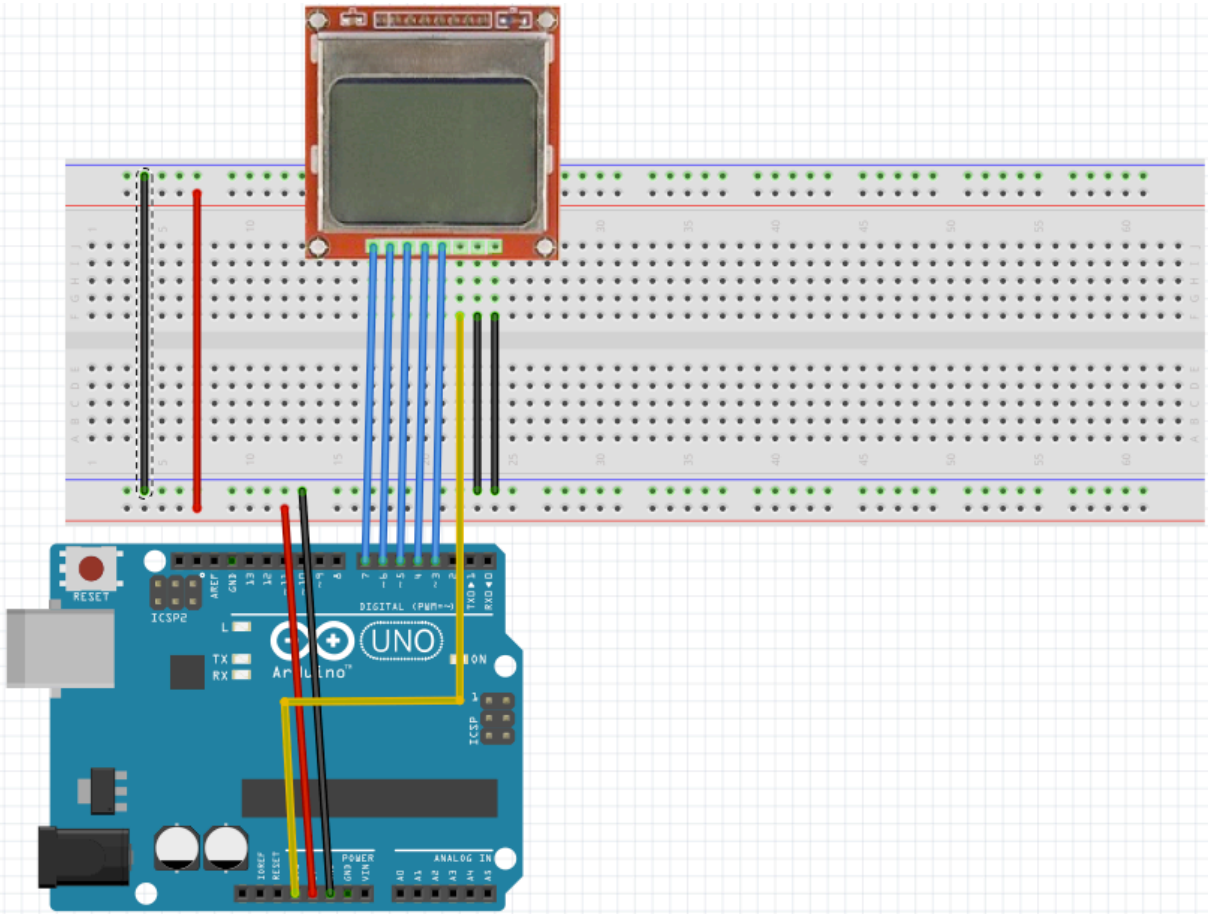
As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folder.

57.5. Sample Nokia 5110/3310 LCD display

This sample script displays a splash screen (Adafruit's logo) for 1 seconds and repeats to show the text "PCD8544 test" and a circle-outline.

Sample Connections

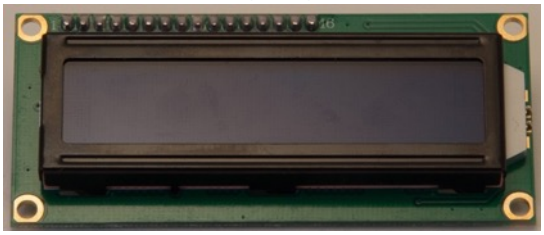
- Connect RST with D7.
- Connect (S)CE with D6.
- Connect DC with D5.
- Connect DIN with D4.
- Connect CLK with D3.
- Connect VCC with 3.3 V
- Connect Light with GND
- Connect GND with GND



017_LCDNokia3310.ino

```
int CLK=3;
int DIN=4;
int DC=5;
int CE=6;
int RST=7;
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Adafruit_PCD8544.h>
Adafruit_PCD8544 display = Adafruit_PCD8544(CLK, DIN, DC, CE, RST);

void setup()
{
  display.begin();
  display.setContrast(50);
}

void loop()
{
  display.clearDisplay();
  display.setCursor(10,18);
  display.println("PCD8544 test");
  display.display();
  delay(1000);
  display.clearDisplay();
  display.drawCircle(42, 23, 23, BLACK);
  display.display();
  delay(1000);
}
```

58. 16x2 Display 1602A



This display needs 6 of its own pins (RS, EN, D7, D6, D5 and D4) connected to the same number of digital pins on your Arduino board. This chapter is for 16x2 displays without a nice LCM1602 chip (check the next chapter if yours is equipped with the LCM1602 chip "59 16x2 Display with LCM1602 chip").

58.1. Specifications 16x2 Display 1602A

- GDM1602X 16x2 display.
- Blue backlight.
- HD44780 parallel interface chipset.

58.2. Datasheets 16x2 Display 1602A

Datasheet GDM1602X 16x2 display

https://www.sparkfun.com/datasheets/LCD/GDM1602K.pdf

Datasheet HD44780 parallel interface chipset

https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

58.3. Connections 16x2 Display 1602A

Pin nr	Name	Description	Arduino pin
1	VSS	GND	GND
2	VDD	+5V	5V
3	V0	Contrast adjustment	Potmeter to GND
4	RS	H/ Register select signal	PWM
5	RW	H/L Read/Write signal	GND
6	E	H/L Enable line	Any digital port
7	D0	H/L Data bus line	Only used in 8 bit setup
8	D1	H/L Data bus line	Only used in 8 bit setup
9	D2	H/L Data bus line	Only used in 8 bit setup
10	D3	H/L Data bus line	Only used in 8 bit setup
11	D4	H/L Data bus line	Any digital port
12	D5	H/L Data bus line	Any digital port
13	D6	H/L Data bus line	Any digital port
14	D7	H/L Data bus line	Any digital port
15	A	+4.2V for LED	Any digital port or static on 5V
16	K	Power supply for BKL 0V (Backlight)	GND

58.4. Libraries need for 16x2 Display 1602A

- LiquidCrystal library from Arduino through the Library Manager .

Library use explanation

```
#include <LiquidCrystal.h>
```

Include the Liquid Crystal library.

```
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
```

Create 'lcd' a new instance of the object type LiquidCrystal. RS, E, D4, D5, D6, D7 are Integer values corresponding to the correct pin assignment.

```
lcd.begin(16,2);
```

Initialize your display for 16 characters on 2 rows (16x2).

```
lcd.setCursor(0,0);
```

Set the cursor to the 0^{th} row and 0^{th} column.

```
lcd.print("Howdie");
```

Print the text "Howdie" to your display.

More information about this library can be found at https://www.arduino.cc/en/Reference/LiquidCrystal.

58.5. Sample 16x2 Display 1602A

The following sketch, shows the text “Hello World!” on the first row, and the text “Arduino is nice!” on row 2.

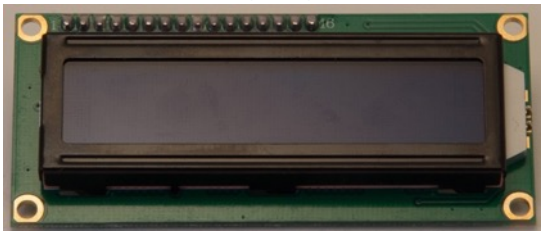
Connections

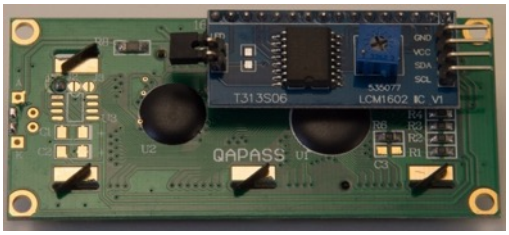
- Connect VSS (pin 1) to GND.
- Connect VDD (pin 2) to 5V.
- Connect V0 (pin 3) to the middle leg of a 10K potentiometer.
- Connect one of the other legs of the 10K potentiometer to GND
- Connect the last leg of the 10K potentiometer to 5V
    - This potentiometer can be used to control the contrast of the LCD
- Connect RS (pin 4) to D2.
- Connect RW (pin 5) to GND.
- Connect E (pin 6) to D3
- Connect D4 (pin 11) to D4
- Connect D5 (pin 12) to D5.
- Connect D6 (pin 13) to D6.
- Connect D7 (pin 14) to D7.
- Connect A (pin 15) to 5V (Backlight is not controlled by the Arduino in this sketch).
- Connect K (pin 16) to GND.

018_LCD16x2.ino

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(2, 3, 4, 5, 6 , 7);

void setup()
{
  lcd.begin(16,2);
  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
  lcd.setCursor(0,1);
  lcd.print("Arduino is nice!");
}

void loop()
{
}
```

59. 16x2 Display with LCM1602 chip (I2C)





This display normally needs 6 of its own pins (RS, EN, D7, D6, D5 and D4) connected to the same number of pins on your Arduino board. Luckily this Display is equipped with a backpack with a LCM1602 chip, so it can be controlled through I^2C. I^2C is a serial connection using 2 wires besides VCC and GND. You can also buy the LCM1602 backpack and solder it to you 16x2 LCD.

59.1. Specifications

- GDM1602X 16x2 display.
- Blue backlight.
- HD44780 parallel interface chipset.
- LCM1602 I^2C.

59.2. Datasheets 16x2 display with LCM1602 chip (I2C)

Datasheet GDM1602X 16x2 display

https://www.sparkfun.com/datasheets/LCD/GDM1602K.pdf

Datasheet HD44780 parallel interface chipset

https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

Datasheet LCM1602 chip

http://www.adafruit.com/datasheets/TC1602A-01T.pdf

59.3. Connections 16x2 Display with LCM1602 chip (I2C)

Pin nr	Name	Description	Arduino pin
1	SCL	I^2C Clock	SCL (A5)
2	SDA	I^2C Data	SDA (A4)
3	VCC	5 V	5V
4	GND	Ground	GND

59.4. Libraries need for 16x2 Display with LCM1602 chip (I2C)

- LiquidCrystal I2C library from Frank de Brabander through the Library Manager .
- Inter Integrated Circuit (I²C) and Two Wire Interface (TWI) library, included with Arduino IDE: ”Wire.h”.

Library use explanation

```
#include <Wire.h>
```

Include the Inter-Integrated Circuit (I²C) and Two Wire Interface (TWI) library.

```
#include <LiquidCrystal_I2C.h>
```

Include the Liquid Crystal I²C library.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

Create ‘lcd’ a new instance of the object type LiquidCrystal_I2C. 0x27 is the I2C address[1] of the display (this can be changed by soldering jumpers).

```
lcd.init();
```

Initialize your display.

```
lcd.backlight();
```

Turn on the backlight.

```
lcd.nobacklight();
```

Turn of the backlight.

```
lcd.setCursor(0,0);
```

Set the cursor to the 0th row and 0th column.

```
lcd.print("Howdie");
```

Print the text “Howdie” to your display.

More information about this library can be found in de “docs” folder inside the library folder.

[1] To check the I²C address of your display, hook up your display and run an I²C-scanner. You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

59.5. Sample 16x2 Display with LCM1602 chip (I2C)

The following sketch shows the text “Hello World!” on the first row and the text “Arduino is nice!” on row 2.

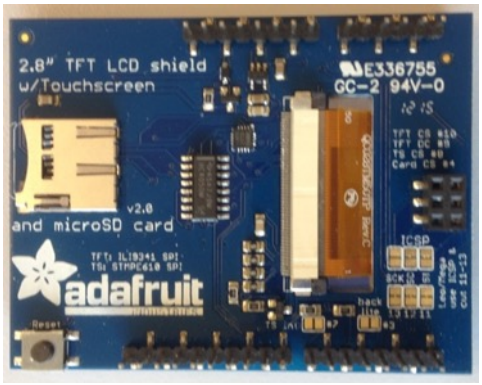
Connections

- Connect SCL to SCL (A5).
- Connect SDA to SDA (A4).
- Connect VCC to 5V.
- Connect GND to GND.

019_LCD16x2_I2C.ino

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,16, 2);

void setup()
{
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
  lcd.setCursor(0,1);
  lcd.print("Arduino is nice!");
}

void loop()
{
}
```

60. Adafruit 2.8" TFT Resistive Touch Shield v2



This 2.8" resistive touch TFT display is constructed as a shield to be placed on top of an Arduino Uno or MEGA (some soldering is required to use this on a MEGA).

60.1. Specifications Adafruit 2.8" TFT Resistive Touch Shield v2

- 2.8" diagonal display
- ILIL9341 display chipset
- STMPE610 touchscreen chipset
- 4 white-LED backlight
- 18 bits color depth (262.000 different shades)
- 240x320 pixels (width x height)
  - (0,0) is upper left corner
  - (240,320) is lower right corner

60.2. Datasheet Adafruit 2.8" TFT Resistive Touch Shield v2

- https://learn.adafruit.com/downloads/pdf/adafruit-2-8-tft-touch-shield-v2.pdf
- http://www.displaytech-us.com/sites/default/files/driver-ic-data-sheet/Ilitek-ILI9341.pdf
- http://pdf1.alldatasheet.com/datasheet-pdf/view/346216/STMICROELECTRONICS/STMPE610.html

60.3. Connections Adafruit 2.8" TFT Resistive Touch Shield v2

This display is constructed as a shield. The table below describes which pins are used by the shield. Three pins are shared by the TFT, touchscreen and SD-reader (13, 12 and 11). Two pins are used for the TFT-display (D10 and D9) and both the touchscreen and the SD-reader use one extra pin each (resp. D8 and D4).

Name	Description	Arduino pin
ICSP SCLK	SPI clock for TFT, Touch and SD	D13
ICSP MISO	SPI Master-In-Slave-Out for TFT, Touch and SD	D12
ICSP MOSI	SPI Master-Out-Slave-In for TFT, Touch and SD	D11
CS ILI9343	Chip Select to select TFT	D10

DC	Data/command to send data to TFT	D9
CS STMPE610	Chip Select to select Touch	D8
CS SD-reader	Chip Select to select SD-reader	D4

The following pins are not used: D0..D3, D5..D7 and A0..A5, but are physically blocked by the shield. If you need those pins, you must find your own way to access them.

60.4. Libraries needed for Adafruit 2.8" TFT Resistive Touch Shield v2

I've been using the following libraries:

- Display ILI9341 library through the Library Manager.
- Adafruit GFX library through the Library Manager. This library is used for displaying graphics on all sorts of displays.
  More information on the use of the Adafruit_GFX library can be find at the following URL: http://learn.adafruit.com/adafruit-gfx-graphics-library?view=all
- Resistive Touchscreen STMPE610 from Adafruit through the Library Manager.
- Secure Digital card library through Library Manager. (only needed when using the SD card).
- SPI (Serial Peripheral Interface) library through the Library Manager.

Library use explanation

```
#include <SPI.h>
```

Include the Serial Peripheral Interface included in the Arduino IDE.

```
#include <Adafruit_GFX.h>
```

Include the display library from Adafruit.

```
#include < Adafruit_ILI9341.h>
```

Include the ILI9341 library from Adafruit to control the TFT display.

```
#include < Adafruit_ STMPE610.h>
```

Include the STMPE610 library from Adafruit to control the resistive touchscreen.

```
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
```

Create 'ts' a new instance of the object type Adafruit_STMPE610. STMPE_CS is an Integer value corresponding to the Arduino Digital Output to which CS pin of the STMPE610 touchscreen is connected.

```
Adafruit_ ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);
```

Create 'tft' a new instance of the object type Adafruit_ILI9341. TFTE_CS is an Integer value corresponding to the Arduino Digital Output to which the CS pin of the ILI9341 display is connected.

```
tft.begin();
```

Turn on display.

```
ts.begin();
```

Turn on touchscreen.

```
tft.fillScreen(ILI9341_BLUE);
```
Fills the screen with ILI9341_BLUE.

```
tft.fillRect(X, Y, W, H, ILI9341_RED);
```
Fills a rectangle origin at X,Y, width W and height H with color ILI9341_RED.

```
tft.setCursor(X, Y);
```
Sets cursor at X, Y.

```
tft.setTextColor(ILI9341_WHITE);
```
Sets text color to ILI9341_WHITE.

```
tft.setTextSize(2);
```
Sets text size at 2.

```
tft.println("Hello Word");
```
Prints the text Hello World at the given coordinates.

```
if (! ts.bufferEmpty())
```
Check if there's any data in the touchscreen buffer. As soon as you touch the display, this buffer is filled. You can collect that data anytime, with the following command.

```
if(ts.touched())
```
Checks if the touchscreen is touched AT THIS MOMENT.

```
TS_Point p;
p=ts.getPoint();
```
The oldest touch data is stored in p.x, p.y (0..4095) and p.z (0..255). p.X and p.y are coordinates, whereas p.z is pressure.

```
p.x = map(p.x, 0, 4095, 0, tft.width());
p.y = map(p.y, 0, 4095, 0, tft.height());
```
You need to map the x and y values of the touchscreen, to the size of the screen (320x240).

```
#define TS_MINX 360
#define TS_MINY 265
#define TS_MAXX 3864
#define TS_MAXY 3929
int px = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
int py = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
```
The touchscreen doesn't have the same size as the TFT display, so we need to ignore readings beyond the display itself.

60.5. Sample Adafruit 2.8" TFT Resistive Touch Shield v2

The following sketch fills the screen with blue, draws a small yellow rectangle and prints the touch coordinates to the serial monitor

Sample Connections

- Place the shield on top of the Arduino UNO.

027_LCDAdafruit2.8.ino

```
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>

// This is calibration data for the raw touch data to the screen
coordinates
#define TS_MINX 360
#define TS_MINY 265
#define TS_MAXX 3864
#define TS_MAXY 3929

#define STMPE_CS 8
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS);
#define TFT_CS 10
#define TFT_DC 9
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC);

void setup(void)
{
  Serial.begin(9600);
  tft.begin();
  if (!ts.begin())
  {
    Serial.println("Unable to start touchscreen.");
  }
  else
  {
    Serial.println("Touchscreen started.");
  }
  tft.fillScreen(ILI9341_BLUE);
}

void loop()
{
  if (!ts.bufferEmpty())
  {
    TS_Point p = ts.getPoint();
    int py = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
    int px = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
    tft.fillRect(px, py, 20, 20, ILI9341_YELLOW);
    Serial.print(px);
    Serial.print(" , ");
    Serial.println(py);
  }
}
```

61. 2.4” TFT LCD Shield Touch Board

This 2.8” resistive touch TFT display is constructed as a shield to be placed on top of an Arduino Uno or MEGA (some soldering is required to use this on a MEGA).

61.1. Specifications 2.4” TFT LCD Shield Touch Board

- 2.4” diagonal display
- ILI9325 display chipset
- 18 bits color depth (262.000 different shades)
- 240x320 pixels

61.2. Datasheet 2.4” TFT LCD Shield Touch Board

- http://www.adafruit.com/datasheets/ILI9325.pdf

61.3. Connections 2.4" TFT LCD Shield Touch Board

This display is constructed as a shield. The table below describes which pins are used by the shield. The shield uses almost all digital and analog pins. The display and touchscreen uses D2..D9 and A0..A4 and the SD reader uses D10..D13 (SPI?).

Name	Description	Arduino pin
LCD Data	TFT data bit 2	D2
LCD Data	TFT data bit 3	D3
LCD Data	TFT data bit 4	D4
LCD Data	TFT data bit 5	D5
LCD Data	TFT data bit 6 /Touchscreen X+	D6
LCD Data	TFT data bit 7 /Touchscreen Y-	D7
LCD Data	TFT data bit 0	D8
LCD Data	TFT data bit 1	D9
SD_SS	SD Slave Select	D10
SD_DI	SD Data IN (MOSI?)	D11
SD_D0	SD Data Out (MISO?)	D12
SD_SCK	SD Serial Clock	D13
LCD_RD	TFT Read	A0
LCD_WR	TFT Write Touchscreen Y+	A1
LCD_CD/LCD_RS	Command/Data TFT Touchscreen X-	A2
LCD_CS	Chip Select TFT	A3
LCD_RESET	TFT Reset	A4

The following pins are not used: D0, D1 and A5, but are physically blocked by the shield. If you need those pins, you must find your own way to access them. You could probably free up D10..D13 if the DS reader is not used.

61.4. Libraries needed for 2.4" TFT LCD Shield Touch Board

I've been using the following libraries:

- Display ILI9325 library from Smoke and Wires (http://www.smokeandwires.co.nz) https://github.com/Smoke-And-Wires/TFT-Shield-Example-Code
- Adafruit GFX library through the Library Manager. This library is used for displaying graphics on all sorts of displays.
  More information on the use of the Adafruit_GFX library can be find at the following URL: http://learn.adafruit.com/adafruit-gfx-graphics-library?view=all
- Adafruit Touchscreen library https://github.com/adafruit/Touch-Screen-Library
- Secure Digital card library through Library Manager. (only needed when using the SD card).

Library use explanation

```
#include <Adafruit_GFX.h>    // Core graphics library
```

Include the display library from Adafruit.

```
#include <SWTFT.h> // Hardware-specific library
```

Include the ILI9325 library from Smoke and Wires (http://www.smokeandwires.co.nz)

```
#include <TouchScreen.h>
```

Include the touchscreen library from Adafruit.

```
TouchScreen ts = TouchScreen(6, A1, A2, 7, 300);
```

Create 'ts' a new instance of the object type Touchscreen. D6 is connected to X+, A1 to Y+, A2 to X- and D7 to Y-. The last argument (300) is needed for better pressure precision and can be determined by measuring the resistance between X+ and X- with a multi-meter.

```
SWFT tft;
```

Create 'tft' a new instance of the object type SWFT.

```
tft.reset();
```

Resets tft values?

```
tft.begin(tft.readID());
```

Turn on the display. tft.readID() returns the TFT chipset.

```
delay(25);
```

Before using the display, you need to add a delay of at least 25ms, otherwise the top part of the display will contain black and white lines!!!!

```
tft.fillScreen(BLUE);
```

Fills the screen with BLUE.

```
tft.fillRect(X, Y, W, H, YELLOW);
```

Fills a rectangle origin at X,Y, width W and height with color YELLOW.

```
TS_Point p;
p=ts.getPoint();
```

The oldest touch data is stored in ps.x, ps.y (0..1024) and .z (0..1024). X and y are coordinates, whereas z is pressure.

```
pinMode(A2,OUTPUT);
pinmode(A1,OUTPUT);
```

I'm not sure about this two lines, it probably triggers the touchscreen. A2 is X- and A1 is Y+. Although pinMode() commands are usually placed in void setup(), this two lines must be placed directly after the ts.getPoint().

```
if (p.z > 10 && p.z < 1000)
```

The library doesn't have a method to test whether the screen has been touched. Checking if p.z is between 10 and 1000 gives the same result. More pressure means a lower value for p.z, but when the display is not touched, the value for p.z=0!

```
p.x = map(p.x, 0, 1024, 0, tft.width());
p.y = map(p.y, 0, 1024, 0, tft.height());
```

You need to map the x and y values of the touchscreen, to the size of the screen (320x240).

```
#define TS_MINX 150
#define TS_MINY 120
#define TS_MAXX 920
#define TS_MAXY 940
p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.width());
p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.height());
```

The touchscreen doesn't have the same size as the TFT display, so we need to ignore readings beyond the display itself.

61.5. Sample 2.4" TFT LCD Shield Touch Board

The following sketch fills the screen with blue, draws a small yellow rectangle and prints the touch coordinates to the serial monitor

Sample Connections

- Place the shield on top of the Arduino UNO.

028_LCDILI9325_2.4.ino

```
#include <Adafruit_GFX.h>
#include <SWTFT.h>
#include <TouchScreen.h>

#define YP A1
#define XM A2
#define YM 7
#define XP 6

#define TS_MINX 150
#define TS_MINY 120
#define TS_MAXX 920
#define TS_MAXY 940

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 300);

#define   BLUE    0x001F
#define YELLOW  0xFFE0

SWTFT tft;

void setup(void) {
  Serial.begin(9600);
  tft.reset();
  tft.begin(tft.readID());
  delay(25);
  tft.fillScreen(BLUE);
  Serial.println(tft.width());
  Serial.println(tft.height());
}

void loop()
{
  TSPoint p = ts.getPoint();
  pinMode(XM, OUTPUT);
  pinMode(YP, OUTPUT);

  if (p.z > 10 && p.z < 1000)
  {
    p.x = map(p.x, TS_MINX, TS_MAXX, tft.width(), 0);
    p.y = map(p.y, TS_MINY, TS_MAXY, tft.height(), 0);
    tft.fillRect(p.x - 20, p.y - 20, 20, 20, YELLOW);
    Serial.print(p.x);
    Serial.print(", ");
    Serial.print(p.y);
    Serial.print(" pressure: ");
    Serial.println(p.z);
  }
}
```

62. 0.96 inch 128x64 OLED display (I2C)

Tiny White/Blue display with 128x64 pixels. Through the use of I2C it only needs A4 (SDA) and A5 (SCL) and those pins can be shared with other I2C devices.

62.1. Specifications 0.96 inch 128x64 OLED display (I2C)

- 128x64 dots
- Dot size 0.148 x 0.148
- Dot pitch 0.17 x 0.17
- Supply voltage: 3.3V
- I2C
- Chip: SSD1306BZ

62.2. Datasheet 0.96 inch 128x64 OLED display (I2C)

- https://www.vishay.com/docs/37902/oled128o064dbpp3n00000.pdf

62.3. Connections 0.96 inch 128x64 OLED display (I2C)

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	VDD	Power	3.3V (to be safe)[1]
3	SCK	I2C clock	SCL (A5)
4	SDA	I2C data	SDA (A4)

62.4. Libraries needed for 0.96 inch 128x64 OLED display (I2C)

The library to use, depends on the fact if you want to display graphics, or just text. Working with graphics consumes a lot of RAM to store variables (sometimes 1KB plus). On an Arduino Uno with only 2KB RAM, this could give same LOW memory problems. If you are only using this display for text, then you can save a lot of RAM by using a text only library (using only a few bytes instead of 1 KB plus). In this chapter, I will only describe the text only library. The full library will be described in the next chapter for the smaller (less RAM hungry) 0.91 inch 128x32 OLED display. Both libraries can be used with either of these displays (0.96 inch 128x64 & 0.91 inch 128x32).

Text only library

- SSD1306ASCII library from Bill Greiman: https://github.com/greiman/SSD1306Ascii The documentation for this library is embedded in the folder structure of the library itself. You can use a browser to open .\libraries\SSD1306Ascii-master\doc\html\index.html.

[1] I ran my display at 5V at least 10 hours in a row, without any problems (both VDD as the I2C data levels).

- Inter Integrated Circuit (I^2C) and Two Wire Interface (TWI) library, included with Arduino IDE: "Wire.h". You don't need to include this library in your sketch, this automatically done by Bill Greiman's SSD1306ASCII library.

Text only library use explanation

```
#include "SSD1306Ascii.h"
```
Include Bill Greiman's general SSD1306Ascii library (needed for both I2C as for SPI displays).

```
#include "SSD1306AsciiWire.h"
```
Include Bill Greiman's SSD1306AsciiWire (I2C) library. This library will also include Arduino's own Wire library.

```
SSD1306AsciiWire oled;
```
Create 'oled' a new instance of the object type SSD1306AsciiWire.

```
Wire.begin();
```
Initialize I2C.

```
Wire.setClock(400000L);
```
Set the I2C clock to fast mode to speed up your display. When you omit this line, your display will be slower, this depends on your needs.

```
oled.begin(&Adafruit128x64, 0x3C);
```
Initialize your 128x64 display with I2C address set to 0x3C. Use a I2C scanner[1] sketch to check the I2C address of your specific OLED display. Use `oled.begin(&Adafruit128x32, 0x3C);` in case you are using an 128x32 display.

```
oled.setFont(TimesNewRoman16_bold);
```
You can choose from a list of fixed width or proportional fonts. The names of these fonts are listed in the documentation that is embedded in the library folder structure. Some of the fixed width fonts are: Adafruit5x7, fontsAdafruit5x7 and ZevvPeep8x16. The following fonts are proportional: Arial14, Callibri11, Roosewood26 and TimesNewRoman16.

```
oled.displayRemap(false);
```
You can use this command to rotate the display by 180 degrees (false is default).

```
oled.clear();
```
Clear the display.

```
oled.set2X();
```
With this command you can double the dots of each character in width and in height.

```
oled.set1X();
```
Set the font to its normal size.

```
oled.println("Top");
```
Print the text Top to the display and place the cursor at the next line (newline).

[1] You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

```
oled.println("Top");
```
Print the text Top to the display and leave the (invisible) cursor where it is.

```
oled.displayRows()
```
This returns the available rows on your display. Each row has a height of 8. So, on an 128x64 there are 8 lines.

```
oled.fontRows()
```
Number (int) of rows needed to display the current font.

```
oled.setRow(0);
```
Sets the (invisible) cursor at the top row.

```
oled.setRow(oled.displayRows() -oled.fontRows());
```
Sets the (invisible) cursor at the bottom row.

62.5. Sample 0.96 inch 128x64 OLED display (I2C)

The following sketch displays a line of text at the top row and a line of text at the bottom row.

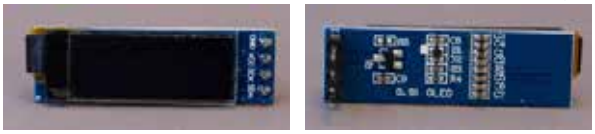
Sample Connections

- Connect VCC to 3.3V (to be safe)
- Connect GND to GND.
- Connect SCK to A5 (SCL)
- Connect SDA to A4 (SDA)

029_OLED_ASCII.ino

```
#include "SSD1306Ascii.h"
#include "SSD1306AsciiWire.h"

SSD1306AsciiWire oled;
SSD1306AsciiWire oled2;


void setup()
{
  Wire.begin();
  Wire.setClock(400000L);

  oled.begin(&Adafruit128x64, 0x3C);

  oled.setFont(Adafruit5x7);

  oled.displayRemap(false);
  oled.clear();
  oled.set1X();
  oled.println("Top");
  oled.setRow(7);
  oled.println("Bottom line");
}

void loop()
{
}
```

63. 0.91 inch 128x32 OLED display (I2C)



Tiny White/Blue display with 128x32 pixels. Through the use of I2C it only needs A4 (SDA) and A5 (SCL) and those pins can be shared with other I2C devices.

63.1. Specifications 0.91 inch 128x32 OLED display (I2C)

- 128x32 dots
- Dot size 0.152 x 0.152
- Dot pitch 0.175 x 0.175
- Supply voltage: 3.3-5V
- I2C
- Chip: SSD1306BZ

63.2. Datasheet 0.91 inch 128x32 OLED display (I2C)

- https://www.vishay.com/docs/37894/oled128o032dlpp3n00000.pdf

63.3. Connections 0.91 inch 128x32 OLED display (I2C)

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	VCC	Power	3.3V - 5V[1]
3	SCK	I2C clock	SCL (A5)
4	SDA	I2C data	SDA (A4)

63.4. Libraries needed for 0.91 inch 128x32 OLED display (I2C)

The library to use, depends on the fact if you want to display graphics, or just text. Working with graphics consumes a lot of RAM to store variables (sometimes 1KB plus). On an Arduino Uno with only 2KB RAM, this could give same LOW memory problems. If you are only using this display for text, then you can save a lot of RAM by using a text only library (using only a few bytes instead of 1 KB plus). In this chapter, I will only describe the full library. The text only library is described in the previous chapter for the larger (more RAM hungry) 0.96 inch 128x64 OLED display. Both libraries can be used with either of these displays (0.96 inch 128x64 & 0.91 inch 128x32).

Full (graphics) library

- Adafruit SSD1306 library through the library manager.
  Documentation: https://learn.adafruit.com/monochrome-oled-breakouts
- Adafruit GFX library through the library manager. You don't need to include this library in your sketch, this automatically done by Adafruit's SSD1306 library.
  Documentation: https://learn.adafruit.com/adafruit-gfx-graphics-library
- Inter Integrated Circuit (I^2C) and Two Wire Interface (TWI) library, included with Arduino IDE: "Wire.h". You don't need to include this library in your sketch, this automatically done by Adafruit's SSD1306 library.

[1] I ran my display at 5V for at least 10 hours in a row, without any problems (both VCC as the I2C data levels).

Text only library use explanation

```
#include <Adafruit_SSD1306.h>
```
Include Adafruit's SSD1306 library.

```
#include <Fonts/FreeSans9pt7b.h>
```
Load the definitions of one of Adafruit GFX library's fonts. They are included in the GFX library.

```
Adafruit_SSD1306 display(W, , &Wire);
```
Create 'display' a new instance of the object type Adafruit_SSD130. In this case 128 and 32 are the width and height of the display (change to 128, 64 for the larger displays. &Wire is a pointer to the I2C library.

```
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
```
Initialize the display at I2C address 0x3C.Use a I2C scanner[1] sketch to check the I2C address of your specific OLED display.

```
display.display();
```
With this command you tell the OLED to display anything that is inside its display buffer. If you haven't stored anything in this display buffer yet, the Adafruit logo will be displayed.

```
display.clearDisplay();
```
Empty the display's buff. This will only be displayed after issuing the command display.display().

```
display.drawCircle(X, Y, R, WHITE);
```
Put the center of a white circle with a radius of R at (X,Y) in the display's buffer.

```
display.fillRect(X, Y, W, H, WHITE);
```
Put a filled white rectangle with the top left corner at coordinate: X, Y and a width of W height of H in the display's buffer.

```
display.drawRoundRect(X, Y, W, H, R, WHITE);
```
Put a white rectangle with the top left corner at coordinate: X, Y and a width of W height of H in the display's buffer. The corners are rounded with radius R.

```
display.drawLine(X1, Y1, X2, Y2, WHITE);
```
Put a straight line from X1,Y1 to X2,Y in the display's buffer.

```
display.height()
```
The result of this function is the height of your display (so either 32 or 64).

```
display.width()
```
The result of this function is the width of your display (so in this case 128).

```
display.setTextSize(X);
```
Set the scaling factor of all following text commands to X.

[1] You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

```
display.setTextColor(WHITE);
```
Set the color of all following text commands to WHITE.

```
display.setTextColor(BLACK, WHITE);
```
Invert the text color, to BLACK on a WHITE background. This doesn't work with all fonts.

```
display.setCursor(X, Y);
```
Set the cursor to X,Y.

```
display.println("Hello");
```
Puts the text "Hello" in the display's buffer.

```
display.setFont(&FreeSans9pt7b);
```
Set a pointer to one of the included fonts.

```
display.setFont();
```
Set the font to the default build-in font.

63.5. Sample 0.91 inch 128x32 OLED display (I2C)

The following sketch alternates between displaying some graphics and some text.

Sample Connections

- Connect VCC to 3.3V (to be safe)
- Connect GND to GND.
- Connect SCK to A5 (SCL)
- Connect SDA to A4 (SDA)

030_OLED_Graphic.ino

```
#include <Adafruit_SSD1306.h>
#include <Fonts/FreeSans9pt7b.h>

Adafruit_SSD1306 display(128, 32, &Wire);

void setup() {
  //Serial.begin(9600);

  display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
  display.display();
  delay(1000); // Pause for 2 seconds
  display.clearDisplay();
}

void loop()
{
  display.clearDisplay();
  display.fillCircle(45, 20, 10, WHITE);
  display.fillCircle(48, 17, 2, BLACK);
  display.fillRect(10, 10, 10, 10, WHITE);
  display.drawRoundRect(111, 12, 15, 15, 5, WHITE);
  display.drawLine(0, 0, display.width()-1, display.height() - 1, WHITE);
  display.display();
  delay(1000);
  display.clearDisplay();
  display.setFont();
  display.setTextSize(3);
  display.setTextColor(WHITE);
  display.setCursor(25, 0);
  display.println("Bye");
  display.setTextColor(BLACK, WHITE);
  display.setTextSize(1);
  display.println("world");
  display.setTextColor(WHITE);
  display.setFont(&FreeSans9pt7b);
  display.setCursor(80, 30);
  display.println("OLED");
  display.display();
  delay(1000);

}
```

64. Waveshare 2.7 e-Paper HAT with an Arduino


Three color 2.7 inch e-paper display HAT designed for Raspberry Pi, but can also be used on other Micro computers and Micro Controllers like Arduino, ESP8266 and STM32 devices, through an 8 pin SPI connection. This chapter describes the combination of this HAT with Arduino.

64.1. Specifications Waveshare 2.7 e-Paper HAT with an Arduino

- SKU: 13357
- Interfaces:
  - Standard Raspberry Pi 40 pin GPIO extension header
  - 8 pin SPI connector (3-wire or 4-wire SPI)
- Display:
  - 264 x 176 pixels
  - Three-color display: red, black and white, 2 gray levels
  - Dot pitch: 0.217 x 0.217
  - 57.288 x 38..192mm
  - Full refresh time: 15 seconds
  - Viewing angle: > 170
- 4 onboard keys
- Onboard voltage translator, compatible with 3.3V/5V MCU's
- Dimensions: 85 x 56mm.
- Standby power: < 0.017mW

64.2. Datasheet Waveshare 2.7 e-Paper HAT with an Arduino

- Datasheet: https://www.waveshare.com/w/upload/d/d8/2.7inch-e-paper-b-specification.pdf
- Manual https://www.waveshare.com/w/upload/3/31/2.7inch_e-paper_hat_user_manual_en.pdf
- Wiki https://www.waveshare.com/wiki/2.7inch_e-Paper_HAT_(B)

64.3. Connections Waveshare 2.7 e-Paper HAT with an Arduino

SPI connector

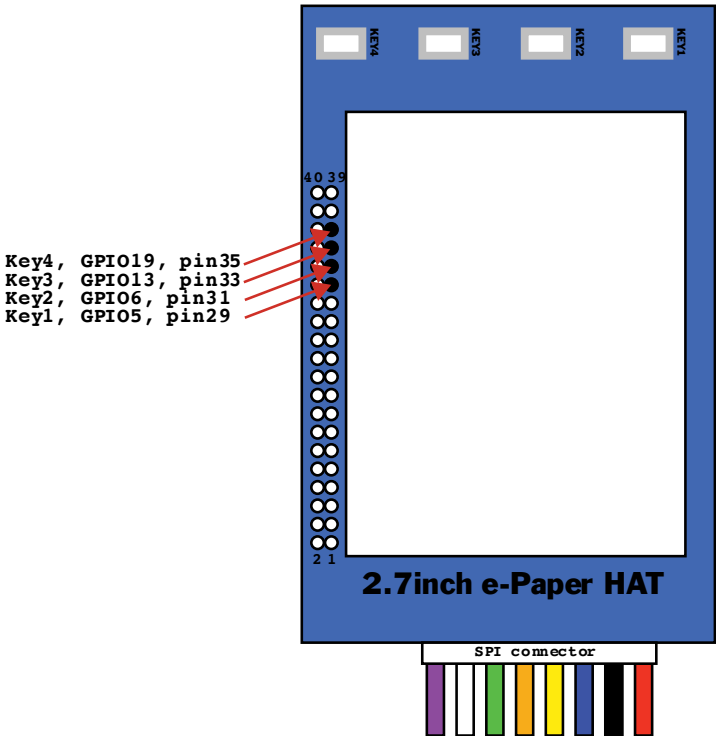
SPI Pin nr	Name	Color	Description	Arduino pin

1	VCC	Red	Power Supply	3.3V
2	GND	Black	Ground	GND
3	DIN	Blue	SPI Mosi	D11
4	CLK	Yellow	SPI SCK	D13
5	CS	Orange	SPI chip selection	D10
6	DC	Green	Data/Command Selection	D9
7	RST	White	Reset	D8
8	Busy	Purple	Busy	D7

Raspberry PI GPIO header

The four keys at the edge of the module are not available through the SPI connector. To use these keys you'll need to connect the correct ports on the GPIO header.

Key	GPIO pin	GPIO port BCM	GPIO port WiringPi
Key1	pin 29	GPIO 5	GPIO 21
Key2	pin 31	GPIO 6	GPIO 22
Key3	pin 33	GPIO 13	GPIO 23
Key4	pin 35	GPIO 19	GPIO 24



64.4. Libraries and samples for Waveshare 2.7 e-Paper HAT with an Arduino

- git clone https://github.com/waveshare/e-Paper

Library use explanation

Before I explain the library, I will first explain the concept of drawing on this e-paper display:

- The width (176) and height (264) are defined in the library epd2in7b.h and should not be changed.

- You can send an image for the whole display at once, but that will consume 176*264/8 = 5808 bytes. The image should then be stored in a hexadecimal array (take a look at the sample that is included with the library).
- You can also change a part of the display only, by declaring the size of that part (multiples of 8)
- You can then prepare the background of the partial image as UNCOLORED or COLORED. This display is a 3 colored display, with White being UNCOLORED and Black and Red being COLORED. Choosing between red and black is done with the command that sends the partial image to the display.
- You can then prepare this partial image with one or more of the following:
  - Single pixel
  - Single char
  - String
  - Line
  - Rectangle
  - Filled rectangle
  - Circle
  - Filled circle
- After this you can send the prepared partial image to a specific coordinate on the display.

```
#include <SPI.h>
```
Include SPI library.

```
#include <epd2in7b.h>
```
Include the epd2in7bc three color e-paper display library.

```
#include <epdpaint.h>
```
Include the epdpaint library.

```
#define COLORED     1
```
Set a constant COLORED to 1.

```
#define UNCOLORED   0
```
Set a constant UNCOLORED to 0.

```
Epd epd;
```
Creates an instance 'epd' of the type Epd (this is the display).

```
if (epd.Init() != 0)
```
With this condition you can handle display failures.

```
epd.ClearFrame();
```
Clear the content of the display.

```
unsigned char image[1024];
```
Declaration of the char array 'image'. This will be used to store the partial image.

```
Paint paint(image, 88  , 48);
```
Creates an instance 'paint' of the type Paint, using the char array 'image' and with a dimension of 88x48 pixels.

```
paint.Clear(COLORED);
```

Prepares the partial image and set all pixels to COLORED (black or red, this will be determined with the command that sends the partial image to the display.

```
paint.DrawRectangle(30, 20, 80, 45 , UNCOLORED);
```

Prepares an unfilled rectangle (from [30,20] to [80,45]) in the partial image and sets the line color to UNCOLORED (white).

```
epd.TransmitPartialBlack(paint.GetImage(), 80 , 132, paint.GetWidth(), paint.GetHeight());
```

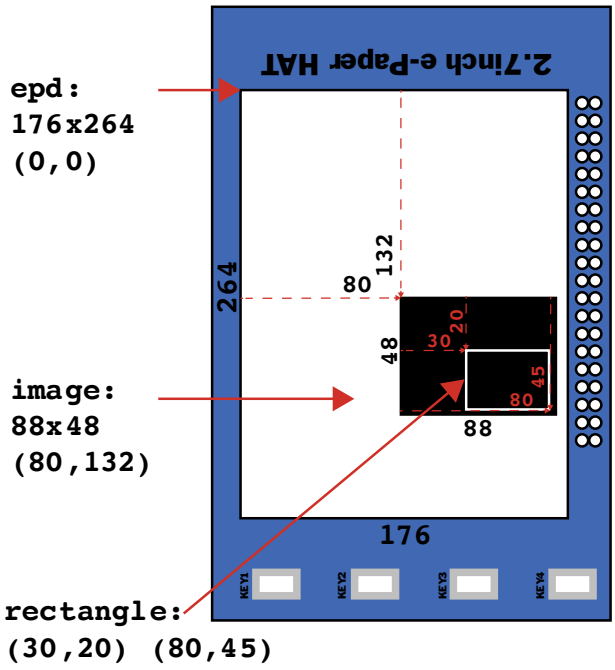
Sends the prepared partial image to the display's memory (SRAM) using [80,132] as the top left coordinate and using paint.GetWidth() and paint.GetHeight() as the size of this partial image. With this command all parts of the partial image that needs to be colored will be Black.

```
epd.DisplayFrame();
```

Displays the data from the SRAM in the e-paper module. The whole display will be refreshed (also partial images that has already been diplayed at a previous excecution of the epd.DisplayFrame() command.. Black will be refreshed first and then red, so using red, will double the refresh-time of your display. Use red with care.

```
epd.Sleep();
```

Sets the display in deep sleep mode. This will save battery but also prolongs the life of your e-paper display. Use this command if you want to maintain the content of your display.



Other functions for placing objects in the partial image

```
DrawAbsolutePixel(<X>, <Y>, <COLOR>);
DrawPixel(<X>, <Y>, <COLOR>);
DrawCharAt(<X>, <Y>, <CHARACTER>, <FONT>, <COLOR>);
DrawStringAt(<X>, <Y>, <STRING>, <FONT>, <COLOR>);
DrawLine(<X0>, <Y0>, <X1>, <Y1>, <COLOR>);
DrawHorizontalLine(<X>, <Y>, <WIDTH>, <COLOR>);
DrawVerticalLine(<X>, <Y>, <HEIGHT>, <COLOR>);
DrawRectangle(<X0>, <Y0>, <X1>, <Y1>, <COLOR>);
DrawFilledRectangle(<X0>, <Y0>, <X1>, <Y1>, <COLOR>);
DrawCircle(<X>, <Y>, <RADIUS>, <COLOR>);
DrawFilledCircle(<X>, <Y>, <RADIUS>, <COLOR>);
```

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

64.5. Sample Waveshare 2.7 e-Paper HAT with an Arduino

The following sketch

Sample Connections

- Connect Red VCC to 3.3V
- Connect Black GND to GND
- Connect Blue DIN to D11 (MOSI)
- Connect Yellow CLK to D13 (SCK)
- Connect Orange CS to D10
- Connect Green DC to D9
- Connect White RST to D8
- Connect Purple Busy to D7

155_epaper_27b.ino

```
#include <SPI.h>
#include <epd2in7b.h>
#include <epdpaint.h>
#define COLORED     1
#define UNCOLORED   0
Epd epd;
unsigned char image[1024];

void setup()
{
  Serial.begin(9600);
  if (epd.Init() != 0) {
    Serial.print("e-Paper init failed");
    return;
  }
  epd.ClearFrame();
}

void loop()
{
  Paint paint(image, 88  , 48);    //width should be the multiple of 8
  paint.Clear(COLORED);
  paint.DrawRectangle(30, 20, 80, 45 , UNCOLORED);
  paint.DrawStringAt(0, 0, "eve_arduino", &Font12, UNCOLORED);
  epd.TransmitPartialBlack(paint.GetImage(), 80 , 132, paint.GetWidth(),
paint.GetHeight());
  epd.DisplayFrame();
  epd.Sleep();
  while(true)
  {

  }
}
```

Input sensors

In this section you will find several input sensors, switches, buttons, potentiometers even Nintendo's Nunchuk.

65. Switches

There are several types of switches, but most of them can be used in the same way, they either open or close one or more connections when activated (pressed or switched).

Very nice information about working with switches can be found at http://www.ladyada.net/learn/arduino/lesson5.htm.

65.1. Specifications Switches

Push button NO

This pushbutton is normally off (NO). A connection will be made as long as the button is pressed down.

The switch above has 4 terminals that are two pairs.

Push button Changeover (NO NC)

Depending on the connections that are used, this pushbutton is normally off, or normally on. If you use all 3 connections, you can alternately make a connection and at the same time break another connection (Changeover).

The switch above has 6 terminals that can be switched as 2 changeover switches (front row and back row). Used on a solder less breadboard only 1 switch can be used.

Toggle switch Changeover

Depending on the position of the switch, a connection is made or broken.

Dipswitches

This is a row of multiple switches in a row, normally used to alter settings.

Tilt Switch Sensor

A tilt switch acts like all other switches, only the way to activate it differs. You must tilt the switch instead of pushing it.

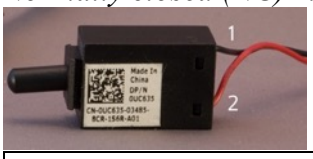
- Straight up, the legs are connected, 90 degrees tilted, the legs are disconnected.

65.2. Overview of my switches

Most of my switches were salvaged from broken electronic devices. In this paragraph I've collected all of them with their pinout.

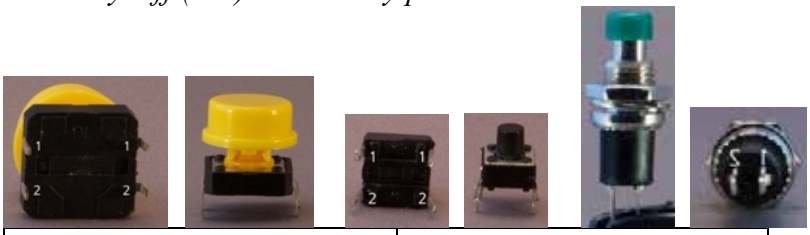
Switches with 2 different positions

Normally closed (NC) momentary push switch



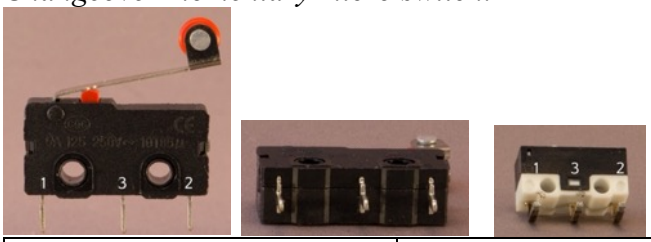
Positions	2
Common pin	n.a
Rest position	OUT
Position: OUT	none
Position: IN	1-2

Normally Off (NO) momentary push switch



Positions	2
Common pin	n.a
Rest position	OUT
Position: OUT	none
Position: IN	1-2

Changeover momentary micro switch.

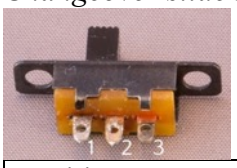


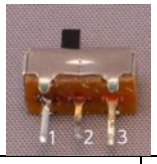
Positions	2
Common pin	1
Rest position	UP
Position: UP	1-2
Position: DOWN	1-3

Changeover toggle switch

Positions	2
Common pin	2
Rest position	either LEFT/RIGHT
Position: LEFT	2-3
Position: RIGHT	2-1

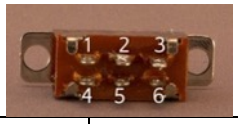
Changeover slide switch

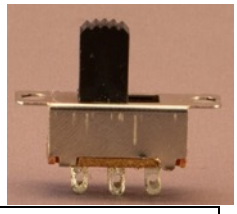




Positions	2
Common pin	2
Rest position	either LEFT/RIGHT
Position: LEFT	2-1
Position: RIGHT	2-3

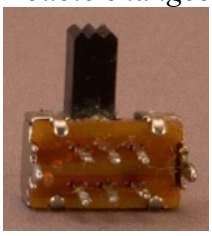
Double changeover toggle switch
Double changeover slide switch I



Positions	2
Common pin	2 and 5 isolated
Rest position	either LEFT/RIGHT
Position: LEFT	2-3 5-6
Position: RIGHT	2-1 5-4

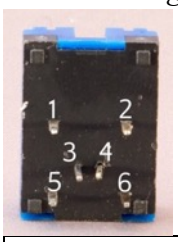
Double changeover slide switch II



Compared to the previous double changeover slide switch, the connections made with LEFT and RIGHT are swapped.

Positions	2
Common pin	2 and 5 isolated
Rest position	either LEFT/RIGHT
Position: LEFT	2-1 5-4
Position: RIGHT	2-3 5-6

Double changeover momentary micro switch



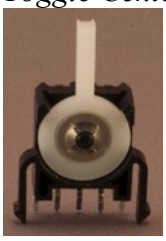
Positions	2
Common pin	1 and 2 isolated
Rest position	UP
Position: UP	1-5 2-6
Position: DOWN	1-3 2-4

Switches with 3 different positions

Toggle Center-off switch I

Positions	3
Common pin	2
Rest position	either LEFT, CENTER, RIGHT
Position: LEFT	2-3
Position: CENTER	none
Position: RIGHT	2-1

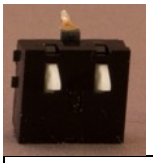
Toggle Center-off switch II



Compared to the previous "Toggle Center-off switch", the connections made with LEFT and RIGHT are swapped.

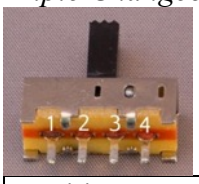
Positions	3
Common pin	2
Rest position	either LEFT, CENTER, RIGHT
Position: LEFT	2-1
Position: CENTER	none
Position: RIGHT	2-3

Two momentary micro switches




Positions	4
Common pin	2
Rest position	both UP
Position: both UP	none
Position: LEFT DOWN	2-3
Position: RIGHT DOWN	2-1
Position: both DOWN	1-2-3

Triple Changeover slide switch



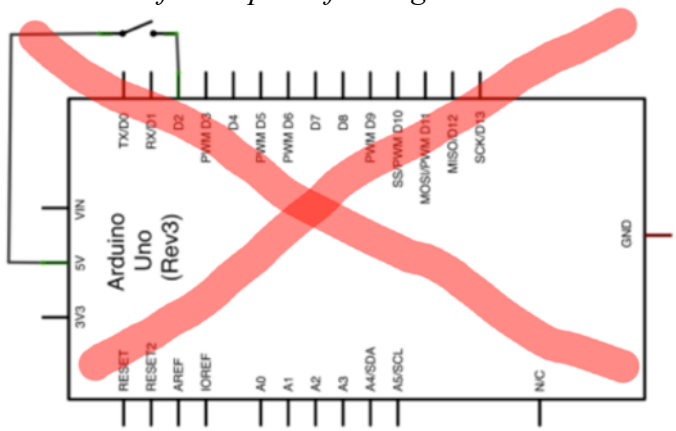
Positions	3
Common pin	none
Rest position	either LEFT, CENTER, RIGHT
Position: LEFT	1-2
Position: CENTER	2-3
Position: RIGHT	3-4

65.3. Libraries needed for Switches

None needed.

65.4. Sample Switches

If you connect one leg of the switch to 5V and the other to a digital I/O port, then the status of a closed switch will be recognized as a HIGH input, but when the switch is open, the state of the input is not known. There is neither GND connected to the input (LOW), nor 5V (HIGH), so the state of the input is floating and the LED will blink erratically.



D2 is floating when switch is open and D2 is HIGH when switch is closed.

003_ButtonUnreliable.ino

```
/* No Pullup, unreliable!!!!!! */
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT);
}

void loop(){
  int Buttonstate = digitalRead(2);
  if (Buttonstate == HIGH) //When button is pressed, buttonstate == HIGH!
  {
    digitalWrite(13, HIGH);
  }
  else
  {
    digitalWrite(13, LOW);
  }
}
```

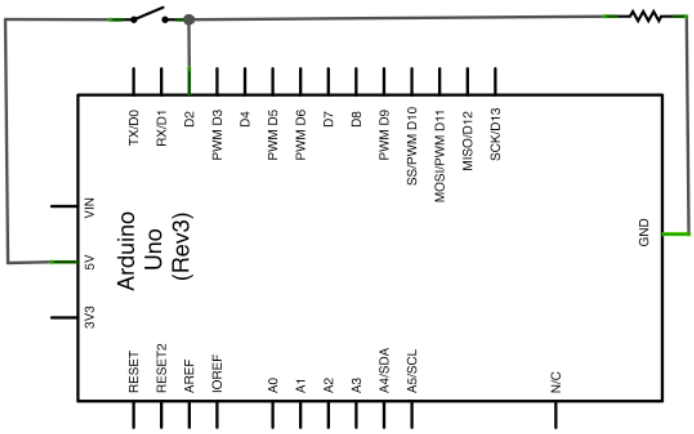
Don't use these schematic or this sketch!!

To prevent this floating, there are 3 solutions:

- Pull-down resistor of 10K ohm between Ground and the digital input.
- Pull-up resistor of 10K ohm between 5V and the digital input.
- Activate an internal pull-up resistor (20K ohm).

10K ohm Pull-down resistor

- Connect a 10K ohm resistor between the digital input and GND.
- Connect a switch between the digital input and 5V.
- Closing the switch makes the input HIGH.



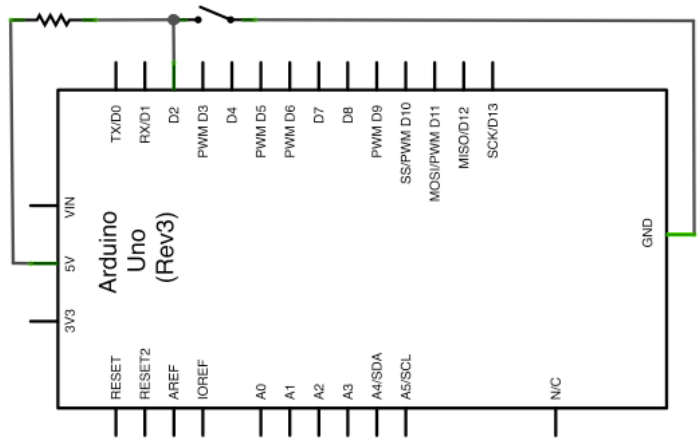
D2 is LOW when switch is open and D2 is HIGH when switch is closed.

031_ButtonExtPulldown.io

```
/* EXTERNAL PULLDOWN */
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT); //This line is needed for external pullup/pulldown
}

void loop(){
  int Buttonstate = digitalRead(2);
  if (Buttonstate == HIGH) //When button is pressed, buttonstate == HIGH!
  {
    digitalWrite(13, HIGH);
  }
  else
  {
    digitalWrite(13, LOW);
  }
}
```

10K ohm Pull-up resistor plus sample sketch

- Connect a 10K ohm resistor between the digital input and 5 V.
- Connect the switch between the digital input and GND.
- In this case closing the switch makes the input LOW (the opposite from the Pull-down schematic, so the sketch was altered accordingly).



D2 is HIGH when switch is open and D2 is LOW when switch is closed

032_ButtonExtPullup.ino

```
/* EXTERNAL PULLUP */
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT); //This line is needed for external pullup/pulldown
}

void loop(){
  int Buttonstate = digitalRead(2);
  if (Buttonstate == LOW) //When button is pressed, buttonstate == LOW!
  {
    digitalWrite(13, HIGH);
  }
  else
  {
    digitalWrite(13, LOW);
  }
}
```

Internal Pull-up resistor (20K ohm)

The MEGA328 has internal Pull-up resistors of 20K ohm available for every digital I/O. To activate such an internal Pull-up resistor you have the define the pinMode to INPUT_PULLUP instead of INPUT.

- So activate the internal Pull-up resistor.

```
pinMode(2, INPUT_PULLUP);
```

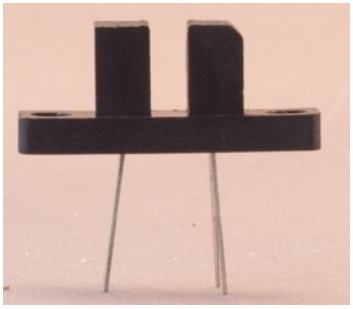
- Connect the switch between the digital input and GND.
- Just as with the external pull-up resistor closing the switch makes the input LOW.

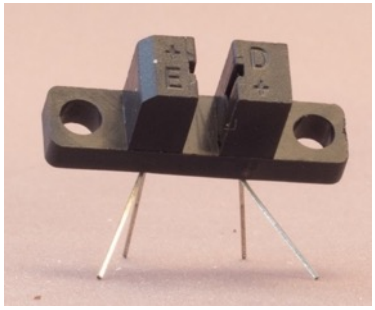
D2 is HIGH when switch is open and D2 is LOW when switch is closed.

033_ButtonIntPullup.ino

```
/* INTERNAL PULLUP */
void setup()
{
  pinMode(13, OUTPUT);
  pinMode(2, INPUT_PULLUP); //This line is only needed for internal pullup
}

void loop(){
  int Buttonstate = digitalRead(2);
  if (Buttonstate == LOW) //When button is pressed, buttonstate == LOW!
  {
    digitalWrite(13, HIGH);
  }
  else
  {
    digitalWrite(13, LOW);
  }
}
```

66. Photo-interrupter ITR8102





This optical switch consists of two components: an IR LED (Emitter side) and an IR phototransistor (Detector side). The connection will be open when the light beam is interrupted, and closes when the light reaches the phototransistor.

66.1. Specifications Photo-interrupter ITR8102

- Input Reverse Voltage: 5V
- Input Forward Current: 20 mA
- Input Forward Voltage: 1.2V (max 1.6V)
- Output Collector Current: 20 mA
- Output Collector-Emitter Voltage: 30 V
- Output Emitter-Collector Voltage: 5 V

66.2. Datasheet Photo-interrupter ITR8102

- http://arduineando.matem.unam.mx/datasheets/ITR8102.pdf

66.3. Connections Photo-interrupter ITR8102

Pin nr	Name	Description	Arduino pin
1	Anode	+	5V
2	Cathode	E (emitting side)	To ground through a 220 ohm resistor
3	Collector	+	5V
4	Emitter[1]	D (detector side)	To Ground through a 10K ohm resistor & To Any Digital port

66.4. Libraries needed for Photo-interrupter ITR8102

None needed.

[1] This is the emitter of the phototransistor, not to be mistaken by the light emitter (emitting side of the opto switch).

66.5. Sample Photo-interrupter ITR8102

The following sketch will switch off the LED on D13 when the light beam is interrupted.

Sample Connections

- Connect + (at E-side) to 5V.
- Connect E to one end of a 220 ohm resistor.
- Connect other end of the 220 ohm resistor to GND.
- Connect + (at D-side) to 5V.
- Connect D to D12.
- Connect D to one end of a 10K ohm resistor.
- Connect other end of the 10K ohm resistor to GND.

034_Optical_ITR8102.ino

```
int Sensor = 12;
int Led    =  13;

int SensorValue = 0;

void setup()
{
  pinMode(Led, OUTPUT);
  pinMode(Sensor, INPUT);
}

void loop()
{
  SensorValue = digitalRead(Sensor);
  if (SensorValue == HIGH)
  {
    digitalWrite(Led, HIGH);
  }
  else
  {
    digitalWrite(Led, LOW);
  }
}
```

67. Photo-interrupter RPI-352


This optical switch consists of two components: an IR LED (Emitter side) and an IR phototransistor (Detector side). The connection will be open when the light beam is interrupted, and closes when the light reaches the phototransistor.

67.1. Specifications Photo-interrupter RPI-352

- Gap between emitter and detector: 3mm
- Input Reverse Voltage: 5V
- Input Forward Current: 50 mA
- Input Forward Voltage: 1.3V (max 1.6V)
- Output Collector Current: 30 mA
- Output Collector-Emitter Voltage: 30 V
- Output Emitter-Collector Voltage: 4,5 V

67.2. Datasheet Photo-interrupter RPI-352

- http://rohmfs.rohm.com/en/products/databook/datasheet/opto/optical_sensor/photointerrupter/rpi-352.pdf

67.3. Connections Photo-interrupter RPI-352

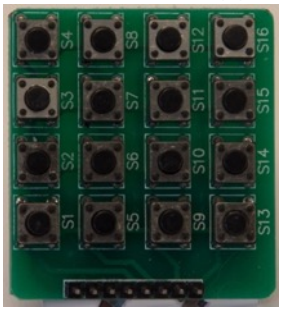
Pin nr	Name	Description	Arduino pin
1	Cathode	E (emitting side)	To ground through a 220 ohm resistor
2	Anode	+	5V
3	Collector	+	5V
4	Emitter[1]	D (detector side)	To Ground through a 10K ohm resistor & To Any Digital port

67.4. Libraries needed for Photo-interrupter RPI-352

None needed.

[1] This is the emitter of the phototransistor, not to be mistaken by the light emitter (emitting side of the opto switch).

68. 4x4 Keypad



68.1. Specifications 4x4 Keypad

- 16 NO push buttons, connected through a 8 pin header row.
- Starting from Library version 3.0 it is possible to detect multiple keys at the same time. A caveat is that not all keys are available.

68.2. Connections 4x4 Keypad

Pin nr	Name	Description	Arduino pin
1	ROW0	Row 0 (top)	Any Digital port
2	ROW1	Row 1	Any Digital port
3	ROW2	Row 2	Any Digital port
4	ROW3	Row 3 (bottom)	Any Digital port
5	COL0	Column 0 (left)	Any Digital port
6	COL1	Column 1	Any Digital port
7	COL2	Column 2	Any Digital port
8	COL3	Column 3 (right)	Any Digital port

68.3. Libraries needed for 4x4 Keypad

- Keypad library from Mark Stanley and Alexander Brevig through the Library Manager.

Library use explanation

```
#include <Keypad.h>
```

Include keypad library from Mark Stanley.

```
char keys[4][4] = {
  {'1','2','3', '+'},
  {'4','5','6', '-'},
  {'7','8','9', '*'},
  {'C','0','=', '/'}
};
```

Create a 2-dimensional array containing the labels for the 4x4 keys.

```
byte rowPins[ROWS] = { ROW0, ROW1, ROW2, ROW3 };
```

Create array containing the digital pin numbers connected to ROW0..ROW3.

```
byte colPins[COLS] = { COL0, COL1, COL2, COL3 };
```

Create an array containing the digital pin numbers connected to COL0..COL3.

```
Keypad mykpd = Keypad( makeKeymap(keys), rowPins, colPins, 4, 4 );
```

Create a new instance of the object Keypad, using the keys-, rowPins-, colPins-arrays and the size of the keypad as parameters.

```
char key = mykpd.getKey();
if(key)
{
  Serial.println(key);
}
```

Read a key from mykpd and if a key is pressed, print the label on the serial port.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

68.4. Sample 4x4 Keypad

The following sketch prints the labels of the keys on the serial monitor.

Sample Connections

- Connect COL0 to D2.
- Connect COL1 to D3.
- Connect COL2 to D4.
- Connect COL3 to D5.
- Connect ROW0 to D6.
- Connect ROW1 to D7.
- Connect ROW2 to D8.
- Connect ROW3 to D9.

035_Keypad.ino

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] =
{
  {'1','2','3', '+'},
  {'4','5','6', '-'},
  {'7','8','9', '*'},
  {'C','0','=', '/'}
};

byte rowPins[ROWS] = { 2, 3, 4, 5 };
byte colPins[COLS] = { 6, 7, 8, 9 };

Keypad kpd = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

#define ledpin 13

void setup()
{
  pinMode(ledpin,OUTPUT);
  digitalWrite(ledpin, HIGH);
  Serial.begin(9600);
}

void loop()
{
  char key = kpd.getKey();
  if(key)
  {
    Serial.println(key);
  }
}
```

69. Potentiometer



A potentiometer is a three connector resistor with a sliding contact. If you use the sliding contact and one of the outer connectors it acts as variable resistor. If you use all three connectors, it acts as an adjustable voltage divider. In most Arduino projects all 3 connectors are used to create an analog input device.

69.1. Specifications Potentiometer

In most projects the value of the potentiometer is not critical. A value of 1K to 10 K ohm is very common.

69.2. Connections Potentiometer

Pin nr	Name	Description	Arduino pin
1	left	Ground	GND
2	middle	Signal	Any Analog input port
3	right	VCC	5V

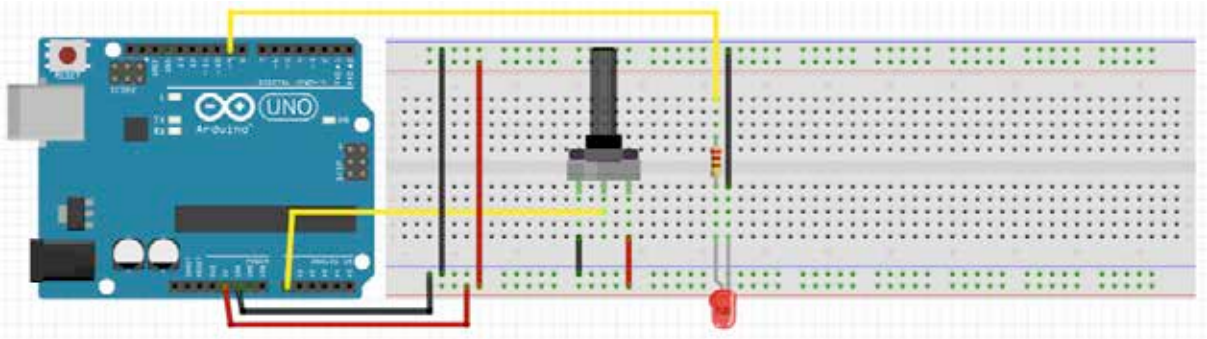
69.3. Libraries needed for Potentiometer

None needed.

69.4. Sample sketch Potentiometer

Connections

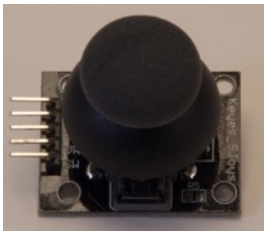
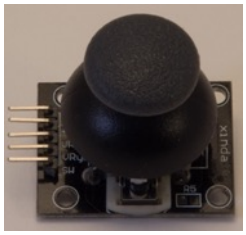
- Connect Left pin with GND.
- Connect Middle pin with A0.
- Connect Right pin with 5V.
- Connect one end of a 220 ohm resistor to D9 and the other end to the Anode of a LED.
- Connect the Cathode of the LED to GND.



036_Potentiometer.ino

```
const int analogInPin = A0;
const int LED = 9;

int sensorValue = 0;
int outputValue = 0;

void setup()
{
}

void loop()
{
  sensorValue = analogRead(analogInPin);
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(LED, outputValue);
  delay(2);
}
```

70. Joystick

This chapters describes 2 different Joysticks, 1 labeled Keyes and the other labeled Xinda. Differences are only in their resistance. This is not interesting for your sketches on Arduino, because you don't measure an absolute resistance value. With the sliding contact of a potentiometer to one side, the analog input measures 0 (0 V). and with the sliding contact to the opposite side, it measures 1023 (5 V).

70.1. Specifications Joystick

Xinda

- Orientation:
  Hold the 5 pin header to the left for the right orientation of X and Y:
- X axis:
  - Full left: 80 ohm.
  - Neutral: 3K3 ohm
  - Full right: 4K4 ohm
- Y axis:
  - Full up: 70 ohm
  - Neutral: 3K4 ohm
  - Full down: 4K4
- By pressing the joystick towards the PCB, the switch will be closed and connected to GND (LOW signal). So you should use a pull-up resistor (either a real or an internal resistor) so that when the switch is not pressed there is a connection to 5V through the resistor (HIGH signal).

Keyes

- X axis:
  - Full left: 60 ohm.
  - Neutral: 3K6 ohm
  - Full right: 4K7 ohm
- Y axis:
  - Full up: 60 ohm
  - Neutral: 3K6 ohm
  - Full down: 4K7

70.2. Connections Joystick

This joystick is a combination of 2 potentiometers and 1 switch.

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	VCC	5/3,3 V spanning	5V
3	VRx	Potentiometer X-as	Any analog port
4	VRy	Potentiometer Y-as	Any analog port
5	SW	Switch (connected to GND)	Any digital port

70.3. Libraries needed for Joystick

None needed.

70.4. Sample Joystick

The following sketch prints the status of the X and Y joystick and the status of the switch (1= not pressed, 0= pressed).

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect VRx to A0.
- Connect VRy to A1.
- Connect SW to D5.

Since SW switches D5 to ground, a Pull-up resistor is needed. In the following sketch the internal Pull-up resistor is activated.

037_Joystick.ino

```
const int JoyX = A0;
const int JoyY = A1;
const int JoyS = 5;
int sensorY;
int sensorX;
int Switch=9;
void setup()
{
  Serial.begin(9600);
  pinMode(5, INPUT_PULLUP);
}

void loop()
{
  sensorX = analogRead(JoyX);
  sensorY = analogRead(JoyY);
  Switch = digitalRead(JoyS);
  Serial.print("X: ");
  Serial.print(sensorX);
  Serial.print(" Y: ");
  Serial.print(sensorY);
  Serial.print(" S: ");
  Serial.println(Switch);
}
```

71. Rotary encoder


Although a rotary encoder looks like a potentiometer, it is far from that. Instead of changing a resistor value, it sends out pulses, Those pulses differ depending on the direction in which you turn the encoder. It also has a continuous 360 degree motion. This specific rotary encoder is also equipped with a button, that can be activated by pressing on the knob.

Rotary encoders are often used to scroll up and down through list.

71.1. Specifications Rotary encoder

- 2-bit Gray code
- 20 pulses per rotation
- Max. rating: 10 mA at 5V
- Button

71.2. Datasheet Rotary encoder

I don't have a datasheet for this specific Rotary encoder.

71.3. Connections Rotary encoder

Pin nr	Name	Description	Arduino pin
1	Output A		Any Digital port, preferably an Interrupt port
2	GND	Ground	GND
3	Output B		Any Digital port, preferably an Interrupt port
4	Button	One leg of the button	Depends: • to GND: when using an internal Pullup • to GND through an external Pullup resistor • to VCC through an external Pullup resistor
5	Button	Other leg of the button	Any Digital port

71.4. Libraries needed for Rotary encoder

No libraries are needed.

71.5. Sample Rotary encoder

The following sketch shows the total number of steps the rotary encoder has turned from the start position. With the button, you can reset the start position.
The original sketch can be found at: https://circuitcrush.com/arduino/2013/09/09/using-rotary-encoder-with-arduino.html

Sample Connections

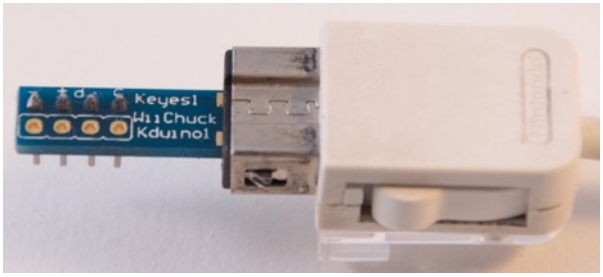
- Connect pin 1 to D2
- Connect pin 2 to GND
- Connect pin 3 to D3
- Connect pin 4 to GND
- Connect pin 5 to D8

165_Rotary_encoder.ino

```
#define ENCODERPIN_A 2
#define ENCODERPIN_B 3
#define CLEARBUTTON 8

volatile int encoderPos = 0;
int lastReportedPos = 1;
static boolean rotating = false;

boolean A_set = false;
boolean B_set = false;

void setup() {
  pinMode(ENCODERPIN_A, INPUT_PULLUP);
  pinMode(ENCODERPIN_B, INPUT_PULLUP);
  pinMode(CLEARBUTTON, INPUT_PULLUP);
  attachInterrupt(0, doEncoderA, CHANGE);
  attachInterrupt(1, doEncoderB, CHANGE);
  Serial.begin(9600);
}


void loop() {
  rotating = true;
  if (lastReportedPos != encoderPos)
  {
    Serial.print("Index:");
    Serial.println(encoderPos, DEC);
    lastReportedPos = encoderPos;
  }
  if (digitalRead(CLEARBUTTON) == LOW )
  {
    encoderPos = 0;
  }
}

void doEncoderA()
{
  if ( rotating ) delay (1);
  if ( digitalRead(ENCODERPIN_A) != A_set )
  {
    A_set = !A_set;
    if ( A_set && !B_set )
    {
      encoderPos += 1;
    }
```

```
    rotating = false;
  }
}

void doEncoderB()
{
  if ( rotating ) delay (1);
  if ( digitalRead(ENCODERPIN_B) != B_set )
  {
    B_set = !B_set;
    if ( B_set && !A_set )
    {
      encoderPos -= 1;
    }
    rotating = false;
  }
}
```

72. Nunchuk with connection board



72.1. Specifications Nunchuk with connection board

- 2-Way analog joystick
- 2 buttons
- 3 axis accelerometer
- Communication through Fast I²C at 400 kHz.

72.2. Connections Nunchuk with connection board

Pin nr	Name	Description	Arduino pin
1	-	Ground	GND
2	+	3.3 V	3.3 V
3	d	I²C Data	SDA (A4)
4	c	I²C Clock	SCL (A5)

72.3. Libraries needed for Nunchuk with connection board

- Inter Integrated Circuit (I^2C) and Two Wire Interface (TWI) library, included with Arduino IDE: "Wire.h".
- Improved Arduino Nunchuk library from Gabriel Bianconi. https://github.com/GabrielBianconi/ArduinoNunchuk

Library use explanation

```
#include <Wire.h>
```

Include the Inter-Integrated Circuit (I^2C) and Two Wire Interface (TWI) library.

```
#include <ArduinoNunchuk.h>
```

Include the Improved Arduino Nunchuk library.

```
ArduinoNunchuk mynunchuk = ArduinoNunchuk();
```

Create mynunchuk a new instance of the object type ArduinoNunchuk.

```
Serial.begin(19200);
```

Set the baud rate for the serial port to something higher than the default 9600. Off course this is only needed if you want to print the received Nunchuk values to the serial port.

```
mynunchuk.init();
```

Initialize mynunchuk.

```
mynunchuk.update();
```

Read a value from mynunchuk.

The value read from mynunchuk is a combination of the following Nunchuk-sensors:

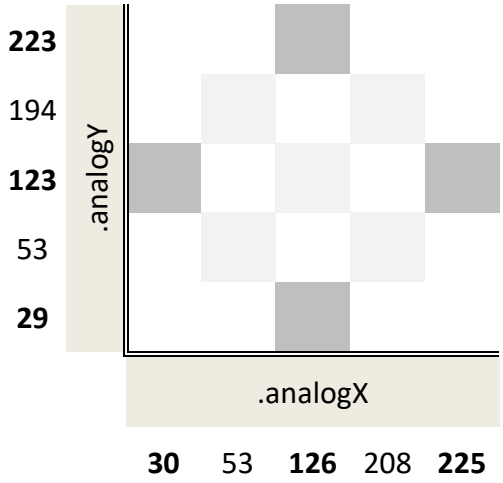
- *Joystick:*
  - *mynunchuk.analogX*
  - *mynunchuk.analogY*
- *Accelerometer:*
  - *mynunchuk.accelX*
  - *mynunchuk.accelY*
  - *mynunchuk.accelZ*
- *Z-button: mynunchuk.zButton*
- *C-button: mynunchuk.cButton*

```
• Serial.print(mynunchuk.analogX, DEC);
• Serial.print(mynunchuk.analogY, DEC);
• Serial.print(mynunchuk.accelX, DEC);
• Serial.print(mynunchuk.accelY, DEC);
• Serial.print(mynunchuk.accelZ, DEC);
• Serial.print(mynunchuk.zButton, DEC);
• Serial.println(mynunchuk.cButton, DEC);
```

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

Values for the analog joystick

The following values for the analog joystick: .analogX and .analogY are not very accurate, but can be used as estimates.



Values for the accelerometer

The accelerometer data uses the full range of 0-1024. However, the full range is only seen when moving or rotating the Nunchuk sharply. All values in the following table are estimates.

Name	Lowest value	Neutral	Highest value
.accelX	280	500	710
.accelY	270	500	700
.accelZ	0 (upside down)	?	?
.zButton	-	0	1 (pressed)
.cButton	-	0	1 (pressed)

72.4. Sample Nunchuk with connection board

The following sample sketch shows the status of the Nunchuk in 7 values on the serial port.

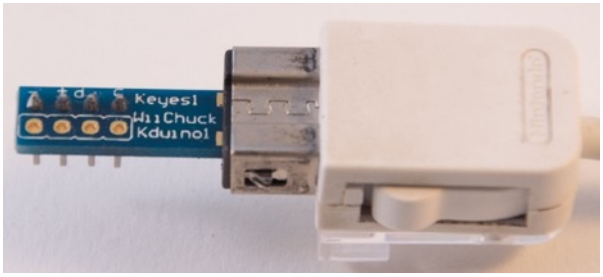
Connections

- Connect the Nunchuk with the Nunchuk connection board.
- Connect – with GND.
- Connect + with 3.3V.
- Connect d with SDA (A4).
- Connect c with SCL (A5).

038_Nunchuk.ino

```
#include <Wire.h>
#include <ArduinoNunchuk.h>

#define BAUDRATE 19200

ArduinoNunchuk nunchuk = ArduinoNunchuk();

void setup()
{
  Serial.begin(BAUDRATE);
  nunchuk.init();
}

void loop()
{
  nunchuk.update();

  Serial.print(nunchuk.analogX, DEC);
  Serial.print(' ');
  Serial.print(nunchuk.analogY, DEC);
  Serial.print(' ');
  Serial.print(nunchuk.accelX, DEC);
  Serial.print(' ');
  Serial.print(nunchuk.accelY, DEC);
  Serial.print(' ');
  Serial.print(nunchuk.accelZ, DEC);
  Serial.print(' ');
  Serial.print(nunchuk.zButton, DEC);
  Serial.print(' ');
  Serial.println(nunchuk.cButton, DEC);
}
```

73. Nunchuk with connection board on A2..A5



It is possible to simulate GND and 3.3V on 2 analog ports. By doing this on A2 (GND) and A3 (3.3V), it is possible to connect the Nunchuk connection board directly on A2..A5. You need to use another Nunchuk library for this (WiiChuck.h from Tim Hirzel).

73.1. Connections Nunchuk with connection board on A2..A5

Pin nr	Name	Description	Arduino pin
1	-	Ground	A2
2	+	3.3 V??	A3
3	d	I^2C Data	SDA (A4)
4	c	I^2C Clock	SCL (A5)

73.2. Libraries needed for Nunchuk with connection board on A2..A5

- Inter Integrated Circuit (I^2C) and Two Wire Interface (TWI) library, included with Arduino IDE: "Wire.h".
- Nunchuk library WiiChuck.h by Tim Hirzel.
  The installation of this library is not straight forward. Create a folder called WiiChuck in your libraries folder. Create a text file called WiiChuck.h inside this folder and past the contents of the following URL in this file:
  http://playground.arduino.cc/Main/WiiChuckClass?action=sourceblock&num=3

Library use explanation

```
#include <Wire.h>
```

Include the Inter-Integrated Circuit (I^2C) and Two Wire Interface (TWI) library.

```
#include "WiiChuck.h"
```

Include the Nunchuk library from Tim Hirzel.

```
WiiChuck chuck = WiiChuck();
```

Create chuck a new instance of the object type WiiChuck.

```
Serial.begin(115200);
```

Set the baud rate for the serial port to something higher than the default 9600. Off course this is only needed if you want to print the received Nunchuk values to the serial port.

```
chuck.begin();
```

Initialize chuck.

```
chuck.update();
```

Read a value from chuck.

The values red from chuck is a combination of the following Nunchuk-sensors:

- *Joystick:*

    - *chuck.readJoyX()*
    - *chuck.readJoyY()*
- *Accelerometer:*
    - *chuck.readAccelX()*
    - *chuck.readAccelY()*
    - *chuck.readAccelZ()*
- *Roll: chuck.readRoll()*
- *Pitch: chuck.readPitch()*
- *Z-button: chuck.buttonZ*
- *C-button: chuck.buttonC*

```
Serial.print((int)chuck.readJoyX());
Serial.print((int)chuck.readJoyY());
Serial.print((int)chuck.readAccelX());
Serial.print((int)chuck.readAccelY());
Serial.print((int)chuck.readAccelZ());
Serial.print((int)chuck.buttonZ);
Serial.print((int)chuck.buttonC);
```

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

73.3. Sample Nunchuk with connection board on A2..A5

The following sample sketch shows the status of the Nunchuk in 7 values on the serial port.

Connections

- Connect the Nunchuk with the Nunchuk connection board.
- Connect the Nunchuk connection board directly to A2..A5, with the Nunchuk cable facing away from the Arduino.

039_Nunchuk_A2-A5.ino

```
#include "Wire.h"
#include "WiiChuck.h"

WiiChuck chuck = WiiChuck();

void setup() {
  Serial.begin(115200);
  chuck.begin();
  chuck.update();
}

void loop() {
  delay(20);
  chuck.update();

  Serial.print((int)chuck.readJoyX());
  Serial.print(", ");
  Serial.print((int)chuck.readJoyY());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelX());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelY());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelZ());
  Serial.print(", ");
  Serial.print((int)chuck.buttonZ);
  Serial.print(", ");
  Serial.print((int)chuck.buttonC);

  Serial.println();
}
```

Sample sketch Nunchuk to Processing

This sketch is an example of how to interface your Arduino with Processing on your computer. The serial output from the Arduino is used to rotate a 3D cube on your computer screen.

040_Nunchuk_Processing.ino

```
#include "Wire.h"
#include "WiiChuck.h"

WiiChuck chuck = WiiChuck();

void setup()
{
  Serial.begin(115200);
  chuck.begin();
  chuck.update();
}


void loop()
{
  delay(20);
  chuck.update();
  Serial.print(chuck.readRoll());
  Serial.print(", ");
  Serial.print(chuck.readPitch());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelX());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelY());
  Serial.print(", ");
  Serial.print((int)chuck.readAccelZ());
  Serial.println();
}
```

Processing sketch

You need to enter the number of your Serial Port in the following line (in this example my Serial Port is nr 3, found by trial on error):

```
  myPort = new Serial(this, Serial.list()[3], BAUDRATE);
```

```
float xmag, ymag = 0;
float newXmag, newYmag = 0;
int sensorCount = 5;                          // number of values to expect

import processing.serial.*;
Serial myPort;                // The serial port

int BAUDRATE = 115200;
char DELIM = ','; // the delimeter for parsing incoming data


void setup()
{
  size(200, 200, P3D);
  noStroke();
  colorMode(RGB, 1);
  myPort = new Serial(this, Serial.list()[3], BAUDRATE);
  // clear the serial buffer:
  myPort.clear();
```

```
}
float x, z;

void draw()
{
  background(0.5, 0.5, 0.45);
  pushMatrix();
  translate(width/2, height/2, -30);
  newXmag = mouseX/float(width) * TWO_PI;
  newYmag = mouseY/float(height) * TWO_PI;
  float diff = xmag-newXmag;
  if (abs(diff) >  0.01) {
    xmag -= diff/4.0;
  }
  diff = ymag-newYmag;
  if (abs(diff) >  0.01) {
    ymag -= diff/4.0;
  }
  z = sensorValues[0] / 180 * PI ;
  x = sensorValues[1] / 180 * PI;
  rotateZ(z);
  rotateX(x);
  scale(50);
  beginShape(QUADS);
  fill(0, 1, 1);
  vertex(-1,  1,  1);
  fill(1, 1, 1);
  vertex( 1,  1,  1);
  fill(1, 0, 1);
  vertex( 1, -1,  1);
  fill(0, 0, 1);
  vertex(-1, -1,  1);
  fill(1, 1, 1);
  vertex( 1,  1,  1);
  fill(1, 1, 0);
  vertex( 1,  1, -1);
  fill(1, 0, 0);
  vertex( 1, -1, -1);
  fill(1, 0, 1);
  vertex( 1, -1,  1);
  fill(1, 1, 0);
  vertex( 1,  1, -1);
  fill(0, 1, 0);
  vertex(-1,  1, -1);
  fill(0, 0, 0);
  vertex(-1, -1, -1);
  fill(1, 0, 0);
  vertex( 1, -1, -1);
  fill(0, 1, 0);
  vertex(-1,  1, -1);
  fill(0, 1, 1);
  vertex(-1,  1,  1);
  fill(0, 0, 1);
  vertex(-1, -1,  1);
  fill(0, 0, 0);
  vertex(-1, -1, -1);
  fill(0, 1, 0);
  vertex(-1,  1, -1);
  fill(1, 1, 0);
  vertex( 1,  1, -1);
  fill(1, 1, 1);
  vertex( 1,  1,  1);
```

```
  fill(0, 1, 1);
  vertex(-1,  1,  1);
  fill(0, 0, 0);
  vertex(-1, -1, -1);
  fill(1, 0, 0);
  vertex( 1, -1, -1);
  fill(1, 0, 1);
  vertex( 1, -1,  1);
  fill(0, 0, 1);
  vertex(-1, -1,  1);
  endShape();
  popMatrix();
}
float[] sensorValues = new float[sensorCount];  // array to hold the
incoming values

void serialEvent(Serial myPort) {
  // read incoming data until you get a newline:
  String serialString = myPort.readStringUntil('\n');
  // if the read data is a real string, parse it:

  if (serialString != null) {
    //println(serialString);
    //println(serialString.charAt(serialString.length()-3));
    // println(serialString.charAt(serialString.length()-2));
    // split it into substrings on the DELIM character:
    String[] numbers = split(serialString, DELIM);
    // convert each subastring into an int
    if (numbers.length == sensorCount) {
      for (int i = 0; i < numbers.length; i++) {
        // make sure you're only reading as many numbers as
        // you can fit in the array:
        if (i <= sensorCount) {
          // trim off any whitespace from the substring:
          numbers[i] = trim(numbers[i]);
          sensorValues[i] =  float(numbers[i]);
        }
        // Things we don't handle in particular can get output to the text
window
        print(serialString);
      }
    }
  }
}
```

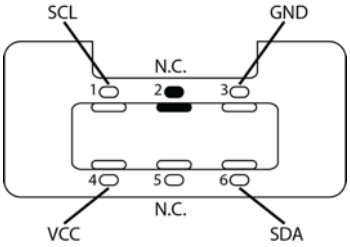
74. Nunchuk without connection board

You don't always have a Nunchuk connection board available, so this chapter describes two alternatives:

- Nunchuk with original connector and jumper cables
- Nunchuk with cut-off connector

74.1. Connections original connector

If you look inside the Nunchuk connector you'll see 2 rows of three little holes each. Jumper cables will fit nicely in these little holes, so making a nice connection with your Arduino board.



Pin nr	Name	Description	Arduino pin
1	c	Clock	SCL (A5)
2	-	-	Not used
3	-	Ground	GND
4	+	3.3 V	3.3 V
5	-	-	Not used
6	d	Data	SDA (A4)

74.2. Connections cut-off connector

Using this method renders you Nunchuk useless for playing Wii games. So use this method only in (semi)permanent Arduino projects.

Cut-off the original connector and remove the outer isolation. Inside you'll find 5 isolated wires, of which you need only the following 4.

Color	Name	Description	Arduino pin
Black	-	Ground	GND
Yellow	+	3.3 V	3.3 V
Red	d	Data	SDA (A4)
White	c	Clock	SCL (A5)
Brown	-	-	Not used

75. Wii u.Draw Game Tablet

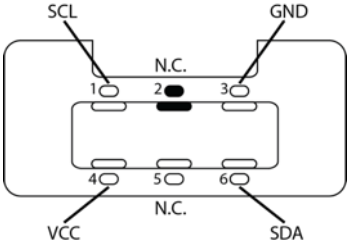
This tablet is created to be used with a Wii Console. It is basically a drawing tablet. It connects to the WiiMote controller through the Nunchuk connector and so it speaks I2C and can be connected to an Arduino.

75.1. Specifications Wii u.Draw Game Tablet

- 2 buttons at the side of the stylus
- If you put a high pressure on the tip of the stylus, this acts as a third button
- The tip of the stylus has a resolution of about 256 pressure levels
- The tablet has a resolution of about 1950x1440 pixels

75.2. Connections Wii u.Draw Game Tablet

If you look inside the connector you'll see 2 rows of three little holes each. Jumper cables will fit nicely in these little holes, so making a nice connection with your Arduino board.



Pin nr	Name	Description	Arduino pin
1	c	Clock	SCL (A5)
2	-	-	Not used
3	-	Ground	GND
4	+	3.3 V	5 V
5	-	-	Not used
6	d	Data	SDA (A4)

75.3. Connections cut-off connector

Using this method renders you u.Draw useless for playing Wii games. So use this method only in (semi)permanent Arduino projects. I've not done this myself, so you are on your own with this.

Cut-off the original connector and remove the outer isolation. I've read that the colors of the wires for this u.Draw Game Tablet are different then from the Nunchuk.

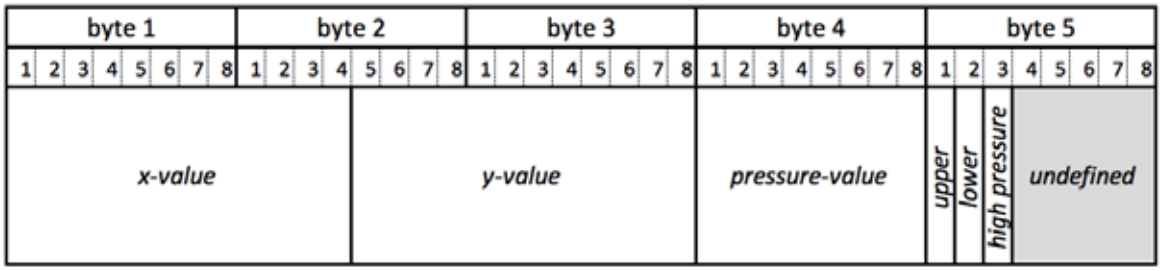
Color	Name	Description	Arduino pin
Blue	-	Ground	GND
Brown	+	3.3 V	5 V
Green	d	Data	SDA (A4)
Yellow	c	Clock	SCL (A5)

75.4. Libraries needed for Wii u.Draw Game Tablet

Although the Nunchuk libraries show data collected from the u.Draw Game Tablet, it is not usable. You can use the Nunchuk libraries to read the 2 (3) buttons, but the pressure values and the coordinates are not very obvious to derive.

Matthew Watts (mwvent on GitHub) has done a lot of research on the u.Draw values. The information below and the sample sketch is derived from his "wii_udraw2arduino2linux" project on GitHub. He has not written a library, but his sample sketch can be used to interpret the values from the u.Draw Game Tablet.

- The x- and y-values are each 12 bits from a total of 3 bytes (12 + 12 = 3 x 8)
  - x-value: derived from all bits from the 1st byte and the first 4 bits of the 2nd byte
  - y-value: derived from the last 4 bits from the 2nd byte and all bits from the 3rd byte.
- The pressure values is 8 bits from the 4th byte
- The two side buttons are each 1 bit from the 5th byte
  - lower side button: 1st bit from the 5th byte
  - upper side button: 2nd bit from the 5th byte
- The third button is 1 bit and is active when the pressure is high
  - 3rd bit from the 5th byte
- The last 5 bits from the 5th byte are undefined



75.5. Sample Wii u.Draw Game Tablet

The following sketch will display the following values
x, y, pressure, highpressure, lower stylus button, upper stylus button.

The sketch is a simplified version of Matthew Watts Arduino sketch described in his "wii_udraw2arduino2linux" project on GitHub.

https://github.com/mwvent/wii_udraw2arduino2linux

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5.
- Connect SDA to A4.

101_uDraw.ini

```
#include <Wire.h>

uint16_t x_raw;
```

```
uint16_t y_raw;
uint8_t pressure_raw;
uint8_t btnStylusUpper;
uint8_t btnStylusLower;
uint8_t btnHighPressure;
uint8_t hasChange = 0;
uint8_t reply[10];
uint8_t lastReply[10];
uint8_t firstRun = 1;

void setup()
{
  Serial.begin(38400);
  Wire.begin();
  Wire.beginTransmission(0x52);
  Wire.write(0xF0);
  Wire.write(0x55);
  Wire.endTransmission();
  delay(100);
  Wire.beginTransmission(0x52);
  Wire.write(0xFB);
  Wire.write(0x00);
  Wire.endTransmission();
  delay(5);
}

void getDataFromTablet()
{
  Wire.beginTransmission(0x52);
  Wire.write(0x00);
  Wire.endTransmission();
  Wire.requestFrom(0x52, 6);
  hasChange = 0;
  unsigned long timeout = millis();
  for (int lp = 0; lp < 6; lp++)
  {
    while ( !Wire.available() )
    {
      if ( millis() - timeout > 2 )
      {
        return;
      }
    };
    reply[lp] = (uint8_t)Wire.read();
    if ( firstRun == 1 )
    {
      lastReply[lp] = reply[lp];
    } else {
      if (lastReply[lp] != reply[lp])
      {
        lastReply[lp] = reply[lp];
        hasChange = 1;
      }
    }
  }
  firstRun = 0;
  uint8_t x_nibble_one = reply[2] & 0x0F;  // 2R
  uint8_t x_nibble_two = reply[0] >> 4;    // 0L
  uint8_t x_nibble_three = reply[0] & 0x0F;// 0R
  uint8_t y_nibble_one = reply[2] >> 4;    // 2L
  uint8_t y_nibble_two = reply[1] >> 4;    // 1L
  uint8_t y_nibble_three = reply[1] & 0x0F;// 1R
```

```
  x_raw = (uint16_t)x_nibble_one << 8 | (uint16_t)x_nibble_two << 4 |
(uint16_t)x_nibble_three;
  y_raw = (uint16_t)y_nibble_one << 8 | (uint16_t)y_nibble_two << 4 |
(uint16_t)y_nibble_three;
  pressure_raw = reply[3];
  btnStylusUpper = !(reply[5] & 1);
  btnStylusLower = !(reply[5] >> 1 & 1);
  btnHighPressure = (reply[5] >> 2 & 1);
}

void sendData()
{
  if ( hasChange == 0 )
  {
    return;
  }
  Serial.print(x_raw);
  Serial.print(" ");
  Serial.print(y_raw);
  Serial.print(" ");
  Serial.print(pressure_raw); //Tip pressure value
  Serial.print(" ");
  Serial.print(btnHighPressure); //Tip pressure high
  Serial.print(" ");
  Serial.print(btnStylusLower); //Lower button
  Serial.print(" ");
  Serial.print(btnStylusUpper); //Upper button
  Serial.println("");
  Serial.flush();
}

void loop()
{
  getDataFromTablet();
  sendData();
}
```

76. PlayStation controller (PSX/PS2)

This controller was developed for Sony's PlayStation gameconsole (PSX/PS2)y. It has a proprietary connector, but you can connect female jumper wires to the pins, so it is very easy to connect it to your Arduino, using only 4 data pins and GND/3.3V.

76.1. Specifications PlayStation controller (PSX/PS2)

- 4 direction keys, Up, Down, Right & Left
- 4 symbol keys, Pink Square ■, Blue Cross ⌛, Red Circle ● & Green Triangle ♦
- 4 shoulder keys R1, L1, R2 & L2
- Select key
- Start key

76.2. Connections PlayStation controller (PSX/PS2)

Pin nr	Name	Description	Arduino pin
1 (left)	Data		Any digital port
2	Command		Any digital port
3	7V Rumble		external power supply
4	GND		Ground
5	Power		3.3V
6	Attention		Any PWM port
7	Clock		Any PWM port
8	nc		-
9 (right)	Acknowledge		not used by me

76.3. Libraries needed for PlayStation controller

- I used the PSX library from Kevin Arendt:
  http://playground.arduino.cc/uploads/Main/Psx1.zip

With this library, every key sets one of 16 different bits. This way it is possible to detect multiple keys. The table below shows the values for the different keys.

Keyname	Decimal value	#define name	Hex value
nothing pressed	0		0x000
Left	1	psxLeft	0x0001
Down	2	psxDown	0x0002
Right	4	psxRight	0x0004
Up	8	psxUp	0x0008
Start	16	psxSrt	0x0010
Select	128	psxSlct	0x0080
■	256	psxSqu	0x0100
⌛	512	psxX	0x0200
●	1024	psxO	0x0400
♦	2048	psxTri	0x0800
R1	4096	psxR1	0x1000
L1	8192	psxL1	0x2000
R2	16384	psxR2	0x4000
L2	32768	psxL2	0x8000

Library use explanation

```
#include <Psx.h>
```

Include Kevin Arendt's PSX library.

```
Psx Psx;
```

Create an instance of the type Psx, called Psx.

```
Psx.setupPins(dataPin, cmndPin, attPin, clockPin, 10);
```

Setup the PSX controller with dataPin, cmndPin, attPin & clockPin being the pins to which the PS2 controller is connected.

```
data = Psx.read();
```

Read the current value from the PS2 controller.

```
if (data & psxR2)
```

Check whether key psxR2 is pressed or not.

76.4. Sample PlayStation 2 controller

The following sketch will show the value of every key pressed. LED 13 will turn as long as you press R2.

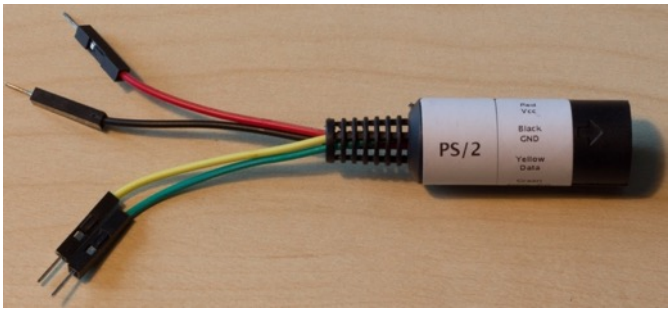
Sample Connections

- Connect pin 1 Data to D8
- Connect pin 2 Command to D9
- Connect pin 4 to Ground
- Connect pin 5 Power to 3.3V
- Connect pin 6 Attention to D11
- Connect pin 7 Clock to D10

041_PSX_Controller.ino

```
#include <Psx.h>
#define dataPin 8
#define cmndPin 9
#define attPin 11
#define clockPin 10

#define LEDPin 13

Psx Psx;

unsigned int data = 0;

void setup()
{
  Psx.setupPins(dataPin, cmndPin, attPin, clockPin, 10);
  pinMode(LEDPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  data = Psx.read();
  Serial.println(data);
  if (data & psxR2)
  {
    digitalWrite(LEDPin, HIGH);
  }
  else
  {
    digitalWrite(LEDPin, LOW);
  }
  delay(20);
}
```

77. PS/2 keyboard



Nowadays all input devices on computers are connected through USB, but before that, keyboards were connected with the PS/2 connector. Through a nice library, it is possible to connect a standard (old) 106-key PS/2 keyboard with the Arduino through only 2 digital ports. This way you have a cheap and port-efficient way to connect 106 switches to your Arduino.

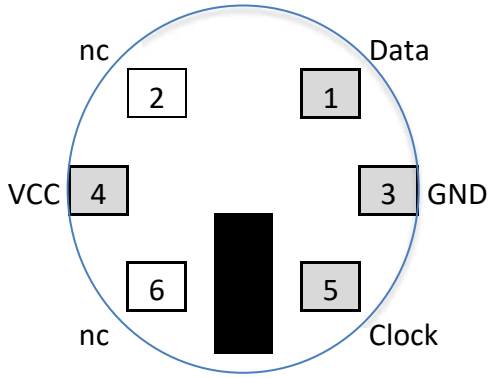
77.1. Specifications PS/2 keyboard

- Mini-DIN-6 male connector
- 6 connectors
- Developed by IBM for their PS/2 computers
- Purple colored for Keyboard
- Green colored connector for Mouse (not described in this chapter)

77.2. Datasheet PS/2 keyboard

- https://en.wikipedia.org/wiki/PS/2_port

77.3. Connections PS/2 keyboard



PS/2 keyboard connector

Pin nr	Name	Description	Arduino pin
1	Data	Data	Any Digital port
2	-	-	not connected
3	GND	Ground	GND
4	VCC	+5V	5V
5	Clock	Clock	IRQ-pin (D2 or D3)
6	-		not connected

77.4. Libraries needed for PS/2 keyboard

- Christian Weichel's (info@32leaves.net) PS2Keyboard library: http://www.pjrc.com/teensy/td_libs_PS2Keyboard.html

Library use explanation

```
#include <PS2Keyboard.h>
```

Include the PS2Keyboard library.

```
PS2Keyboard keyboard;
```

Make a new instance keyboard of the object PS2Keyboard.

```
keyboard.begin(DataPin, IRQpin);
```

Initialize keyboard, where DataPin is the port to which DATA is connected and IRQPin is the Digital Port (only IRQ ports!) to which Clock is connected.

```
keyboard.available()
```

This is true if there is a keystroke in the buffer.

```
keyboard.read();
```

Reads the next character from the keyboard buffer.

```
PS2_ENTER
```

PS2_ENTER is one of the non-printable keys on the keyboard. Check PS2Keyboard.h for more keys.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

77.5. Sample PS/2 keyboard

The following sketch reads the keyboard strokes and prints them to the serial monitor.

Sample Connections

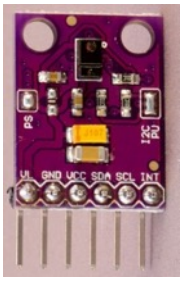
- Connect VCC to 5V.
- Connect GND to GND.
- Connect DATA to D7.
- Connect Clock to D2.

042_PS2Keyboard.ino

```
#include <PS2Keyboard.h>
const int DataPin = 7;
const int IRQpin =  2;

PS2Keyboard keyboard;

void setup() {
  delay(1000);
  keyboard.begin(DataPin, IRQpin);
  Serial.begin(9600);
  Serial.println("Keyboard Test:");
}

void loop() {
  if (keyboard.available()) {
    char c = keyboard.read();
    if (c == PS2_ENTER) {
      Serial.println();
    }
```

```
    else if (c == PS2_TAB) {
      Serial.print("[Tab]");
    }
    else if (c == PS2_ESC) {
      Serial.print("[ESC]");
    }
    else if (c == PS2_PAGEDOWN) {
      Serial.print("[PgDn]");
    }
    else if (c == PS2_PAGEUP) {
      Serial.print("[PgUp]");
    }
    else if (c == PS2_LEFTARROW) {
      Serial.print("[Left]");
    }
    else if (c == PS2_RIGHTARROW) {
      Serial.print("[Right]");
    }
    else if (c == PS2_UPARROW) {
      Serial.print("[Up]");
    } else if (c == PS2_DOWNARROW) {
      Serial.print("[Down]");
    }
    else if (c == PS2_DELETE) {
      Serial.print("[Del]");
    }
    else {
      Serial.print(c);
    }
  }
}
```

78. RGB and Gesture sensor APDS-9960



78.1. Specifications RGB and Gesture sensor APDS-9960

- Ambient light sensing
- RGB color sensing
- Proximity Sensing
- Gesture Detection
- UV & IR blocking filter.
- 2.4 - 3.6 V (Don't use 5V on any of its pins)
- Interrupt driven I2C with data rates up to 400kHz
- Dedicated interrupt pin
- Sensing distance 10-20 cm

78.2. Datasheet RGB and Gesture sensor APDS-9960

- Avago APDS-9960
  https://cdn.sparkfun.com/datasheets/Sensors/Proximity/apds9960.pdf

78.3. Connections RGB and Gesture sensor APDS-9960

Pin nr	Name	Description	Arduino pin
1	UL	Optional power to the IR LED if PS jumper is disconnected (3.0V-4.5V)	
2	GND	Ground	GND
3	VCC	2.4V - 3.6V	3.3V
4	SDA	I2C data	A4 through a level shifter
5	SCL	I2C clock	A5 through a level shifter
6	INT	External interrupt pin. Active LOW on interrupt event	Interrupt pin through a level shifter

78.4. Libraries needed for RGB and Gesture sensor APDS-9960

- Adafruit's APDS9960 library through the Library Manager. But since this specific sensor is not manufactured by Adafruit, you need to change the APDS9960_ID from 0xAB to 0xA8 in the library source file called: Adafruit_APDS9960.cpp.
  So change this:

```
/* Make sure we're actually connected */
  uint8_t x = read8(APDS9960_ID);
  if (x != 0xAB)
  {
    return false;
  }
```

  into this:

```
/* Make sure we're actually connected */
  uint8_t x = read8(APDS9960_ID);
```

```
  if (x != 0xA8)
  {
    return false;
  }
```

After changing the APDS9960_ID, save the file and restart the Arduino IDE.
The APDS-9660 sensor is controlled by I2C, so the Wire library is also needed. (called from within the Adafruit library.

- You could also use Sparkfun's APDS9960 library, also available through the Library Manager. Again, you need to change the APDS9960_ID from 0xAB into 0xA8. You can do this in the library header file called: SparkFun_APDS9960.h. So change this:

```
/* Acceptable device IDs */
#define APDS9960_ID_1           0xAB
#define APDS9960_ID_2           0x9C
```

into this:

```
/* Acceptable device IDs */
#define APDS9960_ID_1           0xA8
#define APDS9960_ID_2           0x9C
```

When using the Sparkfun library, you'll also need the use the Wire library for I2C, but in this case you'll need to include the Wire library from within your sketch.

Library use explanation

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

78.5. Sample RGB and Gesture sensor APDS-9960

I wasn't able to get the gesture sensor working, so instead I only added a simple sketch for reading the color of an object in front of the sensor and display the values for the ambient, the red, the green and the blue light. In this sample the Sparkfun library is used.

Sample Connections

This sensor will get damaged when using 5V VCC and logic, so either use a 3.3V Arduino or use some sort of level shifter. In this sample, I've used "204 Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor".

First connect the APDS-9960 board:

- Connect INT to A3 on the level shifter
- Connect SCL to A2 on the level shifter
- Connect SDA to A1 on the level shifter
- Connect GND to GND
- Connect VCC to 3.3V

Then connect the level shifter to the Arduino:

- Connect both GND pins on the level shifter to GND
- Connect LV on the level shifter to 3.3V
- Connect HV on the level shifter to 5V
- Connect B3 on the level shifter to D2 (Interrupt #0)
- Connect B2 on the level shifter to A5 (SCL)
- Connect B1 on the level shifter to A4 (SDA)

043_APDS-9960_RGBSensor.ino

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint16_t ambient_light = 0;
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;

void setup() {
  Serial.begin(9600);
  apds.init();
  apds.enableLightSensor(false);
  delay(500);
}

void loop() {
  apds.readAmbientLight(ambient_light);
  apds.readRedLight(red_light);
  apds.readGreenLight(green_light);
  apds.readBlueLight(blue_light);
  Serial.print("Ambient: ");
  Serial.print(ambient_light);
  Serial.print(" Red: ");
  Serial.print(red_light);
  Serial.print(" Green: ");
  Serial.print(green_light);
  Serial.print(" Blue: ");
  Serial.println(blue_light);
  delay(1000);
}
```

Sensors

This section describes the use of different kind of sensors, temperature, humidity, distance and more.

79. Temperature Sensor LM35

This sensor measures the temperature in Centigrade.

79.1. Specifications Temperature Sensor LM35

- Linear scale +10 mV/ ^{0}C.
- 0.5 ^{0}C accuracy at +25 ^{0}C.
- Range: -55..+150 ^{0}C

79.2. Datasheet Temperature Sensor LM35

http://www.ti.com/lit/ds/symlink/lm35.pdf

79.3. Connections Temperature Sensor LM35

Pin nr	Name	Description	Arduino pin
1	+Vs	5V	5V
2	Vout	Output	Any analog port
3	GND	Ground	

79.4. Libraries needed for Temperature Sensor LM35

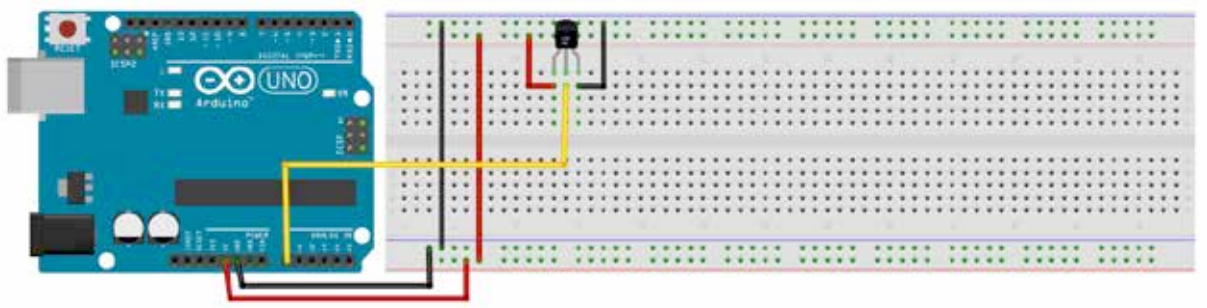
None needed.

79.5. Sample Temperature Sensor LM35

The following sketch measures the temperature and prints the result to the serial port.

Sample Connections

- Connect +Vs to 5V.
- Connect Vout to A0.
- Connect GND to GND.



Sample Sketch

The common equation to calculate the temperature is:

```
tempC=(5.0 * analogRead(tempPin) * 100.0) /1024;
```

A LM35 only produces voltages from 0 to +1V. An analog input has a range from 0 to +5V (1024 steps), so we are wasting 80% of this range and so also accuracy. If you change aRef to 1.1 V you'll increase the accuracy[1].

```
  analogReference(INTERNAL);
```

Now we have 1024 steps in 1.1 V, every 10.0 mV is 1 ^{0}C, so the equation should now be:

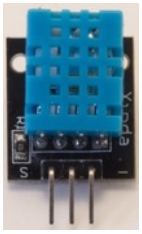
```
 tempC=(analogRead(tempPin)/(10.0/(1100/1024)));
```

044_TemperatureLM35.ino

```
float tempC;
int reading;
int tempPin = 0;

void setup()
{
  analogReference(INTERNAL);
  Serial.begin(9600);
}

void loop()
{
  reading = analogRead(tempPin);
  tempC=reading/(10.0/(1100/1024));
  Serial.println(tempC);
  delay(500);
}
```

[1] http://playground.arduino.cc/Main/LM35HigherResolution

80. Temperature and Humidity sensor board DHT11



This board measures the temperature and humidity.

80.1. Specifications Temperature and Humidity sensor board DHT11

- DHT11.
- 2 ^{0}C accuracy, so no decimals

80.2. Datasheet Temperature and Humidity sensor board DHT11

http://www.micro4you.com/files/sensor/DHT11.pdf

80.3. Connections Temperature and Humidity sensor

Pin nr	Name	Description	Arduino pin
1	S	Signal	Any analog port
2	no name	5V	5V
3	-	Ground	GND

80.4. Libraries needed for Temperature and Humidity sensor board DHT11

- Adafruit Unified Sensor library through the library manager
- DHT sensor library by Adafruit

Library use explanation

```
include <Adafruit_Sensor.h>
```

Load the unified sensor library from Adafruit.

```
#include <DHT_U.h>
```

Load the Unified DHT library from Adafruit.

```
#define DHTPIN A2
```

Define the analog pin used to connect the DHT module.

```
#define DHTTYPE DHT11
```

Define the DHT type (DHT11)

```
DHT_Unified dht(DHTPIN, DHTTYPE);
```

Create an instance of the class DHT_Unified.

```
delay(5000);
```

This delay is needed the let the DHT module stabilize.

```
dht.begin();
```

Initialize the DHT module.

```
sensor_t sensor;
```

Create instance 'sensor' of the 'sensor_t' class

```
dht.temperature().getSensor(&sensor);
```

Read the temperature specs of the DHT module.

```
sensor.name
```
Name of the sensor (DHT11)

```
sensor.version
```
Version of the sensor (1)

```
sensor.sensor_id
```
Both my DHT11 sensors reported -1.

```
sensor.max_value
```
Maximum temperature (50 Celsius)

```
sensor.min_value
```
Minimum temperature (0 Celsius)

```
sensor.resolution
```
Temperature resolution (2 Celsius)

```
dht.humidity().getSensor(&sensor);
```
Read the humidity specs of the DHT module.

```
sensor.max_value
```
Maximum humidity (80%)

```
sensor.min_value
```
Minimum humidity (20%)

```
sensor.resolution
```
Humidity resolution (5%)

```
sensors_event_t event;
```
Create instanc 'event' of the 'sensors_event_t' class

```
dht.temperature().getEvent(&event);
```
Get the current temperature

```
event.temperature
```
Current temperature

```
dht.humidity().getEvent(&event);
```
Get the current humidity

```
event.relative_humidity
```
Current humidity.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

80.5. Sample Temperature and Humidity sensor board DHT11

The following sketch measures the humidity and temperature every 0.8 seconds. No decimals!

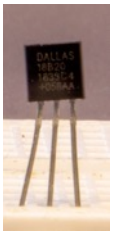
Sample Connections

- Connect S to A2
- Connect the middle pin to 5V
- Connect - to GND

045_Temperature_Humid_DHT11.ino

```
//A full description of this module (and many others) can be downloaded at:
https://eve_arduino

#include <Adafruit_Sensor.h>
#include <DHT_U.h>

#define DHTPIN            A2
#define DHTTYPE           DHT11

DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;

void setup() {
  Serial.begin(9600);
  delay(5000);
  dht.begin();
  Serial.println("DHTxx Unified Sensor Example");
  sensor_t sensor;
  dht.temperature().getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.println("------------------------------------");
  Serial.println("Temperature specs");
  Serial.print  ("Max Value:    "); Serial.println(sensor.max_value);
  Serial.print  ("Min Value:    "); Serial.println(sensor.min_value);
  Serial.print  ("Resolution:   "); Serial.println(sensor.resolution);
  dht.humidity().getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.println("Humidity specs");
  Serial.print  ("Max Value:    "); Serial.println(sensor.max_value);
  Serial.print  ("Min Value:    "); Serial.println(sensor.min_value);
  Serial.print  ("Resolution:   "); Serial.println(sensor.resolution);
  Serial.println("------------------------------------");
  delayMS = sensor.min_delay / 1000;
}

void loop()
{
  delay(delayMS);
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  Serial.print("Temperature: ");
  Serial.println(event.temperature);
  dht.humidity().getEvent(&event);
  Serial.print("Humidity: ");
  Serial.println(event.relative_humidity);
}
```

81. DS18B20 temperature sensor



The DSB18B20 is a 1-Wire Digital Thermometer.

81.1. Specifications DS18B20 temperature sensor

- 1 wire interface
- Multiple sensors on 1 wire through serial number (64 bits)
- Power: 3-5 V
  - external
  - from dataline (parasitair)
- Range: -55 - +125 C +/- 0.5 C
- Zero standby power required
- Programmable resolution 9 - 12 bits
- User-definable, non-volatile temperature alarm setting

81.2. Datasheet DS18B20 temperature sensor

https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

81.3. Connections DS18B20 temperature sensor

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	DQ	Data In/Out	Any Digital Port with a 4.7k Pullup resistor
3	VDD	Power Supply	5V

81.4. Libraries needed for DS18B20 temperature sensor

- DallasTemperature library by Miles Burton, Tim Newsome, Gull Barros & Rob Tillaart through the Library Manager.

Library use explanation

```
#include <OneWire.h>
```

Include Arduino's 1-wire library.

```
#include <DallasTemperature.h>
```

Include the DallasTemperature library.

```
#define ONE_WIRE_BUS 10
```

This is the digital pin onto wich all DS18B20 sensors are connected.

```
#define NR_OF_MODULES 3
```

Use this define to state the numbers of DS18B20 sensors your are using.

```
OneWire oneWire(ONE_WIRE_BUS);
```

Create an instance called oneWire of the Onewire class.

```
DallasTemperature sensors(&oneWire);
```
Create an instance called sensor of the DallasTemperature class.

```
sensors.begin();
```
Initialize all connected sensors.

```
sensors.requestTemperatures();
```
Requests the temperature of all sensors.

```
sensors.getTempCByIndex(0)
```
This holds the temperature of the 1st sensor (index 0).

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

81.5. Sample DS18B20 temperature sensor

The following sketch shows the temperature of all sensors once a second.

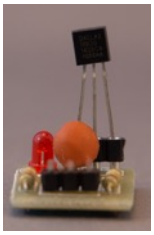
Sample Connections

- Connect 1 GND to GND.
- Connect 2 DQ to D10
- Connect 3 VDD to 5V
- If you want to use multiple DS18B20 temperature sensors, place them all parallel to each other.
- Connect a 4.7Kohm resistor between 2 DQ and 3 VDD

176_DS18B20

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 10
#define NR_OF_MODULES 3

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
}

void loop(void)
{
  for (int count = 0; count < NR_OF_MODULES; count++)
  {
    sensors.requestTemperatures();
    Serial.print(sensors.getTempCByIndex(count));
    Serial.print(" ");
  }
  Serial.println();
  delay(1000);}
```

82. DS18B20 temperature sensor module




This is the same sensor as described in "81 DS18B20 temperature sensor" but sold as a module with a pullup resistor and a power on LED on a PCB.

82.1. Specifications DS18B20 temperature sensor module

- The specifications are listed in the previous chapter "81 DS18B20 temperature sensor"

82.2. Datasheet DS18B20 temperature sensor module

https://cdn.sparkfun.com/datasheets/Sensors/Temp/DS18B20.pdf

82.3. Connections DS18B20 temperature sensor

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	DQ	Data In/Out	Any Digital Port with a 4.7k Pullup resistor
3	VDD	Power Supply	5V

82.4. Libraries needed for DS18B20 temperature sensor module

- The library is already described in the previous chapter "81 DS18B20 temperature sensor"

82.5. Sample DS18B20 temperature sensor module

The following sketch shows the temperature once a second.

Sample Connections

- Connect 1 GND to GND.
- Connect 2 DQ to D10
- Connect 3 VDD to 5V

177_DS18B20_module

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 10

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
}

void loop(void)
{
  sensors.requestTemperatures();
  Serial.println(sensors.getTempCByIndex(0));
  delay(1000);
}
```

83. Barometer BMP180

83.1. Specifications Barometer BMP180

- Supply voltage: 1.8-3.3V power
- Interface I2C
  - I2C ports 5V compliant?
  - address: 0x77[1]
- Temperature measurement
  - -40 - 85 Celsius
- Pressure measurement
  - 300 .. 1100 hPa

83.2. Datasheet Barometer BMP180

https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf

83.3. Connections Barometer BMP180

Pin nr	Name	Description	Arduino pin
1	Vin	3.3V power	3.3V
2	GND	Ground	GND
3	SCL	I2C clock	A5
4	SDA	I2C data	A4

83.4. Libraries needed for Barometer BMP180

- Adafruit's BMP085 library

Library use explanation

```
#include <Wire.h>
```

Include Arduino's I2C library.

```
#include <Adafruit_BMP085.h>
```

Include Adafruit's BMP085/BMP180 library.

```
Adafruit_BMP085 bmp;
```

Create an instance called bmp from the Adafruit_BMP085 type.

```
bmp.begin();
```

Initialize the BMP180 sensor.

```
bmp.readTemperature();
```

Read the temperature.

```
bmp.readPressure();
```

Read the pressure.

[1] You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

83.5. Sample Barometer BMP180

The following sketch

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5.
- Connect SDA to A4.

167_Barometer_BMP180.ino

```
#include <Wire.h>
#include <Adafruit_BMP085.h>

Adafruit_BMP085 bmp;

void setup() {
  Serial.begin(9600);
  bmp.begin();
}

void loop()
{
    Serial.print("Temperature = ");
    Serial.print(bmp.readTemperature());
    Serial.println(" *C");

    Serial.print("Pressure = ");
    Serial.print(bmp.readPressure());
    Serial.println(" Pa");

    Serial.println();
    delay(5000);
}
```

84. Barometer BMP280-5V


This module is based on Bosch BME280 barometer sensor and is equipped with a voltage regulator, so it can be used with 5V MCU's. It is an improved BMP180 sensor and can measure air-pressure and temperature.

84.1. Specifications Barometer BMP280-5V

- Supply voltage: 5V
- Interface I2C
  - address: 0x76[1]
- Temperature measurement
  - -40 - +85 Celsius
- Pressure measurement
  - 300 .. 1100 hPa

84.2. Datasheet Barometer BMP280-5V

http://www.embeddedadventures.com/datasheets/BME280.pdf
Detailed information about the BMP280 sensor can be found at:
https://www.best-microcontroller-projects.com/bmp280.html

84.3. Connections Barometer BMP280-5V

Pin nr	Name	Description	Arduino pin
1	Vin	3.3V power	3.3V
2	GND	Ground	GND
3	SCL	I2C clock	A5
4	SDA	I2C data	A4

84.4. Libraries needed for Barometer BMP280-5V

- Adafruit's BME280 library
- Arduino's I2C library (wire)

Library use explanation

```
#include <Wire.h>
```

Include Arduino's I2C library.

```
#include <Adafruit_Sensor.h>
```

Include Adafruit's multipurpose sensor library.

```
#include <Adafruit_BME280.h>
```

Include Adafruit's BME280 library.

```
#define SEALEVELPRESSURE_HPA (1022)
```

Set the reference sealevel pressure for your location.

[1] You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

```
Adafruit_BME280 bme;
```
Create an instance called bmp from the Adafruit_BME280 type.

```
bme.begin(0x76, &Wire)
```
Initialize the BMP180 sensor, with the address of my specific BME280-5V module.

```
bme.setSampling(Adafruit_BME280::MODE_FORCED,
                Adafruit_BME280::SAMPLING_X1, // temperature
                Adafruit_BME280::SAMPLING_X1, // pressure
                Adafruit_BME280::SAMPLING_X1, // humidity
                Adafruit_BME280::FILTER_OFF   );
```
Set the sampling rates for the weatherstation modus and combine this with an interval of at least 1 minute. This will prevent that the sensor's temperature will interfere it's temperature reading.

```
  bme.takeForcedMeasurement();
```
Read the values for temperature, pressure, altitude and humidity based on the forced sample rates (as defined with the bme.setSampling statement.

```
  bme.readTemperature()
```
Temperature value.

```
  bme.readPressure() / 100.0F
```
Pressure value in Pa and when divided through 100 in hPa.

```
  bme.readAltitude(SEALEVELPRESSURE_HPA)
```
Altitude value based on the local sealevelpressure.

```
  bme.readHumidity()
```
Humidity value.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

84.5. Sample Barometer BMP280-5V

The following sketch reads the temperature, air-pressure, altitude and humidity once every minute.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5.
- Connect SDA to A4.

169_Barometer_BMP280.ino

```
#include <Wire.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1022)
Adafruit_BME280 bme;
unsigned long delayTime;

void setup() {
  Serial.begin(9600);
  Serial.println(F("BME280 test"));

  if (! bme.begin(0x76, &Wire)) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
  Serial.println("-- Weather Station Scenario --");
  Serial.println("forced mode, 1x temperature / 1x humidity / 1x pressure
oversampling,");
  Serial.println("filter off");
  bme.setSampling(Adafruit_BME280::MODE_FORCED,
                  Adafruit_BME280::SAMPLING_X1, // temperature
                  Adafruit_BME280::SAMPLING_X1, // pressure
                  Adafruit_BME280::SAMPLING_X1, // humidity
                  Adafruit_BME280::FILTER_OFF   );
  delayTime = 60000;
  Serial.println();
}

void loop() {
  bme.takeForcedMeasurement();
  printValues();
  delay(delayTime);
}

void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");
  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");
  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");
  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
  Serial.println();
}
```

85. Barometer BME280-5V


This module is based on Bosch BME280 barometer sensor and is equipped with a voltage regulator, so it can be used with 5V MCU's. It can measure air-pressure, temperature and humidity.

85.1. Specifications Barometer BME280-5V

- Supply voltage: 5V
- Interface I2C
  - address: 0x76[1]
- Temperature measurement
  - -40 - +85 Celsius
- Pressure measurement
  - 300 .. 1100 hPa

85.2. Datasheet Barometer BME280-5V

http://www.embeddedadventures.com/datasheets/BME280.pdf

85.3. Connections Barometer BME280-5V

Pin nr	Name	Description	Arduino pin
1	Vin	3.3V power	3.3V
2	GND	Ground	GND
3	SCL	I2C clock	A5
4	SDA	I2C data	A4

85.4. Libraries needed for Barometer BME280-5V

- Adafruit's BME280 library
- Arduino's I2C library (wire)

Library use explanation

```
#include <Wire.h>
```

Include Arduino's I2C library.

```
#include <Adafruit_BME280.h>
```

Include Adafruit's BME280 library.

```
#define SEALEVELPRESSURE_HPA (1022)
```

Set the reference sealevel pressure for your location.

```
Adafruit_BME280 bme;
```

Create an instance called bmp from the Adafruit_BME280 type.

[1] You can download 152_I2C_scanner from my shared sketches-folder: http://bit.ly/eve_arduinosketches.

```
bme.begin(0x76, &Wire)
```

Initialize the BMP180 sensor, with the address of my specific BME280-5V module.

```
bme.setSampling(Adafruit_BME280::MODE_FORCED,
                Adafruit_BME280::SAMPLING_X1, // temperature
                Adafruit_BME280::SAMPLING_X1, // pressure
                Adafruit_BME280::SAMPLING_X1, // humidity
                Adafruit_BME280::FILTER_OFF   );
```

Set the sampling rates for the weatherstation modus and combine this with an interval of at least 1 minute. This will prevent that the sensor's temperature will interfere it's temperature reading.

```
bme.takeForcedMeasurement();
```

Read the values for temperature, pressure, altitude and humidity based on the forced sample rates (as defined with the bme.setSampling statement.

```
bme.readTemperature()
```

Temperature value.

```
bme.readPressure() / 100.0F
```

Pressure value in Pa and when divided through 100 in hPa.

```
bme.readAltitude(SEALEVELPRESSURE_HPA)
```

Altitude value based on the local sealevelpressure.

```
bme.readHumidity()
```

Humidity value.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

85.5. Sample Barometer BME280-5V

The following sketch reads the temperature, air-pressure, altitude and humidity once every minute.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5.
- Connect SDA to A4.

168_Barometer_BME280.ino

```
#include <Wire.h>
#include <Adafruit_BME280.h>
#define SEALEVELPRESSURE_HPA (1022)
Adafruit_BME280 bme;
unsigned long delayTime;

void setup() {
  Serial.begin(9600);
  Serial.println(F("BME280 test"));
  if (! bme.begin(0x76, &Wire)) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }
  Serial.println("-- Weather Station Scenario --");
  Serial.println("forced mode, 1x temperature / 1x humidity / 1x pressure
oversampling,");
  Serial.println("filter off");
  bme.setSampling(Adafruit_BME280::MODE_FORCED,
                  Adafruit_BME280::SAMPLING_X1, // temperature
                  Adafruit_BME280::SAMPLING_X1, // pressure
                  Adafruit_BME280::SAMPLING_X1, // humidity
                  Adafruit_BME280::FILTER_OFF   );
  delayTime = 60000;
  Serial.println();
}

void loop() {
  bme.takeForcedMeasurement();
  printValues();
  delay(delayTime);
}

void printValues() {
  Serial.print("Temperature = ");
  Serial.print(bme.readTemperature());
  Serial.println(" *C");
  Serial.print("Pressure = ");
  Serial.print(bme.readPressure() / 100.0F);
  Serial.println(" hPa");
  Serial.print("Approx. Altitude = ");
  Serial.print(bme.readAltitude(SEALEVELPRESSURE_HPA));
  Serial.println(" m");
  Serial.print("Humidity = ");
  Serial.print(bme.readHumidity());
  Serial.println(" %");
  Serial.println();
}
```

86. Water sensor

This sensor can read the liquid level in a container.

86.1. Specifications Water sensor

- Sensing area (non-linear): 2,2x4cm

86.2. Connections Water sensor

Pin nr	Name	Description	Arduino pin
1	-	Ground	GND
2	+	5V	5V
3	S	Signal	Any analog port

86.3. Libraries needed for Water sensor

None needed.

86.4. Sample Water sensor

The following sketch reads the water level and displays the analog value on the serial monitor. If there is no liquid detected, the analog value is 0. The highest value with this specific water sensor was about 680.

Sample Connections

- Connect GND to GND.
- Connect VCC to 5V.
- Connect S to A0.

046_WaterSensor.ino

```
int analogPin=A0;
int led=13;
int val=0;
int data=0;

void setup()
{
  pinMode(led,OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  val=analogRead(analogPin);
  if (val>600)
  {
    digitalWrite(led,HIGH);
  }
  else
  {
    digitalWrite(led,LOW);
  }
  Serial.println(val);
  delay(100);
}
```

87. Capacitive Soil Moisture sensor v1.2

87.1. Specifications Capacitive Soil Moisture sensor

- Power Supply: 3.3-5.5 V
- Out power: 0-3.0 V
- Power: 5 mA
- Interface: PH2.0-3P
- Dimensions: 97x23mm

87.2. Datasheet Moisture Sensor

https://benselectronics.nl/files/15180/afbeeldingen/bodem-vochtigheisds-sensor.pdf

87.3. Connections Capacitive Soil Moisture sensor

Pin nr	Name	Description	Arduino pin
1	GND	Ground	Ground
2	VCC	Power Supply 3.3-5V	5V
3	AOUT	Analog out	Any analog port

87.4. Libraries needed for Capacitive Soil Humidity sensor

There are no libraries needed for this sensor.

87.5. Sample Capacitive Soil Moisture sensor

The following sketch shows to moisture percentage. The value for the variable Dry = 0% moisture is measured while holding the sensor in the air and the variable Water = 100% moisture is measured in a glass of water.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect Aout to A0.

088_SoilMoisture.ino

```
const int Dry = 622; //Measured in the air
const int Water = 371; //Measured in a glass of water
int Measurement = 0;
int Result = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Measurement = analogRead(A0);
  Result = map(Measurement, Dry, Water, 0, 100);
  Serial.println(String(Result) + " % moisture");
  delay(100);
}
```

88. Touch Sensor TTP223B

88.1. Specifications Touch Sensor TTP223B

- Power Supply: 2-5V
- Output: High: 0.8V, low: 0.3V
- Response time: 60-220ms
- Dimensions: 24x24x7mm
- TTP223B

88.2. Datasheet Touch Sensor TTP223B

- https://raw.githubusercontent.com/SeeedDocument/Grove-Touch_Sensor/master/res/TTP223.pdf

88.3. Connections Touch Sensor TTP223B

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	VCC	Power Supply	5V
3	SIG	Signal	Any digital port

88.4. Libraries needed for Touch Sensor TTP223B

There are no libraries needed for this module.

88.5. Sample Touch Sensor TTP223B

The following sketch turns the onboard LED (D13) on, when the Touch sensor is touched.

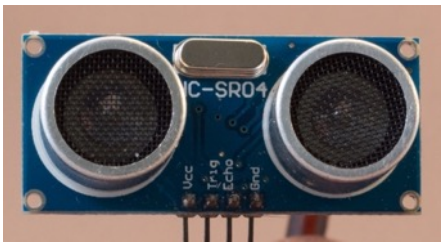
Sample Connections

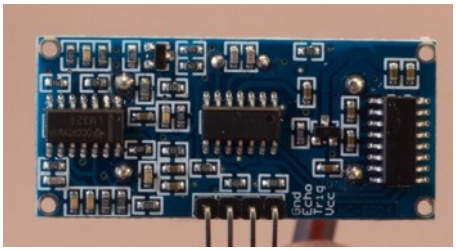
- Connect VCC to 5V.
- Connect GND to GND.
- Connect SIG to D2.

089_TouchSensor.ino

```
const int TouchPin = 2;
const int ledPin = 13;

void setup()
{
  pinMode(TouchPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int sensorValue = digitalRead(TouchPin);
  if (sensorValue == 1)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

89. Distance Sensor HC-SR04





This module sends a short ultrasonic pulse and measures the reflection time, so you can calculate the distance to an object.

89.1. Specifications Distance Sensor HC-SR04

- HC-SR04.
- Distance: 2-500 cm.
- Accuracy : 0,3 cm.
- Angle: 15 degrees.

89.2. Datasheet HC-SR04

- http://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf
- http://www.electroschematics.com/wp-content/uploads/2013/07/HC-SR04-datasheet-version-2.pdf

89.3. Connections for Distance Sensor HC-SR04

Pin nr	Name	Description	Arduino pin
1	VCC	5 V	5 V
2	TRIG	Trigger	Digital output
3	ECHO	Echo	Digital input
4	GND	Ground	GND

89.4. Libraries needed for Distance Sensor HC-SR04

- HCSR04 library from Martin Sosic through the Library Manager.

Library use explanation

```
#include <HCSR04.h>
```

Include Martin Sosic's HCSR04 library.

```
UltraSonicDistanceSensor distanceSensor(T, E);
```

Create distanceSensor a new instance of the object type UltraSonicDistanceSensor. T and E are the digital pins to which the trigger and echo pins are connected.

```
distanceSensor.measureDistanceCm()
```

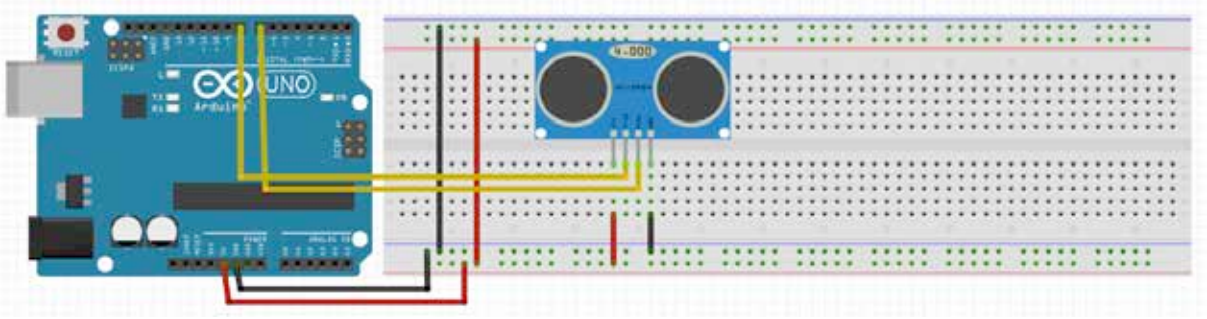
This returns the distance in cm, with 1 decimal.

89.5. Sample Distance Sensor HC-SR04

This sample sketch measures the distance between the sensor board and an object and prints this to the serial port (through USB).

Sample connections

- Connect VCC to 5V.
- Connect TRIG to D6.
- Connect ECHO to D5.
- Connect GND to GND.



047_Distance_HC-SR04.ino

```
#include <HCSR04.h>

UltraSonicDistanceSensor distanceSensor(6, 5);

void setup ()
{
    Serial.begin(9600);
}

void loop ()
{

    Serial.println(int(distanceSensor.measureDistanceCm()));
    delay(500);
}
```

89.6. Processing sketch for the Distance Sensor

A nice Processing script to combine with the above Arduino sketch is listed below.

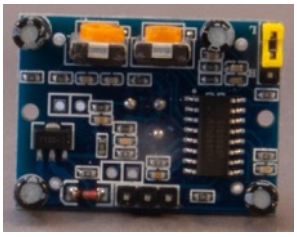
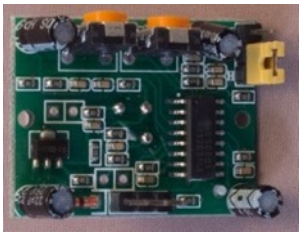
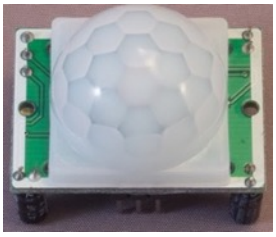
Be careful to select the right serial port and to close the serial monitor in Arduino IDE, before you start the Processing script.

sketch_048_Distance_HC-SR04_Processing.pde

```
/* The following Processing Sketch was created by ScottC on
 the 10 Nov 2012 : http://arduinobasics.blogspot.com/

 Inspired by this Processing sketch by Daniel Shiffman:
 http://processing.org/learning/basics/sinewave.html

 */
import processing.serial.*;

int numOfShapes = 60;
int shapeSpeed = 2;

Square[] mySquares = new Square[numOfShapes];
int shapeSize, distance;
String comPortString;
Serial myPort;

void setup()
{
  size(400, 400);
  smooth();
  shapeSize = (width/numOfShapes);
  for (int i = 0; i<numOfShapes; i++)
  {
    mySquares[i]=new Square(int(shapeSize*i), height-40);
  }
  println(Serial.list());
  String portName = "/dev/cu.usbmodemFD131";
  myPort = new Serial(this, portName, 9600);
  //  or use: myPort = new Serial(this, Serial.list()[3], 9600);
myPort.bufferUntil(10);
}

void draw()
{
  background(0);
  delay(50);
  drawSquares();
}


void serialEvent(Serial cPort)
{
  comPortString = cPort.readStringUntil('\n');
  if (comPortString != null)
  {
    comPortString=trim(comPortString);
    println(comPortString);
    distance = int(map(Integer.parseInt(comPortString), 1, 50, 1, height));
    if (distance<0) {
      distance = 0;
    }
  }
}
```

```
void drawSquares()
{
  int oldY, newY, targetY, redVal, blueVal;
  mySquares[0].setY((height-shapeSize)-distance);
  for (int i = numOfShapes-1; i>0; i--)
  {
    targetY=mySquares[i-1].getY();
    oldY=mySquares[i].getY();
    if (abs(oldY-targetY)<2)
    {
      newY=targetY; //This helps to line them up
    } else
    {
      newY=oldY-((oldY-targetY)/shapeSpeed);
    }
    mySquares[i].setY(newY);
    blueVal = int(map(newY, 0, height, 0, 255));
    redVal = 255-blueVal;
    fill(redVal, 0, blueVal);
    rect(mySquares[i].getX(), mySquares[i].getY(), shapeSize, shapeSize);
  }
}

class Square
{
  int xPosition, yPosition;
  Square(int xPos, int yPos)
  {
    xPosition = xPos;
    yPosition = yPos;
  }
  int getX()
  {
    return xPosition;
  }
  int getY()
  {
    return yPosition;
  }
  void setY(int yPos)
  {
    yPosition = yPos;
  }
}
```

90. PIR/Motion Sensor HC-SR501

90.1. Specifications PIR/Motion Sensor HC-SR501

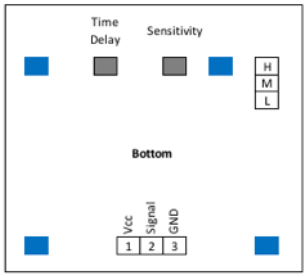
- Delay: 0.3 to 18 seconds
- Range: <120 degree, 7m
- Operating voltage: 5-20V
- Operating current: 65mA
- TTL Output: 0V - 3.3V
- Warm-up time: 1 minute, during this time there will be false positives

90.2. Datasheet PIR/Motion Sensor HC-SR501

- Datasheet HC-SR501 Passive InfraRed Sensor (PIR)
  http://www.datasheetspdf.com/datasheet/download.php?id=775434
- A very nice description of this sensor, can be found at:
  http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/

90.3. Connections PIR/Motion Sensor HC-SR501

Pin nr	Description	Arduino pin
1	VCC	5V
2	Signal	Any analog port
3	Ground	GND



Repeat jumper:

Middle pin with pin H:
Discover motion, even during the delay time after a motion discovery.
Middle pin wit pin L:
Don't discover new motion during the delay time after a motion discovery.

Potentiometers

Left: Trigger delay time 0.3 - 18 seconds
Right: Sensitivity -7m

90.4. Libraries needed for PIR/Motion Sensor HC-SR501

There are no libraries needed.

90.5. ample PIR/Motion Sensor HC-SR501

The following sketch wil detect any movement in its range.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect Signal to D7

049_PIR_HC-SR501.ino

```
int ledPin = 13;
int pirPin = 7;

int pirValue;

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(pirPin, INPUT);
  digitalWrite(ledPin, LOW);
}

void loop()
{
  pirValue = digitalRead(pirPin);
  digitalWrite(ledPin, pirValue);
}
```

91. Photo resistor (LDR)

LDR stands for Light Dependent Resistor. The resistance of a LDR decreases when the light intensity increases. The accuracy of these devices is very low, they are mainly used to detect is it light or dark.

91.1. Specifications Photo resistor (LDR)[1]

- Resistance range: 5M ohm (dark) – 100 ohm (directly under a bright light).

91.2. Connections Photo resistor (LDR)

Both ends of a LDR are equally the same.

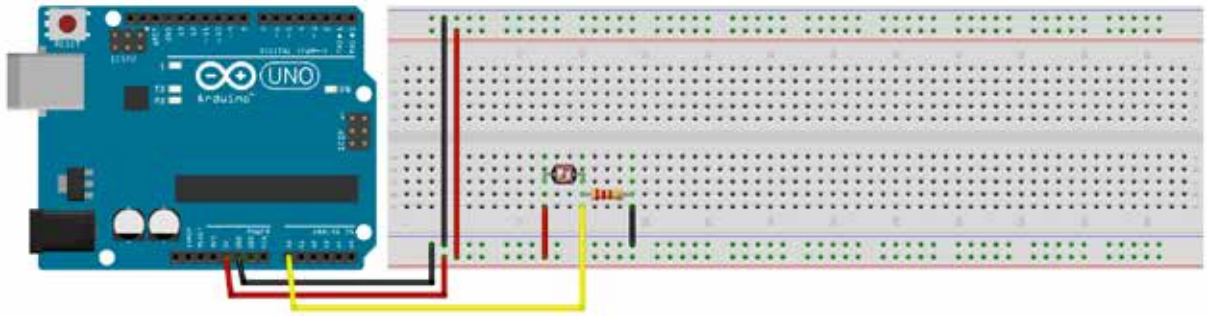
91.3. Libraries needed for Photo resistor (LDR)

None needed.

[1] Background information about LDR's can be found at:
http://learn.adafruit.com/photocells

91.4. Sample Photo resistor (LDR)

The following sketch will show a high value in bright light and a low value in the dark.



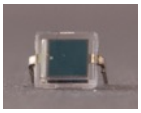
Sample Connections

- Connect one end of the LDR to 5V.
- Connect the other end of the LDR to A0.
- Connect one end of a 10K ohm resistor also to A0.
- Connect the other end of the 10K ohm resister to GND.

050_PIR.ino

```
int sensorPin = A0;
unsigned int sensorValue = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue, DEC);
  delay(500);
}
```

92. Pindiode - Photodiode BPW34



A pin diode or photodiode is a very fast responding light intensity sensor.

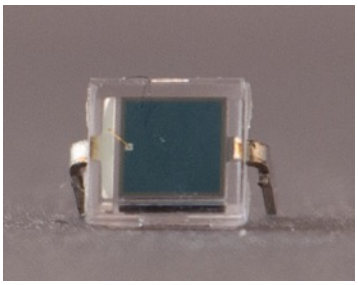
92.1. Specifications Photodiode BPW34

- Spectrums detected: Infrared, visible light
- Wavelength range: 400-1100 nm
- Peak sensitivity: 850 nm
- Peak photo sensitivity: 0.62 A/W
- Angle of half sensitivity: 60 degrees
- Reverse breakdown voltage: 60 V
- Switching speed: 20 ns
- Diode capacitance: 70 pF
- Open circuit voltage: 350 mV
- Rise and fall time: 100 ns each

92.2. Datasheet Photodiode BPW34

http://olivier.granier.free.fr/PC-Montesquieu445072/MOOC-PC-TP/res/bpw34vishay.pdf

92.3. Connections Photodiode BPW34



Pin nr	Name	Description	Arduino pin
1	Anode	Can be recognized by the space between the receiver panel and the pin. (left in the above picture).	To GND through a 10 K resistor & Any analog port
2	Cathode	This pin is directly connected to the receiver panel.	To Vcc

92.4. Libraries needed for Photodiode BPW34

No libraries are needed.

92.5. Sample Photodiode BPW34

The following sketch will detect the flash from a camera.

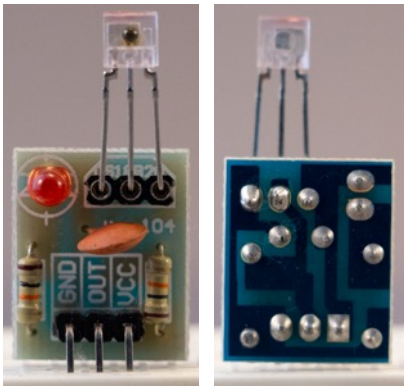
Sample Connections

- Connect 1 (Anode) A0
- Connect 1 (Anode) also to one leg of a 10 K resistor
- Connect the other leg of a 10 K resistor to GND
- Connect 2 (Cathode) to 5V

166_Sample Sketch

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  if (sensorValue > 600)
  {
    Serial.println("Flash");
  }
}
```

93. Laser Sensor module



This laser sensor module is sold as a DS18B20 module, but that is incorrect. The module board is designed to be used with a DS18B20 temperature sensor. but it is also sold with an unknown laser sensor instead of the DS18B20 temperature sensor.

93.1. Specifications Laser Sensor module

- Non-modulator laser receiver (to be used in a dark environment!)

93.2. Datasheet Laser Sensor module

Unknown, since the laser sensor module is unknown.

93.3. Connections Laser Sensor module

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	OUT	Digital output	Any digital port
3	VCC	Power Supply	5 V

93.4. Libraries needed for Laser Sensor module

- No library is needed for this module.

93.5. Sample Laser Sensor module

The following sketch will print the text “Beam detected” when a laser light hits the sensor.

Sample Connections

- Connect GND to GND
- OUT to D6
- Connect VCC to 5V

137_LaserDetector.ino

```
#define DETECTOR 6

void setup() {
  // put your setup code here, to run once:
  pinMode(DETECTOR, INPUT);
  Serial.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  if (digitalRead(DETECTOR)==HIGH)
  {
    Serial.println("Beam Detected");
  }
}
```

94. Flame Sensor (IR photo transistor)

This flame sensor looks like a very dark (black) LED, but actually is a IR photo transistor. It senses IR from a candle light, cigarette lighter or other flames, but also the IR frequencies that are part of some halogen lights.

94.1. Specifications Flame Sensor (IR photo transistor)

94.2. Datasheet Flame Sensor (IR photo transistor)

94.3. Connections Flame Sensor (IR photo transistor)

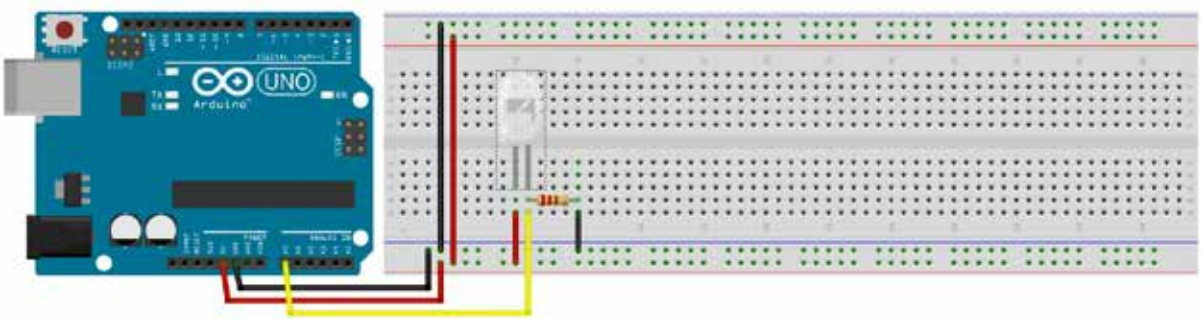
Pin nr	Name	Description	Arduino pin
1	Collector	shortest leg flat edge	5V
2	Emitter	longest leg rounded edge	Any Analog port and to GND through a 22K ohm resistor

94.4. Libraries needed for Flame Sensor (IR photo transistor)

None needed.

94.5. Sample Flame Sensor (IR photo transistor)

The following sketch senses the presence of an IR source. The higher the output level, the stronger (or nearer) the IR source.



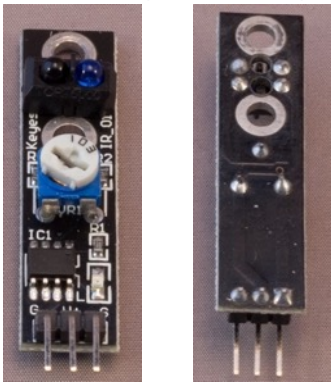
Sample Connections

- Connect the shortest leg to 5V.
- Connect the longest leg to A0 and to one end of a 22K ohm resistor.
- Connect the other end of the 22K ohm resistor to GND.

051_FlameSensor.ino

```
const int analogInPin = A0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int sensorValue = analogRead(analogInPin);
  Serial.println(sensorValue);
  delay(200);
}
```

95. IR proximity sensor board (line finder)



This IR Proximity sensor board can detect an object in its proximity, it can also be used as a line finder.

95.1. Specifications IR proximity sensor board (line finder)

- KY-033 with TCRT5000

The sensitivity can be set through the potentiometer on the sensor board. The output of this board is HIGH when an object is detected and low when there is no object near. If you use a separate TCRT5000 module, without the sensor board, the output is analog and gives an output value between 0-1023. So check whether sample sketches found on the internet are based on the sensor board, or on the separate TCRT5000 module.

Placing three of these sensor boards parallel, you can use them as a Line tracker. When the middle sensor detects a line, the robot should go straight ahead. When the left sensor detects the line, the robot has drifted to much to the right, so the robot should turn to the left. When the right sensor detects the line, the robot has drifted to much to the left, so the robot should turn to the right.

95.2. Datasheets IR proximity sensor board (line finder)

http://pdf1.alldatasheet.com/datasheet-pdf/view/26406/VISHAY/TCRT5000.html

95.3. Connections IR proximity sensor board

Pin nr	Name	Description	Arduino pin
1	G	Ground	GND
2	V+	5 V	5 V

3	S	Signal	Any Digital port

95.4. Libraries needed for IR proximity sensor board (line finder)

None needed.

95.5. Sample IR proximity sensor board (line finder)

With the following sample sketch, the onboard LED (connected to D13) will light up when the sensor detects an object. The distance sensitivity can be changed by the potentiometer on the sensor board.

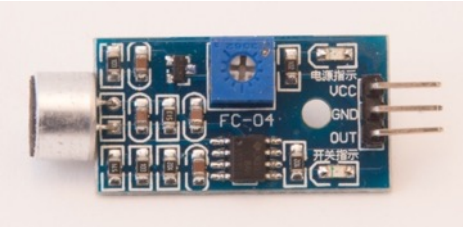
Sample Connections

- Connect G to GND.
- V+ to 5V.
- Connect S to D10.

052_Proximity_TCRT5000.ino

```
int Led=13;
int ProxIR=10;
int val;
void setup()

{
  pinMode(Led,OUTPUT);
  pinMode(ProxIR,INPUT);//
}

void loop()
{
  val=digitalRead(ProxIR);
  if(val==HIGH)
  {
    digitalWrite(Led,LOW);
  }
  else
  {
    digitalWrite(Led,HIGH);
  }
}
```

96. Sound detection FC-04



This sensor board detects the existence of sound (not the intensity).

96.1. Specifications Sound detection

- By default the digital output of this sensor board is HIGH.
- When a sound is detected, the digital output is LOW.
- The threshold on which a sound is detected can be altered with the onboard potentiometer.

96.2. Connections Sound detection

Pin nr	Name	Description	Arduino pin
1	OUT	Data out	Any Digital port
2	GND	Ground	GND
3	VCC	5 V	5V

96.3. Libraries Sound detection

None needed.

96.4. Sample sketch Sound detection

The following sample sketch lights up the onboard LED on D13 for 1 second when the sound level exceeds a specific value.

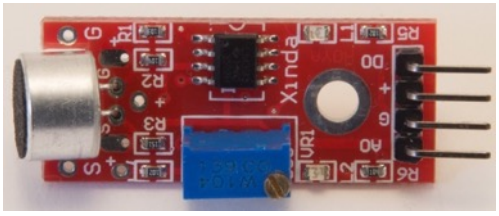
Sample connections

- Connect OUT to D2.
- Connect GND to GND.
- Connect VCC to 5V.

053_SoundDetection_FC-04.ino

```
int Led=13;
int Sound=2;

void setup()

{
  pinMode(Led, OUTPUT);
  pinMode(Sound, INPUT);
}

void loop()
{
  if (digitalRead(Sound) == LOW)
  {
    digitalWrite(Led, HIGH);
    delay(1000);
    digitalWrite(Led, LOW);
  }
}
```

97. Sound detection with digital and analog output



This sensor board detects the existence and the intensity of sound.

97.1. Specifications Sound detection

- By default the digital output of this sensor board is LOW.
- When a sound is detected, the digital output is HIGH.
- The **louder the sound, the lower the value** of the analog output.
- The threshold on which a sound is detected can be altered with the onboard potentiometer.

97.2. Connections Sound detection

Pin nr	Name	Description	Arduino pin
1	AO	Analog Out	Any Analog input port
2	G	Ground	GND
3	+	5 V	5V
4	DO	Digital Out	Any Digital port

97.3. Libraries Sound detection

None needed.

97.4. Sample sketch Sound detection

The following sample sketch lights up the onboard LED on D13 for 1 second when the sound level exceeds a specific value.

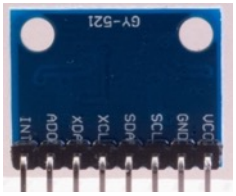
Sample connections

- Connect AO to A0.
- Connect G to GND.
- Connect + to 5V.
- Connect DO to D9.

054_SoundDetectionAnaDig.ino

```
int sensorA=A0;
int sensorD=9;
int sensorvalueD;
int sensorvalueA;

void setup()

{
  pinMode(sensorD, INPUT);
  Serial.begin(9600);
}

void loop()
{
  sensorvalueD=digitalRead(sensorD);
  sensorvalueA=analogRead(sensorA);
  if (sensorvalueD == HIGH)
  {
    Serial.print("Yes I heard you at level: ");
    Serial.println(sensorvalueA);
    delay(1000);
  }
}
```

98. 6DOF MPU-6050 3 Axis Gyro & Accelerometer



This small and chip module can detect movement and its orientation. Used in combination with Processing, you can use it as a game controller.

98.1. Specifications 6DOF MPU-6050 3 Axis Gyro & Accelerometer

- Three-axis gyroscope
- Triaxial accelerometer
- The MPU-6050 sensor contains a MEMS accelerometer and a MEMS gyro in a single chip.
- 16bit AD converter-chip per channel
- I2C communication,
- I2C address programmable through pin AD0 (0x68, 0x69)
- Power supply :3-5v (internal low dropout regulator)
- Communication: IIC communication protocol standard
- Gyro Range: ± 250 500 1000 2000 ° / s
- Acceleration range: ± 2 ± 4 ± 8 ± 16g

98.2. Datasheet 6DOF MPU-6050 3 Axis Gyro & Accelerometer

- MPU-6000 datasheet https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf
- A nice tutorial about this module can be found at: https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/

98.3. Connections 6DOF MPU-6050 3 Axis Gyro & Accelerometer

Pin nr	Name	Description	Arduino pin
1	VCC	3-5V	5V
2	GND	Ground	GND
3	SCL	I2C SCL	A5 (=SCL)
4	SDA	I2C SDA	A4 (=SDA)
5	XDA		Used to connect external magneto sensor
6	XCL		Used to connect external magneto sensor
7	ADO	I2C address selector	pull-down: addr 0x68 pull-up: addr: 0x69
8	INT	Interrupt	D2

98.4. Libraries needed for 6DOF MPU-6050 3 Axis Gyro & Accelerometer

- MPU-6050 library by Jeff Rowberg
  http://diyhacking.com/projects/MPU6050.zip
- I2Cdev library is needed for some MPU-6050 breakout boards, but not in the example below.
  http://diyhacking.com/projects/I2Cdev.zip

Library use explanation

```
#include "MPU6050_6Axis_MotionApps20.h"
```
Include Jeff Rowberg's MPU 6050 library

```
MPU6050 accelgyro;
```
Makes an instance of the MPU6050 class named accelgyro.

```
int16_t ax, ay, az;
```
Variables to hold the values for the x, y and z orientation.

```
int16_t gx, gy, gz;
```
Variables to hold the values for the x, y and z acceleration.

```
accelgyro.initialize();
```
Initialize the MPU-6050.

```
accelgyro.testConnection()
```
Test if the connection to the MPU-6050 was successful.

```
accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
```
Reads the values for orientation and acceleration by sending the ax, ay, az, gx, gy and gz as pointers (call by reference) to the getMotion method.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

98.5. Sample 6DOF MPU-6050 with output to Serial Monitor

The following sketch shows the values for orientation and acceleration.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5
- Connect SDA to A4
- The interrupt pin is not needed in this example.

055_6DOF_MPU6050Serial.ino

```
#include "MPU6050_6Axis_MotionApps20.h"

MPU6050 accelgyro;

int16_t ax, ay, az;
int16_t gx, gy, gz;


void setup() {
    Serial.begin(38400);
    Serial.println("Initializing I2C devices...");
    accelgyro.initialize();
    Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection
successful" : "MPU6050 connection failed");
}

void loop() {
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
        Serial.print(ax); Serial.print("\t");
        Serial.print(ay); Serial.print("\t");
        Serial.print(az); Serial.print("\t");
        Serial.print(gx); Serial.print("\t");
        Serial.print(gy); Serial.print("\t");
        Serial.print(gz);
        Serial.println();
}
```

98.6. Sample 6DOF MPU-6050 to control airplane in Processing

The following sketch uses the MPU-6050 to control an airplane in Processing.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect SCL to A5
- Connect SDA to A4
- Connect INT to D2 (i.e. Interrupt #0)

You must first run the Arduino sketch and then you must start Processing with the library and sketch as is described after the Arduino sketch.

Arduino Sample Sketch control airplane in Processing

Upload the following sketch to your Arduino, but don't start the Serial Monitor (otherwise Processing can't get hold to the serial port of the Arduino.

056_6DOF_MPU6050_2Processing.ino

```
#include "MPU6050_6Axis_MotionApps20.h"

MPU6050 mpu;

bool dmpReady = false;
uint8_t mpuIntStatus;
uint8_t devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
uint8_t teapotPacket[14] = {'$',0x02,0,0,0,0,0,0,0,0,0x00,0x00,'\r','\n'};
volatile bool mpuInterrupt = false;

void dmpDataReady()
{
  mpuInterrupt = true;
}

void setup()
{
  Serial.begin(115200);
  mpu.initialize();
  Serial.println(F("\nWait for Processing to send start key (any key)"));
  while (Serial.available() && Serial.read());
  while (!Serial.available());
  while (Serial.available() && Serial.read());
  devStatus = mpu.dmpInitialize();
  mpu.setXGyroOffset(220);
  mpu.setYGyroOffset(76);
  mpu.setZGyroOffset(-85);
  mpu.setZAccelOffset(1788);
  mpu.setDMPEnabled(true);
  attachInterrupt(0, dmpDataReady, RISING);
  mpuIntStatus = mpu.getIntStatus();
  dmpReady = true;
  packetSize = mpu.dmpGetFIFOPacketSize();
}

void loop()
{
```

```
  while (!mpuInterrupt && fifoCount < packetSize)
  {
  }
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();
  fifoCount = mpu.getFIFOCount();
  if (mpuIntStatus & 0x02)
  {
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    fifoCount -= packetSize;
    teapotPacket[2] = fifoBuffer[0];
    teapotPacket[3] = fifoBuffer[1];
    teapotPacket[4] = fifoBuffer[4];
    teapotPacket[5] = fifoBuffer[5];
    teapotPacket[6] = fifoBuffer[8];
    teapotPacket[7] = fifoBuffer[9];
    teapotPacket[8] = fifoBuffer[12];
    teapotPacket[9] = fifoBuffer[13];
    Serial.write(teapotPacket, 14);
    teapotPacket[11]++;
  }
}
```

Processing sketch

- First you need to install the latest toxiclibs library made by Karsten Schmidt:
- Create a folder called toxiclibs-complete-0020 in the Processing library folder
- Download the toxiclibs library from Karsten Schmidt at: https://bitbucket.org/postspectacular/toxiclibs/downloads/
- Unzip the library into the toxiclibs-complete-0020 folder.
- The Processing sample sketch below is included in the Arduino MPU-6050 library and is called MPUTeapod.pde.
- Find the following line in the MPU Teaport.pde sketch:

```
String portName = /dev/ttyUSB1";
```

  and replace /dev/ttyUSB1 by the name of the serial port to which your Arduino is connected (for Windows, this is something like COMx and for Linux or Mac OSX this is something like /dev/…..)
- Run the Processing script.
- You can now move the MPU-6050 to orientate the Airplane.

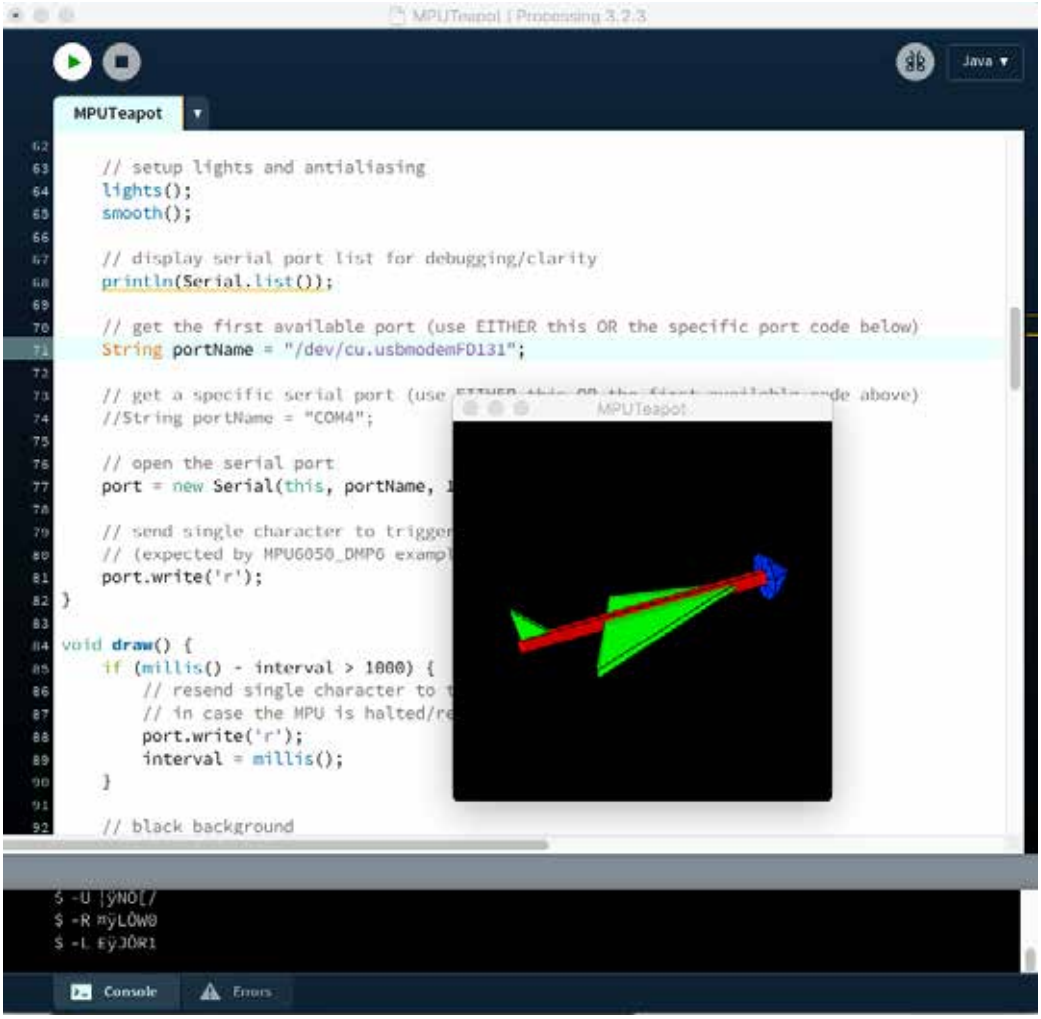
sketch_057_6DOF_MPU6050_2Processing.pde

```
import processing.serial.*;
import processing.opengl.*;
import toxi.geom.*;
import toxi.processing.*;

ToxiclibsSupport gfx;
Serial port;
char[] teapotPacket = new char[14];
int serialCount = 0;
int aligned = 0;
int interval = 0;
float[] q = new float[4];
Quaternion quat = new Quaternion(1, 0, 0, 0);
float[] gravity = new float[3];
float[] euler = new float[3];
```

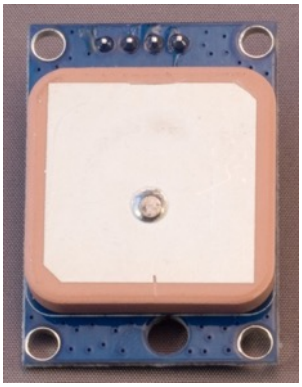
```
float[] ypr = new float[3];

void setup() {
  size(300, 300, OPENGL);
  gfx = new ToxiclibsSupport(this);
  lights();
  smooth();
  println(Serial.list());
  String portName = "/dev/cu.usbmodemFD131";
  port = new Serial(this, portName, 115200);
  port.write('r');
}

void draw() {
  if (millis() - interval > 1000) {
    port.write('r');
    interval = millis();
  }
  background(0);
  pushMatrix();
  translate(width / 2, height / 2);
  float[] axis = quat.toAxisAngle();
  rotate(axis[0], -axis[1], axis[3], axis[2]);
  fill(255, 0, 0, 200);
  box(10, 10, 200);
  fill(0, 0, 255, 200);
  pushMatrix();
  translate(0, 0, -120);
  rotateX(PI/2);
  drawCylinder(0, 20, 20, 8);
  popMatrix();
  fill(0, 255, 0, 200);
  beginShape(TRIANGLES);
    vertex(-100, 2, 30);
    vertex(0, 2, -80);
    vertex(100, 2, 30);
    vertex(-100, -2, 30);
    vertex(0, -2, -80);
    vertex(100, -2, 30);
    vertex(-2, 0, 98);
    vertex(-2, -30, 98);
    vertex(-2, 0, 70);
    vertex( 2, 0, 98);
    vertex( 2, -30, 98);
    vertex( 2, 0, 70);
  endShape();
  beginShape(QUADS);
    vertex(-100, 2, 30);
    vertex(-100, -2, 30);
    vertex(  0, -2, -80);
    vertex(  0, 2, -80);
    vertex( 100, 2, 30);
    vertex( 100, -2, 30);
    vertex(  0, -2, -80);
    vertex(  0, 2, -80);
    vertex(-100, 2, 30);
    vertex(-100, -2, 30);
    vertex(100, -2, 30);
    vertex(100, 2, 30);
    vertex(-2, 0, 98);
    vertex(2, 0, 98);
    vertex(2, -30, 98);
    vertex(-2, -30, 98);
```

```
    vertex(-2, 0, 98);
    vertex(2, 0, 98);
    vertex(2, 0, 70);
    vertex(-2, 0, 70);
    vertex(-2, -30, 98);
    vertex(2, -30, 98);
    vertex(2, 0, 70);
    vertex(-2, 0, 70);
  endShape();

  popMatrix();
}

void serialEvent(Serial port) {
  interval = millis();
  while (port.available() > 0) {
    int ch = port.read();
    print((char)ch);
    if (ch == '$') {
      serialCount = 0;
    }
    if (aligned < 4) {
      if (serialCount == 0) {
        if (ch == '$') aligned++;
        else aligned = 0;
      } else if (serialCount == 1) {
        if (ch == 2) aligned++;
        else aligned = 0;
      } else if (serialCount == 12) {
        if (ch == '\r') aligned++;
        else aligned = 0;
      } else if (serialCount == 13) {
        if (ch == '\n') aligned++;
        else aligned = 0;
      }
      serialCount++;
      if (serialCount == 14) serialCount = 0;
    } else {
      if (serialCount > 0 || ch == '$') {
        teapotPacket[serialCount++] = (char)ch;
        if (serialCount == 14) {
          serialCount = 0;
          q[0] = ((teapotPacket[2] << 8) | teapotPacket[3]) / 16384.0f;
          q[1] = ((teapotPacket[4] << 8) | teapotPacket[5]) / 16384.0f;
          q[2] = ((teapotPacket[6] << 8) | teapotPacket[7]) / 16384.0f;
          q[3] = ((teapotPacket[8] << 8) | teapotPacket[9]) / 16384.0f;
          for (int i = 0; i < 4; i++) if (q[i] >= 2) q[i] = -4 + q[i];
          quat.set(q[0], q[1], q[2], q[3]);
        }
      }
    }
  }
}

void drawCylinder(float topRadius, float bottomRadius, float tall, int
sides) {
  float angle = 0;
  float angleIncrement = TWO_PI / sides;
  beginShape(QUAD_STRIP);
    for (int i = 0; i < sides + 1; ++i) {
      vertex(topRadius*cos(angle), 0, topRadius*sin(angle));
      vertex(bottomRadius*cos(angle), tall, bottomRadius*sin(angle));
      angle += angleIncrement;
```

```
    }
  endShape();
  if (topRadius != 0) {
    angle = 0;
    beginShape(TRIANGLE_FAN);
      vertex(0, 0, 0);
      for (int i = 0; i < sides + 1; i++) {
        vertex(topRadius * cos(angle), 0, topRadius * sin(angle));
        angle += angleIncrement;
      }
    endShape();
  }

  if (bottomRadius != 0) {
    angle = 0;
    beginShape(TRIANGLE_FAN);
      vertex(0, tall, 0);
      for (int i = 0; i < sides + 1; i++) {
        vertex(bottomRadius * cos(angle), tall, bottomRadius * sin(angle));
        angle += angleIncrement;
      }
    endShape();
  }
}
```



99. GPS XM37-1612 (GY-NEO6Mv2?)



99.1. Specifications GPS XM37-1612 (GY-NEO6Mv2?)

- Communication mode: TTL level (3.3/5V system)
- Power supply: 2.7 - 5V
- Current: 45 mA
- Baud rate: 9600
- Catching time:
  - Hot start: 1s
  - Cold start: 27s
- Accuracy: 5m
- 55 channel u-blox
- Rechargeable backup battery can save ephemeris data
- SBAS: WAAS, EGNOS, MSAS, GAGAn
- NavigationDataUpdateRate: 1Hz
- Maximum altitude: 187.000m
- Maximum velocity: 515 m/s

99.2. Datasheet GPS XM37-1612 (GY-NEO6Mv2?)

- E-1612-UB: https://raw.githubusercontent.com/SeeedDocument/Grove-GPS/master/res/E-1612-UB_Datasheets_Sheet.pdf

99.3. Connections GPS XM37-1612 (GY-NEO6Mv2?)

Pin nr	Name	Description	Arduino pin
1	VCC	Power supply	5V
2	RX	Receive data	TX or any Digital pin when using SoftwareSerial
3	TX	Transmit data	RX or any Digital pin when using SoftwareSerial
4	GND	Ground	GND

99.4. Libraries needed for GPS XM37-1612 (GY-NEO6Mv2?)

- TinyGPS++ library from Mikal Hart, immediate inheritor of his TinyGPS library. TinyGPS++ is not so tiny as TinyGPS, but easier to use. https://github.com/mikalhart/TinyGPSPlus/releases

Library use explanation

```
#include <TinyGPS++.h>
```
Include Mikal Hart's TinyGPS++ library.

```
#include <SoftwareSerial.h>
```
Include SoftwareSerial, so you can communicate with the GPS module with self-chosen digital ports, leaving the hardware serial ports free for other purposes.

```
TinyGPSPlus gps;
```
Makes an instance of the TinyGPSPlus class named gps.

```
SoftwareSerial GPSSerial(<TX>, <RX>);
```
Creates a Software Serial port to the GPS module, with <RX> the port to which the GPS's RX pin is connected and <TX> the port to which the GPS's TX pin is connected.

```
GPSSerial.begin(9600);
```
Set the GPS baud rate to 9600 (fixed speed).

```
GPSSerial.available() > 0
```
Test the connection to the GPS.

```
gps.encode(GPSSerial.read())
```
Read a NMEA stream sentence from the GPS module.

```
gps.location.isValid())
```
This is true if the NMEA sentence contains a valid set of coordinates (fix).

```
Serial.print(gps.location.lat(), 4);
```
Prints the latitude with 4 decimals.

```
Serial.print(gps.location.lng(), 4);
```
Prints the longitude with 4 decimals.

Take a look at the following URL from Mikal Hart, the author of TinyGPS++ for more information.

99.5. Sample GPS XM37-1612 (GY-NEO6Mv2?)

The following sketch repeatedly prints the current latitude and longitude.

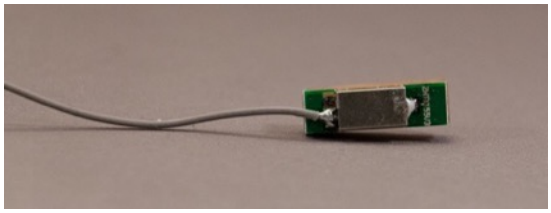
Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect RX to D2.
- Connect TX to D3.

058_GPS_XM37-1612.ino

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

TinyGPSPlus gps;

SoftwareSerial GPSSerial(3, 2);

void setup()
{
  Serial.begin(115200);
  GPSSerial.begin(9600);
}

void loop()
{
  while (GPSSerial.available() > 0)
  {
    if (gps.encode(GPSSerial.read()))
    {
      displayInfo();
    }
  }
}

void displayInfo()
{
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 4);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 4);
  }
  else
  {
    Serial.print(F("No Fix yet, pleas wait"));
  }
  Serial.println();
}
```

100. NEO6 GPS moduel GPS GY-NEO6MV2

100.1. Specifications NEO6 GPS moduel GPS GY-NEO6MV2

- Power Supply Range: 3 V to 5 V
- Model: GY-GPS6MV2
- Ceramic antenna
- EEPROM for saving the configuration data when powered off
- Backup battery
- LED signal indicator
- Default Baud Rate: 9600 bps

100.2. Datasheet NEO6 GPS moduel GPS GY-NEO6MV2

- https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

100.3. Connections NEO6 GPS moduel GPS GY-NEO6MV2

Pin nr	Name	Description	Arduino pin
1	PPS		
2	RxD	Read Data	Tx pin defined in Software serial
3	TxD	Transmit Data	Rx pin defined in Software serial
4	GND	Ground	Ground
5	VCC		5V

100.4. Libraries needed for NEO6 GPS moduel GPS GY-NEO6MV2

For information about the libraries needed, take a look at:

100.5. Sample NEO6 GPS module GPS GY-NEO6MV2

For a sample sketch, take a look at "99.5 Sample GPS XM37-1612 (GY-NEO6Mv2?)".

101. GPS GY-GPS6MV2

101.1. Specifications GPS GY-GPS6MV2

- Power Supply Range: 3.7-3.6 V
- Model: Neo_6M
- Ceramic antenna
- EEPROM for saving the configuration data when powered off
- Backup battery
- LED signal indicator
- Default Baud Rate: 9600 bps

101.2. Datasheet GPS GY-GPS6MV2

- https://www.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_(GPS.G6-HW-09005).pdf

101.3. Connections GPS GY-GPS6MV2

Pin nr	Name	Description	Arduino pin
1	VCC	Power Supply	?
2	Rx	Read Data	Tx pin defined in Software serial
3	Tx	Transmit Data	Rx pin defined in Software serial
4	GND	Ground	Ground

101.4. Libraries needed for GPS GY-GPS6MV2

- TinyGPS++ library from Mikal Hart, immediate inheritor of his TinyGPS library. TinyGPS++ is not so tiny as TinyGPS, but easier to use. https://github.com/mikalhart/TinyGPSPlus/releases

Library use explanation

```
#include <TinyGPS++.h>
```

Include Mikal Hart's TinyGPS++ library.

```
#include <SoftwareSerial.h>
```

Include SoftwareSerial, so you can communicate with the GPS module with self-chosen digital ports, leaving the hardware serial ports free for other purposes.

```
TinyGPSPlus gps;
```

Makes an instance of the TinyGPSPlus class named gps.

```
SoftwareSerial GPSSerial(<TX>, <RX>);
```

Creates a Software Serial port to the GPS module, with <RX> the port to which the GPS's RX pin is connected and <TX> the port to which the GPS's TX pin is connected.

```
GPSSerial.begin(9600);
```

Set the GPS baud rate to 9600 (fixed speed).

```
GPSSerial.available() > 0
```

Test the connection to the GPS.

```
gps.encode(GPSSerial.read())
```
Read a NMEA stream sentence from the GPS module.

```
gps.location.isValid())
```
This is true if the NMEA sentence contains a valid set of coordinates (fix).

```
Serial.print(gps.location.lat(), 4);
```
Prints the latitude with 4 decimals.

```
Serial.print(gps.location.lng(), 4);
```
Prints the longitude with 4 decimals.

Take a look at the following URL from Mikal Hart, the author of TinyGPS++ for more information.

101.5. Sample GPS GY-GPS6MV2

The following sketch repeatedly prints the current latitude and longitude.

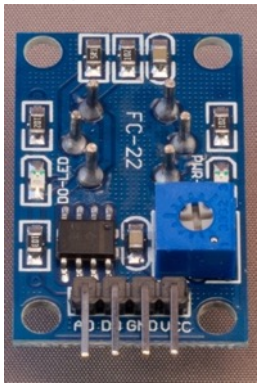
Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect RX to D2.
- Connect TX to D3.

058_GPS_XM37-1612.ino

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>

TinyGPSPlus gps;

SoftwareSerial GPSSerial(3, 2);

void setup()
{
  Serial.begin(115200);
  GPSSerial.begin(9600);
}

void loop()
{
  while (GPSSerial.available() > 0)
  {
    if (gps.encode(GPSSerial.read()))
    {
      displayInfo();
    }
  }
}

void displayInfo()
{
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 4);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 4);
  }
  else
  {
    Serial.print(F("No Fix yet, pleas wait"));
  }
  Serial.println();
}
```

102. MQ-3 alcohol gas sensor board

This sensor senses Alcohol, ethanol and smoke. The sensor board had an analog output to show the concentration or a digital output that will be triggered when a specific threshold is exceeded.

102.1. Specifications MQ-6 alcohol gas sensor board

- High sensitivity for alcohol
- Small sensitivity for petroleum gas, natural gas, city gas and smoke
- Fast response
- Stable and long life
- Power supply voltage: 5V
- Current: 150 ma
- DO output: TTL digital 1 and 0 (0.1 and 5V)
- AO output: 0.1-0.3 V (pollution free) and 4V (maximum concentration)
- Detection range of alcohol: 10-1000 PPM
- Burn in time: over 24 hours.

102.2. Other MQ-x gas sensors

The MQ-3 is part of a whole series of gas sensors. Consult the datasheets for specs like preheat time, heater voltage and load resistor.

- MQ-2: Sensitive for Methane, Butane, LPG, smoke.
  This sensor is sensitive for flammable and combustible gasses.
- MQ-3: Sensitive for Alcohol, Ethanol, smoke
- MQ-4: Sensitive for Methane, CNG Gas
- MQ-5: Sensitive for Natural gas, LPG
- MQ-6: Sensitive for LPG, butane gas
- MQ-7: Sensitive for Carbon Monoxide
- MQ-8: Sensitive for Hydrogen Gas
- MQ-9: Sensitive for Carbon Monoxide, flammable gasses.
- MQ131: Sensitive for Ozone
- MQ135: For Air Quality. Sensitive for Benzene, Alcohol, smoke.
- MQ136: Sensitive for Hydrogen Sulfide gas.
- MQ137: Sensitive for Ammonia.
- MQ138: Sensitive for Benzene, Toluene, Alcohol, Acetone, Propane, Formaldehyd gas, Hydrogen gas.
- MQ214: Sensitive for Methane, Natural gas.
- MQ216: Sensitive for Natural gas, Coal gas.

102.3. Datasheet MQ-3 alcohol gas sensor board

Data sheet MQ-3

- https://www.sparkfun.com/datasheets/Sensors/MQ-3.pdf

A description of working with the MQ-x sensors can be found at:

- http://playground.arduino.cc/Main/MQGasSensors
- https://www.youtube.com/watch?v=BIf_mpnsZvY
- https://www.youtube.com/watch?v=QYSDSKn2Vf8

102.4. Burn in MQ-6 alcohol gas sensor board

The best and most accurate results will be reached after a burn in time of at least 24 hours. In the datasheet this is called Preheat.

- Connect pins VCC and GND and leave this on for at least 24 hours.

This burn in/preheat is only needed once.

102.5. Connections MQ-3 alcohol gas sensor board

Pin nr	Name	Description	Arduino pin
1	VCC	Power supply	5V
2	GND	Ground	GND
3	DO	Digital output	Any digital pin
4	AO	Analog output	Any analog pin

102.6. Libraries needed for MQ-3 alcohol gas sensor board

There is no library needed.

102.7. Sample MQ-3 alcohol gas sensor board

The following simple sketch will show an substantial increase on the values measured on A4 when held above a bottle of spirit.

Sample Connections

- Connect VCC to 5V
- Connect GND to GND
- Connect AO to A4

059_Gas_MQ-3.ino

```
int sensorValue;
int GasSensorPin = 4;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 sensorValue = analogRead(GasSensorPin);
 Serial.println(sensorValue, DEC);
 delay(100);
}
```

A more complex sketch can be found at the following URL:

- http://nootropicdesign.com/projectlab/2010/09/17/arduino-breathalyzer/

103. MQ-6 LPG, iso-butane and propane gas sensor

This sensor senses LPG, iso-butane and propane.

103.1. Specifications MQ-6 LPG, iso-butane and propane gas sensor

- High sensitivity to LPG, iso-butane and propane
- Small sensitivity to alcohol and smoke, you should avoid
- Fast response
- Stable and long life
- Simple drive circuit
- Heater voltage: 5V
- Burn in time: 12-24 hours.

103.2. Other MQ-x gas sensors

The MQ-6 is part of a whole series of gas sensors. Consult the datasheets for specs like preheat time, heater voltage and load resistor.

- MQ-2: Sensitive for Methane, Butane, LPG, smoke.
  This sensor is sensitive for flammable and combustible gasses.
- MQ-3: Sensitive for Alcohol, Ethanol, smoke
- MQ-4: Sensitive for Methane, CNG Gas
- MQ-5: Sensitive for Natural gas, LPG
- MQ-6: Sensitive for LPG, butane gas
- MQ-7: Sensitive for Carbon Monoxide
- MQ-8: Sensitive for Hydrogen Gas
- MQ-9: Sensitive for Carbon Monoxide, flammable gasses.
- MQ131: Sensitive for Ozone
- MQ135: For Air Quality. Sensitive for Benzene, Alcohol, smoke.
- MQ136: Sensitive for Hydrogen Sulfide gas.
- MQ137: Sensitive for Ammonia.
- MQ138: Sensitive for Benzene, Toluene, Alcohol, Acetone, Propane, Formaldehyd gas, Hydrogen gas.
- MQ214: Sensitive for Methane, Natural gas.
- MQ216: Sensitive for Natural gas, Coal gas.

103.3. Datasheet MQ-6 LPG, iso-butane and propane gas sensor

Data sheet MQ-6

- https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-6.pdf

A description of working with the MQ-x sensors can be found at:

- http://playground.arduino.cc/Main/MQGasSensors
- https://www.youtube.com/watch?v=BIf_mpnsZvY
- https://www.youtube.com/watch?v=QYSDSKn2Vf8

103.4. Burn in

The best and most accurate results will be reached after a burn in time of at least 24 hours. In the datasheet this is called Preheat.

- Connect pins A together.
- Connect pins B together.
- Connect 1 H-pin to a 5V power supply (don't use an Arduino, since the heather needs at least 150mA.)
- Leave this on for 24 hours.

This burn in/preheat is only needed once.

103.5. Connections MQ-6 LPG, iso-butane and propane gas sensor

If you look at the bottom of the sensor and turn it so three pins are on the left and the other three pins are on the right. It does not matter which three are left and which three are on the right.

Pin nr	Name	Description	Arduino pin
Left 3 pins	B (upper pin)		Any analog pin
	H (middle pin)	Heater	Ground
	B (lower pin)		Ground through a 10 K resistor
Right 3 pins	A (upper pin)		5V external power supply
	H (middle pin)	Heater	5V external power supply
	A (lower pin		5V external power supply

103.6. Libraries needed for MQ-6 LPG, iso-butane and propane gas sensor

There is no library needed.

103.7. Sample MQ-6 LPG, iso-butane and propane gas sensor

The following simple sketch will show an substantial increase on the values measured on A4.

You can test the sensor with the gas of a simple cigarette lighter. Be careful, you do this on your own risk. In my room A4 gives a reading between 300-400 in clean air and above 900 when the gas is directly under the sensor.

Sample Connections

Left three pins (A - H - A)

- Connect all three pins (A - H - A) to 5V on an external power supply.

Right three pins (B - H - B)

- Connect 1 of the B-pins to A4.
- Connect the H-pin to GND.
- Connect the other B-pin to GND through a 10K resistor.

External 5V power supply

- Connect GND to GND on the Arduino.

060_Gas_MQ-6.ino

```
int sensorValue;
int GasSensorPin = 4;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 sensorValue = analogRead(GasSensorPin);
 Serial.println(sensorValue, DEC);
 delay(100);
}
```

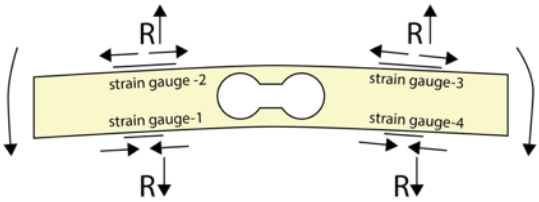
A more complex sketch can be found at the following URL's:

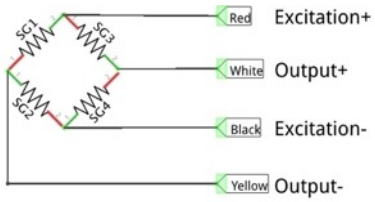
- http://blog.circuits4you.com/2015/05/mq-6-gas-sensor-interfacing-with.html
- http://www.savvymicrocontrollersolutions.com/arduino.php?topic=arduino-mq6-lpg-gas-sensor

With those sketches the sensor will first be calibrated and then the ppm values of the detected gasses will be shown.

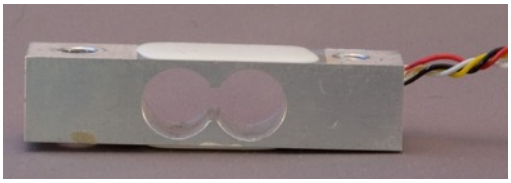
104. Load cells

A load cell consists of 4 strain gauges glued on top and on the bottom of a load cell. When you put a weight on the load cells, the 2 strain gauges on top of the load cell, will be stretched out and the 2 strain gauges on the bottom will be pushed together. Stretching a strain gauge will raise its resistor value a tiny bit and compressing it, will do the opposite. Since the change is very tiny, these strain gauges are organized in a so called Wheatstone bridge raising the accuracy. Excitation+ is attached to VCC and Excitation to ground. Then by measuring the voltage drop between Output+ and Output-, you have a measurement of the amount of weight on top of the load cell. By using a Wheatstone bridge setup, your measurement will be very accurate, but the voltage drop is still very tiny. To use this voltage drop as an input to your Arduino, you need some kind of amplifier. A very well-known amplifier is the HX711, that is described in one of the next chapters.





104.1. Unknown load cell from a kitchen scale



This load cell was salvaged from a kitchen scale.

Specifications

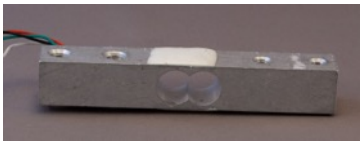
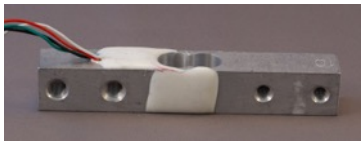
The capacity of the kitchen scale, from which this load cell was salvaged, could have been 2, 3 or 5 kg with a graduation of 1-2 grams, so nothing is sure about the specs.

104.2. 3 kg load cell

Specifications

- Load: 3 kg
- Output: 1.0 +/- 0.15 mV
- Temperature range: 20-60 Celsius
- Supply voltage: 3-12 V
- IP65

104.3. 10 kg load cell

Specifications

- Load: 10 kg
- Output: 1.0 +/- 0.15 mV
- Temperature range: 20-60 Celsius
- Supply voltage: 3-12 V
- IP65

104.4. Connections Load cells

Pin nr	Name	Arduino
Red	Excitation+	5V
Black	Excitation-	GND
Green/Yellow	Output-	to any digital port through an amplifier (like the HX711)
White	Output+	to any digital port through an amplifier (like the HX711)

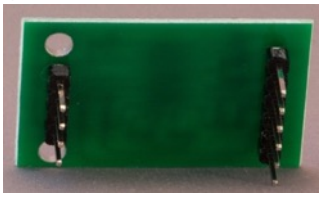
104.5. Libraries needed for Load cells

There are no libraries needed for the load cell, but you'll need a library for the HX711 as described in one of the next chapters.

104.6. Sample Load cells

Without an amplifier like the HX711, you cannot use an Arduino to measure a load.

105. Load cell amplifier HX711



With this load cell amplifier it is possible to amplify the tiny voltage drops from a load cell when loaded with a weight. It uses a two-wire interface for communication with your Arduino.

105.1. Specifications Load cell amplifier HX711

- 24 bit Analog to Digital Converter (ADC)
- Operation supply voltage 2.6-5.5V
- Refresh frequency: 10/80 Hz
- Differential input voltage: +/- 40mV
- Two input channels

105.2. Datasheet Load cell amplifier HX711

- HX711: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

105.3. Connections Load cell amplifier HX711

Connections between Load cell amplifier HX711 and the Arduino.

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	DT	Data Out	Any I/O
3	SCK	Clock	Any I/O
4	VCC	5V	5V

Connections between the Load cell amplifier HX711 and the load cell.

Pin nr	Name	Description	Load cell
1	E+	Excitation plus	Red
2	E-	Excitation min	Black
3	A-	Load cell A Output min	Yellow/Green
4	A+	Load cell A Output plus	White
5	B-	Load cell B Output min	?
6	B+	Load cell B Output min	?

105.4. Libraries needed for Load cell amplifier HX711

- HX711_ADC by Olav Kallhovd through the Library Manager.

Library use explanation

```
#include <HX711_ADC.h>
```

Load the HX711_ADC library

```
HX711_ADC LoadCell(DOUT, CLK);
```
Load the HX711_ADC library

```
float calibration_value=460.0;
```
Set the calibration value for a known weight.

```
LoadCell.begin();
```
Initialize the load cell.

```
LoadCell.start(stabilisingtime);
```
Start the load cell but wait for it to stabilize.

```
LoadCell.setCalFactor(calibration_value);
```
Set the calibration value and Tare the scale.

```
LoadCell.update();
```
??

```
int measurement = LoadCell.getData();
```
Read the weight from the load cell.

```
LoadCell.tareNoDelay();
```
Start the Tare action.

```
LoadCell.getTareStatus()
```
Check whether the Tare action has performed.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

105.5. Sample Load cell amplifier HX711

The following sketch measures a weight in grams. You can use Serial Monitor to perform a Tare by inputting the letter 't'. Before you can get accurate measurements, you'll first need to determine the calibration value.

- Put a known weight on the load cell and check the measurement.
- When the measurement is too high, raise the calibration value.
- When the measurement is too low, lower the calibration value.
- Repeat these steps until you measurement corresponds with the known weight.

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect DO to D3
- Connect SCK to D4
- Connect E+ to the Red wire of the load cell
- Connect E- to the Black wire of the load cell
- Connect A- to the Yellow/Green wire of the load cell
- Connect A+ to the White wire of the load cell

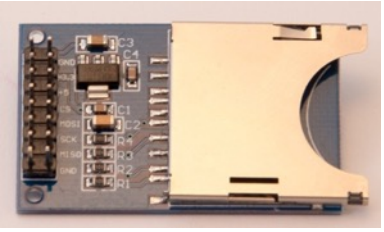
061_LoadCellAmplifier_HX711.ino

```
#include <HX711_ADC.h>
HX711_ADC LoadCell(3, 4);
long time_last_measurement;
float calibration_value=460.0;

void setup()
{
  Serial.begin(9600);
  LoadCell.begin();
  long stabilisingtime = 2000;
  LoadCell.start(stabilisingtime);
  LoadCell.setCalFactor(calibration_value);
  Serial.println("Tare is complete");
}

void loop()
{
  LoadCell.update();
  if (millis() > time_last_measurement + 250)
  {
    int measurement = LoadCell.getData();
    Serial.print("Weight: ");
    Serial.println(measurement);
    time_last_measurement = millis();
  }
  if (Serial.available() > 0)
  {
    char inByte = Serial.read();
    if (inByte == 't') LoadCell.tareNoDelay();
  }
  if (LoadCell.getTareStatus() == true)
  {
    Serial.println("Tare complete");
  }
}
```

Storage

This section describe the use of storage devices on your Arduino board, like an SD card, but also the use of RFID cards.

106. SD (Secure Digital)



106.1. Specifications SD (Secure Digital)

LC-TECH SD

106.2. Datasheet SD (Secure Digital)

Unkown.

106.3. Connections SD (Secure Digital)

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	+3.3	3.3 V	3.3 V
3	+5	5 V (dangerous for SD card??)	n.c.
4	CS	CS	free to choose
5	MOSI	SPI Master Out Slave In	D11
6	SCK	SPI Serial Clock	D13
7	MISO	SPI Master In Slave Out	D12
8	GND	Ground (connect either 1 or 8)	GND

106.4. Libraries needed for SD (Secure Digital)

- Secure Digital card library through Library Manager.
- SPI (Serial Peripheral Interface) library through the Library Manager.

Library use explanation

```
#include <SD.h>
```

Include the Secure Digital card library included with Arduino IDE.

```
Sd2Card mycard;
```

Create mycard a new instance of the object type Sd2Card.

```
SdVolume myvolume;
```

Create myvolume, a new instance of the object type SdVolume.

```
mycard.init(SPI_HALF_SPEED, chipSelect)
```

Initialize connection to mycard, the Boolean result is true if a card is present and if the wiring is correct.

```
mycard.type()
```

Possible values: SD1, SD2 or SDHC.

```
myvolume.init(card)
```

Initialize the connection to myvolume, the Boolean result is true if a readable partition (FAT16/FAT32) is available. (EXT1, EXT2, EXT3, NTFS and exFAT are not supported).

```
volumesize = myvolume.blocksPerCluster();
```

Blocks per cluster.

```
volumesize *= myvolume.clusterCount();
```
Total number of blocks.

```
volumesize *= 512;
```
Total volumsize, since on a SD card, the block size is always 512 bytes.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

106.5. Sample SD (Secure Digital)

The following sketch shows the SD card type, the volume type and the volume size.

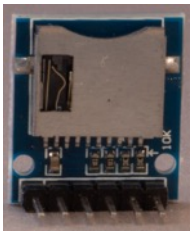
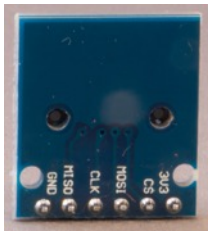
Sample Connections

- Connect GND to GND.
- Connect 3.3V to 3.3V.
- Don't connect 5V!
- Connect CS to D4.
- Connect MOSI to D11.
- Connect SCK to D13.
- Connect MISO to D12.
- Don't connect second GND.

062_SDCardReader.ino

```
#include <SD.h>
#include <SPI.h>
Sd2Card card;
SdVolume volume;

const int chipSelect = 4;

void setup()
{
  Serial.begin(9600);
  Serial.print("\nInitializing SD card...");
  if (!card.init(SPI_HALF_SPEED, chipSelect))
  {
    Serial.println("initialization failed");
    return;
  }
  else
  {
   Serial.println("Wiring is correct and a card is present.");
  }

  Serial.print("\nCard type: ");
  switch(card.type())
  {
    case SD_CARD_TYPE_SD1:
      Serial.println("SD1");
      break;
    case SD_CARD_TYPE_SD2:
      Serial.println("SD2");
      break;
    case SD_CARD_TYPE_SDHC:
      Serial.println("SDHC");
      break;
    default:
      Serial.println("Unknown");
  }

  if (!volume.init(card)) {
    Serial.println("Could not find FAT16/FAT32 partition.");
    return;
  }

  uint32_t volumesize;
  Serial.print("\nVolume type is FAT");
  Serial.println(volume.fatType(), DEC);
  Serial.println();

  volumesize = volume.blocksPerCluster();
  volumesize *= volume.clusterCount();
  volumesize *= 512;
  Serial.print("Volume size (bytes): ");
  Serial.println(volumesize);
}


void loop(void)
{
}
```

107. Micro-SD (Secure Digital)

107.1. Specifications Micro-SD (Secure Digital)

- SPI
- 3.3 V
- Micro-SD sllot

107.2. Datasheet Micro-SD (Secure Digital)

Unknown.

107.3. Connections Micro-SD (Secure Digital)

Pin nr	Name	Description	Arduino pin
1	+3.3	3.3 V	3.3 V
2	CS	CS	free to choose
3	MOSI	SPI Master Out Slave In	D11
4	CLK	SPI Serial Clock	D13
5	MISO	SPI Master In Slave Out	D12
6	GND	Ground	GND

107.4. Libraries needed for Micro-SD (Secure Digital)

Take a look at the library description of the previous chapter.

107.5. Sample Micro-SD (Secure Digital)

Take a look at the sample sketch of the previous chapter.

108. Mifare RFID RC522

108.1. Specifications Mifare RFID RC522

- Contactless RFID reader/writer.
- Supports all variant of the MIFARE mini, 1K, 4K, Ultralight, DESFire EV1 and MIFARE Plus products.
- SPI up to 10 Mbits/s.
- Up to 50 mm.

108.2. Datasheet Mifare RFID RC522

http://www.nxp.com/documents/data_sheet/MFRC522.pdf

108.3. Connections Mifare RFID RC522

Pin nr	Name	Description	Arduino pin
1	SDA	SPI Slave Select	Any Digital Port
2	SCK	SPI Serial Clock	SPI SCK (D13)
3	MOSI	SPI Master Out Slave In	SPI MOSI (D11)
4	MISO	SPI Master In Slave Out	SPI MISA (D12)
5	IRQ	Interrupt	IRQ (D2 or D3 on UNO)
6	GND	Ground	GND
7	RST	Reset	Any Digital Port
8	3.3V	3.3 V	3.3V

108.4. Libraries needed for Mifare RFID RC522

- SPI (Serial Peripheral Interface) library through the Library Manager.
- Mifare RC522 library from Miguel Balboa.
  https://github.com/miguelbalboa/rfid

108.4.1. Library use explanation

```
#include <SPI.h>
```
Include the Serial Peripheral Interface included in the Arduino IDE.

```
#include <MFRC522.h>
```
Include the Mifare RC522 library from Miquel Balboa.

```
MFRC522 mymfrc522(SS_PIN, RST_PIN);
```
Create 'mymfrc522' a new instance of the object MFRC522.
SS_PIN is an integer value corresponding to the digital port SDA is connected to.
RST_PIN is an integer value corresponding to the digital port RST is connected to.

```
SPI.begin();
```
Initialize SPI communication.

```
mymfrc522.PCD_Init();
```
Initialize Mifare RC522 reader.

```
mymfrc522.PICC_IsNewCardPresent())
```
Boolean that shows whether a new card is present.

```
mymfrc522.PICC_ReadCardSerial())
```
Boolean that shows whether the new card has a card ID.

```
Serial.print(mfrc522.uid.uidByte[0], HEX);
```
Print first byte of card ID.

```
byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
```
piccType of the selected card.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

108.5. Sample Mifare RFID RC522

The following sketch shows the ID of every compatible RFID card touching the RFID reader.

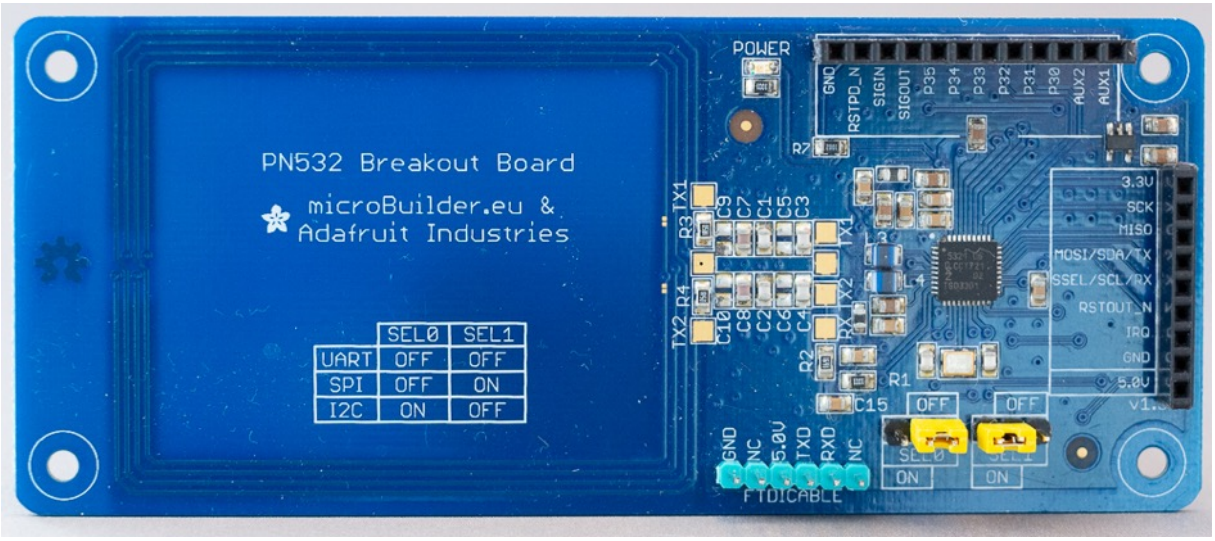
108.5.1. Connections

- Connect SDA to D10.
- Connect SCK to D13.
- Connect MOSI to D11.
- Connect MISO to D12.
- Do not connect IRQ
- Connect GND to GND.
- Connect RST to D9.
- Connect 3.3V to 3.3V.

063_RFID_Mifare_RC522.ino

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

void setup()
{
     Serial.begin(9600);
     SPI.begin();
     mfrc522.PCD_Init();
     Serial.println("Scan a RFID card");
}

void loop()
{
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }

  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }

  Serial.print("Card UID:");
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();

  byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
  Serial.print("PICC type: ");
  Serial.println(mfrc522.PICC_GetTypeName(piccType));
  delay(1000);
}
```

109. Adafruit PN532 NFC/RFID controller breakout board



This PN532 NFC/RFID controller breakout board can read all ISO14443A compatible cards through TTL serial, SPI or I2C.

109.1. Specifications Adafruit PN532 NFC/RFID controller breakout board

- Communication through
  - TTL serial
  - SPI
  - I2C
- Breakout board version 1.3
- Sold together with a 4050 HEX-inverter chip, this can be used as a level shifter
- Power: 3.3v (you need to use a level shifter in combination with a 5V Arduino)
- 13.56MHz RFID
- ISO14443A compatible cards (Mifare: Classic, Ultralight, Plus, DESFire etc..)
- NDEF (NFC Data Exchange Format)
- Passive Communication
- Active Communication (Peer-to-Peer)

109.2. Datasheet Adafruit PN532 NFC/RFID controller breakout board

- https://www.nxp.com/docs/en/nxp/data-sheets/PN532_C1.pdf

109.3. Connections Adafruit PN532 NFC/RFID controller breakout board

To set the communication mode to either UART, I2C or SPI, you must put jumpers on the correct positions on SEL0 and SEL1.

SEL0/SEL1 jumper

	SEL0	SEL1
UART	OFF	OFF
I2C	OFF	**ON**
SPI	**ON**	OFF
n.d.	ON	ON

FTDICABLE

Pin nr	Name	Description	FTDI pin

1	GND	Ground	GND
2	n.c.	not connected	CTS
3	5.0V	Power supply	VCC
4	TXD	Transmit Data	TX
5	RXD	Receive Data	RX
6	n.c.	not connected	DTS

jp3

Pin nr	Name	Description	Arduino pin
1	GND	Ground	-
2	RSTPD_N		-
3	SIGIN		-
4	SIGOUT		-
5..10	P35..P30		-
11	AUX2		-
12	AUX1		-

jp4

Pin nr	Name	Description	Arduino pin
1	3.3V	3.3 Volt	3.3V
2	SCK	SPI SCK	In case of SPI mode: • D13[1]
3	MISO	SPI MISO	In case of SPI mode: • D12[1]
4	MOSI/SDA/TX	Dep. on mode • SPI MOSI • I2C SDA • Uart Tx	Dep. on mode[2] • D11 • A4 • Soft. serial Rx?
5	SSEL/SCL/RX	Dep. on mode • SPI SS • I2C SCL • Uart Rx	Dep. on mode[2] • D10 • A5 • Soft. serial Tx?
6	IRQ	Interrupt	D2/D3[1]
7	GND	Ground	GND
8	5.0V	5 V	5V

[1] Use a level shifter like the 4050 that was part of the Adafruit package
[2] Use a level shifter like the 4050, in case of I2C, also use a 1.5K pullup resistor to 3.3V

109.4. Libraries needed for Adafruit PN532 NFC/RFID controller breakout board

- Adafruit's PN532 library through the library manager.

Library use explanation

```
#include <SPI.h>
```
Include Arduino's builtin SPI library.

```
#include <Adafruit_PN532.h>
```
Include Adafruit's PN532 library.

```
Adafruit_PN532 nfc(<SCK>, <MISO>, <MOSI>, <SS>);
```
Creates an instance of the Adafruit_PN532 class named nfc, with SCK, MISO, MOSI, SS being the pins used for software SPI.

```
nfc.begin();
```
Initialize the nfc reader.

```
uint32_t versiondata = nfc.getFirmwareVersion();
```
Gets the firmware information of the PN5 chip. The first 8 bits corresponds to the chip type. The second 16 bits correspond to the integer part of the firmware and the last 8 bits correspond to the decimal part of the firmware version.

```
nfc.setPassiveActivationRetries(0xFF);
```
Sets the number of retries for trying to read the card presented to the reader.

```
nfc.SAMConfig();
```
Configure board to read RFID tags.

```
success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0],
&uidLength);
```
Gets the UID and the length of the UID.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

109.5. Sample Adafruit PN532 NFC/RFID controller breakout board

The following sketch uses SPI to communicate with the PN532 breakout board. The sketch reads the UID and content of a Mifare classic card.

Sample Connections

- Connect 3.3V to Arduino's 3.3V
- Connect SCK to 4050 pin 10
- Connect MISO to D5
- Connect MOSI to 4050 pin 12
- Connect SSEL to 4050 pin 15
- Connect GND to Arduino's GND
- Connect 4050 pin 1 to 3.3V
- Connect 4050 pin 8 to GND
- Connect 4050 pin 9 to D2
- Connect 4050 pin 11 to D3
- Connect 4050 pin 14 to D4
- Place a jumper at SEL0 on the OFF position[1]
- Place a jumper at SEL1 on the ON position

154_PN532_NFC.ino

```
#include <SPI.h>
#include <Adafruit_PN532.h>

Adafruit_PN532 nfc(2, 5, 3, 4);

void setup(void)
{
  Serial.begin(115200);
  Serial.println("Hello!");
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata)
  {
    Serial.print("Didn't find PN53x board");
    while (1);
  }
  nfc.setPassiveActivationRetries(0xFF);
  nfc.SAMConfig();
  Serial.println("Waiting for an ISO14443A card");
}
```

[1] If either SEL0 or SEL1 is in the ON position, the other needs to be set in the OFF position. So when selecting SPI or I2C you'll always need 2 jumpers, but if you want to use the UART serial mode, you can omit both jumpers (or put both in the OFF position)!

```
void loop(void)
{
  boolean success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };
  uint8_t uidLength;
  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0],
&uidLength);
  if (success)
  {
    Serial.print("UID: ");
    for (uint8_t i = 0; i < uidLength; i++)
    {
      Serial.print(" 0x"); Serial.print(uid[i], HEX);
    }
    Serial.println("");
    delay(1000);
  }
  else
  {
    Serial.println("Timed out waiting for a card");
  }
}
```

Real Time Clock

In this section you'll find two small Real Time Clock modules. Such modules keeps the clock ticking even if the Arduino loses power.

110. RTC module with DS1302 chip

This Real Time Clock module is equipped with a CR2032 battery so time will keep ticking after losing power from the Arduino. It is also used in projects where timing must be accurate.

110.1. Specifications RTC module with DS1302 chip

- Real-Time Clock counts Seconds, Minutes, Hours, Day of the Week, Date of Month, Month and Year with Leap-Year compensation up to 2100.
- 31 bytes RAM memory (battery backed)
- 3 Wire Interface.

110.2. Datasheet DS1302 Trickle-Charge Timekeeping Chip

http://datasheets.maximintegrated.com/en/ds/DS1302.pdf

110.3. Connections RTC module with DS1302 chip

Pin nr	Name	Description	Arduino pin
1	VCC	5 V	5V
2	GND	Ground	GND
3	CLK (SLCK)	Serial Clock	Any Digital port
4	DAT (I/O)	Data	Any Digital port
5	RST (CE)	Reset	Any Digital port

110.4. Libraries needed for RTC module with DS1302 chip

- DS1302 RTC library from Matt Sparks
  https://github.com/msparks/arduino-ds1302

Library use explanation

```
#include <DS1302.h>
```
Include Matt Sparks DS1302 RTC library

```
DS1302 rtc(R, D,C);
```
Create rtc an instance of the DS1302 object with R as the RST pin, D as the DAT pin and C as the CLK pin.

```
Time t = rtc.time();
```
Read the current time from the DS1302 chip.

```
t.day
```
Weekday number 1..7 => Sunday .. Saturday

```
t.yr
```
Year in 4 digits

```
t.mon
```
Month in 1-2 digits

```
t.date
```
Day of the month in 1-2 digits

```
t.hr
```
Hour in 1-2 digits

```
t.min
```
Minutes in 1-2 digits

```
t.sec
```
Seconds in 1-2 digits

```
rtc.writeProtect(false);
```
Run this at least once for every new DS1302 chip. It enables you to change the time. You don't need to change it back to protected.

```
rtc.halt(false);
```
Run this at least once and don't change it back to halted.

```
Time t(2018, 11, 23, 22, 43, 10, Time::kFriday);
```
Set the variable t to: Friday 23rd of November 2018,22:43:10

```
rtc.time(t);
```
Set the time of the DS1302 chip to t.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

110.5. Sample RTC module with DS1302 chip

The following sketch shows the time and date every second.

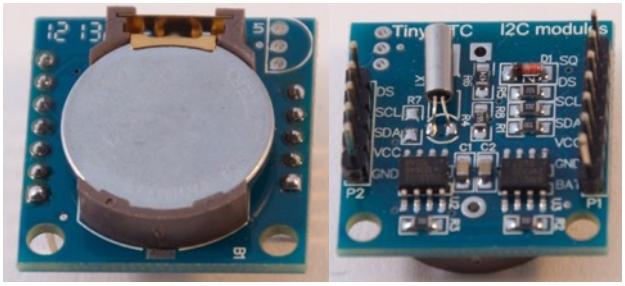
Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect CLK to D7.
- Connect DAT to D8.
- Connect RST to D9.

064_RTC_DS1302.ino

```
#include <DS1302.h>

const int RSTPin = 9;
const int DATPin = 8;
const int CLKPin = 7;

DS1302 rtc(RSTPin, DATPin, CLKPin);

String dayName[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"};

void printTime()
{
  Time t = rtc.time();
  String tString = String(dayName[t.day - 1]) + " " + String(t.yr)
                   + "-" + String(t.mon) + "-" + String(t.date) + " "
                   + String(t.hr) + ":" + String(t.min) + ":" +
String(t.sec);
  Serial.println(tString);
}

void setClockOnce() //Call this routine to change the clock
{
  Time t(2019, 12, 8, 16, 57,  30, Time::kSunday);
  rtc.time(t);
}

void setup()
{
  Serial.begin(9600);
  rtc.writeProtect(false);
  rtc.halt(false);
  setClockOnce();
}

void loop()
{
  printTime();
  delay(1000);
}
```

111. Tiny RTC I²C module with DS1307 chip



This Real Time Clock module is equipped with a CR2032 battery so time will keep ticking after losing power from the Arduino. It is also used in projects where timing must be accurate. It uses the I²C bus.

111.1. Specifications Tiny RTC I²C module with DS1307 chip

- Real-Time Clock counts Seconds, Minutes, Hours, Day of the Week, Date of Month, Month and Year with Leap-Year compensation up to 2100.
- 56 bytes RAM memory (battery backed)
- I²C.

111.2. Datasheet Tiny RTC I²C module with DS1307 chip

- http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

111.3. Connections Tiny RTC I²C module with DS1307 chip

P1: 7 pin header

Pin nr	Name	Description	Arduino pin
P1-1	SQ	?	not needed
P1-2	DS	?	not needed
P1-3	SCL	I²C Clock	SCL (A5)
P1-4	SDA	I²C Data	SDA (A4)
P1-5	VCC	VCC	5V
P1-6	GND	Ground	GND
P1-7	BATT	Battery	not needed

P2: 5 pin header

Pin nr	Name	Description	Arduino pin
P2-1	DS	?	not needed
P2-2	SCL	I²C Clock	not needed
P2-3	SDA	I²C Data	not needed
P2-4	VCC	VDD	not needed
P2-5	GND	Ground	not needed

111.4. Libraries needed for Tiny RTC I²C module with DS1307 chip

- Inter Integrated Circuit (I²C) and Two Wire Interface (TWI) library, included with Arduino IDE: "Wire.h".
- DS1307 Real Time Clock library through Library Manager

Library use explanation

```
#include <Wire.h>
```
Include the Inter-Integrated Circuit (I²C) and Two Wire Interface (TWI) library.

```
#include "RTClib.h"
```
Include the DS1307 Real Time Clock library from Adafruit.

```
RTC_DS1307 myrtc;
```
Create myrtc a new instance of the object RTC_DS1307.

```
Wire.begin();
```
Start the I²C communication through Wire.

```
myrtc.begin();
```
Start the rtc module.

```
myrtc.isrunning()
```
Boolean showing whether myrtc is running.

```
myrtc.adjust(DateTime(__DATE__, __TIME__));
```
Set the time on myrtc to the time at the moment of compiling this sketch.

```
DateTime now = myrtc.now();
```
Time from myrtc is copied to the Arduino clock.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

111.5. Sample Tiny RTC I²C module with DS1307 chip

The following sketch sets the clock to the current date/time (moment of compilation) and shows the current time every 3 seconds.

Sample Connections

- Connect SCL to A5.
- Connect SDA to A4.
- Connect VCC to 5V.
- Connect GND to GND.

065_RTC_DS1307_I2C.ino

```
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 rtc;

void setup () {
  Serial.begin(9600);
  Wire.begin();
  rtc.begin();

  if (! rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
    rtc.adjust(DateTime(__DATE__, __TIME__));
  }
}

void loop () {
    DateTime now = rtc.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();

    Serial.println();
    delay(3000);
}
```

Servo's, Motors & Steppers

Working with motors looks simple, but it isn't. Switching a motor back and forwards and the high currents that even motors from toys can draw needs special equipment. This section describes how to deal with servo's, motors and steppers.

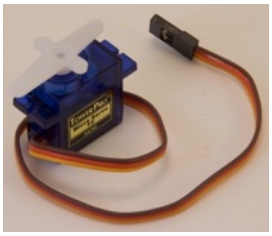
112. Standard Servo

Servo's are DC motors with a potentiometer connected to the motor shaft. This potentiometer tells the servo driver in which position the shaft is. This way you can turn this shaft very accurate, but only for a limited angle. Most common servos can only turn for about 180 degrees. In lots of projects (RC cars, model planes etc.) only 90 degrees is used.

112.1. Specifications Standard Servo

112.2. Connections Standard Servo

Tower Pro



Pin nr	Name	Description	Arduino pin
1	brown	Ground	GND
2	Red	5 V	5 V
3	orange	Signal	Any PWM port

RC Spring

Pin nr	Name	Description	Arduino pin
1	black	Ground	GND
2	Red	5 V	5 V
3	white	Signal	Any PWM port

112.3. Libraries needed for Standard Servo

- Servo library through the Library Manager.

Library use explanation

```
#include <Servo.h>
```

Include the servo library included in Arduino IDE.

```
Servo myservo;
```

Create myservo a new instance of the object type Servo.

```
myservo.attach(6);
```

Connect myservo to D6 (a PWM port).

```
myservo.writeMicroseconds(1500);
```

Set servo at its middle position. Depending on the servo this value can vary between 500..2300. Be careful with the minimum and maximum values. Check every new servo before you implement it in your project. If you've determined the maximum values, you could calculate the middle position.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

112.4. Sample Standard Servo

This servo sweeps between three positions, full left, middle, full right, middle, etc..

After testing 600 was determined as full left and 2150 was determined as full right so 600+(2150-600)/2 = 1375 is the middle position of this particular servo. This is just under 180 degrees.

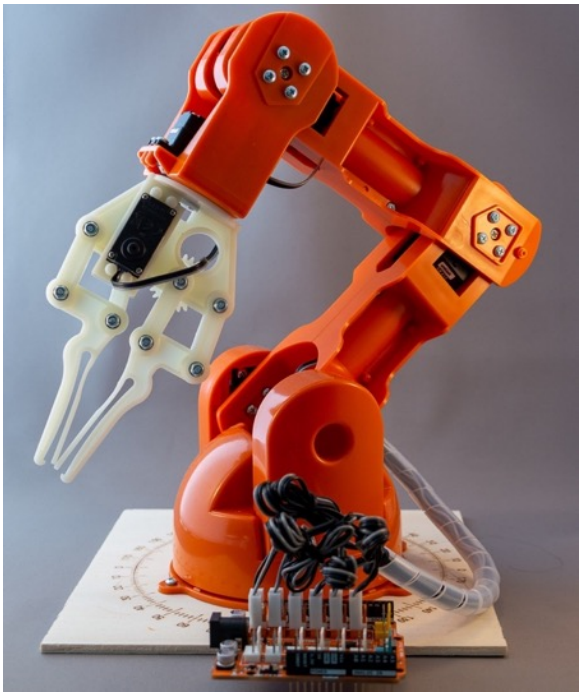
Connections

- Connect 1 to GND.
- Connect 2 to 5V.
- Connect 3 to D6.

066_ServoSweep.ino

```
#include <Servo.h>
Servo myservo;

int pos = 0;

void setup()
{
  myservo.attach(6);
}

void loop()
{
  myservo.writeMicroseconds(600);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
  myservo.writeMicroseconds(2150);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
}
```

113. Tinkerkit Braccio Robot



This robot arm has 6 degrees of freedom (DOF) and can be controlled through an Arduino.

113.1. Contents Tinkerkit Braccio Robot

- Braccio Shield v4
- 2 Servo motors SR 311
- 4 Servo motors SR 431
- Power supply 5V, 4000mA
- Parts and screws to build the robot
- Philips screwdriver
- Wooden bottom plate with markings

113.2. Specifications Tinkerkit Braccio Robot

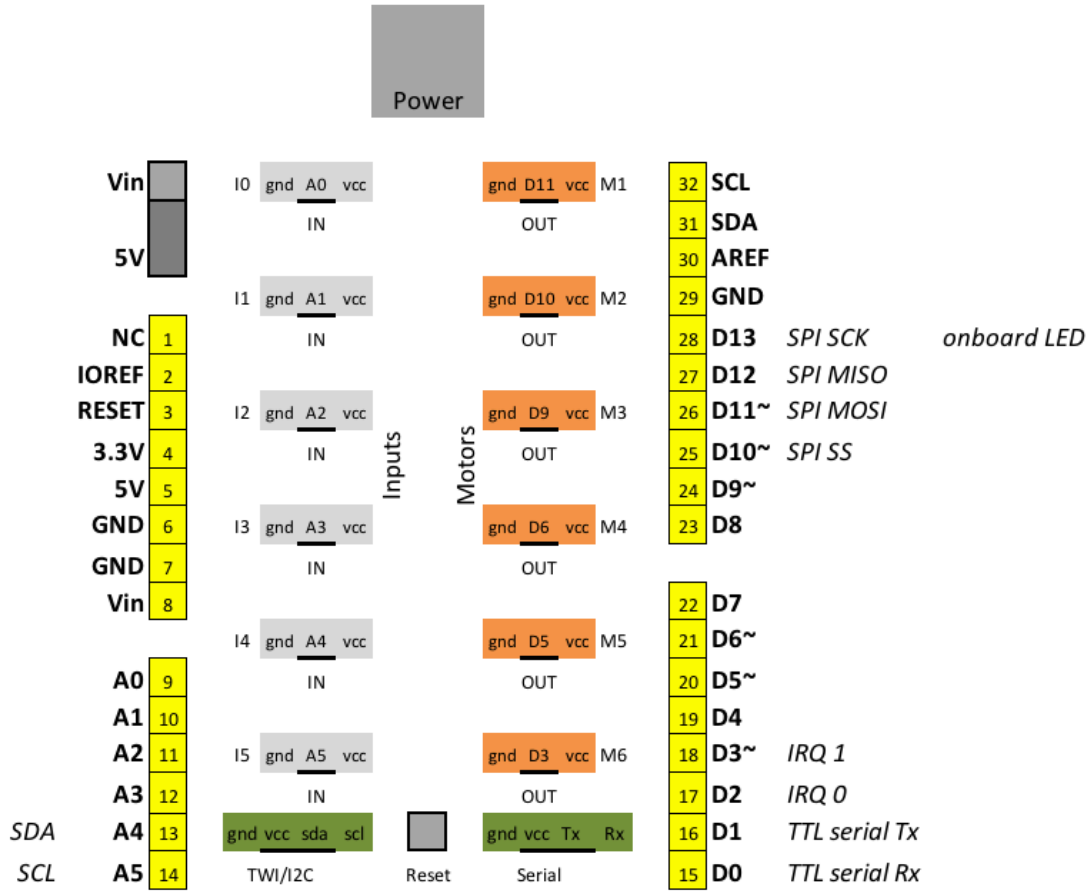
Power	Separate power connector (for the 5V 5000mA power supply that came with the robot)
I2C / TWI	1x 4 pin Tinkerkit connector, connects to Arduino's I2C pins (SDA & SCL) through a buffer chip
UART	1 x 4 pin Tinkerkit connector, connects to Rx & Tx
Analog input I0..I5	6 x 3 pin Tinkerkit connectors, connects to the analog ports • I0 <=> A0 • I1 <=> A1 • I2 <=> A2 • I3 <=> A3 • I4 <=> A4 • I5 <=> A5
Digital output M1..M5, or motor connections	6 x 3 pin Tinkerkit conectors, connects to the PWM capable digital pins, all with protection fuses. • M1 <=> D11 (max 1.1A) • M2 <=> D10 (max 1.1A)

	• M3 <=> D9 (max 1.1A) • M4 <=> D6 (max 1.1A) • M5 <=> D5 (max 0.75A) • M6 <=> D3 (max 0.75A)
Status LED's	• LED ON: Green, shield is on • LED OK: Green, shield is correctly powered • LED ERR: Red: shield is not correctly powered
Voltage level test	During soft-start, D12 is used to monitor the voltage level, this can be disabled to free D12 for other purposes.

113.3. Documentation Tinkerkit Braccio Robot

The documentation can be found at: https://www.arduino.cc/en/Guide/Braccio

113.4. Layout/connections Braccio shield v4



For most Arduino's the power switch on this shield should be in the 5V position, so the Arduino is powered through the 5V pin. This is NOT THE CASE with the Arduino Yun, Tian and Primo, in these cases set the switch in the Vin position.

113.5. Servo's

Nr	Type	Function	Start	End	Braccio.begin() Safety position
M1	SR 431	Base	0	180	90
M2	SR 431	Shoulder	15	165	45
M3	SR 431	Elbow	0	180	180
M4	SR 431	Wrist vertical	0	180	180
M5	SR 311	Wrist rotation	0	180	90
M6	SR 311	Gripper	10 open tongue	73 closed tongue	10 open tongue

Specifications Servo SR 311

- Dimensions: 38.1 x 31.3 x 16.5 mm
- Operating voltage: 4.8 - 6V
- Operating speed:
  - 0.14s/60° (4.8 V)
  - 0.12sec/60° (6.0 V)

- Stall torque:
  - 3.1 kg/cm (4.8 V)
  - 3.8 kg/cm (6.0 V)
- Rotation angle: 180°

Specifications Servo SR 431

- Dimensions: 50.3 x 41.3 x 20.7 mm
- Operating voltage: 4.8 - 6V
- Operating speed:
  - 0.2s/60° (4.8 V)
  - 0.18sec/60° (6.0 V)
- Stall torque:
  - 12.2 kg/cm (4.8 V)
  - 14.5 kg/cm (6.0 V)
- Rotation angle: 180°

113.6. Libraries needed for Tinkerkit Braccio Robot

- You need the Braccio libray from Andrea Martino through the Library Manager.
- You need the Servo library from Arduino through the Library Manager.

Library use explanation

```
#include <Braccio.h>
```

Include the Braccio library for the Braccio Shield.

```
#include <Servo.h>
```

Include the servo library for the servo's.

```
Servo base;
```

Initialize the Base servo: M1.

```
Servo shoulder;
```

Initialize the Shoulder servo: M2.

```
Servo elbow;
```

Initialize the Elbow servo: M3.

```
Servo wrist_ver;
```

Initialize the Vertical Wrist servo: M4.

```
Servo wrist_rot;
```

Initialize the Rotation Wrist servo: M5.

```
Servo gripper;
```

Initialize the Gripper servo: M6.

```
Braccio.begin();
```

Initialization of the servo's. Positions the servo motors to their "safety" position.

Servo nr	***Servo name***	***Initial value***
<M1>	*Base*	*90*

<M2>	*Shoulder*	*45*
<M3>	*Elbow*	*180*
<M4>	*Vertical wrist*	*180*
<M5>	*Rotating wrist*	*90*
<M6>	*Gripper*	*10* *open*

Braccio.ServoMovement(<step_delay>, <M1>, <M2>, <M3>, <M4>, <M5>, <M6>);

Moves all servo's at once, to the indicated postions <M1>, <M2>, <M3>, <M4>, M5>, <M6>, with a delay of <step_delay> between the servo's.

Servo nr	***Servo name***	***Min. value***	***Max. value***
<step_delay>	*n.a.*	*10*	*30*
<M1>	*Base*	*0*	*180*
<M2>	*Shoulder*	*15*	*165*
<M3>	*Elbow*	*0*	*180*
<M4>	*Vertical wrist*	*0*	*180*
<M5>	*Rotating wrist*	*0*	*180*
<M6>	*Gripper*	*10* *open*	*73* *closed*

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

113.7. Sample Tinkerkit Braccio Robot

The following sketch will instruct the Tinkerkit Braccio Robot to pickup a little object and place it on another location, then it picks up a second object and places it on the previous position of the first object. This will the 2 objects will be moved continously between 3 locations.

Sample Connections

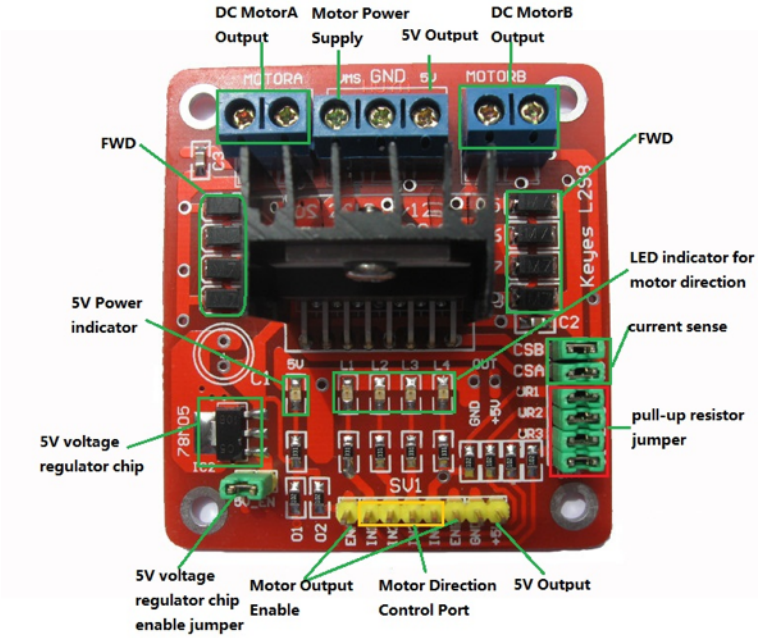
- Connect the Braccio shield on top of an Arduino Uno.
- Connect the Base servo to the shields M1 output
- Connect the Shoulder servo to the shields M2 output
- Connect the Elbow servo to the shields M3 output
- Connect the Vertical Wrist servo to the shields M4 output
- Connect the Rotating Wrist servo to the shields M5 output
- Connect the Gripper servo to the shields M6 output
- Connect the external power supply to the Braccio shield
- Connect the Arduino's USB port to your computer

164_BraccioRobotArm.in

```
#include <Braccio.h>
#include <Servo.h>

Servo base;
Servo shoulder;
Servo elbow;
Servo wrist_rot;
Servo wrist_ver;
Servo gripper;


void setup() {
  Braccio.begin();
  Serial.begin(9600);
}

void loop() {

  Serial.println("Straight up, hand open");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  15);

  Serial.println("Rotate to the right");
  Braccio.ServoMovement(20, 180,  85,  90,  95,  80,  15);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 180,  45,  50,   0,  80,  15);

  Serial.println("Wait 1 second before grabbing object");
  delay(1000);
  Braccio.ServoMovement(20, 180,  45,  50,   0,  80,  60);

  Serial.println("Straight up, hand closed");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  60);

  Serial.println("Rotate to the left");
  Braccio.ServoMovement(20, 0,  85,  90,  95,  80,  60);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 0,  45,  50,   0,  80,  60);
```

```
  Serial.println("Drop object");
  Braccio.ServoMovement(20, 0,  45,  50,   0,  80,  15);

  Serial.println("Straight up, hand open");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  15);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 100,  75,  10,   10,  100,  15);

  Serial.println("Wait 1 second before grabbing object");
  delay(1000);
  Braccio.ServoMovement(20, 100,  75,  10,   10,  100,  60);

  Serial.println("Straight, up hand closed");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  60);

  Serial.println("Rotate to the right");
  Braccio.ServoMovement(20, 180,  85,  90,  95,  80,  60);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 180,  45,  50,   0,  80,  60);

  Serial.println("Drop object");
  Braccio.ServoMovement(20, 180,  45,  50,   0,  80,  15);

  Serial.println("Straight up, hand open");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  15);

  Serial.println("Rotate to the left");
  Braccio.ServoMovement(20, 0,  85,  90,  95,  80,  15);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 0,  45,  50,   0,  80,  15);

  Serial.println("Wait 1 second before grabbing object");
  delay(1000);
  Braccio.ServoMovement(20, 0,  45,  50,   0,  80,  60);

  Serial.println("Straight up, hand closed");
  Braccio.ServoMovement(20, 100,  85,  90,  95,  180,  60);

  Serial.println("Bent over");
  Braccio.ServoMovement(20, 100,  75,  10,   10,  100,  60);

  Serial.println("Drop object");
  Braccio.ServoMovement(20, 100,  75,  10,   10,  100,  15);

}
```

114. DC/Stepper Motor Driver board L298n



Driving a DC motor directly from an Arduino board is not recommended. The current drawn by the motor could be way more than the Arduino can deliver. Another challenge is to reverse the direction of a DC motor, since PWM can only address values from 0..255. Using this Motor Driver board gives a solution to both challenges. The speed of one motor can be changed through 1 PWM output, while the direction can be changed through 2 digital outputs (opposite values). So a total of three digital outputs are needed. You can reduce this to 2 digital outputs when using a TTL inverter (NOT gate), because 2 of the 3 inputs should always have opposite/inverted values, this is called Sign-Magnitude. It is even possible to use only 1 PWM output pin by changing the duty cycle to change both speed and direction (0-49% ➔ reverse, 50% ➔ stop and 51-100% ➔ forward), which is called Locked Antiphase PWM. An article about these techniques can be found at:

http://electronics.stackexchange.com/questions/6912/how-many-control-pins-needed-for-l298n-h-bridge/6914#6914

An excellent description of the L298N board can be found at:

http://www.geekonfire.com/wiki/index.php?title=Dual_H-Bridge_Motor_Driver

114.1. Specifications DC/Stepper Motor Driver board L298n

- Heavy load heat sinks
- Power selection switch
- 2 DC motors
- 1 4 wire dual phase stepper motor
- Motor direction indication LED
- Motor power supply: 6-35V
- Driver peak: 2A

114.2. Connections DC/Stepper Motor Driver board L298n

8 pin header

Pin nr	Name	Description	Arduino pin
1	ENA	Enable motor A (speed control)	PWM
2	IN1	Clockwise	Digital pin
3	IN2	Anti-clockwise	Digital pin
4	IN3	Clockwise	Digital pin
5	IN4	Anti-clockwise	Digital pin
6	ENB	Enable motor B (speed control)	PWM
7	GND	Ground	Not connected
8	+5V	5V	Not connected

114.3. Library DC/Stepper Motor Driver board L298n

None needed.

114.4. Datasheet L298N Motor Driver

https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

114.5. Sample sketch DC/Stepper Motor Driver board L298n with a motor

This is a sample of how to connect 1 single motor to the L298N. With the following sketch Motor A will start turning clockwise (CW) at top speed for 2 seconds, and then it wil turn counter clockwise for 2 seconds.

Connection

- Connect ENA to D5
- Connect IN1 to D3
- Connect IN2 to D2
- Connect VMS to external power supply
- Connect GND to external power supply
- Connect both MotorA pins to Motor A

067_L298N_Motor.ino

```
int MotorA=5;
int MotorA_CCW=2;
int MotorA_CW=3;

int Motorspeed=255;

void setup()
{
 pinMode(MotorA,OUTPUT);
 pinMode(MotorA_CW,OUTPUT);
 pinMode(MotorA_CCW,OUTPUT);
}

void loop()
{
  digitalWrite(MotorA_CW,LOW);
  digitalWrite(MotorA_CCW,HIGH);
  analogWrite(MotorA, Motorspeed);
  delay(2000);
  analogWrite(MotorA, LOW);
  delay(2000);
}
```

114.6. Sample sketch DC/Stepper Motor Driver board L298n with a NEMA-17 stepper

This is a sample of how to connect a NEMA-17 to the L298N. With the following sketch a NEMA-17 stepper rotate 360 degrees clockwise and then 360 degrees counterclockwise.

Connections to the Arduino

- Connect GND to GND.
- Connect In1 to D4
- Connect In2 to D5
- Connect In3 to D6
- Connect In4 to D7

Connections to the Stepper Motor NEMA-17

- Connect 1 pin of MOTORA to the Yellow wire of the NEMA-17
- Connect the other pin of MOTORA to the Red wire of the NEMA-17
- Connect 1 pin of MOTORB to the Green wire of the NEMA-17
- Connect the other pin of MOTORB to the Gray wire of the NEMA-17

Connections to an external power supply

- Connect VMS to 12 V of an external power supply
- Connect GND also to GND of an external power supply

180_L298_NEMA17.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 4, 5, 6, 7);

void setup()
{
  myStepper.setSpeed(60);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(1000);

  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

115. Stepper Motor 28BYJ-48 5V with ULN2003 Interface board

This simple stepper motor is readily available on eBay, best to use a ULN2003 interface board to drive the stepper, often sold together.

115.1. Specifications Stepper Motor 28BYJ-48 5V with ULN2003 Interface board

- 4-phase.
- One bipolar winding is on motor pins 1&3.
- The other is on motor pins 2&4.
- Roughly 2040 steps per revolution.
- Max speed is 14 rpm.

115.2. Datasheets Stepper Motor 28BYJ-48 5V with ULN2003 Interface board

Datasheet ULN2003 Seven Darlington Arrays

http://pdf1.alldatasheet.com/datasheet-pdf/view/25575/STMICROELECTRONICS/ULN2003.html

Datasheet Stepper Motor 28BYJ-49 5V

http://robocraft.ru/files/datasheet/28BYJ-48.pdf

115.3. Connections Stepper Motor 28BYJ-48 5V

4 pin header

Pin nr	Name	Description	Arduino pin
1	IN1	Phase 1	Any Digital ports
2	IN2	Phase 2	Any Digital ports
3	IN3	Phase 3	Any Digital ports
4	IN4	Phase 4	Any Digital ports

2 pin header

Pin nr	Name	Description	Arduino pin
1	-	Ground	GND
2	+	5-12 V (depending on motor)	5V

5 pin header

Connect 5-wire-connector with stepper motor interface board.

115.4. Libraries needed for Stepper Motor 28BYJ-48 5V

- Stepper motor driver through the Library Manager.

Library use explanation

```
#include <Stepper.h>
```

Include the stepper motor library included in Arduino IDE.

```
Stepper myStepper(stepsPerRevolution, IN1, IN3, IN2, IN4);
```

Create 'mystepper' a new instance of the object type Stepper.
IN1..IN4 are integer values corresponding the Arduino Digital Output
to which IN1, IN3, IN2 and IN4 are connected.

```
myStepper.setSpeed(14);
```

Set the speed in rotations per minute. Maximum speed is 14 rpm.

```
myStepper.step(stepsPerRevolution);
```

Turn stepper 1 revolution.

```
myStepper.step(-stepsPerRevolution);
```

Turn stepper 1 revolution backwards.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

115.5. Sample Stepper Motor 28BYJ-48 5V

The following sketch rotates the stepper 1 turn CW (clockwise) and then 1 turn CCW (counter clockwise) and repeats this.

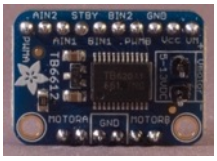
Connections

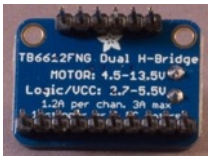
- Connect VCC to 5V.
- Connect GND to GND.
- Connect IN1 to D8.
- Connect IN2 to D9.
- Connect IN3 to D10.
- Connect IN4 to D11.

068_Stepper_28BYJ-48-5V.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 2040;

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup()
{
  myStepper.setSpeed(14);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(1000);

  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

116. Adafruit TB6612 Stepper/motor driver





With this Adafruit TB6612 Stepper/motor driver you can either drive 4 solenoids, 2 separate DC motors or uni polar steppers or 1 bi-polar stepper motor with 1.2A per channel. It contains 2 full H-bridges (four half H-bridges).

116.1. Specifications Adafruit TB6612 Stepper/motor driver

- Power supply voltage VM = 15V max
- Output current: Iout = 1.2A average per channel, 3.2A peak
- Logic level: 2.7 - 5V

116.2. Datasheet TB6612

- https://cdn-shop.adafruit.com/datasheets/TB6612FNG_datasheet_en_20121101.pdf

116.3. Connections Adafruit TB6612 Stepper/motor driver

Name	Description	Arduino
MotorA	Output to Motor(coil) A controlled by INA1, INA2 and PWMA	
MotorA	Output to Motor(coil) A controlled by INA1, INA2 and PWMA	
GND	Ground	GND
GND	Ground	
MotorB	Output to Motor(coil) B controlled by INA1, INA2 and PWMA	
MotorB	Output to Motor(coil) B controlled by INA1, INA2 and PWMA	
GND	Ground	
GND	Ground	
VMotor-	Power supply Motor Ground	
VMotor+	Power supply Motor Plus (4,5-13,4V)	
VM		
VCC	Power supply TB6612 logic levers	5V
GND	Ground	
PwmB	PWM input to Motor(coil) B H-bridges, if PWM is not needed, connect to VCC	Any PWM port or 5V
BIn2	Input 2 to Motor(coil) B H-bridges	Any digital port
BIn1	Input 1 to Motor(coil) B H-bridges	Any digital port
Stdby	Standby	
AIn1	Input 1 to Motor(coil) A H-bridges	Any digital port
AIn2	Input 2 to Motor(coil) A H-bridges	Any digital port
PwmA	PWM input to Motor(coil) A H-bridges, if PWM is not needed, connect to VCC	Any PWM port or 5V

116.4. Libraries needed for Adafruit TB6612 Stepper/motor driver

- Stepper motor driver through the Library Manager.

Library use explanation

```
#include <Stepper.h>
```

Include the stepper motor library included in Arduino IDE.

```
Stepper myStepper(stepsPerRevolution, IN1, IN3, IN2, IN4);
```

Create 'mystepper' a new instance of the object type Stepper.
IN1..IN4 are integer values corresponding the Arduino Digital Output
to which IN1, IN3, IN2 and IN4 are connected.

```
myStepper.setSpeed(14);
```

Set the speed in rotations per minute. Maximum speed is depends on the stepper motor, for the NEMA-17 of the next chapter, this is roughly 280 rpm.

```
myStepper.step(stepsPerRevolution);
```

Turn stepper 1 revolution.

```
myStepper.step(-stepsPerRevolution);
```

Turn stepper 1 revolution backwards.

Library use explanation

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

116.5. Sample Adafruit TB6612 Stepper/motor driver

The following sketch is used with the NEMA-17 stepper motor of the next chapter.

Sample Connections

Connections to the Arduino

- Connect VCC to 5V.
- Connect GND to GND.
- Connect AIn2 to D4
- Connect AIn1 to D5
- Connect BIn1 to D6
- Connect BIn2 to D7
- Connect PWMA to 5V
- Connect PWMB to 5V

Connections to the Stepper Motor

- Connect Motor A to the Red wire of the Stepper Motor
- Connect the other Mother A to the Yellow wire of the Stepper Motor
- Connect Motor B to the Green wire of the Stepper Motor
- Connect the other Mother B to the Gray wire of the Stepper Motor

Connections to an external power supply

- Connect VM+ to 12 V of an external power supply
- Connect VN- to GND of an external power supply

069_TB6612_NEMA17.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 4, 5, 6, 7);

void setup()
{
  myStepper.setSpeed(30);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(1000);

  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

116.6. Sample Adafruit TB6612 Stepper/motor driver with a motor

This shows how to connect a motor to the TB6612. This sketch rotates the motor clockwise for 2 seconds and then 2 seconds counterclockwise.

Connections to the Arduino

- Connect GND to GND
- Connect PWMA to D5
- Connect AIN1 to D3
- Connect AIN2 to D2

Connections to the Motor

- Connect 1 pin of MOTORA to the Red wire of the motor
- Connect the other pin of MOTORA to the Black wire of the motor

Connections to an external power supply

- Connect VM+ to 5 V of an external power supply
- Connect VN- to GND of an external power supply

178_TB6612_Motor.ino

```
int MotorA=5;
int MotorA_CCW=2;
int MotorA_CW=3;

int Motorspeed=255;

void setup()
{
 pinMode(MotorA,OUTPUT);
 pinMode(MotorA_CW,OUTPUT);
 pinMode(MotorA_CCW,OUTPUT);
}

void loop()
{
  digitalWrite(MotorA_CW,LOW);
  digitalWrite(MotorA_CCW,HIGH);
  analogWrite(MotorA, Motorspeed);
  delay(2000);
  digitalWrite(MotorA_CW,HIGH);
  digitalWrite(MotorA_CCW,LOW);
  analogWrite(MotorA, Motorspeed);
  delay(2000);
}
```

117. Pololu A4988 Stepper driver

This stepper driver from Pololu is often used in DIY CNC or 3D printers.

117.1. Specifications Pololu A4988 Stepper driver

- Logic Supply voltage, Vdd: 2.8-5.5V
- Load Supply voltage: 8-35V
- Output current per phase: 0-2A (can be limited by a potentiometer)
- Microsteps resolutions: 1, 1/2, 1/4, 1/8, 1/16

117.2. Datasheet Pololu A4988 Stepper Driver

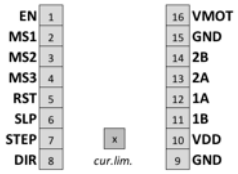
Datasheet:

- https://www.pololu.com/file/0J450/A4988.pdf

A very good explanation for this stepper driver can be found at:

- https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/

117.3. Connections Pololu A4988 Stepper Driver



Pin nr	Name	Description	Meaning	Arduino pin
1	EN	Enable Default: Low	Low: Enabled High: Disabled	Digital Port, 5V or GND/floating
2	MS1	Microstep 1 Default Low	See microsteps table.	Digital Port, 5V or GND/floating
3	MS2	Microstep 2 Deafult: Low	See microsteps table.	Digital Port, 5V or GND/floating
4	MS3	Microstep 3 Default: Low	See microsteps table.	Digital Port, 5V or GND/floating
5	RST	Reset Default: floating	Low: Ignore steps and reset motor to initial position High: Accept steps	Digital Port, 5V or GND Often directly connected to SLP
6	SLP	Sleep Default: HIGH	Low: Sleep High: Awake	Digital Port, GND or 5V/floating
7	STEP	Step: Default: Floating	Each high pulse, steps the motor by number of microsteps set by microsteps selection pins	Any Digital port
8	DIR	Direction Default: Floating	High: CW Low: CCW	Any Digital port
9	GND	Ground		Ground
10	VDD	Power supply		5V
11	1B	Coil 1		-
12	1A	Coil 1		-
13	2A	Coil 2		-

Pin nr	Name	Description	Meaning	Arduino pin
14	2B	Coil 2		-
15	GND	Ground		Ground
16	VMOT	Power supply motor		External Power supply 8-35V

Microsteps settings Pololu A4988 Stepper Driver

Carefully check the settings below, the table is sorted on microsteps and not on the binary representation of the pins MS1-MS3.

MS1	MS2	MS3	Microsteps
0	0	0	full (1)
1	0	0	1/2
0	1	0	1/4
1	1	0	1/8
1	1	1	1/16
1	0	1	n.a.
0	1	1	n.a.
0	0	1	n.a.

117.4. Libraries needed for Pololu A4988 Stepper Driver

- You can install StepperDriver by Laurentlu Badea through the Library Manager
- You can also work with this Stepper Driver without using a library (take a look at the second sample sketch 183_A4988_Stepper.ino.

Library use explanation

```
#include "BasicStepperDriver.h"
```

Include the BasisStepperDriver by Laurentlu Badea.

```
#define MOTOR_STEPS 320
```

This value depends on the Stepper Motor you are using. With my DVD-Stepper motor, I found out that 320 steps moves the shaft on complete movement. This is not realy 320 steps per rotation, but 320 steps per complete movement.

```
#define RPM 120
```

Speed for the stepper

```
#define MICROSTEPS 1
```

Set this according to the settings on the Microstep Settings pin (see table).

```
#define DIR 2
```

The pin connected to DIR.

```
#define STEP 3
```

The pin connected to STEP

```
BasicStepperDriver stepper(MOTOR_STEPS, DIR, STEP);
```

Initialize an stepper, on instance of BasicStepperDriver, with MOTOR_STEPS, DIR-pin and STEP-pin.

```
stepper.begin(RPM, MICROSTEPS);
```

Define the current position of the stepper motor as home.

```
stepper.rotate(360);
```

Rotate the stepper for one complete rotation (360 degrees), or in the case of a DVD-Stepper Motor, a complete movement.

```
stepper.move(-MOTOR_STEPS*MICROSTEPS);
```

Another way to rotate the stepper, in this case CCW, one complete rotation (360 degrees), or in the case of a DVD-Stepper Motor, one complete movement.

```
stepper.disable();
```

Disable the stepper, it should be possible to move the stepper bij hand (not the case with my stepper motors).

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

117.5. Sample Pololu A4988 Stepper Driver

The following sketch moves the stepper-motor for one complete revolution CW and then CCW.

Sample Connections

- Connect VDD to 5V
- Connect GND to GND
- Connect DIR to D2
- Connect STEP to D3
- Connect SLP to RST
- Connect both wires from Coil_A of a DVD Stepper motor to A1 &A2
- Connect both wires from Coil_B of a DVD Stepper motor to B1 &B2
- Connect Vmot to a 9 V external power supply (depending on the Stepper Motor)
- Connect GND to the external Power's ground

183_Stepper_A4988_Library

```
#include "BasicStepperDriver.h"

#define MOTOR_STEPS 320
#define RPM 20

#define MICROSTEPS 1

#define DIR 2
#define STEP 3
BasicStepperDriver stepper(MOTOR_STEPS, DIR, STEP);

void setup()
{
    stepper.begin(RPM, MICROSTEPS);
}

void loop()
{
    stepper.rotate(1200);
    stepper.disable();
    delay(5000);
}
```

117.6. Sample Pololu A4988 Stepper Driver without using a library

The following sketch moves the stepper-motor for one complete revolution CW and then CCW.

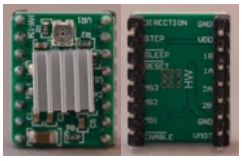
Sample Connections

- Connect VDD to 5V
- Connect GND to GND
- Connect DIR to D2
- Connect STEP to D3
- Connect SLP to RST
- Connect both wires from Coil_A of a DVD Stepper motor to A1 &A2
- Connect both wires from Coil_B of a DVD Stepper motor to B1 &B2
- Connect Vmot to a 9 V external power supply (depending on the Stepper Motor)
- Connect GND to the external Power's ground

182_Stepper_A4988

```
const int dirPin = 2;
const int stepPin = 3;
const int stepsPerRevolution = 320;

void setup()
{
  pinMode(stepPin, OUTPUT);
  pinMode(dirPin, OUTPUT);
}

void loop()
{
  digitalWrite(dirPin, HIGH);
  for(int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(10000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(10000);
  }
  delay(1000);

  digitalWrite(dirPin, LOW);
  for(int x = 0; x < stepsPerRevolution; x++)
  {
    digitalWrite(stepPin, HIGH);
    delayMicroseconds(10000);
    digitalWrite(stepPin, LOW);
    delayMicroseconds(10000);
  }
  delay(1000);
}
```

118. Yongfukang HR4988 Stepper driver



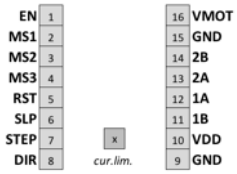
This is a cheap clone of the popular A4988 Stepper Driver made by Shenzhenshi Yongfukang Technology Co., Ltd. They differ in specs and the microsteps settings. Although this HR4988 has a larger range of microsteps settings, some of them are very noisy and probably also very inaccurate/stable.

Some web-shops sell this incorrectly as A4988 Stepper Drivers

118.1. Specifications Yongfukang HR4988 Stepper driver

- Logic Supply voltage, Vdd: 2.8-5.5V
- Load Supply voltage: 8-35V
- Output current: 0-1.8A (can be limited by a potentiometer)
- Microsteps resolutions: 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128

118.2. Connections Yongfukang HR4988 Stepper driver



Th HR4988 is pin compatable with the A4988. For a detailed description take a look in the previous chapter.

118.3. Datasheet Yongfukang HR4988 Stepper driver

Datasheet:

- http://www.szczkjgs.com/UploadFiles/fujian/3721/HR4988.pdf

A very good explanation of this stepper driver can be found at:

- https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/

118.4. Connections Yongfukang HR4988 Stepper driver

The connections of this HR4988 are the same as for the A4988 in the previous chapter.

Microsteps settings

Carefully check the settings below, the table is sorted on microsteps and not on the binary representation of the pins MS1-MS3.

MS1	MS2	MS3	Microsteps
0	0	0	full (1)
1	0	0	1/2
0	1	0	1/4
1	1	0	1/8
1	1	1	1/16
1	0	1	1/32
0	1	1	1/64
0	0	1	1/128

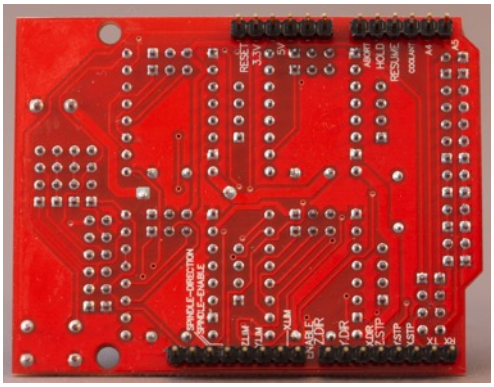
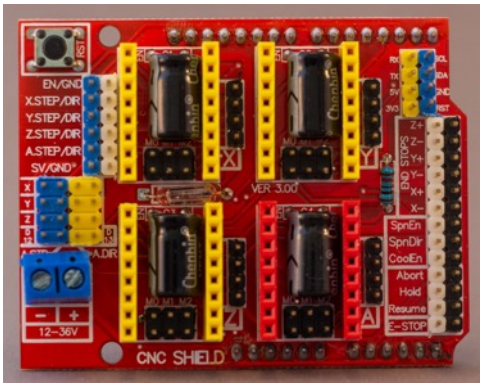
118.5. Libraries needed for Yongfukang HR4988 Stepper driver

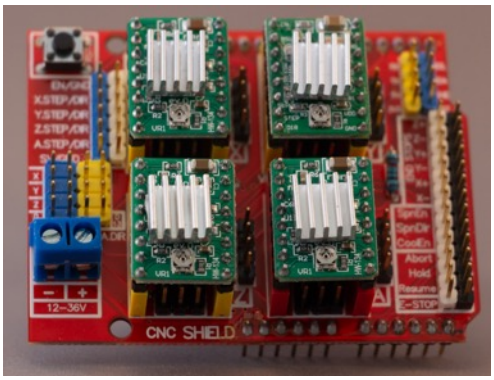
Check the previous chapter (A4988) for the use of libraries.

118.6. Sample Yongfukang HR4988 Stepper driver

Check the previous chapter (A4988) for a sample sketch.

119. CNC Shield v3.0



This stepper driver from Pololu is often used in DIY CNC or 3D printers.

119.1. Specifications CNC Shield v3.0

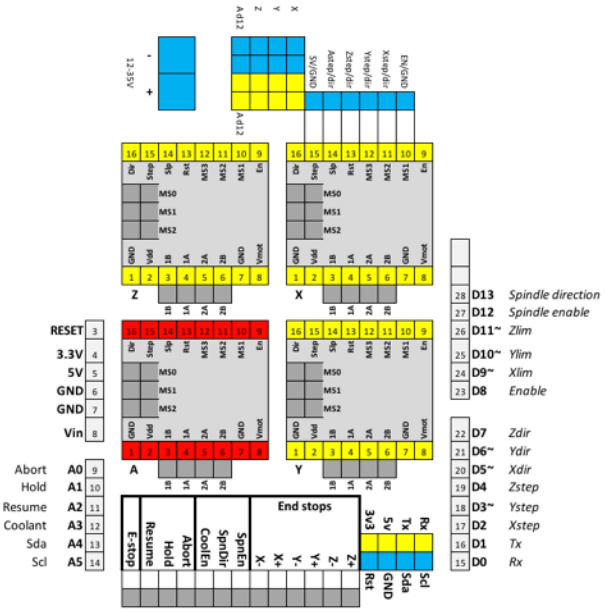
- 4-axis support.
- A axis can duplicate X, Y or Z or or function as 4th axis (custom firmware, because GRBL doesn't support 4 axis)
- Connector for switches (normaly open, active low)
  - 2 end stops for each axis (X, Y & Z). Each axis pair shares the same IO pin
  - Spindle enblae
  - Spindle direction
  - Abort
  - Hold
  - Resum
  - Emergency stop
- GRBL firmware v0.8 compatible (v0.9 & v1.1 with small modifications to config.h, as described in the datasheet listed below)
- Load Supply voltage: 12-36V, be careful when using HR4988/A4988 stepper drivers since they can not exceed 24 V
- Use removable stepper drivers HR4988, A4988 or DRV8825, take a good look at their orientation, because the current limiting potentiometer is oriented differently on the DRV8825 compared to the HR4988/A4988. Orientate the stepper drivers by looking at the Enable pin.
- Stepper motors can be connected with 4 pin dupont connectors
- Output current per phase: depending on the stepper driver
- Microsteps resolutions: depending on the stepper driver (can be set through jumper blocks for each stepper driver individually).

119.2. Datasheet CNC Shield v3.0

Datasheet:

- https://www.makerstore.com.au/download/publications/CNC-Shield-Guide-v1.0.pdf

119.3. Connections Pololu A4988 Stepper Driver



119.4. Libraries needed for CNC Shield v3.0

- You can install StepperDriver by Laurentlu Badea through the Library Manager
- You could also manage without using a library, but it is not recommend when using multiple steppers.

Library use explanation

```
#include "BasicStepperDriver.h"
```

Include the BasisStepperDriver by Laurentlu Badea.

```
#define X_MOTOR_STEPS 320
```

This value depends on the Stepper Motor you are using. With my DVD-Stepper motor, I found out that 320 steps moves the shaft on complete movement. This is not realy 320 steps per rotation, but 320 steps per complete movement.

```
#define X_RPM 20
```

Speed for the stepper

```
#define X_MICROSTEPS 1
```

Set this according to the settings on the Microstep Settings pin (see table).

```
#define X_DIR 5
```

The pin connected to DIR.

```
#define X_STEP 2
```

The pin connected to STEP

```
#define ENABLE 8
```

The pin connected to ENABLE

```
BasicStepperDriver X_stepper(X_MOTOR_STEPS, X_DIR, X_STEP);
```
Initialize an stepper, on instance of BasicStepperDriver, with MOTOR_STEPS, DIR-pin and STEP-pin.

```
X_stepper.begin(X_RPM, X_MICROSTEPS);
```
Define the current position of the stepper motor as home.

```
X_stepper.rotate(360);
```
Rotate the stepper for one complete rotation (360 degrees), or in the case of a DVD-Stepper Motor, a complete movement.

```
X_stepper.move(-X_MOTOR_STEPS* X_MICROSTEPS);
```
Another way to rotate the stepper, in this case CCW, one complete rotation (360 degrees), or in the case of a DVD-Stepper Motor, one complete movement.

```
X_stepper.disable();
```
Disable the stepper, it should be possible to move the stepper bij hand (not the case with my stepper motors).

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

119.5. Sample CNC Shield v3.0

The following sketch moves the stepper-motor for one complete revolution CW and then CCW.

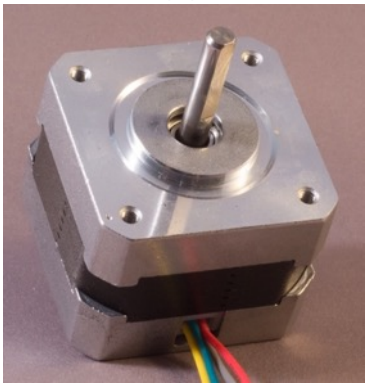
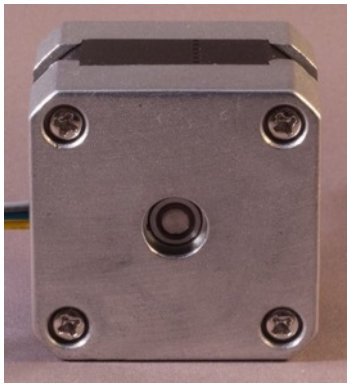
Sample Connections

- Place a Stepper Driver like the A4988 or HR4988 to X
- Connect both wires from Coil_A of a DVD Stepper motor to X A1 &A2
- Connect both wires from Coil_B of a DVD Stepper motor to X B1 &B2
- Connect + on the terminal block to a 9 V external power supply (depending on the Stepper Motor)
- Connect - on the terminal block to GND to the external Power's ground

183_Stepper_A4988_Library

```
#include "BasicStepperDriver.h"

#define X_MOTOR_STEPS 320
#define X_RPM 20

#define X_MICROSTEPS 1

#define X_DIR 5
#define X_STEP 2
#define ENABLE 8
BasicStepperDriver X_stepper(X_MOTOR_STEPS, X_DIR, X_STEP);

void setup()
{
  pinMode (ENABLE, OUTPUT);
  digitalWrite (ENABLE, LOW);
  X_stepper.begin(X_RPM, X_MICROSTEPS);
}

void loop()
{
  X_stepper.rotate(360);
  delay(2000);
  X_stepper.move(-X_MOTOR_STEPS * X_MICROSTEPS);
  X_stepper.disable();
  delay(2000);
}
```

120. Stepper motor NEMA-17

This 42 mm high torque hybrid stepper motor can be driven by the Adafruit TB6612 Stepper/motor driver from the previous chapter.

120.1. Specifications Stepper motor NEMA-17

- Bipolar steppers, requires 2 full H-bridges
- 200 steps/revolution, 1,8 degree per step
- 5% step accuracy (non-load)
- Coil #1: Red & Yellow wire pair
- Coil #2: Green and Gray wire pair
- Power supply: 12V (lower is possible, but with lower torque)
- Current: 350 mA.
- 2 Kg/cm torque per phase
- 35 ohms per winding

120.2. Datasheet Stepper motor NEMA-17

- https://cdn-shop.adafruit.com/product-files/324/C140-A+datasheet.jpg

120.3. Connections Stepper motor NEMA-17

Pin nr	Name	Description	Connection on stepper drive
1	Red wire	Coil #1	MotorA
2	Yellow wire		MotorA
3	Green wire	Coil #2	MotorB
4	Gray wire		MotorB

120.4. Sample Stepper motor NEMA-17

This is a copy of the sample for the Adafruit TB6612 Stepper/motor driver in the previous chapter. Used stepsPerRevolution = 200 and a maximum speed of 280.

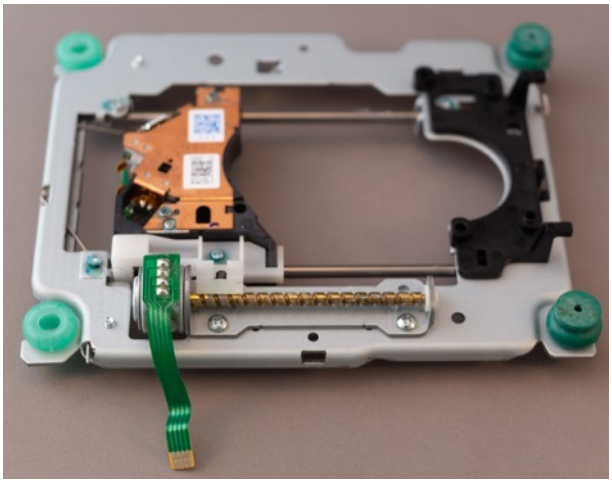
Connections

DON't connect the 4 wires from the stepper motor directly to an Arduino, use a Stepper driver like the L298N or the TB6612. See the corresponding chapters for the right connections.

178_Stepper_TB6612_NEMA17.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 200;

Stepper myStepper(stepsPerRevolution, 4, 5, 6, 7);

void setup()
{
  myStepper.setSpeed(280);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(1000);

  myStepper.step(-stepsPerRevolution);
  delay(1000);
}
```

121. Stepper motor from a DVD player



This frame with stepper motor has been salvaged from on old DVD-player. It used to move the laser back and forth, while a brushless motor spinned the DVD-disc. A lot of makers use these stepper motors to build small CNC devices.

121.1. Specifications Stepper motor from a DVD player

- Bipolar steppers, requires 2 full H-bridges
- Unknown steps/revolution
- 2 coils
- Power supply: 5V?
- 14 ohms per coil

121.2. Connections Stepper motor from a DVD player

Pin nr	Description	Connection on stepper drive
1	Coil #1	MotorA
2		MotorA
3	Coil #2	MotorB
4		MotorB

121.3. Sample Stepper motor from a DVD player

The number of steps per revolution is unknown, but setting it to 240-240 steps per revolution, it equals to one complete movement up or down.

Connections

DON't connect the 4 wires from the stepper motor directly to an Arduino, use a Stepper driver like the L298N or the TB6612. See the corresponding chapters for the right connections.

181_Stepper_DVD.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 240;

Stepper myStepper(stepsPerRevolution, 4, 5, 6, 7);

void setup()
{
  myStepper.setSpeed(80);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  myStepper.step(-stepsPerRevolution);
  delay(2000);
}
```

122. Stepper motor from a Floppy disc drive

This frame with stepper motor has been salvaged from on old floppy disc drive. It used to move the heads back and forth, while a brushless motor spinned the floppy disc.

122.1. Specifications Stepper motor from a DVD player

- Bipolar steppers, requires 2 full H-bridges
- Unknown steps/revolution
- 2 coils
- Power supply: 5V?
- 17 ohms per coil

122.2. Connections Stepper motor from a DVD player

Pin nr	Description	Connection on stepper drive
1	Coil #1	MotorA
2		MotorA
3	Coil #2	MotorB
4		MotorB

122.3. Sample Stepper motor from a DVD player

The number of steps per revolution is unknown, but setting it to 240 steps per revolution, it equals to one complete movement up or down.

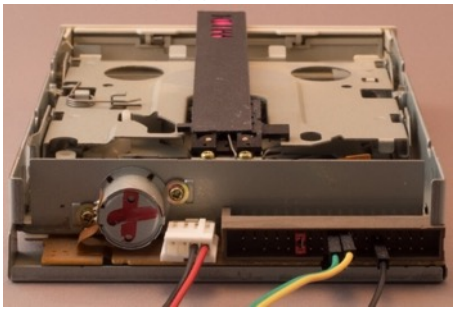
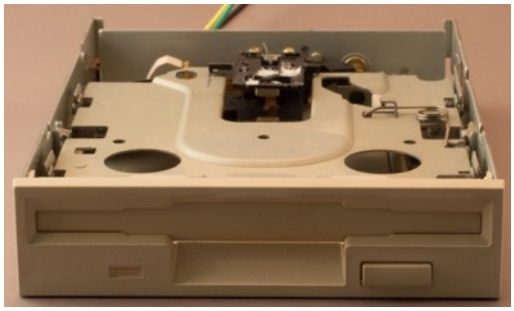
Connections

DON't connect the 4 wires from the stepper motor directly to an Arduino, use a Stepper driver like the L298N or the TB6612. See the corresponding chapters for the right connections.

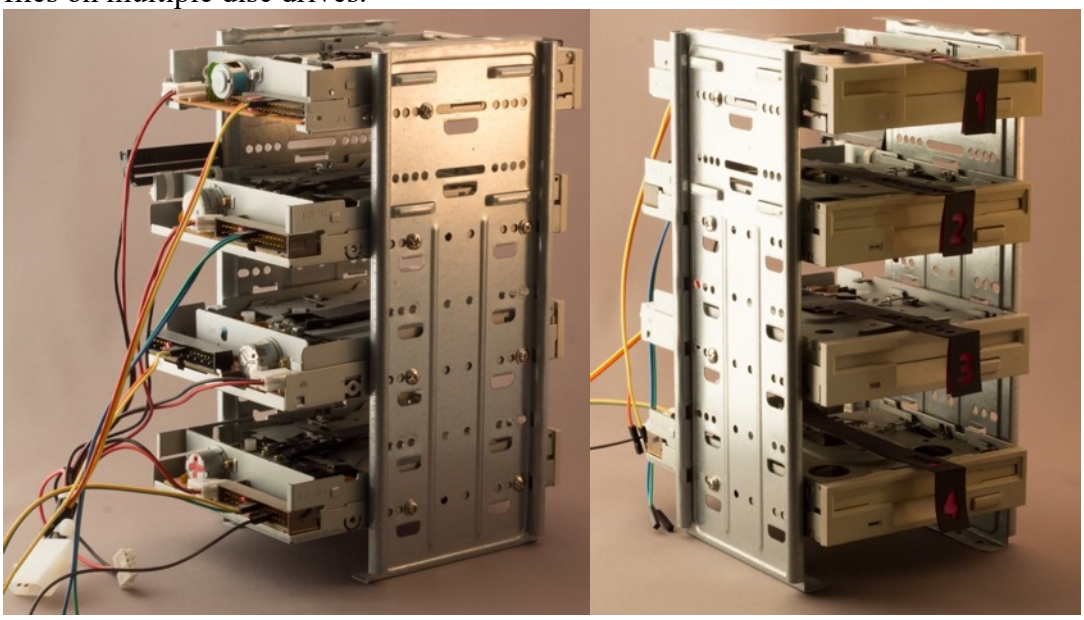
181_Stepper_DVD.ino

```
#include <Stepper.h>

const int stepsPerRevolution = 240;

Stepper myStepper(stepsPerRevolution, 4, 5, 6, 7);

void setup()
{
  myStepper.setSpeed(80);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  myStepper.step(-stepsPerRevolution);
  delay(2000);
}
```

123. Floppy disc drive

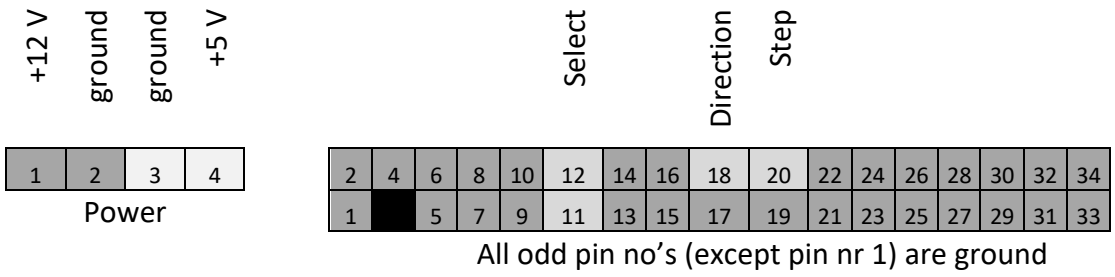



Floppy disc drives (both 3.5 and 5.25 inch) contains a stepper motor to move the read/write heads. This stepper motor can be used in Arduino object for building robots, but also as simple musical instrument. It is not recommended to connect this stepper motor directly to our Arduino. You would need some kind of motor driver, but fortunately, floppy drives are always equipped with some kind of motor driver under the hood. By connecting jumper wires to the 34 pins header (3,5 inch drives) you can control the stepper motor with your Arduino by using 2 digital I/O pins. Preferably you'll need an external +5V power source for the floppy drive (see "208 ATX Power Supply"). There is a project on the Internet to play MIDI files on multiple disc drives.



123.1. Connections

There are 2 connectors on the back of a floppy disc drive. The smallest one (4 pins) is for the power source, the other one (2 rows of 17 pins) is for controlling the motors and reading/writing data. Only the following pins are needed:



123.2. Libraries needed for Floppy disc drive

None needed

123.3. Sample Floppy Disc Drive

The following sample sketch drives the read/write head repeating 200 steps inwards and 200 steps outwards.

Sample Connections

- Connect Floppy drive pin 12 and Floppy drive pin 11 to select drive A.
- Connect Floppy drive pin18 with D12.
- Connect Floppy drive pin 20 with D13.
- Connect Floppy power pin 3 with ground.
- Connect Floppy power pin 4 with +5 V (external)
- Connect External GND with Arduino GND.

070_Stepper_FloppyDrive.ino

```
#define DELAY 1200
#define STEPS 200

int dir_pin = 12;
int step_pin = 13;

void setup()
{

  pinMode(dir_pin, OUTPUT);
  pinMode(step_pin, OUTPUT);
}

void loop()
{
  delay(1);
  digitalWrite(dir_pin, HIGH);
  delay(1);
  perform_step(STEPS);
  delay(500);
  digitalWrite(dir_pin, LOW);
  delay(1);
  perform_step(STEPS);
  delay(500);
}

void perform_step(long steps)
{
  for (long i = 0; i < steps; i++)
  {
    digitalWrite(step_pin, LOW);
    delayMicroseconds(100);
    digitalWrite(step_pin, HIGH);
    delayMicroseconds(DELAY);
  }
  digitalWrite(step_pin, LOW);
}
```

124. Vibration motor

This small vibration motor is suitable as a non-audible indicator, just as in your smartphone. It vibrates as long as the input is high.

124.1. Specifications Vibration motor

- Rated voltage: 5V
- Operating voltage: 3.0-5.3V
- Speed: 9000 rpm
- Current: 60 mA (I've measured only 0.6 mA)
- Start current: 90 mA (I've measured only 0.9 mA)

124.2. Datasheet Vibration motor

None found.

124.3. Connections Heartbeat sensor

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	VCC	Power Supply 5V	5V
3	IN	Signal	Any digital pin

124.4. Libraries needed for Vibration motor

No specific libraries needed.

124.5. Sample Vibration motor

The following sketch turns the vibration motor 3 seconds on and 1 second off.

Sample Connections

- Connect GND to GND
- Connect VCC to 5V
- Connect IN to D9

172_VibrationMotor.ino

```
#define VIBRATIONMOTOR 9

void setup() {
  pinMode(VIBRATIONMOTOR, OUTPUT);
}

void loop() {
  digitalWrite(VIBRATIONMOTOR, HIGH);
  delay(3000);
  digitalWrite(VIBRATIONMOTOR, LOW);
  delay(1000);
}
```

Camera triggers

In this section you will find some examples in which you can use an Arduino with triggering a camera.

125. Self-made trigger cable for Canon DSLR camera's

125.1. Specifications trigger cable for Canon DSLR camera's

http://www.doc-diy.net/photo/remote_pinout/#canon

125.2. Connections trigger cable for Canon DSLR camera's

Pin nr	Name	Description	Arduino pin
1	SHUT	Shutter	
2	FOC	Focus	
3	GND	Ground	

125.3. Libraries needed for trigger cable for Canon DSLR camera's

None needed.

125.4. Sample trigger cable for Canon DSLR camera's

The following script and sketch controls the shutter of the camera through an Arduino and takes a picture every 10 seconds.

Sample connections

-

Sample Arduino sketch Single trigger

126. Self-made trigger cable for Sony ILCE system camera

Sony's ILCE system camera's like the Alpha A7, A7R, A7ii, A3000, A5000, A5100 and 6000 can be triggered through a proprietary micro-USB connector called 'Multiport'. This proprietary micro-USB connector differs in the fact that it has 10 contacts extra compared to the original 5 pin micro-USB connector. Those extra contacts are for shutter and focus and other functions not studied by me yet. There are 2 ways to get such a Multiport connector:

- Buy an RM-VPR wired shutter release cable and cut of the big switch. The cable connected to the Multiport connector consists of 3 wires already to the only 3 pins you'll need (ground, focus and shutter). For me this was the way to go. Easy and cheap, you can find it in China for prices as low as $ 3,-.
- Buy a Multiport 15 pin plug, they come with a breakout board, so you can solder your own cable to any of the 10+5 contacts, but in this chapter I'll only be using ground, focus and shutter. The price for 1 plug is about the same as the cheap RM-VPR: https://www.studio1productions.com/parts/sony-multiport-connector.htm

126.1. Specifications trigger cable for Sony ILCE system camera

http://www.doc-diy.net/photo/remote_pinout/#sony

126.2. Connections trigger cable for Sony ILCE system camera

Pin nr	Name	Description	Arduino pin
1	Power On/Off	-	not used
2	Ground	Ground	
3	Composite Video Out	*Video is not supported when using this connector to trigger the shutter!*	not used
4	Audio Out L Shutter	Shutter *Audio is not supported when using this connector to trigger the shutter!*	
5	Audio Out R Activate Camera Focus	Focus *Audio is not supported when using this connector to trigger the shutter!*	
6	Select	-	not used
7	UART RX Boot Serial IN	-	not used
8	UART TX LANC sig	-	not used
9	XReset Req	-	not used
10	2.8-3.3V output	-	not used

https://www.studio1productions.com/parts/sony-multiport-connector.htm

126.3. Libraries needed for trigger cable for Sony ILCE system camera

None needed.

126.4. Sample trigger cable for Sony ILCE system camera

The following script and sketch controls the shutter of the camera through an Arduino and takes a picture every 10 seconds.

Sample connections

-

Sample Arduino sketch Single trigger

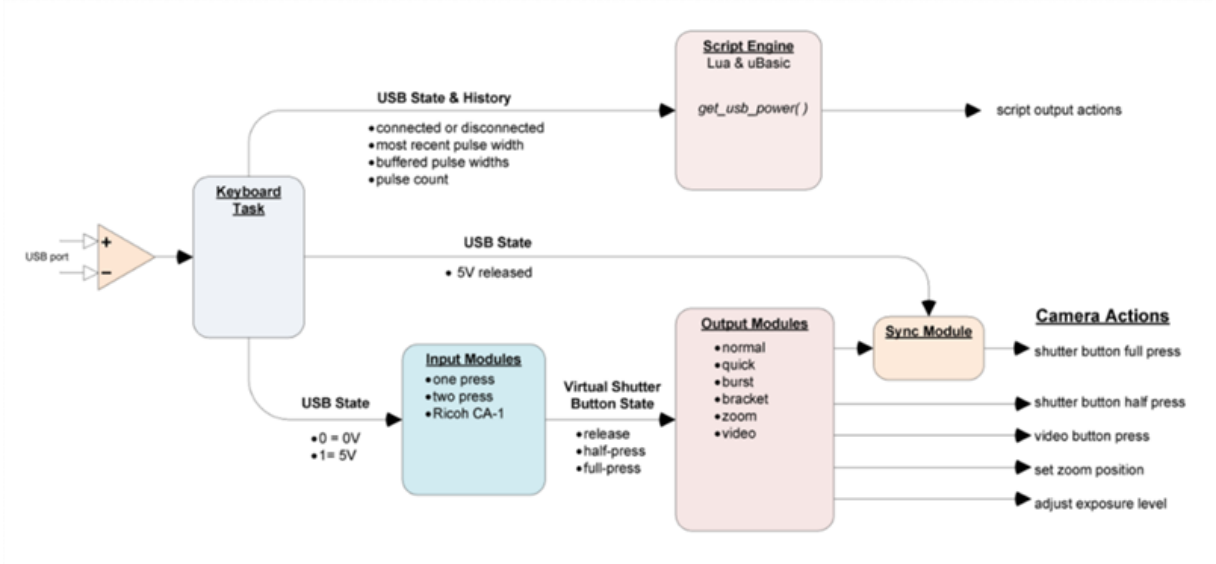
```
```

127. Self-made Canon CHDK/SDM trigger cable

This modified USB cable can trigger a Canon compact camera loaded with the CHDK or SDM firmware. When sending specified pulses to the USB port, a script loaded on the camera can even distinguish several different levels.

127.1. Specifications Self-made Canon CHDK/SDM trigger cable

Depending on the settings of CHDK you can use this cable to just trigger the camera to take a picture or to trigger a maximum of 18 different events (6 different levels is much more reliable).



Pict. 1 Source **http://chdk.wikia.com/wiki/USB_Remote**

127.2. Connections Canon CHDK/SDM trigger cable

Take a USB A to mini-USB cable and cut off the USB-A connector. You'll only need the red and black wire. I've connected those 2 wires to a 3 pin female servo-like connector.

Pin nr	Name	Description	Arduino pin
1	S (red)	Signal	Any digital port
2	nc	not connected	nc
3	GND (black)	Ground	GND

127.3. Libraries needed for Self-made Canon CHDK/SDM trigger cable

None needed on the Arduino, but you need to (non-permanent) install the CHDK firmware on your Canon camera.

Preparing a Canon compact camera with CHDK

To install CHDK you will need:

- A Canon compact camera compatible with CHDK, see the list of supported cameras at http://chdk.wikia.com/wiki/CHDK.
- The correct CHDK build for your Camera's firmware.
- A SD-card with a "lock-switch" (read-only). Most SD-cards have such a "lock-switch", but some cheap cards and some WiFi-SD cards like the EyFi cards are not equipped with a "lock-switch".
- STICK, the Simple Tool for Installing CHDK. Stick is an application written in JAVA, so it is compatible with Windows, Mac OSx as well as Linux.
    - Download link: http://zenoshrdlu.com/stick/stick.zip
    - Instructions: http://www.zenoshrdlu.com/stick/stick.html

After preparing your SD-card with STICK, put the "lock-switch" in the LOCK position and load the SD-card in your camera. After power on, your camera will show the CHDK logo and some information about the camera and its firmware.[1]

Now you can prepare the camera to accepts trigger signals from the USB cable.
ALT[2], MENU, CHDK Settings, Remote Parameters

- Enable Remote: ON
- Switch Type: None
- Control Type: None

Final step is to place an uBasic[3] or lua[4] script on your SD card in /CHDK/SCRIPTS (remove SD from camera, unlock card, place SD in cardreader, copy script to SD card, remove from cardreader, lock card, place SD in camera and power on camera).

To load the uBasic or Lua script follow the next steps:

- ALT-SET
- Load Script from File ....
- Select the script from the folder /CHDK/SCRIPTS and press the SET button
- Select BACK.
- To start the script you need to full press the shutter button.

---

[1] If your camera says "Memory Card Locked" it means that the SD-card is not bootable and/or CHDK is not correct installed. Check the CHDK wiki to correct this.
[2] The ALT key is one of the buttons at the back of your camera. After pressing the correct button, you will see ALT on the bottom line of the display. At that point most buttons will have an alternate function.
[3] http://chdk.wikia.com/wiki/UBASIC
[4] http://chdk.wikia.com/wiki/Lua

127.4. Sample Single trigger

The following Arduino sketch and corresponding CHDK Lua script controls the shutter of the camera through an Arduino and takes a picture every 10 seconds.

Sample connections

- Connect the red wire with D9.
- Connect the black wire with GND
- Connect the USB connector to a Canon Camera with the CHDK firmware
- After uploading the following sketch, continue with the camera preparations on the next page

071_CameraCanonCHDK.ino

```
int LED=13;
int CHDKport=9;

void setup() {
  pinMode(LED, OUTPUT);
  pinMode(CHDKport, OUTPUT);
  Serial.begin(9600);
  Serial.println("Ready to take pictures");
}

void loop() {
  delay(10000);
  Serial.println("shoot");
  digitalWrite(LED, HIGH);
  digitalWrite(CHDKport, HIGH);
  delay(100);
  digitalWrite(LED, LOW);
  digitalWrite(CHDKport, LOW);
}
```

Camera preparations

- Install the right firmware for your camera on an SD card
- Copy the CHDK_001_CameraCanon.lua on the SD card in the \CHDK\SCRIPTS folder (PHOTOS partition)
- Set the READ/WRITE-switch of the SD-card to READ (this will trigger the CHDK firmware)
- Place the SD-card in your camera.
- Turn the camera on
- Press the ALT[1]-key, MENU, CHDK Settings, Remote Parameters
    - Enable Remote: ON
    - Switch Type: None
    - Control Type: None
- Press ALT-SET
    - Load Script from File ….
    - Select the CHDK_001_CameraCanon.lua script from the folder /CHDK/SCRIPTS and press the SET button
    - Select BACK.
- To start the script you need to full press the shutter button.

CHDK_001_CameraCanon.lua

```
--[[
@title Single shot
@chdk_version 1.4
--]]

while (1) do
    wait_click(1)
    if is_pressed "remote" then shoot() end
end
```

[1] The ALT key is one of the buttons at the back of your camera. After pressing the correct button, you will see ALT on the bottom line of the display. At that point most buttons will have an alternate function.

127.5. Sample Multiple trigger events

By sending pulses ranging in length from 10 to 180 ms and a gap in between these pulses of at least 55ms, you can trigger different events.
You'll need to make some settings in the CHDK menu of the camera, so the LUA script can recognize the different levels.

- CHDK SETTINGS
  - REMOTE PARAMETERS
    - ENABLE REMOTE to ON
    - SWITCH TYPE to CA-1

These pulses can be measured in a CHDK LUA-script with the following command:

```
a = get_usb_power
```

The result is 1/10th of the pulse length, so ranging pulses from 10 to 180ms, the function **`get_usb_power`** will range from 1 to 18. Unfortunately this is not very accurate. A minimum of 6 different evens can be distinguished without mistakes.

Pulse length	`get_usb_power` minimum value	`get_usb_power` middle value	`get_usb_power` maximum value
20 ms	1	2	3
50 ms	4	5	6
80 ms	7	8	9
110 ms	10	11	12
140 ms	13	14	15
170 ms	16	17	18

Arduino sketch Multiple trigger events

The following Arduino sketch and CHDK Lua script controls zoom level and shutter of the camera. The Arduino sketch sends 6 different pulses to the Canon camera with 10 seconds in between. The corresponding LUA script will take these levels to zoom in 0, 20, 40, 60, 80 or 100% and then take a picture.

Sample connections

- Connect the red wire with D9.
- Connect the black wire with GND
- Connect the USB connector to a Canon Camera with the CHDK firmware
- After uploading the following sketch, continue with the camera preperations on the next page.

072_CameraCanonCHDKMulti.ino

```
int LED = 13;
int USB_Power_Out = 9;

void setup() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  pinMode(USB_Power_Out, OUTPUT);
  digitalWrite(USB_Power_Out, LOW);
  Serial.begin(9600);
  Serial.println("Camera Controller Started");
}

void pulse(int duration) {
  digitalWrite(LED, HIGH);
  Serial.print("Send a ");
  Serial.print(duration);
  Serial.println(" ms pulse");
  digitalWrite(USB_Power_Out, HIGH);
  delay(duration);
  digitalWrite(USB_Power_Out, LOW);
  delay(55);
  digitalWrite(LED, LOW);
  delay(10000);
}

void loop() {
  pulse(20);
  pulse(50);
  pulse(80);
  pulse(110);
  pulse(140);
  pulse(170);
}
```

Camera preperations

- Install the correct firmware for your camera on an SD card
- Copy the CHDK_002_CameraCanonMultiple.lua on the SD card in the \CHDK\SCRIPTS folder (PHOTOS partition)
- Set the READ/WRITE-switch of the SD-card to READ (this will trigger the CHDK firmware)
- Place the SD-card in your camera.
- Turn on the camera
- Press the ALT[1]-key, MENU, CHDK Settings, Remote Parameters
    - Enable Remote: ON
    - Switch Type: CA-1
    - Control Type: None
- Press ALT-SET
    - Load Script from File ….
    - Select the CHDK_002_CameraCanonMultiple.lua script from the folder /CHDK/SCRIPTS and press the SET button
    - Select BACK.
- To start the script you need to fully press the shutter button.

CHDK_002_CameraCanonMultiple.lua

```
--[[
@title Multi Trigger
@chdk_version 1.4
--]]
function triggerA()
     print("20 ms pulse")
     set_zoom(0)
     shoot()
end

function triggerB()
     print("50 ms pulse")
     set_zoom(20)
     shoot()
end

function triggerC()
     print("80 ms pulse")
     set_zoom(40)
     shoot()
end

function triggerD()
     print("110 ms pulse")
     set_zoom(60)
     shoot()
end

function triggerE()
     print("140 ms pulse")
     set_zoom(80)
     shoot()
```

[1] The ALT key is one of the buttons at the back of your camera. After pressing the correct button, you will see ALT on the bottom line of the display. At that point most buttons will have an alternate function.

```
end

function triggerF()
     print("170 ms pulse")
     set_zoom(100)
     shoot()
end


print "Start Multi trigger levels by Arduino"
while (1) do
     repeat
        a = get_usb_power(0)
    until a>0
    if (a >= 1 and  a <= 3) then triggerA() end
    if (a >= 4 and  a <= 6) then triggerB() end
    if (a >= 7 and  a <= 9) then triggerC() end
    if (a >= 10 and a <= 12) then triggerD() end
    if (a >= 13 and a <= 15) then triggerE() end
    if (a >= 16 and a <= 18) then triggerF() end
    if (a > 19) then print "error" end
end
```

Wired communication

In this section you will find several techniques used for wired communication between Arduino <=> Arduino, Arduino <=> Computer and between Arduino <=> other microcontrollers.

128. SoftEasyTransfer communication between 2 Arduino's

Bill Porter wrote the EasyTransfer library to exchange binary data between 2 Arduino boards. At first this library was to be used only for communication between the standard Rx (D0) and Tx (D1), but now EasyTransfer is available for the following types of communication:

- EasyTransfer
  Serial communication between Rx (D0) and Tx (D1)
- SoftEasyTransfer
  Software Serial communication between other digital ports.
- EasyTransferVirtualWire
  Virtual Wire (experimental) communication through cheap radio's.
- EasyTransferI2C
  Communication through the I2C protocol by using SDA (A4) and SCL (A5).

This chapter only describes the SoftEasyTransfer!

128.1. Specifications for SoftEasyTransfer communication between 2 Arduino's

- User definable data structure for communication.
- Checksums.

128.2. Datasheet SoftEasyTransfer communication between 2 Arduino's

More information can be found at Bill Porter's website called "The mind of Bill Porter": http://www.billporter.info/2011/05/30/easytransfer-arduino-library/.

128.3. Libraries needed for SoftEasyTransfer communication between 2 Arduino's

- EasyTransfer libraries from Bill Porter
  https://github.com/madsci1016/Arduino-EasyTransfer
- SoftwareSerial library included with Arduino IDE.

Library use explanation

```
#include <SoftEasyTransfer.h>
```

Include the SoftEasyTransfer library from Bill Porter.

```
#include <SoftwareSerial.h>
```

Include the SoftwareSerial library included in Arduino IDE.

```
SoftwareSerial mySerial(2, 3);
```

Create 'mySerial' a new instance of the object type SoftwareSerial. 10 is an Integer value corresponding to the digital port used for SoftwareSerial RX. 11 is an Integer value corresponding to the digital port used for SoftwareSerial TX. These can be any digital ports.

```
SoftEasyTransfer ET;
```

Create 'ET' a new instance of the object SoftEasyTransfer.

```
struct SEND_DATA_STRUCTURE
{
  char text[32];
};
```

This data structure needs to be the same on both transmitter and receiver.

```
SEND_DATA_STRUCTURE mydata;
```

Define SEND_DATA_STRUCTURE variable called 'mydata'.

```
mySerial.begin(9600);
```
Initialize mySerial.

```
ET.begin(details(mydata), &mySerial);
```
Initialize the communication using mydata as the transmitting/receiving data structure and use mySerial as connection.

```
String string="Hello World";
string.toCharArray(mydata.text,12);
ET.sendData();
```
Fill a string and convert this to a Char Array (as that is a part of the data structure). Send the data structure to the other Arduino.

```
  if (ET.receiveData())
  {
    Serial.println(mydata.text);
  }
```
If data was received, print this on the Serial Monitor.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

128.4. Sample SoftEasyTransfer communication between 2 Arduino's

The following sketch sends the text "Hello World" to another Arduino.

Sample Connections

- Connect Receiver GND to Transmitter GND.
- Connect Receiver D2 to Transmitter D3.
- Connect Receiver D3 to Transmitter D2.
- Connect Receiver USB to computer for the Serial Monitor.
- Connect Transmitter to a Power Source.

073_SoftEasyTransfer_Transmit.ino

```
#include <SoftEasyTransfer.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
SoftEasyTransfer ET;

struct SEND_DATA_STRUCTURE{
  char text[32];
};

SEND_DATA_STRUCTURE mydata;

void setup()
{
  mySerial.begin(9600);
  ET.begin(details(mydata), &mySerial);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}

void loop()
{
  String string="Hello World";
  string.toCharArray(mydata.text,12);
  ET.sendData();
  delay(2000);
}
```

074_SoftEasyTransfer_Receive.ino

```
#include <SoftEasyTransfer.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);
SoftEasyTransfer ET;

struct RECEIVE_DATA_STRUCTURE
{
  char text[32];
};

RECEIVE_DATA_STRUCTURE mydata;

void setup()
{
  mySerial.begin(9600);
  ET.begin(details(mydata), &mySerial);
  Serial.begin(9600);
  pinMode(13, OUTPUT);
}
```

```
void loop()
{
  if (ET.receiveData())
  {
    Serial.println(mydata.text);
  }
  delay(250);
}
```

Wireless communication

In this section you will find several components used for wireless communication, using Infrared, Radio Frequencies and Bluetooth.

129. IR sensor (receive)

1338B


TSOP38238

129.1. Specifications IR Sensor (receive)

- 1338B IR sensor
- TSOP38238

Background information can be found at the following URL's:

- https://learn.sparkfun.com/tutorials/ir-communication/all
- http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

129.2. Connections IR Sensor (receive)

Pin nr	Name	Description	Arduino pin
1	OUT	Signal	Any PWM port
2	GND	Ground	GND
3	VCC	5 V	5 V

129.3. Libraries needed for IR Sensor (receive)

- IRremote library from Ken Shirriff through the Library Manager.

Library use explanation

```
#include <IRremote.h>
```

Include the Infra-Red remote library from Ken Shirriff.

```
IRrecv myirrecv(RECV_PIN);
```

Create myirrecv a new instance of the object type IRrecv. RECV_PIN is an Integer value corresponding to the Arduino PWM port to which the Signal pin is connected.

```
decode_results myresults;
```

Create myresult of the 'decode_results' type. The data received by the IR sensor will be stored in myresults.

```
myirrecv.enableIRIn(); // Start the receiver
```

Start the receiver.

```
if (myirrecv.decode(&myresults))
{
    Serial.println(myresults.value, HEX);
    myirrecv.resume(); // Receive the next value
}
```

Receive the first value and temporary disable the receiving of new values. The method .decode needs to store the result in myresults, so you use a "Call by Reference", by placing the & sign in front of myresults. So .decode is giving the location where myresults is stored, instead of the value. As a result the decoded value is stored in myresults. If a value has been received, then myresults.value will be printed to the serial port and the receiving of new values will be resumed.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

129.4. Sample IR sensor

This sample sketch shows the code send by a compatible IR Remote Control.

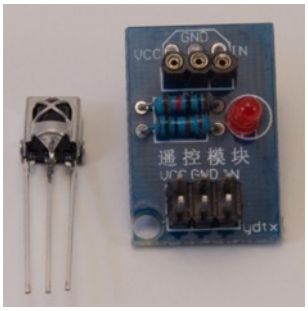
Sample Connections

- Connect Signal with D11.
- Connect GND with GND.
- Connect VCC with 5V.

075_IR_receive_universal.ino

```
#include <IRremote.h>

int RECV_PIN = 11;

IRrecv myirrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  myirrecv.enableIRIn(); // Start the receiver
}
void loop() {
  if (myirrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    myirrecv.resume(); // Receive the next value
  }
}
```

130. IR sensor board (receive)



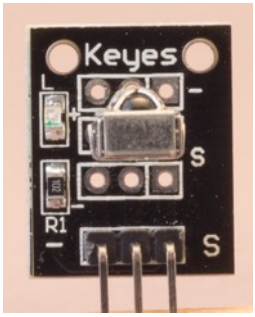
This is a board with the same 1338B that can be connected through a 3 pin connector. A red LED shows that the module is connected to VCC and GND

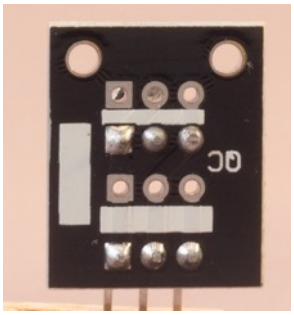
130.1. Connections IR Sensor Board (receive)

Compared to the onboard 1338B IR receiver the pins VCC and GROUND are swapped.

Pin nr	Name	Description	Arduino pin
1	OUT	Signal	Any PWM port
2	GND	Ground	GND
3	VCC	5 V	5 V

131. Keyes IR sensor board (receive)





This is a board with the same 1338B soldered on it as is described in the chapter before this chapter, the only difference is that the pin assignment is different.

131.1. Connections IR Sensor Board (receive)

Compared to the onboard 1338B IR receiver the pins VCC and GROUND are swapped.

Pin nr	Name	Description	Arduino pin
1	-	Ground	GND
2	+	5 V	5 V
3	S	Signal	Any PWM port

132. Various IR remote controls

132.1. IR remote control 'Car'

CH-	CH	CH+
<<	>>	>
-	+	EQ
0	100+	200+
1	2	3
4	5	6
7	8	9

This is a simple Infrared remote control.

132.2. Specifications IR Remote Control 'Car'

- NEC encoding 32 bits.
- Repeating of a button is encoded with FFFFFF until the button is released, or until another key is pressed.

Codes

Button	HEX code	DEC code
CH-	FFA25D	16753245
CH	FF629D	16736925
CH+	FFE21D	16769565
<<	FF22DD	16720605
>>	FF02FD	16712445
>	FFC23D	16761405
-	FFE01F	16769055
+	FFA857	16754775
EQ	FF906F	16748655
0	FF6897	16738455
100+	FF9867	16750695
200+	FFB04F	16756815
1	FF30CF	16724175
2	FF18E7	16718055
3	FF7A85	16743045
4	FF10EF	16716015
5	FF38C7	16726215
6	FF5AA5	16734885
7	FF42BD	16728765
8	FF4AB5	16730805
9	FF52AD	16732845

```
String Button_name[21] =
{
     "CHmin", "CH",
"CHplus",
     "Rwd",   "FF",
"Play",
     "Min",   "Plus",
"EQ",
     "Zero",  "OneHundredPlus",
"TwoHundredPlus",
     "One",   "Two",
"Three",
     "Four",  "Five",
"Six",
     "Seven", "Eight",
"Nine"
};

long    Button_DEC_code[21] =
{
     16753245, 16736925, 16769565,
     16720605, 16712445, 16761405,
     16769055, 16754775, 16748655,
     16738455, 16750695, 16756815,
     16724175, 16718055, 16743045,
     16716015, 16726215, 16734885,
     16728765, 16730805, 16732845
};
```

132.3. Sample IR remote control 'Car'

The following sample prints the name of the pressed button to the serial port.

Sample Connections

For connections see "129.2 Connections IR Sensor (receive)".

076_IR_Receive_Remote_Car.ino

```
#define BAUDRATE 9600
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
int nobuttons=21;
```

.....

Insert the codes for this specific IR remote control here!

.....

```
void setup()
{
  Serial.begin(BAUDRATE);
  irrecv.enableIRIn(); // Start the receiver
}

void loop()
{
  if (irrecv.decode(&results))
  {
    int count = results.rawlen;
    int teller =0;
    boolean notfound = true;
    if (results.decode_type == NEC)
    {
      while (teller< nobuttons && notfound)
      {
        long button=results.value;
        if (button==Button_DEC_code[teller])
        {
          Serial.println(Button_name[teller]);
          notfound = false;
        }
        teller++;
      }
    }
    if (notfound)
    {
      String button=String(results.value, HEX);
      if (button.substring(0,6)!="ffffff")
      {
        button.toUpperCase();
        Serial.println("Unknown code:\n0x" + button + "\n" +
String(results.value) + "\n");
      }
    }
    irrecv.resume(); // Receive the next value
  }
}
```

132.4. IR remote control 'Keyes'

	Up	
Left	OK	Right
	Down	
1	2	3
4	5	6
7	8	9
*	0	#

This is a simple Infrared remote control.

Specifications IR Remote Control 'Keyes'

- NEC encoding 32 bits.
- Repeating of a button is encoded with FFFFFF until the button is released, or until another key is pressed.
- Powered by 1 CR 2025 battery.

077_IR_Receive_Remote_Keyes.ino

Button	HEX code	DEC code
Up	FF629D	16736925
Left	FF22DD	16720605
OK	FF02FD	16712445
Right	FFC23D	16761405
Down	FFA857	16754775
1	FF6897	16738455
2	FF9867	16750695
3	FFB04F	16756815
4	FF30CF	16724175
5	FF18E7	16718055
6	FF7A85	16743045
7	FF10EF	16716015
8	FF38C7	16726215
9	FF5AA5	16734885
*	FF42BD	16728765
0	FF4AB5	16730805
#	FF52AD	16732845

```
String Button_name[17] =
{
                "Up",
     "Left",    "OK",     "Right",
                "Down",
     "One",     "Two",    "Three",
     "Four",    "Five",   "Six",
     "Seven",   "Eight",  "Nine",
     "*",       "Zero",   "Hash"
};

long     Button_DEC_code[17] =
{
                 16736925,
     16720605,  16712445, 16761405,
                 16754775,
     16738455,  16750695, 16756815,
     16724175,  16718055, 16743045,
     16716015,  16726215, 16734885,
     16728765,  16730805, 16732845
};
```

132.5. IR Remote Control 3

Power	Vol+	Func/Stop
<<	> ||	>>
Down	Vol-	Up
0	EQ	St/Rept
1	2	3
4	5	6
7	8	9

This is a simple Infrared remote control.

Specifications IR Remote Control 3

- NEC encoding 32 bits.
- Repeating of a button is encoded with FFFFFF until the button is released, or until another key is pressed.
- Powered by 1 CR 2025 battery.

078_IR_Receive_Remote_Control3.ino

Button	HEX code	DEC code
Power	FD00FF	16580863
VOL+	FD807F	16613503
FUNC/STOP	FD40BF	16597183
<<	FD20DF	16589023
>	FDA05F	16621663
>>	FD609F	16605343
Down	FD10EF	16584943
VOL-	FD906F	16617583
Up	FD50AF	16601263
0	FD30CF	16593103
EQ	FDB04F	16625743
ST/REPT	FD708F	16609423
1	FD08F7	16582903
2	FD8877	16615543
3	FD48B7	16599223
4	FD28D7	16591063
5	FDA857	16623703
6	FD6897	16607383
7	FD18E7	16586983
8	FD9867	16619623
9	FD58A7	16603303

```
String Button_name[21] =
{
     "Power", "Vol+", "Func/Stop",
     "Rwd", "Play", "FF",
     "Down", "Vol-", "Up",
     "Zero", "EQ", "St/Rept",
     "One", "Two", "Three",
     "Four", "Five", "Six",
     "Seven", "Eight", "Nine"
};

long    Button_DEC_code[21] =
{
     16580863, 16613503, 16597183,
     16589023, 16621663, 16605343,
     16584943, 16617583, 16601263,
     16593103, 16625743, 16609423,
     16582903, 16615543, 16599223,
     16591063, 16623703, 16607383,
     16586983, 16619623, 16603303
};
```

132.6. IR Remote Control 4

Power	Vol+	Func/Stop
<<	> ||	>>
Down	Vol-	Up
0	EQ	St/Rept
1	2	3
4	5	6
7	8	9

This is a simple Infrared remote control.

Specifications IR Remote Control 4

- NEC encoding 32 bits.
- Repeating of a button is encoded with FFFFFF until the button is released, or until another key is pressed.
- Powered by 1 CR 2025 battery.

079_IR_Receive_Remote_Control4.ino Codes

Button	HEX code	DEC code
Power	FFA25D	16753245
VOL+	FF629D	16736925
FUNC/STOP	FFE21D	16769565
<<	FF22DD	16720605
>	FF02FD	16712445
>>	FFC23D	16761405
Down	FFE01F	16769055
VOL-	FFA857	16754775
Up	FF906F	16748655
0	FF6897	16738455
EQ	FF9867	16750695
ST/REPT	FFB04F	16756815
1	FF30CF	16724175
2	FF18E7	16718055
3	FF7A85	16743045
4	FF10EF	16716015
5	FF38C7	16726215
6	FF5AA5	16734885
7	FF42BD	16728765
8	FF4AB5	16730805
9	FF52AD	16732845

```
String Button_name[21] =
{
     "Power", "Vol+", "Func/Stop",
     "Rwd", "Play", "FF",
     "Down", "Vol-", "Up",
     "Zero", "EQ", "St/Rept",
     "One", "Two", "Three",
     "Four", "Five", "Six",
     "Seven", "Eight", "Nine"
};

long    Button_DEC_code[21] =
{
     16753245, 16736925, 16769565,
     16720605, 16712445, 16761405,
     16769055, 16754775, 16748655,
     16738455, 16750695, 16756815,
     16724175, 16718055, 16743045,
     16716015, 16726215, 16734885,
     16728765, 16730805, 1632845
};
```

132.7. IR remote Apple White

	+	
|<<	> ||	>>|
	-	
	MENU	

This is an old remote control for an Apple iMac. Every remote has its own codes.

132.8. Specifications IR Remote Control 'Car'

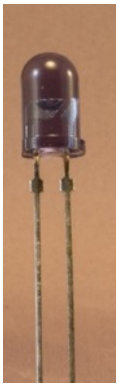
- NEC encoding 32 bits.
- Repeating of a button is encoded with FFFFFF until the button is released, or until another key is pressed.

057_IR_Receive_Remote_AppleWhite

Button	HEX code	DEC code
+	77E1D0D8	2011287768
|<<	77E110D8	2011238616
> ||	77E120D8	2011242712
>>|	77E1E0D8	2011291864
-	77E1B0D8	2011279576
MENU	77E140D8	2011250904

```
int nobuttons = 6;

String Button_name[21] =
{
                "Vol+",
  "Previous",   "Play/pause",
"Next",
                "Vol-",

                "Menu"
};

long    Button_DEC_code[21] =
{
              2011287768,
  2011238616, 2011242712,
2011291864,
              2011279576,

              2011250904
};
```

133. IR LED (send)



133.1. Specifications IR LED (send)

133.2. Connections IR LED (send)

Pin nr	Name	Description	Arduino pin
1	Cathode		GND
2	Anode		Us a 330 ohm resistor to connect it to the pin with the default timer. On an UNO this in TIMER 2 on Pin3, on a MEGA this is also TIMER2, but on pin 9.

133.3. Libraries needed for IR LED (send)

- Infrared remote library from??

```
#include <IRremote.h>
```

??

```
IRsend irsend; //Connect IR LED to D3!!
```

??

```
irsend.sendNEC(16720605, 32);
```

??

133.4. Sample IR LED (send)

This sketch uses a joystick to determine which IR code is going to be send to another Arduino!

Sample Connections

- Connect IR_LED Cathode with GND
- Connect a 330 Ohm resistor between IR_LED Anode and D3 (TIMER2)
- Connect Joystick GND to GND
- Connect Joystick +5V to 5V
- Connect Joystick VRx to A0
- Connect Joystick VRy to A1

080_IR_Transmit.ino

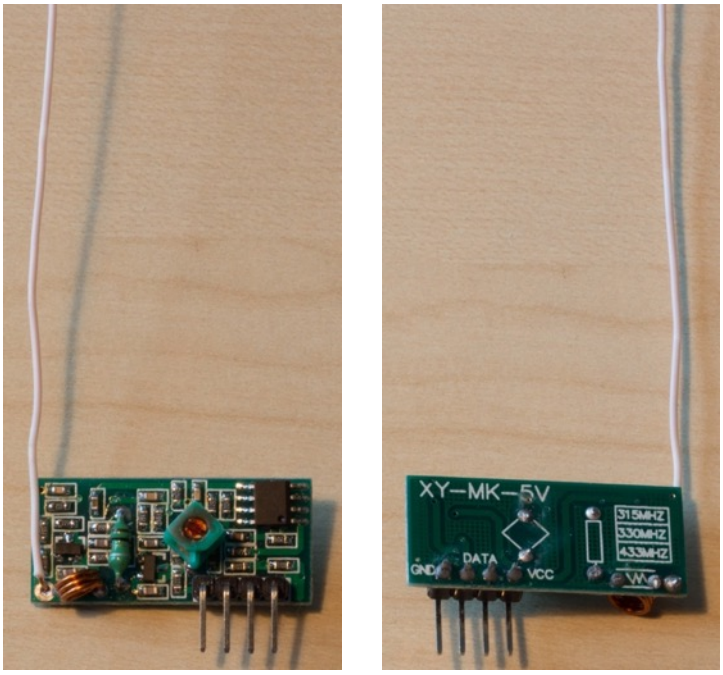
```
#include <IRremote.h>

IRsend irsend; //Connect IR LED to D3!!

long    Button_DEC_code[4] =
{
  16736925, 16720605, 16761405, 16754775
};

void setup()
{
  Serial.begin(9600);
}

void loop()
{
    int sensorx=map(analogRead(A0), 0, 1023, -100, 100);
    if (sensorx>=-5 && sensorx<=5)
    {
      sensorx=0;
    }
    int sensory=map(analogRead(A1), 0, 1023, -100, 100);
    if (sensory>=-5 && sensory<=5)
    {
      sensory=0;
    }
    Serial.print(sensorx, DEC);
    Serial.print(", ");
    Serial.println(sensory, DEC);
    if (sensorx>0) irsend.sendNEC(Button_DEC_code[2], 32); //UP
    if (sensorx<0) irsend.sendNEC(Button_DEC_code[1], 32); //DOWN
    if (sensory>0) irsend.sendNEC(Button_DEC_code[0], 32); //RIGHT
    if (sensory<0) irsend.sendNEC(Button_DEC_code[3], 32); //LEFT
    delay(100);
}
```

138_IR_Receive_Arduino.ino

Button	HEX code	DEC code
RIGHT	FF629D	16736925
DOWN	FF22DD	16720605
UP	FFC23D	16761405
LEFT	FFA857	16754775

```
String Button_name[4] = {
  "RIGHT", "DOWN", "UP", "LEFT"};

long    Button_DEC_code[4] = {
  16736925, 16720605, 16761405,
16754775};
```

134. 315/330/433 MHz RF Receiver XY-MK-5V



Cheap 315/330/433 MHz RF receiver.

134.1. Specifications 315/330/433 RF Receiver XY-MK-5V

- Voltage: 5V
- locked at 433.92 MHz

134.2. Datasheet 315/330/433 MHz RF Receiver XY-MK-5V

- http://www.567.dk/datasheet/fs1000a.zip
- http://www.electrodragon.com/w/index.php?title=433RF_module

134.3. Connections 315/330/433 RF Receiver XY-MK-5V

Pin nr	Name	Description	Arduino pin
1	VCC	VCC	5V
2	DATA	DATA	Any digital port
3	DATA	DATA	Not needed, same as pin 2
4	GND	Ground	GND
Hole at the lower corner opposite to the pins.			17 cm solid wire (433 MHz)

134.4. Libraries needed for 315/330/433 MHz RF Receiver XY-MK-5V

- Virtual Wire library from Mike McCauley[1] http://www.pjrc.com/teensy/arduino_libraries/VirtualWire.zip, more information can be found at: http://www.open.com.au/mikem/arduino/VirtualWire.pdf.

[1] Another library you could use is the RCSwitch library from Suat: https://github.com/sui77/rc-switch.

Library use explanation

```
#include <VirtualWire.h>
```

Include the Virtual Wire library from Mike McCauley.

```
vw_set_rx_pin(11);
```

Set the receive pin to D11.

```
vw_setup(2000);
```

Set the receive speed to 2000 bps.

```
vw_rx_start();
```

Start the receiver.

```
if (vw_get_message(message, &messageLength))
for (int i = 0; i < messageLength; i++)
{
   Serial.write(message[i]);
}
```

Loop to display all characters in the messagelength.

134.5. Sample 315/330/433 RF Receiver XY-MK-5V

The following sketch displays the received text string from the transmitter. Check the next chapter for the transmitter and the corresponding sample sketch.

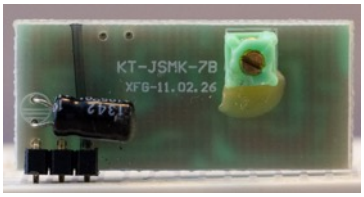
Sample Connections

- Solder 17 cm solid wire (for example 1 thread of UTP) to the hole at the lower corner opposite to the pins.
- Connect VCC to 5V.
- Connect one of the data pins to D11.
- Connect GND to GND.

081_RF433_Receive_XY-MK-5V.ino

```
//Receiver
#include <VirtualWire.h>

byte message[VW_MAX_MESSAGE_LEN];
byte messageLength = VW_MAX_MESSAGE_LEN;

void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  Serial.println("Device is ready");
  vw_set_rx_pin(11);
  vw_setup(2000);
  vw_rx_start();
}

void loop()
{
  if (vw_get_message(message, &messageLength))
  {
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    Serial.print("Received: ");
    for (int i = 0; i < messageLength; i++)
    {
      Serial.write(message[i]);
    }
    Serial.println();
  }
}
```

135. 433 MHz RF Receiver KT-JSMK-7B



I salvaged this receiver module from a Flaming HA500S wireless socket.

135.1. Specifications 433 MHz RF Receiver KT-JSMK-7B

- Voltage: 5V
- Locked at 433.92 MHz

135.2. Datasheet 433 MHz RF Receiver KT-JSMK-7B

I was not able to find the corresponding datasheet.

135.3. Connections 433 MHz RF Receiver KT-JSMK-7B

Pin nr	Name	Description	Arduino pin
1	VCC	Power supply	5V
2	OUT	Data in (Rx)	IRQ0 (D2) or IRQ1 (D3)
3	GND	Ground	GND

I'm not sure were to solder an antenna, but the module performed well enough without it.

135.4. Libraries needed for 433 MHz RF Receiver KT-JSMK-7B

- Virtual Wire library from Mike McCauley[1] http://www.pjrc.com/teensy/arduino_libraries/VirtualWire.zip, more information can be found at: http://www.open.com.au/mikem/arduino/VirtualWire.pdf.

Library use explanation

```
#include <VirtualWire.h>
```

Include the Virtual Wire library from Mike McCauley.

```
vw_set_rx_pin(11);
```

Set the receive pin to D11.

```
vw_setup(2000);
```

Set the receive speed to 2000 bps.

```
vw_rx_start();
```

Start the receiver.

```
if (vw_get_message(message, &messageLength))
for (int i = 0; i < messageLength; i++)
{
   Serial.write(message[i]);
}
```

Loop to display all characters in the messagelength.

[1] Another library you could use is the RCSwitch library from Suat: https://github.com/sui77/rc-switch.

135.5. Sample 433 MHz RF Receiver KT-JSMK-7B

The following sketch (the same as for the XY-MK-5V transceiver module) displays the received text string from the transmitter. Check the next chapter for the FS1000a transmitter and the corresponding sample sketch.

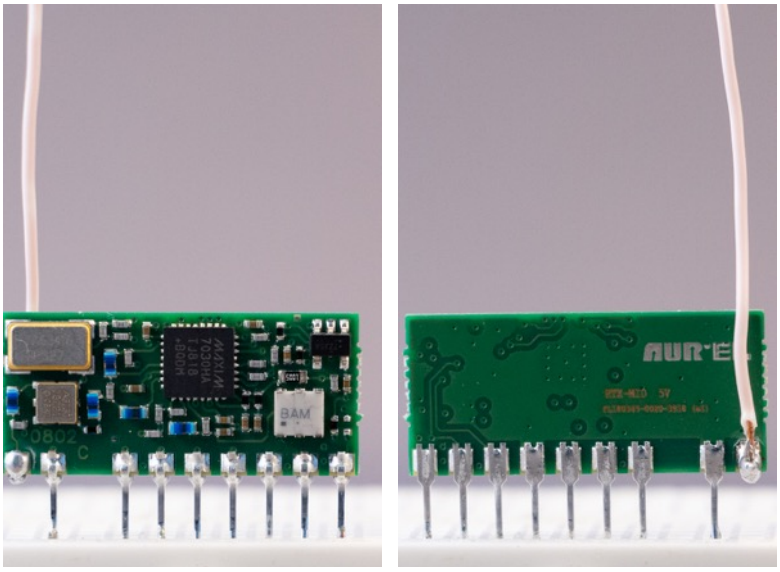
Sample Connections

- Connect VCC to 5V.
- Connect OUT to D2.
- Connect GND to GND.

144_RF433_Receive_KT-JSMK-7B.ino

```
//Receiver
#include <VirtualWire.h>

byte message[VW_MAX_MESSAGE_LEN];
byte messageLength = VW_MAX_MESSAGE_LEN;

void setup()
{
  Serial.begin(9600);
  pinMode(13, OUTPUT);
  Serial.println("Device is ready");
  vw_set_rx_pin(11);
  vw_setup(2000);
  vw_rx_start();
}

void loop()
{
  if (vw_get_message(message, &messageLength))
  {
    digitalWrite(13, HIGH);
    delay(200);
    digitalWrite(13, LOW);
    Serial.print("Received: ");
    for (int i = 0; i < messageLength; i++)
    {
      Serial.write(message[i]);
    }
    Serial.println();
  }
}
```

136. 433 MHz RF transceiver Aurel RTX-MID-5V



I've added this module, because I've used it with RFLink software on an Arduino Mega2560 together with Domoticz.

136.1. Specifications 433 MHz RF transceiver Aurel RTX-MID-5V

- Single RF channel 433.92 Mhz ASK
- Low current consumption
- Low cost
- Max bit rate: 9600 bps
- Max Output Power: 10 mW
- Voltage supply 5V (or 3V for the Aurel RTX-MID-3V module)
- Shared antenna for Rx and Tx, switched based on pin 5
    - PWRDN -> Rx 480 microseconds
    - PWRDN -> Tx 420 microseconds
    - Tx -> Rx 260 microseconds
    - Rx -> Tx 400 microseconds

136.2. Datasheet 433 MHz RF transceiver Aurel RTX-MID-5V

The datasheet for this transceiver can be found at:

- https://www.aurelwireless.com/wp-content/uploads/user-manual/650201033G_um.pdf

136.3. Connections 433 MHz RF transceiver Aurel RTX-MID-5V

Pin nr	Name	Description	Arduino pin
1	Antenna	Antenna connection 50 ohm, for both Tx (RF output) and Rx (RF input)	a 17 cm piece of solid wire
2	GND	Ground	GND
3	no pin	There is no pin at the 3rd position	Any digital port
4	Data In	Data input of Tx 1 = emission of carrier 0 = no emission	Any digital port
5	TX/RX	Switch between Rx and Tx 0 (or n.c.) = Receiver mode 1 = Transmit mode	Any digital port
6	Enable	Set power mode 0 = PWND powerdown, typical use: 1uA 1 = Active (either Receive or Transmit mode depending on pin 5)	Any digital port
7	GND	Ground	GND
8	Analog Out	Analog output for testing purposes	to GND through a 20K resistor
9	Data Out	Digital data output of receiver	Any digital port ??
10	Vcc	5V power supply	5V (also connect a 100nF to GND)

136.4. Sample 433 MHz RF transceiver Aurel RTX-MID-5V (Transmit)

The following sketch sends a text string. Check "135 433 MHz RF Receiver KT-JSMK-7B" for the receiver and the corresponding receiver sketch.

Sample Connections

- Connect 2 GND to GND10.
- Connect 4 to D10.
- Connect 5 TX/RX to VCC (set Tx)
- Connect 6 Enable to VCC Enable power)
- Connect 10 to 5V.

082_RF433_Transmit_FS1000A.ino (this sketch also works for the Aurel RTX-MID-5V)

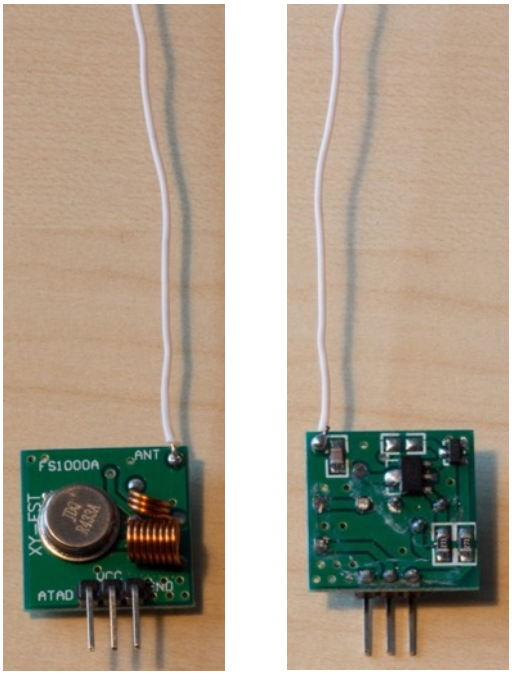
```
//Transmitter
#include <VirtualWire.h>

void setup()
{
  vw_set_tx_pin(10);
  vw_setup(2000);
  pinMode(13, OUTPUT);
}

void loop()
{
  send("Is there anybody out there?");
  delay(1000);
}

void send (char *message)
```

```
{
  vw_send((uint8_t *)message, strlen(message));
  digitalWrite(13, HIGH);
  vw_wait_tx();
  digitalWrite(13, LOW);
}
```

137. 433 MHz RF Wireless Socket Flamingo FA500S

I got this Wireless socket from a nephew, but he lost the remote control. I've tried various protocols (first wiping the pairing details by sending OFF-signals during setup mode and than ON signals to pair it with a virtual switch), but I didn't manage to get the socket paired. So I decided to take it apart and I removed the receiver module a KT-JSMK-7B. I did manage to get the receiver working on a Arduino. So I destroyed a € 5,- wireless socket to salvage a € 0,80 receiver module.

138. 433 MHz RF Transmitter FS1000A



Cheap 433 MHz RF transmitter.

138.1. Specifications 433 MHz RF Transmitter FS1000A

- Voltage: 3-12V
- 433,92 MHz
- > 500M (??)
- < 10Kbps
- Antenna 17 cm, 50 ohm single conductor (for example 1 thread of UTP)

138.2. Datasheet 433 MHz RF Transmitter FS1000A

- http://www.567.dk/datasheet/fs1000a.zip
- http://www.electrodragon.com/w/index.php?title=433RF_module

138.3. Connections 433 MHz transmitter FS1000A

Pin nr	Name	Description	Arduino pin
1	ATAD	DATA	Any Digital port
2	VCC	VCC	5V
3	GND	Ground	GND
ANT	ANT	Antenna	17 cm solid wire

138.4. Libraries needed for 433 MHz RF Transmitter FS1000A

- Virtual Wire library from Mike McCauley[1] http://www.pjrc.com/teensy/arduino_libraries/VirtualWire.zip, more information can be found at: http://www.open.com.au/mikem/arduino/VirtualWire.pdf.

[1] Another library you could use is the RCSwitch library from Suat: https://github.com/sui77/rc-switch.

Library use explanation

```
#include <VirtualWire.h>
```
Include the Virtual Wire library from Mike McCauley.

```
vw_setup(2000);
```
Set the receive speed to 2000 bps.

```
send("Hello there");
```
Calls the procedure send, declared further down.

```
void send (char *message)
```
Declaration

```
vw_send((uint8_t *)message, strlen(message));
```
Send a message.

```
vw_wait_tx();
```
*Wait until
the message has left the transmitter.*

138.5. Sample 433 MHz RF Transmitter FS1000A

Check “135 433 MHz RF Receiver KT-JSMK-7B” for the receiver and the corresponding receiver sketch.

Sample Connections FS1000A

- Connect ATAD to D10.
- Connect VCC to 5V.
- Connect GND to GND.

082_RF433_Transmit_FS1000A.ino

```
//Transmitter
#include <VirtualWire.h>

void setup()
{
  vw_set_tx_pin(10);
  vw_setup(2000);
  pinMode(13, OUTPUT);
}

void loop()
{
  send("Is there anybody out there?");
  delay(1000);
}

void send (char *message)
{
  vw_send((uint8_t *)message, strlen(message));
  digitalWrite(13, HIGH);
  vw_wait_tx();
  digitalWrite(13, LOW);
}
```

139. Silvercrest Wireless Socket set

This Wireless socket set from Silvercrest (284705) was bought from Lidl a German company than can be found in lots of countries in Europe.

139.1. Specifications Silvercrest Wireless Socket set

Sockets RCR DP3 3711-A:

- Rated voltage: 230 V, 50 Hz
- Breaking capacity: 16 A, 3.680 W
- Standby power: < 0.7 W
- Radio frequency: 433,92 MHz
- Each socket can be synced with a maximum of 6 transmitters

Handheld transmitter RCT DS1 CR-A 3725:

- Frequency band: 433,05 – 434,79 MHz
- Radio frequency: 433,92 MHz
- Max. transmitting power: 5 dBm
- Batteries: CR2032, lithium 3 V
- HandheldRange transmitter: 40 m max

Arduino based transmitter:

- I was able to sync the following transmitter through Arduino:
  - 433 MHz RF Transmitter FS1000A (previous chapter)

139.2. Datasheet Silvercrest Wireless Socket set

- Manual from an identical set: https://www.dvw-service.com/index.php?module=explorer&displayAction=download&downloadFile=products/de-DE/pdf/bedienungsanleitung_1048510706_0486621_ba.pdf
- https://forum.arduino.cc/index.php?topic=202556.15
- https://github.com/pimatic/pimatic/issues/386

139.3. Syncing transmitter with Silvercrest Wireless Socket set

You can sync up to 6 different transmitters to each socket. This paragraph describes how to sync or remove the syncing of transmitters.

Sync a new transmitter

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds

- Use the transmitter of your choice
  - Arduino
    Start a sketch that sends the ON signal of the channel that you want to sync with this socket.
  - Handheld transmitter
    Press the ON button of the channel that you want to sync with this socket.
- The status light will now stop blinking. You can now use the transmitter on the channel you have selected in the previous step.

Remove a synced transmitter

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds
- Use the transmitter of your choice
  - Arduino
    Start a sketch that sends the OFF signal of the channel that you want to remove from this socket.
  - Handheld transmitter
    Press the OFF button of the channel that you want to remove from this socket.
- The status light will now flash rapidly for approx. 2 seconds and then start to blink slowly for 30 seconds, so you can sync another transmitter.

Remove ALL synced transmitters

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds
- Use the transmitter of your choice
  - Arduino
    Start a sketch that sends the OFF signal of the Master channel.
  - Handheld transmitter
    Press the OFF button of the Master channel.
- The status light will now flash rapidly for approx. 2 seconds and then start to blink slowly for 30 seconds, so you can sync another transmitter.

139.4. Libraries needed for Silvercrest Wireless Socket set

No libraries are needed to switch these sockets, but you could use the sample sketch to create your own. I was to laze to record the original binary codes from the buttons of my own manual remote, so I copied those from someone else and synced those with my Wireless Sockets.

You'll need the whole sketch, but the only function you'll need to call is:

```
ActivatePlug(PLUG_A, on);
```

With this function call you switch PLUG_A to ON (You could also use PLUG_B, PLUG_C, PLUG_D and PLUG_MASTER).

```
ActivatePlug(PLUG_A, off);
```

With this function call you switch PLUG_A to OFF (You could also use PLUG_B, PLUG_C, PLUG_D and PLUG_MASTER).

139.5. Sample Silvercrest Wireless Socket set

The following sketch will alternate the status of Plug A. Source was modified from http://forum.arduino.cc/index.php?topic=202556.msg1492685#msg1492685 (modified by Poopi and keematic).

Sample Connections if you are using the FS1000A as the transmitter

433 MHz transmitter (like the 433 MHz RF Transmitter FS1000A)

- Connect ATAD to D10.
- Connect VCC to 5V.
- Connect GND to GND.

Sample Connections if you are using the Aurel RTX-MID-5V as the transmitter

- Connect 2 GND to GND10.
- Connect 4 to D10.
- Connect 5 TX/RX to VCC (set Tx)
- Connect 6 Enable to VCC Enable power)
- Connect 10 to 5

Make sure your that one of your Silvercrest Wireless Sockets is synced with your Arduino (by using the below sketch directly after plugging in the Silvercrest Wireless Socket).

083_RF433_Transmit_FS1000A_Silvercrest.ino

```
#define MAX_CODE_CYCLES      4
#define SHORT_DELAY          380
#define NORMAL_DELAY         500
#define SIGNAL_DELAY         1500
#define SYNC_DELAY           2650
#define EXTRASHORT_DELAY     3000
#define EXTRA_DELAY          10000
#define RF_DATA_PIN          10

const boolean on = true;
const boolean off = false;
unsigned char swap;

enum {
  PLUG_A = 0,
  PLUG_B = 1,
  PLUG_C = 2,
  PLUG_D = 3,
  PLUG_MASTER = 4,
};

unsigned long signals[5][2][MAX_CODE_CYCLES] = {
  { /*A*/
    { /*ON */
      0b111101000101011000101100, 0b111111110111000001111100,
      0b111100010110111010111100, 0b111111000001001010001100
    },
    { /*OFF*/
      0b111110001100010110101100, 0b111101101010101001101100,
      0b111110011101110100011100, 0b111111011001100101011100
    }
  },
  { /*B*/
    { /*ON */
      0b111101010010010011010101, 0b111100111011110010010101,
```

```
      0b11111010111001111000101, 0b11111011100010000100010
    },
    { /*OFF*/
      0b11111110001111111110101, 0b11110000111100110010101,
      0b11110010000010110000010, 0b11110111010000011100101
    }
  },
```

```
  { /*C*/
    { /*ON */
      0b111110011101110100011110, 0b111101101010101001101110,
      0b111110001100010110101110, 0b111111011001100101011110
    },
    { /*OFF*/
      0b111111110111000001111110, 0b111100010110111010111110,
      0b111111000001001010001110, 0b111101000101011000101110
    }
  },
  { /*D*/
    { /*ON */
      0b111100100000101100000111, 0b111101110100000111100111,
      0b111111100011111111110111, 0b111100001111001100110111
    },
    { /*OFF*/
      0b111101010010010011010111, 0b111100111011110010010111,
      0b111110101110011111000111, 0b111110111000100001000111
    }
  },
  { /*MASTER*/
    { /*ON */
      0b111101101010101001100010, 0b111110011101110100010010,
      0b111111011001100101010010, 0b111110001100010110100010
    },
    { /*OFF*/
      0b111111110111000001110010, 0b111101000101011000100010,
      0b111111000001001010000010, 0b111100010110111010110010
    }
  },
};

void setup() {
  pinMode(RF_DATA_PIN, OUTPUT);
  Serial.begin(9600);
  swap  = 0;
}

void sendSync()
{
  digitalWrite(RF_DATA_PIN, HIGH);
  delayMicroseconds(SHORT_DELAY);
  digitalWrite(RF_DATA_PIN, LOW);
  delayMicroseconds(SYNC_DELAY - SHORT_DELAY);
}

void sendValue(boolean value, unsigned int base_delay)
{
  unsigned long d = value ? SIGNAL_DELAY - base_delay : base_delay;
  digitalWrite(RF_DATA_PIN, HIGH);
  delayMicroseconds(d);
  digitalWrite(RF_DATA_PIN, LOW);
  delayMicroseconds(SIGNAL_DELAY - d);
}

void longSync()
{
  digitalWrite(RF_DATA_PIN, HIGH);
  delayMicroseconds(EXTRASHORT_DELAY);
  digitalWrite(RF_DATA_PIN, LOW);
  delayMicroseconds(EXTRA_DELAY - EXTRASHORT_DELAY);
}
```

```
void ActivatePlug(unsigned char PlugNo, boolean PlugStatus)
{
  digitalWrite(RF_DATA_PIN, LOW);
  delayMicroseconds(1000);
  unsigned long signal = signals[PlugNo][PlugStatus][swap];
  swap++;
  swap %= MAX_CODE_CYCLES;
  Serial.print(PlugNo == PLUG_MASTER ? 'M' : (char)('A' + PlugNo));
  Serial.print(" ");
  Serial.println(PlugStatus ? "On" : "Off");
  for (unsigned char i = 0; i < 4; i++)
  {
    sendSync();
    for (unsigned char k = 0; k < 24; k++)
    {
      sendValue(bitRead(signal, 23 - k), SHORT_DELAY);
    }
  }
  for (unsigned char i = 0; i < 4; i++)
  {
    longSync();
    for (unsigned char k = 0; k < 24; k++)
    {
      sendValue(bitRead(signal, 23 - k), NORMAL_DELAY);
    }
  }
}

void loop()
{
  ActivatePlug(PLUG_A, on);
  delay(500);
  ActivatePlug(PLUG_A, off);
  delay(5000);
}
```

140. 433 MHz RF Wireless Socket set Flamingo SF-500S/3

This Wireless socket set from Smartwares was bought from Action a company than can be found in lots of countries in Europe. This set should be compatible with all the Smartwares CoCo (Click-On/Click-Off) or as they are known in The Netherlands KaKu (Klik-Aan/Klik-Uit). Too bad this set is not compatible with my Silvercrest set in the previous chapter.

140.1. Specifications 433 MHz RF Wireless Socket set Flamingo SF-500S/3

Sockets SF-50P:

- Rated voltage: 230 V, 50 Hz
- Breaking capacity: 1.000 W
- Standby power: < 0.7 W
- Radio frequency: 433,92 MHz
- Each socket can be synced with a maximum of 5 transmitters

Handheld transmitter SF-500R:

- Frequency band: 433,05 – 434,79 MHz
- Radio frequency: 433,92 MHz
- Batteries: CR2032, lithium 3 V
- Handheld transmitter range: 30 m max

Arduino based transmitter:

- I was able to sync the following transmitter through Arduino:
  - 433 MHz RF Transmitter FS1000A (previous chapter)

140.2. Datasheet 433 MHz RF Wireless Socket set Flamingo SF-500S/3

- Manual:
  - http://service.smartwares.eu/Downloads.ashx?productId=10.043.85&filename=Manual_1004385_20170620.pdf&type=1
  - http://service.smartwares.eu/Downloads.ashx?productId=10.043.85&filename=Manual_1004385_20170925.pdf&type=1
  - http://service.smartwares.eu/Downloads.ashx?productId=10.043.85&filename=Manual_1004385_remote.pdf&type=1
  - http://service.smartwares.eu/Downloads.ashx?productId=10.043.85&filename=Manual_1004385_switch.pdf&type=1

140.3. Syncing transmitter with Flamingo SF-500/3 Wireless Socket set

You can sync up to 5 different transmitters to each socket. This paragraph describes how to sync or remove the syncing of transmitters.

Sync a new transmitter

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds
- Use the transmitter of your choice
    - Arduino
      Start a sketch that sends the ON signal of the channel that you want to sync with this socket.
    - Handheld transmitter
      Press the ON button of the channel that you want to sync with this socket.
- The status light will now stop blinking. You can now use the transmitter on the channel you have selected in the previous step.

Remove a synced transmitter

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds
- Use the transmitter of your choice
    - Arduino
      Start a sketch that sends the OFF signal of the channel that you want to remove from this socket.
    - Handheld transmitter
      Press the OFF button of the channel that you want to remove from this socket.
- The status light will now flash rapidly for approx. 2 seconds and then start to blink slowly for 30 seconds, so you can sync another transmitter.

Remove ALL synced transmitters

- Place Socket in power Outlet
- The status light will now slowly blink for 30 seconds
- Use the transmitter of your choice
    - Arduino
      Start a sketch that sends the OFF signal of the Master channel.
    - Handheld transmitter
      Press the OFF button of the Master channel.
- The status light will now flash rapidly for approx. 2 seconds and then start to blink slowly for 30 seconds, so you can sync another transmitter.

140.4. Libraries needed for 433 MHz RF Wireless Socket set Flamingo SF-500S/3

You'll need to download and install the NewRemoteSwitch library from the 433 MHz libraries-set from Randy Simons.
https://bitbucket.org/fuzzillogic/433mhzforarduino/get/latest_stable.zip

```
#include <NewRemoteTransmitter.h>
```

Include Randy Simons NewRemoteTransmitter library

```
NewRemoteTransmitter transmitter(ADDRESS, PIN, PERIOD);
```

Create transmitter, a new instance of the object NewRemoteTransmitter. In this example ADDRESS is the device address you give to your transmitter sketch, PIN is the Arduino digital pin, to which the transmitters is connected and 268 is the period that is used in the KaKu protocol of this set.

```
transmitter.sendUnit(1, true);
```
Switch 1st plug ON.

```
transmitter.sendUnit(1, false);
```
Switch 1st plug OFF.

```
transmitter.sendGroup(1,true);
```
Switch all plugs ON (or OFF).

140.5. Sample 433 MHz RF Wireless Socket set Flamingo SF-500S/3

The following sketch will alternate the status of the first.

Sample Connections if you are using the FS1000A as the transmitter

433 MHz transmitter (like the 433 MHz RF Transmitter FS1000A)

- Connect ATAD to D10.
- Connect VCC to 5V.
- Connect GND to GND.

Sample Connections if you are using the Aurel RTX-MID-5V as the transmitter

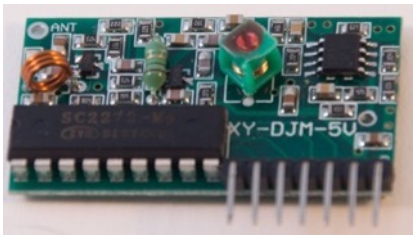
- Connect 2 GND to GND10.
- Connect 4 to D10.
- Connect 5 TX/RX to VCC (set Tx)
- Connect 6 Enable to VCC Enable power)
- Connect 10 to 5V

Make sure your that one of your Flamingo Wireless Sockets is synced with your Arduino (by using the below sketch directly after plugging in the Flamingo Wireless Socket).

084_RF433_Transmit_FS1000A_Flamingo.ino

```
#include <NewRemoteTransmitter.h>
NewRemoteTransmitter transmitter(12345678, 10, 268);

void setup()
{
}

void loop()
{
  transmitter.sendUnit(1, true);
  delay(1000);
  transmitter.sendUnit(1, false);
  delay(1000);
}
```

141. 433 MHz RF 4 channel Wireless kit XY-DJM-5V



141.1. Specifications 433 MHz RF 4 channel Wireless kit XY-DJM-5V

- Rx: SC2272-4M chip.
- Tx: SC2262 chip.

141.2. Datasheet 433 MHz RF 4 channel Wireless kit XY-DJM-5V

- http://www.electrodragon.com/w/2262_%26_2272_Wireless_Kits_(Module_and_Hand-Held_Controller)#Button_Action
- SC2272 chip: http://www.datasheet4u.net/download.php?id=643896
- SC2262 chip: http://www.sc-tech.cn/en/sc2262.pdf

141.3. Connections 433 MHz RF 4 channel Wireless kit XY-DJM-5V

Pin nr	Name	Description	Arduino pin
1	UT	Signal received	Any digital port (not needed)
2	D3	Button C	Any analog port
3	D2	Button A	Any analog port
4	D1	Button D	Any analog port
5	D0	Button B	Any analog port
6	5V	5V	5V
7	GND	Ground	GND

As you can see, the order of the pins D0..D3 is not the same as the buttons A..D!

141.4. Libraries needed for 433 MHz RF 4 channel Wireless kit XY-DJM-5V

None needed.

141.5. Sample 433 MHz RF 4 channel Wireless kit XY-DJM-5V

The following sketch shows which button is pressed and lights the onboard LED if either button is pressed.

Sample Connections

- Connect UT to D13.
- Connect D0 to A1.
- Connect D1 to A3.
- Connect D2 to A0.
- Connect D3 to A2.
- Connect VCC to 5V.
- Connect GND to GND.

085_RF433_Receive_XY-DJM-5V.ino

```
void setup()
{
  Serial.begin(9600);
}

void loop() {
  int ChannelA = map(analogRead(A0),0,1023,0,1);
  int ChannelB = map(analogRead(A1),0,1023,0,1);
  int ChannelC = map(analogRead(A2),0,1023,0,1);
  int ChannelD = map(analogRead(A3),0,1023,0,1);
  Serial.print(ChannelA, DEC);
  Serial.print(";");
  Serial.print(ChannelB, DEC);
  Serial.print(";");
  Serial.print(ChannelC, DEC);
  Serial.print(";");
  Serial.println(ChannelD, DEC);
}
```

142. NRF24L01 2.4GHZ Wireless Transceiver

This very cheap radio transceiver (both transmitter and receiver) can be used to setup wireless communication between two boards.

142.1. Specifications NRF24L01 2.4 GHz Wireless Transceiver

http://arduino-info.wikispaces.com/nRF24L01-RF24-Examples
http://www.mysensors.org/

- Worldwide 2.4GHz ISM band operation
- Up to 2 Mbps on air data rate
- Ultra-low power operation
- SPI protocol
- Distance open air 50m?

142.2. Datasheet NRF24L01 2.4 GHz Wireless Transceiver

- http://yourduino.com/docs/nRF24L01_Product_Specification_v2_0%20.pdf

142.3. Connections NRF24L01 2.4 GHz Wireless Transceiver

	GND	1	10	**GND**	
MISO	**MI**	2	9	**IRQ**	
	SCK	3	8	**MO**	*MOSI*
	CE	4	7	**CSN**	
3.3V	**VCC**	5	6	**VCC**	*3.3V*

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	MI	MISO	D12
3	SCK		D13
4	CE		Any digital port
5	VCC	VCC 3.3V	3.3V
6	VCC	VCC 3.3V	not needed
7	CSN		Any digital port
8	MO	MOSI	D11
9	IRQ	IRQ	(D2)
10	GND	Ground	not needed

142.4. Libraries needed for NRF24L01 2.4 GHz Wireless Transceiver

- RF24 library from TMRh through the Library Manager.

- SPI (Serial Peripheral Interface) library through the Library Manager.

Library use explanation NRF24L01 2.4 GHz Wireless Transceiver

```
#include <SPI.h>
```
Include the Serial Peripheral Interface included in the Arduino IDE.

```
#include "RF24.h"
```
Include the RF24 library for 2.4 GHz communication.

```
#include "printf.h"
```
This library needs to be copied from one of the example directories to the directory of the sketch. It is needed to format the output of radio.printdetails().

```
RF24 radio(9, 10);
```
Create radio, a new instance of the object RF24. In this example CD is connected to D9 and CSN is connected to D10.

```
byte address[][5] = { 0xCC, 0xCE, 0xCC, 0xCE, 0xCC , 0xCE, 0xCC, 0xCE,
0xCC, 0xCE};
```
Addresses used between Transmitter and Receiver.

```
void check_radio(void);
```
Check_radio is the name of the Interrupt.

```
printf_begin();
```
Start printf, so radio.printDetails can use it.

```
radio.begin();
```
Initialize the radio.

```
radio.enableAckPayload();
radio.enableDynamicPayloads();
```
Enable ACK (Acknowledgement) Payload feature.

```
radio.openWritingPipe(address[0]);
```
Open a pipe to the receiver that is listening on address[0].

```
radio.openReadingPipe(1, address[1]);
```
Open a pipe to the transmitter that is sending on address[1].

```
radio.printDetails();
```
Print the details of the radio.

```
attachInterrupt(0, check_radio, LOW);
```
Attach Interrupt handler check_radio to IRQ0 (IRQ is connected on D2). Both transmitter and receiver needs this.

```
radio.startWrite( &time, sizeof(unsigned long), 0 );
```
Sends the current clock tick.

```
radio.whatHappened(tx, fail, rx);
```
Tx becomes true after a successful transmission.
fail is true after a transmission has failed.
Rx is true after receiving a message.

```
radio.powerDown();
```
The radio can power down for now after a successful or failed transmission.

```
radio.read(&message_count, sizeof(message_count));
```

Read the message (receiver) or read the Acknowledgement &message_count (transmitter).

```
radio.startListening();
```

Set the radio to listening.

```
radio.writeAckPayload( 1, &message_count, sizeof(message_count) );
```

Send an acknowledgement to the transmitter.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders and at the following link: http://maniacbug.github.io/RF24/classRF24.html

142.5. Sample NRF24L01 2.4 GHz Wireless Transceiver

The following sketch sends some data between two Arduino's.

Sample Connections

Perform the following steps on 2 Arduino's with each a NFR24L01 2.4 GHz Wireless Transceiver.

- Connect GND to GND.
- Connect MI to D12 (MISO)
- Connect SCK to D13
- Connect CE to D9
- Connect VCC to 3.3 V
- Connect CSN to D10
- Connect MO to D11 (MOSI)
- Since this sketch is using interrupts, connect IRQ to D2 (=IRQ0)

Upload the Receiver sketch to the first Arduino, then upload the Transmitter sketch to the other. Leave the Transmitter-Arduino connected through USB and open the serial monitor.

You should see something like the following:

```
RF24/examples/pingpair_irq Sending
STATUS          = 0x0e RX_DR=0 TX_DS=0 MAX_RT=0 RX_P_NO=7 TX_FULL=0
RX_ADDR_P0-1    = 0xe8e8f0f0e1
RX_ADDR_P2-5    = 0xc3 0xc4 0xc5 0xc6
TX_ADDR         = 0xe8e8f0f0e1
RX_PW_P0-6      = 0x20 0x20 0x00 0x00 0x00 0x00
EN_AA           = 0x3f
EN_RXADDR       = 0x03
RF_CH           = 0x4c
RF_SETUP        = 0x07
CONFIG          = 0x0e
DYNPD/FEATURE   = 0x03 0x06
Data Rate       = 1MBPS
Model           = nRF24L01+
CRC Length      = 16 bits
PA Power        = PA_HIGH
Now sending 65
Send:OK
```

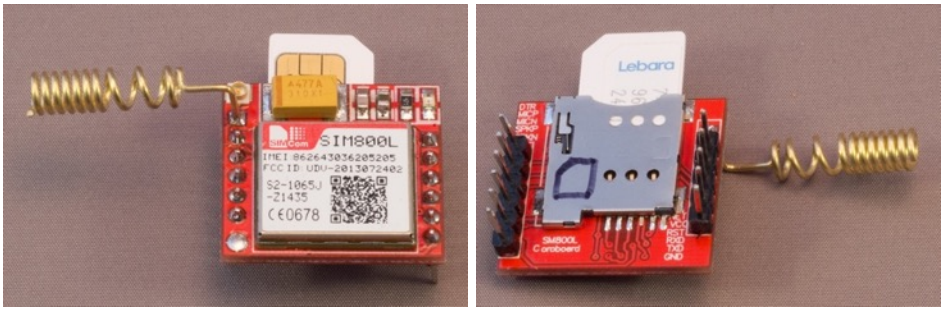
086_RF2.4_Transmit_NFR24L01

```
//Transmitter
#include <SPI.h>
#include "RF24.h"
#include "printf.h"

RF24 radio(9, 10);

byte address[][5] = { 0xCC, 0xCE, 0xCC, 0xCE, 0xCC , 0xCE, 0xCC, 0xCE,
0xCC, 0xCE};
static uint32_t message_count = 0;

void setup()
{
  Serial.begin(115200);
  printf_begin();
  Serial.print("RF24/examples/pingpair_irq");
  Serial.println("ROLE: Transmitter");
  radio.begin();
  radio.enableAckPayload();
  radio.enableDynamicPayloads();
  radio.openWritingPipe(address[0]);
  radio.openReadingPipe(1, address[1]);
  radio.printDetails();
  delay(50);
  attachInterrupt(0, check_radio, LOW);
}


void loop()
{
  unsigned long time = millis();
  Serial.print("Now sending ");
  Serial.println(time);
  radio.startWrite( &time, sizeof(unsigned long) , 0);
  delay(2000);
}

void check_radio(void)
{
  bool tx, fail, rx;
  radio.whatHappened(tx, fail, rx);
  if ( tx )
  {
    Serial.println("Send:OK");
  }
  if ( fail )
  {
    Serial.println("Send:Failed");
  }
  if ( rx || radio.available())
  {
    radio.read(&message_count, sizeof(message_count));
    Serial.print("Ack: ");
    Serial.println(message_count);
  }
}
```

087_RF2.4_Receive_NFR24L01

```
//Receiver
#include <SPI.h>
#include "RF24.h"
#include "printf.h"

RF24 radio(9, 10);

byte address[][5] = { 0xCC, 0xCE, 0xCC, 0xCE, 0xCC , 0xCE, 0xCC, 0xCE,
0xCC, 0xCE};
static uint32_t message_count = 0;

void setup()
{
  Serial.begin(115200);
  printf_begin();
  Serial.print("RF24/examples/pingpair_irq");
  Serial.println("ROLE: Receiver");
  radio.begin();
  radio.enableAckPayload();
  radio.enableDynamicPayloads();
  radio.openWritingPipe(address[1]);
  radio.openReadingPipe(1, address[0]);
  radio.startListening();
  radio.writeAckPayload( 1, &message_count, sizeof(message_count) );
  ++message_count;
  radio.printDetails();
  delay(50);
  attachInterrupt(0, check_radio, LOW);
}

void loop()
{
}

void check_radio(void)
{
  bool tx, fail, rx;
  radio.whatHappened(tx, fail, rx);
  if ( tx )
  {
    Serial.println("Ack Payload:Sent");
  }

  if ( fail )
  {
    Serial.println("Ack Payload:Failed");
  }

  if ( rx || radio.available())
  {
    static unsigned long got_time;
    radio.read( &got_time, sizeof(got_time) );
    Serial.print("Got payload ");
    Serial.println(got_time);
    radio.writeAckPayload( 1, &message_count, sizeof(message_count) );
    ++message_count;
  }
}
```

143. GSM/GPRS SIM800L module

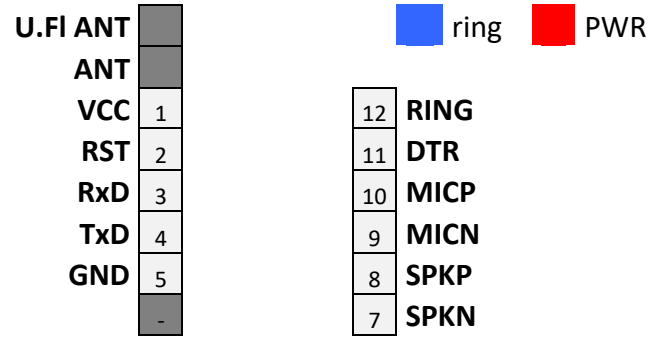


143.1. Specifications GSM/GPRS SIM800L module

- Power Supply 3.7-4.2 V with 1.5 A peak levels[1]
- Data levels 3.3 V[2]
- Quad band: GSM 850, EGSM 900, DCS 1800 and PCS 1900
- Transmitting power:
  - Class 4 (2W) at GSM 850 and EGSM 900
  - Class 1 (1W) at DCS 1800 and PCS 1900
- GPRS:
  - multi-slot class 1-12
  - Down-/Uplink max: 85.6 kbps
  - 2G (not 3G)
- SMS
- Phone Calls
- RTC from network
- Data
- UART
- Wire antenna
- U.Fl antenna

143.2. GSM/GPRS SIM800L module

143.3. Connections GSM/GPRS SIM800L module



[1] I've tested this with a 5V external power supply without frying the module. I can't guarantee this. Using the 5V directly from the Arduino DOES NOT WORK, because the Arduino does not deliver enough current (sometimes 1.5 A is needed!!).
[2] I've used 5V as data level without frying the module, so directly connected to the Arduino Digital pins, without logical level shifters. I can't guarantee this.

Pin nr	Name	Description	Arduino pin
1	VCC	Power supply	External power 2A
2	RST	Reset	Any Digital pin
3	RxD	Receive Data	Any Digital pin
4	TxD	Transmit Data	Any Digital pin
5	GND	Ground	GND
6	nc	not connected	Nc
7	SPKN	Speaker N?	Differential audio output
8	SPKP	Speaker P?	
9	MICN	Microphone N?	Differential audio input
10	MICP	Microphone P?	
11	DTR	Data Terminal Ready	Any Digital pin (optional)
12	RING	Incoming call/message	Any Digital pin (optional)

143.4. GSM/GPRS SIM800L module

- Datasheet SIM800L http://datasheetcafe.databank.netdna-cdn.com/wp-content/uploads/2016/03/SIM800L.pdf
- AT Command SIM800 https://www.waveshare.com/w/upload/2/20/SIM800_Series_AT_Command_Manual_V1.09.pdf

143.5. Libraries needed for GSM/GPRS SIM800L module

There are several libraries available that support the SIM800 chip. I've chosen for the Adafruit FONA library.
https://github.com/adafruit/Adafruit_FONA_Library/archive/master.zip

Library use explanation

```
#include "Adafruit_FONA.h"
```

Include Adafruit's FONA library.

```
#include <SoftwareSerial.h>
```

Include SoftwareSerial for communication between the Arduina and the 800L Sim module.

```
SoftwareSerial fonaSS = SoftwareSerial(<FONA_TX>, <FONA_RX>)
```

Create fonaSS the softwareserial port with FONA_TX being the Digital port connected to the TxD port on the SIM800L module and FONA_RX being the digital port connected to the RxD port on the SIM800L.

```
SoftwareSerial *fonaSerial = &fonaSS
```

Create a pointer to the softwareserial fonaSS.

```
Adafruit_FONA fona = Adafruit_FONA(FONA_RST)
```

Create fona an instance of the Adafruit_FONA clas. FONA_RST is the

```
fonaSerial->begin(4800)
```

Initialize the softwareserial fonaSS at 4800 baud.

```
fona.begin(*fonaSerial)
```

Initialize the SIM800L module.

```
fonaSerial->print("AT+CNMI=2,1\r\n");
```
Set up the SIM800L so it will sent a +CMTI notification on receiving an SMS/Text.

```
fona.available())
```
Is true if there is data available on the SIM800L.

```
*bufPtr = fona.read()
```
This pointer to a buffer gets the data available from the SIM800L.

```
while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount < (sizeof(fonaNotificationBuffer) - 1)))
```
Repeat the loop, as long as there are characters left in the buffer and as long as there is no End of Line character (\n) found.

```
if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE ",%d", &slot))
```

```
fona.getSMSSender(slot, callerIDbuffer, 31)
```
Gets the telephone number of the sender of the SMS/Text and stores it in callerIDbuffer.

```
fona.readSMS(slot, smsBuffer, 250, &smslen)) // pass in buffer & max len!
```
Gets the received SMS/Text message and stores is in smsBuffer.

```
fona.sendSMS(callerIDbuffer, "Hey, I got your text!")
```
Sends the message "Hey, I got you text!" to callerIDbuffer.

```
fona.deleteSMS(slot)
```
Deletes the last received message (this didn't work with my setup!!!).

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

143.6. Sample GSM/GPRS SIM800L module

The following sketc will answer every received SMS/Text message with the response SMS/Text: "Hey, I got your text!".

Sample Connections

- Connect VCC to 5V from an external power supply
- Connect GND to GND from an external power supply and to Arduino GND
- Connect RST to D9
- Connect TxD to D10
- Connect RxD to D11 through a voltage divider (to lower 5V output from Arduino to 3.3V).

146_SIM800L_smsresponse.ino

```
#include "Adafruit_FONA.h"

#define FONA_RST 9
#define FONA_TX 10
#define FONA_RX 11

char replybuffer[255];
```

```
#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

void setup() {
  while (!Serial);
  Serial.begin(115200);
  Serial.println(F("FONA SMS caller ID test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while (1);
  }
  Serial.println(F("FONA is OK"));
  fonaSerial->print("AT+CNMI=2,1\r\n");  //set up the FONA to send a +CMTI
notification when an SMS is received
  Serial.println("FONA Ready");
}

char fonaNotificationBuffer[64];          //for notifications from the FONA
char smsBuffer[250];

void loop() {
  char* bufPtr = fonaNotificationBuffer;    //handy buffer pointer
  if (fona.available())      //any data available from the FONA?
  {
    int slot = 0;            //this will be the slot number of the SMS
    int charCount = 0;
    //Read the notification into fonaInBuffer
    do  {
      *bufPtr = fona.read();
      Serial.write(*bufPtr);
      delay(1);
    } while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount <
(sizeof(fonaNotificationBuffer) - 1)));
    //Add a terminal NULL to the notification string
    *bufPtr = 0;
    //Scan the notification string for an SMS received notification.
    //  If it's an SMS message, we'll get the slot number in 'slot'
    if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE
",%d", &slot)) {
      Serial.print("slot: "); Serial.println(slot);
      char callerIDbuffer[32];  //we'll store the SMS sender number in here
      // Retrieve SMS sender address/phone number.
      if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
        Serial.println("Didn't find SMS message in slot!");
      }
      Serial.print(F("FROM: ")); Serial.println(callerIDbuffer);
      // Retrieve SMS value.
      uint16_t smslen;
      if (fona.readSMS(slot, smsBuffer, 250, &smslen)) { // pass in buffer
and max len!
        Serial.println(smsBuffer);
      }
      //Send back an automatic response
      Serial.println("Sending reponse...");
      if (!fona.sendSMS(callerIDbuffer, "Hey, I got your text!")) {
        Serial.println(F("Failed"));
```

```
      } else {
        Serial.println(F("Sent!"));
      }
      // delete the original msg after it is processed
      //   otherwise, we will fill up all the slots
      //   and then we won't be able to receive SMS anymore
      if (fona.deleteSMS(slot)) {
        Serial.println(F("OK!"));
      } else {
        Serial.print(F("Couldn't delete SMS in slot "));
Serial.println(slot);
        fona.print(F("AT+CMGD=?\r\n"));
      }
    }
  }
}
```

144. GSM/GPRS Neoway 590 DIY kit

This DIY kit came in parts. 2 SMD resistors, 1 SMD diode, 1 SMD LED, the SIM-card holder the DIL NEOWAY M590 and a PCB. You need good eyes and a steady hand to solder this together.

A nice instruction is available at http://guides.cyntech.co.uk/raspberry-pi/tutorial/soldering-the-m590-gsmgprs-module/

144.1. GSM/GPRS Neoway 590 DIY kit

- Power Supply 5V with 2 A peak levels
- Data levels 3.3 V
- Dual band
- Transmitting power:
  - Class 4 (2W) at GSM 850 and EGSM 900
  - Class 1 (1W) at DCS 1800 and PCS 1900
- GPRS:
  - Class 1-12
  - Down-/Uplink max: 85.6 kbps
  - 2G
- SMS
- Data
- UART
- U.Fl antenna

144.2. GSM/GPRS Neoway 590 DIY kit

144.3. GSM/GPRS Neoway 590 DIY kit

Pin nr	Name	Description	Arduino pin
1	GND		
2	+5V		

Pin nr	Name	Description	Arduino pin
1	I		
2	T		
3	R		
4	V		
5	K		
6	G	Ground	

144.4. GSM/GPRS Neoway 590 DIY kit

- Datasheet Neo_M590 http://cyntech.co.uk/downloads/neoway-m590-hardware-design-manual-v1.pdf
- AT Command Neo_M590 http://cyntech.co.uk/downloads/neoway-m590-at-command-sets-v3.pdf

144.5. Libraries needed for GSM/GPRS Neoway 590 DIY kit

There are several libraries available that support the SIM800 chip. I've chosen for the Adafruit FONA library.
https://github.com/adafruit/Adafruit_FONA_Library/archive/master.zip

Library use explanation

```
#include "Adafruit_FONA.h"
```
Include Adafruit's FONA library.

```
#include <SoftwareSerial.h>
```
Include SoftwareSerial for communication between the Arduina and the 800L Sim module.

```
SoftwareSerial fonaSS = SoftwareSerial(<FONA_TX>, <FONA_RX>)
```
Create fonaSS the softwareserial port with FONA_TX being the Digital port connected to the TxD port on the SIM800L module and FONA_RX being the digital port connected to the RxD port on the SIM800L.

```
SoftwareSerial *fonaSerial = &fonaSS
```
Create a pointer to the softwareserial fonaSS.

```
Adafruit_FONA fona = Adafruit_FONA(FONA_RST)
```
Create fona an instance of the Adafruit_FONA clas. FONA_RST is the

```
fonaSerial->begin(4800)
```
Initialize the softwareserial fonaSS at 4800 baud.

```
fona.begin(*fonaSerial)
```
Initialize the SIM800L module.

```
fonaSerial->print("AT+CNMI=2,1\r\n");
```
Set up the SIM800L so it will sent a +CMTI notification on receiving an SMS/Text.

```
fona.available())
```
Is true if there is data available on the SIM800L.

```
*bufPtr = fona.read()
```
This pointer to a buffer gets the data available from the SIM800L.

```
while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount <
(sizeof(fonaNotificationBuffer) - 1)))
```
Repeat the loop, as long as there are characters left in the buffer and as long as there is no End of Line character (\n) found.

```
if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE
",%d", &slot))
```

```
fona.getSMSSender(slot, callerIDbuffer, 31)
```
Gets the telephone number of the sender of the SMS/Text and stores it in callerIDbuffer.

```
fona.readSMS(slot, smsBuffer, 250, &smslen)) // pass in buffer & max len!
```
Gets the received SMS/Text message and stores is in smsBuffer.

```
fona.sendSMS(callerIDbuffer, "Hey, I got your text!")
```
Sends the message "Hey, I got you text!" to callerIDbuffer.

```
fona.deleteSMS(slot)
```

Deletes the last received message (this didn't work with my setup!!!).

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

144.6. Sample GSM/GPRS Neoway 590 DIY kit

The following sketch will answer every received SMS/Text message with the response SMS/Text: "Hey, I got your text!".

Sample Connections

- Connect VCC to 5V from an external power supply
- Connect GND to GND from an external power supply and to Arduino GND
- Connect RST to D9
- Connect TxD to D10
- Connect RxD to D11 through a voltage divider (to lower 5V output from Arduino to 3.3V).

146_SIM800L_smsresponse.ino

```
#include "Adafruit_FONA.h"

#define FONA_RST 9
#define FONA_TX 10
#define FONA_RX 11

char replybuffer[255];

#include <SoftwareSerial.h>
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

void setup() {
  while (!Serial);
  Serial.begin(115200);
  Serial.println(F("FONA SMS caller ID test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while (1);
  }
  Serial.println(F("FONA is OK"));
  fonaSerial->print("AT+CNMI=2,1\r\n");  //set up the FONA to send a +CMTI
notification when an SMS is received
  Serial.println("FONA Ready");
}

char fonaNotificationBuffer[64];          //for notifications from the FONA
char smsBuffer[250];

void loop() {
  char* bufPtr = fonaNotificationBuffer;    //handy buffer pointer
  if (fona.available())      //any data available from the FONA?
  {
```

```
    int slot = 0;            //this will be the slot number of the SMS
    int charCount = 0;
    //Read the notification into fonaInBuffer
    do  {
      *bufPtr = fona.read();
      Serial.write(*bufPtr);
      delay(1);
    } while ((*bufPtr++ != '\n') && (fona.available()) && (++charCount <
(sizeof(fonaNotificationBuffer) - 1)));
    //Add a terminal NULL to the notification string
    *bufPtr = 0;
    //Scan the notification string for an SMS received notification.
    //  If it's an SMS message, we'll get the slot number in 'slot'
    if (1 == sscanf(fonaNotificationBuffer, "+CMTI: " FONA_PREF_SMS_STORAGE
",%d", &slot)) {
      Serial.print("slot: "); Serial.println(slot);
      char callerIDbuffer[32];  //we'll store the SMS sender number in here
      // Retrieve SMS sender address/phone number.
      if (! fona.getSMSSender(slot, callerIDbuffer, 31)) {
        Serial.println("Didn't find SMS message in slot!");
      }
      Serial.print(F("FROM: ")); Serial.println(callerIDbuffer);
      // Retrieve SMS value.
      uint16_t smslen;
      if (fona.readSMS(slot, smsBuffer, 250, &smslen)) { // pass in buffer
and max len!
        Serial.println(smsBuffer);
      }
      //Send back an automatic response
      Serial.println("Sending reponse...");
      if (!fona.sendSMS(callerIDbuffer, "Hey, I got your text!")) {
        Serial.println(F("Failed"));
      } else {
        Serial.println(F("Sent!"));
      }
      // delete the original msg after it is processed
      //   otherwise, we will fill up all the slots
      //   and then we won't be able to receive SMS anymore
      if (fona.deleteSMS(slot)) {
        Serial.println(F("OK!"));
      } else {
        Serial.print(F("Couldn't delete SMS in slot "));
Serial.println(slot);
        fona.print(F("AT+CMGD=?\r\n"));
      }
    }
  }
}
```

145. SIMCOM SIM7020E (GSM) LTE NBIoT breakout board


145.1. Specifications SIMCOM SIM7020E GSM LTE NBIoT breakout board

- Bands: B1, B3, B5, B8, B20 & B28
- Compatible with SIM800C
- Broad coverage compared to GSM (good for inside buildings):
  - Strong gain
  - Wide signal coverage
- Supply voltage range 2.1-3.6V (typical 3.3V)
- NB-IoT:
  - Upstream 62,5 kbps
  - Downlink 26.15 kbps
- SMS:
  - Text and PDU modes
- Certification:
  - CE / GCF
  - RoHS / REACH
- Interface
  - Controlled by AT commands
  - Serial port USIm card (1.8/3V)
  - GPIO
  - ADC
- Other features:
  - Upgrade software through serial port
  - LWM2M / COAP
  - MQTT FTP, HTTP, HTTPS, SSL, DTLS, FOTA
  - PSM, eDRX

145.2. Datasheet SIMCOM SIM7020E GSM LTE NBIoT breakout board

Datasheets, a list of AT commands and other interesting information about the SIM7020 can be found at:

- http://simcomm2m.com/En/service/down.aspx?proType=&type=&proName=188&time=

145.3. Connections SIMCOM SIM7020E GSM LTE NBIoT breakout board

Pin nr	Name	Description	Arduino pin
1	VIO		n.c.
2	TxD	Transmit Data	D0, or another port when using Software Serial

3	RxD	Receive Data	D1, or another port when using Software Serial
4	GND	Ground	GND + GND External Power (both need to be connected)
5	Vin	V 5-16V	External Power 5-16V
6	PWK	?	n.c.
7	Rx2	?	n.c.
8	Tx2	?	n.c.
9	DTR	Data Terminal Ready	n.c.
10	RI		n.c.
11	GPIO0	General Purpose IO-0	n.c.
12	Eint	?esp	n.c.

145.4. Libraries needed for SIMCOM SIM7020E GSM LTE NBIoT breakout board

- Xx

Library use explanation

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

145.5. Sample SIMCOM SIM7020E GSM LTE NBIoT breakout board

The following sketch

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
void setup()
{
}

void loop()
{
}
```

146. LoRa

LoRa stands for Low Power/Long Range wireless modules. Under the right circumstances two LoRa modules can communicate over a distance of 10 km or even more. They can play an important role in Internet Of Things networks like The Things Network.

Several of these modules are covered in the Things Network Nodes section.

- RFM95W module
- Dragino Lora Shield
- RN2483

147. Grove Lora Radio 868

This module is a LoRa radio for long rang, low power communication. There is a RFM95W LoRa 868 MHz radio module and a ATMega168 MCU. The ATMega168 MCU serves as an interface between your Arduino (serial connection) and the RFM95W LoRa radio module through SPI.

It can be used for stand alone networks, but there are no libraries to communicate with LoRaWAN network (yet) like The Things Network.

147.1. Specifications Grove Lora Radio 868

- Grove Connector: D2..D8
- RFM95W based on SX1276 LoRa
- Supply voltage: 5V/3.3V
- ATMega168 MCU onboard as interface between RFM95W (SPI) and your Arduino (serial)
- Simple wire antenna or MHF connector for external high gain antenna
- 868 MHz
- Datarate: 0.3 - 50 kps
- +20dBm 100 mW power output capacity

147.2. Seeed documentation Grove Lora Radio 868

- http://wiki.seeedstudio.com/Grove_LoRa_Radio/

147.3. Connections Grove Tilt v1.1

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 -	-	-		-
1 Yellow	SIG	Signal		Any digital port

147.4. Libraries needed for Grove Lora Radio 868

- Grove - LoRa Radio 433MHz 868 MHz library by Seeed studio, through the Library Manager.

Library use explanation

```
#include <SoftwareSerial.h>
```

Include the SoftwareSerial libray to let your Arduino connect with the ATMega168 through a virtual serial connection

```
#include <RHReliableDatagram.h>
```
Include the RHReliableDatagram library from Seed Studio

```
#include <RH_RF95.h>
```
Include the RH_RF95 library from Seed Studio

```
#define CLIENT_ADDRESS 12
```
This is the address the client is going to use to send on and it is the address on which the server is going to listen.

```
#define SERVER_ADDRESS 34
```
This is the addres the server is going to use to send on and it is the address on which the client is going to listen.

```
uint8_t outgoingMessage[] = "Hello World!";
```
This array contains the message that is going to be send.

```
uint8_t incommingMessage[RH_RF95_MAX_MESSAGE_LEN];
```
This array will be used to store the received message.

```
SoftwareSerial ss(5, 6); //D5
```
Create an instance of SoftwareSerial named 'ss' with the ports on which the Grove connector is connected. Digital Arduino Ports 5,6 correspond with the D5 grove connector.

```
RH_RF95 driver(ss);
```
Create an instance of RH_RF95 named 'driver' with the software serial on which the grove connector is connected.

```
RHReliableDatagram manager(driver, CLIENT_ADDRESS);
```
Create an instance of RHReliableDatagram named 'manager', with 'driver' as the RFM95 module and the address that is going to be used for sending messages.

```
manager.init()
```
Initialize 'manager'.

```
manager.available()
```
This is true when a message was received from the Client

```
manager.recvfromAck(incommingMessage, &len, &sourceAddress)
```
Retrieve message from sourceAddress and store this in incommingMessage.

```
manager.sendtoWait(replyMessage, sizeof(replyMessage), sourceAddress)
```
Send the message stored in replyMessage to the client on sourceAddress.

```
manager.recvfromAckTimeout(incommingMessage, &len, 2000, &sourceAddress)
```
Wait for 2000ms for a message from sourceAddress.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

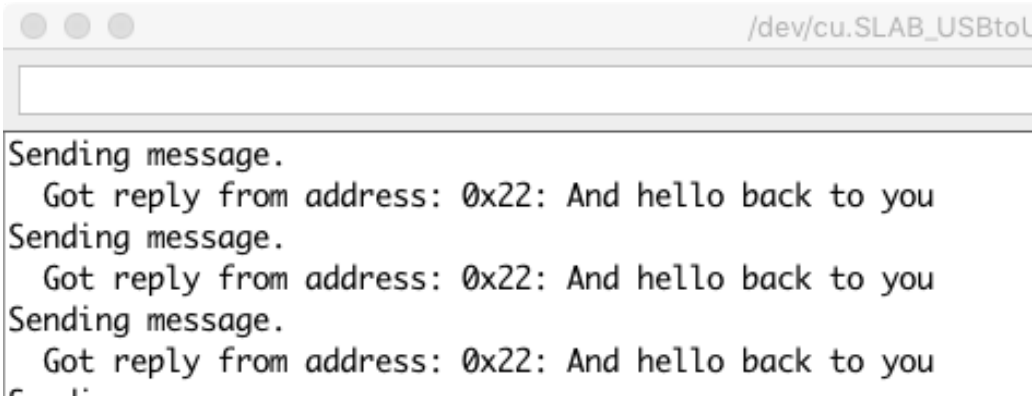
147.5. Sample Grove Lora Radio 868

The following 2 sketches exchange messages. The client sends a message to the server and displays the response of the server. The server waits for a message and replies to that message.

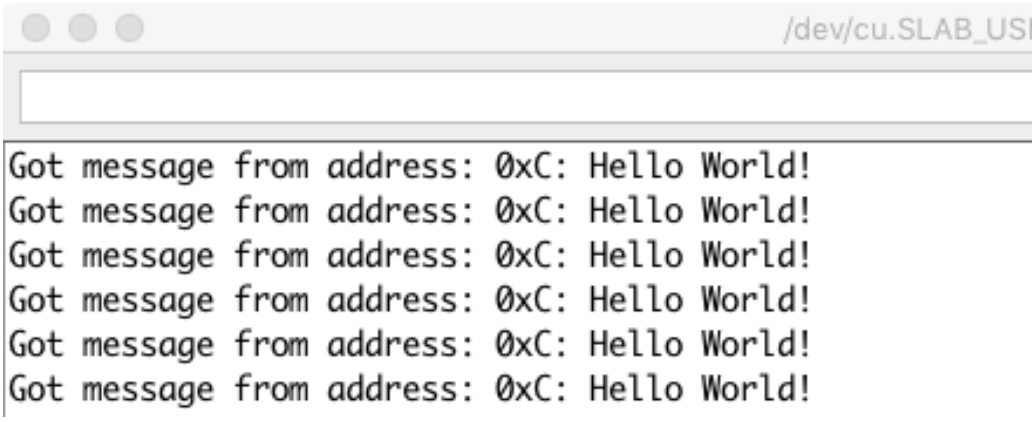
Sample Connections

Use 2 Arduino's with a Grove-shield and 2 Grove LoRa Radio's and make the following connections on both Arduino's.

- Connect this module to D5 on your Grove-shield.



This screendump shows the output on the Client.



This screendump shows the output on the server.

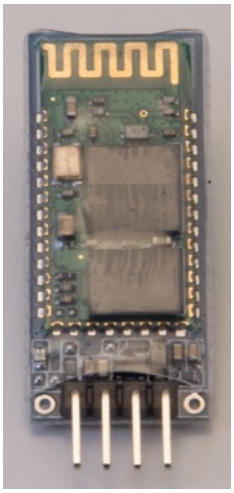
170_Grove_LoraRadioClient

```
#include <SoftwareSerial.h>
#include <RHReliableDatagram.h>
#include <RH_RF95.h>
#define CLIENT_ADDRESS 12
#define SERVER_ADDRESS 34

uint8_t outgoingMessage[] = "Hello World!";
uint8_t incommingMessage[RH_RF95_MAX_MESSAGE_LEN];

SoftwareSerial ss(5, 6); //D5
RH_RF95 driver(ss);

RHReliableDatagram manager(driver, CLIENT_ADDRESS);

void setup()
{
  Serial.begin(115200);
  if (!manager.init())
  {
    Serial.println("init failed");
  }
}

void loop()
{
  Serial.println("Sending message.");
  if(manager.sendtoWait(outgoingMessage, sizeof(outgoingMessage),
SERVER_ADDRESS))
  {
    uint8_t len = sizeof(incommingMessage);
    uint8_t sourceAddress;
    if(manager.recvfromAckTimeout(incommingMessage, &len, 2000,
&sourceAddress))
    {
      Serial.print("  Got reply from address: 0x");
      Serial.print(sourceAddress, HEX);
      Serial.print(": ");
      Serial.println((char*)incommingMessage);
    }
    else
    {
      Serial.println("No reply, is the erver running?");
    }
  }
  else
  {
    Serial.println("sendtoWait failed");
  }
  delay(2000);
}
```

171_Grove_LoraRadioServer

```
#include <SoftwareSerial.h>
#include <RHReliableDatagram.h>
#include <RH_RF95.h>
#define CLIENT_ADDRESS 12
#define SERVER_ADDRESS 34

uint8_t replyMessage[] = "And hello back to you";
uint8_t incommingMessage[RH_RF95_MAX_MESSAGE_LEN];

SoftwareSerial ss(5, 6);  //D5
RH_RF95 driver(ss);

RHReliableDatagram manager(driver, SERVER_ADDRESS);

void setup()
{
  Serial.begin(115200);
  if (!manager.init())
  {
    Serial.println("init failed");
  }
}

void loop()
{
  if(manager.available())
  {
    uint8_t len = sizeof(incommingMessage);
    uint8_t sourceAddress;
    if(manager.recvfromAck(incommingMessage, &len, &sourceAddress))
    {
      Serial.print("Got message from address: 0x");
      Serial.print(sourceAddress, HEX);
      Serial.print(": ");
      Serial.println((char*)incommingMessage);
      if(!manager.sendtoWait(replyMessage, sizeof(replyMessage),
sourceAddress))
      {
        Serial.println("sendtoWait failed");
      }
    }
  }
}
```

148. Bluetooth Keyes BT_Board v2.0



148.1. Specifications Bluetooth Keyes BT_Board v2.0

- HC-06
- Slave-only, this means it can only pair with master Bluetooth modules like smartphones & computers, but not with other slave modules.

148.2. Datasheet Bluetooth Keyes BT_Board v2.0

- http://silabs.org.ua/bc4/hc06.pdf

148.3. Connections Bluetooth Keyes BT_Board v2.0

Pin nr	Name	Description	Arduino pin
1	RxD	Receive data	Any digital port with Software Serial through a voltage divider[1]
2	TxD	Transmit data	Any digital port with Software Serial
3	GND	Ground	GND
4	VCC	3.6 – 6 V	5V

148.4. LED status

onboard LED status	Meaning
fast blink	either AT command mode or UART mode (no difference in blinking speed)
twice every 2 seconds	paired and connected

148.5. Libraries needed for Bluetooth Keyes BT_Board

- SoftwareSerial library included with Arduino IDE

[1] You can make a voltage divider by connecting a 2.2k ohm resistor to ground and a 1k ohm resistor to Tx on the Arduino (D1). Connect the other legs of both resistors to Rxd on the Bluetooth module. This gives an input voltage of 2200 / (2200 + 1000)*5 = 3.4 V and this is close enough to 3.3 V.

148.6. AT command mode Keyes BT_Board v2.0

All Bluetooth modules can be configured through AT commands. Before you can enter AT commands, you must first set the module in AT command mode.

This module will automatically enter AT mode after boot, but will step into UART mode as soon as another device connects to it (it can be paired, but it can't be connected).

To set this specific Bluetooth module in AT command mode follow the steps below

- Beside Vcc, GND, RxD and TxD , no extra connections are needed.
- Make sure the Bluetooth module is **not connected** with another device (it can be paired, but it can't be connected)!
- Bluetooth communication settings: 9600,0,0 (baudrate between Arduino and Bluetooth module)
- Terminal settings: no line ending

The blinking speed of the onboard LED does NOT reflect whether it is in command mode or not.

AT commands Keyes BT_Board v2.0

Below is a list of some of the AT commands for this Bluetooth module.

Action	AT-command	Result	Description
test	`AT`	OK	*Test connection with the Bluetooth module*
Factory default	**no command available**		*Set Bluetooth module to factory defaults:* *name:HC-06* *baud: 9600,0,0* *pincode: 1234* *role: slave*
Show name	**no command available**		
Set name	`AT+NAMEhc-06`	OKsetname	
Show role	**no command available**		*always slave*
Set role	**no command available**		*always slave*
Show baudrate	**no command available**		
Set baudrate	`AT+BAUD4 (9600)`	OK9600	*Set baudrate to 9600* *1 1.200* *2 2.400* *3 4.800* *4 9.600* *5 19.200* *6 38.400* *7 57.600* *57.600 is highest baudrate when using Software Serial!*
Show pincode	**no command available**		
Set Pincode	`AT+PIN1234`	ATsetpin	*Set pincode to 1234*
Show version	`AT+VERSION`	OKlinvorV1.5	*Shows current firmware version*
Reboot	**no command available**		
Show Address	**no command available**		

More information can be found at:
http://www.puntoflotante.net/BOLT-SYSTEM-BLUETOOTH-AT-COMMANDS.htm

099_BT_HC06_AT.ino

The following sketch copies all commands typed in Serial Monitor to the Bluetooth module. All responses from the Bluetooth module are then copied to the Serial Monitor. This is a convenient way to change the Bluetooth module's settings.

- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (these 2 resistors act as a Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V
- Set Serial Monitor to 57600 and to "no line ending"
- Make sure the Bluetooth module is not connected with another device!

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // Tx pin of hc06 module | Rx pin of
module

void setup()
{
  Serial.begin(57600); // Speed depends on the Serial monitor settings
  Serial.println("Enter AT commands:");
  BTSerial.begin(9600);  // Speed depends on the module used hc06: 9600
}

void loop()
{
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

148.7. UART mode Keyes BT_Board v2.0

UART mode is needed when you want to transfer data from slave to master and vice versa. In UART mode you can not issue AT commands. Your sketch needs to interpret the meaning of the received data. To enter UART mode, you'll just need to pair this Bluetooth module with another device (Apple's IOS does not support this cheap Bluetooth module). At the master, you'll need some kind of terminal program on the paired device (Serial Monitor, PuTTY on Windows/OSx or Serial Bluetooth Terminal on Android smartphone) and make sure you select the paired Bluetooth module as your port (COMx, /dev/cu.hc-06-DevB, hc-06).

098_BT_HC06_HelloWorld.ino

This sketch sends the famous line "Hello World" to the master device.

- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (resistors act as Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  BTSerial.begin(9600);
}

void loop()
{
  BTSerial.println("Hello World");
  delay(1000);
}
```

Two way transfer of data (099_BT_HC06_AT.ino)

You can also use the sketch that was used for AT commands and send data back and forth between the Bluetooth module and the master. Input at one device will be output at the other. Make sure you are in UART mode.

149. Bluetooth JY-MCU BT_Board v1.06

149.1. Specifications Bluetooth JY-MCU BT_Board v1.06

- HC-05
- Slave or master

149.2. Datasheet Bluetooth JY-MCU BT_Board v1.06

- http://www.electronica60norte.com/mwfls/pdf/newBluetooth.pdf
- AT command set HC-05 http://www.instructables.com/files/orig/FOR/4FP2/HKZAVRT6/FOR4FP2HKZAVRT6.pdf

149.3. Connections Bluetooth JY-MCU BT_Board v1.06

Pin nr	Name	Description	Arduino pin
1	State	Reflects the paired state.	Any digital port
2	RxD	Receive data	Any digital port with Software Serial through a voltage divider[1]
3	TxD	Transmit data	Any digital port with Software Serial
4	GND	Ground	GND
5	VCC	3.6 – 6 V	5V
6	KEY	Connected to pin 34 of the HC-05	Needs to be HIGH at boot, to enter AT command mode

149.4. LED status

onboard LED status	Meaning
fast blink	UART mode
slow blink	AT command mode
ON	paired and connected

149.5. Libraries needed for Bluetooth JY-MCU BT_Board v1.06

- SoftwareSerial library included with Arduino IDE.

[1] You can make a voltage divider by connecting a 2.2k ohm resistor to ground and a 1k ohm resistor to Tx on the Arduino (D1). Connect the other legs of both resistors to Rxd on the Bluetooth module. This gives an input voltage of 2200 / (2200 + 1000)*5 = 3.4 V and this is close enough to 3.3 V.

149.6. AT command mode JY-MCU BT_Board v1.06

All Bluetooth modules can be configured through AT commands. Before you can enter AT commands, you must first set the module in AT command mode.

The default mode of this module is UART mode. To enter AT command mode, you'll have to connect KEY to 5V during boot of the module. You can recognize AT command mode at the slow blink speed of the onboard LED.

To set this specific Bluetooth module in AT command mode follow the steps below

- Connect KEY to 5V.
- Beside Key, Vcc, GND, RxD and TxD , no extra connections are needed.
- Make sure the Bluetooth module is **not connected** with another device (it can be paired, but can't be connected)!
- Bluetooth communication settings: 38400,0,0 (baudrate between Arduino and Bluetooth module)
- Terminal settings: both NL & CR

AT commands JY-MCU BT_Board v1.06

Below is a list of some of the AT commands for this Bluetooth module.

Action	AT-command	Result	Description
test	`AT`	OK	*Test connection with the Bluetooth module*
Factory default	`AT+ORGL`	OK	*Set Bluetooth module to factory defaults:* *name: H-C-2010-06-01* *baud: 38400,0,0* *pin code: 1234* *role: slave*
Show name	`AT+NAME`	+NAME:H-C-2010-06-0 OK	*Show current name*
Set name	`AT+NAME=JY-MCU5-1`	OK	*Set the name to: JY-MCU5-1*
Show role	`AT+ROLE`	+ROLE:0 OK	*Show current role* *0 slave* *1 master* *2 slave-loop*
Set role	`AT+ROLE=0`	OK	*Set role to slave*
Show baudrate	`AT+UART`	+UART:38400,0,0 OK	*Current baudrate*
Set baudrate	`AT+UART:38400,0,0`	OK	*Set baudrate to 38.400* *57.600 is highest baudrate when using Software Serial!*

Action	AT-command	Result	Description
Show pincode	`AT+PSWD`	+PSWD:1234	*Show current pin code*
Set Pincode	`AT+PSWD=1234`	OK	*Set pincode to 1234*
Show version	`AT+VERSION`	+VERSION:2.0-20100601 OK	*Shows current firmware version*
Reboot	`AT+RESET`		*Reboots the module*
Show Address	`AT+ADDR`	+ADDR:14:1:212230 OK	*Shows the address*

More information can be found at:
https://cdn.instructables.com/ORIG/FOR/4FP2/HKZAVRT6/FOR4FP2HKZAVRT6.pdf

097_BT_JY-MCU_AT.ino

The following sketch copies all commands typed in Serial Monitor to the Bluetooth module. All responses from the Bluetooth module are then copied to the Serial Monitor. This is a convenient way to change the Bluetooth module's settings.

- Connect KEY to 5V
- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (these 2 resistors act as a Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V
- Set Serial Monitor to 57600 and to "Both NL & CR"
- Make sure the Bluetooth module is not connected with another device!

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // Tx pin of hc06 module | Rx pin of
module

void setup()
{
  Serial.begin(57600); // Speed depends on the Serial monitor settings
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);  // Speed depends on the module used JY-MCU: 38400
}

void loop()
{
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

149.7. UART mode JY-MCU BT_Board v1.06

UART mode is needed when you want to transfer data from slave to master and vice versa. In UART mode you can not issue AT commands. Your sketch needs to interpret the meaning of the received data. To enter UART mode, you'll just need to pair this Bluetooth module with another device (Apple's IOS does not support this cheap Bluetooth module). At the master, you"ll need some kind of terminal program on the paired device (Serial Monitor, PuTTY on Windows/OSx or Serial Bluetooth Terminal on Android smartphone) and make sure you select the paired Bluetooth module as your port (COMx, /dev/cu.hc-06-DevB, hc-06)

098_BT_JY-MCU_HelloWorld.ino

This sketch sends the famous line "Hello World" to the master device.

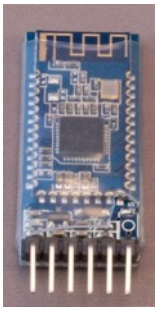
- Make sure KEY is **not connected** to 5V
- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (resistors act as Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V

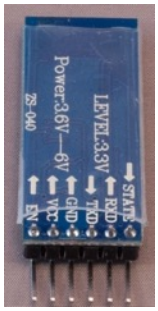
```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  BTSerial.begin(38400);
}

void loop()
{
  BTSerial.println("Hello World");
  delay(1000);
}
```

Two way transfer of data (097_BT_JY-MCU_AT.ino)

You can also use the sketch that was used for AT commands and send data back and forth between the Bluetooth module and the master. Input at one device will be output at the other. Make sure you are in UART mode.

150. Bluetooth 4.0 BLE CC41A (CC2541) module





150.1. Specifications Bluetooth 4.0 BLE CC41A (CC2541) module

- Bluetooth 4.0 BLE
- HM-10
- Bluetooth Class 1 and Class 2 mode
- Standard BLE protocol
- UART, I2C protocol
- Low power
- Input voltage: 3.6-6V
- Data levels: 3.3V
- Supports Android (4.3 or newer), IOS (4s or newer), Windows, OSx
- Max distance 60m
- Default baud rate: 9600

Note: BLE is only compatible with devices that also have BLE. It is not compatible with normal Bluetooth 2.0. It needs a different way of pairing!

150.2. Datasheet Bluetooth 4.0 BLE CC41A (CC2541) module

- iBeacon
  http://www.blueluminance.com/HM-10-as-iBeacon.pdf
- HM-10 BLE-CC41A
  http://www.martyncurrey.com/bluetooth-modules/
- AT commands
  http://denethor.wlu.ca/arduino/MLT-BT05-AT-commands-TRANSLATED.pdf

150.3. Connections Bluetooth 4.0 BLE CC41A (CC2541) module

Pin nr	Name	Description	Arduino pin
1	STATE		?
2	RxD	Receive data	Any digital port with Software Serial through a voltage divider[1]
3	TxD	Transmit data	Any digital port with Software Serial
4	GND	Ground	GND
5	VCC	3.6 – 6 V	5V
6	EN		?

150.4. LED status

onboard LED status	Meaning
blink (once per second)	either AT command mode or UART mode (no difference in blinking speed)
ON	paired and connected

150.5. Libraries needed for Bluetooth 4.0 BLE CC41A (CC2541) module

- SoftwareSerial library included with Arduino IDE

[1] You can make a voltage divider by connecting a 2.2k ohm resistor to ground and a 1k ohm resistor to Tx on the Arduino (D1). Connect the other legs of both resistors to Rxd on the Bluetooth module. This gives an input voltage of 2200 / (2200 + 1000)*5 = 3.4 V and this is close enough to 3.3 V.

150.6. AT command mode Bluetooth 4.0 BLE CC41A (CC2541)

All Bluetooth modules can be configured through AT commands. Before you can enter AT commands, you must first set the module in AT command mode.

This module will automatically enter AT mode after boot, but will step into UART mode as soon as another device connects to it (it can be paired, but it can't be connected

To set this specific Bluetooth module in AT command mode follow the steps below

- Beside Vcc, GND, RxD and TxD , no extra connections are needed.
- Make sure the Bluetooth module is **not connected** with another device (it can be paired, but it can't be connected)!
- Communication settings: 9600,0,0 (baudrate between Arduino and Bluetooth module)
- Terminal settings: both NL & CR

AT commands Bluetooth 4.0 BLE CC41A (CC2541)

Below is a list of some of the AT commands for this Bluetooth module.

Action	AT-command	Result	Description
test	`AT+name` Just AT does not respond with OK	+NAME=MLT-BT05	*Test connection with the Bluetooth module*
Factory default	`AT+DEFAULT`	O ?MLD-BT05-V2.1	*Set Bluetooth module to factory defaults:* *name: MLT-BT05* *baud: 9600,0,0* *pin code: 123456* *role: slave*
Show name	`AT+NAME`	+NAME=MLT-BT05	*Show current name*
Set name	`AT+NAMEzs-04`	+NAME=zs-04 OK	*Set name to zs-04*
Show role	`AT+ROLE`	+ROLE=0	*Show current role* *0 slave* *1 master* *2 slave-loop*
Set role	`AT+ROLE0`	+ROLE=0 OK	
Show baudrate	`AT+BAUD`	+BAUD=0	*Current baudrate* *Description on Internet about which nr is which baudrate is probably incorrect*
Set baudrate	`AT+BAUD=0`	+BAUD=0 OK	*Description on Internet about*

			which nr is which baudrate is probably incorrect
Show pincode	`AT+PIN`	+PIN=123456	*Show current pincode (6 digits)*
Set Pincode	`AT+PIN=123456`	+PIN=123456 OK	*Set pincode to 123456 (6 digits)*
Show version	`AT+VERSION`	MLT-BT05-V3.9	*Shows current firmware version*
Reboot	`AT+RESET`	OK MLD-BT05-V2.1	*Reboot the module*
Show Address	`AT+LADDR`	+LADDR=A8:1B:6A:9F:72:49	*Show address module*

More information can be found at:
http://denethor.wlu.ca/arduino/MLT-BT05-AT-commands-TRANSLATED.pdf

147_BTBLE_ZS-04_AT.ino

The following sketch copies all commands typed in Serial Monitor to the Bluetooth module. All responses from the Bluetooth module are then copied to the Serial Monitor. This is a convenient way to change the Bluetooth module's settings.

- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (these 2 resistors act as a Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V
- Set Serial Monitor to 9600 and to "Both NL & CR" (I tried at 57600, but that didn't work)
- Make sure the Bluetooth module is not connected with another device!

```
SoftwareSerial BTSerial(10, 11); // Tx pin of hc06 module | Rx pin of
module

void setup()
{
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(9600);  // Speed depends on the module used: ZS-04
Bluetooth 4.0 BLE CC41A (CC2541): 9600
}

void loop()
{
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

150.7. UART mode Bluetooth 4.0 BLE CC41A (CC2541)

UART mode is needed when you want to transfer data from slave to master and vice versa. In UART mode you can not issue AT commands. Your sketch needs to interpret the meaning of the received data. To enter UART mode, you'll just need to pair this Bluetooth module with another device (Apple's IOS does not support this cheap Bluetooth module). At the master, you'll need some kind of terminal program on the paired device (Serial Monitor, PuTTY on Windows/OSx or Serial Bluetooth Terminal on Android smartphone) and make sure you select the paired Bluetooth module as your port (COMx, /dev/cu.hc-06-DevB, hc-06).

148_BTBLE_ZS-04_HelloWorld.ino

This sketch sends the famous line "Hello World" to the master device.

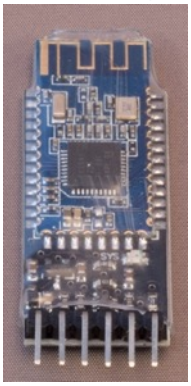
- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (resistors act as Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V

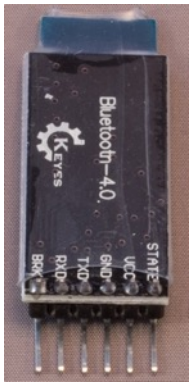
```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  BTSerial.begin(38400);
}

void loop()
{
  BTSerial.println("Hello World");
  delay(1000);
}
```

Two way transfer of data (147_BTBLE_ZS-04_AT.ino)

You can also use the sketch that was used for AT commands and send data back and forth between the Bluetooth module and the master. Input at one device will be output at the other. Make sure you are in UART mode.

151. Keyes Bluetooth 4.0 BLE





151.1. Specifications Keyes Bluetooth 4.0 BLE

- Bluetooth 4.0 BLE
- HM-10
- Standard BLE protocol
- Input voltage: 5V
- Logic level shifters so the data levels are 5V
- Supports Android (4.3 or newer), IOS (4s or newer), Windows, OSx
- Default baud rate: 9600

Note: BLE is only compatible with devices that also have BLE. It is not compatible with normal Bluetooth 2.0

151.2. Datasheet Bluetooth Keyes Bluetooth 4.0 BLE

http://www.martyncurrey.com/bluetooth-modules/

151.3. Connections Keyes Bluetooth 4.0 BLE

Pin nr	Name	Description	Arduino pin
1	STATE		?
2	VCC	5V	5V
3	GND	Ground	GND
4	TxD	Transmit data	Any digital port with Software Serial through a voltage divider[1]
5	RxD	Receive data	Any digital port with Software Serial
6	BRK		?

151.4. LED status

onboard LED status	Meaning
blink (once per second)	either AT command mode or UART mode (no difference in blinking speed)
ON	paired and connected

[1] You can make a voltage divider by connecting a 2.2k ohm resistor to ground and a 1k ohm resistor to Tx on the Arduino (D1). Connect the other legs of both resistors to Rxd on the Bluetooth module. This gives an input voltage of 2200 / (2200 + 1000)*5 = 3.4 V and this is close enough to 3.3 V.

151.5. Libraries needed for Keyes Bluetooth 4.0 BLE

- SoftwareSerial library included with Arduino IDE.

151.6. AT command mode Keyes Bluetooth 4.0 BLE

All Bluetooth modules can be configured through AT commands. Before you can enter AT commands, you must first set the module in AT command mode.

This module will automatically enter AT mode after boot, but will step into UART mode as soon as another device connects to it (it can be paired, but it can't be connected

To set this specific Bluetooth module in AT command mode follow the steps below

- Beside Vcc, GND, RxD and TxD , no extra connections are needed.
- Make sure the Bluetooth module is **not connected** with another device (it can be paired, but it can't be connected)!
- Communication settings: 9600,0,0 (baudrate between Arduino and Bluetooth module)
- Terminal settings: no line ending

AT commands Keyes Bluetooth 4.0 BLE

Below is a list of some of the AT commands for this Bluetooth module.

Action	AT-command	Result	Description
test	`AT`	OK	*Test connection with the Bluetooth module*
Factory default	`AT+RENEW`	OK+RENEW	*Set Bluetooth module to factory defaults:* *name: HMSoft* *baud: 9600,0,0* *pin code: 000000* *role: slave*
Show name	`AT+NAME?`	OK+NAME:HMSoft	*Show current name*
Set name	`AT+NAMEkeyes04`	OK+Set:keyes04	*Set name to keyes04*
Show role	`AT+ROLE?`	OK+Get:0	*Show current role* *0 slave* *1 master* *2 slave-loop*
Set role	`AT+ROLE0`	OK+Set:0	
Show baudrate	`AT+BAUD?`	OK_Get:0	*Current baudrate*
Set baudrate	`AT+BAUD0`	OK+Set:0	
Show pincode	`AT+PASS?`	OK+Get:000000	*Show current pincode (6 digits)*
Set Pincode	`AT+PASS123456`	+PIN=123456 OK	*Set pincode to 123456 (6 digits)*
Show version	`AT+VERR?`	HMSoft V540	*Shows current firmware version*
Reboot	`AT+RESET`	OK+RESET	*Reboot the module*

Show Address	`AT+ADDR?`	OK+ADDR:606405D13411	*Show address module*

More information can be found at:
https://www.instructables.com/id/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/

149_BTBLE_Keyes_AT.ino

The following sketch copies all commands typed in Serial Monitor to the Bluetooth module. All responses from the Bluetooth module are then copied to the Serial Monitor. This is a convenient way to change the Bluetooth module's settings.

- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (these 2 resistors act as a Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V
- Set Serial Monitor to 9600 and to "no line ending" (I tried at 57600, but that didn't work)
- Make sure the Bluetooth module is not connected with another device!

```
SoftwareSerial BTSerial(10, 11); // Tx pin of hc06 module | Rx pin of
module

void setup()
{
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(9600);  // Speed depends on the module used: ZS-04
Bluetooth 4.0 BLE CC41A (CC2541): 9600
}

void loop()
{
  if (BTSerial.available())
    Serial.write(BTSerial.read());

  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

151.7. UART mode Keyes Bluetooth 4.0 BLE

UART mode is needed when you want to transfer data from slave to master and vice versa. In UART mode you can not issue AT commands. Your sketch needs to interpret the meaning of the received data. To enter UART mode, you'll just need to pair this Bluetooth module with another device (Apple's IOS does not support this cheap Bluetooth module). At the master, you'll need some kind of terminal program on the paired device (Serial Monitor, PuTTY on Windows/OSx or Serial Bluetooth Terminal on Android smartphone) and make sure you select the paired Bluetooth module as your port (COMx, /dev/cu.hc-06-DevB, hc-06).

150_BTBLE_Keyes_HelloWorld.ino

This sketch sends the famous line "Hello World" to the master device.

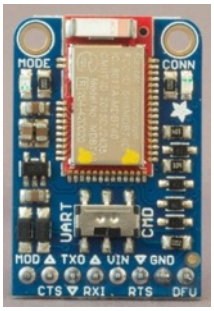
- Connect RxD to GND through a 2K (2.2K) resistor
- Connect RxD to D11 through a 1K resistor (resistors act as Voltage divider)
- Connect TxD to D10
- Connect GND to GND
- Connect VCC to 5V

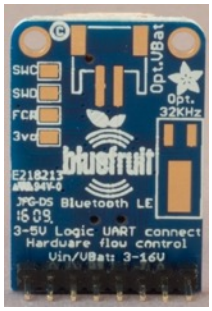
```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  BTSerial.begin(38400);
}

void loop()
{
  BTSerial.println("Hello World");
  delay(1000);
}
```

Two way transfer of data (149_BTBLE_Keyes_AT.ino)

You can also use the sketch that was used for AT commands and send data back and forth between the Bluetooth module and the master. Input at one device will be output at the other. Make sure you are in UART mode.

152. Adafruit Bluefruit LE UART Friend (BLE)





152.1. Specifications Adafruit Bluefruit LE UART Friend (BLE)

- ARM Cortex MO core running at 16 MHz
- 256 KB Flash memory
- 32 KB SRAM
- Transport: UART 9600 baud (CTS + RTS required)
- 5V safe
- Power supply: 3.3-16V (Onboard 3.3 V voltage regulation)
- AT commands
- MDBT40 chip
- nRF51822 CHIP
- Firmware: 0.70

Note: BLE is only compatible with devices that also have BLE. It is not compatible with normal Bluetooth 2.0

152.2. Datasheet Adafruit Bluefruit LE UART Friend (BLE)

- MDBT40 chip
  https://cdn-shop.adafruit.com/product-files/2267/MDBT40-P256R.pdf
- NRF51822 chip
  http://infocenter.nordicsemi.com/pdf/nRF51822_PS_v3.1.pdf
- Whole lot more information can be found at:
  https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/introduction

152.3. LED status

onboard red LED status	Meaning
RED: 2 blinks every 2 seconds	UART mode
RED: 3 blinks every 2 seconds	CMD = AT Command mode
Blue: ON	paired/connected

152.4. Connections Adafruit Bluefruit LE UART Friend (BLE)

Pin nr	Name	Description	Arduino pin
1	MOD	Mode HIGH => command mode LOW => UART/DATA mode	When connected, this will override the mode select switch
2	CTS	Clear to send (MCU => Bluefruit) HIGH = default LOW => send data to MCU	Any digital port or Ground if you don't need hardware flow control
3	TXO	Transmit pin out (Bluefruit => MCU)	Any digital port when using SoftwareSerial
4	RXI	Receive pin in (MCU => Bluefruit)	Any digital port when using SoftwareSerial
5	VIN	Power supply (3.3-16V)	5V
6	RTS	Request to send	Probably not needed at 9600 baud
7	GND	Ground	GND
8	DFU		

152.5. Libraries needed for Adafruit Bluefruit LE UART Friend (BLE)

- Adafruit Bluefruit nRF51 library through library manager.

Library use explanation

```
#include <SoftwareSerial.h>
```

Include the SoftwareSerial library.

```
#include "Adafruit_BLE.h"
```

Include the common Adafruit Bluetooth Low Energy library.

```
#include "Adafruit_BluefruitLE_UART.h"
```

Include the Bluefruit LE UART specific library.

```
#define BUFSIZE                         128
```

Set size of the read buffer for incoming data.

```
#define VERBOSE_MODE                    true
```

Sets Verbose mode to true makes it more noisy, but now you can see all communication between the MCU and the BLE.

```
#define BLUEFRUIT_SWUART_RXD_PIN        9
#define BLUEFRUIT_SWUART_TXD_PIN        10
#define BLUEFRUIT_UART_CTS_PIN          11
#define BLUEFRUIT_UART_RTS_PIN          -1
#define BLUEFRUIT_UART_MODE_PIN         12
```

Set the pin numbers to be used.

```
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN,
BLUEFRUIT_SWUART_RXD_PIN);
```

Set SoftwareSerial to the pins on the Bluefruit.

```
Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
```

Create an instance of the Adafruit_BluefruitLE_UART type, called 'ble', using the pins on the Bluefruit.

```
if ( !ble.begin(VERBOSE_MODE) )
```
Start the Bluefruit module and tells what VERBOSE_MODE you want to use.

```
ble.echo(false);
```
ATE=0. *Set the echo of the AT parser to off*

```
ble.info();
```
ATI. *Displays basic information about the Bluefruit.*

```
ble.verbose(false);
```
At the start of your script, it is wise to set verbose_mode to true, but as soon as the connection is stable, you can set verbose_mode to false with this command.

```
while (! ble.isConnected())
```
AT+GAPGETCONN. *Displays the current Bluetooth connection status.*

```
ble.println("AT+BLEUARTRX");
ble.readline();
```
AT+BLUEARTRX. *Dump the UART service's RX buffer and read it with ble.readline() into ble.buffer.*

```
if (strcmp(ble.buffer, "OK") == 0)
```
If there is no data in ble.buffer…

```
Serial.println(ble.buffer);
```
Print de content of ble.buffer to Serial Monitor.

A great place to learn more about the library can be found at:
https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/configuration

152.6. AT command mode Adafruit Bluefruit LE UART Friend (BLE)

All Bluetooth modules can be configured through AT commands. Before you can enter AT commands, you must first set the module in AT command mode.

Set the switch on the module to CMD

To set this specific Bluetooth module in AT command mode follow the steps below

- Beside Key, Vcc, GND, RxD and TxD , no extra connections are needed.
- Make sure the Bluetooth module is **not connected** with another device (it can be paired, but it can't be connected)!
- Communication settings: 9600,0,0 (baudrate between Arduino and Bluetooth module)
- Terminal settings: No line ending

AT commands Adafruit Bluefruit LE UART Friend (BLE)

Below is a list of some of the AT commands for this Bluetooth module.

Action	AT-command	Result	Description
test	`AT`	OK	*Test connection with the Bluetooth module*
Factory default	`AT+FACTORYRESET`	OK	*Set Bluetooth module to factory defaults: name: HMSoft baud: 9600,0,0 pin code: 000000 role: slave*
Show name	`AT+GAPDEVNAME`	Adafruit Bluefruit LE OK	*Show current name*
Set name	`AT+GAPDEVNAME=BLUEFRUIT`	OK	*Set name to keyes04*
Show role			
Set role			
Show baudrate	`AT+BAUDRATE`	9600 OK	*Current baudrate*
Set baudrate	`AT+BAUDRATE=9600`	OK	*Set baudrate to 9600*
Show pincode			
Set Pincode			
Show version	`ATI`	BLEFRIEND32 nRF51822 QFACA10 758C5C8ABBF298AD 0.8.0 0.8.0	*Shows current firmware version*

Action	AT-command	Result	Description
		Sep 25 2017 S110 8.0.0, 0.2 OK	
Reboot	`ATZ`	OK	*Reboot the module*
Show Address	`AT+BLEGETADDR`	EA:8C:98:9D:C9:C8 OK	*Show address module*

More information can be found at:
https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/standard-at

145_BTBLE_Bluefruit_AT.ino

The following sketch copies all commands typed in Serial Monitor to the Bluetooth module. All responses from the Bluetooth module are then copied to the Serial Monitor. This is a convenient way to change the Bluetooth module's settings.

- Connect CTS to D9
- Connect TxD to D10
- Connect RxD to D11
- Connect VCC to 5V
- Connect GND to GND
- Set Serial Monitor to 115200 and to " No line ending "
- Set the switch on top of the module in the CMD position.

```
#include <Arduino.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_UART.h"

#define BUFSIZE                         160
#define VERBOSE_MODE                    true //otherwise you won't see the
result of the AT commands

SoftwareSerial bluefruitSS = SoftwareSerial(10,11);
Adafruit_BluefruitLE_UART ble(bluefruitSS, -1, 9, -1);

void setup(void)
{
  delay(500);
  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit AT Command Example"));
  Serial.print(F("Initialising the Bluefruit LE module: "));
  ble.begin(VERBOSE_MODE);
  Serial.println( F("OK!") );
  ble.echo(false);
  Serial.println("Requesting Bluefruit info:");
  ble.info();
}

void loop(void)
{
  char command[BUFSIZE+1];
  getUserInput(command, BUFSIZE);
  ble.println(command);
  ble.waitForOK();
}

void getUserInput(char buffer[], uint8_t maxSize)
{
  memset(buffer, 0, maxSize);
  while( Serial.available() == 0 ) {
    delay(1);
  }
  uint8_t count=0;
  do
  {
    count += Serial.readBytes(buffer+count, maxSize);
    delay(2);
  } while( (count < maxSize) && !(Serial.available() == 0) );
}
```

152.7. UART mode Adafruit Bluefruit LE UART Friend (BLE)

UART mode is needed when you want to transfer data from slave to master and vice versa. In UART mode you can not issue AT commands. Your sketch needs to interpret the meaning of the received data. To enter UART mode, you'll just need to move the switch on top of the module in the UART position. At the master, you'll need some kind of terminal program on the paired device (Serial Monitor, PuTTY on Windows/OSx or Serial Bluetooth Terminal on Android smartphone) and make sure you select the paired Bluetooth module as your port (COMx, /dev/cu.hc-06-DevB, hc-06).

153_BTBLE_Bluefruit_HelloWorld.ino

This sketch sends the famous line "Hello World" to the master device.

- Connect CTS to D9
- Connect TxD to D10
- Connect RxD to D11
- Connect VCC to 5V
- Connect GND to GND
- Set Serial Monitor to 115200 and to " No line ending "
- Set the switch on top of the module in the UART position.

```
#include <Arduino.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_UART.h"

#define BUFSIZE                         128
#define VERBOSE_MODE                    true

SoftwareSerial bluefruitSS = SoftwareSerial(10,11);
Adafruit_BluefruitLE_UART ble(bluefruitSS, -1, 9, -1);

void setup(void)
{
  delay(500);
  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Command <-> Data Mode Example"));
  Serial.print(F("Initialising the Bluefruit LE module: "));
  ble.begin(VERBOSE_MODE);
  Serial.println( F("OK!") );
  ble.echo(false);
  Serial.println("Requesting Bluefruit info:");
  ble.info();
  ble.verbose(false);  // debug info is a little annoying after this point!
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);
}

void loop(void)
{
  ble.println("Hello World");
  delay(500);
}
```

151_BTBLE_Bluefruit_TwoWay.ino

The following sample can be used to send data back and forth between the Bluetooth module and the master. Input from one device will be output at the other.

- Connect CTS to D9
- Connect TxD to D10
- Connect RxD to D11
- Connect VCC to 5V
- Connect GND to GND
- Set Serial Monitor to 115200 and to " No line ending "
- Set the switch on top of the module in the UART position.

```
#include <Arduino.h>
#include "Adafruit_BLE.h"
#include "Adafruit_BluefruitLE_UART.h"

#define BUFSIZE                         128
#define VERBOSE_MODE                    true

SoftwareSerial bluefruitSS = SoftwareSerial(10,11);
Adafruit_BluefruitLE_UART ble(bluefruitSS, -1, 9, -1);

void setup(void)
{
  delay(500);
  Serial.begin(115200);
  Serial.println(F("Adafruit Bluefruit Command <-> Data Mode Example"));
  Serial.print(F("Initialising the Bluefruit LE module: "));
  ble.begin(VERBOSE_MODE);
  Serial.println( F("OK!") );
  ble.echo(false);
  Serial.println("Requesting Bluefruit info:");
  ble.info();
  ble.verbose(false);  // debug info is a little annoying after this point!
  Serial.println( F("Switching to DATA mode!") );
  ble.setMode(BLUEFRUIT_MODE_DATA);
}

void loop(void)
{
  ble.println("Hello World");
  delay(500);
}
```

152.8. Adafruit Bluefruit LE Connect on smartphone

Adafruit developed the "Adafruit Bluefruit LE Connect" so you can use you smartphone or tablet to control your Arduino. This app is available for both Android and IOS.

Android	IOS
https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect&hl=en	https://itunes.apple.com/us/app/adafruit-bluefruit-le-connect/id830125974?mt=8

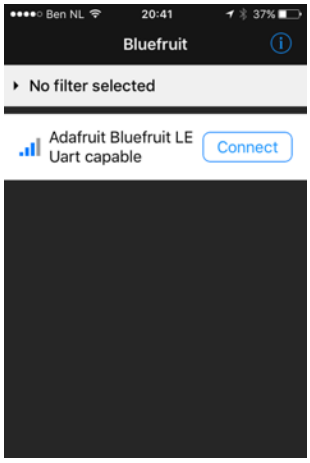
You can either scan the QR code for your Smartphone or use the URL.

151_BTBLE_Bluefruit_TwoWay.ino

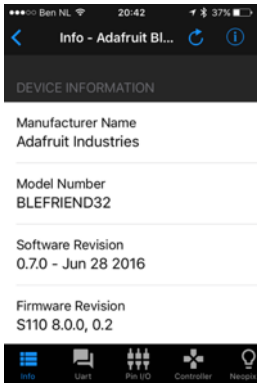
The following sample can be used to communicate with the Adafruit Bluefruit LE Connect smartphone app.

- Connect CTS to D9
- Connect TxD to D10
- Connect RxD to D11
- Connect VCC to 5V
- Connect GND to GND
- Set Serial Monitor to 115200 and to " No line ending "
- Set the switch on top of the module in the UART position.

- Open the Adafruit Bluefruit LE Connect app on your smartphone or tablet. Your Bluefruit should be detected and shown as can be seen on the next screen:

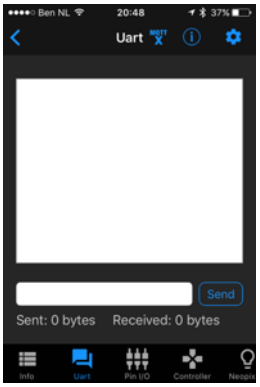


- Click on your Adafruit Bluefruit LE, your device will start connecting . As soon as the connection is established, the blue connection LED on the Bluefruit will turn on and the following screen will be visible on your smartphone or tablet.

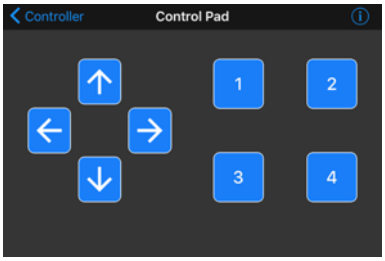


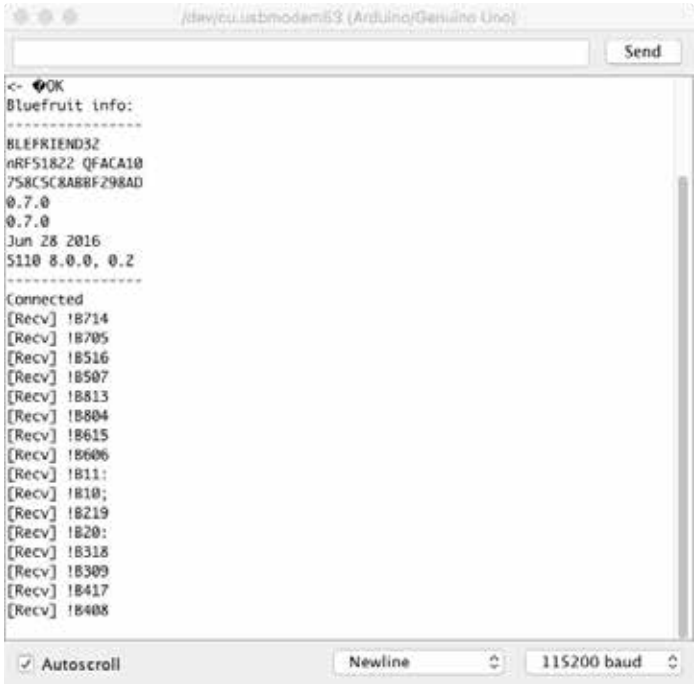
- On the Serial Monitor your sketch will also state that you are connected.

- By choosing the UART tab on your smart device, you can type a text and send it to the Bluefruit.



- In the controller tab, you can open the control pad. Every button in the control pad can send two different strings. One string for pressing the button and another one for releasing the button.





R/C transmitters & receivers

In this chapter some R/C transmitters and receivers are described.

153. R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)

153.1. Specifications R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)

Transmitter Attack T2DR:

- 2 channels AM transmitter
- Frequency: 40 MHz
- Modulation: AM
- Power: 12V (8xAA)
- Current drain: 250mA

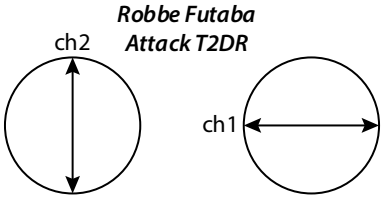
Receiver FP-R122JE:

- 2 channels AM receiver
- Frequency: 40 MHz
- Intermediate frequency: 455 kHz
- Power requirement: 4.8-8.4V
- Current drain: 30mA (at 4.8V / No Signal)
- Dimensions: 47.2x33.3x17..3mm
- Weight: 16.6g

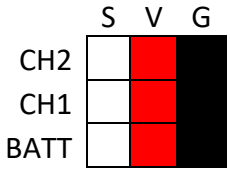
153.2. Datasheet R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)

Manual: https://fccid.io/AZPT2DR-75A/User-Manual/User-Manual-827719

153.3. Channel mapping 2ch transmitter T2DR



153.4. Connections 2ch receiver FF-R122JE (Rx)



CH2..CH2

Pin nr	Name	Description	Arduino pin
1	Signal	CH2..CH1 signal	n.c.
2	VCC	-	n.c.
3	GND	-	n.c.

BATT

Pin nr	Name	Description	Arduino pin
1	n.c.	-	n.c.
2	VCC	Power Supply	5V
3	GND	Ground	GND

153.5. Libraries needed for R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)

There are no libraries needed.

153.6. Sample R/C: 2ch Robbe Futaba Attack T2DR (Tx) + FF-R122JE (Rx)

The following sketch displays the received values of the transmitter.

Sample Connections from the 2 ch FP-122JE receiver to an Arduino

- Connect VCC to 5V
- Connect GND to GND
- Connect CH1 signal to D3
- Connect CH2 signal to D4

090_RCAttack_T2DR.ino

```
const int ch1_pin=3;
const int ch2_pin=4;

int ch1_value;
int ch2_value;

void setup()
{
  Serial.begin(115200);
  pinMode(ch1_pin, INPUT);
  pinMode(ch2_pin,INPUT);
}

void loop()
{
  ch1_value = pulseIn (ch1_pin,HIGH);
  Serial.print ("Ch1:");
  Serial.print (ch1_value);
  Serial.print (" | ");

  ch2_value = pulseIn (ch2_pin,HIGH);
  Serial.print ("Ch2:");
  Serial.print (ch2_value);
  Serial.println ();
}
```

154. R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)

154.1. Specifications R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)

Transmitter Attack T2DR:

- 7(/8) channels FM transmitter
- Frequency: 40 MHz
- Intermediate frequency: 455 kHz
- Channel interval: 10 kHz
- Modulation: FM (PPM)
- Power: 9.6V NC battery

Receiver FP-R122JE:

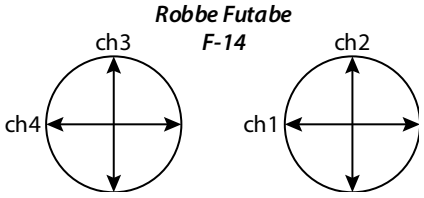
- 8 channels FM receiver
- Frequency: 40 MHz
- Power supply: 4.8-6V
- Weight: 35g
- Dimensions: 60x36.5x20.5mm

154.2. Datasheet R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)

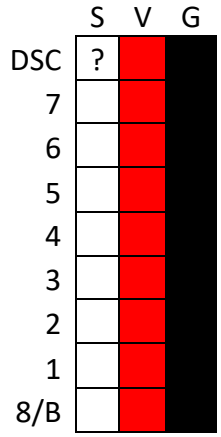
Manual:

- https://www.gebruikershandleiding.com/Robbe-Futaba-F-14/preview-handleiding-757655.html (Dutch)
- http://www.ripmax.de/media/download//f/1/f14__f14_navy_-_gb_print.pdf (slightly different model, but useful)

154.3. Channel mapping 7 (8ch) transmitter F-14



154.4. Connections receiver 8ch FP-R118F (Rx)



DSC

Pin nr	Name	Description	Arduino pin
1	Signal	?	?
2	VCC	?	?
3	GND	?	?

CH7..CH1

Pin nr	Name	Description	Arduino pin
1	Signal	CH2..CH1 signal	n.c.
2	VCC	-	n.c.
3	GND	-	n.c.

8/B

Pin nr	Name	Description	Arduino pin
1	n.c.	-	n.c. ?
2	VCC	Power Supply	5V
3	GND	Ground	GND

154.5. Libraries needed for R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)

There are no libraries needed.

154.6. Sample R/C: 7 (8ch) Robbe Futaba F-14 (Tx) + FP-R118F (Rx)

The following sketch displays the received values of the transmitter.

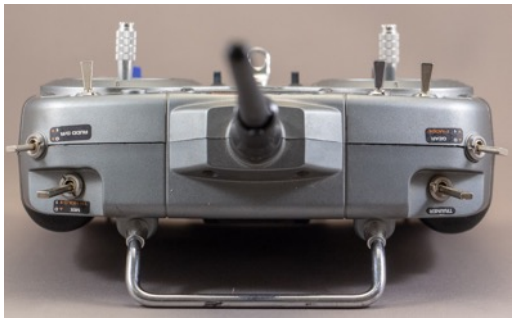
Sample Connections from the 8 ch FP-R118F receiver to an Arduino

- Connect VCC to 5V
- Connect GND to GND
- Connect CH1 signal to D3
- Connect CH2 signal to D4
- Connect CH3 signal to D5
- Connect CH4 signal to D6
- Connect CH5 signal to D7
- Connect CH6 signal to D8
- Connect CH7 signal to D9

093_RC_FP-R118F.ino

```
const int CH1_pin=3;
const int CH2_pin=4;
const int CH3_pin=5;
const int CH4_pin=6;
const int CH5_pin=7;
const int CH6_pin=8;
const int CH7_pin=9;

void setup()
{
  Serial.begin(115200);
  pinMode(CH1_pin, INPUT);
  pinMode(CH2_pin, INPUT);
  pinMode(CH3_pin, INPUT);
  pinMode(CH4_pin, INPUT);
  pinMode(CH5_pin, INPUT);
  pinMode(CH6_pin, INPUT);
  pinMode(CH7_pin, INPUT);
}

void read_value(String channel_name, int channel_pin)
{
  int channel_value=pulseIn(channel_pin,HIGH);
  Serial.print(channel_name);
  Serial.print(":");
  Serial.print (channel_value);
  Serial.print (" | ");
}

void loop()
{
  read_value("CH1", CH1_pin);
  read_value("CH2", CH2_pin);
  read_value("CH3", CH3_pin);
  read_value("CH4", CH4_pin);
  read_value("CH5", CH5_pin);
  read_value("CH6", CH6_pin);
  read_value("CH7", CH7_pin);
  Serial.println();
}
```

155. R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)



155.1. Specifications R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)

Transmitter Dx6i:

- 6 channels
- Frequency: 2.4 GHz
- Modulation: DSM2
- Power: 4,8-6V (4xAA NimH/Alkaline)
- Programming features: Helicopter & Airplane
- Model memory: 10
- Modes: Mode 2

Receiver AR6000e:

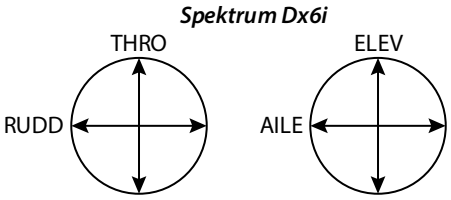
- 8 channels
- Frequency: 2.4 GHz
- Modulation: DSM2
- Voltage range: 3.5-9.6V
- Dimensions: 40x19x9mm
- Weight: 4.4g

155.2. Datasheet R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)

Manual:

- Dx6i: https://www.spektrumrc.com/ProdInfo/Files/SPM6600-Manual_DX6i.pdf
- AR6100e: https://www.spektrumrc.com/ProdInfo/Files/SPMAR6100.pdf

155.3. Channel mapping 6ch transmitter Dx6i



155.4. Connections 6ch receiver AR6100e

AUX1, GEAR, RUDD, ELEC, AILE, THRO

Pin nr	Name	Description	Arduino pin
1	Signal	AUX1, GEAR, RUDD, ELEC, AILE & THRO	Any Digital pin
2	VCC	-	n.c.
3	GND	-	n.c.

BATT

Pin nr	Name	Description	Arduino pin
1	n.c.	-	n.c.
2	VCC	Power Supply	5V
3	GND	Ground	GND

155.5. Libraries needed for R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)

There are no libraries needed.

155.6. Binding AR6000e receiver to Spektrum Dx6i transmitter

A few seconds after turning on the Transmitter, an amber LED on the receiver should lit. If this is not the case, you should probably rebind the receiver to the transmitter, use the following procedure if this is the case:

- Place the binding plug on the battery port of the receiver (this will pull signal to ground)
- On one of the other ports: connect 5V tot + and GND to -, the amber LED should now flash
- On your Spektrum Dx6i transmitter, place the left stick in the lowest position

Hold the trainer switch and turn on the transmitter. The receiver should now respond with a lower flashing frequency. Remove the binding plug. You receiver is now bound to the transmitter.

155.7. Sample R/C: 6ch Spektrum Dx6i (Tx) + AR6100e (Rx)

The following sketch displays the received values of the transmitter.

Sample Connections from the 6ch AR6100e receiver to an Arduino

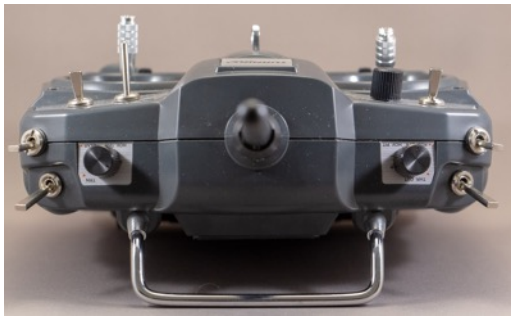
- Connect VCC to 5V
- Connect GND to GND
- Connect AUX1 signal to D3
- Connect GEAR signal to D4
- Connect RUDD signal to D5
- Connect ELEV signal to D6
- Connect AILE signal to D7
- Connect THRO signal to D8

091_RC_ar6100e.ino

```
const int AUX1_pin=3;
const int GEAR_pin=4;
const int RUDD_pin=5;
const int ELEV_pin=6;
const int AILE_pin=7;
const int THRO_pin=8;

void setup()
{
  Serial.begin(115200);
  pinMode(AUX1_pin, INPUT);
  pinMode(GEAR_pin,INPUT);
  pinMode(RUDD_pin, INPUT);
  pinMode(ELEV_pin,INPUT);
  pinMode(AILE_pin, INPUT);
  pinMode(THRO_pin,INPUT);
}

void read_value(String channel_name, int channel_pin)
{
  int channel_value=pulseIn(channel_pin,HIGH);
  Serial.print(channel_name);
  Serial.print(":");
  Serial.print (channel_value);
  Serial.print ("/t");
```

```
}

void loop()
{
  read_value("AUX1", AUX1_pin);
  read_value("GEAR", GEAR_pin);
  read_value("RUDD", RUDD_pin);
  read_value("ELEV", ELEV_pin);
  read_value("AILE", AILE_pin);
  read_value("THRO", THRO_pin);
  Serial.println();
}
```

156. R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)



156.1. Specifications R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)

Transmitter TGY 9x:

- 6 channels
- Frequency: 2.4 GHz
- Modulation: DSM2

Receiver 9x8Cv2:

- 8 channels
- Frequency: 2.4 GHz
- Antenna type: Dipole single wire (recommended orientation: vertical)
- Voltage range: 4..5-6.5V
- Dimensions: 52x35x15mm
- Weight: 18g

156.2. Datasheet R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)

Manual:

- TGY 9x: https://app.box.com/s/cb3b695b7552a051d351

156.3. Channel mapping 9ch transmitter TGY 9x

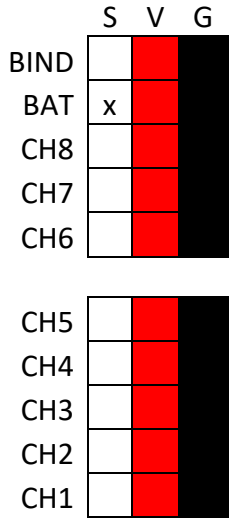
The radio seems like a Mode 2 radio, so the left stick is Rudder and Throttle and the right stick is Ailerons and Elevator. The mapping to the channels can be changed in the settings. By default, the sticks are mapped to the first 4 channels and the other channels are free. With the free available firmware OpenTX[1] and ER9x you can map any of the sticks to any channel (physical 1..9 and virtual 10..16). With the original Turnigy firmware not all switches are available to use and you can only map to the physical channels 1..9. Working with the original receiver 9x8Cv2 only channels 1..8 can be received.

In the table below you can see the labels on the 9x and the names used in the three different firmwares.

Label on 9x	Type	Turnigy	OpenTX	ER9x
Rud	L Stick horizontal	Rud	Rud	Rud
Thr	L Stick vertical	Thr	Thr	Thr
Ail	R Stick horizontal	Ail	Ail	Ail
Ele	R Stick vertical	Ele	Ele	Ele
RUD D/R	2 position switch	n.a.	RUD	sRUD
THR CUT	2 position switch	THRO HOLD	THR	sTHR
HOV PIT	Potentiometer	HOV PIT	P1	P1
ELE D/R	2 position switch	n.a.	ELE	sELE
PIT TRIM	Potentiometer	PIT TRIM	P3	P3
HOV THR (AUX 2)	Potentiometer	HOV THRO	P2	P2
TRN	Momentary switch	n.a.	TRN	sTRN
GEAR	2 position switch	GEAR	GEAR	sGEAR
F.MODE (AUX 3)	3 position switch	n.a.	3POS	sIDx
AIL D/R	2 position switch	n.a.	AIL	sAIL

[1] The MCU used in this radio is made by Atmel, so it can be programmed using ICSP through an USBasp adapter. This way you can read or write both the EEProm (for settings) and the Firmware. For this, I've used OpenTX Companion with the 2.0 branch, using a newer version then 2.0.20 could ruin the content of your EEPROM (ie. Your settings).

156.4. Connections 9ch receiver 9x8Cv2



BIND

Pin nr	Name	Description	Arduino pin
1	n.c.	?	n.c. ?
2	VCC	?	?
3	GND	?	n.c.

BATT

Pin nr	Name	Description	Arduino pin
1	Signal	-	n.c.
2	VCC	Power Supply	5V
3	GND	Ground	GND

CH8..CH1

Pin nr	Name	Description	Arduino pin
1	Signal	CH8..CH1 signal	Any digital port
2	VCC	-	n.c.
3	GND	-	n.c.

156.5. Libraries needed for R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)

There are no libraries needed.

156.6. Sample R/C: 9ch Turnigy TGY 9x (Tx) + 9x8Cv2 (Rx)

The following sketch displays the received values of the transmitter and maps them to a value between -99 and +100.

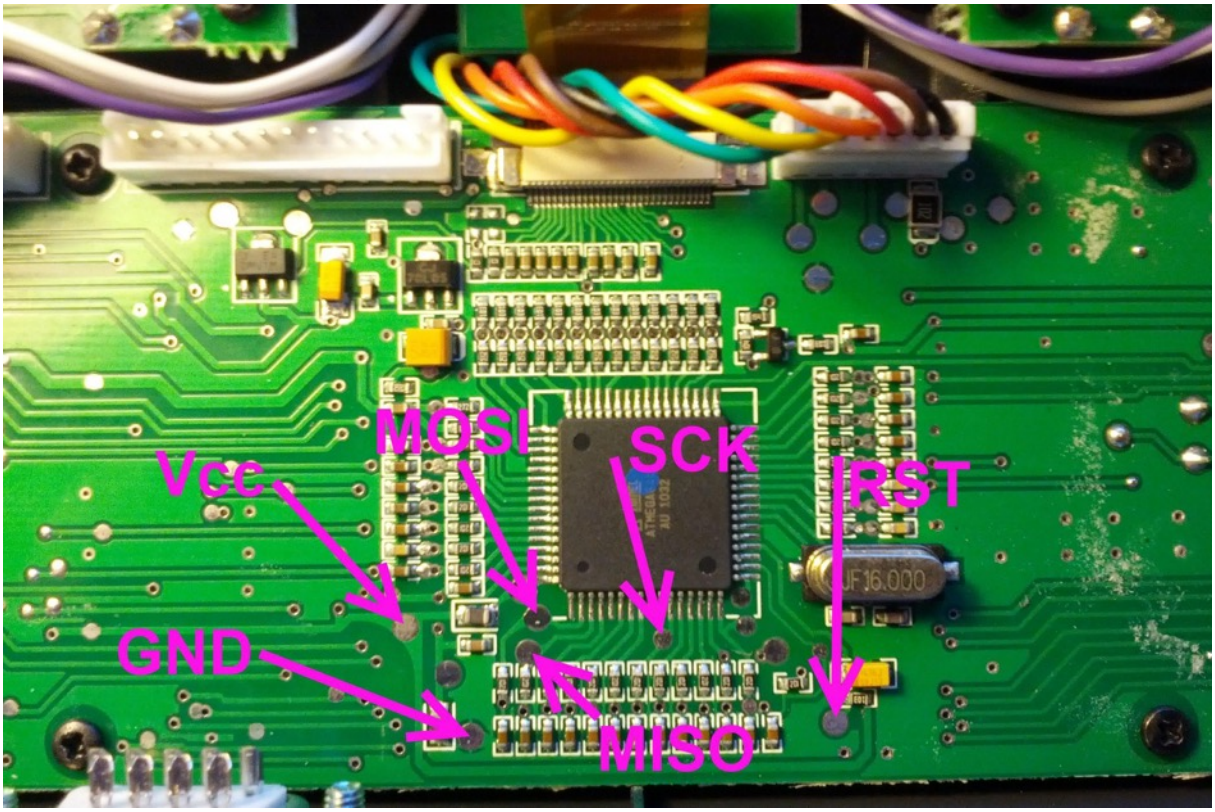
Sample Connections from the 8ch 9x8Cv2 receiver to an Arduino

- Connect VCC to 5V
- Connect GND to GND
- Connect CH1..CH8 signal to D3..D10

092_RC_9x8Cv2.ino

```
const int CH1_pin = 2;
const int CH2_pin = 3;
const int CH3_pin = 4;
const int CH4_pin = 5;
const int CH5_pin = 6;
const int CH6_pin = 7;
const int CH7_pin = 8;
const int CH8_pin = 9;

void setup()
{
  Serial.begin(115200);
  pinMode(CH1_pin, INPUT);
  pinMode(CH2_pin, INPUT);
  pinMode(CH3_pin, INPUT);
  pinMode(CH4_pin, INPUT);
  pinMode(CH5_pin, INPUT);
  pinMode(CH6_pin, INPUT);
  pinMode(CH7_pin, INPUT);
  pinMode(CH8_pin, INPUT);
}

void read_value(String channel_name, int channel_pin)
{
  //Raw data
  //int channel_value = pulseIn(channel_pin, LOW);
  //TGY 9x ER9x & OpenTx firmware
  int channel_value = map(pulseIn(channel_pin, LOW), 17713, 18730, 99, -
99);

  Serial.print(channel_name);
  Serial.print(":");
  Serial.print (channel_value);
  Serial.print ("\t");
}

void loop()
{
  read_value("CH1", CH1_pin);
  read_value("CH2", CH2_pin);
  read_value("CH3", CH3_pin);
  read_value("CH4", CH4_pin);
  read_value("CH5", CH5_pin);
  read_value("CH6", CH6_pin);
  read_value("CH7", CH7_pin);
  read_value("CH8", CH8_pin);
  //read_value("CH9", CH9_pin);
  Serial.println();
}
```

157. R/C: 9ch Turnigy TGY 9x (Tx) firmware upgrade



As described in the previous chapter, it is possible to read or write both the settings (EEPROM) and the firmware. To accomplish this, you'll either need to do some soldering and us an USBasp adapter, or buy a programming board for the 9x.

- Solder 6 wires to solder pads on the motherboard and connect those to an USBasp adapter. Offcourse this ruins your warenty and is totally on your own risk. It was the way I chose.
- Order a programmer board and mount this on top of the 9x main board. This is the most safe and convenient wayand is a very convenient way to add programming capabilities to the 9x radio. You can buy one at the following webshop:
  http://smartieparts.com/shop/index.php?main_page=product_info&cPath=3&products_id=383
  I haven't tested this, but is seems very straight forward. Open your radio, mount the programmer board on top of the mainboard and connect an USB cable.

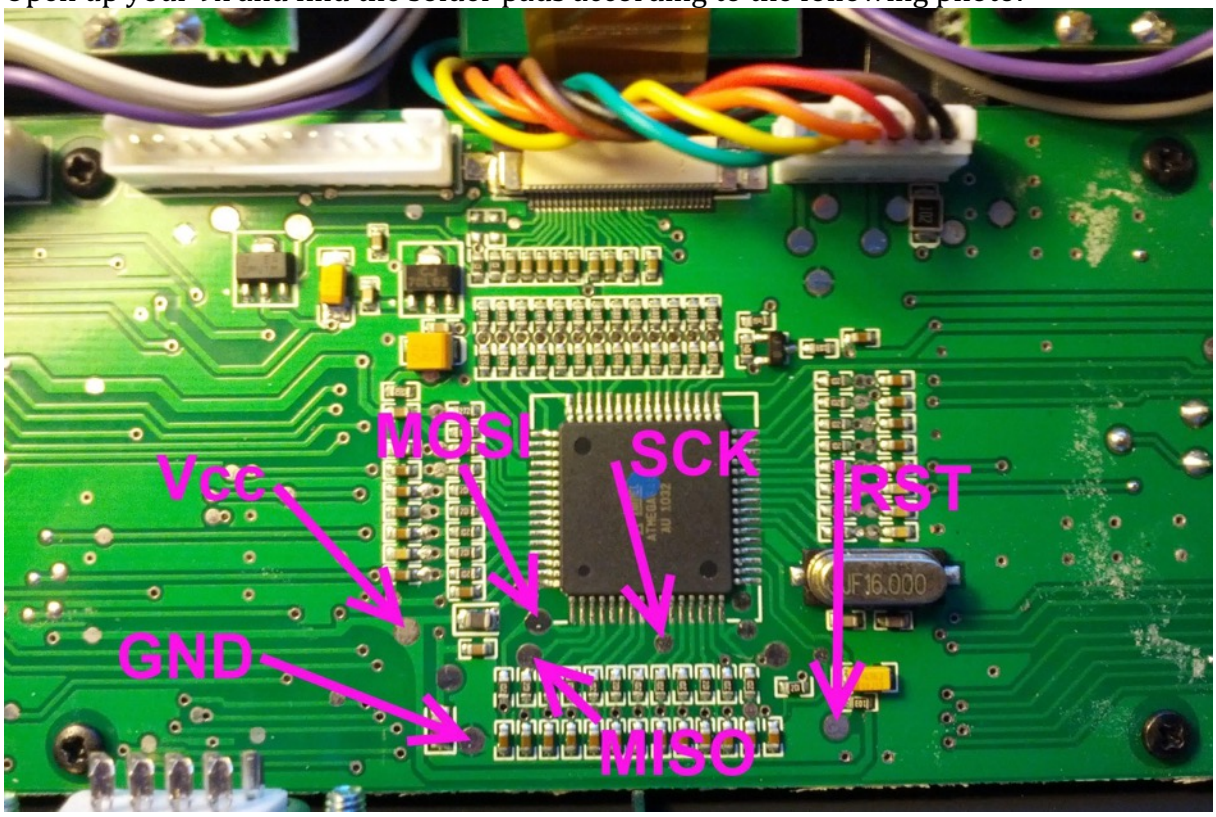
157.1. Preparing your 9x (the solder method)

Bill of materials:

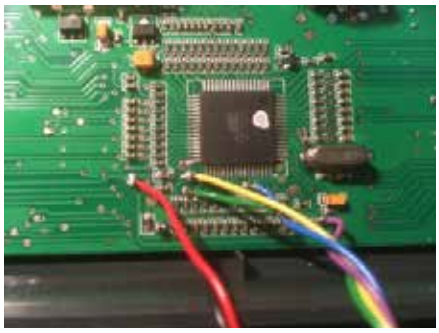
- Six pieces of isolated wire
- Solder
- Soldering iron
- An ISP male connector or 2 strips or 5 male header pins (0.1 inch pitch)
- USBasp adapter (this is described in "31 USBasp v2.0 programmer")
- Flatcable with two female 2x5 (0.1 inch pitch) connector (came with my USBasp adapter).

For this procedure you'll need to do some soldering, but only basic soldering skills are needed, since you only need to solder to solder pads. There is no SMD soldering needed!

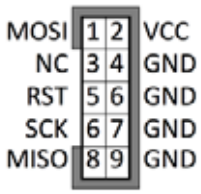
1. Decide where you want to place the ISP connector. Some have placed this connector on the inside, but then you'll need to open the radio every time you want to flash another firmware, but als when you want to save your settings. I used a dremmle to make a hole in the backpanel, to place my 2 rows of 5 male header pins.
2. Remove the batteries from the radio.
3. Open up your 9x and find the solder pads according to the following photo:



4. Remove the large connector that connects your motherboard to the radio module in the back panel.
5. Solder short piece of isolated wire to all these 6 solderpads. I've used Red for VCC, Black for GND, Yellow for MOSI, Green for MISO, Blue for SCK and Purple for RST.



6. (Hot) glue your ISP male header in place and solder the wires to the correct pins (below is connector side of the ISP header, not the solder side!).



7. Carefully close your radio.
8. Reinstall the batteries to your radio (not needed for flashing).

157.2. OpenTX

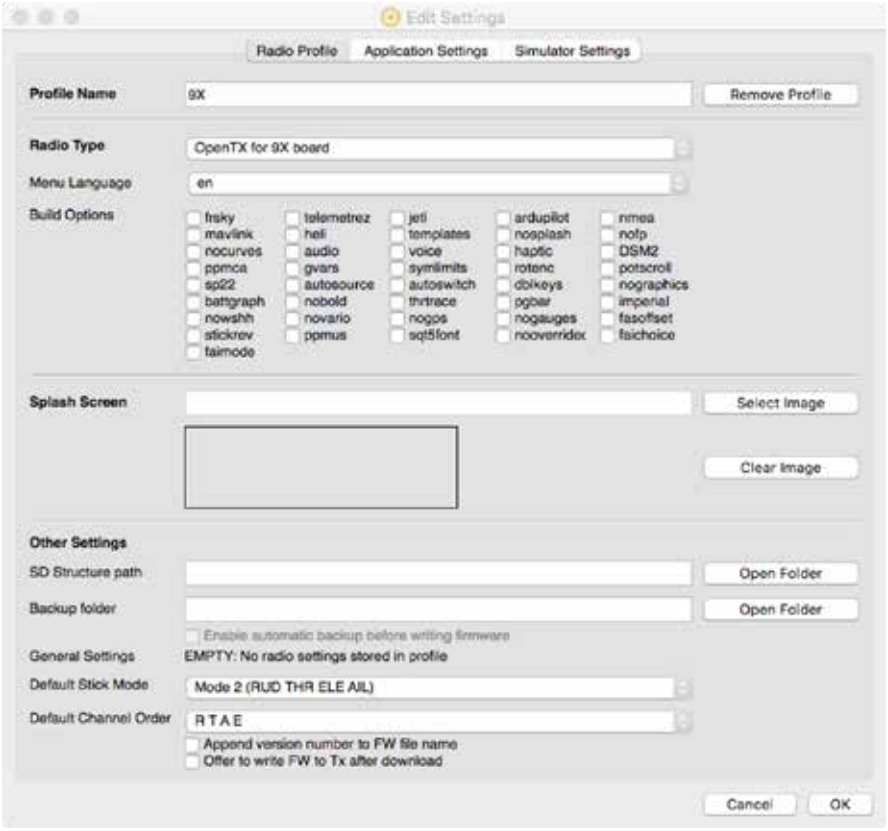
In this paragraph I'll describe how to install the needed software, backup and flashing settings & firmware.

Installing OpenTX companion, avrdude and USBasp

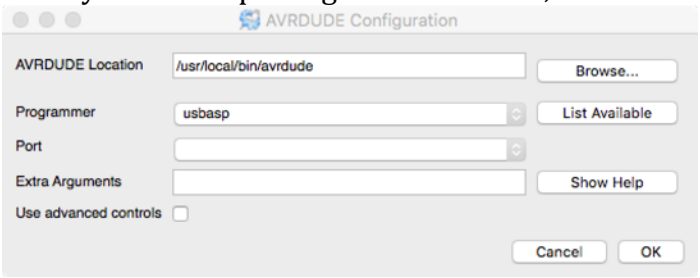
1. Download the OpenTX companion 2.0 branch software (don't use anything newer then 2.0.20!
   a. Version 2.0.20. Both Windows and Linux users can get their software at the following link: https://www.open-tx.org/2016/09/15/opentx-2.0.20
   b. Version 2.0.19 is the latest 2.0 branch MAC software available, but is not listed at the open-tx.org website. Changing the 2.0.20 in the url above to 2.0.19 was an easy fix for this: https://www.open-tx.org/2016/01/15/opentx-2.0.19
2. Install avrdude software
   a. OSx (I first installed brew and then used brew to install avrdude)

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/ins
tall)"
brew install avrdude --with-usb
```

   b. Windows. The procedure for Windows is described at (not tested by me): http://www.ladyada.net/learn/avr/setup-win.html
3. Install drivers for the the USBasp (not needed for Linux and OSx). The installation for Windows is not tested by me, but the link leads to the website of the manufacturers of the original USBasp adapters. https://www.fischl.de/usbasp/
4. Start OpenTX for the first time and DON'T downloaded the suggested firmware, since this is not the firmware for the 9x!
5. Go to SETTINGS, RADIO PROFILE, ADD RADIO PROFILE.
6. Go to:
   a. Windows: SETTINGS, SETTINGS
   b. OSx: COMPANION, PREFERENCES



7. Name your radio profile and choose the “OpenTX for 9x board”.
8. I selected the Mode 2 (RUD, THR, ELE, AIL) default stick mode according to my radio.
9. I also chose the R T A E as my default Channel Order. So, Rudder is channel 1, Throttle is channel 2, Ailerons channel 3 en Elevator is channel 4.
10. I don’t understand anything about the build profiles, so I didn’t select anything there. I accepted the above with the OK button.
11. If Companion now asks you to download the firmware, please confirm it now. Otherwise go to FILE, DOWNLOAD and press DOWNLOAD FW.
12. After downloading click on OK.
13. The size of this OpenTX firmware (.hex file) is about 138 KB.
14. Place your USBasp and go to: SETTINGS, CONFIGURE COMMUNICATIONS



Check the above (default settings)

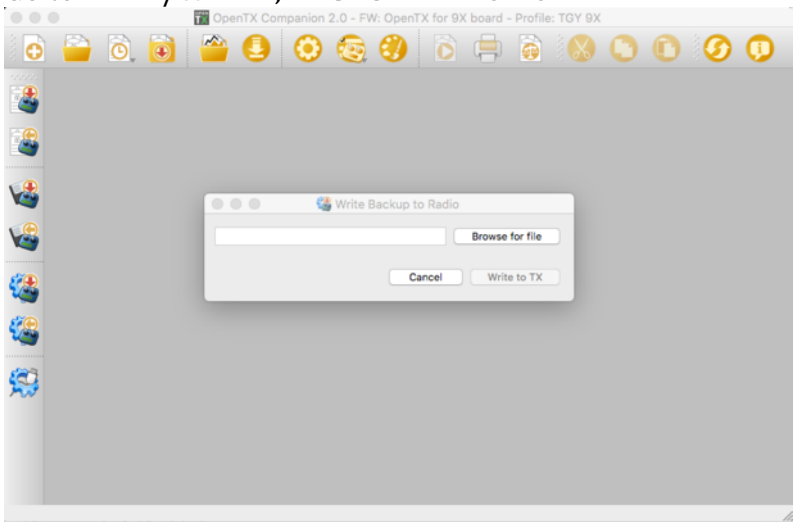
15. The software is now installed.

Backup radio settings to file

With this steps you can backup the settings of your radio. OpenTX Companion can backup its own settings, but it can also backup and restore the settings from the original firmware and from the ER9x firmware.

This procedure can be used to save your current settings, whatever firmware your radio is flashed with. It's nice to have a backup of both the original settings and firmware.

1. Turn off your Radio (or remove the batteries), power to the MCU will be provided by your USBasp!
2. Place your USBasp.
3. Go to READ/WRITE, BACKUP RADIO TO FILE



4. Save it to the location of your choice.
5. The size backup file (.bin) is about 2KB
6. Remove the USBasp and reinsert it again (*I had some problems with my USBasp, but solved it by reinserting the USBasp after every read or write action.)*

Read Firmware from radio

With this steps you can backup the firmware of your radio. OpenTX Companion can backup its own firmware, but it can also backup other firmware, like the original firmware and the ER9x firmware.

This procedure can be used to save your current firmware, whatever firmware your radio is flashed with. It's nice to have a backup of both the original settings and firmware.

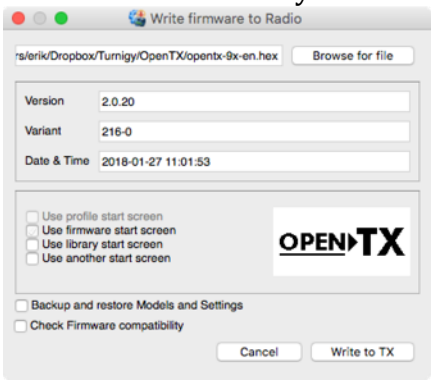
1. Turn off your Radio (or remove the batteries), power to the MCU will be provided by your USBasp!
2. Place your USBasp.
3. Go to READ/WRITE, READ FIRMWARE FROM RADIO
4. Save it to the location of your choice.
5. The size of the firmware (.bin) is about 60-64 KB
6. Remove the USBasp and reinsert it again (*I had some problems with my USBasp, but solved it by reinserting the USBasp after every read or write action.)*

Write Firmware to Radio

With this steps you can write/restore firmware to your radio. OpenTX can write its own firmware, but it can alse write other firmware, like the original firmware and the ER9x firmware.

1. Turn off your Radio (or remove the batteries), power to the MCU will be provided by your USBasp!
2. Place your USBasp.
3. Go to READ/WRITE, WRITE FIRMWARE TO RADIO.

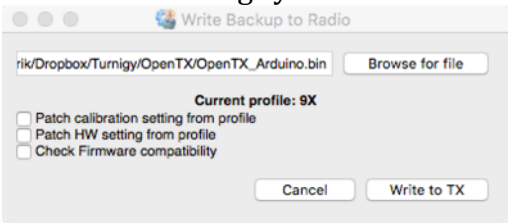
4. Choose the firmware you would like to flash to your radio (use Browse).



5. Press WRITE TO TX and keep your fingers crossed (the first time is scary!).
6. Turn on your radio.
7. If the settings in your radio don't match with this new firmware, both OpenTX and ER9x will format the EEPROM at first start. The original firmware doesn't, so before you start the orginal firmware, you should first restore original settings before your start your radio the first time.

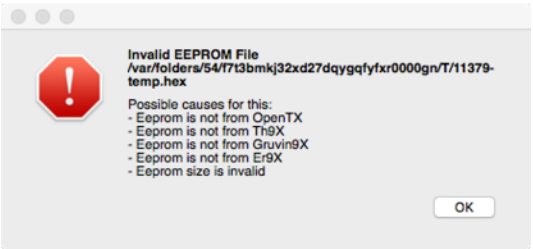
Write Backup to Radio

1. Turn off your Radio (or remove the batteries), power to the MCU will be provided by your USBasp!
2. Place your USBasp.
3. Go to READ/WRITE, WRITE BACKUP TO RADIO
4. Choose the settings you would like to flash to your radio (use Browse for file).



5. Press WRITE TO TX.

6. Press YES



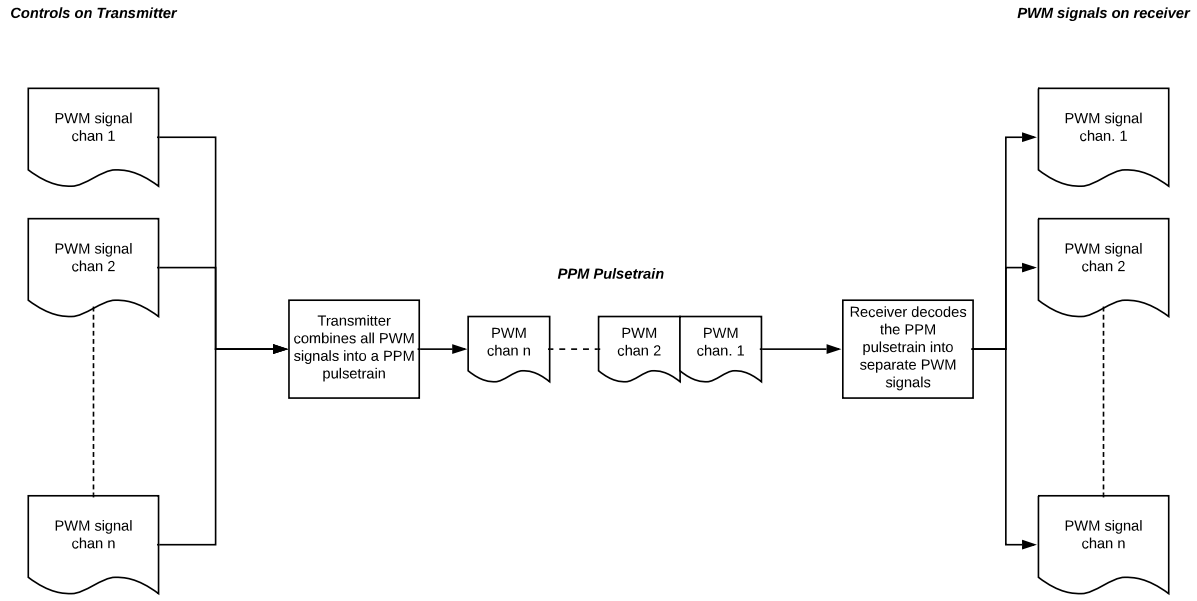
157.3. ER9x

In this paragraph I'll describe how to install the needed software, backup and flashing settings & firmware.

158. Decoding a PPM pulse train to separate channel signals

In the previous chapters every channel on the receiver has been connected to a separate pin on the Arduino. By code you can then decode the PWM signals by using the pulsein() command.

Whereas the Transmitter combines the PWM signals of all the channels and then strings it into a PPM pulse train. The pulse train is then transmitted to the receiver. The receiver then decodes the PPM signal back to the separate channel PWM signals.



In this chapter I will describe how to use this PPM signal and decode it with an Arduino. This way, You'll only need 1 digital pin in total, instead of 1 single digital pin per channel!

158.1. Finding the pin that holds the pulse train

Before you can start coding, you'll need to find out how to connect the Arduino with the PPM signal.

If your receiver has a specialized PPM output, then you are lucky and use the signal pin of this PPM output to your Arduino. But in most cases it's not clear where the PPM output can be obtained (if ever). From all the receivers described in this document, I was only able to use the Robbe Futaba FP-R11F 40 MHz FM receiver with the F-14 as transmitter. Futaba uses the BU4015B dual 4-bit static shift register in a lot of their receivers (the old ones?). If your receiver uses this chip, then the PPM signal can be found on pin 1 and on pin 9 (clock), so you'll just need to solder a wire to this chip and connect that wire to your Arduino. Below, you'll find pictures of my Futaba FP-R118F receiver with the extra wires. White for Signal (pin 1 on the BU4015B), red for VCC and black for GND. On the outside, I've placed a connector which I can use to connect with an Arduino.

158.2. Background information

- More information can be found at:
  - https://diydrones.com/profiles/blogs/705844:BlogPost:38393
- The datasheet for the BU4015B can be found at:
  - https://api.ning.com/files/YFV2*duuIj34M6bSbdvwqmj5Rl0158VZr**SA97mPhDFcumDUtyPluTTbYISfHE7uWQuASCELegj5FVivdDX99LS5jHfvl2D/BU4015.pdf

158.3. Sample sketch to decode the PPM signal with Arduino

The following sketch decodes the separate channel signals from the PPM signal.

Sample Connections

- Connect VCC to 5V
- Connect GND to GND
- Connect PPM signal to D3

094_RC_FP-R118F_PPM.ino

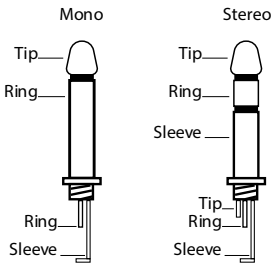
```
const int PPM_pin = 3;
#define channels 7

void setup()
{
  Serial.begin(115200);
  pinMode(PPM_pin, INPUT);
}

void loop()
{
  int channel_nr;
  int channel_value;
  while (pulseIn(PPM_pin, LOW) < 5000)
  {
    //Wait for the beginning of the frame
  }
  for (channel_nr = 0; channel_nr <= channels - 1; channel_nr++)
  {
    channel_value = pulseIn(PPM_pin, LOW);
    Serial.print("CH" + String(channel_nr));
    Serial.print(":");
    Serial.print (channel_value);
    Serial.print (" | ");
  }
  Serial.println("");
}
```

159. Decoding a PPM pulse train from a trainer connector

My Spektrum Dx6i has a Trainer output at the back of the transmitter. The signal on this output is a PPM pulse train, so you are able to decode this PPM pulse train signal into its original channel PWM signals. In this chapter, I'll describe to setup for my Turnigy TGY 9x transmitters and my Spektrum Dx9i transmitter.

159.1. Trainer cable

On both transmitters, the Trainer output is a 3,5mm female mini jack (mono). The same dimensions as most smartphones have. So you'll need to solder a two wire cable to a 3,5mm Male mini jack mono:



Mono mini-jack

Pin	Description	Arduino pin
Tip	Signal (normally HIGH)	Any digital pin (or interrupt pin if interrupt is needed)
Ring	Ground	GND

Stereo mini-jack

Pin	Description	Arduino pin
Tip	Signal (normally HIGH)	Any digital pin (or interrupt pin if interrupt is needed)
Ring	Ground	GND
Sleeve	Ground	GND

159.2. Spektrum Dx9i transmitter

- Turn your Transmitter of.
- Insert the jack plug in the Trainer port in the back. At this point, your transmitter will turn on automatically (with the Power switch in the OFF position).
- Connect the Tip to the correct digital port.
- Connect the ring (and sleeve when using a stereo jack) to GND
- Start you Arduino.

159.3. Turnigy TGY 9x

Using this Trainer cable on the TGY is less straight-forward then with the Dx9i. Before you connect the Trainer cable, you'll first have to disable the Tx module on the back.

- Turn your Transmitter of.
- Remove the Tx module on the back. It will stay attached to the antenna cable, so this is not very convenient. Their seems to be hacks for this:
    - By moving the complete antenna onto the Tx module. So when you remove the Tx-Module, you'll also remove the antenna. Do this on your own risk!
    - By altering the PCB inside the Transmitter. Do this on your own risk!
- Insert the jack plug in the Trainer port in the back. At this point, your transmitter will turn on automatically (with the Power switch in the OFF position).
- Connect the Tip to the correct digital port.
- Connect the ring (and sleeve when using a stereo jack) to GND
- Start you Arduino.

159.4. Sample sketch to decode the PPM signal with Arduino without interrupts

The following sketch decodes the separate channel signals from the PPM signal without using interrupts. The sketch is based on the previous chapter, but in this case you'll need to trigger on a HIGH flank instead of a low flank (`pulseIn(PPM_pin, HIGH)` instead `pulseIn(PPM_pin, LOW)`).

Sample Connections

- Connect VCC to 5V
- Connect GND to GND
- Connect PPM signal to D2 (the sketch does not use the interrupt capability of this pin)

128_TrainerPPM_NoIRQ.ino

```
const int PPM_pin = 2;
#define channels 7


void setup()
{
  Serial.begin(115200);
  pinMode(PPM_pin, INPUT);
}

void loop()
{
  int channel_nr;
  int channel_value;
  while (pulseIn(PPM_pin, HIGH) < 5000)
  {
    //Wait for the beginning of the frame
  }
  for (channel_nr = 0; channel_nr <= channels - 1; channel_nr++)
  {
    channel_value = pulseIn(PPM_pin, HIGH);
    Serial.print("CH" + String(channel_nr));
    Serial.print(":");
    Serial.print (channel_value);
    Serial.print (" | ");
  }
  Serial.println("");
}
```

159.5. Sample sketch to decode the PPM signal with Arduino using interrupts

The following sketch (a copy from the previous chapter) decodes the separate channel signals from the PPM signal while using interrupts. It is based on the RCTrainer library from mikem that can be found at:

http://www.airspayce.com/mikem/arduino/RcTrainer/RcTrainer-1.0.zip

More information can be found at:

http://www.airspayce.com/mikem/arduino/RcTrainer/index.html

Only 8 channels can be decode with this library.

Sample Connections

- Connect VCC to 5V
- Connect GND to GND
- Connect PPM signal to D2 (this sketch is based on using interrupt on D2)

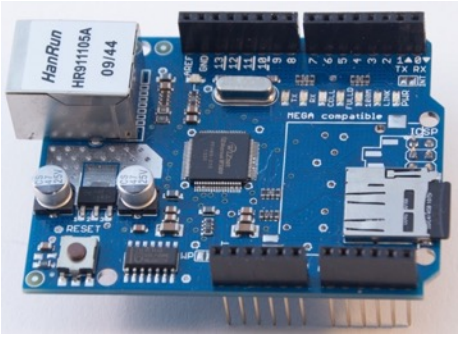
129_RC_TrainerPPM_IRQ.ino

```
#include <RcTrainer.h>
#define channels 6

RcTrainer tx;

void setup()
{
    Serial.begin(115200);
}

void loop()
{
  int channel_nr;
  int channel_value;
  for (channel_nr = 0; channel_nr < channels; channel_nr++)
  {
    channel_value = tx.getChannel(channel_nr);
    Serial.print("CH" + String(channel_nr));
    Serial.print(":");
    Serial.print (channel_value);
    Serial.print (" | ");
  }
  Serial.println("");
}
```

Shields

Shields are PCB's that can be placed directly on top of an Arduino UNO, so they are very easy to connect.

160. Ethershield



160.1. Specifications Ethershield

- Wiznet W5100 chip
- On-board micro-SD card slot

160.2. Datasheet Ethershield

- Wiznet W5100: http://www.wiznet.co.kr/UpLoad_Files/ReferenceFiles/W5100_Datasheet_v1.2.2.pdf
- Power-Over-Ethernet: http://arduino.cc/en/uploads/Main/PoE-datasheet.pdf

160.3. Connections Ethershield

Name	Description	Arduino pin
SCK	SPI Serial Clock	D13
MISO	SPI Master In Slave Out	D12
MOSI	SPI Master Out Slave In	D11
SS Ethernet	SPI Slave Select Ethernet	D11
SS SD card	SPI Slave Select SD card	D4

160.4. Libraries needed for Ethershield

- The Serial Peripheral Interface library included in the Arduino IDE.
- The Ethernet library from Arduino through the Library Manager.
- Secure Digital card library through Library Manager. (only needed when using the SD card).

Library use explanation

```
#include <SPI.h>
```

Include the Serial Peripheral Interface library included in the Arduino IDE.

```
#include <Ethernet.h>
```

Include the Ethernet library included with the Arduino IDE.

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
```

Array to hold the Ethershield MAC address.

```
EthernetServer server(80);
```

Website at port 80.

```
Ethernet.begin(mac);
```
Start Ethershield connection with DHCP enabled.

```
server.begin();
```
Start webserver.

```
Ethernet.localIP()
```
Variable holding the assigned IP address.

```
EthernetClient client = server.available();
```
Create "client" a new instance of the class EthernetClient.

```
client
```
Boolean, indicating if the Ethernet client is ready.

```
client.connected()
```
Boolean, indicating whether or not the client is connected. Note that a client is considered connected if the connection has been closed but there is still unread data.

```
client.available()
```
Returns the number of bytes available for reading (that is, the amount of data that has been written to the client by the server it is connected to).

```
char c = client.read();
```
Request from client.

```
client.println("<html>");
```
Output HTML code to client (in this case opening tag <HTML>).

```
client.stop();
```
Stop connection with client.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

160.5. Sample Ethershield

The following sketch starts a webserver that shows the value of the 6 analog ports in HTML.

Sample Connections

- Shield on top of Arduino UNO.

102_Ethershield.ino

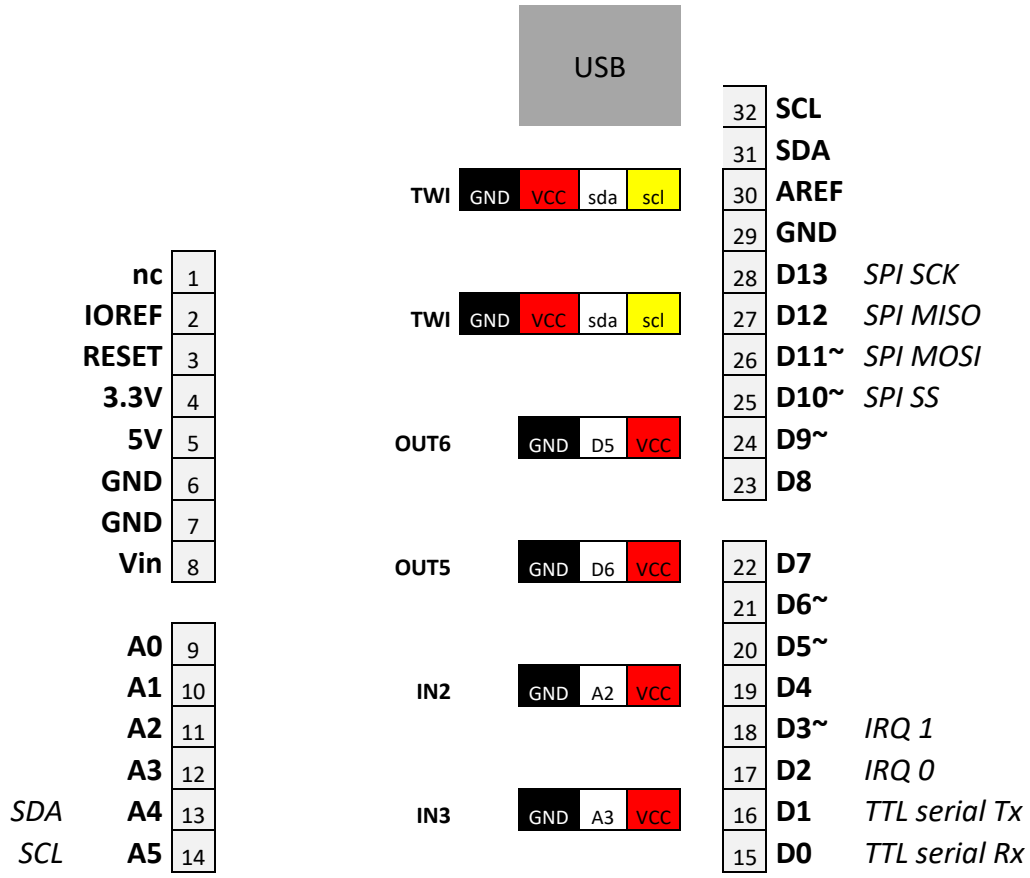
```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

EthernetServer server(80);

void setup()
{
  Serial.begin(9600);
  Ethernet.begin(mac);
  server.begin();
  Serial.print("server is at ");
  Serial.println(Ethernet.localIP());
}


void loop()
{
  EthernetClient client = server.available();
  if (client)
  {
    Serial.println("new client");
    boolean currentLineIsBlank = true;
    while (client.connected())
    {
      if (client.available())
      {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank)
        {
          client.println("<html>");
          for (int analogChannel = 0; analogChannel < 6; analogChannel++)
          {
            int sensorReading = analogRead(analogChannel);
            client.print("analog input ");
            client.print(analogChannel);
            client.print(" is ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n')
        {
          currentLineIsBlank = true;
        }
        else if (c != '\r')
        {
          currentLineIsBlank = false;
        }
```

```
      }
    }
    delay(1);
    client.stop();
    Serial.println("client disconnected");
  }
}
```

161. USB Host shield

This shield adds USB host functionality to Arduino. Through this shield, all sorts of USB devices can communicate with your sketches.

161.1. Layout/connections Arduino UNO R3



OUT5, OUT6, IN2 & IN3 are only pin-compatible with TinkerKit sensors. The non-TinkerKit 3-pin servo cables for example are not compatible, because those cables will have VCC in the middle!

161.2. Specifications USB Host shield

- 2 tinkerkit TWI =I2C connectors (4 pins)
- 2 tinkerkit Inputs: IN2 = A2 & IN3 = A3
- 2 TinkerKit Outputs: OUT5 = D5 & OUT6 = D6 (3 pins), both are PWM

- Operating voltage: 5V
- USB controller: MAX3421E
- Max current: 500 mA when powered through the Arduino power jack with a suitable power supply.
- Max current: 400 mA when powered through the USB port
- Support for the following USB device classes:
    - HID devices: keyboards, mice, joysticks, etc.
    - Game controllers: Sony PS3, Nintendo Wii, Xbox360
    - USB to serial converters: FTDI, PL2303, ACM, as well as certain cell phones and GPS receivers
    - ADK-capable Android phones and tablets
    - Mass storage devices: USB sticks, memory card readers, external HDD's, etc.
    - Bluetooth dongles.

161.3. Datasheet USB Host shield

- Datasheet for the MAX3421E: https://www.maximintegrated.com/en/products/interface/controllers-expanders/MAX3421E.html
- More information about the Arduino USB Host Shield can be found at: https://www.arduino.cc/en/Main/ArduinoUSBHostShield

161.4. Connections Xx

4-pin TinkerKit TWI (=I2C) connector.

Pin nr	Name	Description
1	GND	Ground
2	VCC	VCC
3	SDA	TWI/I2C SDA
4	SCL	TWI/I2C SCL

3-pin TinkerKit IN2/IN3 connector.

Pin nr	Name	Description
1	GND	Ground
2	A2/A3	Analog INPUT
3	VCC	VCC

IN2 & IN3 are only pin-compatible with TinkerKit sensors. The popular 3-pin servo cables for example are not compatible, because those cables will have VCC in the middle!

3-pin TinkerKit OUT5/OUT6 connector.

Pin nr	Name	Description
1	GND	Ground
2	D5/D6	Digital IN/OUT PWM
3	VCC	VCC

OUT5 & OUT6 are only pin-compatible with TinkerKit sensors. The popular 3-pin servo cables for example are not compatible, because those cables will have VCC in the middle!

161.5. Libraries needed for USB Host shield

- USB Host Library for Arduino by Oleg Mazurov and Alexei Gluschenko from circuits@home:
  https://github.com/felis/USB_Host_Shield_2.0

Library use explanation

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

161.6. Sample USB Host shield

The following sketch will send keystrokes from a keyboard connected to the USB Host Shield to the serial monitor.

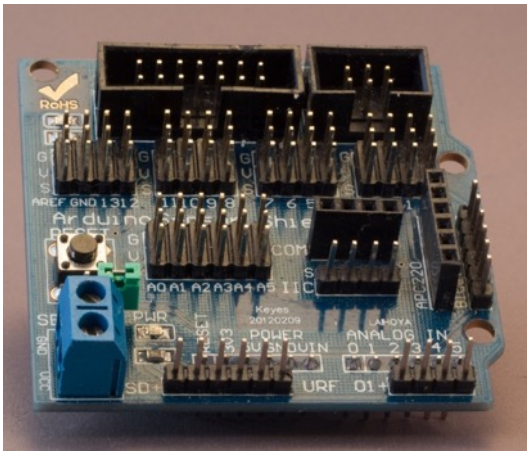
Sample Connections

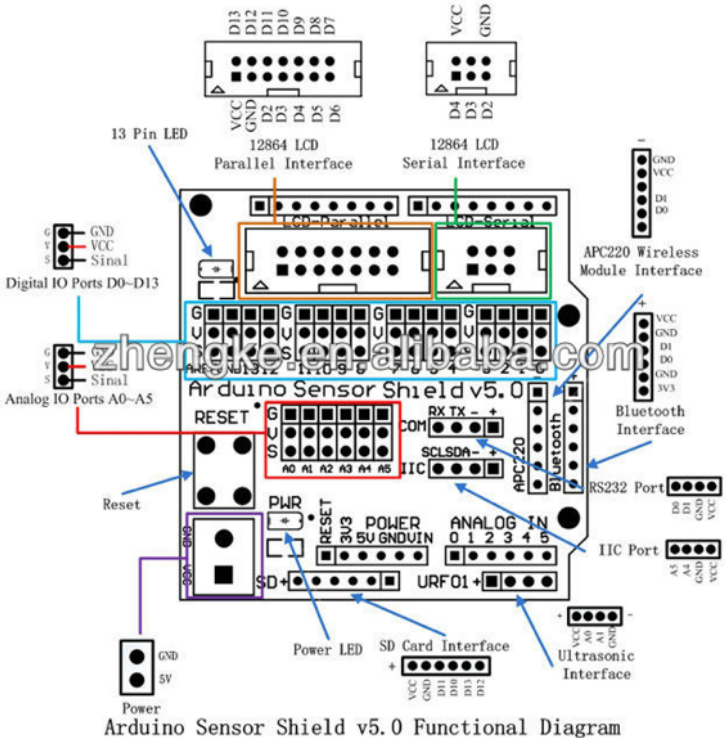
- Place USB Host Shield on top of Arduino.
- Connect USB Keyboard (without built-in Hub) to the USB Host Shield.

103_USB-shield.ino

```
#include <hidboot.h>

class KbdRptParser : public KeyboardReportParser
{
  protected:
    void OnKeyDown  (uint8_t mod, uint8_t key);
};

void KbdRptParser::OnKeyDown(uint8_t mod, uint8_t key)
{
  uint8_t c = OemToAscii(mod, key);
  if (c)
  Serial.print((char)c);
}

USB     Usb;
HIDBoot<USB_HID_PROTOCOL_KEYBOARD>    HidKeyboard(&Usb);
KbdRptParser Prs;

void setup()
{
  Serial.begin( 115200 );
  Serial.println("Start");
  if (Usb.Init() == -1)
    Serial.println("OSC did not start.");
  delay( 200 );
  HidKeyboard.SetReportParser(0, (HIDReportParser*)&Prs);
}

void loop()
{
  Usb.Task();
}
```

162. Arduino Sensor Shield v5.0

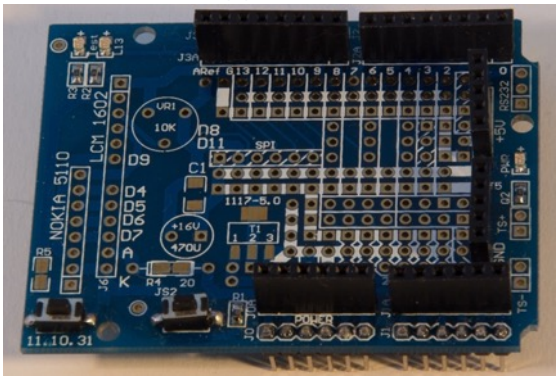


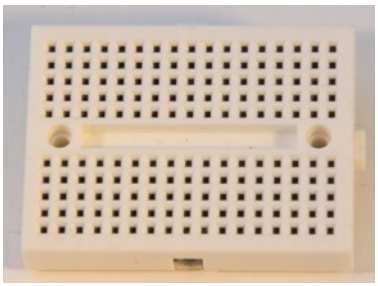


Arduino Sensor Shield v5.0 Functional Diagram

This shield can be placed on top of an Arduino Uno, so you can easily remove your project from your Arduino. Every digital port has been laid out as a 3 wire interface (Ground, VCC and Signal), so it's easy to connect several sensors by just connecting their 3 wire connector to any digital port.

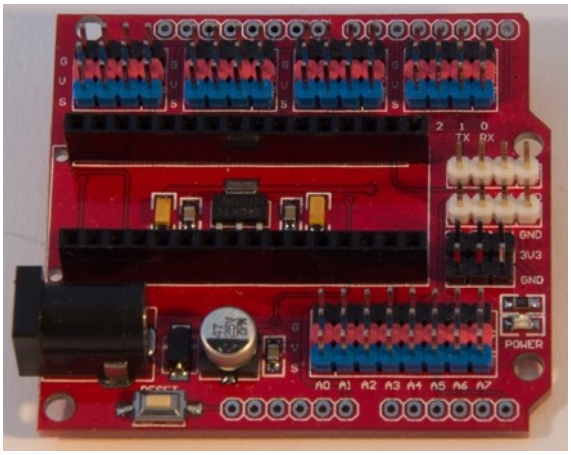
163. Proto type shield





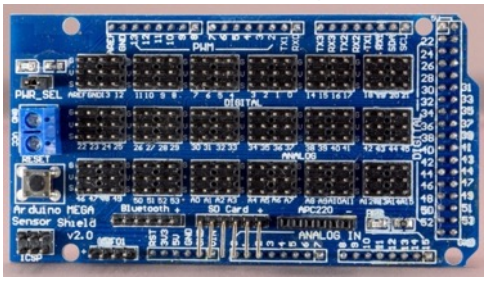
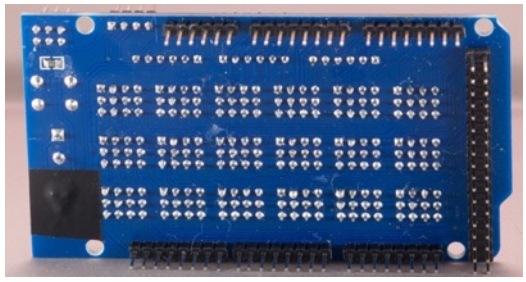
This shield can be placed on top of an Arduino Uno, so you can easily remove your project from your Arduino. You can either connect your components to the female headers, to a tiny solder less breadboard that fits in between the female headers, or you can (more permanently) solder your components directly on the Proto type shield.

164. Nano sensor shield

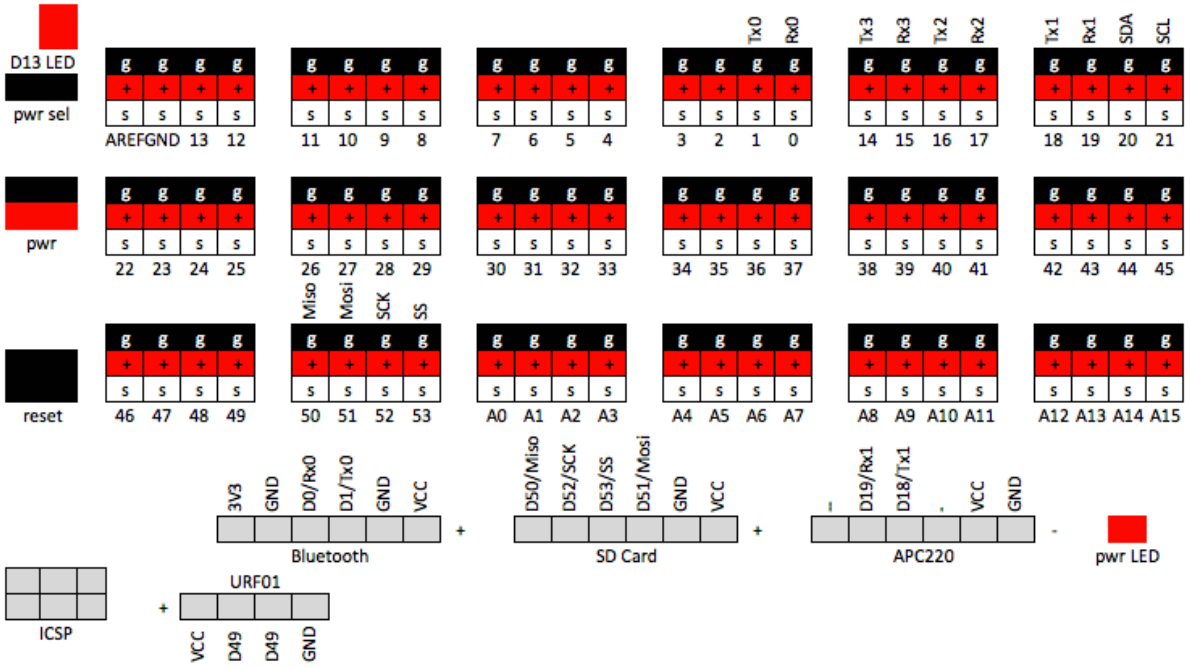


This shield is designed for the Arduino Nano. You can place the Nano on top of the shield. Every digital port has been laid out as a 3 wire interface (Ground, VCC and Signal), so it's easy to connect several sensors by just connecting their 3 wire connector to any digital port.

165. Arduino Mega Sensor Shield v 2.0

This Arduino Mega Sensor Shield, makes it easy to connect 3-lead sensor cables, like they use on servo's. All 54 digital ports and all 16 analog ports are redirected to 3 pins male connectors. By removing the Power Select jumper (pwr sel) you can use an external power supply and disconnect the 5V power from Arduino to the + connectors at the digital ports. The analog ports are always connected to the 5V power from the Arduino!
You can connect an external power supply (5V) on the power terminals. There is now voltage regulator on the shield, so make sure this external power supply delivers a stable regulated 5V.



166. Adafruit 2.8" TFT Resistive Touch Shield v2

This 2.8” resistive touch TFT display is constructed as a shield to be placed on top of a Arduino Uno or MEGA (some soldering is required to use this on a MEGA).

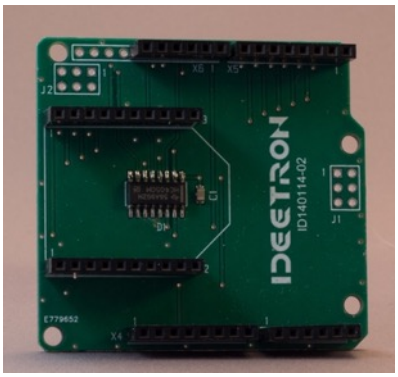
This shield is described in section LCD/TFT Displays in chapter 60 “Adafruit 2.8" TFT Resistive Touch Shield v2”.

167. 2.4” TFT LCD Shield Touch Board

This 2.4” resistive touch TFT display is constructed as a shield and can be placed on top of an Arduino UNO and probably also on a MEGA (some alteration might be necessary in the SWFT library).

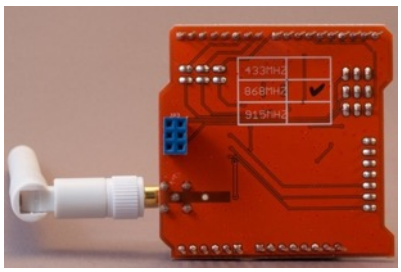
This shield is described in section LCD/TFT Displays in chapter 61 “2.4” TFT LCD Shield Touch Board".

168. LoRa: Shield for HopeRF board



This shield is described in section LoRa Module in chapter "279 LoRa: Shield for HopeRF board”.

169. LoRa: Dragino-shield

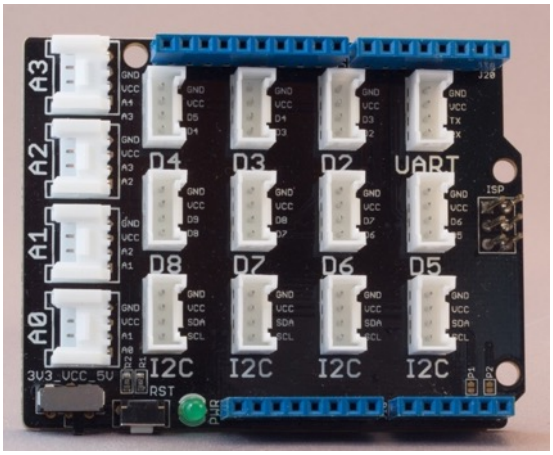


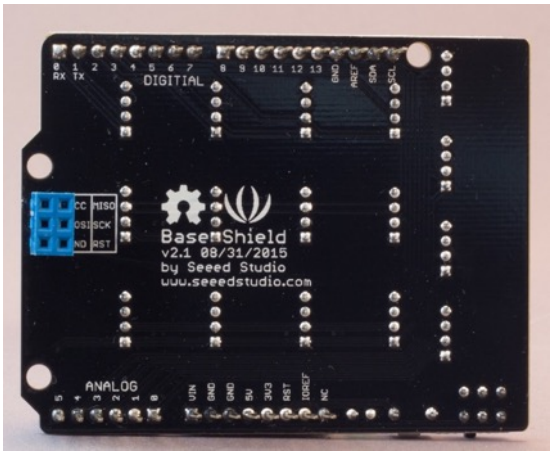
This shield is described in section LoRa Module in chapter "280 LoRa: Dragino-shield”.

Grove

Grove is a modular system from Seeed Studio. It's based around a Shield for Arduino or a HAT for Raspberry Pi and about 150 sensors which can be connected through the proprietary Grove connector. Every sensor can be connected through a 4 wire cable, either on the analog or digital ports or on the I2C port on the Grove Shield on an Arduino or the Grove HAT on a Raspberry Pi. By altering the proprietary Grove cable you can even connect the Grove with your Arduino or Pi without the Shield or HAT.

170. Grove Base Shield for Arduino





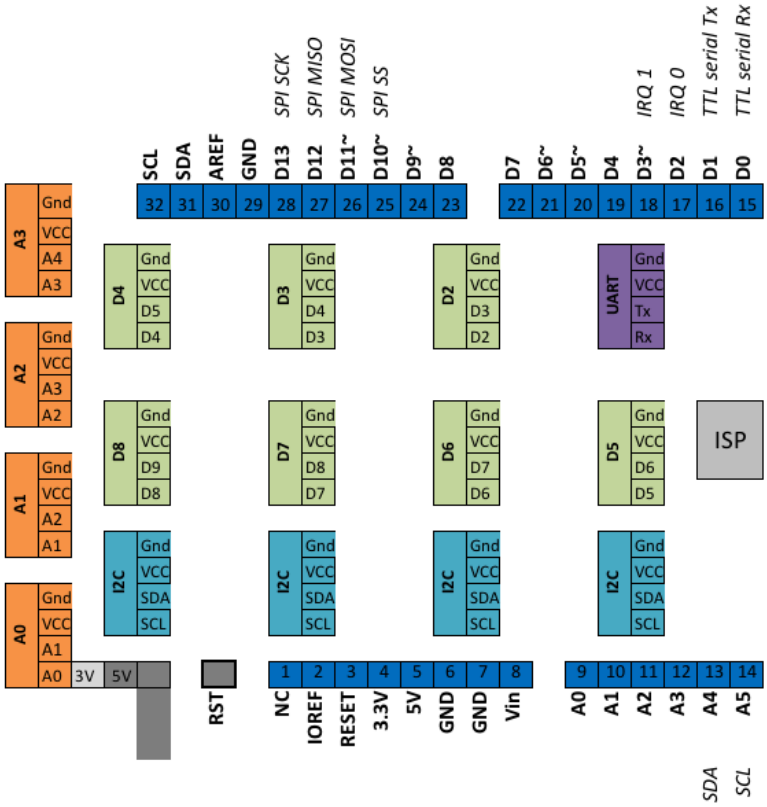
170.1. Specifications Grove Base Shield for Arduino

- 1 Serial port (D0/D1)
- 7 Digital Port (D2 .. D8)
- 4 Analog Ports (A0 .. A4)
- 4 I2C connectors (SCL and SDA)
- Power LED
- Reset Switch
- 3,3V - 5V switch
- ISP connector
- All connectors of the UNO headers are available on the header pins on the shield.

170.2. Datasheet Grove Base Shield for Arduino

- http://wiki.seeed.cc/Base_Shield_V2/

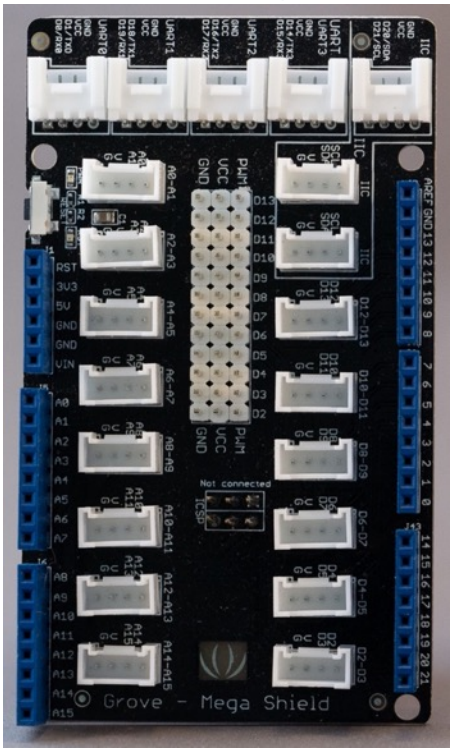
170.3. Connections Grove Base Shield for Arduino

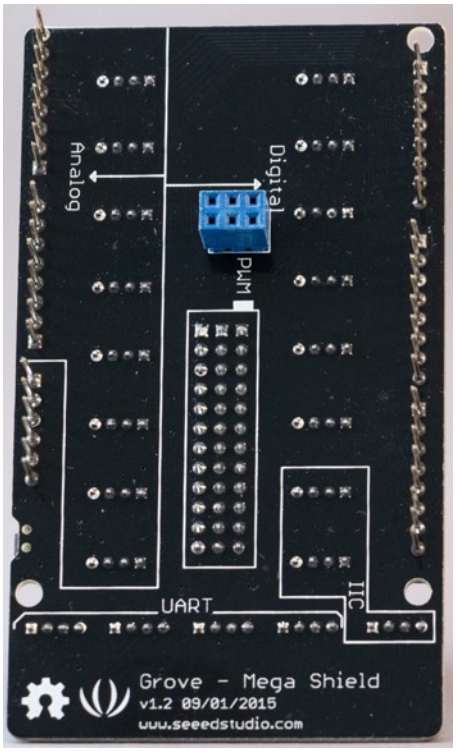


170.4. Libraries needed for Grove Base Shield for Arduino

Some Grove modules need one or more libraries, this is documented in their corresponding chapters. There are no specific libraries needed for the Grove Base itself.

171. Grove Mega Shield v1.2





171.1. Specifications Grove Mega Shield v1.2

- 4 Serial ports (UART0..UART3)
- 6 Digital Port (D2 .. D12)
- 8 Analog Ports (A0 .. A14)
- 3 I2C connectors (SCL and SDA)
- 12 PWM Servo connectors
- Power LED
- Reset Switch
- 3,3V - 5V switch
- ISP connector
- All connectors of the Mega headers are available. either on the shield or on the Mega itself.

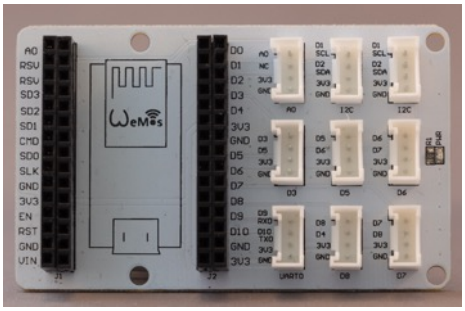
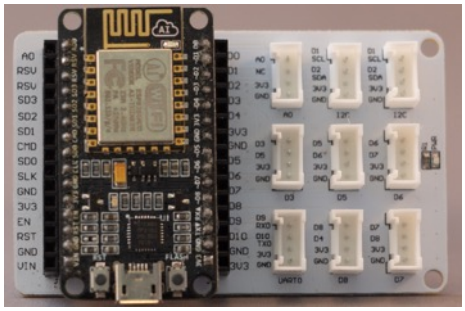
171.2. Datasheet Grove Mega Shield v1.2

- http://wiki.seeedstudio.com/Grove-Mega_Shield/

171.3. Libraries needed for Grove Base Shield for Arduino

Some Grove modules need one or more libraries, this is documented in their corresponding chapters. There are no specific libraries needed for the Grove Base itself.

172. Grove Shield NodeMCU

This shield allows you to connect a NodeMCU (ESP8266) with the modular Grove system from Seeed Studio. The shield was sold as a WeMos, but I think WeMos is one of the firms that build NodeMCU's.

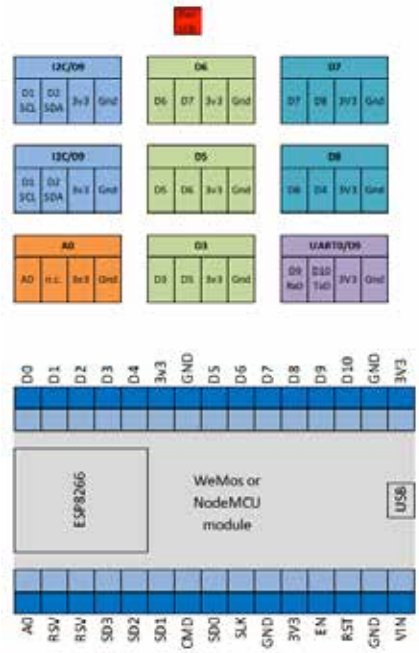
172.1. Specifications Grove Shield NodeMCU

- 1 Serial port UART0 (Rx0/D9 & Tx0/D10)
- 5 Digital ports (D3, D5, D6, D7 and D8)
- 1 Ananlog port (A0)
- 2 I2C connectors (SCL/D1 & SDA/D2)
- Double row of Female headers for all NodeMCU pins
- Power LED

172.2. Datasheet Grove Shield NodeMCU

- A simular shield from Seeed Studio: https://github.com/SeeedDocument/Grove_Base_Shield_for_NodeMCU_V1.0/raw/master/res/Grove_Base_Shield_for_NodeMCU_pdf_v1.0.rar

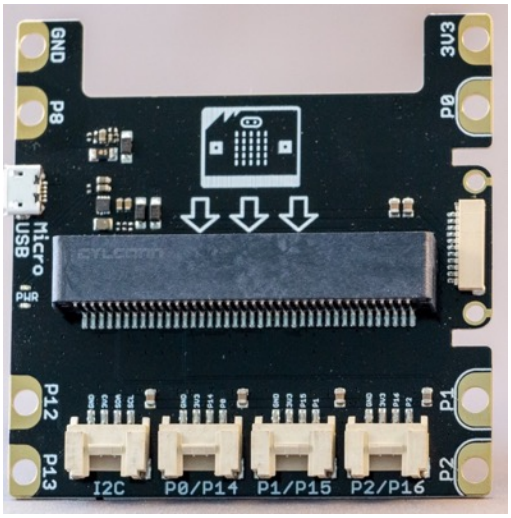
172.3. Connections Grove Shield NodeMCU

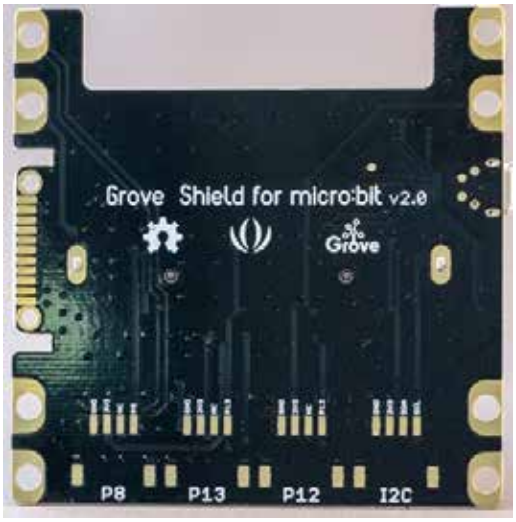


172.4. Libraries needed for Grove Shield NodeMCU

Some Grove modules need one or more libraries, this is documented in their corresponding chapters. There are no specific libraries needed for the Grove Base itself.

173. Grove Shield for micro:bit v2.0





173.1. Specifications Grove Shield for micro:bit v2.0

- Micro USB connector for an external power supply, for current hungry projects
- 4 grove connectors
  - 1x I2C
  - 3x Digital input (P0/P14, P1/P15, P2/P16
- 8x edge ring connectors (for banana plug and crocodile clips)
  - GND
  - 3V3
  - P0, P1, P2, P8, P12 & P13
- 1x Grove Zero connector

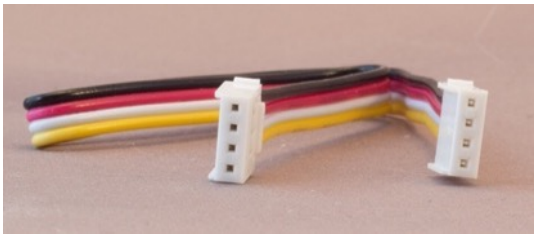
173.2. Datasheet Grove Shield for micro:bit v2.0

- There is a nice review of this shield at: https://www.element14.com/community/roadTestReviews/2767/l/grove-inventor-kit-bbc-microbit-review

173.3. Libraries needed for Grove Shield for micro:bit v2.0

Some Grove modules need one or more libraries, this is documented in their corresponding chapters. There are no specific libraries needed for the Grove Base itself.

174. Grove cable



The Grove cable is used to connect a Grove module to the Grove Base, making it easier to build your projects.

174.1. Seeed documentation Grove cable

http://wiki.seeed.cc/Grove_System/

174.2. Connections when used with analog modules

Pin nr	Name	Description	Grove connector
4 Black	GND	Ground	A0..Ax
3 Red	VCC	VCC	
2 White	Sec. Sig or NC	Secondary signal or Not used	
1 Yellow	Prim. Sig	Primary Signal	

174.3. Connections when used with digital modules (single digital ports)

Most modules that use a digital port, only use 1 digital port (single).

Pin nr	Name	Description	Grove connector
4 Black	GND	Ground	D2..D8
3 Red	VCC	VCC	
2 White	NC	Secondary signal or Not used	
1 Yellow	Prim. Sig	Primary Signal	

174.4. Connections when used with digital modules (double digital ports)

Some modules uses both digital ports on the Grove Connector, this means that the next Grove Connector can't be used for an other module. So when you connect this type of modules on Grove Connector D2, then D3 can't be used for other modules (since Grove Connectors D2 and D3 both use Arduino's digital port 3).

Pin nr	Name	Description	Grove connector
4 Black	GND	Ground	D2..D8 The next grove connector is not available
3 Red	VCC	VCC	
2 White	Sec. Sig.	Secondary Signal	
1 Yellow	Prim. Sig	Primary Signal	

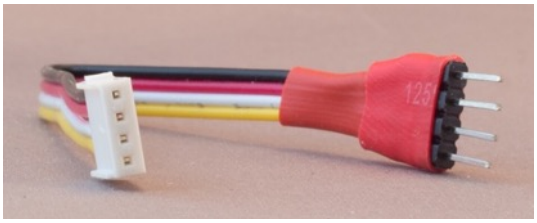
174.5. Connections when used with I2C modules

Pin nr	Name	Description	Grove connector
4 Black	GND	Ground	I2C1.I2Cx
3 Red	VCC	VCC	
2 White	SDA	I2C SDA	
1 Yellow	SCL	I2C SCL	

174.6. Connections when used with UART modules

Pin nr	Name	Description	Grove connector
4 Black	GND	Ground	UART
3 Red	VCC	VCC	
2 White	SDA	TX	
1 Yellow	SCL	RX	

175. Grove to 4 pin Male cable (self-made)



The above cable is a self-made cable that can be used to connect Grove modules to an Arduino with the Grove base, or to connect a non-Grove module to a Grove Base. You can also buy this cable from Seeed Studio "Grove to 4 pin Female/Male jumper.

175.1. Seeed documentation Grove cable

http://wiki.seeed.cc/Grove_System/

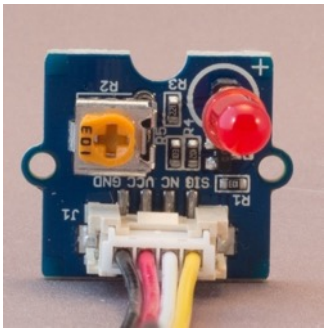
175.2. Connections when used with analog and digital modules

Pin nr	Name	Description	Arduino pin
4 Black	GND	Ground	GND
3 Red	VCC	VCC	5V
2 White	Sec. Sig or NC	Secondary signal or Not used	Not connected
1 Yellow	Prim. Sig	Primary Signal	Any analog or digital port

175.3. Connections when used with I2C modules

Pin nr	Name	Description	Arduino pin
4 Black	GND	Ground	GND
3 Red	VCC	VCC	5V
2 White	SDA	I2C SDA	A4 (=SDA)
1 Yellow	SCL	I2C SCL	A5 (=SCL)

176. Grove LED Socket Kit v1.4 (=LED)




This is a LED module with an onboard current limiting resistor and an onboard potentiometer to control the intensity of the LED.

176.1. Specifications Grove LED Socket Kit

- Grove connector:
- Onboard current limiting resistor.
- Onboard potentiometer to change the intensity of the LED. If you want to control this from your sketch, you'll need to connect this module to a PWM port (D3, D5 & D6) and you need to turn the onboard potentiometer to max (fully CCW).

176.2. Seeed documentation Grove LED Socket Kit

Make sure the LED is connected in the right orientation (Anode to + and Cathode to -). Check "41.1 Specifications LED" for more information about Anode and Cathode.

176.3. Connections Grove LED Socket Kit

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port. PWM is only needed if you want to control the intensity of the LED by your sketch.

176.4. Libraries needed for Grove LED Socket Kit

There are no libraries needed for this module.

176.5. Sample Grove LED Socket Kit

The following sketch will blink a LED every second.

Sample Connections

- Connect this module to D2 on your Grove-shield.

104_Grove_LED-socket.ino

```
int led = 2 ;

void setup() {
  pinMode(led, OUTPUT);
}

void loop()
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

177. Grove Chainable RGB LED v2.0

This module contains an individually addressable RGB LED, than can be chained with a maximum of 1024 LEDs.

177.1. Specifications Grove Chainable RGB LED v2.0

- Grove connector: D2..D8 (both primary and secondary signal wires are used)
- Operating voltage: 5 V
- Supply current: 20 mA
- P9813S14 controller chip
- RGB with 256 grayscales per color
- Maximum of 1024 LED's in a chain.

177.2. Seeed documentation Grove Chainable RGB LED v2.0

http://wiki.seeedstudio.com/Grove-Chainable_RGB_LED/

Datasheet P9813 RGB controller chip:

https://raw.githubusercontent.com/SeeedDocument/Grove-Chainable_RGB_LED/master/res/P9813_datasheet.pdf

177.3. Connections Grove Chainable RGB LED v2.0

This module uses both digital ports on the Grove Connector, this means that the next Grove Connector can't be used for an other module. So when you connect this module on Grove Connector D2, then D3 can't be used for other modules (since Grove Connectors D2 and D3 both use Arduino's digital port 3).

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx The next grove connector is not available	GND
3 Red	VCC	VCC		5V
2 White	Sec. Sig.	Signal 2		Any digital port
1 Yellow	Prim. Sig.	Signal 1		Any digital port

177.4. Libraries needed for Grove Chainable RGB LED v2.0

- Grove Chainable LED by pjp.margues@gmail.com through the library manager.

Library use explanation Grove 16x2 LCD (White on Blue)

```
#include <ChainableLED.h>
```

Include the Chainable LED library.

```
#define NUM_LEDS  1
```

Set the number of LED's in the chain.

```
ChainableLED leds(7, 8, NUM_LEDS);
```

Create leds an instance of ChainableLED, with the digital ports used. The first number is the number of the Grove connector (in this case D7) and the other port is 1 higher (8).

```
leds.setColorHSB(<lednr>, <hue>, <saturation>, <brightness>);
```

Set the HSB-color of LED <lednr> to <hue>, <saturation> and <brightness> (all can vary between 0..1)

```
leds.setColorRGB(<lednr>, <RED>, <GREEN>, <BLUE>);
```

Set the RGB-color of LED <lednr> to <RED>, <GREEN>, <BLUE>.

```
if (up)
  hue+= 0.005;
else
  hue-= 0.005;
```

Either increase or decrease the value of hue with 0.005;

```
if (hue>=1.0 && up)
  up = false;
else if (hue<=0.0 && !up)
  up = true;
```

Change whether hue should be decreased or increased.

177.5. Sample Grove Chainable RGB LED v2.0

The following sketch cycles though all the colors in an uniform way. This is accomplished using a HSB color space.

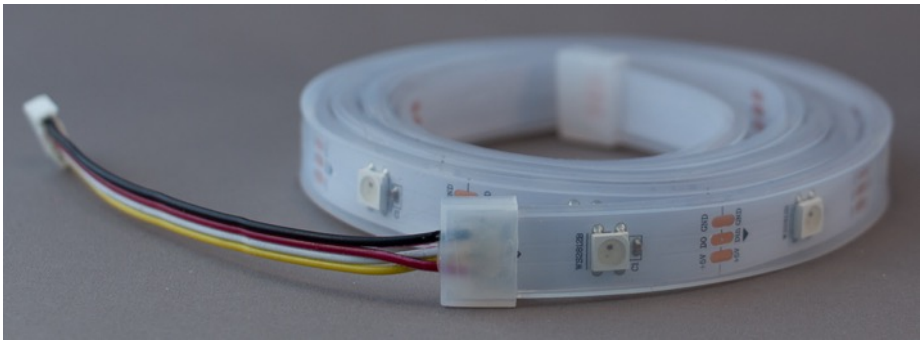
Sample Connections

- Connect this module to D7 on your Grove-shield.

158_Grove-ChainLED.ino

```
#include <ChainableLED.h>
#define NUM_LEDS  1

ChainableLED leds(7, 8, NUM_LEDS);

void setup()
{
}

float hue = 0.0;
boolean up = true;

void loop()
{
  for (byte i=0; i<NUM_LEDS; i++)
    leds.setColorHSB(i, hue, 1.0, 0.1);

  delay(50);

  if (up)
    hue+= 0.005;
  else
    hue-= 0.005;

  if (hue>=1.0 && up)
    up = false;
  else if (hue<=0.0 && !up)
    up = true;
}
```

178. Grove WS2812 Waterproof LED strip



This module contains a chain of 30 individually addressable RGB LED's with the WS2812B controller chip. These WS2812B controlled RGB LED's are also known as Neopixels.

178.1. Specifications Grove WS2812 Waterproof LED strip

- Grove connector: D2..D8 (both primary and secondary signal wires are used)

178.2. Connections Grove WS2812 Waterproof LED strip

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 -	-	-		-
1 Yellow	SIG	Signal		Any digital port

178.3. Libraries needed for Grove WS2812 Waterproof LED strip

- Adafruit's NeoPixel library through the Library Manager.[1]

Library use explanation

```
#include <Adafruit_NeoPixel.h>
```

Include the Adafruit NeoPixel library.

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB +
NEO_KHZ800);
```

Create 'strip' a new instance of the object type Adafruit_NeoPixel. PIN is an Integer value corresponding to the Arduino Digital Output to which I is connected. PIXELS is the number of WS2812B-modules used. NEO_GRB and NEO_KHZ800 are constants defined in the NeoPixel library and need to match the specs of the type of WS2812B-modules used.

```
strip.begin();
strip.show();
```

Initialize the strip and subsequently clear all WS2812B-modules in the strip (switch them off).

```
uint32_t color = strip.Color(255 , 0, 0);
```

Fill a 32 bit integer with the value for Red (255,0,0).

```
strip.setPixelColor(0, color);
```

Set the buffer for WS2812B-module 0 (first WS2812B-module) to the color Red.

[1] Another nice library is FastLED from Daniel Garcia, also through the Library Manager.

```
strip.show();
```

Set all the WS2812B-modules to the colors defined in the WS2812B-modules buffers.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

178.4. Sample Grove WS2812 Waterproof LED strip

The following sketch

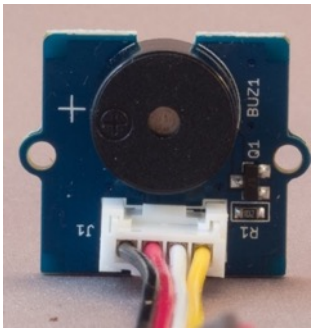
Sample Connections

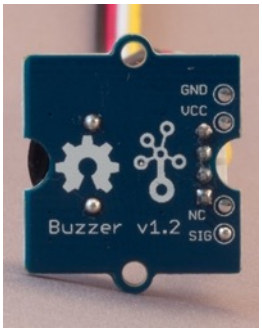
- Connect this module to D7 on your Grove-shield.

16_WS2812B.ino

```
#include <Adafruit_NeoPixel.h>

#define PIN 7
#define PIXELS 30
Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXELS, PIN, NEO_GRB +
NEO_KHZ800);

void setup() {
  strip.begin();
  strip.show();
}

void loop() {
  uint32_t color;
  for (int count = 0; count < PIXELS; count++)
  {
    color = strip.Color(255 , 0, 0);
    strip.setPixelColor(count, color);
    strip.show();
    delay(100);
    color = strip.Color(0, 50, 0);
    strip.setPixelColor(count, color);
    strip.show();
    delay(100);
  }
}
```

179. Grove Buzzer v1.2 (=piezo speaker)





This buzzer is actually a piezo speaker. It can produces monotone sounds or melodies.

179.1. Specifications Grove Buzzer

- Grove connector: D2..Dx
- Operating voltage: 4-8V
- Sound output: >= 85 dB
- Resonant frequency: 2300 +/- 300 Hz

179.2. Seeed documentation Grove Buzzer

http://wiki.seeed.cc/Grove-Buzzer/

179.3. Connections Grove Buzzer

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port

179.4. Libraries needed for Grove Buzzer

There are no libraries needed for this module.

179.5. Sample Grove Buzzer

The following sketch will play a little melody.

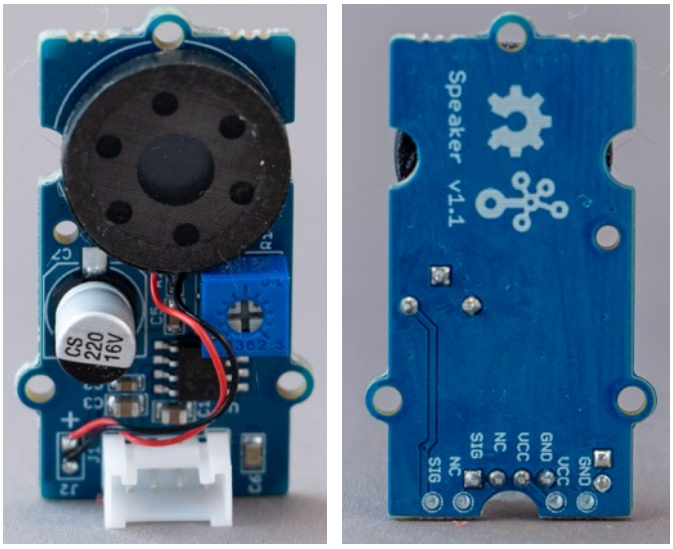
Sample Connections

- Connect this module to D2 on your Grove-shield.

105_Grove_Buzzer.ino

```
int speakerPin = 2;
int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration)
{
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration)
{
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++){
    if (names[i] == note)
    {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++){
    if (notes[i] == ' ')
    {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}
```

180. Grove Speaker v1.1



This module can produce simple melodies. The loudness can be adjusted by the on-board potentiometer.

180.1. Specifications Grove Speaker v1.1

- Grove connector: D3, D5 or D6 (PWM pins)
- Operating voltage: 4.0-5.5 V
- Voltage Gain: 46 db
- Band width: 20 kHz
- LM386 Low Voltage Audio Power Amplifier

180.2. Seeed documentation Grove Speaker v1.1

http://wiki.seeedstudio.com/Grove-Speaker/

Datasheet LM386 Low Voltage Audio Power Amplifier
https://raw.githubusercontent.com/SeeedDocument/Grove-Speaker/master/res/LM386_Low_Voltage_Audio_Power_Amplifier_Datasheet.pdf

180.3. Connections Grove Speaker v1.1

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	Any PWM port (D3, D5 or D6)	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any PWM port: D3, D5, D6, D9, D10 & D11

180.4. Libraries needed for Grove Speaker v1.1

There are no libraries needed for this module.

180.5. Sample Grove Speaker v1.1

The following sketch will play a little melody (it is the same as for the Grove Buzzer).

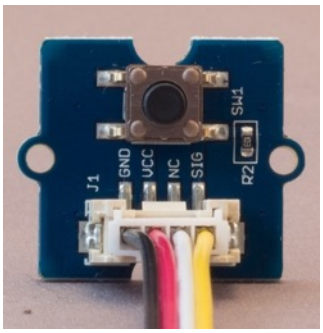
Sample Connections

- Connect this module to D2 on your Grove-shield.

105_Grove_Buzzer.ino

```
int speakerPin = 2;
int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration)
{
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration)
{
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++){
    if (names[i] == note)
    {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++){
    if (notes[i] == ' ')
    {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}
```

181. Grove Button v1.1 (=switch)



This module consists of a simple switch (normally open) and a pulldown resistor. This means that when the switch is in rest position, the output is LOW and when the switch is pressed the output is HIGH.

181.1. Specifications Grove Button

- Grove connector: D2..Dx

181.2. Seeed documentation Grove Button

http://wiki.seeed.cc/Grove-Button/

181.3. Connections Grove Button

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port

181.4. Libraries needed for Grove Button

There are no libraries needed for this module.

181.5. Sample Grove Button

The following sketch will light the internal LED on port 13 when you press the button.

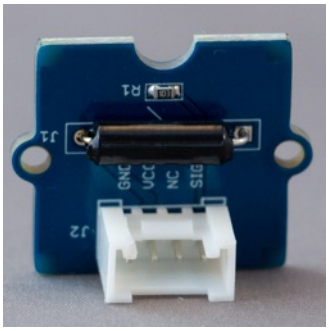
Sample Connections

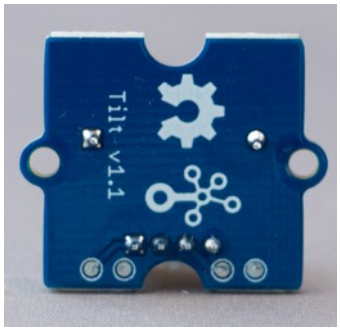
- Connect this module to D2 on your Grove-shield.

106_Grove_Button.ino

```
const int Button= 2;
const int Led =  13;

int Buttonstate = 0;

void setup() {
  pinMode(Led, OUTPUT);
  pinMode(Button, INPUT);  // when using external Pullup/Pulldown resistor
}

void loop(){
  Buttonstate = digitalRead(Button);

  if (Buttonstate == HIGH)
  {
    digitalWrite(Led, HIGH);
  }
  else
  {
    digitalWrite(Led, LOW);
  }
}
```

182. Grove Tilt v1.1





This module can be used to detect a tilt movement. Inside the switch is a ball that make contact with the pins when the case is upright and no contact when the pin is not upright.

182.1. Specifications Grove Tilt v1.1

- Grove connector: D2..D8
- Connecting angle: 10-170 degrees
- Disconnecting angle: 190-350 degrees

182.2. Seeed documentation Grove Tilt v1.1

http://wiki.seeedstudio.com/Grove-Tilt_Switch/

182.3. Connections Grove Tilt v1.1

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 -	-	-		-
1 Yellow	SIG	Signal		Any digital port

182.4. Libraries needed for Grove Tilt v1.1

There are no libraries needed for this module.

182.5. Sample Grove Tilt v1.1

The following sketch changes the status of the built-in LED.

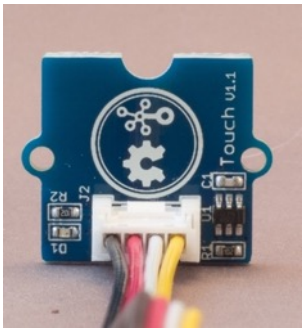
Sample Connections

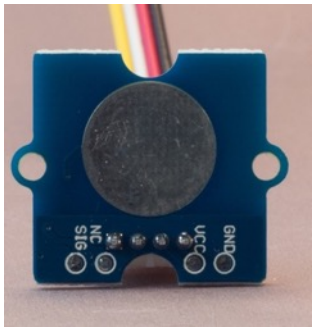
- Connect this module to D7 on your Grove-shield.

159_Grove_Tilt.ino

```
int tiltPin = 7;
int LEDPin = 13;

void setup()
{
    pinMode(LEDPin, OUTPUT);
    pinMode(tiltPin, INPUT);
}

void loop()
{

    if (digitalRead(tiltPin)==HIGH)
    {
        digitalWrite(LEDPin, HIGH);
        delay(200);
        digitalWrite(LEDPin, LOW);
    }

}
```

183. Grove Touch v1.1





183.1. Specifications Grove Touch

- Grove connector: D2..Dx
- Operating voltage: 2-5.5V
- Response time: 60mS at phase mode / 220mS at low power mode (3V)
- Power indicator LED
- Chipset TTP223-BA6

183.2. Seeed documentation Grove Touch

http://wiki.seeed.cc/Grove-Touch_Sensor/

183.3. Connections

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port

183.4. Libraries needed for Grove Touch

There are no libraries needed for this module.

183.5. Sample Grove Touch

The following sketch turns the onboard LED (D13) on, when the Touch sensor is touched.

Sample Connections

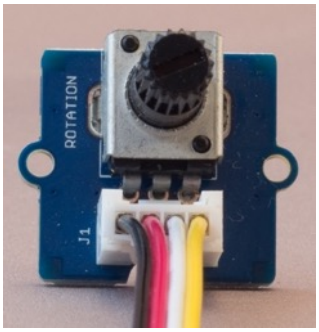
- Connect this module to D2 on your Grove-shield.

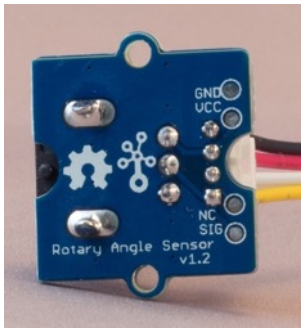
107_Grove_Touch.ino

```
const int TouchPin = 2;
const int ledPin = 13;
void setup()
{
  pinMode(TouchPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  int sensorValue = digitalRead(TouchPin);
  if (sensorValue == 1)
  {
    digitalWrite(ledPin, HIGH);
  }
  else
  {
    digitalWrite(ledPin, LOW);
  }
}
```

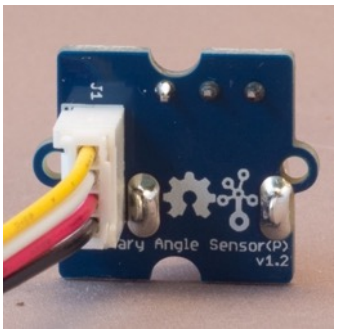
184. Grove Rotary Angle Sensor v1.2 (=potentiometer)

This module is actually a variable resistor, also known as potentiometer. By connecting this module to an analog port, you can measure the angle of the sensor. That's why Seeed calls this a Rotary Angle Sensor.





The pictures below are for the Grove Rotary Angle Sensor P. In this 'P' means panel mount. The Grove connector is at the back so you can mount the module in a panel.



184.1. Specifications Grove Rotary Angle Sensor

- Grove connector: A0..Ax
- Angular range: 300 degrees
- Resistance: 10 Kohm

184.2. Seeed documentation Grove Rotary Angle Sensor

http://wiki.seeed.cc/Grove-Rotary_Angle_Sensor/

184.3. Connections

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	A0..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any analog port

184.4. Libraries needed for Grove Rotary Angle Sensor

There are no libraries needed for this module.

184.5. Sample Grove Rotary Angle Sensor

The following sketch shows the analog output value of the Rotary Angle Sensor. When turned fully CCW the value will be 1024 and when turned fully CW the value will be 0. The Panel version of the Rotary Angle sensor will have opposite values (CCW => 0 and CW => 1024).

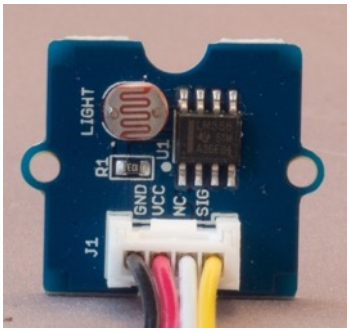
Sample Connections

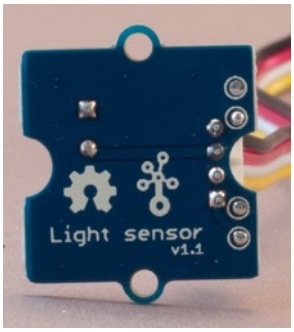
- Connect this module to A0 on your Grove-shield.

108_Grove_Rotary.ino

```
const int analogInPin = A0;
int sensorValue = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(analogInPin);
  Serial.println(sensorValue);
  delay(200);
}
```

185. Grove Light Sensor v1.1 (=LDR)





This module can measure light intensity.

185.1. Specifications Grove Light Sensor v1.1 (LDR)

- Grove connector: A0..Ax
- Operating voltage: 3-5 V
- Supply current: 0.5 - 3 mA
- Photoresistor: GL5528
- Light resistance: 20 Kohm
- Dark resistance: 1 Mom
- Response time: 20-30 milliseconds
- Peak wavelength: 540 nm

185.2. Seeed documentation Grove Light Sensor v1.1 (LDR)

http://wiki.seeed.cc/Grove-Light_Sensor/

Datasheet GL5528 light sensor
https://pi.gate.ac.uk/pages/airpi-files/PD0001.pdf

185.3. Connections Grove Light Sensor v1.1 (LDR)

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	A0..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any analog port

185.4. Libraries needed for Grove Light Sensor v1.1 (LDR)

There are no libraries needed for this module.

185.5. Sample Grove Light Sensor v1.1 (LDR)

The following sketch shows the analog output value of the Light Sensor. The value will be zero when it is totally dark and 1023 when fully lid.

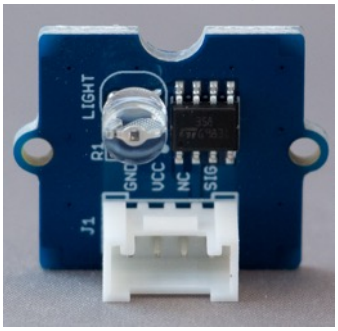
Sample Connections

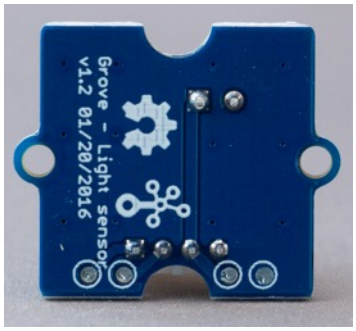
- Connect this module to A0 on your Grove-shield.

109_Grove_Light-Sensor.ino

```
int sensorPin = A0;
unsigned int sensorValue = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue, DEC);
  delay(500);
}
```

186. Grove Light Sensor v1.2 (Linear Light Sensor)





This the successor of the Grove Light Sensor v1.1 with an LS06-S light sensor replacing the GL5528 photoresistor of v1.1.

186.1. Specifications Grove Light Sensor v1.2 (Linear Light Sensor)

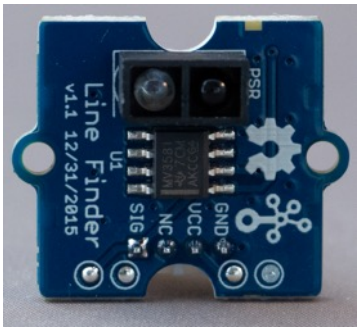
- See Grove Light Sensor v1.1

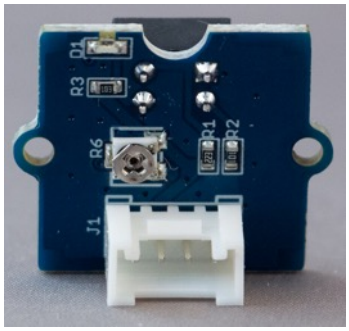
186.2. Seeed documentation Grove Light Sensor v1.2 (Linear Light Sensor)

http://wiki.seeed.cc/Grove-Light_Sensor/

Datasheet LS06-S Linear Light Sensor:
http://www.image.micros.com.pl/_dane_techniczne_auto/cz%20ls06-s5.pdf

187. Grove Line Finder v1.1





This module can detect a black line and can be used in line-tracking robots, but you'll need at least 3 Line Finder modules for such a vehicle.

187.1. Specifications Grove Line Finder v1.1

- Grove connector: D2..D8
- Operating voltage: 5 V
- High when black is detected, Low when white is detected
- RS-06WD photo reflective diode
- MV358 comparator

187.2. Seeed documentation Grove Line Finder v1.1

http://wiki.seeedstudio.com/Grove-Line_Finder/

Datasheet MV358 comparator:

https://github.com/SeeedDocument/Grove_Line_Finder/raw/master/res/Lmv358.pdf

Datasheet RS-06WD photo reflective diode

https://cdn.boxtec.ch/pub/diverse/rs-06wd.pdf

187.3. Connections Grove Line Finder v1.1

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 -	-	-		-
1 Yellow	SIG	Signal		Any digital port

187.4. Libraries needed for Grove Line Finder v1.1

There are no libraries needed for this module.

187.5. Sample Grove Line Finder v1.1

The following sketch changes the status of the built-in LED.

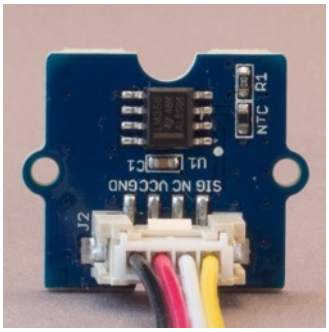
Sample Connections

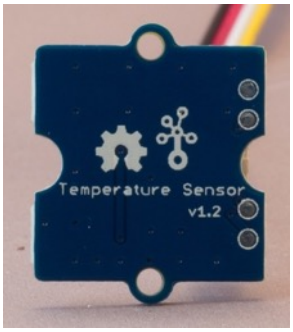
- Connect this module to D7 on your Grove-shield.

159_Grove_LineFinder.ino

```
int signalPin =  3;

void setup()
{
  pinMode(signalPin, INPUT);
  Serial.begin(9600);
}

void loop()
{
  if (HIGH == digitalRead(signalPin))
  {
    Serial.println("black");
  }
  else
  {
    Serial.println("white");
  }
  delay(1000);
}
```

188. Grove Temperature Sensor v1.2





This module can measure the air temperature.

188.1. Specifications Grove Temperature Sensor

- Grove connector: A0..Ax
- Operating voltage: 3.3 - 5V
- Thermistor Zero power resistance: 100 Kohm
- Resistance Tolerance: +/- 1%
- Operating temperature range: -40 - 125 Celsius
- Accuracy: +/- 1.5 degrees Celsius

188.2. Seeed documentation Grove Temperature Sensor

http://wiki.seeed.cc/Grove-Temperature_Sensor_V1.2/

188.3. Connections Grove Temperature Sensor

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	A0..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any analog port

188.4. Libraries needed for Grove Temperature Sensor

- Math library from Arduino to calculate the temperature.

188.5. Sample Grove Temperature Sensor

The following sketch shows the measured temperature.

Sample Connections

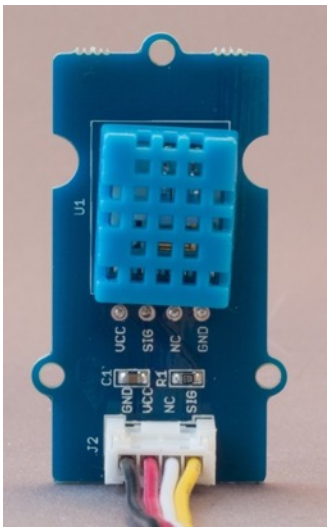
- Connect this module to A0 on your Grove-shield.

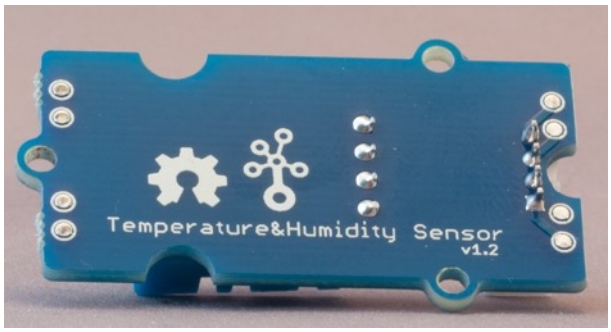
110_Grove_Temperature.ino

```
#include <math.h>

const int B=4275;                 // B value of the thermistor
const int R0 = 100000;            // R0 = 100k
const int pinTempSensor = A0;     // Grove - Temperature Sensor connect to
A5

void setup()
{
    Serial.begin(9600);
}

void loop()
{
    int a = analogRead(pinTempSensor );

    float R = 1023.0/((float)a)-1.0;
    R = 100000.0*R;

    float temperature=1.0/(log(R/100000.0)/B+1/298.15)-273.15;//convert to
temperature via datasheet ;

    Serial.print("temperature = ");
    Serial.println(temperature);

    delay(100);
}
```

189. Grove Temperature & Humidity Sensor v1.2

With this sensor you measure both temperature and humidity.





189.1. Specifications Grove Temperature & Humidity Sensor

- Grove connector: A0..Ax
- Relative humidity and temperature measurement
- Long transmission distance
- Low power consumption
- DOES NOT WORK BELOW 0 degrees Celsius
- No decimals

189.2. Seeed documentation Grove Temperature & Humidity Sensor

http://wiki.seeed.cc/Grove-TemperatureAndHumidity_Sensor/

189.3. Connections

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	A0..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any analog port

189.4. Libraries needed for Grove Temperature & Humidity Sensor

There are no libraries needed for this module.

189.5. Sample Grove Temperature & Humidity Sensor

The following sketch shows the humidity and temperature every 2 seconds (sample copied from SEEED). No decimals

Sample Connections

- Connect this module to A0 on your Grove-shield.

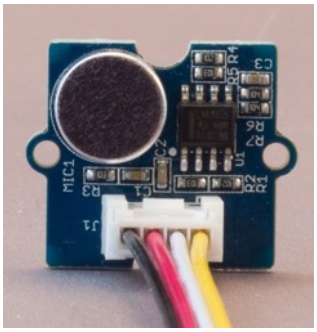
111_Grove_Temperature-Humidity.ino

```
#define DHT11_PIN 0

byte read_dht11_dat()
{
  byte i = 0;
  byte result = 0;
  for (i = 0; i < 8; i++)
  {
    while (!(PINC & _BV(DHT11_PIN))); // wait for 50us
    delayMicroseconds(30);
    if (PINC & _BV(DHT11_PIN))
      result |= (1 << (7 - i));
    while ((PINC & _BV(DHT11_PIN))); // wait '1' finish
  }
  return result;
}

void setup()
{
  DDRC |= _BV(DHT11_PIN);
  PORTC |= _BV(DHT11_PIN);
  Serial.begin(9600);
  Serial.println("Ready");
}

void loop()
{
  byte dht11_dat[5];
  byte dht11_in;
  byte i;
  // start condition
  // 1. pull-down i/o pin from 18ms
  PORTC &= ~_BV(DHT11_PIN);
  delay(18);
  PORTC |= _BV(DHT11_PIN);
  delayMicroseconds(40);
  DDRC &= ~_BV(DHT11_PIN);
  delayMicroseconds(40);
  dht11_in = PINC & _BV(DHT11_PIN);
  if (dht11_in)
  {
    Serial.println("dht11 start condition 1 not met");
    return;
  }
  delayMicroseconds(80);
  dht11_in = PINC & _BV(DHT11_PIN);
  if (!dht11_in)
  {
    Serial.println("dht11 start condition 2 not met");
    return;
  }
  delayMicroseconds(80);
  // now ready for data reception
```

```
  for (i = 0; i < 5; i++)
  {
    dht11_dat[i] = read_dht11_dat();
  }
  DDRC |= _BV(DHT11_PIN);
  PORTC |= _BV(DHT11_PIN);
  byte dht11_check_sum = dht11_dat[0] + dht11_dat[1] + dht11_dat[2] +
dht11_dat[3];
  // check check_sum
  if (dht11_dat[4] != dht11_check_sum)
  {
    Serial.println("DHT11 checksum error");
  }
  Serial.print("Current humdity = ");
  Serial.print(dht11_dat[0], DEC);
  Serial.print(".");
  Serial.print(dht11_dat[1], DEC);
  Serial.print("%  ");
  Serial.print("temperature = ");
  Serial.print(dht11_dat[2], DEC);
  Serial.print(".");
  Serial.print(dht11_dat[3], DEC);
  Serial.println("C  ");
  delay(2000);
}
```

190. Grove Sound Sensor v1.6

With this sensor you can detect sound, so your sketch can respond to it. This sensor is not capable of analyzing frequencies, nor is it capable to record sound.



190.1. Specifications Grove Sound Sensor

- Grove connector: A0..Ax
- Operating voltage: 4-12V
- Operating current: 4-5 mA
- LM358 amplifier
- Microphone sensitivity: 52-48 dB
- Microphone frequency: 16-20 kHz

190.2. Seeed documentation Grove Sound Sensor

http://wiki.seeed.cc/Grove-Sound_Sensor/

190.3. Connections

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	A0..Ax	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any analog port

190.4. Libraries needed for Grove Sound Sensor

There are no libraries needed for this module.

190.5. Sample Grove Sound Sensor

The following sketch will only show the sound level when it exceeds a specific level.

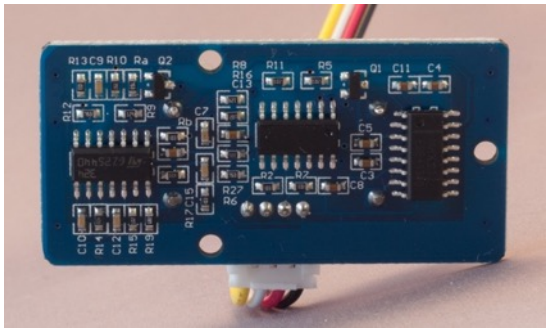
Sample Connections

- Connect this module to A0 on your Grove-shield.

112_Grove_Sound-Sensor.ino

```
int sensorA=A0;
int sensorvalueA;

void setup()

{
  Serial.begin(9600);
}

void loop()
{
  sensorvalueA=analogRead(sensorA);
  if (sensorvalueA > 550)
  {
    Serial.println(sensorvalueA);
    delay(1000);
  }
}
```

191. Grove Ultrasonic Ranger v2.0 (=distance sensor)



191.1. Specifications Grove Ultrasonic Ranger

- Grove connector: D2..Dx
- Operating voltage: 3.3-5V
- Operating current: 15mA
- Ultrasonic frequency: 42 kHz
- Measuring range: 3-400 cm
- Resolution: 1cm

191.2. Seeed documentation Grove Ultrasonic Ranger

http://wiki.seeed.cc/Grove-Ultrasonic_Ranger/

191.3. Connections Grove Ultrasonic Ranger

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port

191.4. Libraries needed for Grove Ultrasonic Ranger

- Ultrasonic library from Seeed-Studio
  https://github.com/Seeed-Studio/Grove_Ultrasonic_Ranger .

Library use explanation Grove Ultrasonic Ranger

```
#include "Ultrasonic.h"
```

Include the Ultrasonic Ranger library from Seeed Studio.

```
Ultrasonic ultrasonic(DistanceSensorpin);
```

Create 'ultrasonic' a new instance of the object type Ultrasonic, with DistanceSensorpin as the port to which the module is connected.

```
RangeInCentimeters = ultrasonic.MeasureInCentimeters();
```

Measure the distance in centimeters.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

191.5. Sample Grove Ultrasonic Ranger

The following sketch the distance between the Ultrasonic ranger and an object.

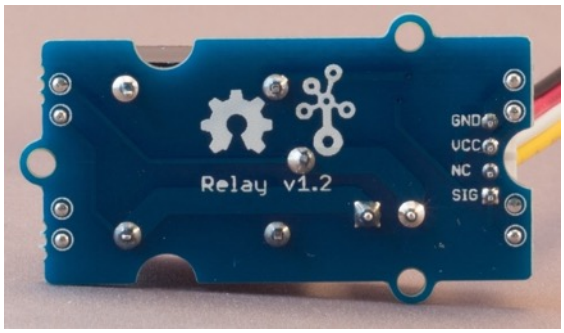
Sample Connections

- Connect this module to D2 on your Grove-shield.

113_Grove_Ultrasonic.ino

```
#include "Ultrasonic.h"

int DistanceSensorpin=2;

Ultrasonic ultrasonic(DistanceSensorpin);

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  long RangeInCentimeters;
  RangeInCentimeters = ultrasonic.MeasureInCentimeters();
  delay(150);
  Serial.print("The distance is: ");
  Serial.println(RangeInCentimeters);
}
```

192. Grove Relay v1.2[1]



Take extra care, because working with mains electricity CAN BE VERY DANGEROUS.

192.1. Specifications Grove Relay 1.2

- Grove connector: D2 .. Dx
- Normally open switch
- Operating Voltage 3.3V-5V
- Operating Current 100 mA
- Maximum Switching voltage 250 VAC / 30 VDC
- Maximum Switching Current: 5 A
- Indicator LED
- Peak voltage at 250V at 10 A

192.2. Seeed documentation

http://wiki.seeed.cc/Grove-Relay/

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with mains electricity, consult someone with the right skills.

192.3. Connections

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 White	NC	Not used		Not connected
1 Yellow	SIG	Signal		Any digital port

192.4. Libraries needed for

There are no libraries needed for this module.

192.5. Sample

The following sketch will close or open the relay every second.

Sample Connections

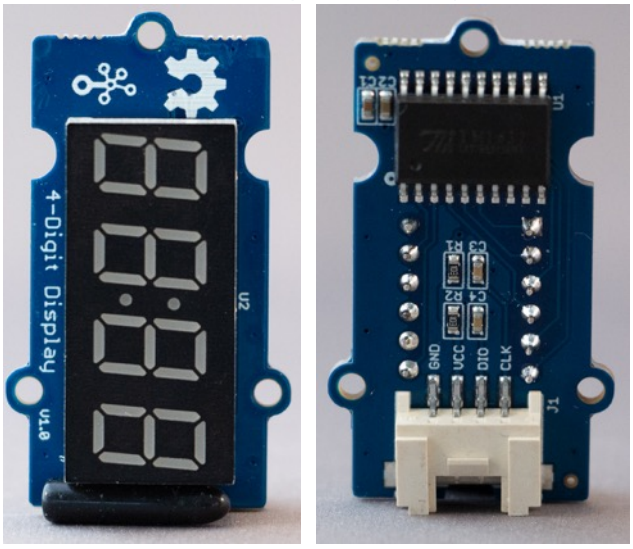
- Connect this module to D2 on your Grove-shield..

Take extra care, because working with mains electricity CAN BE VERY DANGEROUS. Only connect mains electricity components if you are well trained.

114_Grove_Relay.ino

```
int Relaypin = 2 ;

void setup()
{
  pinMode(Relaypin, OUTPUT);
}

void loop()
{
  digitalWrite(Relaypin, HIGH);
  delay(1000);
  digitalWrite(Relaypin, LOW);
  delay(1000);
}
```

193. Grove 4 Digit Display v1.0



This module can show 4 x 7 segment digits and one colon.

193.1. Specifications Grove 4 Digit Display v1.0

- Grove connector: D2..D8 (both primary and secondary signal wires are used)
- Operating voltage: 3.3-5.5 V
- Supply current: 0.2 - 80 mA
- TM1637 chip to scale the number of controlling pins to only 2.

193.2. Seeed documentation Grove 4 Digit Display v1.0

http://wiki.seeedstudio.com/Grove-4-Digit_Display/

193.3. Connections Grove 4 Digit Display v1.0

This module uses both digital ports on the Grove Connector, this means that the next Grove Connector can't be used for an other module. So when you connect this module on Grove Connector D2, then D3 can't be used for other modules (since Grove Connectors D2 and D3 both use Arduino's digital port 3).

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx The next grove connector is not available	GND
3 Red	VCC	VCC		5V
2 White	CLK	Clock		Any digital port
1 Yellow	DIO	Data input		Any digital port

193.4. Libraries needed for Grove 4 Digit Display v1.0

The following library is needed:

- Grove-4-digit Display library by Seeed studio through the library manager.

Library use explanation Grove

```
#include "TM1637.h"
```

Include the TM1637 library from Seed studio.

```
#define CLK 2
```
This is the digital port for the clock signal and also the nr of the Grove connector D2.

```
#define DIO 3
```
This is the digital port for the DIO port and must be 1 higher then the nr of the Grove connector D2.

```
TM1637 tm1637(CLK, DIO);
```
Create tm1637, an instance of TM1637.

```
tm1637.init();
```
Initialize the TM1637 chip.

```
tm1637.set(BRIGHT_TYPICAL);
```
Set the brightness of the display (0..8) with BRIGHT_DARKEST = 0, BRIGHT_TYPICAL = 2 and BRIGHTEST = 7;

```
tm1637.display(3, count / 1 % 10);
```
Extract the Least Significant digit from the variable count and display this on the 4th digit.

```
tm1637.display(2, count / 10 % 10);
```
Extract the second digit from the variable count and display this on the 3rd digit.

```
tm1637.display(1, count / 100 % 10);
```
Extract the third digit from the variable count and display this on the 2nd digit.

```
tm1637.display(0, count / 1000 % 10);
```
Extract the Most Significant digit from the variable count and display this on the 1st digit.

193.5. Sample Grove 4 Digit Display v1.0

The following sketch counts down from 9999 to 0000 and afterwards shows the hexnumber

Sample Connections

- Connect this module to D2 on your Grove shield

155_Grove_4digit.ino

```
#include "TM1637.h"
#define CLK 2
#define DIO 3
TM1637 tm1637(CLK, DIO);

void setup()
{
  tm1637.init();
  tm1637.set(BRIGHT_TYPICAL);
}

void loop()
{
  for (int count = 9999; count >= 0; count--)
  {
    tm1637.display(3, count / 1 % 10);
    tm1637.display(2, count / 10 % 10);
    tm1637.display(1, count / 100 % 10);
    tm1637.display(0, count / 1000 % 10);
```

```
    delay(10);
  }
  while (true);
  {

  }
}
```

194. Grove 16x2 LCD (White on Blue)


This module can display 2 lines of 16 white characters on a blue background.

194.1. Specifications Grove 16x2 LCD (White on Blue)

- Grove connector: I2C
- I2C Communication uses only 2 IO's
- Built-in English fonts
- 16x2 LCD
- Blue coloured background
- Operating voltage: 5 V
- Automatic power-on reset

194.2. Seeed documentation Grove 16x2 LCD (White on Blue)

http://wiki.seeedstudio.com/Grove-16x2_LCD_Series/

194.3. Connections Grove LCD RGB Backlight

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	I2C0..I2Cx	GND
3 Red	VCC	VCC		5V
2 White	SDA	I2C SDA		A4 (=SDA)
1 Yellow	SCL	I2C SCL		A5 (=SCL)

194.4. Libraries needed for Grove 16x2 LCD (White on Blue)

- Grove LCD RGB Backlight library from Seeed Studio
  https://github.com/Seeed-Studio/Grove_LCD_RGB_Backlight/archive/master.zip

Library use explanation Grove 16x2 LCD (White on Blue)

```
#include <Wire.h>
```

Include the Inter-Integrated Circuit (I²C) and Two Wire Interface (TWI) library.

```
#include "rgb_lcd.h"
```

Include the Grove LCD RGB library from Seeed Studio.

```
rgb_lcd lcd;
```

Create 'lcd' a new instance of the object type rgb_lcd.

```
lcd.begin(16, 2);
```

Initialize your display for 16 characters on 2 rows (16x2).

```
lcd.setRGB(colorR, colorG, colorB);
```

Set the backlight to a specific RGB value, but this doesn't work on this 16x2 White on Blue LCD.

```
lcd.print("hello, world!");
```

Print the text "hello world".

```
lcd.setCursor(0, 1);
```

Set the cursor to the 0th row and 1st column.

194.5. Sample Grove 16x2 LCD (White on Blue)

The following sketch will show the text Hello World with a green background light, on the second row a counter will be shown.

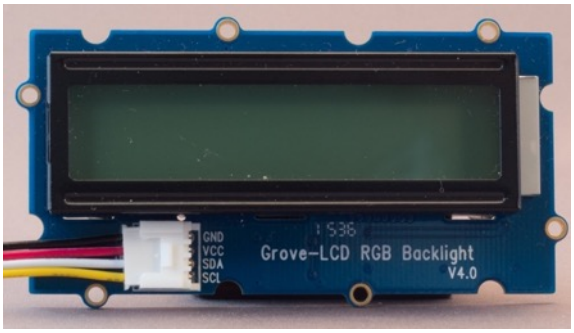
Sample Connections

- Connect this module to any of the I2C ports on your Grove-shield.

157_Grove-16x2_blue.ino

```
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

void setup()
{
  lcd.begin(16, 2);
  lcd.print("hello, world!");
  delay(1000);
}

void loop()
{
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
  delay(100);
}
```

195. Grove LCD 16x2 RGB Backlight v4.0



This module can display 2 lines of 16 white characters on a RGB background.

195.1. Specifications Grove LCD 16x2 RGB Backlight

- Grove connector: I2C
- I2C communication, uses only 2 IO's
- Input Voltage: 5V
- Built-in English fonts
- 16x2 LCD
- Automatic power-on reset

195.2. Seeed documentation Grove LCD 16x2 RGB Backlight

http://wiki.seeed.cc/Grove-LCD_RGB_Backlight/

195.3. Connections Grove LCD RGB Backlight

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	I2C0..I2Cx	GND
3 Red	VCC	VCC		5V
2 White	SDA	I2C SDA		A4 (=SDA)
1 Yellow	SCL	I2C SCL		A5 (=SCL)

195.4. Libraries needed for Grove LCD 16x2 RGB Backlight

- Grove LCD RGB Backlight library from Seeed Studio
  https://github.com/Seeed-Studio/Grove_LCD_RGB_Backlight/archive/master.zip

Library use explanation Grove LCD 16x2 RGB Backlight

```
#include <Wire.h>
```

Include the Inter-Integrated Circuit (I²C) and Two Wire Interface (TWI) library.

```
#include "rgb_lcd.h"
```

Include the Grove LCD RGB library from Seeed Studio.

```
rgb_lcd lcd;
```

Create 'lcd' a new instance of the object type rgb_lcd.

```
lcd.begin(16, 2);
```

Initialize your display for 16 characters on 2 rows (16x2).

```
lcd.setRGB(colorR, colorG, colorB);
```
Set the backlight to a specific RGB value.

```
lcd.print("hello, world!");
```
Print the text "hello world".

```
lcd.setCursor(0, 1);
```
Set the cursor to the 0th row and 1st column.

195.5. Sample Grove LCD 16x2 RGB Backlight

The following sketch will show the text Hello World with a green background light, on the second row a counter will be shown.

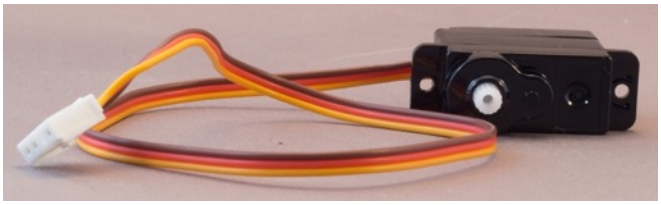
Sample Connections

- Connect this module to any of the I2C ports on your Grove-shield.

020_LCD16x2_I2C_RGB.ino

```
#include <Wire.h>
#include "rgb_lcd.h"

rgb_lcd lcd;

const int colorR = 0;
const int colorG = 255;
const int colorB = 0;

void setup()
{
  lcd.begin(16, 2);
  lcd.setRGB(colorR, colorG, colorB);
  lcd.print("hello, world!");
  delay(1000);
}

void loop()
{
  lcd.setCursor(0, 1);
  lcd.print(millis() / 1000);
  delay(100);
}
```

196. Grove mini Servo



Servo's are DC motors with a potentiometer connected to the motor shaft. This potentiometer tells the servo driver in which position the shaft is. This way you can turn this shaft very accurate, but only for a limited angle. Most common servos can only turn for about 180 degrees. In lots of projects (RC cars, model planes etc.) only 90 degrees is used.

196.1. Specifications Grove mini Servo

- Grove connector: D2..Dx
- Working voltage: 4,8-6V
- Torque: 1.5/1.8 Kg.cm
- Speed: 0.12/0.16 s/60 degrees

196.2. Seeed documentation Grove mini Servo

http://wiki.seeedstudio.com/wiki/Grove_-_Servo

196.3. Connections Grove mini Servo

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Brown	GND	Ground	D2..Dx	GND
3 Red	VCC	VCC		5V
2 -	-	-		-
1 Yellow	SIG	Signal		Any digital port

Libraries needed for Grove mini Servo

- Servo library through the Library Manager.

Library use explanation

```
#include <Servo.h>
```

Include the servo library included in Arduino IDE.

```
Servo myservo;
```

Create myservo a new instance of the object type Servo.

```
myservo.attach(2);
```

Connect myservo to D2.

```
myservo.writeMicroseconds(1500);
```

Set servo at its middle position. Depending on the servo this value can vary between 500..2300. Be careful with the minimum and maximum values. Check every new servo before you implement it in your project. If you've determined the maximum values, you could calculate the middle position.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

196.4. Sample Grove mini Servo

This servo sweeps between three positions, full left, middle, full right, middle, etc..

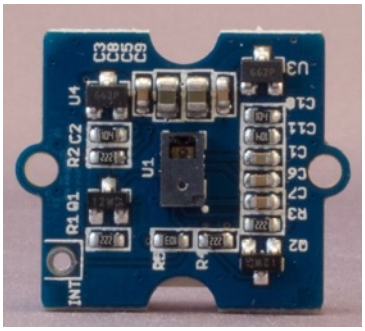
Sample Connections

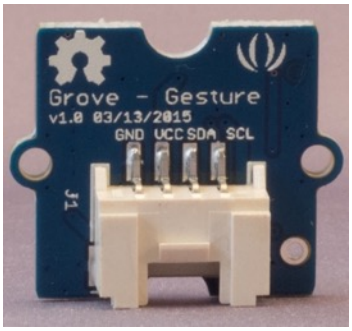
Connect this module to D2 on your Grove-shield.

115_Grove_Servo.ino

```
#include <Servo.h>
Servo myservo;

int pos = 0;

void setup()
{
  myservo.attach(6);
}


void loop()
{
  myservo.writeMicroseconds(600);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
  myservo.writeMicroseconds(2150);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
}
```

197. Grove Gesture v1.0





With this module, you can recognize 9 different gestures (and with some extra coding you can even add 4 extra gestures).

197.1. Specifications Grove Gesture

- Grove connector: I2C
- I2C communication,
- Built-in proximity detection
- Detection range: 5-15 mm
- 9 basic gestures
  - Up
  - Down
  - Left
  - Right
  - Forward
  - Backward
  - Clockwise
  - Counter Clockwise
  - Wave
- Power supply: 5V

197.2. Seeed documentation Grove Gesture

http://wiki.seeed.cc/Grove-Gesture_v1.0/

197.3. Datasheet Grove Gesture

- PAJ7620U2 Integrated Gesture Recognition Sensor
  http://www.pixart.com/upload/PAJ7620U2_GDS_v1.0_29032016_20160623194552.pdf

197.4. Connections Grove Gesture

Pin nr	Name	Description	Grove connector	Arduino pin (without Grove Shield)
4 Black	GND	Ground	I2C0..I2Cx	GND
3 Red	VCC	VCC		5V
2 White	SDA	I2C SDA		A4 (=SDA)
1 Yellow	SCL	I2C SCL		A5 (=SCL)

197.5. Libraries needed for Grove Gesture

- You need the paj7620 library from Seeed Studio.
  https://github.com/Seeed-Studio/Gesture_PAJ7620

Library use explanation

```
#include <Wire.h>
```

```
#include "paj7620.h"
```

Include the paj7620 library for the Grove Gesture sensor.

```
#define GES_REACTION_TIME        500
#define GES_ENTRY_TIME           800
#define GES_QUIT_TIME            1000
```

This are timeout values, you can tweak these values to optimize the recognition of your gestures.

```
uint8_t error = paj7620Init();
```

Initialize the Grove Gesture module.

```
error = paj7620ReadReg(0x43, 1, &data);
```

Read from the Grove Gesture sensor to determine the value of register 0x43.

```
case GES_RIGHT_FLAG:
      Serial.println("Right");
      delay(GES_QUIT_TIME);
      break;
```

Print the text "Right" when the "Right" gesture was recognized.

```
    case GES_LEFT_FLAG:
        ……
    case GES_UP_FLAG:
        ……
    case GES_DOWN_FLAG:
        ……
    case GES_CLOCKWISE_FLAG:
        ……
    case GES_COUNT_CLOCKWISE_FLAG:
        ……
```

Do the same with LEFT, UP, DOWN, CLOCKWISE and COUNT_CLOCKWISE. Doing the same with FORWARD and BACKWARD gives very inaccurate results. That's because when moving FORWARD/BACKWARD you must move very straight, otherwise LEFT, RIGHT, UP and DOWN will also be recognized. Take a look at the examples that came with the library to see how you can recognize FORWARD and BACKWARD more accurate. There is even an example that can also recognize Up & Down, Down & Up, Left & Right and Right & (9 + 4 = 15 gestures).

```
    default:
      paj7620ReadReg(0x44, 1, &data1);
      if (data1 == GES_WAVE_FLAG)
      {
        Serial.println("wave");
      }
      break;
```

The Wave gesture is stored in register 0x44, so if no particular gesture was recognized, register 0x44 is checked for the Wave gesture.

197.6. Sample Grove Gesture

The following sketch will display 9 different gestures.

Sample Connections

- Connect this module to any of the I2C ports on your Grove-shield.

Sample Sketch

```
#include <Wire.h>
#include "paj7620.h"

#define GES_REACTION_TIME        500
#define GES_ENTRY_TIME           800
#define GES_QUIT_TIME            1000

void setup()
{
  Serial.begin(9600);
  Serial.println("\nPAJ7620U2 TEST DEMO: Recognize 9 gestures.");
  uint8_t error = paj7620Init();
  Serial.println("Please input your gestures:\n");
}

void loop()
{
  uint8_t data = 0, data1 = 0, error;
  error = paj7620ReadReg(0x43, 1, &data);
  switch (data)
  {
    case GES_RIGHT_FLAG:
      Serial.println("Right");
      delay(GES_QUIT_TIME);
      break;

    case GES_LEFT_FLAG:
      Serial.println("Left");
      delay(GES_QUIT_TIME);
      break;

    case GES_UP_FLAG:
      Serial.println("Up");
      delay(GES_QUIT_TIME);
      break;

    case GES_DOWN_FLAG:
      Serial.println("Down");
      delay(GES_QUIT_TIME);
      break;

    case GES_CLOCKWISE_FLAG:
      Serial.println("Clockwise");
      break;


    case GES_COUNT_CLOCKWISE_FLAG:
      Serial.println("anti-clockwise");
      break;

    default:
      paj7620ReadReg(0x44, 1, &data1);
      if (data1 == GES_WAVE_FLAG)
      {
```

```
        Serial.println("wave");
      }
      break;
  }
  delay(100);
}
```

Isolation from higher voltages or higher currents

Most sensors and actuators work on low voltages and low currents so they can safely been used with your Arduino board. Switching voltages higher then 5 V or driving currents highr then 20-40 mA, need some special equipment and special care if it concerns mains electricity. This section describe the use of relays and optocouplers.

198. Relay 5V[1] board

With this relay you can use your Arduino to switch equipment to a maximum of 10 A at 250 VAC. So you could switch lightbulbs and other household or workshop equipment connected to mains electricity.

Take extra care, because working with mains electricity CAN BE VERY DANGEROUS.

198.1. Specifications Relay 5V

- SRD-05VDC-SL-C
- Input: 3-48 V
- Load 10 A:
  - 250 VAC
  - 125 VAC
  - 30 VDC
  - 28 VDC
- 3 Terminals which you can use as a Changeover switch or 1 NO (normally open) and 1 NC (normally closed) pair.

198.2. Datasheet Relay 5 V

http://www.parallax.com/sites/default/files/downloads/27115-Single-Relay-Board-Datasheet.pdf

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with mains electricity, consult someone with the right skills.

198.3. Connections Relay 5V

3 pin header

This 3 pin header row is ONLY TO BE USED with low voltages like on the Arduino and SHOULD NEVER BE CONNECTED TO MAINS ELCTRICITY!!!

Pin nr	Name	Description	Arduino pin
1	S	Signal	Any digital port
2	+	Transmit data	5V
3	-	Ground	GND

3 screws terminal

These connections are to be used with mains electricity (max. 10A at 250V) so be extra careful!!!

Pin nr	Name	Description
1	NC	Normally closed
2	CO	Common
3	NO	Normally open

Use screw 1 and 2 as a switch on one mains electricity wire of your device, if this device should normally be switched OFF. Giving a HIGH signal on pin header 1 will switch your device to ON.
Use screw 2 and 3 as a switch on one mains electricity wire of your device if this device should normally be switched ON. Giving a HIGH signal on pin header 1 will switch your device to OFF.

198.4. Libraries needed for Relay 5V

None needed

198.5. Sample Relay 5V

The following sketch will turn on a mains electricity lightbulb for 3 seconds, then turn it off for 3 seconds and repeat this sequence forever.

Sample Connections[1]

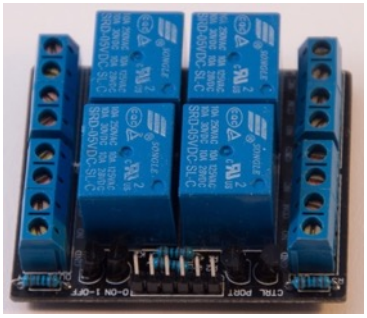
- Send your sketch to the Arduino
- Disconnect your laptop/computer from the Arduino (as a precaution for a faulty relay).Connect S to D12.
- Connect + to 5V.
- Connect - to GND.
- Remove your mains electricity connector from the wall outlet.
- Connect one end of the LIFE wire (coming from a light bulb) to screw 2. Remove just enough isolation, so there is not too much blank wire visible. Touching the blank wire, or touch the metal parts of the screws is very dangerous.
- Connect the other end of the LIFE wire (the part that comes from the mains plug of your device) to screw 1.
- Check all wiring.
- Connect another external power source (battery pack) to your Arduino.

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with mains electricity, consult someone with the right skills.

117_Relay-1x.ino

```
int Relay = 12;

void setup()
{
  pinMode(Relay, OUTPUT);
}

void loop()
{
     digitalWrite(Relay, HIGH);
     delay(3000);
     digitalWrite(Relay, LOW);
     delay(3000);
}
```

199. 4x Relay 5V[1] Board



With this relay you can use your Arduino to switch 4 different devices individually to a maximum of 10 A at 250 VAC each. So you could switch lightbulbs and other household or workshop equipment connected to mains electricity.

Take extra care, because working with mains electricity CAN BE VERY DANGEROUS.

199.1. Specifications 4x Relay 5V Board

- SRD-05VDC-SL-C
- Input: 3-48 V
- Load 10 A:
  - 250 VAC
  - 125 VAC
  - 30 VDC
  - 28 VDC
- 3 screws which you can use as a Changeover switch or 1 NO (normally open) and 1 NC (normally closed) pair.
- 1 screw to ground, this is a common ground with the Arduino ground, so be careful not to make any mistakes.

199.2. Datasheet 4x Relay 5V Board

http://www.parallax.com/sites/default/files/downloads/27115-Single-Relay-Board-Datasheet.pdf

199.3. Connections 4x Relay 5V Board

6 pin header

This 6 pin header row is ONLY TO BE USED with low voltages like on the Arduino and SHOULD NEVER BE CONNECTED TO MAINS ELCTRICITY!!!

Pin nr	Name	Description	Arduino pin
1	VCC	5V	5V
2..5	K1..K4	Relay K1..K4	Any Digital port
6	GND	Ground	GND

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with mains electricity, consult someone with the right skills.

4 Screws terminals OUT1..OUT4

These connections are to be used with mains electricity (max. 10A at 250V) so be extra careful!!!

Pin nr	Name	Description
1	GND	Ground
2	NO	Normally Open
3	COM	Common
4	NC	Normally Closed

- Case 1: The device should normally be switched off
  Use screw 4 and 3. Giving a HIGH signal on one of the relay pinheaders, the corresponding device will be switched to ON.
- Case 2: The device should normally be switched on.
  Use screw 2 and 3. Giving a HIGH signal on one of the relay pinheaders, the corresponding device will be switched to OFF

199.4. Libraries needed for 4x Relay 5V Board

None needed.

199.5. Sample 4x Relay 5V Board

The following sketch will turn on and off 4 light bulbs in a row.

Sample Connections[1]

- Send your sketch to the Arduino
- Disconnect your laptop/computer from the Arduino (as a precaution for a faulty relay).
- Connect K1..K4 to D2..D5.
- Connect + to 5V.
- Connect - to GND.
- Remove your mains electricity connector from the wall outlet.
- Connect one end of the LIFE wire (coming from a light bulb) to screw 3 (Common). Remove just enough isolation, so there is not too much blank wire visible. Touching the blank wire, or touching the metal parts of the screws is very dangerous.
- Connect the other end of the LIFE wire (the part that comes from the mains plug of your device) to screw 4 (NC).
- Repeat the previous 3 steps for the other 3 light bulbs.
- Check all wiring.
- Connect another external power source (battery pack) to your Arduino.

118_Relay-4x.ino

```
#define RELAYS 4

void setup()
{
  for (int RELAY = 1; RELAY <= RELAYS; RELAY++)
  {
    pinMode(RELAY + 1, OUTPUT);
  }
}

void loop()
{
  for (int RELAY = 1; RELAY <= RELAYS; RELAY++)
  {
    digitalWrite(RELAY + 1, HIGH);
    delay(500);
    digitalWrite(RELAY + 1, LOW);
    delay(500);
  }
  delay(1500);
}
```

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with or knowledge of mains electricity, consult someone with the right skills.

200. 8x Relay 5V[1] Board HL-58S V1.2

With this relay you can use your Arduino to switch 8 different devices individually to a maximum of 10 A at 250 VAC each. So you could switch lightbulbs and other household or workshop equipment connected to mains electricity.

Take extra care, because working with mains electricity CAN BE VERY DANGEROUS.

200.1. Specifications 8x Relay 5V Board HL-58S V1.2

- SRD-05VDC-SL-C
- Input: 3-48 V
- Load 10 A:
  - 250 VAC
  - 125 VAC
  - 30 VDC
  - 28 VDC
- 3 screws which you can use as a Changeover switch or 1 NO (normally open) and 1 NC (normally closed) pair.

200.2. Connections 8x Relay 5V Board HL-58S V1.2

10 pin header

This 10 pin header row is ONLY TO BE USED with low voltages like on the Arduino and SHOULD NEVER BE CONNECTED TO MAINS ELCTRICITY!!!

Pin nr	Name	Description	Arduino pin
1	GND	Ground	External Power Supply GND & Arduino GND
2..9	IN1..IN8	Relay 1..8	Any Digital port
10	Vcc	5V	External Power Supply 5V

8 Screws terminals OUT1..OUT4

These connections are to be used with mains electricity (max. 10A at 250V) so be extra careful!!!

Pin nr	Name	Description
1	NC	Normally Closed

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with mains electricity, consult someone with the right skills.

2	COM	Common
3	NO	Normally Open

- Case 1: The device should normally be switched off
  Use screw 2 and 3. Giving a HIGH signal on one of the relay pinheaders, the corresponding device will be switched to ON.
- Case 2: The device should normally be switched on.
  Use screw 2 and 3. Giving a HIGH signal on one of the relay pinheaders, the corresponding device will be switched to OFF

200.3. Libraries needed for 8x Relay 5V Board HL-58S V1.2

None needed.

200.4. Sample 8x Relay 5V Board HL-58S V1.2

The following sketch will turn on and off 8 light bulbs in a row.

Sample Connections[1]

- Send your sketch to the Arduino
- Disconnect your laptop/computer from the Arduino (as a precaution for a faulty relay).
- Connect GND to GND of an external Power Supply and to GND of the Arduino
- Connect IN1..IN8 to D2..D9
- Connect Vcc to 5V of an external Power Supply
- Remove your mains electricity connector from the wall outlet.
- Connect one end of the LIFE wire (coming from a light bulb) to screw 3 (Common). Remove just enough isolation, so there is not too much blank wire visible. Touching the blank wire, or touching the metal parts of the screws is very dangerous.
- Connect the other end of the LIFE wire (the part that comes from the mains plug of your device) to screw 1 (NC).
- Repeat the previous 3 steps for the other 7 light bulbs.
- Check all wiring.
- Connect another external power source (battery pack) to your Arduino.

139_Relay-8x.ino

```
#define RELAYS 8

void setup()
{
  for (int RELAY = 1; RELAY <= RELAYS; RELAY++)
  {
    pinMode(RELAY + 1, OUTPUT);
  }
}

void loop()
{
  for (int RELAY = 1; RELAY <= RELAYS; RELAY++)
  {
    digitalWrite(RELAY + 1, HIGH);
    delay(500);
    digitalWrite(RELAY + 1, LOW);
    delay(500);
  }
  delay(1500);
}
```

[1] Working with mains electricity can be very dangerous. I'm not responsible for any errors in this documentation. If you have not enough experience with or knowledge of mains electricity, consult someone with the right skills.

201. Optocoupler MOC3023

This optocoupler isolates the output from your Arduino with the device you want to switch ON/OFF.

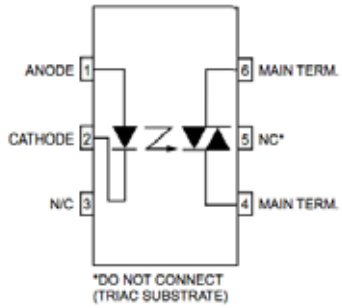
201.1. Specifications Optocoupler MOC3023

- U_f=1.2 V.
- I_f=10 mA.
- Current limiting resistor to be used with Arduino's 5V = 390 ohm[1].
- Volt peak at terminal max 400V.

201.2. Datasheet Optocoupler MOC3023

- http://www.futurlec.com/LED/MOC3023.shtml

201.3. Connections Optocoupler MOC3023



Pin nr	Name	Description	Arduino pin
1	Anode	Anode input	Any digital port
2	Cathode	Cathode input	Ground through a 390 ohm resistor
3	NC	Not connected	-
4	Main term.	Main terminal output	Used to switch a device
5	NC	Not connected Do not connect (TRIAC substrate)	-
6	Main term.	Main terminal output	Used to switch a device

201.4. Libraries needed for Optocoupler MOC3023

None needed.

[1] Calculated with the calculator on the following website: http://led.linear1.org/1led.wiz

201.5. Sample Optocoupler MOC3023

The following sketch turns on a photoflash after pressing the spacebar followed by the Enter key.

In my setup, the optocoupler seemed to stick once in a while (connection stayed closed), so the optocoupler didn't responded to succeeding trigger signals. By disconnecting and connecting the flash this could be solved. In "293 High Speed Photography" I've described how to use a relay to temporary disconnect the flash after each trigger.

- Connect 1 to D4.
- Connect 2 to one end of a 390 ohm resistor.
- Connect the other end of the 390 ohm resistor to GND.
- Connect 4 to one pin of the flash cable.
- Connect 6 to the other pin of the flash cable.
- Change the settings in Serial Monitor to "No line ending" otherwise you will get multiple triggers instead of one (double triggers at setting "Newline" or "Carriage return" or even triple triggers at the setting "Both NL & CR").

119_Optocoupler.ino

```
int FLASH=4;
int LED=13;

void setup()
{
  pinMode(FLASH, OUTPUT);
  digitalWrite(FLASH, LOW);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  Serial.begin(9600); // open serial
  Serial.println("Press the spacebar followed by the Enter key");
}

void loop()
{
  int key;

  while (Serial.available() > 0)
  {
    int key = Serial.read();

    if (key = ' ')
    {
      digitalWrite(FLASH, HIGH);
      digitalWrite(LED, HIGH);
      Serial.println("Flash is triggered");
      delay(100);
      digitalWrite(FLASH, LOW);
      digitalWrite(LED, LOW);
      break;
    }
    else
    {
      Serial.println("Press the spacebar followed by the Enter key");
    }
  }
}
```

202. Optocoupler 817C

This optocoupler isolates the output from your Arduino with the device you want to switch ON/OFF.

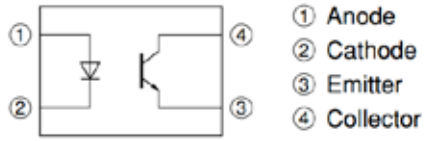
202.1. Specifications Optocoupler 817C

- U_f=5 V.
- I_f=5 mA.
- Current limiting resistor to be used with Arduino's 5V = 220 ohm.

202.2. Datasheet Optocoupler 817C

- http://henrysbench.capnfatz.com/wp-content/uploads/2015/05/817C-Optocoupler-Datasheet.pdf
- More information can be found at: https://www.instructables.com/id/Isolating-circuits-from-your-arduino-with-optocoup/

202.3. Connections Optocoupler 817C



Pin nr	Name	Description	Arduino pin
1	Anode	Anode input	Any digital port
2	Cathode	Cathode input	Ground through a 220 ohm resistor
3	Emiter		Used to switch a device
4	Collector		Through a 220 ohm resistor to the device to be switched.

202.4. Sample Optocoupler MOC3023

The following sketch turns on a photoflash after pressing the spacebar followed by the Enter key.

- Connect 1 to D4.
- Connect 2 to one end of a 220 ohm resistor.
- Connect the other end of the 220 ohm resistor to GND.
- Connect 3 to one pin of the flash cable.
- Connect 4 to one end of a second 220 ohm resistor
- Connect the other end the second 220 ohm resistot to the other pin of the flash cable.
- Change the settings in Serial Monitor to “No line ending” otherwise you will get multiple triggers instead of one (double triggers at setting “Newline” or “Carriage return” or even triple triggers at the setting “Both NL & CR”).

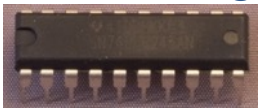
119_Optocoupler.ino

```
int FLASH=4;
int LED=13;

void setup()
{
  pinMode(FLASH, OUTPUT);
  digitalWrite(FLASH, LOW);
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
  Serial.begin(9600); // open serial
  Serial.println("Press the spacebar followed by the Enter key");
}

void loop()
{
  int key;

  while (Serial.available() > 0)
  {
    int key = Serial.read();

    if (key = ' ')
    {
      digitalWrite(FLASH, HIGH);
      digitalWrite(LED, HIGH);
      Serial.println("Flash is triggered");
      delay(100);
      digitalWrite(FLASH, LOW);
      digitalWrite(LED, LOW);
      break;
    }
    else
    {
      Serial.println("Press the spacebar followed by the Enter key");
    }
  }
}
```

Logical Level shifters

Not all sensors and actuators use 5V logical levels, some use 3.3V. Logical Level shifters can shift from 3.3V to 5V and vice versa.

203. 8-bit Logic Level Shifter 74LVC245



With this IC you can connect the outputs of 8 5V devices to the input ports of a 3.3V system like the Raspberry Pi.

203.1. Specifications 8-bit Logic Level Shifter 74LVC245

- Operates from 1.65V to 3.6V
- Converts 8 inputs from up to 5.5V to VCC (1.65V to 3.6V)
- Outputs are always
- Supports mixed-mode signal on all ports (5V I/O voltage with 3.3V VCC).
- To connect 8 5V devices to a 3.3V system like the Raspberry Pi.
- Not suitable for I2C or 1-wire devices
- Should be suitable for SPI and TTL serial

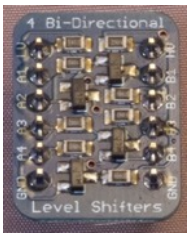
203.2. Datasheet 8-bit Logic Level Shifter 74LVC245

- Datasheet 74LVC245
  https://cdn-shop.adafruit.com/datasheets/sn74lvc245a.pdf

203.3. Connections 8-bit Logic Level Shifter 74LVC245

Pin nr	Name	Description	
1	DIR	HIGH: A => B LOW: B => A	3.3V (A=> B)
2	A1	Input or Output, depending on DIR	output 5V device
3	A2	Input or Output, depending on DIR	output 5V device
4	A3	Input or Output, depending on DIR	output 5V device
5	A4	Input or Output, depending on DIR	output 5V device
6	A5	Input or Output, depending on DIR	output 5V device
7	A6	Input or Output, depending on DIR	output 5V device
8	A7	Input or Output, depending on DIR	output 5V device
9	A8	Input or Output, depending on DIR	output 5V device
10	GND	Ground	GND
11	B8	Input or Output, depending on DIR	input 3.3V system
12	B7	Input or Output, depending on DIR	input 3.3V system
13	B6	Input or Output, depending on DIR	input 3.3V system
14	B5	Input or Output, depending on DIR	input 3.3V system
15	B4	Input or Output, depending on DIR	input 3.3V system
16	B3	Input or Output, depending on DIR	input 3.3V system
17	B2	Input or Output, depending on DIR	input 3.3V system
18	B1	Input or Output, depending on DIR	input 3.3V system
19	not-OE	not-Output Enable LOW: enable HIGH (through pullup): disable	GND
20	VCC	VCC	3.3V

204. Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor



This little board converts logic levels between a low and a high voltage and vice-versa.

204.1. Specifications Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor

- 4 bi-directional channels
- High/Low voltage is set by connecting a reference voltage to HV/LV (1.8-10V)
- Step up/down from LV signals to HV
- Connects a 3.3V device to a 5V system (like most Arduino's)
- Connects a 5V device to a 3.3V system (like the Raspberry Pi)
- I2C, TTL Serial and slow SPI (<2 MHz)
- Based on 4 BSS138 FET's with 10K pullups.

204.2. Datasheet Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor

- Datasheet BSS138 FET's
  https://www.fairchildsemi.com/datasheets/BS/BSS138.pdf

204.3. Connections Adafruit 4 chan. I2C safe bi-directional Logic Level Convertor

Pin nr	Name	Description	Arduino pin
1	GND	Ground	Connect both GND pins to GND
2	B4	HIGH O/I (Output/Input) paired with A4 as LOW I/O (Input/OUT)	Any digital port or ext. 5V device
3	B3	HIGH O/I paired with A3 as LOW I/O	Any digital port or external 5V device
4	B2	HIGH O/I paired with A2 as LOW I/O	Any digital port or external 5V device
5	B1	HIGH O/I paired with A1 as LOW I/O	Any digital port or external 5V device
6	HV	Reference voltage HIGH	5V
7	LV	Reference voltage LOW	3.3V
8	A1	LOW I/O paired with B1 as HIGH O/I	Any digital port or external 3.3V device
9	A2	LOW I/O paired with B2 as HIGH O/I	Any digital port or external 3.3V device
10	A3	LOW I/O paired with B3 as HIGH O/I	Any digital port or external 3.3V device
11	A4	LOW I/O paired with B4 as HIGH O/I	Any digital port or external 3.3V device
12	GND	Ground	Connect both GND pins to GND

205. 4 channel bi-directional Logic Level Convertor

This little board converts logic levels between a low and a high voltage and vice-versa.

205.1. Specifications 4 channel bi-directional Logic Level Convertor

- 4 channels
- Bi-directional
- High/Low voltage is set by connecting a reference voltage to HV/LV (1.8-5V)
- Step up from LV signals to HV
- Step down from HV signals to LV
- Connects a 3.3V device to a 5V system (like most Arduino's)
- Connects a 5V device to a 3.3V system (like the Raspberry Pi)
- Probably I2C safe for low speed?
- Probably based on 4 KSA1298 PNP transistors and 10K pullups.

205.2. Datasheet 4 channel bi-directional Logic Level Convertor

- Datasheet KSA1298 PNP transistor
  https://www.fairchildsemi.com/datasheets/KS/KSA1298.pdf

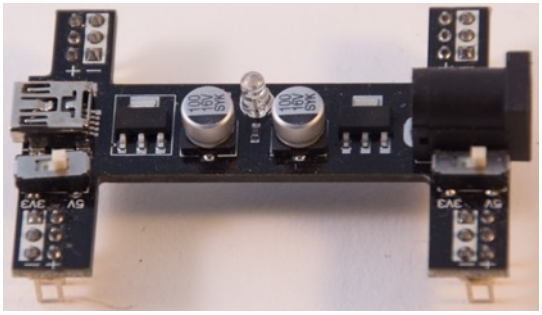
205.3. Connections 4 channel bi-directional Logic Level Convertor

Pin nr	Name	Description	Arduino pin
1	LV1	LOW I/O (Input/Output) paired with HV1 as HIGH O/I (Output/Input)	Any PWM or external 3.3V sensor/actuator
2	LV2	LOW I/O paired with HV2 as HIGH O/I	Any PWM or external 3.3V sensor/actuator
3	LV	Reference voltage LOW	3.3V
4	GND	Ground	Connect both GND pins to GND
5	LV3	LOW I/O paired with HV3 as HIGH O/I	Any PWM or external 3.3V sensor/actuator
6	LV4	LOW I/O paired with HV4 as HIGH O/I	Any PWM or external 3.3V sensor/actuator
7	HV4	HIGH O/I paired with LV4 as LOW I/O	Any PWM or external 3.3V sensor/actuator
8	HV3	HIGH O/I paired with LV3 as LOW I/O	Any PWM or external 3.3V sensor/actuator
9	GND	Ground	Connect both GND pins to GND
10	HV	Reference voltage HIGH	5V
11	HV2	HIGH O/I paired with LV2 as LOW I/O	Any PWM or external 3.3V sensor/actuator
12	HV1	HIGH O/I paired with LV1 as LOW I/O	Any PWM or external 3.3V sensor/actuator

Power supplies

There are several ways to supply power to your Arduino and other components. Some of them are described here.

206. Black Wings breadboard power regulator



Black wings when placed on the bus strips of a breadboard, provides regulates 5V or 3.3V to the bus strips, regardless of the input power voltage (must be higher than 5V/3.3V).

Input power can be either a power supply or a USB cable connected to a USB power source (or to a computer).

Each side of the Black Wings consist of 3 connectors for + (VCC) and 3 connectors for – (GND) to give physical stability.

Make sure that the + connector is placed on the red bus-strip of your breadboard and the – connector is placed on the black/blue bus-strip of your breadboard.

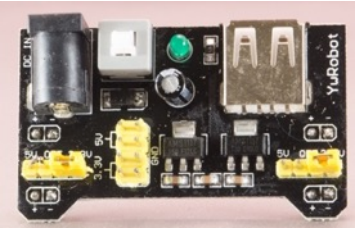
206.1. Specifications Black Wings breadboard power regulator

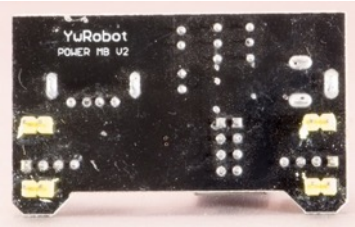
- Input: 4.8V – 15V => Output 3.3V 1A
- Input: 6.5V – 15V => Output 5V 1A
- Input: USB mini => Output 3.3/5V 500 mA?
- Left and right independent, either 3.3V or 5V.
- AMS1117-3.3, 1A low dropout voltage regulator.
- AMS1117-5.0, 1A low dropout voltage regulator.
- Suitable for 2 different breadboard sizes:
  - Width 20 pins (0.1 inch spacing) = 4.8 cm.
  - Width 22 pins (0.1 inch spacing) = 5.3 cm.

206.2. Datasheet

- AMS1117 http://www.advanced-monolithic.com/pdf/ds1117.pdf

207. YuRobot breadboard power regulator





Breadboard power regulators, when placed on the bus strips of a breadboard, provides regulated 5V or 3.3V to the bus strips, regardless of the input power voltage (must be higher than 5V/3.3V).

Input power can be either a power supply or a USB cable connected to a USB power source (or to a computer).

Each side of the Black Wings consist of 3 connectors for + (VCC) and 3 connectors for – (GND) to give physical stability.

Make sure that the + connector is placed on the red bus-strip of your breadboard and the – connector is placed on the black/blue bus-strip of your breadboard.

207.1. Specifications Black Wings breadboard power regulator

- Input: 6,5V – 12V => Output 5V/3.3V 700 mA
- Output on USB port => Output 5V (to feed an Arduino)
- Power switch
- Left and right independent, either 3.3V or 5V.
- AMS1117-3.3, 1A low dropout voltage regulator.
- AMS1117-5.0, 1A low dropout voltage regulator.
- Suitable for the following breadboard size:
    - Width 20 pins (0.1 inch spacing) = 4.8 cm.

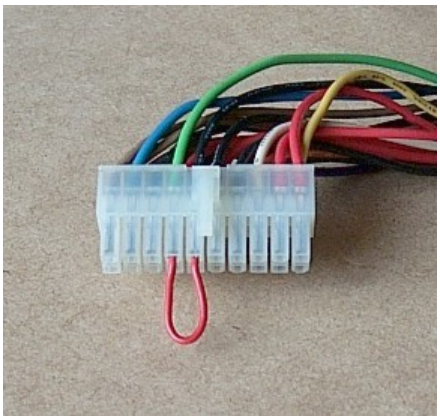
207.2. Datasheet

- AMS1117 http://www.advanced-monolithic.com/pdf/ds1117.pdf

208. ATX Power Supply (PSU)

For some projects you'll need an external power supply. A great power source is the PSU (Power Supply Unit) of an old computer. These PSU's have several regulated +5V and +12V connections available. If you use such a PSU without a mother board and without the ON/OFF switch on your computer case, you'll need a simple mod to switch on your PSU.

Locate the 20/24 pin ATX motherboard connector and connect the green cable (only one) with one of the black cables (ground). Like in the following picture. Be careful, remove the Power cord (230 V cable) before you do this and only afterwards connect the Power cord to mains again. If the PSU fan won't start, immediately remove the Power cord again and check you wiring!

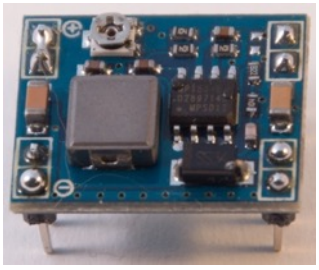


208.1. Connections

The connectors on the 20/24 pin ATX motherboard and on the floppy drive molex connectors use color coded wiring:

Color	**Power**
Green	Power ON
Orange	3.3 V
Red	+5 V
Yellow	+12 V
White	-5 V (minus 5 V)
Black	Ground

209. DC Step-Down Adjustable Power module



This module is a small and cheap DC Step-Down module, given a slightly higher input voltage, it regulates a specific output Voltage. For example you can get a steady 5,0 V output for your components with 2x 3,7 V 18650 LiOn batteries (2x3,7). Even when the voltage of your batteries drops to just above 6 V, the 5.0 V output voltage remains steady.

209.1. Specifications DC Step-Down Adjustable Power module

- Input Voltage: 4.5-28V
- Output Voltage: 0.8-20V
- Output Current: 3A (Max.)

209.2. Datasheet DC Step-Down Adjustable Power module

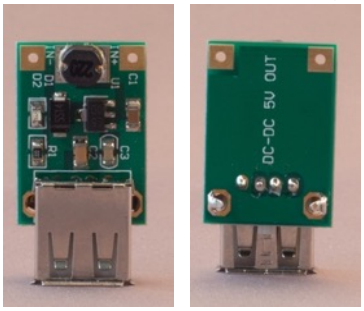
- http://www.electrodragon.com/w/images/d/d3/MP1584.pdf

209.3. Connections DC Step-Down Adjustable Power module

Pin nr	Name	Description
1	IN-	Ground
2	IN+	Input power source non regulated
3	OUT-	Ground
4	OUT+	Output power regulated

By turning the little Potentiometer and a Volt-meter connected to OUT- and OUT+ you can set the required output Voltage.

210. DC to DC Step-Up Boost Power Supply with USB



This module will step up your power supply to 5V, making it possible to use batteries to feed your Arduino.

210.1. Specifications DC Step-Up Adjustable Power module

- Input Voltage: 1-5V
- Output Voltage: 5V
- Output Current: 300 mA
- Chip: E50D (probably a ce8301 chip)

211. DC to DC Step-Up Boost Power Supply with USB


This module will step up your power supply to 5V, making it possible to use batteries to feed your Arduino.

211.1. Specifications DC Step-Up Adjustable Power module

- Input Voltage: 0.9 - 5V
- Output Voltage: 5V
- Output Current: unknown
- Chip: E50D (probably a ce8301 chip)

USB to Serial adapters

In this section you will find several USB to Serial adapters. You can use them to program Arduino's without an onboard USB to serial adapter, like the mini Pro or some ESP8266 boards.

212. PL203HX USB to TTL Serial adapter

With this cable/adapter you can add a serial port connection to an Arduino board that lacks an UART, see chapters 19 “AVR Development board as Arduino”, 20 “Arduino on a breadboard” and chapter 21 “Attiny45/Attiny85”.

212.1. Specifications PL203HX USB to TTL Serial adapter

- PL203HX chip.
- USB connector, no cable needed between adapter and computer.

This adapter can’t be used for uploading sketches from Arduino IDE to the ESP8266 modules or for flashing firmware with esptool-ck to the same ESP8266 modules. https://github.com/esp8266/Arduino/issues/710 .

Use an FTDI cable (FT232RL) or a USB-Serial cable with a CP2102 chip instead.

212.2. Datasheet PL203HX USB to TTL Serial adapter

- http://v-comp.kiev.ua/download/pl2303HX.pdf

212.3. Connections USB to TTL Serial cable

Pin nr	Name	Description
1	5V	5 Volt
2	GND	Ground
3	TxD	Transmit Data
4	RxD	Receive Data
5	3V3	3.3 Volt

213. P203HX USB to TTL Serial cable

With this cable/adapter you can add a serial port connection to an Arduino board that lacks an UART, see chapters 18 “Boarduino”, 19 “AVR Development board as Arduino”, 20 “Arduino on a breadboard” and chapter 21 “Attiny45/Attiny85”.

213.1. Specifications PL203HX USB to TTL Serial adapter

- PL203HX chip.
- USB cable is integrated.

This cable can’t be used for uploading sketches from Arduino IDE to the ESP8266 modules or for flashing firmware with esptool-ck to the same ESP8266 modules. https://github.com/esp8266/Arduino/issues/710 .

Use an FTDI cable (FT232RL) or a USB-Serial cable with a CP2102 chip instead.

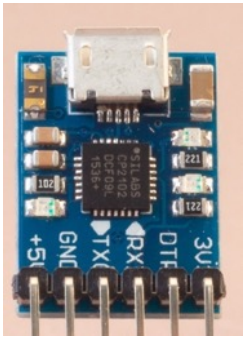
213.2. Datasheet PL203HX USB to TTL Serial adapter

- http://v-comp.kiev.ua/download/pl2303HX.pdf

213.3. Connections USB to TTL Serial cable

Pin nr	Name	Description	Color
1	VCC out	VCC (5V)	Red
2	GND	Ground	Black
3	TxD	Transmit Data	Green
4	RxD	Receive Data	White

214. CP2102 USB to TTL Serial adapter



With this cable/adapter you can add a serial port connection to an Arduino board that lacks an UART, see chapters 18 “Boarduino”, 19 “AVR Development board as Arduino”, 20 “Arduino on a breadboard” and chapter 21 “Attiny45/Attiny85”.

214.1. Specifications CP2102 USB to TTL Serial adapter

- CP2102 chip.
- Micro USB connector, USB cable needed between adapter and computer.

214.2. Datasheet CP2102 USB to TTL Serial adapter

- https://www.sparkfun.com/datasheets/IC/cp2102.pdf

214.3. Drivers

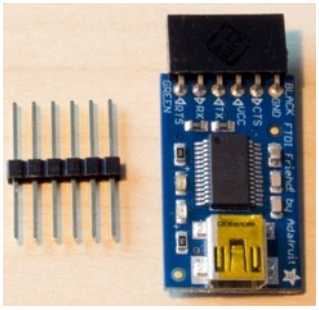
You can find the drivers for, Windows, Linux, OSx and Android at the following link:
https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

214.4. Connections USB to TTL Serial cable

Bottom connector

Pin nr	Name	Description
1	+5	5 Volt
2	GND	Ground
3	TxD	Transmit Data
4	RxD	Receive Data
5	DTR	Data Terminal Ready
6	3V3	3.3 V

215. FTDI friend by Adafruit's (USB to TTL Serial adapter)




This board is basically a USB to TTL Serial board, with 2 extra connectors (CTS and RTS).

215.1. Specifications FTDI friend by Adafruit's (USB to TTL Serial adapter)

Be careful not to buy an adapter with a counterfeit FTDI chip[1]

- FTDI FT232RL chip
- Mini USB connector, USB cable needed between adapter and computer.
- CTS connector
- Signal logic level 3.3 or 5V (solder jumpers)
- VCC out: 3.3 or 5V through (solder jumpers)
- RTS or DTR (solder jumpers), with a simple mod, you can use both RTS and DTR. I've soldered a small wire and a female header to the left part of the solder pad, as can be seen on the picture above and the detailed picture below. This solder pad is now connected to DTR and pin 5 to RTS.

215.2. Datasheet FTDI friend by Adafruit's (USB to TTL Serial adapter)

- More information about the FTDI Friend can be found at: https://learn.adafruit.com/downloads/pdf/ftdi-friend.pdf.

[1] FTDI is fighting a war against counterfeit chips, in chapter "217 Counterfeit FTDI FT232RL chips" you'll find information about this problem and possible solutions.

215.3. Connections FTDI friend by Adafruit's (USB to TTL Serial adapter)

Pin nr	Name	Description
1	GND	Ground
2	CTS	Clear to Send
3	VCC	3.3 or 5V determined by solder pad
4	Tx	Transmit Data 3.3 or 5 V determined by solder pad
5	Rx	Receive Data
6	RTS/DTR	Request to Send or Data Terminal Ready Function is determined by solder pad

216. FTDI FT232L USB to TTL Serial adapter


This board is a USB to TTL Serial board, with lots of extra connection capabilities (you need to add extra headers for this.

216.1. Specifications FTDI FT232RL USB to TTL Serial adapter

Be careful not to buy an adapter with a counterfeit FTDI chip[1]

- FT232RL chip
- DTR and CTS connector
- Signal logic level and VCC 3.3 or 5V (through jumper)
- All FT232RL chip pins are available after soldering pin headers.

216.2. Datasheet FTDI FT232RL USB to TTL Serial adapter

- http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf.

216.3. Connections FTDI FT232RL USB to TTL Serial adapter

This board has 3 header rows for all FT232RL chip pins and 1 jumper row to determine the voltage levels.

Jumper

- 5V Connect Left and Middle pin
- 3.3V Connect Right and Middle pin

Bottom header row

Pin nr	Name	Description
1	DTR	Data Terminal Ready
2	Rx	Receive Data
3	Tx	Transmit Data
4	VCC	3.3 or 5V
5	CTS	Clear to Send
6	GND	Ground

[1] FTDI is fighting a war against counterfeit chips, in chapter "217 Counterfeit FTDI FT232RL chips" you'll find information about this problem and possible solutions.

Left header row

Pin nr	Name	Description
1	PWREN	
2	TEN	TXDEN Transmit enable for RS485 designs?
3	SLEEP	USB suspend mode
4	CTS	Clear to Send
5	3.3V	3.3 Volt
6	5V	5 Volt
7	RXL	Receive Data LED drive
8	TXL	Transmit Data LED drive
9	GND	Ground

Right header row

Pin nr	Name	Description
1	DCD	Data Carrier Detect
2	RSD	?
3	GND	Ground
4	RI	Ring Indicator Control Input
5	RXD	Receive Data
6	VCC	VCC
7	RTS	Request to Send
8	DTR	Data Terminal Ready
9	TXD	Transmit Data

217. Counterfeit FTDI FT232RL chips

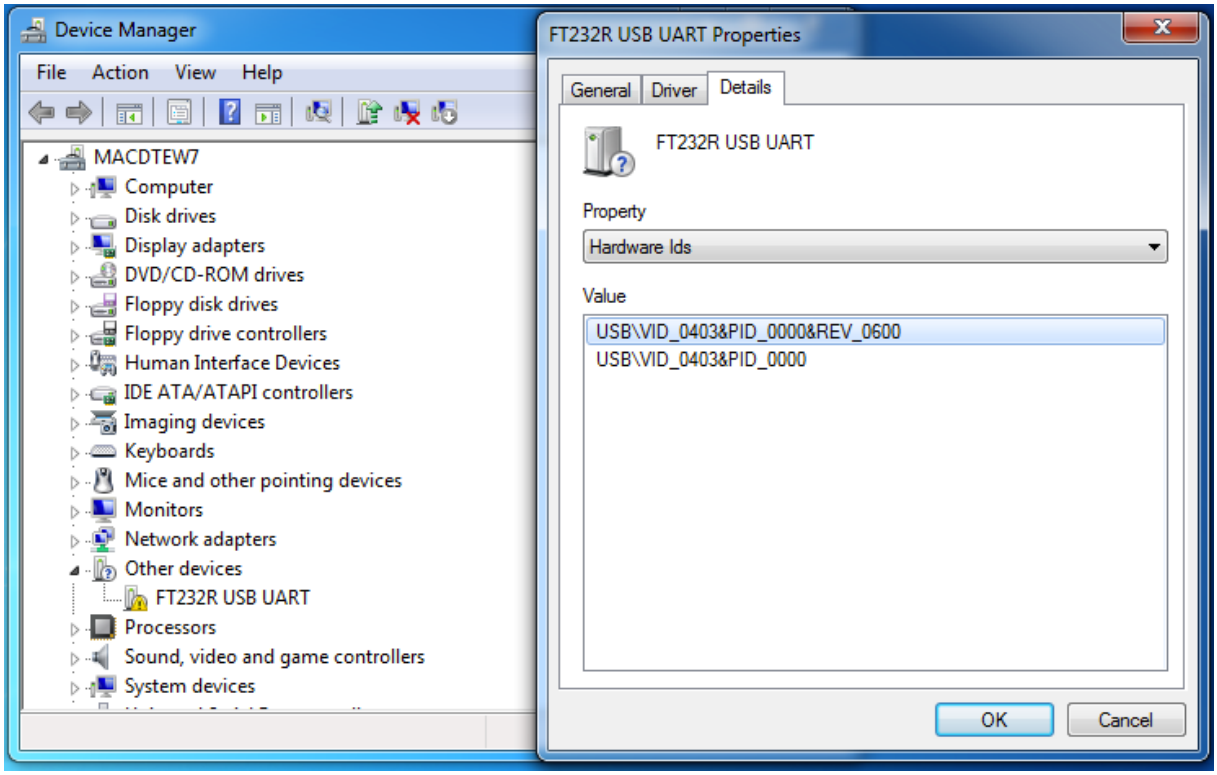
The FTDI FT232RL chip is very popular for USB to TTL Serial conversion. A lot of companies sell devices with counterfeit FTDI chips. Some try to save some money, others are not aware of the fact that they've used fake FTDI chips. Those counterfeits are functional, but made by other companies. In an attempt to fight the war against these counterfeited FTDI chips, FTDI has published a Windows driver to disable these fake FTDI chips. This doesn't only affect USB to TTL Serial adapters, but also some Arduino Nano and other equipment using these fake FTDI chips.

Up till now this hasn't affected Linux nor has it affected OSx.

217.1. Problem description

If you are using a device with a counterfeited FTDI chip and the device driver you're using is higher than 2.10.0, you'll be facing one of the following cases:

1. If you've updated your Windows device driver to version 2.11.0-2.12.0, your device won't show up as a COM port anymore. This driver software bricked your device and it cannot be used on any OS before you de-brick it. To accomplish this, the USB PID of fake FTDI chips will be set to 0 instead of the normal value 6001. In Device Manager, your device will be shown as an FT232R USB UART with an exclamation mark:

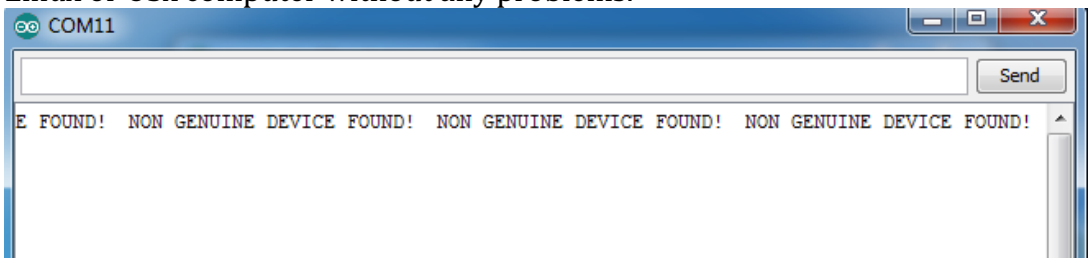


This software bricking action from FTDI was highly criticized. Arguments used against this bricking were:

   a. Consumers were not always aware about the fact they were using fake FTDI chips.
   b. The driver update was part of Windows Update, so consumers didn't voluntarily update to this new driver.

As a response to this, FTDI decided to update the driver again. This new driver doesn't brick you fake FTDI chip, as you can see in situation 2.

2. If you've updated your FTDI device driver to version 2.12.4 and higher, your device will show up as a COM port, but can't be used for USB-serial communication any more. Output on Serial monitor shows the text NON GENUINE DEVICE FOUND! repeatedly. The device is not bricked, but the driver refuses to communicate with it. In this case you can connect your device on a Linux or OSx computer without any problems.



217.2. Solution Case 1

The solution for case 1 involves, removing the newest device driver, installing an older driver and then de-bricking your device.[1]

Although I've tested the following procedure and files, you do this at your own risk.

You will probably have to replace the new drivers with the older version for every FTDI device you own, to prevent that those other FTDI devices will install the newer drivers again.

Uninstall current FTDI drivers (case 1)

- Make sure your device with the fake FTDI chip is connected.
- Download CDM uninstaller from FTDI website: http://www.ftdichip.com/Support/Utilities/CDMUninstaller_v1.4.zip
- Extract this zip file and run the CDM Uninstaller GUI.
- Type in Vendor ID - 0403 and Product ID - 6001 and click on the ADD-button.
- Type in Vendor ID - 0403 and Product ID - 0000 and click on the ADD-button.
- Click on REMOVE DEVICES (ignore an error about failing to remove one of these drivers).
- Close the uninstaller.

Install 2.04.06 driver for a USB SERIAL CONVERTER (case 1)

- Make sure your device with the fake FTDI chip is still connected.
- Download driver version 2.04.06: http://www.mediafire.com/download/ku9tckrt8rpf62w/CDM_2.04.06_WHQL_Certified.7z [2]
- Unzip this driver.
- Open Device manager.
- Open PORTS (COM & LPT).
- Right click on the FT232R USB UART device and choose UPDATE DRIVER SOFTWARE.
- Click on BROWSE MY COMPUTER FOR DRIVER SOFTWARE.
- Click on LET MET PICK FROM A LIST OF DEVICE DRIVERS ON MY COMPUTER.
- Select UNIVERSAL SERIAL BUS CONTROLLERS and click on NEXT.
- Select Generic USB HUB and click on HAVE DISK.
- Click on BROWSE.
- Browse to the map where you've unzipped the 2.04.06 driver.
- Select FTDIBUS.INF and click on OPEN.
- Click on OK.
- Select the USB SERIAL CONVERTER and click on NEXT.
- Click on YES to approve this not recommended driver.
- Close all windows and reboot your computer.

---

[1] This procedure can be found at the following link: http://forum.gsmhosting.com/vbb/f684/ft232r-repair-official-guide-1875025/

[2] An alternative for this original FTDI driver is the open source driver Smarttronik, that can be downloaded at: http://smarttronik.com/index.php?controller=attachment&id_attachment=138

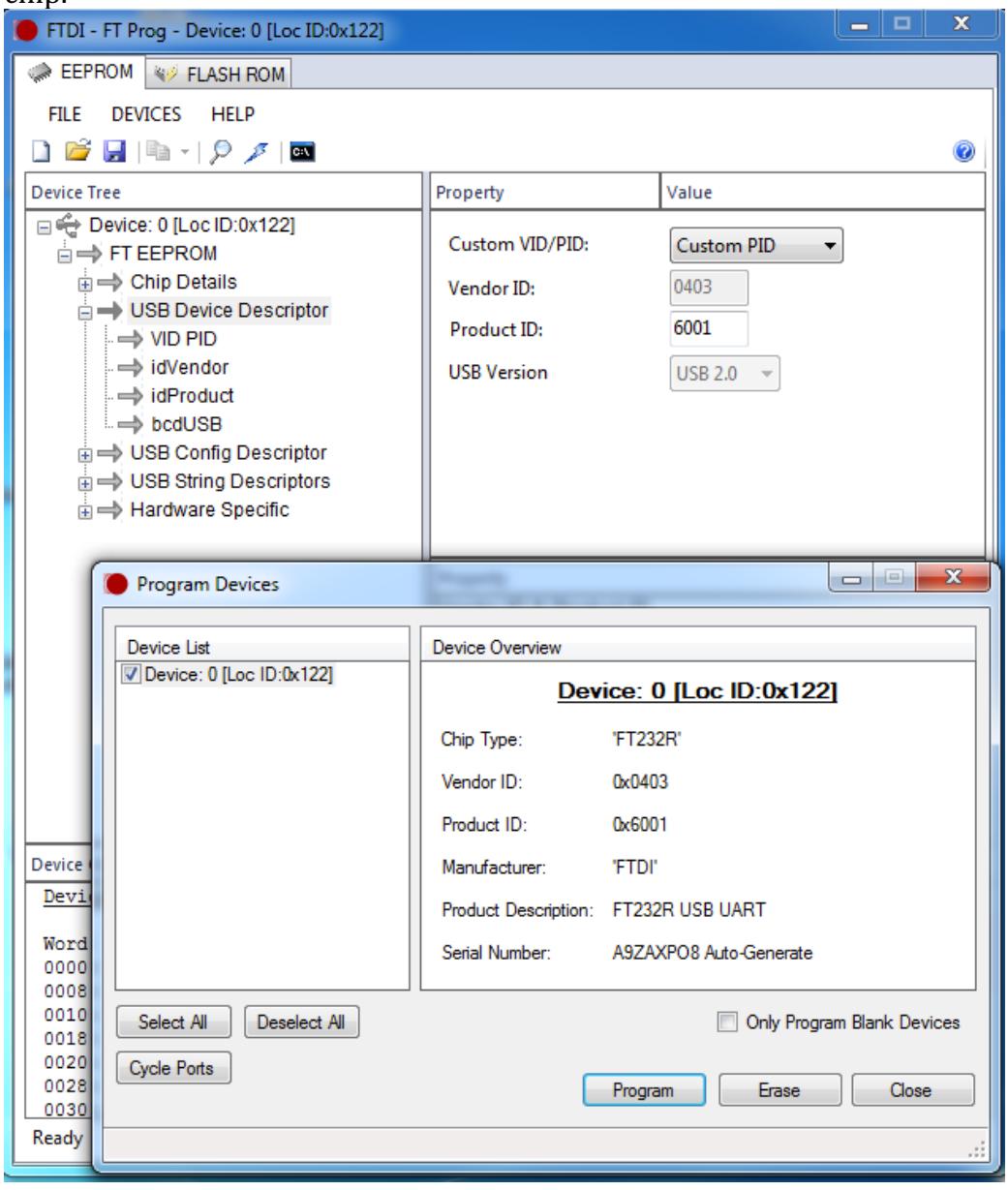
- Open device manager, you will see that the FT232R USB UART has been replaced by a USB Serial Port and that a USB SERIAL CONVERTER has been added to UNIVERSAL SERIAL BUS CONTROLLERS.

Install 2.04.06 driver for a USB Serial Port (case 1)

- Right click on the USB SERIAL PORT device and choose UPDATE DRIVER SOFTWARE.
- Click on BROWSE MY COMPUTER FOR DRIVER SOFTWARE.
- Click on LET MET PICK FROM A LIST OF DEVICE DRIVERS ON MY COMPUTER.
- Select PORTS (COM & LPT) and click on NEXT.
- Select COMMUNICATIONS PORT and click on HAVE DISK.
- Click on BROWSE.
- Browse to the map where you've unzipped the 2.04.06 driver.
- Select FTDIPORT.INF and click on OPEN.
- Click on OK.
- Select USB SERIAL PORT and click on NEXT.
- Click on YES to approve this not recommended driver.
- Close all windows and return to Device Manager.
- In PORTS (COM & LPT) a USB SERIAL PORT (COMx) has been installed, but still with USB PID of 0000 instead of 6001.
- Although your device is now functional on this computer, it is still software bricked on others (including other OS like Linux and OSx).

De-bricking (case 1)

- Download FT_PROG from the FTDI website: http://www.ftdichip.com/Support/Utilities/FT_Prog_v3.0.60.276%20Installer.zip
- Unzip this file and start the installer.
- Open FT_PROG.
- Press F5 to search for FTDI devices.
- Click on USB DEVICE DESCRIPTOR.
- At CUSTOM VID/PID, select CUSTOM PID.
- Change PRODUCT ID to 6001 and press CTRL-P to program this new value to the chip.



- Click on PROGRAM.
- Close FT_PROG.
- Disconnect your device and reconnect it again.
- The driver for your device will be reinstalled (strange it now says its 2.12.00.2!!!).
- Check the USB PID in device manager, it should say 6001.

- You are now up and running again, your device should now show up as a port in Arduino IDE.
- If you need it, you could now install driver version 2.08.14.7: http://www.mediafire.com/download/km17l279yni9vms/FTDI_2.08.14.7z.
- Don't update these drivers to anything higher than 2.08.14.7, or you'll need to repeat the previous steps!
- If you ever connect another FTDI device (genuine or fake) to your computer, you'll probably have to repeat these steps for that other FTDI device, but probable also for the first device.

217.3. Solution Case 2

The solution for case 2 involves, replacing the new drivers with an older version. [1]

Although I've tested the following procedure and files, you do this at your own risk.

You will probably have to replace the new drivers with the older version for every FTDI device you own, to prevent that those other FTDI devices will install the newer drivers again.

Install 2.08.14 driver for a USB SERIAL CONVERTER (case 2)

- Make sure your device with the fake FTDI chip is still connected.
- Download driver version 2.08.14: http://www.mediafire.com/download/km17l279yni9vms/FTDI_2.08.14.7z[2]
- Unzip this driver.
- Open Device manager.
- Open UNIVERSAL SERIAL BUS CONTROLLERS.
- Right click on USB SERIAL CONVERTER and choose UPDATE DRIVER SOFTWARE.
- Click on BROWSE MY COMPUTER FOR DRIVER SOFTWARE.
- Click on LET MET PICK FROM A LIST OF DEVICE DRIVERS ON MY COMPUTER.
- Click on HAVE DISK.
- Click on BROWSE.
- Browse to the map where you've unzipped the 2.04.06 driver.
- Select FTDIBUS.INF and click on OPEN.
- Click on OK.
- Select the USB SERIAL CONVERTER and click on NEXT.
- Click on Close
- The driver for the USB SERIAL CONVERTER has now been replaced by version 2.08.14.

Install 2.08.14 driver for a USB Serial Port (case 2)

- Right click on the USB SERIAL PORT device and choose UPDATE DRIVER SOFTWARE.
- Click on BROWSE MY COMPUTER FOR DRIVER SOFTWARE.
- Click on LET MET PICK FROM A LIST OF DEVICE DRIVERS ON MY COMPUTER.
- Click on HAVE DISK.
- Click on BROWSE.
- Browse to the map where you've unzipped the 2.04.06 driver.
- Select FTDIPORT.INF and click on OPEN.
- Click on OK.
- Click on CLOSE.

---

[1] This procedure can be found at the following link: http://forum.gsmhosting.com/vbb/f684/ft232r-repair-official-guide-1875025/

[2] An alternative for this original FTDI driver is the open source driver Smarttronik, that can be downloaded at: http://smarttronik.com/index.php?controller=attachment&id_attachment=138

- Reboot your computer.
- You are now up and running again, your device should now show up as a port in Arduino IDE.
- Don't update these drivers to anything higher than 2.08.14.7, or you'll need to repeat the previous steps!
- If you ever connect another FTDI device (genuine or fake) to your computer, you'll probably have to repeat these steps for that other FTDI device, but probable also for the first device.

Miscellaneous

In this section you will find miscellaneous components not falling in the other categories.

218. Adafruit Thermal Printer CSN-A2-T

A mini thermal printer, also called a receipt printer. This printer only needs 2 digital ports and an external power supply. A complete tutorial can be found at:
https://learn.adafruit.com/mini-thermal-receipt-printer

218.1. Specifications Adafruit Thermal Printer CSN-A2-T

- External power supply: 5-9 V DC 1.5A
- Protocol: TTL serial, 19200 baud
- Printing speed: 50-80 mm/s
- Resolution: 8 dots/mm, 384 dots/line
- Effective width: 48 mm
- Character set: ASCII & GB2312-80 (Chinese)
- Font: ANK:5x7, Chinese 12x24, 24x24
- Paper type: thermal paper
- Paper size:
  - Roll diameter: max 39 mm
  - Width: 57 mm
- Size: 111x65x57 mm
- Support for various barcodes through libraries.

218.2. Datasheet Adafruit Thermal Printer CSN-A2-T

- User manual:
  https://cdn-shop.adafruit.com/datasheets/CSN-A2+User+Manual.pdf
- Datasheet:
  https://cdn-shop.adafruit.com/datasheets/cashino+thermal+printer+a2.pdf

218.3. Connections Adafruit Thermal Printer CSN-A2-T

Pin nr	Name	Description	Arduino pin
1	GND	Ground (black)	ground on external power supply
2	VH	VCC	5-9 V on external power supply
3	GND	GND	ground on Arduino
4	Rx	Receive	Any digital I/O
5	Tx	Transmit	Any digital I/O

218.4. Libraries needed for Adafruit Thermal Printer CSN-A2-T

- Adafruit's Thermal Printer library through library manager.
- SoftwareSerial through library manager.

Library use explanation

```
#include "Adafruit_Thermal.h"
```
Include Adafruit's thermal printer library

```
#include "SoftwareSerial.h"
```
Include software serial, so you don't need the hardware serial. On the UNO, hardware serial (D0 + D1) is also used to flash the sketch to the Arduino. If you would use hardware serial on the UNO, you could get unwanted printout during flashing (waste of paper).

```
#define TX_PIN 6 // Arduino transmit  YELLOW WIRE  labeled RX on printer
```
This pin will be used to transmit data to the printer (should be connected to the yellow wire = Rx on the printer)

```
#define RX_PIN 5 // Arduino receive   GREEN WIRE   labeled TX on printer
```
This pin will be used to receive data from the printer (should be connected to the green wire = Tx on the printer)

```
SoftwareSerial mySerial(RX_PIN, TX_PIN);
```
Create 'mySerial' a new instance of the object SoftwareSerial using the pins RX_PIN, TX_PIN as defined in the lines above.

```
Adafruit_Thermal printer(&mySerial);
```
Create 'printer' a new instance of the object type Adafruit_Thermal, with the address of the object mySerial.

```
mySerial.begin(19200);
```
Initialize mySerial at 19200 baud.

```
  printer.begin();
```
Initialize the printer object.

```
  printer.justify('L');
```
Set justify to left (L = left, C = center and R = right)

```
  printer.println(F("Hello World"));
```
Prints the line 'Hello World' (what else?)

```
  printer.sleep();
```
Tell printer to sleep.

```
  delay(3000);
```
Let the printer sleep for 3 seconds.

```
  printer.wake();
```
I'm not sure why this line is needed, but without this line, the FEED button will not work.

```
  printer.setDefault();
```
Restore printer to defaults.

As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folders.

218.5. Sample Adafruit Thermal Printer CSN-A2-T

The following sketch will print the text Hello World on two separate lines.

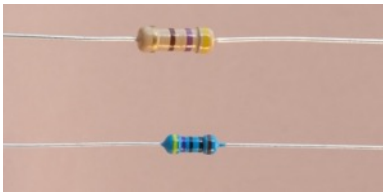
Sample Connections

- Connect VCC to 5-9V on an external power supply
- Connect GND to GND on an external power supply
- Connect GND (Black) to GND
- Connect Rx (Yelllow) to D6
- Connect Tx (Green) to D5

120_Thermal-Printer.ino

```
#include "Adafruit_Thermal.h"
#include "SoftwareSerial.h"

#define TX_PIN 6 // Arduino transmit  YELLOW WIRE  labeled RX on printer
#define RX_PIN 5 // Arduino receive   GREEN WIRE   labeled TX on printer

SoftwareSerial mySerial(RX_PIN, TX_PIN);
Adafruit_Thermal printer(&mySerial);

void setup()
{
  mySerial.begin(19200);
  printer.begin();
  printer.justify('L');
  printer.println(F("Hello"));
  printer.justify('C');
  printer.println(F("World"));

  printer.sleep();
  delay(3000);
  printer.wake();
  printer.setDefault();
}

void loop()
{
}
```

219. Resistor



A resistor is a passive electrical component used to limit current (by electrical resistance) and at the same time lower voltage levels in a circuit.

219.1. Connections

Both ends of a resistor are equally the same.

219.2. Color coding

The value of most resistors are color coded.

Color	1st band	2nd band	3rd band Only used with 5 bands!	Second to last band Multiplier	Last band Tolerance
Black	0	0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1.000	
Yellow	4	4	4	x10.000	
Green	5	5	5	x100.000	± 0.5%
Blue	6	6	6	x1000.000	± 0.25%
Violet	7	7	7	x10.000.000	± 0.10%
Grey	8	8	8	x100.000.000	± 0.05%
White	9	9	9		
Gold				x0.1	± 5%
Silver				x0.01	± 10%

Two useful mnemonics to remember the order of the color code:

- *"**B**ad **B**eer **R**ots **O**ur **Y**oung **G**uts **B**ut **V**odka **G**oes **W**ell - **G**et **S**ome **N**ow"*
- *"**B.B. ROY** of **G**reat **B**ritain had a **V**ery **G**ood **W**ife who wore **G**old and **S**ilver **N**ecklace"*
-

Black	Brown	Red	Orange	Yellow	Green	Blue	Violet	Grey	White	Gold	Silver	None
Bad	Beer	Rots	Our	Young	Guts	But	Vodka	Goes	Well	Get	Some	Now
B.	B.	R	O	Y of	Great	Britain had a	Very	Good	Wife who wore	Gold and	Silver	Necklace

The following mnemonic is Dutch:

"***Zij BR**acht **R**ozen **O**p **GE**rrits **GR**af **B**ij **V**ies **GR**auw **W**eer*"

Zwart	Bruin	Rood	Oranje	Geel	Groen	Blauw	Violet	Grijs	Wit
Zij	**BR**acht	**R**ozen	**O**p	**GE**rrits	**GR**af	**B**y	**V**ies	**GR**auw	**W**eer

Example 4 band code

Most 4 band resistors have a silver or gold last band that is used for tolerance. For these resistors it is therefore easy to determine the orientation.

For example a resistor with the following colors:
Red Red Brown Gold

1st band | 2nd band | Second to last band | Last band

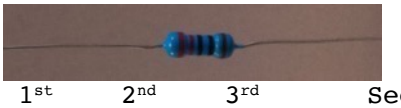
Multiple. | Tolerance

Red	Red	Brown	Gold
2	2	x10	5%

- This is a resistor with a 4 band code.
- The band on the right hand side is used for the tolerance. Gold = 5%.
- The left two bands are used for the value. Red Red = 22
- The second to last band is used for the multiplier. Brown = x10
- This makes: 22 x10 = 220 ohm +/- 5%

Example 5 band code

For example a resistor with the following colors:
Red Red Black Black Brown



1st band | 2nd band | 3rd band | Second to last band | Last band

Multiple. | Tolerance

Red	Red	Black	Black	Brown
2	2	0	x1	1%

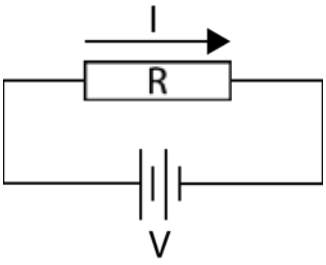
- This is a resistor with a 5 band code.
- The band on the right hand side is used for the tolerance. Brown = 1%
- The left three bands are used for the value. Red Red Black = 220
- The second to last band is used for the multiplier. Black = x1
- This makes: 220 x 1 = 220 ohm +/- 1%

Other examples

Colors	Result	Tolerance
Green Blue Black Black Brown	560 x1= 560 ohm	1 %
Red Red Orange Gold	22 x 1000 = 22.000 ohm	5%
Yellow Violet Brown Gold	47 x 10 = 470 ohm	5%
Blue Gray Black Silver	68 x 1 = 68 ohm	10%

219.3. Ohm's law

The relationship between the current I (through a resistor), the voltage V(over a resistor) and the resistance R (of the resistor) is represented in OHM's law:



$$I = \frac{V}{R}$$

From this you can derive:

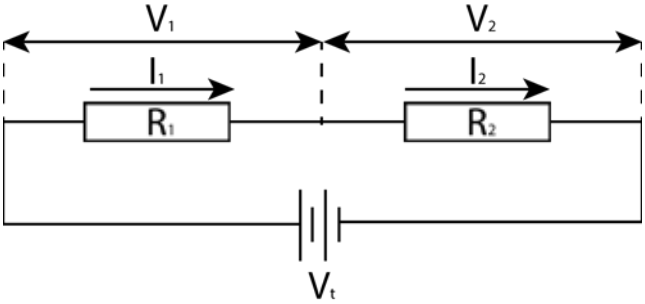
$$R = \frac{V}{I}$$

and

$$V = I \times R$$

219.4. Two resistors in series[1]

Useful formula's when working with two resistors in series:



Replacement resistor for R1 and R2:

$$Rt = R1 + R2$$

$$It = I1 = I2 = \frac{Vt}{Rt}$$

$$Vt = V1 + V2$$

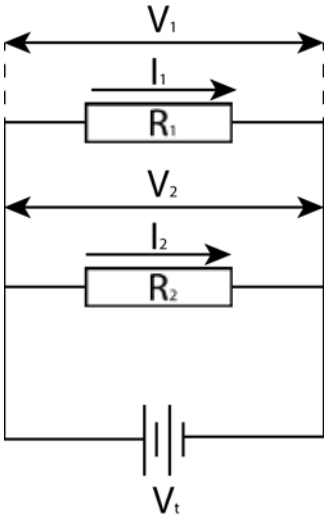
$$V1 = \frac{R1}{Rt} \times Vt = \frac{R1}{R1+R2} \times Vt$$

$$V2 = \frac{R2}{Rt} \times Vt = \frac{R2}{R1+R2} \times Vt$$

[1] You can also apply this to multiple resistors in series.

219.5. Two resistors parallel to each other:

Useful formula's when working with two resistors parallel to each other.



$$\frac{1}{Rt} = \frac{1}{R1} + \frac{1^{1}}{R2} \text{ or } Rt = \frac{R1 \times R2}{R1+R2}$$

$$It = I1 + I2 = \frac{Vt}{Rt}$$

$$I1 = \frac{V1}{R1} = \frac{Vt}{R1}$$

$$I2 = \frac{V2}{R2} = \frac{Vt}{R2}$$

$$Vt = V1 = V2$$

[1] For multiple Resistors the formula is $\frac{1}{Rt} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} \ldots\ldots +$

219.6. Current limiting resistor for LED's

A LED has a specific potential difference (diode forward Voltage) and a limited Current (diode forward current). In the datasheet of the LED you can find these values as V_F and I_F. On average:

- V_F=1.9 V
- I_F =20mA

Exceeding the value for I_F the LED will burnout.

When connecting a LED to the 5V an Arduino you must use a current limiting resistor as a voltage divider, so the potential difference over the LED is V_F and the current of I_F will not be exceeded.

In this example the current limiting resistor must have a potential difference of:

5-1.9=3.1 Volt

With a current of 20mA the resistor can be calculated by using Ohm's law:

$R = \frac{V}{I}$, with V=3.1 Volt and I=20 mA= 0.020 A ==> R=155 ohm.

At the following URL, you can find a nice calculator:

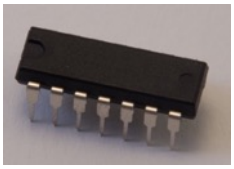
http://led.linear1.org/1led.wiz

This calculator rounds the resistance up to the next standard resistor value of 180 ohm.

In most diagrams a resistor of 220 ohm is being used instead. Increasing the resistor, will limit the current and will also decrease the intensity of the LED. However the difference in intensity when using 180 or 220 ohm is very small, so using 220 is much safer, without sacrificing intensity.[1]

[1] It is even safe to use this resistor with a diode forward voltage of 1.7 Volt and a diode forward current of 18mA!

220. Inverter 7404



This chips contains 6 inverters. A HIGH value on one of the inputs, will give a LOW level output and vice versa. Normally you could achieve the same result in your sketch. The reasons this chip was added to this document, is to spare two digital ports with the "DC/Stepper Motor Driver board L298n". With that board IN1 and IN2 should always be each other's inversion (same for IN3 and IN4). Without the 7404 chip you'll need two digital outputs per motor. When you direct one digital output to IN1 and also to one input of the 7404 and the output of the 7404 to IN2, you get the same result (IN1 and IN2 are each other's inversion). This way you can save 2 digital ports when driving a robot (and some code lines as well).

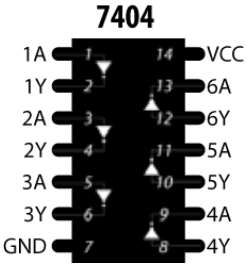
220.1. Specifications Inverter 7404

- 6 inputs and corresponding inverted output.

220.2. Datasheet Inverter 7404

- http://www.ti.com/lit/ds/symlink/sn74ls04.pdf

220.3. Connections Inverter 7404

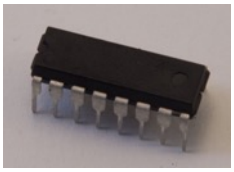


Pin nr	Name	Description	Arduino pin
1	1A	Input 1	Any Digital Output
2	1Y	Output 1 (Input 1 inverted)	To external device
3	2A	Input 2	Any Digital Output
4	2Y	Output 2 (Input 2 inverted)	To external device
5	3A	Input 3	Any Digital Output
6	3Y	Output 3 (Input 3 inverted)	To external device
7	GND	Ground	GND
8	4Y	Output 4 (Input 4 inverted)	To external device
9	4A	Input 4	Any Digital Output
10	5Y	Output 5 (Input 5 inverted)	To external device
11	5A	Input 5	Any Digital Output
12	6Y	Output 6 (Input 6 inverted)	To external device
13	6A	Input 6	Any Digital Output
14	VCC	VCC	5V

220.4. Libraries needed for Inverter 7404

None needed

221. Shift register 74HC595



This shift register is connected through a 3 pin serial interface with the Arduino and drives an 8-bit serial or parallel output. Daisy chaining several shift registers adds 8 bits output for every shift register module.

221.1. Specifications Shift register 74HC595

- 8 bit serial input
- 8 bit serial or parallel output

You can find an excellent tutorial for using the 74HC595 at:
http://arduino.cc/en/tutorial/ShiftOut#.UwO2cUJ5OhE

221.2. Datasheet Shift register 74HC595

- http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

221.3. Connections Shift register 74HC595

Pin nr	Name	Description	Arduino pin
1	Q1	Parallel data output 1	
2	Q2	Parallel data output 2	
3	Q3	Parallel data output 3	
4	Q4	Parallel data output 4	
5	Q5	Parallel data output 5	
6	Q6	Parallel data output 6	
7	Q7	Parallel data output 7	
8	GND	Ground	GND
9	Q7S	Serial data output	To DS of next 74HC595
10	In^{verted}	Master Reset (active LOW)	5V
11	SHCP	Shift Register Clock input	Any digital port
12	STCP	Storage Register Clock input	Any digital port
13	In^{verted}	Output enable input (active LOW)	GND
14	DS	Serial data input	Any digital port
15	Q0	Parallel data output 0	
16	VCC	Supply Voltage	5V

221.4. Libraries needed for Shift register 74HC595

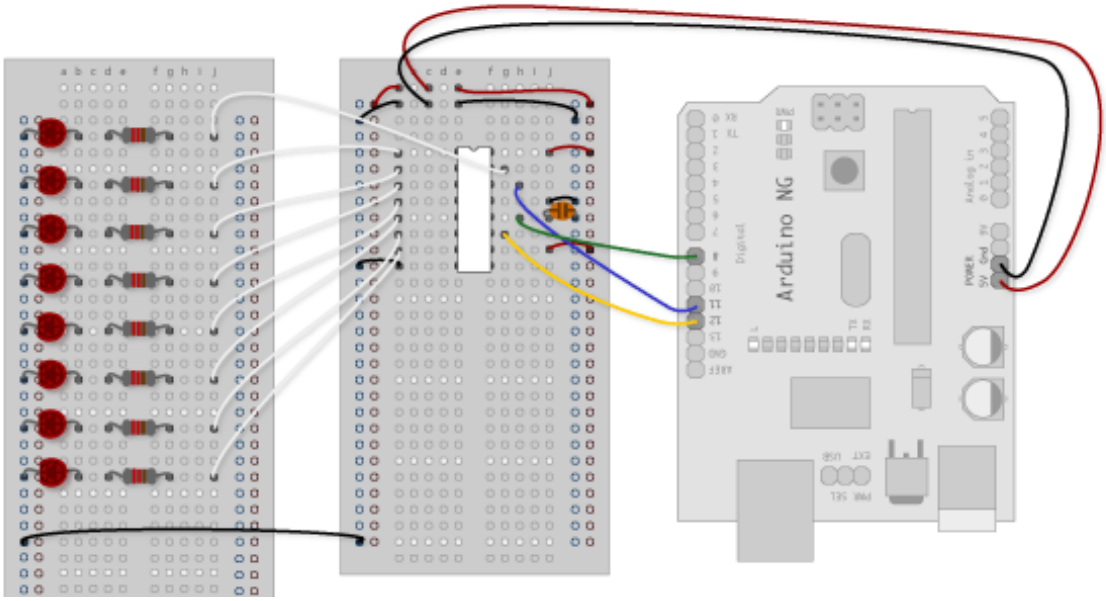
None needed.

221.5. Sample Shift register 74HC595

The following sketch displays the binary values 0 .. 255 on 8 LED's connected to the outputs of the 74HC595 shift register. Only 3 Digital Arduino Pins are needed, instead of one per LED (8). By daisy chaining several shift registers, you can increase the number of LED's and still only us 3 Digital Arduino Pins. Doing this you will need to use Darlington Transistors (like the BC517) or a Darlington Array (ULN2803L) to keep up with the higher current needed by adding more LED's.

Sample Connections

- Connect the Cathode of 8 LEDs to GND.
- Connect the Anode of these LEDs to a 220 ohm resistor.
- Connect pin 15 and 1 through 7 to the other legs of the 8 resistors.
- Connect pins 8 (GND) and 13 (OE) to GND.
- Connect pins 16 (VCC) and 10 (MR) to +5V.
- Connect pin 12 (ST_CP) to D8.
- Connect pin 11 (SH_CP) to D12.
- Connect pin 14 (DS) to D11.

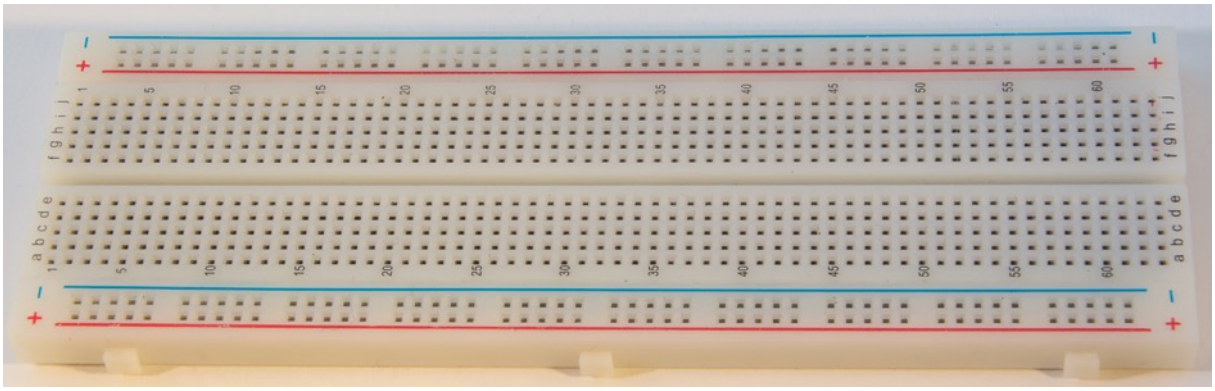


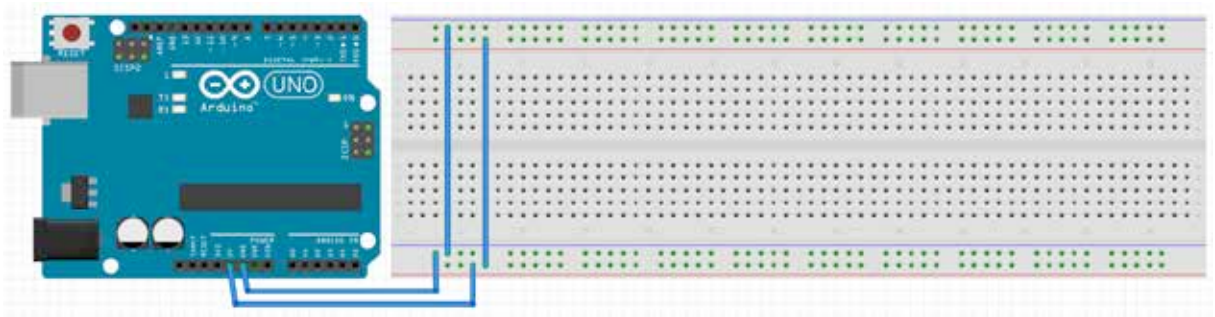
Pict. 2 Source: **http://arduino.cc/en/tutorial/ShiftOut#.UwO2cUJ5OhE**

121_Shiftregister_74HC595.ino

```
//Pin connected to ST_CP of 74HC595
int latchPin = 8;
//Pin connected to SH_CP of 74HC595
int clockPin = 12;
//Pin connected to DS of 74HC595
int dataPin = 11;

void setup()
{
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop()
{
  // count from 0 to 255 and display the corresponding LED's
  for (int numberToDisplay = 1; numberToDisplay < 256; numberToDisplay++)
  {
    // take the latchPin low so
    // the LEDs don't change while you're sending in bits:
    digitalWrite(latchPin, LOW);
    // shift out the bits:
    shiftOut(dataPin, clockPin, MSBFIRST, numberToDisplay);

    //take the latch pin high so the LEDs will light up:
    digitalWrite(latchPin, HIGH);
    delay(200);
  }
}
```

222. Solderless breadboard





A solderless breadboard is a base plate for proto typing. It typically consists of:

- Bus strips parallel to both long sides:
  - 1 row of connections dived in groups of 5, all together connected to each other, marked red, often used for VCC.
  - 1 row of connections dived in groups of 5, all together connected to each other, marked blue or black, often used for Ground.
- Terminal strips perpendicular to the long sides:
  - Several columns of 5 connections, those 5 connections are connected to each other but separated from the neighbor columns and separated from the column of 5 connections below.
- A nodge parallel to the long sides in the middle of the breadboard separates the terminal strips on both sides and provides limited airflow for DIP IC's.

ESP8266 WiFi modules

Up until the beginning of 2014, embedding WiFi in your projects was very expensive. This changed with the introduction of the ESP8266 module with list prices of a few Euro. Besides using this ESP8266 as an add on to your Arduino, it can also be used as a stand-alone module, with specs that outperform the Arduino, it can even be programmed with the Arduino IDE, using the same (sometimes modified) libraries.

A great piece of documentation has been written by Neil Kolban and can be found at: http://neilkolban.com/tech/esp8266/.

223. Common ESP8266

The ESP8266 is a microcontroller developed by the Chinese manufacturer Espressif and entered the western marked in August 2014

223.1. Specifications ESP8266

- 32-bit RISC CPU at 80 MHz
- External Flash 512 KiB to 4 MiB
- 802.11 b/g/n protocol
- WiFi Direct (P2P) and soft-AP
- Integrated Protocol stack.
- 16 GPIO pins (not available on all models).
- 1 10 bit ADC (= analog port with 1024 values)
- SPI, I2C
- PWM
- VCC and logical levels differ between 1.6-3.3V, some accept higher voltages through a voltage regulator or level shifters, so check the specks of your module.

223.2. Datasheet ESP8266

- https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf

223.3. ESP-xx model family

Some ESP8266's come as a SOC, other firms will solder the SOC on a IO Adapter plate, some with 5-3.3 Voltage regulators, others without.

The most common models come from AI-Thinker and are listed in the following table:

ID	Pins	Form	LED	Antenna	Antenna socket	Shielded
ESP-01	8	2X4 DIL	Yes	Etched on PCB	No	No
ESP-02	8	2X4 notch	No?	None	Yes	No
ESP-03	14	2x7 notch	No	Ceramic	No	No
ESP-04	14	2x4 notch	No?	None	No	No
ESP-05	5	1x5 DIL	No	None	Yes	No
ESP-06	12+GND	4x3 dice	No	None	No	Yes
ESP-07	16	2x8 pinhole	Yes	Ceramic	Yes	Yes
ESP-08	14	2x7 notch	No	None	No	Yes
ESP-09	12+GND	4x3 dice	No	None	No	No
ESP-10	5	1x5 notch	No	None	No	No
ESP-11	8	1x8 pinhole	No?	Ceramic	No	No
ESP-12	16	2x8 notch	Yes	Etched on PCB	No	Yes
ESP-12-E	22	2x8 notch	Yes	Etched on PCB	No	Yes
ESP-13	18	2x9	?	Etched on PCB	No	Yes

223.4. Usage of the ESP8266

The ESP8266 is very versatile and can be used in different ways.

- As a "standalone" MCU:
  - Programmed by Arduino IDE.
  - By uploading Lua scripts.
  - By uploading MicroPython scripts.
  - Sensors and/or actuators can be connected through various GPIO ports.
- As a MCU communicating with another MCU (for example an Arduino) via I2C or Serial, wired or wireless.
  - Programmed by Arduino IDE.
  - By uploading Lua scripts.
  - By uploading MicroPython scripts.
  - Sensors and/or actuators can be connected through various GPIO ports.
- As a Serial to Wi-Fi device.
  - By sending Lua or AT commands from another MCU (for example an Arduino).

224. Connection schemes for ESP8266 modules

There are 2 different setups for connecting an ESP8266 for specific situations. Use this chapter to decide which setup you must follow.

There is a setup for communicating between ESP8266 and a computer and a setup for communication between ESP8266 and an Arduino.

224.1. Communication between a computer and an ESP8266 module

In the following situations, your computer needs to communicate with an ESP8266 module.

- Typing AT commands (through Serial monitor, Putty or ESPlorer).
- Typing Lua commands (through Serial monitor, Putty or ESPlorer).
- Flashing firmware.
- Uploading sketches from Arduino IDE directly to the ESP8266.

Connections for communication with a computer

- When using an Arduino as a substitute for a USB to TTL Serial adapter:
  - Upload the Bare Minimum sketch to your Arduino. We are only using its UART (USB to Rx/Tx) functionality. In this situation Rx and Tx MUST NOT BE CROSSED AGAIN.[1]
  - Connect Rx to Arduino Rx (both are Rx)
  - Connect Tx to Arduino Tx (both are Tx)
  - Connect GND to Arduino GND
- When using a USB to TTL Serial adapter or FTDI adapter
  - Connect Rx (white) to USB-TTL Tx (cross Rx with Tx)
  - Connect Tx (green) to USB-TTL Rx (cross Rx with Tx)
  - Connect GND (black) to USB-TTL GND.
- Connect VCC to the correct voltage on an **EXTERNAL POWER SOURCE** (see the specs of your module). Neither the Arduino, nor the USB-TTL/FTDI gives enough current.
- Connect GND to an external power GND (so GND is connected to the external power and to the Arduino or the USB to TTL/FTDI connector).
- If you want to upload firmware, or if you want to upload sketches with the Arduino IDE software, put your module in boot load mode (see the specs of your module).

[1] The Rx/Tx of the computer's Serial Monitor is already crossed with the Tx/Rx pins of the Arduino. So connecting the Tx/Rx pins of the ESP8266, they will also be crossed with the Rx/Tx of your computer's Serial Monitor.

224.2. Communication between Arduino and an ESP8266 module

This setup is only needed when a sketch on your Arduino needs to communicate with the ESP8266. Your Arduino will be sending AT commands to the ESP8266, so Rx and Tx need to be crossed!

Connections for communication with an Arduino

- SoftwareSerial is not reliable when working with an ESP8266, so we are using Rx and Tx of the Arduino. Since the Arduino uses these pins also for uploading sketches, you must upload your sketch first, before you hookup your ESP8266.
    - Connect Rx to Arduino Tx (cross Rx with Tx)
    - Connect Tx to Arduino Tx (cross Tx with Rx)
    - Connect GND to Arduino GND
- Connect VCC to the correct voltage on an **EXTERNAL POWER SOURCE** (see the specs of your module). The Arduino does not give enough current.
- Connect GND to an external power GND (so GND is connected to the external power and to the Arduino).
- Put your module in boot load mode (see the specs of your module).

225. Using AT commands

Just like the Bluetooth modules and old-fashioned modems, you can change some of the settings on an ESP8266 with so-called AT-commands. This chapter describes how to connect to your ESP8266 and how to use the most common AT commands. These AT commands can also be used in sketches on an Arduino to talk to the ESP8266 as a Serial to WiFi device. Although AT commands are accepted by most firmware's (like the Espressif and AI-Thinker firmware), for some firmware's you need another command set/language. One of them is the NodeMCU firmware, that accepts the language Lua. With Lua it is possible to create complex scripts so you can use the whole power of your ESP8266. See "226 Using Lua scripts".

Look at "224.1 Communication between a computer and an ESP8266 module" for the connections you need to make.

225.1. Most common AT commands

Not all ESP8266 modules use the same AT command set. I've tried to list the most common commands in this paragraph. Check the following links for the Espressif and AI-Thinker firmware versions described in this document.

- AI-Thinker AT command set
  http://www.pridopia.co.uk/pi-doc/ESP8266ATCommandsSet.pdf
- Espressif AT commands
  http://firatdeveci.com/wp-content/uploads/Electronics/ESP8266_AT_Command.pdf

To use these commands, you'll first need to hookup your ESP8266 as described in "229.3 Programming the ESP8266 standalone" and then either type your commands in the Serial Monitor, or use the ESPlorer tool as described in "227 ESPlorer to use AT commands or LUA".

AT command	Response	Description/remarks
AT	*OK*	**Check if AT commands are accepted.**
AT+RST	*OK* *Some Boot messages (partly distorted because of the 76800 BAUDRATE during boot).* *ready*	**Restarts the ESP8266.** This does not reset the settings.
AT+GMR	*Firmware version information* *SDK version (SDK used to build the firmware)* *OK*	**Display Firmware version information.**

AT command	Response	Description/remarks
AT+CWMODE?	*+CWMODE:1* *or +CWMODE:2* *or +CWMODE:3* *OK*	**Show the mode of the ESP8266.** Mode 1: Client Mode 2: AP (Access Point) Mode 3: Client & AP The default SSID set on your ESP8266 depends on the used firmware. Samples: AI_THINKER_<part mac> ESP_<part_mac>
AT+CWMODE=<mode>	*OK*	**Change mode to <mode>**
AT+CWLAP	*+CWLAP:<een>, <ssid>, <rssi>, <mac>, <chan>* *+CWLAP:<een>, <ssid>, <rssi>, <mac>, <chan>* *.....* *OK*	**List available AP's.** *Only available in mode 2 and mode 3.* *<een>* *(security)* 0 Open 1 WEP 2 WPA_PSK 3 WPA2_PSK 4 WPA_WPA2_PSK *<rssi>* signal strength <ch> channel (not with all firmware)
AT+ CWJAP =<ssid>,<pwd>	*OK* *or ERROR*	**Join AP** *Only available in mode 2 and mode 3.*
AT+CWJAP?	*+CWJAP:<ssid>*	**Show info of AP to which ESP8266 is connected to.** *Only available in mode 2 and mode 3.*
AT+CWQAP	*OK*	**Quit AP** *Only available in mode 2 and mode 3.*

AT command	Response	Description/remarks
AT+CIFSR	*+CIFSR:<IP>* *+CIFST:<IP>* *OK* *or ERROR*	**Get local IP address**

Check the documentation of your specific firmware for more AT commands, like sending data, DHCP, closing connections etc.

226. Using Lua scripts

Lua is a very simple programming language with great potential. You can type and execute simple commands at a CLI (command line interface) or you can write a script and either run or even compile it. The easiest way to type your Lua commands or to upload and run your Lua-scripts is by using ESPlorer as described in "227 ESPlorer to use AT commands or LUA".

This chapter only describes some statements specifically for the ESP8266. More info can be found at:

- https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en
- http://www.lua.org/manual/5.1/

Look at "224.1 Communication between a computer and an ESP8266 module**" for the connections you need to make.**

226.1. Most common Lua commands

Lua command	Response	Description/remarks
file.format	format done. >	**Delete all scripts and other files.**
wifi.getmode()	1 or 2 or 3 >	**Shows the mode of the ESP8266**. 1 STATION (Client) 2 SOFTAP (AP) 3 STATIONAP (Client & AP)
wifi.setmode(wifi.STATION)	>	**Change to Client mode.**
wifi.setmode(wifi.SOFTAP)	>	**Change to AP mode.**
wifi.setmode(wifi.STATIONAP)	>	**Change to Client & AP mode.**
wifi.ap.getip()	<ip> <mask><gateway> or nill >	**Show its own IP address.** *does not work in mode 1*
wifi.sta.getip()	<ip> <mask><gateway> or nill >	**Show its own IP address.** *only works in mode 1*
print("Hello World.")	Hello World. >	**Prints a line of text**

function listap(t)
  for k,v in pairs(t) do
  print(k.." : "..v)
  end
end
wifi.sta.getap(listap)
	list of available AP's	**Shows a list of available AP's**

Some script examples, specifically for running on the ESP8266 can be found at:

- https://github.com/nodemcu/nodemcu-firmware.
- https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/using-nodemcu-lua

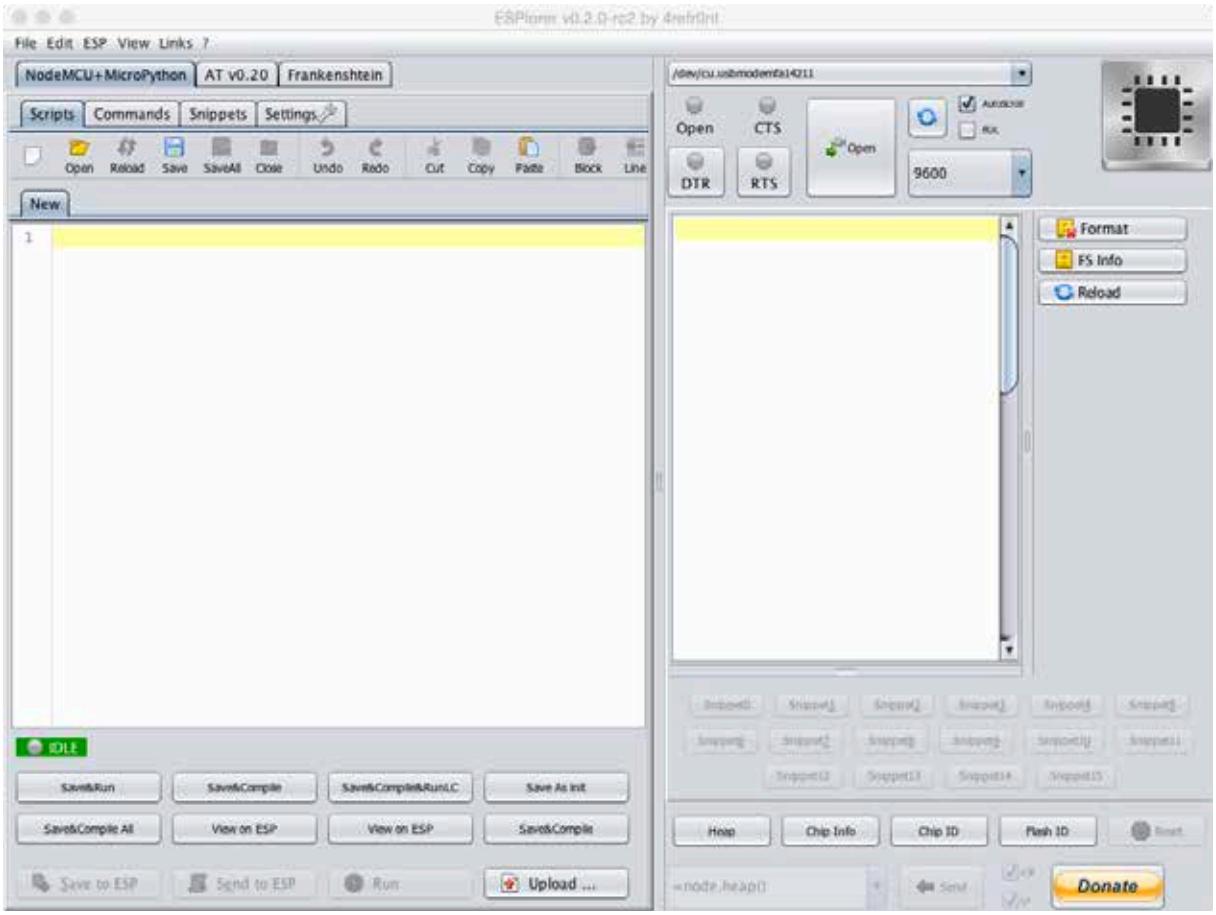
227. ESPlorer to use AT commands or LUA scripts

ESPlorer is a nice IDE you can use with firmware's that understand either AT-commands or Lua scripts. It is a JAVA program, so it should work on all operating systems with JAVA support.

Look at "224.1 Communication between a computer and an ESP8266 module" for the connections you need to make.

You can download it at:
http://esp8266.ru/esplorer/#download

You don't need to install this tool, just unzip the tool, hookup your ESP8266 as described in "229.3 Programming the ESP8266 standalone" and double click either ESPlorer.bat (Windows) or ESPlorer.jar (Linux or OSx).



Now select your Serial Connection at the top right part of the screen, select the correct BAUD rate (either 74880, 9600 or 115200) and then click OPEN.
To use Lua, open the tab called NODEMCU_MICROPYTHON. For most other firmware, open AT V0.20 for AT commands.

228. ESP8266 Firmware

After uploading a sketch to the ESP8266, the original firmware will be overwritten. To restore/update/change the firmware you need a firmware file and an upload tool. I've described 2 different firmware's and 2 different upload tools.

228.1. Firmware versions

There is lots of different firmware to be found. You can even build your own firmware (beyond the scope of this document). In this chapter only three versions are described, feel free to try others.

- Espressif firmware (AT)
- AI-Thinker's AT firmware
- NodeMCU Lua firmware

Espressif firmware

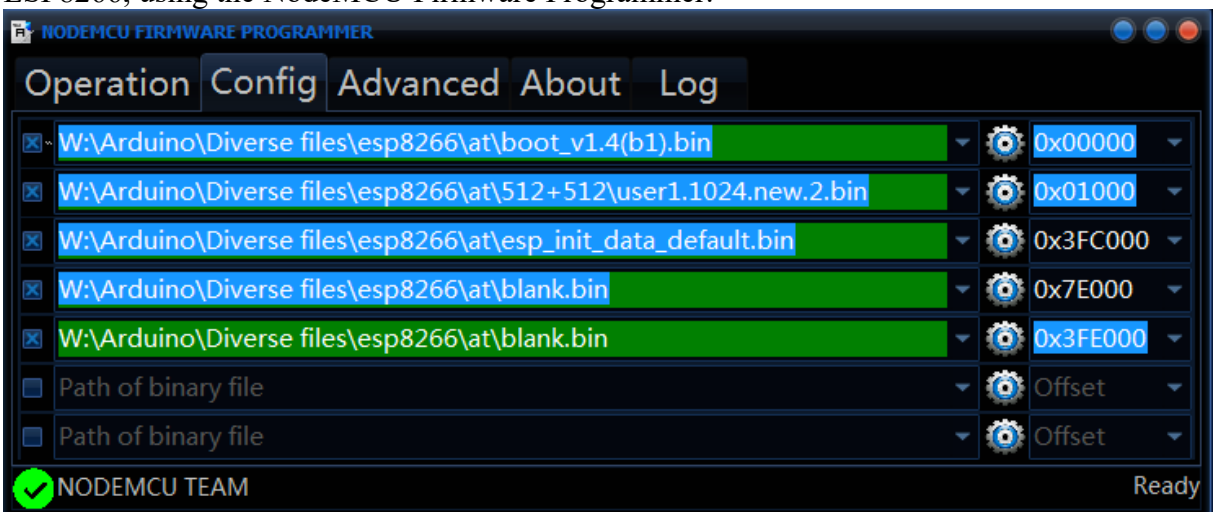
This firmware is very often used and comes from the original manufacturer of the ESP8266. It consist of 4 files that needs to be uploaded at the correct addresses.

you can download Espressif's firmware at:
http://bbs.espressif.com/viewtopic.php?f=46&t=1123L the link is at bottom of that page.

I used the Espressif firmware and the NodeMCU Firmware programmer. My ESP8266 board is a 32 Mbit (512K + 512K), so conform the README.TXT, I used the following settings :

- boot_v1.4(b1).bin at 0x00000
- user1.1024.new.2.bin at 0x01000
- esp_init_data_default.bin at 0x3FC000
- blank.bin at 0x7E000
- blank.bin at 0x3FE000

Below is a Screenshot of uploading Espressif firmware to a 32 Mbit (512K + 512K) ESP8266, using the NodeMCU Firmware Programmer.



AI-Thinker firmware

AI-Thinker is the manufacturer of most ESP-xx module boards and has developed its own firmware. It consists of only 1 file, that needs to be uploaded to 0x00000.

You can download AI-Thinker's firmware at:

http://www.electrodragon.com/w/ESP8266_AT-Command_firmware

After installing this firmware, the default BAUD rate is 115200.

NodeMCU Lua firmware

The NodeMCU firmware is a powerful firmware that is built around the Lua language. Where other firmware's accept AT command to change its settings and perform some simple programs, Lua is a complete language for complex tasks. You can find the latest NodeMCU firmware at the following link. It consist of only 1 file, that needs to be uploaded to 0x00000.

You can download NodeMCU's LUA firmware at:
https://github.com/nodemcu/nodemcu-firmware/releases

After installing this firmware, the default BAUD rate is 9600.

228.2. Uploading firmware

There are several tools you can use to upload firmware. The most common are:

- Command line:
  - Esptool-ck from Christian Klippel (Windows, Linux and OSx).
  - Esptool.py python scripts (Windows, Linux and OSx).
- GUI:
  - NodeMCU flasher (Windows only).

Look at "224.1 Communication between a computer and an ESP8266 module**" for the connections you need to make.**

Esptool-ck

You can download the binaries for Windows, Linux and OSx at:

- https://github.com/igrr/esptool-ck/releases.

Source code, including make files can be found at:

- https://github.com/igrr/esptool-ck/.

The syntax for esptool-ck is as follows:

```
esptool –cp <port> -ca <address> -cd <boardtype> -cf <firmware>
```

```
-cp <port>
The serial port of the device you are using to program the firmware.
For Windows this is something like COMx. On Linux or OSX, this is
something like /dev/cu.usb….. You can check this name in Arduino by
choosing TOOLS, PORT.

-ca <address>
The starting address to which your firmware needs to be written to.
You must check your documentation, default is 0x00000.

-cf <firmware>
The name of the firmware file you've downloaded. Spaces in the name of
this file can be escaped by a preceding \.

-cd <board type>
With <board type> you can describe how to set the board to boot load
mode.
   none      manually
   ck        RTS controls RESET or CH_PD, DTR controls GPIO0
   wifio     TxD controls GPIO0 via PNP transistor and DTR controls
             RESET via a capacitor
   nodemcu   GPIO and RESET controlled using two PNP transistors
```

This tool can't be used with a USB-Serial with the PL203HX chip. It works fine though with a USB-Serial cable with the CP2102 chip or with an FTDI cable (FT232RL chip).

Esptool.py with Python

The Esptool.py is a Python script using the Pyserial library. You need to download the following.

- Python (included in OSx): https://www.python.org/downloads/windows/
- Pyserial: https://pypi.python.org/pypi/pyserial
- ESPtool.py script: https://github.com/themadinventor/esptool

The syntax for Esptool.py is as follows:

```
python esptool.py --port <port> write_flash <address> <firmware>
```

<port>
The serial port of the device you are using to program the firmware. For Windows this is something like COMx. On Linux or OSX, this is something like /dev/cu.usb….. You can check this name in Arduino by choosing TOOLS, PORT.

<address>
The starting address to which your firmware needs to be written to. You must check your documentation, but often this is 0x00000.

<firmware>
The name of the firmware file you've downloaded. Spaces in the name of this file can be escaped by a preceding \.

Some firmware consists of multiple files, you need to use the esptool.py separate for each file with its corresponding <address>. These addresses are often mentioned in a 'readme.txt'.

You can also use the esptool.py to read information about your ESP8266, for example manufacturer and model.

```
esptool.py –port <port> flash_id
```

Shows Manufacturer and Device, by then checking this combo at the following link you can find out the model of the Flash chip. Searching the Internet for that specific model gives information about the flash size.

```
esptool.py –port <port> read_mac
```

Shows MAC address of ESP8266.

NodeMCU flasher

The NodeMCU firmware programmer has been built for the NodeMCU firmware, but can also be used for other firmware versions.

- https://github.com/nodemcu/nodemcu-flasher

Use the following procedure to flash your firmware:

- Double click on the file: ESP8266Flasher.exe.
- Select the correct COM port at the OPERATION tab.
- Open the CONFIG tab.
- Click on the cog-wheel at the first line and browse to the (first) firmware.
- Type the correct address on the right of your firmware.
- Repeat this 2 steps for every firmware part, each on a separate line.
- Make sure there is a check mark to the left of each firmware part.
- Delete not needed lines.
- Return to the OPERATION tab.
- Slide the PROGRAM/UART switch in the PROGRAM position and then press the RESET button. [1]
- Press FLASH.
- Slide the PROGRAM/UART switch in the UART position and then press the RESET button.

Test your new firmware with the AT command, see "225 Using AT commands".

[1] Not all ESP8266 modules have a PROGRAM/UART nor a RESET switch. In that case you need to pull some GPIO port either HIGH or LOW. I only own a ESP8266 equipped with these switches so I will need to refer you to GOOGLE at this moment. I will update this document as soon as I get hold of some other ESP8266 modules without switches.

229. Programming ESP8266 with Arduino IDE

This module can be used as a WiFi to serial module to connect an Arduino with WiFi., but it is also a standalone microprocessor with much better specs than the Arduino.

229.1. Connections ESP8266 board standalone

The connections are dependent of the ESP8266 module/board used.

229.2. First time preparation of Arduino IDE for the ESP8266 board standalone

This section describes how to add support for the ESP8266 standalone to the Arduino IDE.

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and past the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator.) http://arduino.esp8266.com/package_esp8266com_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for ESP (this could take a few seconds) and select the ESP BY ESP8266 COMMUNITY.
- Click on the INSTALL button, on a Windows computer you will now be asked to install (or update) the drivers for the ESP8266 boards.
- Press the CLOSE button.
- The ESP boards are now listed at TOOLS, BOARD.
- During this process, all necessary libraries and some example sketches are also made available in Arduino IDE.

These steps are also described at the following link:
http://www.arduinesp.com/getting-started

229.3. Programming the ESP8266 standalone

Look at "224.1 Communication between a computer and an ESP8266 module**" for the connections you need to make.**

- Choose TOOLS, BOARD, GENERIC ESP8266 MODULE.
- You don't need to select a programmer.
- Choose TOOLS, FLASH MODE, DIO
- Choose TOOLS, FLASH FREQUENCY, 40 MHZ
- Choose TOOLS, CPU FREQUENCY, 80 MHz
- Choose TOOLS, FLASH SIZE, 512K (64K SPIFFS)
- Choose TOOLS, FLASH SPEED, 115200
- Choose TOOLS, PORT, the port to your USPASP or to you Arduino
- Put your module in boot load mode (see the specs of your module).
- Press the UPLOAD button to upload your sketch.

This does not work with a USB-Serial cable with the PL203HX chip. It works fine though with a USB-Serial cable with the CP2102 chip or with an FTDI cable (FT232RL chip).
https://github.com/esp8266/Arduino/issues/710

230. ESP8266 as WiFi to Serial

The previous chapter described the ESP8266 ESP-07 as a standalone microcontroller board. This chapter describes the other role, i.e. the ESP8266 ESP-07 as a WIFI to serial communication device for the Arduino. By sending AT commands from Arduino to the ESP8266 you can setup a connection to an access point and send and receive information through TCP/IP. Make sure an AT-firmware is installed (like the Espressif or the AI-Thinker firmware). Communication with websites will be using the HTTP protocol. Information about both AT commands and the HTTP protocol used with ESP8266 is described at: http://rancidbacon.com/files/kiwicon8/ESP8266_WiFi_Module_Quick_Start_Guide_v_1.0.4.pdf

230.1. Connections ESP8266 as WiFi to Serial

Pin nr	Name	Description	External Power	Arduino pin
1	GND	Ground	GND	GND
2	VCC	5 Volt	Check the specs of your module.	Not connected
3	Tx	Transmit data		Rx (or SoftwareSerial)
4	Rx	Receive data		Tx (or SoftwareSerial)

230.2. Libraries needed for ESP8266 as WiFi to Serial

The libraries you need, depend on the project.

230.3. Sample ESP8266 Blynk and a smartphone

Blynk is an app available for Android and iPhone for building an interface from your smartphone to Arduino, ESP8266 and Raspberry Pi. In this sample you create a simple app with one button controlling the internal LED on the Arduino at D13.

Check the following video:
https://www.youtube.com/watch?v=jbJi8VX5Dg0

Preparation of Arduino IDE

Install the following libraries, part of one ZIP file:

- Blynk
- BlynkESP8266_HardSer
- BlynkESP8266_SoftSer
- SimpleTimer

You can download these libraries at:
Library: https://github.com/blynkkk/blynk-library/releases

Explanation of the code used in this example

```
#define BLYNK_PRINT Serial
```

This line takes care of printing to the Serial Monitor so you can see whether things are working. You should see some AT commands running by.

```
#include <ESP8266_HardSer.h>
```

The library from Blynk to support the ESP8266 on hardware serial.

```
#include <BlynkSimpleShieldEsp8266_HardSer.h>
```

Second part of the Blynk library to support the ESP8266 on hardware serial.

```
#define EspSerial Serial
```
EspSerial will be the name of the Serial connection to the ESP8266.

```
ESP8266 wifi(EspSerial);
```
Create wifi, an instance of the object ESP8266, connected to EspSerial.

```
char auth[] = "<token>";
```
This is where you need to type the token to your smartphone project.

```
EspSerial.begin(115200);
```
Start the serial connection to the ESP8266 at the speed that is set with your firmware.

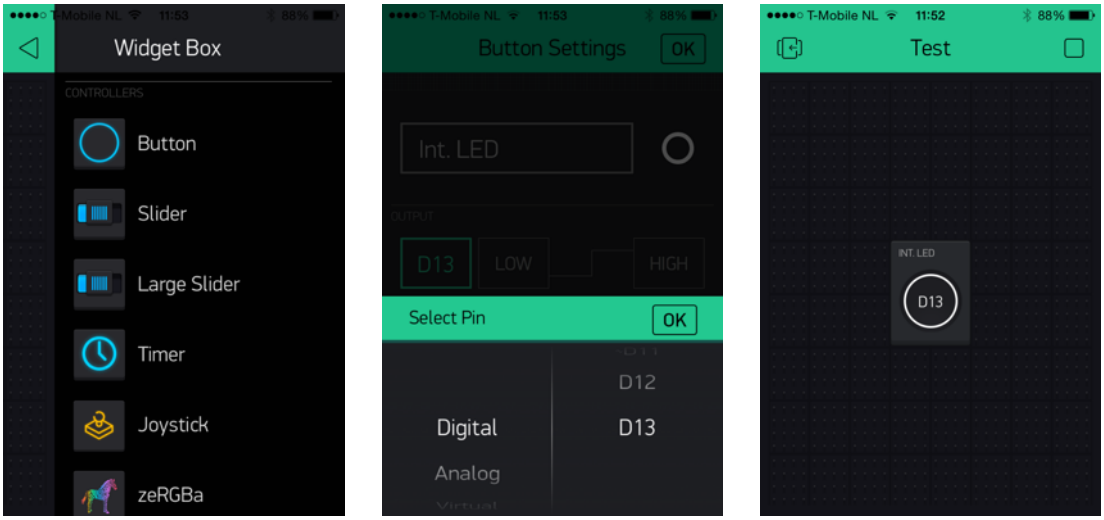
```
Blynk.begin(auth, wifi, "<SSID>", "<WPA2key>");
```
Type in the name of your Wi-Fi <SSID> and the corresponding <WPA2key>.

```
Blynk.run();
```
This function will continuously check the input from the project in Blynk on your smartphone.

Preparation of your smartphone

- Install the Blynk app on your smartphone, more details at: http://www.blynk.cc/
- Run the app and create a new account.
- Create a new project and mail the created token to your computer.
- Place a button and direct this to the port of your onboard LED. Check your ESP8266//ESP32 for the correct port number (D4 for NodeMCU).
- Don't start this application before you've started the sketch on your ESP8266.

Below you find some screenshots from Blynk on an iPhone.



Sample Connections

- Create a project on your smartphone and copy the corresponding <token> into the sketch below.
- Put your SSID and WPA2key into the sketch below.
- Upload the sketch first before you connect the ESP8266.
- **Look at "224.2** Communication between Arduino and an ESP8266 module**" for the connections you need to make.**

122_ESP8266_Blynk.ino

```
#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

char auth[] = "token";

char ssid[] = "ssid";
char pass[] = "WPA2-key";
BlynkTimer timer;

void setup()
{
  Serial.begin(9600);
  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  Blynk.run();
}
```

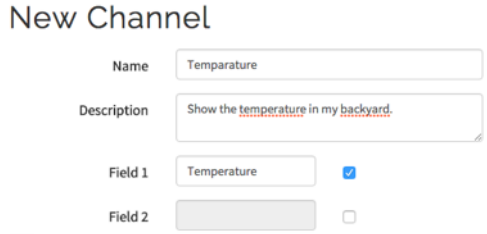
230.4. Sample 2 log data on Thingspeak

Thingspeak is a website that accepts input through both GET and POST requests. In this sample I'm going to describe how to log data from a temperature sensor on your own channel/page on Thingspeak.

Check the following instructable for more information:
http://www.instructables.com/id/ESP8266-Wifi-Temperature-Logger/

Preparation of Thingspeak

- Sign up for an account at https://thingspeak.com
- Create a new Channel.



- Save you Channel.
- Copy your CHANNEL ID. You can use this ID to test whether data has been logged.
- Go to API KEYS
- Copy the key listed at: WRITE API KEY, you'll need this in your sketch.
    - You can manually enter a temperature in your channel by copying your WRITE API KEY in the following URL: http://api.thingspeak.com/update?key=[THINGSPEAK_KEY]&field1=0
    - You can check your results by copying your channel in to the following URL: http://api.thingspeak.com/channels/[CHANNEL_ID]/feed.json?key=[THINGSPEAK_KEY]

Library explanation

```
AT+RST
```

Reset ESP module

```
Set ESP in client mode
AT+CWMODE=1
```

Set ESP in WiFi Client mode, so it can connect to a Access Point

```
AT+CWJAP="SSID","WPA2KEY"
```

Connect to Access Point

```
Send GET sequence (connect, datalength, get...)
AT+CIPSTART="TCP","184.106.153.149",80
```

Make a connect with the Thingspeak website

```
AT+CIPSEND=51
```

Set the data length to 51?

```
GET /update?api_key=0123456778901234&temperature=21.3
```

Update the temperature to 21.3

Sample Connections

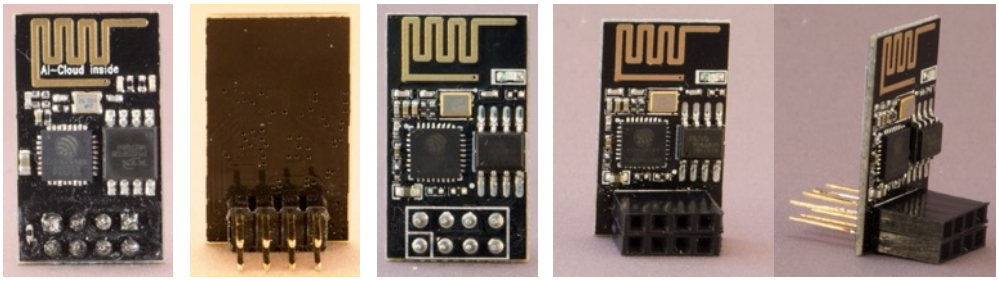
- Put your SSID and WPA2key into the sketch below.
- Upload the sketch first before you connect the ESP8266.
- Connect Rx to D11
- Connect Tx to D10
- **Look at "224.2** Communication between Arduino and an ESP8266 module" **for the connections you need to make.**

123_ESP8266_Thingspeak.ino

```
#include <SoftwareSerial.h>
#include <stdlib.h>
#define SSID "<type here your SSID>"
#define PASS "<type here your WPS2 key>"
#define IP "184.106.153.149" // ThingSpeak IP Address: 184.106.153.149

int lm35Pin = A0;
int aantal = 1;
String apiKey = "<Type here your WRITE API KEY>";

SoftwareSerial ser(10, 11); // RX, TX

void setup()
{
  Serial.begin(9600);
  ser.begin(115200);
  ser.println("AT+RST");
  Serial.println("AT+RST");
  while (!connectWiFi())
  {
    Serial.println("No WiFi connection, retrying");
    delay(2000);
  }
}

boolean connectWiFi()
{
  ser.println("AT+CWMODE=1");
  Serial.println("AT+CWMODE=1");
  delay(2000);
  String cmd = "AT+CWJAP=\"";
  cmd += SSID;
  cmd += "\",\"";
  cmd += PASS;
  cmd += "\"";
  ser.println(cmd);
  Serial.println(cmd);
  delay(7000);
  if (ser.find("OK"))
  {
    Serial.println("verbinding");
    return true;
  }
  else
  {
    Serial.println("Geen verbinding");
    return false;
  }
}

void loop()
```

```
{
  int val = 0;
  for (int i = 0; i < 10; i++)
  {
    val += analogRead(lm35Pin);
    delay(500);
  }
  float temp = val * 50.0f / 1023.0f;
  char buf[16];
  String strTemp = dtostrf(temp, 4, 1, buf);
  Serial.println(strTemp);
  String cmd = "AT+CIPSTART=\"TCP\",\"";
  cmd += "184.106.153.149";
  cmd += "\",80";
  ser.println(cmd);
  if (ser.find("Error"))
  {
    Serial.println("AT+CIPSTART error");
    return;
  }
  String getStr = "GET /update?api_key=";
  getStr += apiKey;
  getStr += "&temperature=";
  getStr += String(strTemp);
  getStr += "\r\n\r\n";
  cmd = "AT+CIPSEND=";
  cmd += String(getStr.length() - 2);
  ser.println(cmd);
  if (ser.find(">"))
  {
    ser.print(getStr);
    Serial.println(aantal++);
  }
  else
  {
    ser.println("AT+CIPCLOSE");
    // alert user
    Serial.println("AT+CIPCLOSE");
  }
  delay(16000);
}
```

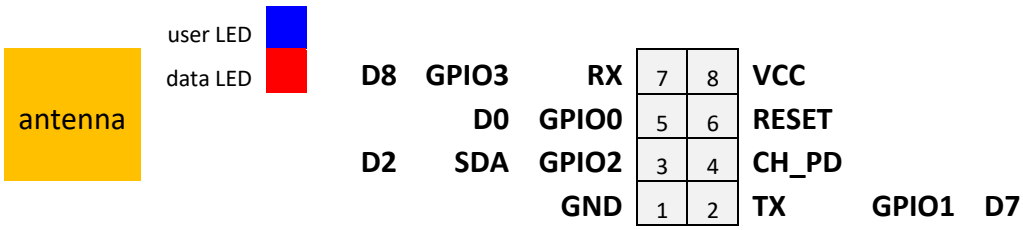
231. ESP-module: ESP8266 ESP-01



Specifications ESP8266 ESP-01

Operating voltage: 3.0-3.6V
Very breadboard UNfriendly.
PCB etched antenna without connector
1MB flash memory
Enable-pin
2x GPIO-pins (3.3V logic)
80 MHz
64 KB instruction RAM
96 KB data RAM
64 KB boot ROM
1 MB Flash memory
Default baud: 115200 8N1
Power LED: red
TX LED = User LED (GPIO1): blue

Layout/connections ESP8266 ESP-01



Voltage levels ESP8266 ESP-01

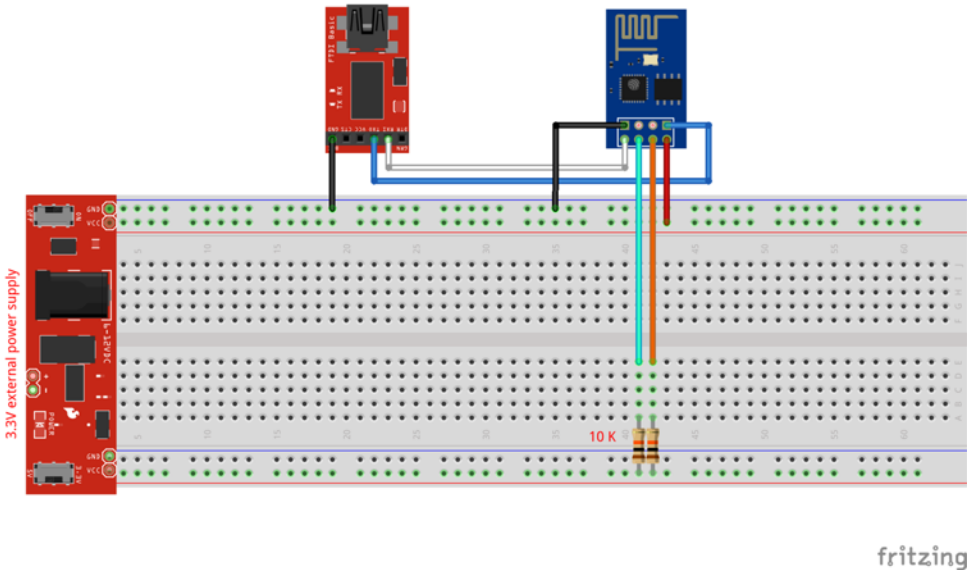
Input voltage: 3.3V
Logic Level: 3.3V
Rx, RST level: 3.3V

Preferably use an external power supply. Don't use the 3.3V from an Arduino, nor from an USB-Serial nor from an FTDI. The maximum current those devices can deliver is not enough.

Documentation ESP8266 ESP-01

- Datasheet:
  - https://nurdspace.nl/ESP8266
  - https://cdn.sparkfun.com/datasheets/Wireless/WiFi/ESP8266ModuleV1.pdf
- Forum: http://www.esp8266.com/
- AT-command set: https://cdn.sparkfun.com/datasheets/Wireless/WiFi/Command%20Doc.pdf

Run mode ESP8266 ESP-01 with an Arduino

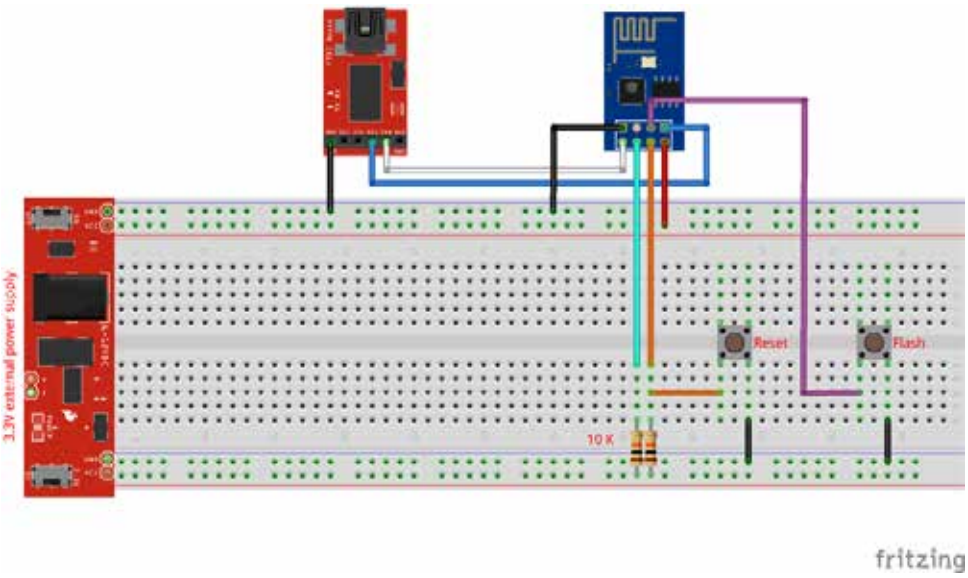


You can test AT commands by using Serial Monitor and a USB to Serial convertor like an FTDI, by following these steps:

- Make the following connections
  - VCC => 3.3V (external power supply)
  - GND => GND
  - Rx => TxD
  - Tx => RxD (also Tx to Tx)
  - Rst => through a 10 Kohm resistor to 3.3V
  - CH_PD => through a 10Kohm resistor to 3.3 V
- Open Serial Monitor
- Set Serial Monitor to 115200 and Both NL & CR.

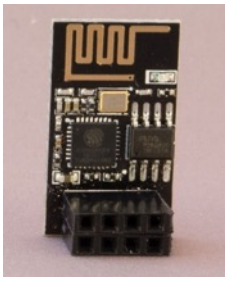
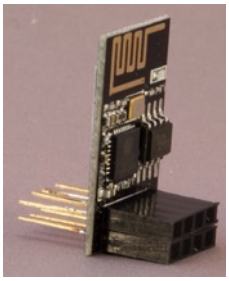
Boot load mode ESP8266 ESP-01

Flash-setting	Value
Crystal Frequency	26 MHz
SPI Speed	40 MHz
SPI mode	DIO
Flash Size	8 Mbit (512KB+512KB)
Baud rate	115200



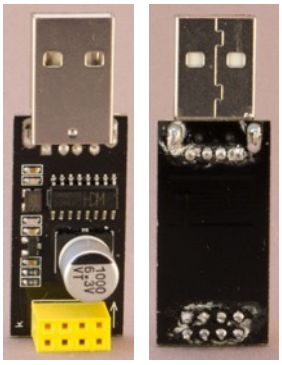
- Add the following connections
  - RST => through a switch to GND
  - GPI0 => through a switch to GND
- To start flash mode
  - Press RST Switch and keep it pressed
  - Press CH_PD switch and keep it pressed
  - Release the RST switch
  - Release the CP_PD switch
  - Now you can upload a sketch or another firmware

232. ESP-module: ESP8266 ESP-01 mod

I replaced the original male-header pins of an ESP-01 with two Arduino shield stack pins. This way all pins are available when using it with an USB-ESP-01 adapter.

233. Module: USB to ESP-01



The USB to ESP-01 is not an ESP8266, but a way to communicate between a computer and an ESP8266 ESP-01 module. It can't be used to flash new firmware, but you can use it to upload lua-scripts in case your ESP-01 is flashed with nodemcu firmware.

- When you insert the ESP-01 onto this board Tx and Rx from this module will automatically be crossed with Rx and Tx from the ESP-01.
- You can also use this USB to ESP-01 module to connect other 3.3V serial devices (like other ESP8266 modules).
- With the mod in the next chapter it is even possible to use this USB to ESP-01 to flash firmware or Arduino sketches to an ESP-01.

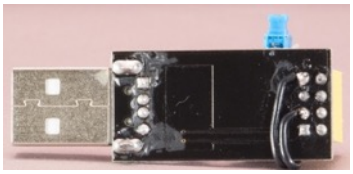
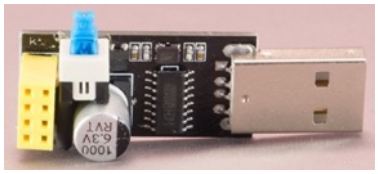
Specifications USB to ESP8266 ESP-01

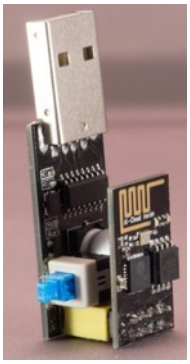
Operating voltage: 3.3V
UART: CH340
No GPIO ports reachable
1000uF capacitor for a stable VCC

Run mode USB to ESP-01

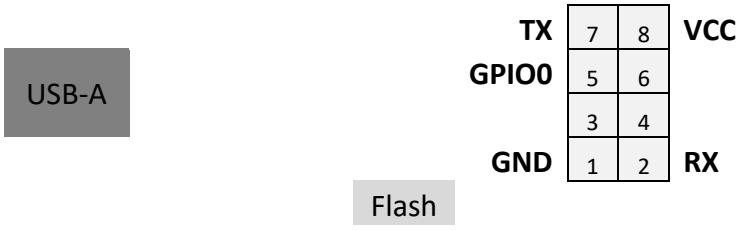
- Make sure your ESP-01 module is flashed with the nodemcu firmware so it understands Lua commands.
- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector
- Insert USB-01 into your computer
- You will now be able to type Lua-commands using any Serial Terminal like Putty, Serial Monitor in Arduino IDE or ESPlorer
- Using ESPlorer you will even be able to upload Lua script. See “227 ESPlorer to use AT commands or LUA“.

234. Module: USB to ESP-01 flash mod v1



With this simple mod, it is possible to use a normal USB to ESP-01 module to flash or program an ESP-01.
To flash any ESP8266 module GPIO0 needs to be low during Reset. By soldering a changeover switch between GPIO0 and GND. You can now use this switch to enter flash mode.



- When you insert the ESP-01 onto this board Tx and Rx from this module will automatically be crossed with Rx and Tx from the ESP-01.
- You can also use this USB to ESP-01 module to connect other 3.3V serial devices (like other ESP8266 modules).

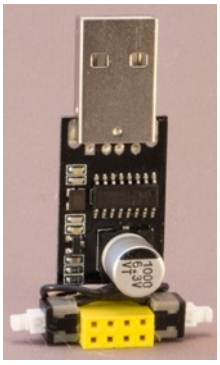
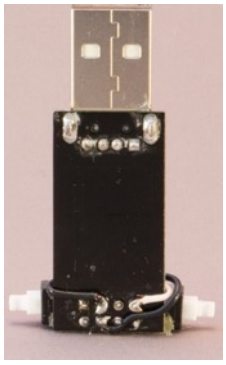
Run mode USB to ESP-01 flash mod

- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Make sure the flash switch is not closed and Insert USB-01 into your computer.
- The ESP-01 is now in run mode and you can communicate with it using any Serial Terminal like Putty, Serial Monitor in Arduino IDE or ESPlorer.

Boot load mode USB to ESP-01 flash mod

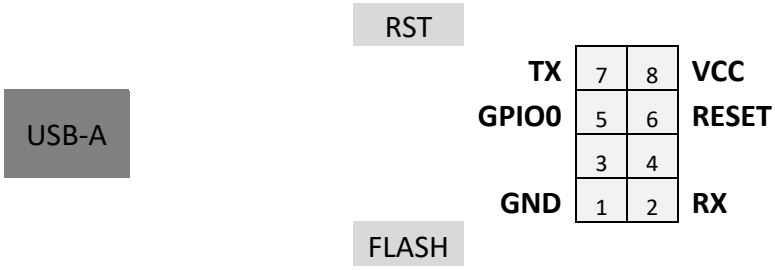
- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Make sure the flash switch is closed and insert the USB to ESP-01 into your computer.
- Now you can upload a sketch or another firmware

235. Module: USB to ESP-01 flash mod v2

With this second mod, it is possible to use a normal USB to ESP-01 module to flash or program an ESP-01, you can even reset the ESP-01.

To flash any ESP8266 module GPIO0 needs to be low during Reset. By soldering a momentary switch between GPIO0 and GND (Flash button) and one between RST and GND (Reset button). You can now use these switches to enter flash mode.



- When you insert the ESP-01 onto this board Tx and Rx from this module will automatically be crossed with Rx and Tx from the ESP-01.
- You can also use this USB to ESP-01 module to connect other 3.3V serial devices (like other ESP8266 modules).

Run mode USB to ESP-01 flash mod v2

- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Insert the USB-01 into your computer.
- If necessary you can press and release the Reset button.
- The ESP-01 is now in run mode and you can communicate with it using any Serial Terminal like Putty, Serial Monitor in Arduino IDE or ESPlorer.

Boot load mode USB to ESP-01 flash mod v2

- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Insert the USB-01 into your computer.
- Press the Flash button and keep it pressed.
- Press and release the Reset button.
- Release the Flash button.
- Now you can upload a sketch or another firmware

236. Module: USB to ESP-01 with flash switch

The USB to ESP-01 is not an ESP8266, but a way to communicate between a computer and an ESP8266 ESP-01 module. Most USB to ESP-01 module can't be used to flash new firmware, but this version is equipped with a flash switch (like the modded version from the previous chapter.

- When you insert the ESP-01 onto this board Tx and Rx from this module will automatically be crossed with Rx and Tx from the ESP-01.
- You can also use this USB to ESP-01 module to connect other 3.3V serial devices (like other ESP8266 modules).

Specifications USB to ESP8266 ESP-01

Operating voltage: 3.3V
UART: CH340
No GPIO ports reachable
1000uF capacitor for a stable VCC

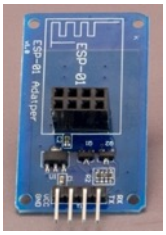
Run mode USB to ESP-01 flash mod

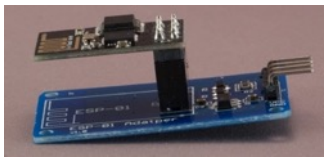
- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Make sure the flash switch is in the UART mode and Insert USB-01 into your computer.
- The ESP-01 is now in run mode and you can communicate with it using any Serial Terminal like Putty, Serial Monitor in Arduino IDE or ESPlorer.
- Using ESPlorer you will even be able to upload Lua script. See “227 ESPlorer to use AT commands or LUA

Boot load mode USB to ESP-01 flash mod

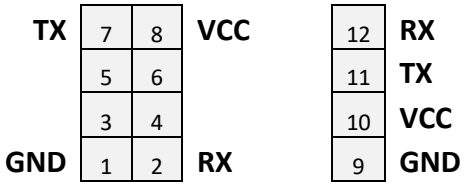
- Place an ESP8266 ESP-01on top of the USB to ESP-01 with the antenna facing the USB connector.
- Make sure the flash switch is the PROG position and insert the USB to ESP-01 into your computer.
- Now you can upload a sketch or another firmware

237. Module: ESP-01 5V-3.3V adapter





ESP-01 modules are all 3.3V modules and will be damaged when connected to a 5V Arduino. This ESP-01 5V-3.3V adapter regulates the 5V from an Arduino (or other MCU) to 3.3V (Rx, Tx and VCC). So with this adapter you can use an ESP-01 as a WiFi module on an Arduino. Since only GND, VCC, Tx and Rx are available you cannot use any GPIO ports nor can you use this adapter to flash sketches nor firmware on the ESP-01.



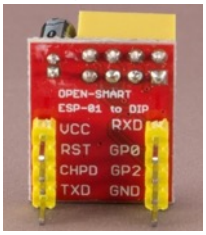
Run mode

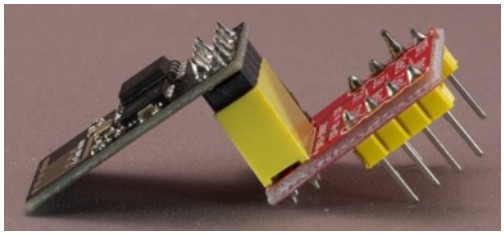
- Connect Rx to a Tx from an USB to TTL adapter (like an FTDI).
- Connect Tx to Rx from an USB to TTL adapter (like an FTDI).
- Connect GND to GND from an USB to TTL adapter (like an FTDI).
- Connect GND to GND on an external power supply.
- Connect VCC to 5V on an external power supply.

Boot load mode

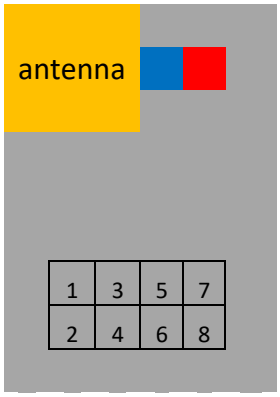
- Since RST and GPIO0 are not available, boot load mode is not support with this adapter.

238. Module: Open Smart ESP-01 to DIP





ESP-01 modules are not breadboard friendly. With this ESP-01 to DIP you can fit an ESP-01 to a breadboard. A capacitor is attached, to stabilize the VCC level. There is no regulator on this adapter, so you'll need an external 3.3 power supply.



RX	7	8	VCC
GPIO0	5	6	RESET
GPIO2	3	4	CH_PD
GND	1	2	TX

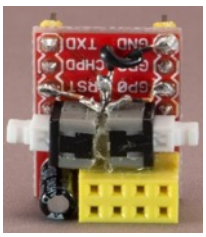
Run mode

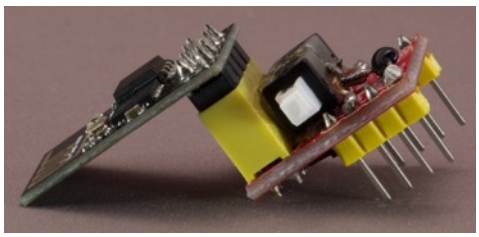
- See "231 ESP-module: ESP8266 ESP-01" to enter Run Mode.

Boot load mode

- See "231 ESP-module: ESP8266 ESP-01" to enter Boot load mode.

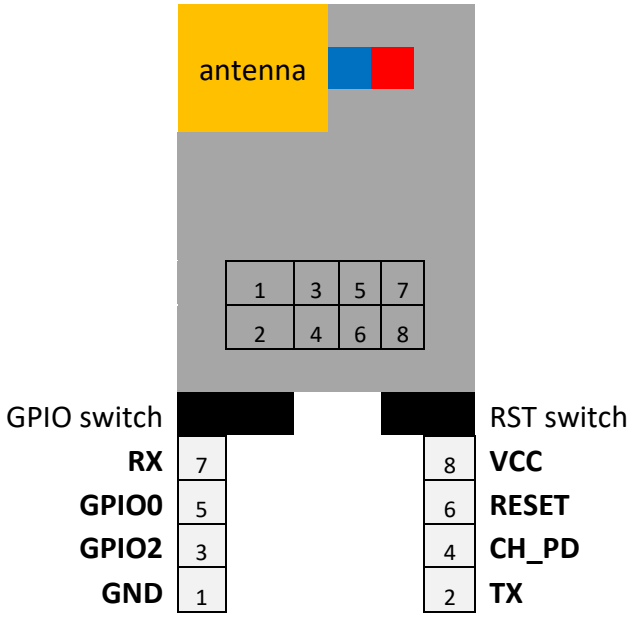
239. Module: Open Smart ESP-01 to DIP Mod





I modified the Open Smart ESP-01 to DIP adapter so I can use it to flash sketches and firmware to the ESP-01.

- Added a switch between RST and GND and between GPIO0 and GND
- Added a 10K resistor between CH_PD and VCC and between RST and VCC



Run mode

- Connect GND to GND on an external power supply.
- Connect VCC to 3.3V on an external power supply.
- Run mode is the default mode of this modded ESP-01 to DIP adapter. You can press the Reset switch to restart ESP-01 module.

Boot load mode

- Connect Rx to a Tx from an USB to TTL adapter (like an FTDI).
- Connect Tx to Rx from an USB to TTL adapter (like an FTDI).
- Connect GND to GND from an USB to TTL adapter (like an FTDI).
- Connect GND to GND on an external power supply.
- Connect VCC to 3.3 V on an external power supply.
- Press the GPIO0 switch and keep it pressed.
- Press and release the RST switch.
- Release the GPIO0 switch.
- You can now upload a sketch or flash a firmware.

240. ESP-module: Adafruit HUZZAH ESP8266 breakout

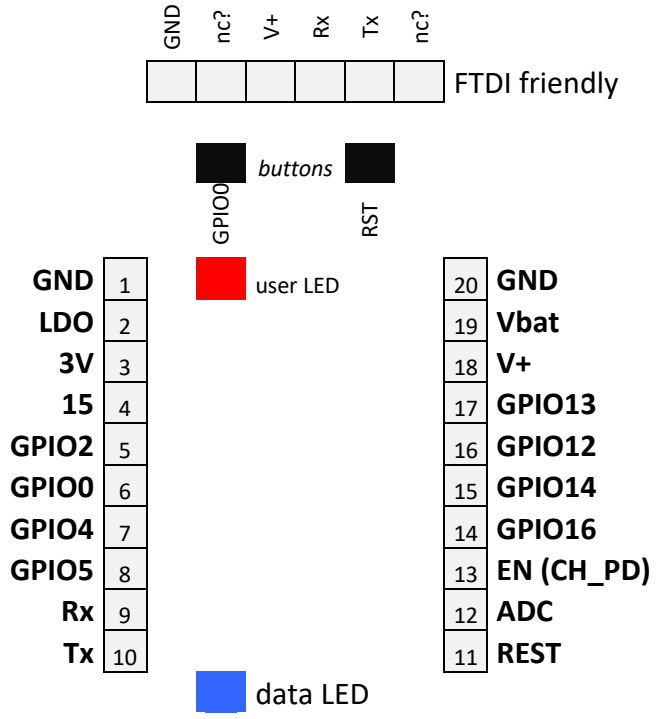
Small relatively cheap ESP8266-ESP12 breakout board from Adafruit.

Specifications Adafruit HUZZAH ESP8266 breakout

ESP-12	Breadboard friendly.
Default firmware: NodeMCU 0.9.5 build 20150318 (Lua 5.1.4)	1 analog input (1.8V max)
Reset button	LDO-disable-pin
User/Bootloader button	2x UART pins-pin
Red LED you can blink	2x 3.5-12 V power inputs-pin
Level shifting on the UART and Reset pin	Reset-pin
3.5 -12V input	Enable-pin
Two diode-protected power inputs (USB and Vbat)	9x GPIO-pins (3.3V logic) *can be used for I2C or SPI*
PCB etched antenna without connector	3.3V output-pin
UART header with FTDI friendly layout	
CE and FCC emitter certified	

More details can be found at: http://www.adafruit.com/product/2471.

Layout/connections Adafruit HUZZAH ESP8266 breakout



More information on this layout can be found at:
https://learn.adafruit.com/adafruit-huzzah-esp8266-breakout/pinouts.

Voltage levels Adafruit HUZZAH ESP8266 breakout

Input voltage: 3.5 – 12V on Vbat, or 5V on USB
Logic Level: 3.3V
Rx, RST level: 3.3V, but 5V is OK.

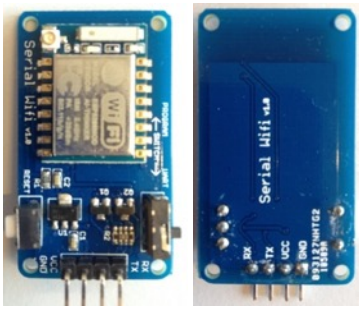
My board crashed when using at 3.3V as input voltage on the FTDI header, so 3.3V is just out of specs! Flashing firmware or uploading sketches seemed no problem with an FTDI cable on 3.3 V. My FTDI cable can be resoldered to work on 3.3 or 5 V. So I leave this FTDI on 3.3 V for programming/flashing all my ESP8266 modules (including this HUZZAH) and I use an external power on 5 V and optional some other USB-Serial cable for input/output to my computer.

Boot load mode Adafruit HUZZAH ESP8266 breakout

Flash-setting	**Value**
Crystal Frequency	26 MHz
SPI Speed	40 MHz
SPI mode	DIO
Flash Size	32 Mbit (512KB+512KB)
Baud rate	115200

To start boot load mode, hold down GPIO button, while pressing Reset. The User LED will now be dimly lid.

241. ESP-module: ESP8266 ESP-07 AI-Thinker



This board doesn't have any GPIO pins on its headers. Main use for this ESP8266 ESP-07 AI-Thinker board is as a Serial <=> Wi-Fi module.

Specifications ESP8266 ESP-07 AI-Thinker board

ESP-07	No GPIO on headers
Default firmware: AI-Thinker AT	Ceramic antenna with connector
Reset button	Program (boot load) <=> UART switch
Operation LED	Data LED
4.5-5.5V Input	

Layout/connections Adafruit

Nr	Name	Description
1	Rx	Receive data
2	Tx	Transmit data
3	VCC	4.5-5.5 V
4	GND	Ground

Voltage levels ESP8266 ESP-07 AI-Thinker board

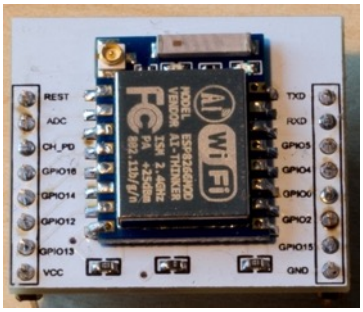
Input voltage: 4.5-5.5V on VCC
Logic Level: 3.3 – 5V

Preferably use an external power supply. Don't use the 5V from an Arduino, nor from a USB-Serial nor from an FTDI, it doesn't provide enough current.

Boot load mode ESP8266 ESP-07 AI-Thinker board

To start boot load mode, move switch into the PROGRAM position (press RESET once?). Don't forget to move switch back into the UART position afterwards.

242. ESP-module: ESP8266 ESP-07 with IO Adapter Plate

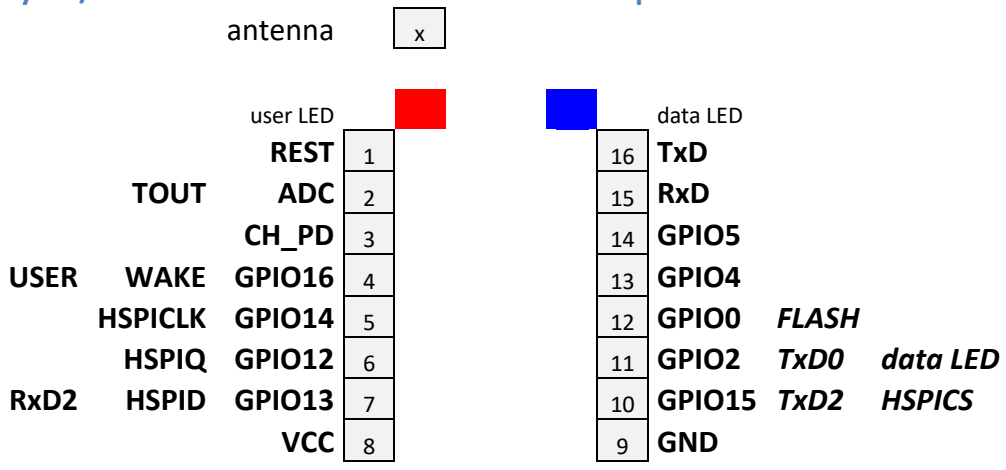


This ESP8266 ESP-07 was sold including a nice IO Adapter Plate. With a steady hand, you had to solder the ESP-07 to the PCB with 0.1 inch spaced headers. Although these headers fit on a breadboard, this module is not breadboard friendly, since it covers all breadboard connections, leaving no connections available for jumper cables.

Specifications ESP8266 ESP-07 with IO Adapter Plate

ESP-07	9 GPIO ports on headers
Default firmware: AI-Thinker AT	1 ADC port on header
Default bitrate: 115200	1 CH_PD on header
No buttons	RST on header
Operation LED	Tx and Rx on headers
Data LED (connected to GPIO2)	Ceramic antenna with connector
3.3V Input	
Breadboard **UNFRIENDLY**	

Layout/connections ESP8266 ESP-07 with IO Adapter Plate



Boot load mode ESP8266 ESP-07 with IO Adapter Plate

To start boot load mode, follow the next steps:

- Connect GPIO0 to GND.
- Pull RST to Low and then High.

Automatic boot load/flash mode with a modified FTDI Friend

Arduino IDE and esptool-ck (parameter -cd ck) can automatically put the ESP-07 into boot load/flash mode. You need a modified FTDI Friend and the following connections:

- Modify the FTDI Friend according to page 17 of the following document: https://learn.adafruit.com/downloads/pdf/ftdi-friend.pdf
- Connect (modified) FTDI DTR to ESP-07 GPIO0.
- Connect FTDI RTS to ESP-07 REST.

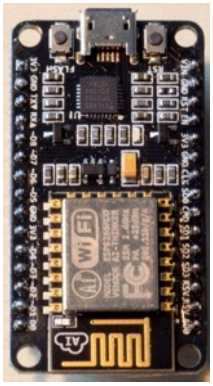
With this configuration, DTR will pull GPIO0 low and RTS will pull REST low. After uploading a firmware you'll probably need to disconnect DTR from GPIO and disconnect and reconnect power to the ESP-07.

Voltage levels ESP8266 ESP-07 with IO Adapter Plate

Input voltage: 3.3-3.6V on VCC
Logic Level: 3.3V

Preferably use an external power supply. Don't use the 3.3V from an Arduino, nor from an USB-Serial nor from an FTDI. The maximum current those devices can deliver is not enough. When using an external power supply, always the programmers GND with the GND of the external power supply.

243. NodeMCU ESP-12E Doit Devkit

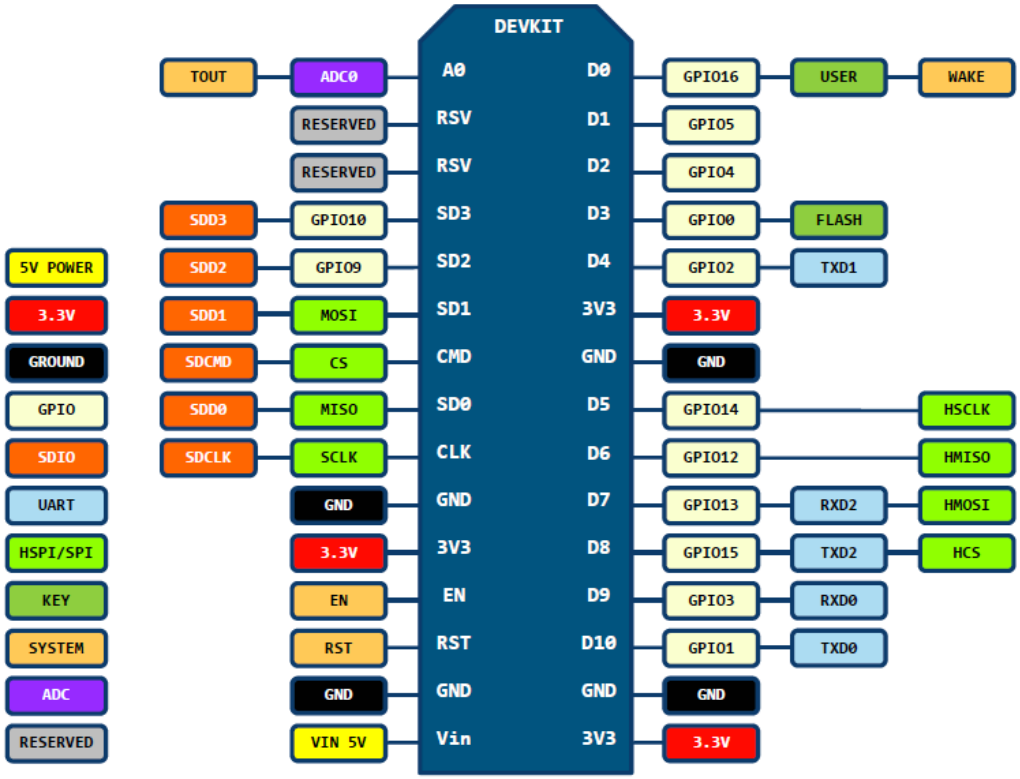



This board is made by doit.com and has lots of GPIO ports on the headers.

Specifications NodeMCU ESP-12E Doit Devkit

ESP-07	Micro USB port for power and UART
Default firmware: NodeMCU LUA	13 GPIO ports on headers
Default bitrate: 9600	I2C, 1-Wire, SPI
Flash button and Reset button	2 UARTS on headers CP2102 chip
Operation LED	Data LED connected to D4=GPIO2
5V Input	PCB etched antenna without connector
Breadboard friendly	

Layout/connections NodeMCU ESP-12E Doit Devkit



Voltage levels NodeMCU ESP-12E Doit Devkit

Input voltage 5V on Vin
Logic Level: 3.3V

Drivers USB to TTL Serial adapter for NodeMCU ESP-12E Doit Devkit

My board uses the CP210x USB to UART bridge. You can find the drivers for, Windows, Linux, OSx and Android at the following link:
https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

Boot load mode NodeMCU ESP-12E Doit Devkit

I didn't have to press any button to start boot load mode. The latest ESP8266Flasher tool, esptool-ck (–cd nodemcu) and the Arduino IDE will all send the right signal for the board to enter boot load mode.

I was not successful using the esptool.py script to upload new firmware.

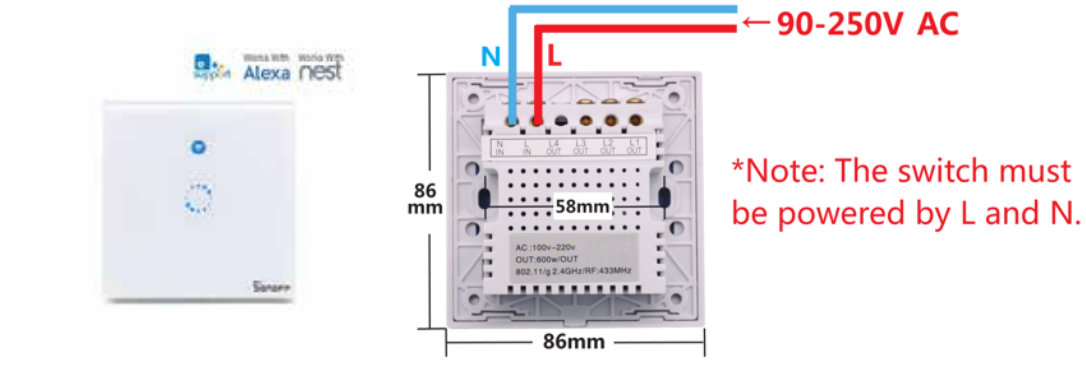
Board pins to ESP8266 GPIO pin numbers

The board numbers D0..D16 do not correspond to the ESP8266 GPIO numbers. The following code lines are automatically included when using any ESP8266 device. This way you can use D0..d10 instead of the GPIO port numbers.

124_ESP8266_GPIOmap.ino

```
//static const uint8_t D0   = 16;
//static const uint8_t D1   = 5;
//static const uint8_t D2   = 4;
//static const uint8_t D3   = 0;
//static const uint8_t D4   = 2;
//static const uint8_t D5   = 14;
//static const uint8_t D6   = 12;
//static const uint8_t D7   = 13;
//static const uint8_t D8   = 15;
//static const uint8_t D9   = 3;
//static const uint8_t D10  = 1;
```

244. Sonoff T1 UK 1 gang



This is smart wall mount touch light control an can be controlled through WiFi and RF 433 MHz It is based on the ESP8285 MCU, which is basically an ESP8266 with 1 MB integrated directly on the platter. This wall switch can be controlled by an iOS or Android phone via WiFi or through an RF 433 remote control.

https://www.itead.cc/sonoff-t1.html

244.1. Specifications Sonoff T1 UK 1 gang

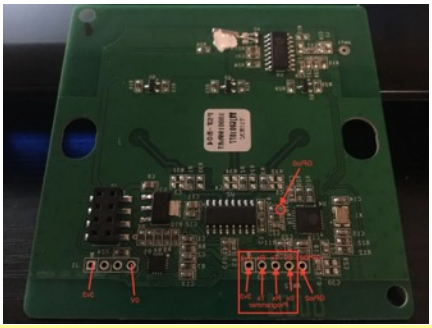
ESP8284	Wall mount
Default firmware: ITEAD (not open source!)	90 – 250 V AC
Default bitrate: 74880 baud	600 W/gang
Touch button and Reset button	WiFi 2.4 GHz & RF 433 MHz
Operation LED	Compatible with: Amazon Alexa, Google Assistant, IFTTT, Google Nest
Onboard antenna	Compatible with MQTT wit third party firmware Tasmota
Not breadboard friendly	SYN4703 433 MHz radio chip

Datasheets

- ESP8285 MCU/WiFi derived from ESP8266 http://www.espressif.com/sites/default/files/documentation/0a-esp8285_datasheet_en.pdf
- EFM8BB1 is used to extend the number of IO's needed to control 3 buttons in a 3 gang version https://www.silabs.com/documents/public/data-sheets/efm8bb1-datasheet.pdf
- SYN470R 433 MHz radio https://www.birdandgua.net/bird/wp-content/uploads/2016/09/SYN470R-Synoxo.pdf

Layout/connections Sonoff T1 UK 1 gang

The Sonoff T1 is a commercial product and not open source. Therefore, the information below is not complete. I derived it from: https://github.com/arendst/Sonoff-Tasmota/wiki/Sonoff-T1-UK-and-T1-EU.



The labels Rx and Tx in this picture are incorrect and should be swapped!

J1 *connected to the onboard EFM8BB1*
It's not clear to me how to use this EFM8BB1!

Nr	Name
1	VCC
2	DATA
3	CLOCK
4	GND

J2 *connected to the relay board (Clock wise)*

Nr	Name
1	Relay 3 (only applicable on 3 gang version)
2	Relay 1
3	Relay 2 (only applicable on 2/3 gang version)
4	Some kind of feedback

J3 *Use this one if you want to upload another firmware or an Arduino sketch*
Pin nr's start at the white marker.

Nr	Name	Use
1	VCC	3.3 V
2	Tx	Connect to Rx of a programmer
3	Rx	Connect to Tx of a programmer
4	GND	Ground

Voltage levels Sonoff T1 UK 1 gang

Input voltage: 3.3 on VCC
Logic Level: 3.3V

Use an external power supply. Don't use the 3.3V from an Arduino, nor from an USB-Serial nor from an FTDI. The maximum current those devices can deliver is not enough. When using an external power supply, always the programmers GND with the GND of the external power supply.

244.2. Programming the Sonoff T1 UK 1 gang

You can program the Sonoff like any other ESP module, either by the Arduino IDE or by using esptool-ck. First you need to enter boot load mode.

Boot load mode Sonoff T1 UK 1 gang

The original firmware is not open source, so you can't return to the original firmware, although the following link describes a way to back up your original firmware so you can restore is when needed.

https://github.com/mirko/SonOTA/issues/1

Important to know is that the original firmware is bound to the MAC address of your Sonoff, so you can't restore the firmware of another Sonoff.

After reading the note above, make the following connections:

- Connect J3:Vcc to 3.3V on an external power supply (don't use an Arduino for this purpose)
- Connect J3:Rx to Tx of a programmer (like an FTDI)
- Connect J3:Tx to Rx of a programmer
- Connect J3:GND to GND on an external power supply
- Connect J3:GND also to GND of the programmer

Depending on your version (from left to right 1 gang, 2 gang, 3 gang) you can find touch button 1 through the picture above.

- Touch button 1 and keep it touched
- Shortly press button 4
- Keep touch button 1 touched during about 2 seconds and then release button 1

The ESP8285 should now be in boot load mode. You could check this if you have serial monitor connected at 74880 baud. You should see the ESP8285 is now in boot mode (1,6).

Arduino IDE to program Sonoff T1 UK 1 gang with Arduino IDE

You can program the Sonoff T1 through the Arduino IDE, by first entering the boot load mode (see the previous paragraph).

- Settings Arduino IDE
  - Board: "Generic ESP8285 module"
  - CPU Frequency: "80 MHz"
  - Flash Size: "1M (512K SPIFFS)"
  - Upload Speed: "115200"

Upload another firmware to Sonoff T1 UK 1 gang

First enter the boot load mode (see previous paragraph). If you've used Serial Monitor to check the boot mode of the ESP8285, close it after entering boot load mode, otherwise the serial port of your programmer is unavailable.

Use esptool-ck to upload the firmware of your choice.

```
esptool –cp <serial-port> -ca 0x00000 –cd none –cf <firmware>
```

244.3. Tasmota firmware on the Sonoff T1 UK 1 gang

Tasmota is an open source alternative for the original firmware of iTead Studio. There are two ways to get the Tasmota firmware:

- Download a bin file from https://github.com/arendst/Sonoff-Tasmota/releases
- Or create your own modified Tasmota firmware through the Arduino IDE: https://github.com/arendst/Sonoff-Tasmota/wiki/Arduino-IDE

The downloaded or modified firmware can be uploaded to the Sonoff T1 as described in the previous paragraph. You can also upload Over The Air, but that is not described in this document.

You can find more information about Tasmota at: https://github.com/arendst/Sonoff-Tasmota/wiki.

Configure Tasmota to connect to the correct WiFi

By default Tasmota will try to connect to WiFi SSID indebuurt1 or indebuurt2, you can change this behavior by either making the correct changes in the source file and create a modified firmware, or you can enter the WiFi config mode as described at 4 short presses in paragraph "Button usage".

You can also change other settings like MQTT, but that is beyond the scope of this document.

Configure Sonoff module

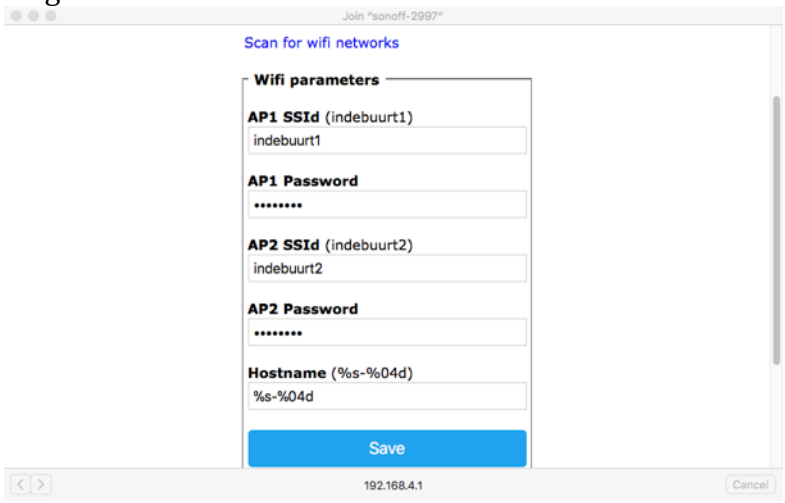
After configuring the correct WiFi you can open the Configuration website of Tasmota by browsing to its IP address.

- First go to CONFIGURATION, CONFIGURE MODULE and then select the correct Sonoff T1 (1/2 or 3 gang).

Button usage

Button 1 can be used to turn on/off the relays, but also to perform a specific task. Since my Sonoff T1 with Tasmota wasn't very stable, I used a terminal connection (like Serial Monitor) to check the output of Tasmota on the screen. More information can be found at: https://github.com/arendst/Sonoff-Tasmota/wiki/Button-Usage

- 4 short presses
  The LED will now start blinking and Tasmota will enter AP mode with SSID: sono-xxxx and IP 192.168.4.1. During 3 minutes you can tell the Sonoff T1 which WiFi it should connect to. You can abort the WiFi config mode by pressing button 1 again.



- 5 short presses
  The LED will now start blinking and Tasmota will enter WPS mode, so your SSID and key can be configured by using the WPS button on your WiFi access point.
- 6 short presses
  Tasmota will now restart.
- 7 short presses
  The LED will now turn on and Tasmota will start OTA (Over The Air) download of a newer firmware. Before you can use this, you should enter the address of the webserver serving the firmware. By default this is a local webserver with the following URL: http://domus1:80/api/arduino/sonoff.ino.bin

ESP32 WiFi modules

This section describes the ESP32 WiFi modules.

245. Common ESP32

The ESP32 is the successor of the ESP8266 microcontroller also developed by the Chinese manufacturer Espressif

245.1. Specifications ESP32

- Xtensa Dual-Core 32 bit LX6 microprocessor at 160 or 200 MHz
- External Flash 512 KiB to 4 MiB
- 802.11 b/g/n/e/I protocol
- WiFi Direct (P2P) and soft-AP
- Bluetooth: v4.2 BR/EDR and BLE
- 12-bit SAR ADC up to 18 channels
- 2x 8-bit DAC's
- 10x touch sensors
- Temperature sensor
- 4x SPI
- 2x I2C
- 2x I2S
- 3x UART
- Hall effect sensor
- Ultra-low power analog pre-amplifier

245.2. ESP8266 vs ESP32

Specifications	**ESP8266**	**ESP32**
MCU	Xtensa Single-Core 32-bit L106	Xtensa Dual-Core 32-bit LX6
WiFi 802.11 b/g/n	HT20	HT40
Bluetooth	none	4.2
Typical frequency	80 MHz	160 MHz
SRAM	160 KB	512 KB
SPI-Flash	up to 16 MB	up to 16 MB
GPIO	17	36
PWM	8 software channels	1 hardware channel 16 software channels
SPI / I2C / I2S / UART	2 / 1 / 2 / 2	4 / 2 / 2 / 3
ADC	10-bit	12-bit
CAN	none	yes
Ethernet MAC interface	none	yes
Touch sensor	none	yes
Temperature sensor	none	yes

245.3. Datasheet ESP32

- https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

245.4. ES32 models

There is a large variety of ESP32 boards on the market.

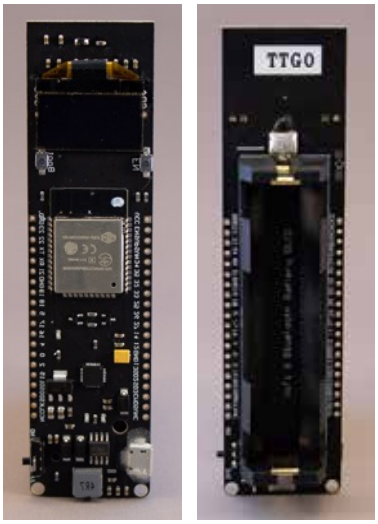
- The ESP32 SOC
  Just the ESP32 chip so you can solder it yourself.
- ESP32 development kits
  - With USB port, headers and Boot and Enable buttons
  - An OLED 128x64 screen added to it
  - 18650 battery-holder and charger added to it
  - LORA SX1278 or SX1276 chips added to it.

245.5. Usage of the ESP32

The ESP32 is very versatile and can be used in different ways.

- As a "standalone" MCU:
  - Programmed by Arduino IDE.
  - By uploading Lua scripts.
  - By uploading MicroPython scripts.
  - Sensors and/or actuators can be connected through various GPIO ports.
- As an MCU communicating with another MCU (for example an Arduino) via I2C or Serial, wired or wireless.
  - Programmed by Arduino IDE.
  - By uploading Lua scripts.
  - By uploading MicroPython scripts.
  - Sensors and/or actuators can be connected through various GPIO ports.
- As a Serial to Wi-Fi device.
  - By sending Lua or AT commands from another MCU (for example an Arduino).

246. TTGO ESP32 WiFi & Bluetooth Battery OLED

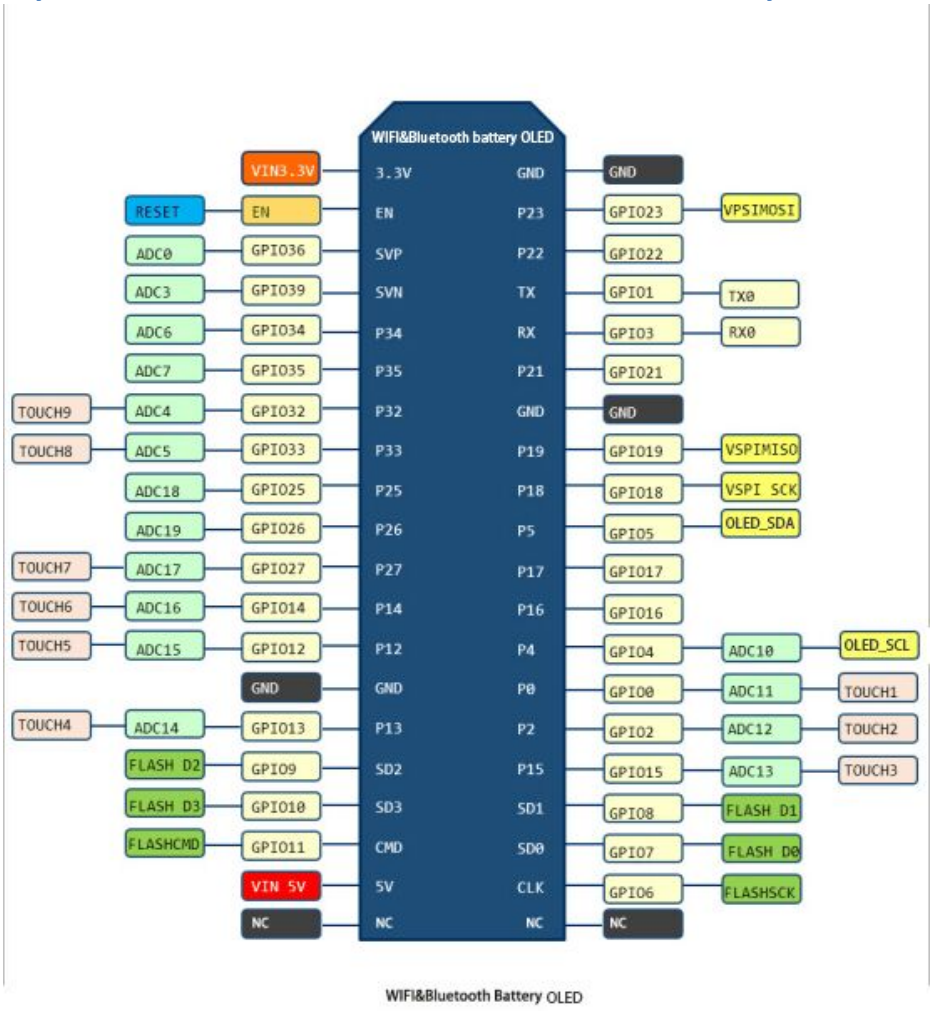


This development board has all the features of the ESP32 (WiFi, MCU, Bluetooth, etc..) but also an USB port, an OLED 128x64 screen and an 18650 battery holder including charge functionality. Everything you need to start your project.

Specifications TTGO ESP32 WiFi & Bluetooth Battery OLED

ESP-WROOM-32	Micro USB port for power and UART
18650 Battery slot (+charging)	Power switch
Charging system with protection, 0.5 A (green LED: full, red LED: charging)	Data LED connected to D0=GPIO16
Full ESP32 pins breakout	1A output
OLED screen (OLED_SCL=GPIO04, OLED_SDA=GPIO05)	Enable button and Boot button
Default bitrate:	PCB etched antenna without connector
Default firmware:	13 GPIO ports on headers
WiFi 802.11 b/g/n (up to 150 Mbps)	I2C, 1-Wire, SPI, DAC, ADC, PWM, Touch
Bluetooth v4.2 BR/EDR and BLE	CP2102 chip

Layout/connections TTGO ESP32 WiFi & Bluetooth Battery OLED



Voltage levels TTGO ESP32 WiFi & Bluetooth Battery OLED

Input voltage:

- 5V on VIN5V
- 3.3V on VIN3.3V
- 3.7V through 18650 battery

Logic Level: 3.3V

Drivers USB to TTL Serial adapter for TTGO ESP32 WiFi & Bluetooth Battery OLED

My board uses the CP210x USB to UART bridge. You can find the drivers for, Windows, Linux, OSx and Android at the following link:
https://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

Boot load mode TTGO ESP32 WiFi & Bluetooth Battery OLED

It is very easy to start boot load mode with this module. As soon as your upload software starts connecting, press the BOOT button and keep it pressed until your upload starts. A RST can be send by the upload software, directly after uploading, setting your ESP32 in run mode.

247. ESP32-CAM

This ESP32 module comes with a small camera module a built-in FLASH and an SD card.

247.1. Specifications ESP32-CAM

HW ESP32S up to 160 MHz	Supports multiple sleep modes
8 GPIO ports on headers	Embedded Lwip and FreeRTOS
WiFi b/g/n	Firmware update: - local through USB-Serial convertor - remote upgrade (FOTA)
Bluetooth v4.2 BR/EDR and BLE standards	No USB-Serial convertor
Built-in 520 KB SRAM	OV2640 Camera module
External 4M PSRAM	Also supports OV7670 Camera module
Supports: Serial, SPI, I2C, PWM, ADC, DAC	Built-in FLASH
Power supply: 5V	Onboard antenna
TF-card slot, support 4GB	SPI Flash: 32Mbit
UART default baudrate: 115200 bps, through an external USB-Serial convertor	

247.2. Documentation ESP32-CAM

- Datasheer: https://loboris.eu/ESP32/ESP32-CAM%20Product%20Specification.pdf
- Documentation: https://randomnerdtutorials.com/esp32-cam-video-streaming-face-recognition-arduino-ide/

247.3. Layout/connections ESP32-CAM

ESP32-CAM	ESP32	Function
U0R	GPIO3	UART0 Rxd
U0T	GPIO1	UART0 TxD
GPIO	GPIO0	GND: Flash mode

MicroSD card reader

ESP32-CAM	ESP32	Function
GPIO14		CLK
GPIO15		CMD
GPIO2		Data 0
GPIO4		Data 1 (also FLASH LED)
GPIO12		Data 2
GPIO13		Data 3

247.4. First time preparation of Arduino IDE for ESP32-CAM

This section describes how to add support for the ESP32-CAM to tthe Arduino IDE. More details can be found at: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-mac-and-linux-instructions/

- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and paste the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (you can add multiple links here, by using a ';' as seperator. https://dl.espressif.com/dl/package_esp32_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for ESP32 and select the ESP32 BY ESPRESSIF SYSTEMS
- Click on the INSTALL button.
- Press the CLOSE button.
- AI-THINKER ESP32-CAM is now listed at TOOLS, BOARD

247.5. Programming a sketch on the ESP32-CAM

Follow the following steps to program ESP32-CAM:

- Connect your ESP32-CAM to your computer by using an USB to Serial convertor
  - Connect 5V to 5V (set your USB to Serial convertor to 5V)
  - Connect U0R to Txd
  - Connect U0T to RxD
  - Connect GND to GND
- Set your ESP32-CAM module in flashing mode, by connecting GPIO to GND, then press RST (at the backside of the module).
- Load the sketch you want to compile and upload.
- Choose TOOLS, BOARD, then select the AI-THINKER ESP32-CAM.
- Choose TOOLS, PROGRAMMER, AVRISP mkll.
- Choose TOOLS, SERIAL PORT, then select the correct Serial Port
- Choose FILE, UPLOAD
- After uploading the sketch, set your ESP32-CAM back in running mode by disconnecting GPIO from GND, then press RST again.

247.6. Sample CameraWebServer

This sketch will start a video streaming WebServer with face detection and face recognition.

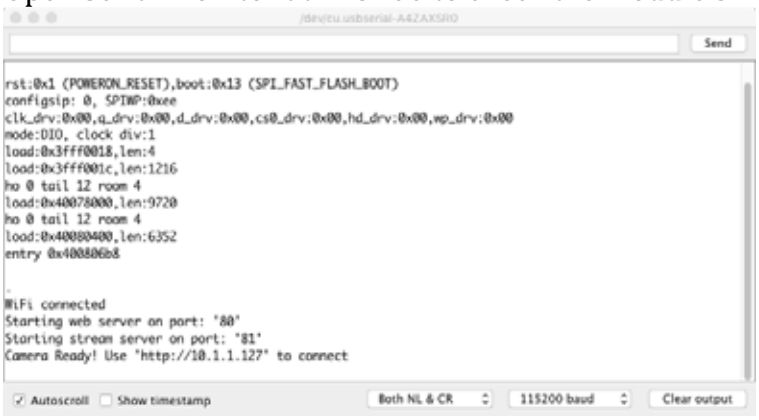
- Choose TOOLS, BOARD, then select the AI-THINKER ESP32-CAM.
- Go to FILE, EXAMPLES, ESP32, CAMERA, CameraWebServer.
- Edit the part where the SSID and PASSWORD (=WPA2-KEY) is declared and replace ********** with your SSID and your WPA2-KEY

```
const char* ssid = "*********";
const char* password = "*********";
```

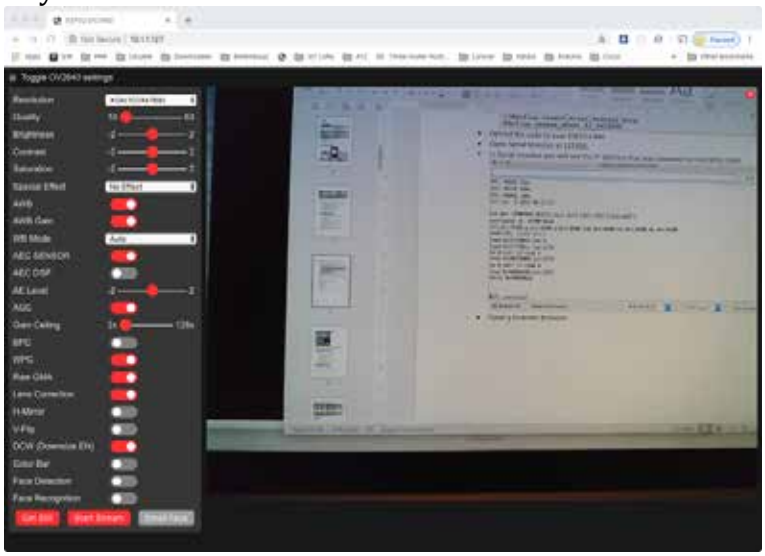
- Comment all camera models and uncomment the AI_THINKER camera.

```
//#define CAMERA_MODEL_WROVER_KIT
//#define CAMERA_MODEL_ESP_EYE
//#define CAMERA_MODEL_M5STACK_PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE
#define CAMERA_MODEL_AI_THINKER
```

- Upload the code to your ESP32-CAM.
- Open Serial Monitor at 115200 to check the module's IP address.



- Open a Internet Browser
- Press GET STILL to capture a photo or press START STREAM to stream live video to your browser.



248. Programming ESP32 with Arduino IDE

Just like the ESP8266, the ESP32 can be used as a WiFi to Serial module to connect and Arduino with WiFi, but you can skip the Arduino and use the ESP32 own microprocessor with much better specs than the Arduino.

248.1. Connections

The connections are dependent of the ESP32 module/board. Most ESP32 modules are sold as a development kit, so you don't need to make any connections.

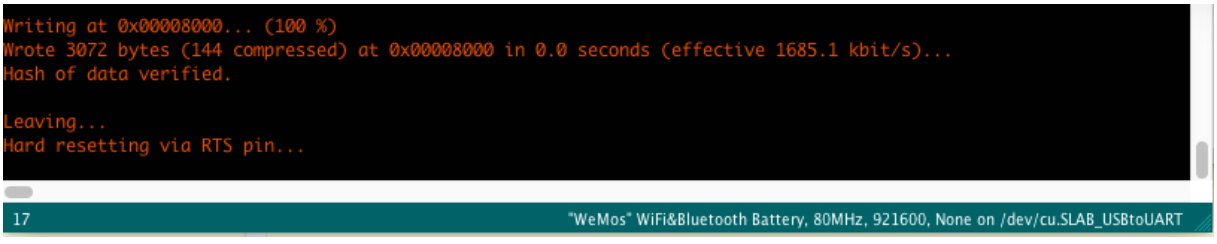
248.2. First time preparation of Arduino IDE for the ESP32 board

At this point (June 2018) it is not yet possible to use the Additional Boards Manager. You will need to do some command line typing.

Instructions for Windows, OSx, Debian, Fedora and OpenSUSE, can be found at:
https://github.com/espressif/arduino-esp32/tree/master/docs/arduino-ide

248.3. Programming the ESP32

- Choose TOOLS, BOARD
- Select your ESP32 board (or a similar one)
- Choose TOOLS, FLASH FREQUENCY and choose 80 MHz
- Choose TOOLS, UPLOAD SPEED and choose 912600
- Choose TOOLS, CORE DEBUG LEVEL and choose none
- Choose the correct serial port
- Open the sketch you want to upload
- Press the UPLOAD button and wait until it says: Connecting
- Now press the BOOT button and keep it press until Arduino IDE starts uploading



STM32 modules

Lorem ipsem??.

249. Common STM32

The STM32 is a family of ARM MCU's developed by STMicroelectronics.

250. Programming STM32 with Arduino IDE

Some of the STM32 modules can be programmed using the Arduino IDE.

251. Programming with System Workbench for STM32

Only a few STM32 modules can be programmed using the Arduino IDE, This is described in the previous chapter. This chapter will give you a head start in programming STM32 modules without the Arduino IDE. This has a very steep learning curve and I will only help you getting a "Hello World"/ "Blinky" application up and running.

There are several toolchains available, like Keil, IAR and MBED, but most of them are not for free. In this chapter, I'll describe the only toolchain I've found to be free and available for Windows, Linux and OSx and that is STM32 CubeMX and Eclipse with the Software Workbench for STM32 plugin from AC6 (a.k.a. SW4STM32).

251.1. Installing and configuring the toolchain

In this chapter, I will describe the installation of the two applications STM32 CubeMX and Eclipse and the plugin Workbench for STM32.

Install and configure: STM32 CubeMX

All STM32 modules are equipped with multiple GPIO busses, each with multiple GPIO pins. Most GPIO pins have multiple functions and modes. Besides this complex GPIO structure there are also multiple clock configurations possible. At the beginning of your application, you need to initialize the GPIO pins and clock configuration you are going to use. This is not an easy task, so most STM32 developers use specialized configuration tools to help them creating the needed project structure with the correct initialization lines in it. I choose STM32 CubeMX from STMicroelectronics and as they describe STM32 CubeMX as a: "*software configuration tool that allows the generation of C initialization code using graphical wizards.*".

This software is free, but you'll need to create an account first at:
https://www.st.com/content/st_com/en/user-registration.html?referrer=https%3a%2f%2fmy.st.com%2fcontent%2fmy_st_com%2fen%2fabout%2fst_company_information%2fwho-we-are.html

All files at st.com are listed with a part number. The part number for STM32 CubeMX is simply: STM32CubeMX, you can find more information about STM32CubeMX at the following URL:
https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html
The download link can be found at the bottom of that page. This download is a zip file with the installers for Windows, Linux and OSx. You'll notice that both the Linux (.linux) and the OSx installer (.app) are very small compared to the Windows installer (.exe). But in fact both the Linux and the OSx installer will extract the java installer inside the Windows installer .exe file. So whatever operating system you are using, you'll always need to have the .exe file. If you know how to unpack the java installer from the Windows installer, you don't need the Linux nor the OSx installer at all.

Use the following steps to install and configure the software, create your first project and to install the libraries for your first STM32 module.

- Download STM32CubeMX

- Install STM32 CubeMX
- The software will store it's libraries in your home folder in a folder named: STM32Cube. You can also use this folder to store your projects.

Install and configure: Eclipse

Working with Eclipse as your toolchain, you can choose between 2 flavors.

- Software Workbench for STM32 Bare Metal (SW4STM32)
  - This is a version of Eclipse specialized for STM32 development. Too bad, this version crashes very often on OSX, so this flavor is not covered in this book.
- Eclipse with the Software Workbench for STM32 plugin.
  - For this toolchain, you'll need to install the regular version of Eclipse and then install the SW4STM32 plugin.

Java Runtime Environment (>= 1.7.0)

Install and configure: Software Workbench for STM32 (SW4STM32) plugin

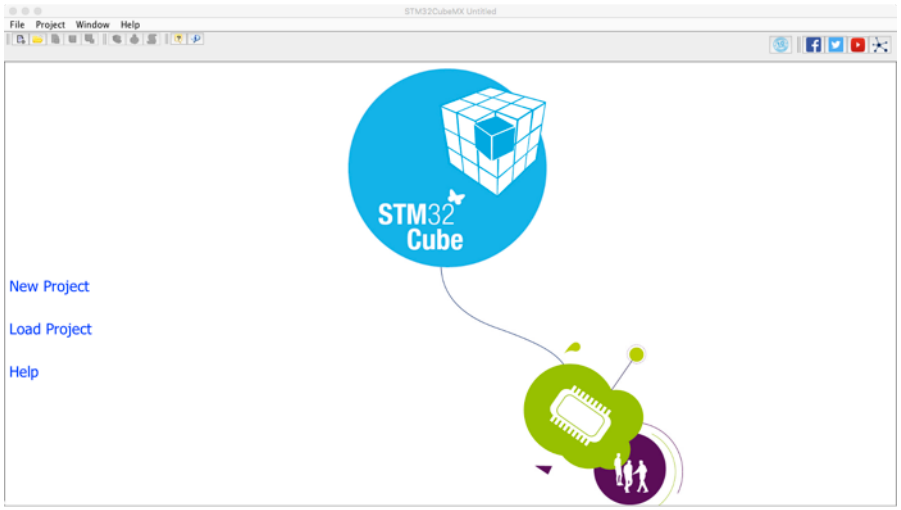
Software Workbench for STM32 is developed by AC6 and not by STMicroelectronics. Still STMicroelectronics lists SW4STM32 and the SW4STM32 plugin as

251.2. Hello World/Blinky

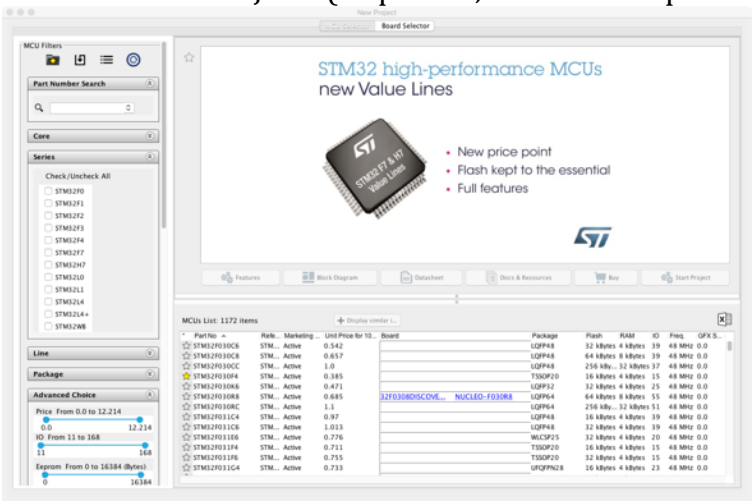
In this paragraph I'm going to create the equivalent for Hello World: Blinky. Blinking an LED on my cheap STM32F030F4-V2 breakout board. After some trial on error, I found out that the onboard LED is connected to PA4. So I'm developing an application that toggles the GPIO PA4 once every second. I will start with generating the basic initialization code with STM32 CubeMX. After that, I'll develop and test my Blinky application with Eclipse and SW4STM.

Generating Project folder structure and initialization with STM32 CubeMX

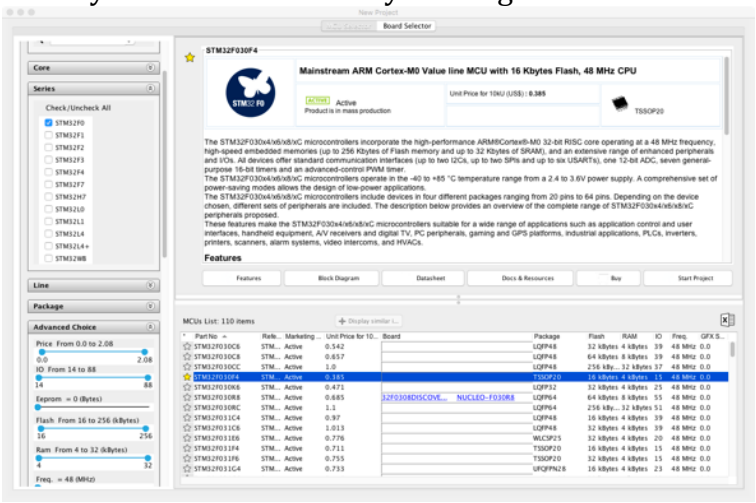
- Start STM32 CubeMX.



- Click on NEW PROJECT (be patient, it doesn't respond right away).

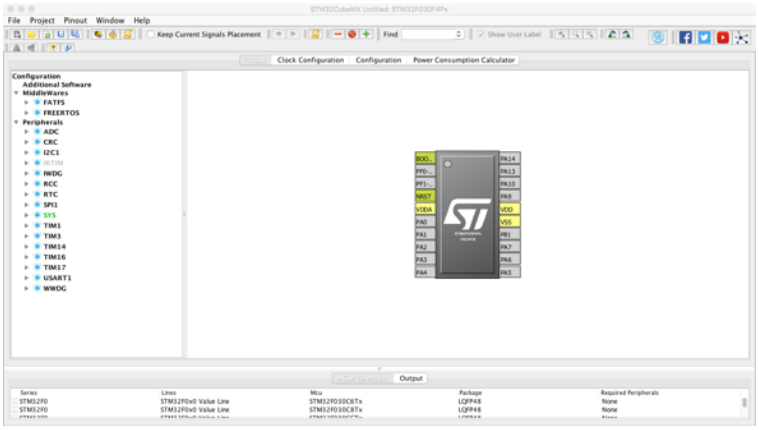


- Select the correct tab depending on whether you want to install the libraries for a Board (Board Selector) or for a bare MCU (MCU Selector). If your board is not listed, choose the MCU Selector. In this case I'll choose the MCU Selector to install the libraries for my cheap Chinese STM32F030F4-V2 breakout board.
- Next, search for the MCU you want to install (example: STM32F030F4). It can be useful to mark this MCU as one of your favorites, by clicking on the star in front of it.
- Select your MCU in the list by clicking on its name

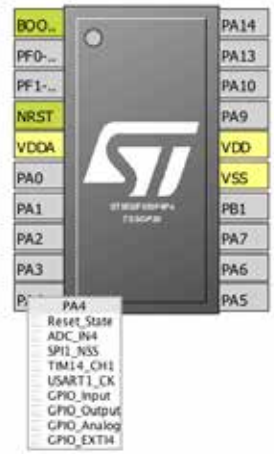


- You will now see the features of your MCU.

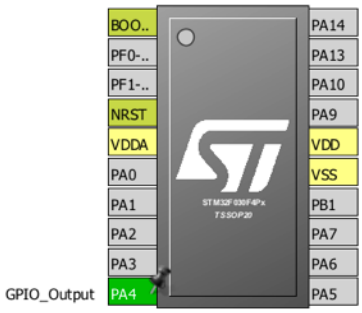
- Next click on START PROJECT and you will see a graphical presentation of your MCU.



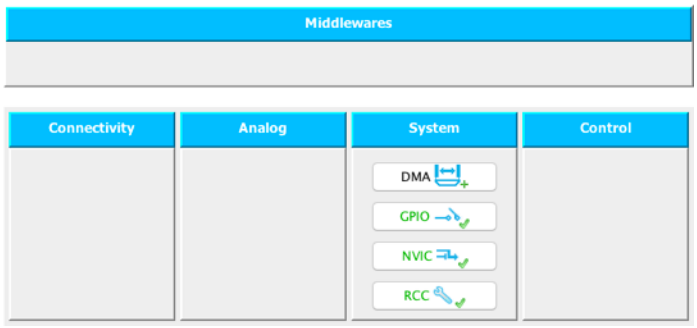
- A basic set of pins are already preselected and you can add pins by clicking on them and then select its purpose. In this example, I'm going to select PA4 as this GPIO is connected to the onboard LED.



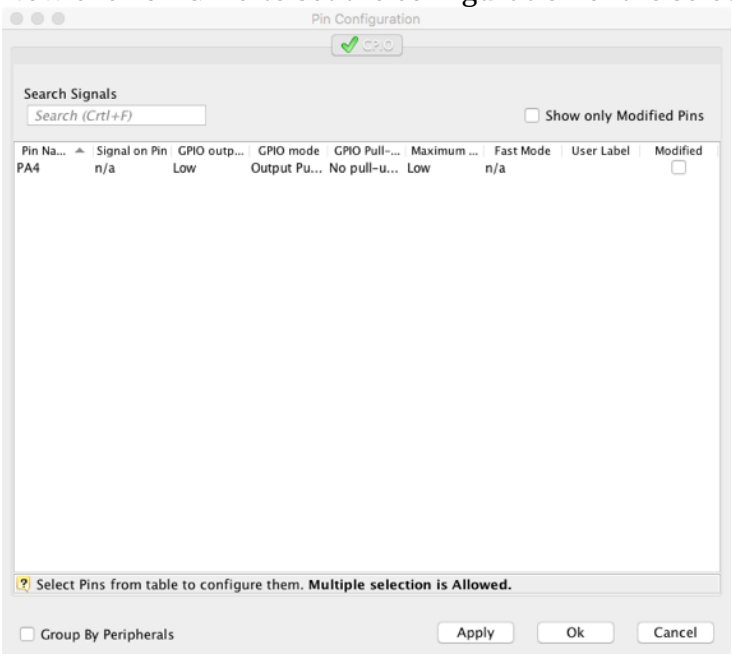
- The purpose for this GPIO-port will be driving an LED, so I will select GPIO_OUTPU.



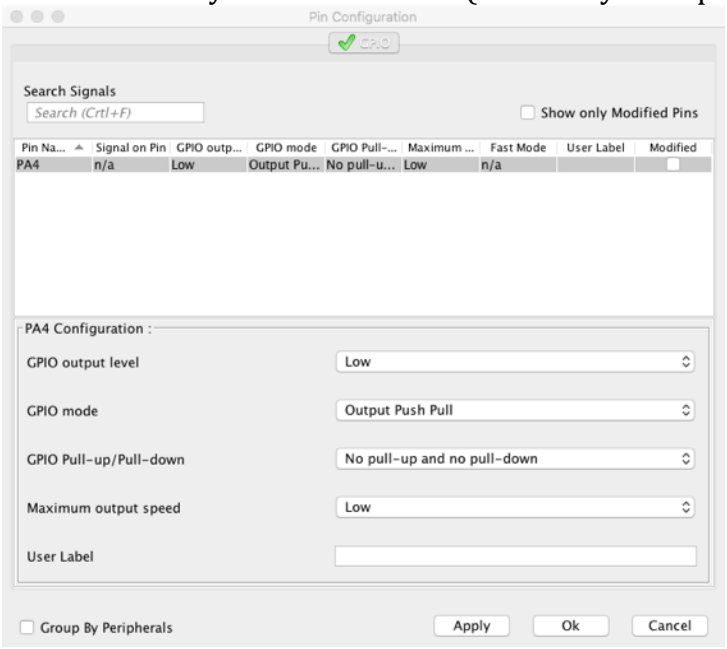
- Next is to select the CONFIGURATION tab.



- Now click on GPIO to set the configuration of the selected GPIO-pins.

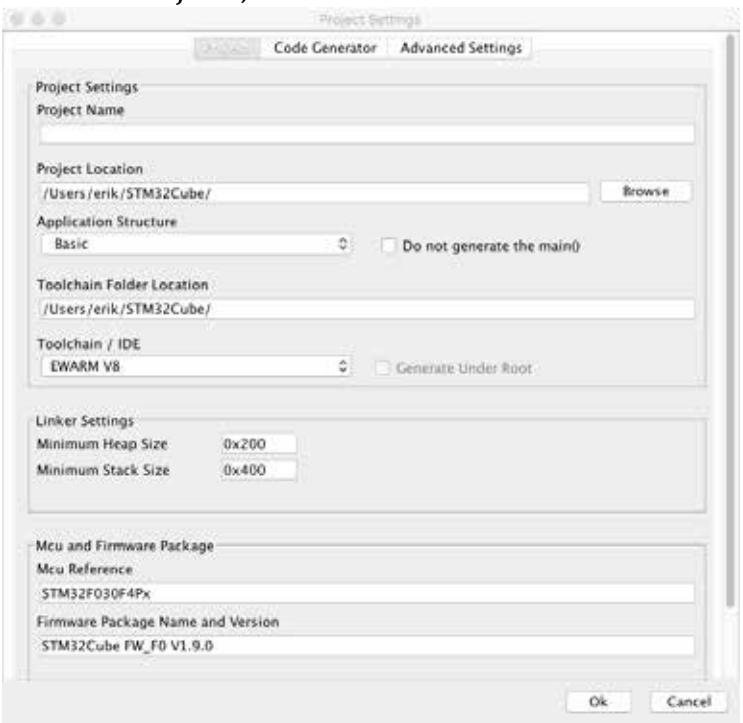


- Click on one of your selected Pins (PA4 in my example).



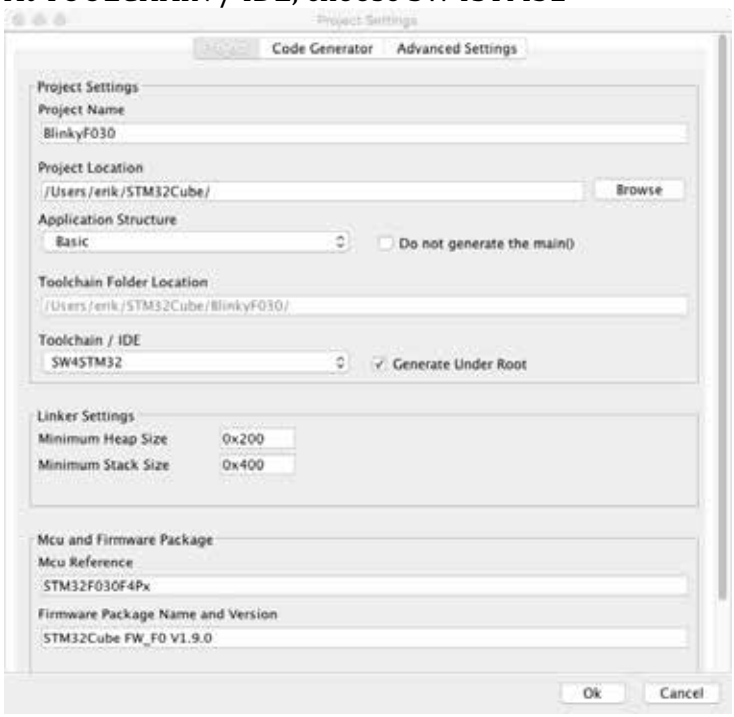
- Most settings are already set for my purpose:
  - GPIO OUTPUT LEVEL: *Low*
    PA4 will be low at the start of my application

  - GPIO MODE: *Output Push Pull*
    With this setting, you can set PA4 either too HIGH or LOW with your application.
  - GPIO PULL-UP/PULL-DOWN: *No pull-up and no pull-down*
    Since I'm not going to read the status of a switch or button, I will no need an internal PULL-UP nor an internal PULL-DOWN resistor.
  - MAXIMUM OUTPUT SPEED: *Low*
    A low speed setting is needed when blink an LED only a few times per second.
  - USER LABEL: *empty*
    I will set the USER LABEL to ONBOARDLED, so I can refer to this pin as ONBOARDLED.
- Click on OK to apply this configuration and close the dialogue board.
- In the next step I will save this project, so I use these settings again.
- Choose: PROJECT, SETTINGS:

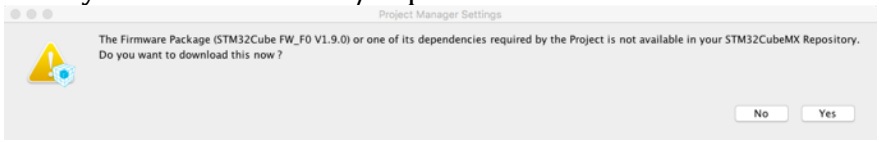


- Fill in a name for your project (this will create a corresponding folder in the STM32 folder.

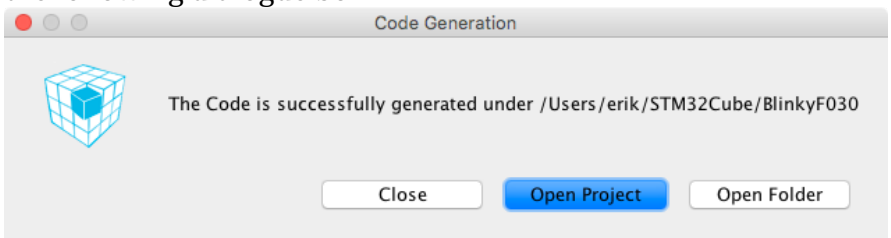
- At TOOLCHAIN / IDE, choose SW4STM32



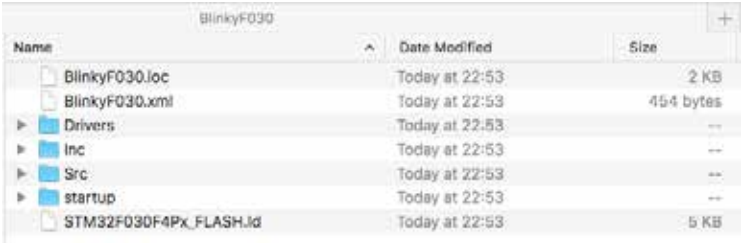
- Now press OK.
- If the firmware package for the selected MCU/Board has not been downloaded yet. The following dialogue box will appear. Press YES to download the needed library to the STM32Cube/Repositories folder.



- At this point all settings made with CubeMX are now stored in a CubeMX project file called ~/STM32Cube/BlinkyF030/BlinkyF030.ioc.
- The next step will be to generate the SW4STM32 project folder structure with the initialization code in place.
- Choose PROJECT, GENERATE CODE. After generating the code, this will result in the following dialogue box:



- Too bad, but on my Mac clicking on OPEN PROJECT resulted in an error message, so I clicked OPEN FOLDER here.

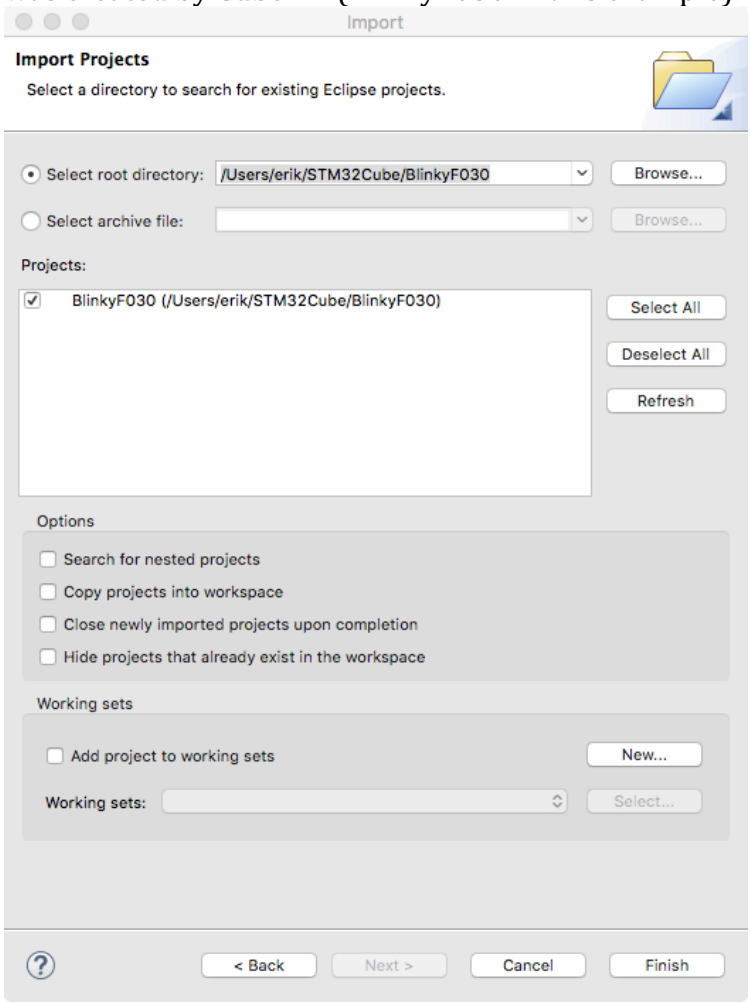


- Close STM32 CubeMX.

- The basic project structure has now been created, so the next step is to start Eclipse with SW4STM32.

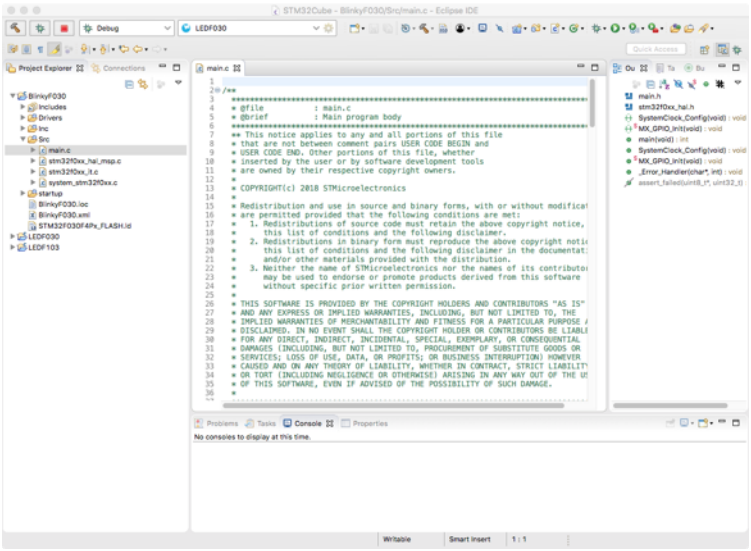
Developing and testing with Eclipse and SW4STM32

- Start Eclipse software and choose the workspace you want to use (~/STM32Cube in this example).
- Choose FILE, IMPORT.
- Choose: GENERAL, EXISTING PROJECTS INTO WORKSPACE and click on NEXT.
- Click BROWSE at the SELECT ROOT DIRECTORY and select the project folder that was created by CubeMX (BlinkyF030 in this example).



- Now click on FINISH

- In the PROJECT EXPLORER tab browse to BLINKYF030/SRC and double click on MAIN.C.



- Browsing this main.c file, you will find at least the initialization of the ONBOARDLED as I called my PA4 pin.
- Search for the line `while (1)`. This s the equivalent of the `void loop()` in Arduino programming. Type the following lines in between the curly brackets below the line `while (1)`.

```
HAL_GPIO_TogglePin(ONBOARDLED_GPIO_Port, ONBOARDLED_Pin);
HAL_Delay(1000);
```

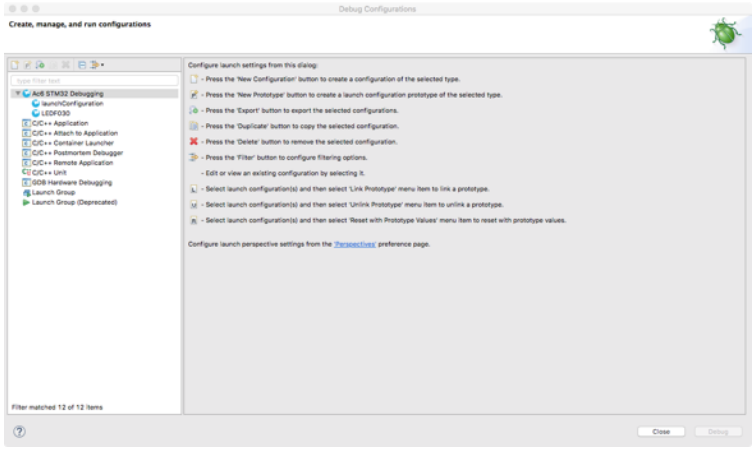
- If you want to use the original name of the GPIO pin, you'll need to replace the first line with the following line:

```
HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_4);
```

This means, that you want to use pin nr 4 on GPIO port A (= PA4).

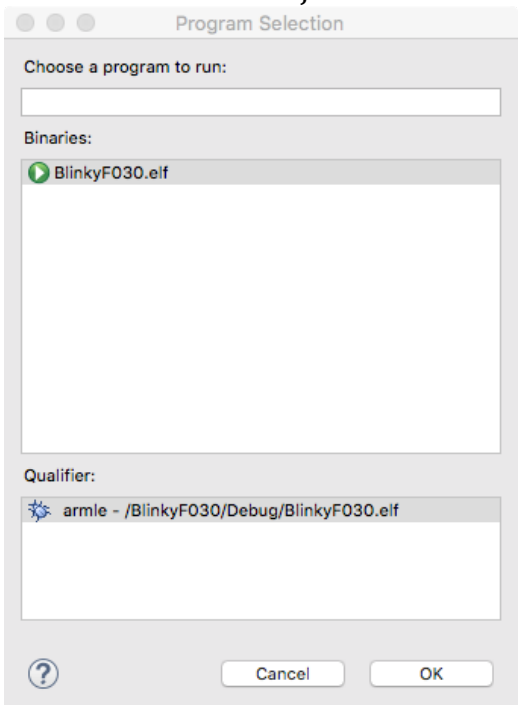
- Now you can compile this code by choosing: PROJECT, BUILD PROJECT. This will result in the creating of the following binary file: ~/STM32Cube/BlinkyF030/Debug/BlinkyF030.elf.
- Save the changes to main.c.
- The following steps are needed to upload the binary .elf file to the STM32 module and start debugging/running your application. At first this was a sheer mystery to me. To a certain account it still is, so I'm not sure if the following steps are the correct way, but it works.

- Choose: RUN, DEBUG CONFIGURATIONS.



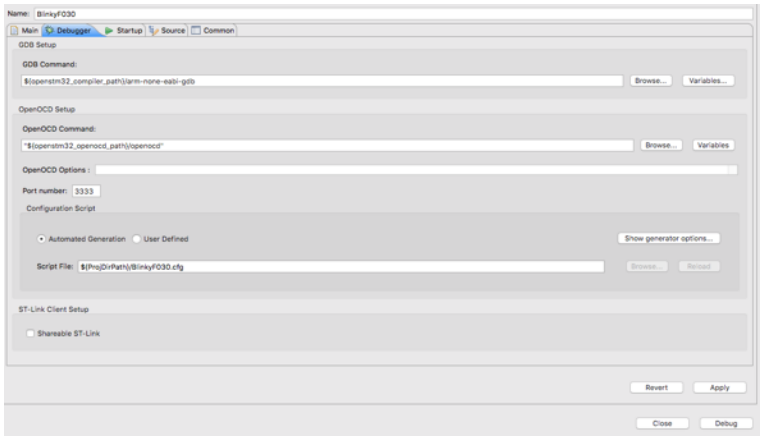
- In the left column, select AC6 STM32 DEBUGGING, right click on it and choose: NEW CONFIGURATION.

- Give this debug configuration a name, I copied the name of this project, so: BlinkyF030
- Click on SEARCH PROJECT



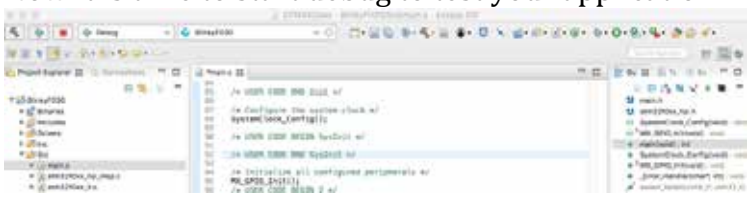
- No select the .elf file you've just created and click on OK.
-
- Make a connection between the ST Link V2 and your MCU and computer.

  - Connect the correct pins form the ST Link adapter to your MCU (in this case GND-GND, 3.3V-3.3V, SWDIO-DIO and SWCLK-CLK).
  - Connect your ST Link V2 adapter to an USB port
  - Also make sure your STM32 module is in BOOT0 mode. In this case remove the jumper from BOOT0 and press the RESET button.
- Next, set the configuration needed to communicate with your MCU.
- Select the tab DEBUGGER.

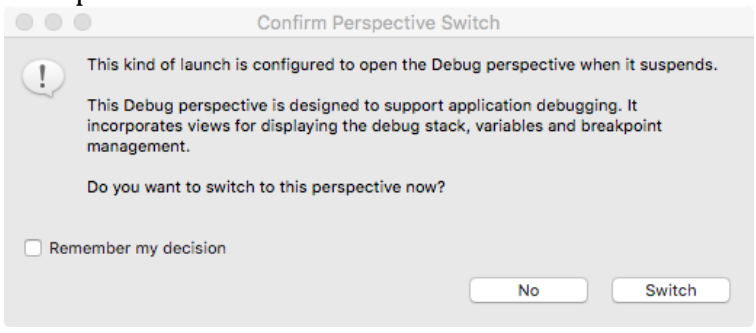


- Click on SHOW GENERATOR OPTIONS

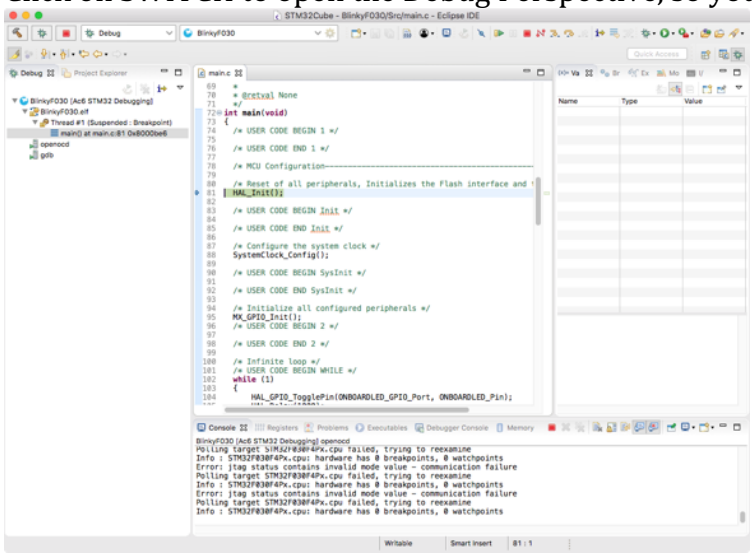
- If you're board has an ST Link build in, leave RESET MODE to *Connect under reset*. In my case, I've just an external ST Link adapter, so I choose: *Software System Reset.*
- Next click SHAREABLE ST-LINK and then click on APPLY and CLOSE.
- Now it is time to start debug to test your application.



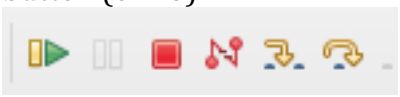
- Make sure the correct debug configuration BlinkyF030 is shown.
- Now press on the DEBUG button next to the hammer.



- Click on SWITCH to open the Debug Perspective, so you can follow your program.



- Debug is now paused at the line HAL_INIT(); you can step through your application or resume your application. I will choose for yellow/green RESUME button (or F8).



  - With my cheap STM32F030F4-V2 breakout board, the onboard LED didn't start blinking. I needed to close the jumper on BOOT0 and had to press RESET.
- If you press the red STOP button, the debug process will be stopped (but your STM32 module will continue the new application.
- You can close the Debug tab to return to main.c.

251.3. Summary

- Eclipse
- FILE, NEW, PROJECT
- C/C++, C/C++ Project, NEXT
- C++ MANAGED BUILD, NEXT
- PROJECT NAME: *<project name>*
- PROJECT TYPE: EXECUTABLE, AC6 STM32 MCU PROJECT, AC6 STM32 MCU GCC, NEXT
- NEXT
- Tab: MCU, SERIES: STM32F1, MCU: STM32F103C8Tx, NEXT
- HARDWARE ABSTRACTION LAYER (CUBE HAL)
- (only the first time: DOWNLOAD TARGET FIRMWARE)
- FINISH
- Open: ***<project name>*/srv/main.c**
- PROJECT, BUILD PROJECT
- RUN, DEBUG CONFIGURATIONS
- Right click on AC6 STM32 DEBUGGING, NEW CONFIGURATION
- NAME: <project name>
- SEARCH PROJECT
- Select <project name>.elf, OK
- TAB DEBUGGER
- SHOW GENERATOR OPTIONS, RESET MODE: SOFTWARE SYSTEM RESET
- APPLY, CLOSE
- CONFIRM PERSPECTIVE SWITCH, SWITCH (not after selecting: REMEMBER MY DECISION)
-

252. STM32F103C8T6 Minimum Development Board (aka Blue Pill)


This STM32 module can be found on Chinese web shops for very little money. It is well known as Blue Pill.

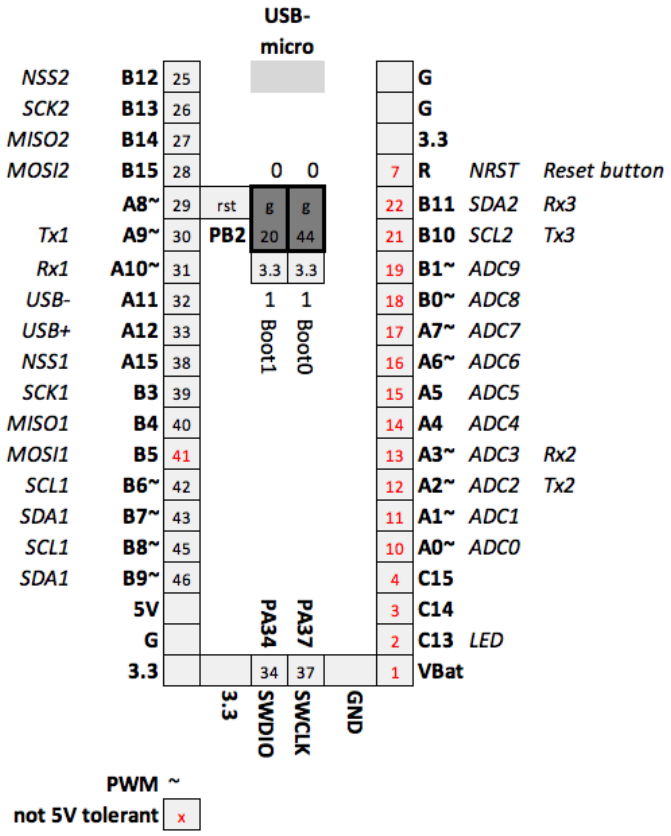
252.1. Specifications STM32F103C8T6 Minimum Development Board

- 32 bit MCU at 72 MHz, ARM Cortex-M3
- 20 KB SRAM
- 64 KB Flash
- 3.3V or 5V power
- 37 GPIO pins
  - Data level: 3.3V (some of them are 5V tolerant)
  - 15 x PWM
  - 10 x ADC
- 2x I2X
- 2x SPI
- 3x USART
- CAN
- RTC
- User LED: PC13 (GPIOC, GPIO_PIN_13)

252.2. Datasheet STM32F103C8T6 Minimum Development Board

- The datasheets can be downloaded at: https://www.st.com/en/microcontrollers/stm32f103c8.html
- A complete description of STM32F1 HAL and Low-layer drivers can be found at: https://www.st.com/content/ccc/resource/technical/document/user_manual/72/52/cc/53/05/e3/4c/98/DM00154093.pdf/files/DM00154093.pdf/jcr:content/translations/en.DM00154093.pdf

252.3. Connections STM32F103C8T6 Minimum Development Board



252.4. First time preparation of Arduino IDE for the STM32F1 family

This section describes how to add support for the STM32F1 family to the Arduino IDE.

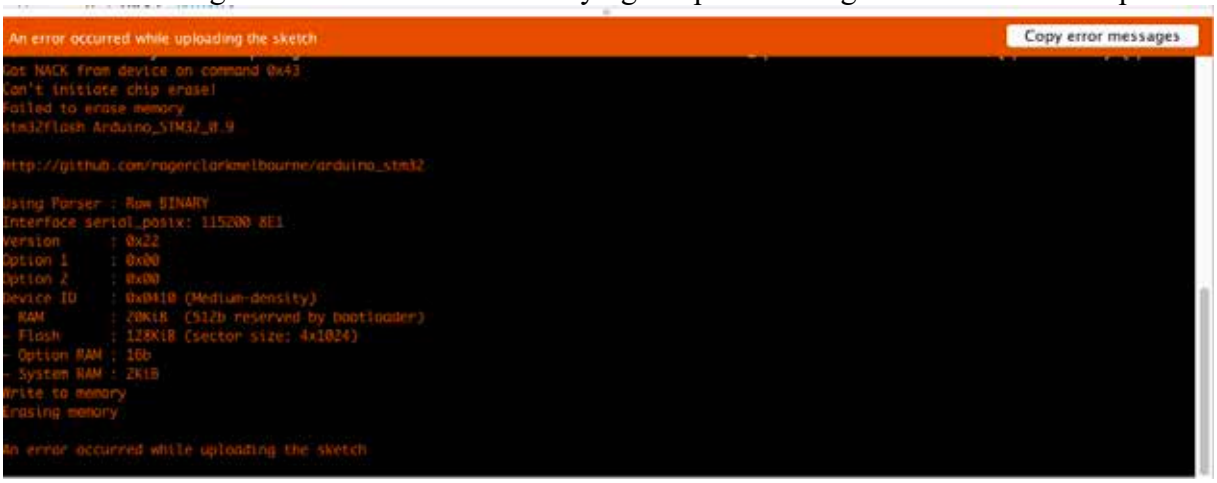
- Open Arduino IDE and go to FILE, PREFERENCES.
- Cut and past the following link in the box ADDITIONAL BOARDS MANAGER URL'S. (You can add multiple links here, by using a ';' as separator.)
  http://dan.drown.org/stm32duino/package_STM32duino_index.json
- Close PREFERENCES by clicking on the OK button.
- Go to TOOLS, BOARD, BOARDS MANAGER.
- Search for STM32F1 (this could take a few seconds) and select the STM32F1xx/GD32F1xx board definitions.
- Click on the INSTALL button, on a Windows computer you will now be asked to install (or update) the drivers for the STM32F1 family.
- Press the CLOSE button.
- The STM32F1 family boards are now listed at TOOLS, BOARD.
- During this process, all necessary libraries and some example sketches are also made available in Arduino IDE.

252.5. First time preparation of the Flash Memory of the STM32F103C8T6

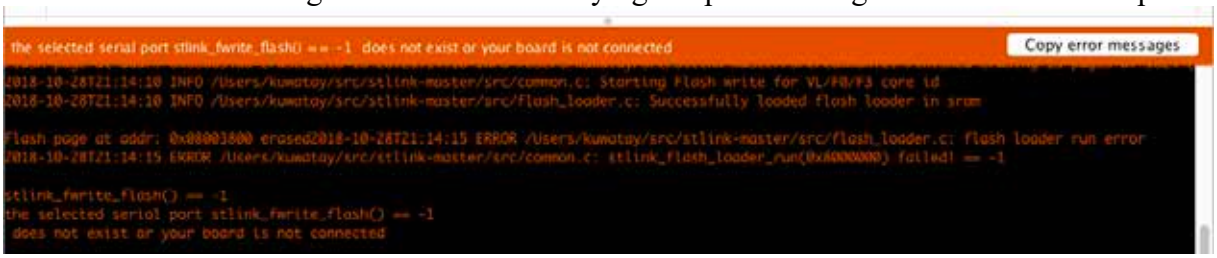
The Flash memory of My STM32F103C module was READ protected resulting in the following error messages when uploading a sketch

This resulted in the following error messages:

The error message below was shown when trying to upload through an USB-Serial adapter.



The second error message was shown when trying to upload through a ST-LINK V2 adapter.



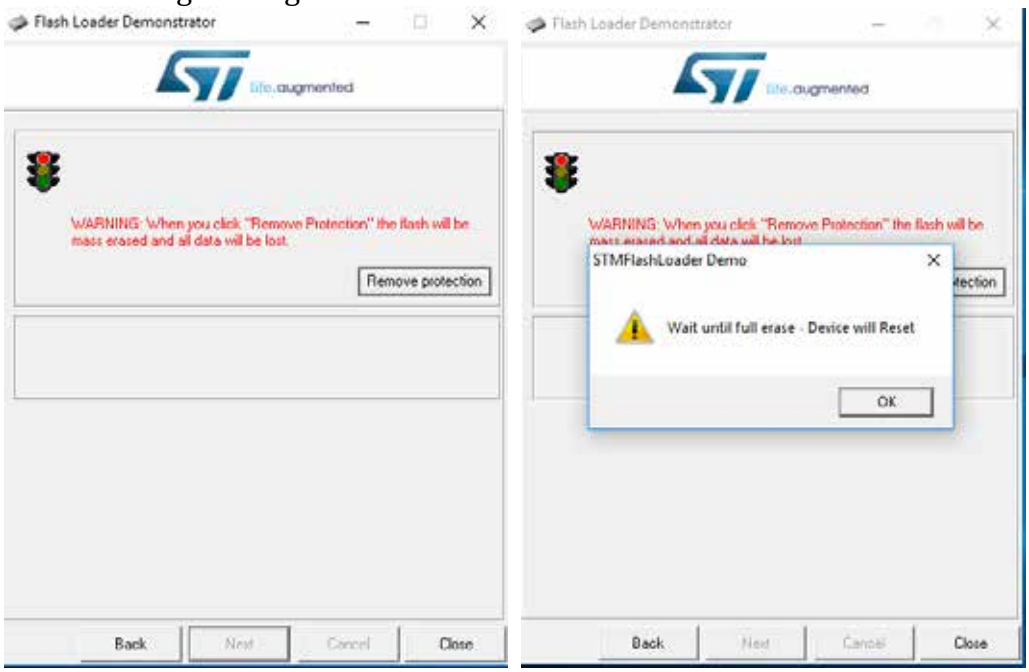
In case the Flash Memory of your STM32F103C8T6 is also locked, you'll need a PC with Windows (sorry for the Mac and Linux lovers) and either the ST-LINK utility or the STM32 Flash loader Demonstrator to unlock the Flash Memory.

Remove Read Out protection with the STM32 Flash Loader Demonstrator utility through an USB-Serial adapter

You can find this software at the following URL:
https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/flasher-stm32.license=1540484776179.product=FLASHER-STM32.html

- Connect an USB-Serial adapter as is described in the next paragraph: 252.6 Uploading a sketch to the STM32F1 family.
- Start the STM32 Flash loader Demonstrator utility.

- The following messages will be shown:

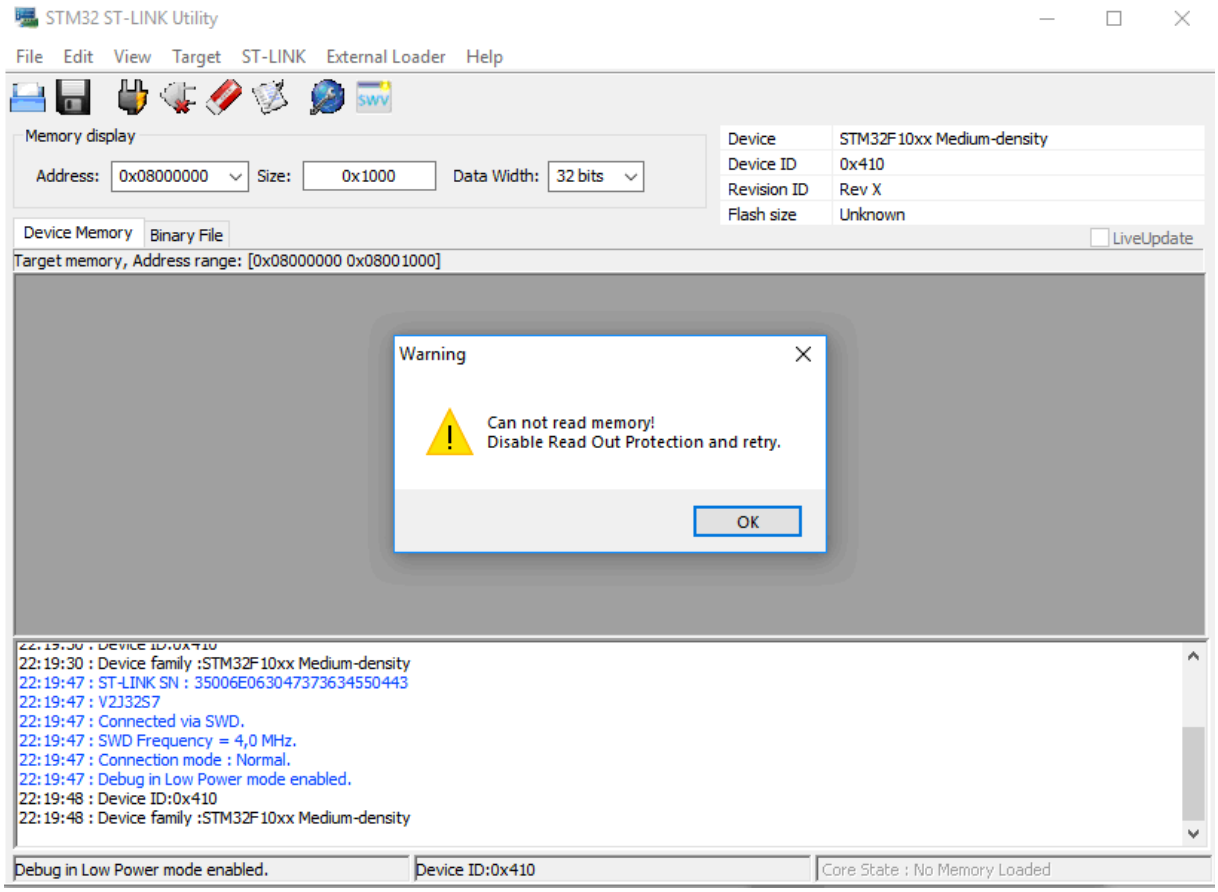


- Pres the REMOVE PROTECTION button (your Flash Memory will be erased!!!) and after that press on OK.
- The Flash Memory is now completely erased. You can stop the utility by clicking on OK.
- You will now be able to upload sketches.

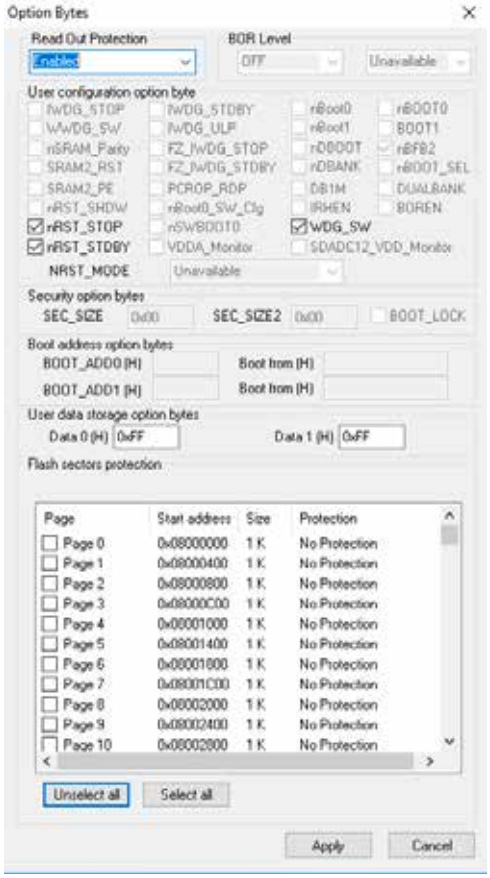
Remove Read Out protection with the STM32 ST-LINK Utility through an ST LINK adapter

You can find this software at the following URL:
https://my.st.com/content/my_st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/flasher-stm32.license=1540484776179.product=FLASHER-STM32.html

- Connect an USB-Serial adapter as is described in the next paragraph: 252.6 Uploading a sketch to the STM32F1 family.
- Start the STM32 ST-LINK Utility.
- Choose TARGET, CONNECT
- The following messages will be shown:



- Choose: TARGET, OPTION BYTES



- Now change READ OUT PROTECTION to DISABLED and press the APPLY button. Your Flash memory will be erased WITHOUT WARNING!.
- You will now be able to upload sketches.

252.6. Uploading a sketch to the STM32F1 family

There are 2 ways to upload a sketch to the STM32F1 family:

- Through an USB-Serial adapter
- Through an ST Link V2 adapter

Uploading a sketch by using an USB-Serial adapter.

- Connect GND of the USB-serial adapter to GND.
- Connect 3.3V of the USB-serial adapter to 3.3V.
- Connect Rx of the USB-Serial adapter to A9 (Tx1).
- Connect Tx of the USB-Serial adapter to A10 (Rx1).
- Place the Boot0 jumper on pin44 and 3.3V.
- Press the reset button.
- Choose TOOLS, BOARD, GENERIC STM32F103C.
- Choose TOOLS. VARIANT: STM32F103C8 (20k RAM, 64k Flash).
- Choose TOOLS, CPU SPEED(MHZ): 72MHZ (NORMAL).
- Choose TOOLS, UPLOAD METHOD: SERIAL.
- Choose TOOLS, OPTIMIZEL SMALLEST (DEFAULT).
- Choose TOOLS, PORT the port on which the USB-Serial Adapter is connected.
- Choose TOOLS, PROGRAMMER: AVRISP MKIII.
- Press upload.

- After uploading, replace the Boot0 jumper to pin44 and GND.

Uploading a sketch by using an ST-Link V2 adapter.

- Connect GND of the ST-Link V2 adapter to GND.
- Connect 3.3V of the ST-Link V2 adapter to 3.3V.
- Connect SWCLK of the ST-Link V2 adapter to SWCLK.
- Connect SWDIO of the ST-Link V2 adapter to SWDIO.
- Place the Boot0 jumper on pin44 and 3.3V.
- Press the reset button.
- Choose TOOLS, BOARD, GENERIC STM32F103C.
- Choose TOOLS. VARIANT: STM32F103C8 (20k RAM, 64k Flash).
- Choose TOOLS, CPU SPEED(MHZ): 72MHZ (NORMAL).
- Choose TOOLS, UPLOAD METHOD: ST-LINK.
- Choose TOOLS, OPTIMIZEL SMALLEST (DEFAULT).
- Since the serial port is not used, you don't have to bother about its setting.
- Choose TOOLS, PROGRAMMER: AVRISP MKIII.
- Press upload.
- After uploading, replace the Boot0 jumper to pin44 and GND.

252.7. Sample code SW4STM32 for the STM32F103C8T6 Minimum Development Board

```
#include "stm32f1xx.h"

#define ONBOARDLED_Pin GPIO_PIN_13
#define ONBOARDLED_GPIO_Port GPIOC


static void GPIO_Init(void)
{
     GPIO_InitTypeDef GPIO_InitStruct = {0};

     /* GPIO Ports Clock Enable */
     __HAL_RCC_GPIOC_CLK_ENABLE();
     /*            ^            */

     /*Configure GPIO pin */
     HAL_GPIO_WritePin(ONBOARDLED_GPIO_Port, ONBOARDLED_Pin,
GPIO_PIN_RESET);
     GPIO_InitStruct.Pin = ONBOARDLED_Pin;
     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
     GPIO_InitStruct.Pull = GPIO_NOPULL;
     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
     HAL_GPIO_Init(ONBOARDLED_GPIO_Port, &GPIO_InitStruct);
}

void SystemClock_Config(void)
{
     RCC_ClkInitTypeDef clkinitstruct = {0};
     RCC_OscInitTypeDef oscinitstruct = {0};

     oscinitstruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
     oscinitstruct.HSEState = RCC_HSE_ON;
     oscinitstruct.HSIState = RCC_HSI_OFF;
     oscinitstruct.PLL.PLLState = RCC_PLL_ON;
     oscinitstruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
     oscinitstruct.PLL.PLLMUL = RCC_PLL_MUL9;
     if (HAL_RCC_OscConfig(&oscinitstruct)!= HAL_OK)
     {
          while(1);
     }
     clkinitstruct.ClockType = (RCC_CLOCKTYPE_SYSCLK |
               RCC_CLOCKTYPE_HCLK |
               RCC_CLOCKTYPE_PCLK1 |
               RCC_CLOCKTYPE_PCLK2);
     clkinitstruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
     clkinitstruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
     clkinitstruct.APB2CLKDivider = RCC_HCLK_DIV1;
     clkinitstruct.APB1CLKDivider = RCC_HCLK_DIV2;
     if (HAL_RCC_ClockConfig(&clkinitstruct, FLASH_LATENCY_2)!= HAL_OK)
     {
          while(1);
     }
}

int main(void)
{
     HAL_Init();
     SystemClock_Config();
     GPIO_Init();
     while (1)
```

```
    {
        HAL_GPIO_TogglePin(ONBOARDLED_GPIO_Port, ONBOARDLED_Pin);
        HAL_Delay(500);
    }
}
```

253. STM32F030F4P6 Minimum Development Board

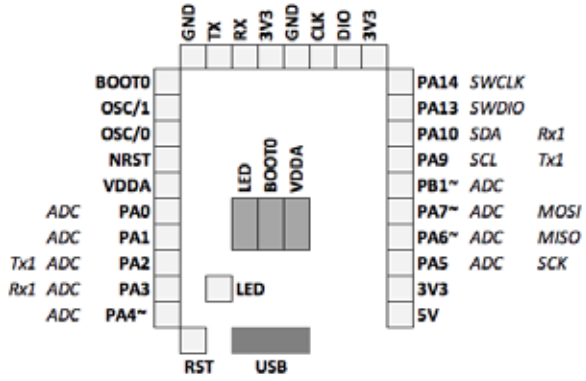

253.1. Specifications STM32F030F4P6 Minimum Development Board

- 32 bit MCU up to 48 MHz, ARM Cortex-M0
- 4 KB SRAM
- 16 KB Flash
- 3.3V or 5V power
- 15 GPIO pins
  - Data level: 3.3V (some of them are 5V tolerant)
  - 4 x PWM
  - 9 x ADC
- 1x I2C
- 1x SPI
- 1x USART
- RTC
- User LED: PA4 (GPIOA, GPIO_PIN4)

253.2. Datasheet STM32F030F4P6 Minimum Development Board

- The datasheets can be downloaded at: http://www.farnell.com/datasheets/2554494.pdf?_ga=2.266070799.1805474593.1540929874-1372663092.1536083465&_gac=1.192797016.1540929874.Cj0KCQjwguDeBRDCARIsAGxuU8bfOBFQ5z3zjhQRocfQT94wmIkqDG41Lnddd9Z5xJRhaSPqN4a1OrcaAr8NEALw_wcB

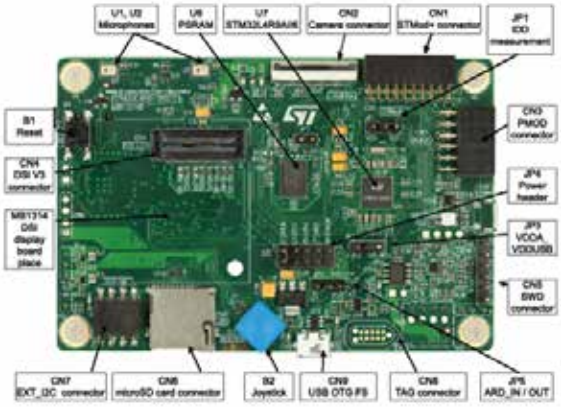
253.3. Connections STM32F030F4P6 Minimum Development Board

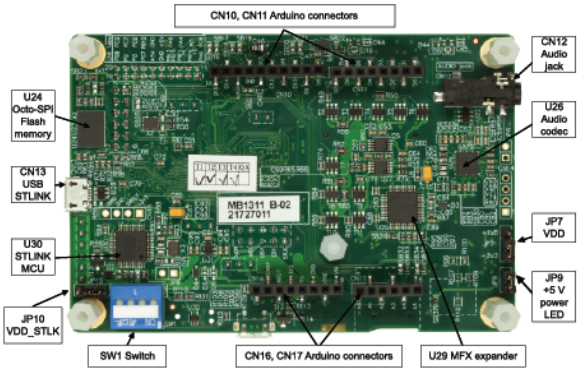


253.4. No support with Arduino IDE

Too bad, I didn't find information to use Arduino IDE to program this STM32F0304 module. One of the main reasons is that this module only has 16 KB of Flash and 4 KB of SRAM. I've read about some experiments, but even the blink sketch compiled into 12 KB, making this module useless with the Arduino IDE. To program this module, you'll need specialized software for STM32 like Keil, IAR or SW4STM from AC6. This is beyond the scope of this document.

254. STM32L4R9I-Discovery kit





254.1. Specifications STM32L4R9I-Discovery kit

- STM32L4R9AI MCU features:
  - STM32L4R9AII6 MCU, 120 MHz/150 DMIPS ARM Cortex-M4 core and 2D graphics accelerator
  - Flash: 2 Mbytes
  - SRAM: 640 Kbytes
- Power:
  - Battery or USB
  - On-board current measurement
- 512-Mbit Octo SPI Flash memory interface
- PSRAM 16-Mbit
- Display:
  - 390x390, 24bpp AMOLED
  - Capacitive Touch sensing
  - MIPI-DSI interface
- Audio:
  - DAC with integrated Class D speaker driver (SAI audio codec)
  - ST-MEMS digital microphones
  - Stereo headset jack including analog microphone input
- 2 user LEDs
- Reset button
- 4-direction joystick with selection button
- Connectors:
  - Arduino UNO shield connector
  - 8-bit camera
  - Micro SD slot
  - Micro USB OTG 2.0 full speed
  - EXT-I2C
- Embedded ST-LINK/V2-1 debugger/programmer

254.2. Datasheet STM32L4R9I-Discovery kit

The following link leads to the User manual:

- https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/ae/a3/af/94/27/d8/46/3f/DM00421316/files/DM00421316.pdf/jcr:content/translations/en.DM00421316.pdf

254.3. User LED's

Name	MCU port control	Color
LD1 (LED1)	MFX_GPIO0, PB0	Orange
LD2 (LED2)	PH4	Green
LD3	PB13	Green (Arduino D13)

254.4. Status LED's

Name	Function	Description	Color
LD4	ST-LINK COM	Green during communication	Red/green
LD5	ST-LINK USB FAULT	Current > 625 mA	Red
LD6	VBUS USB OTG full speed		Green
LD7	USB OTG full speed overcurrent	Overcurrent USB OTG	Red
LD8	5 V Power		Green

254.5. Buttons STM32L4R9I-Discovery kit

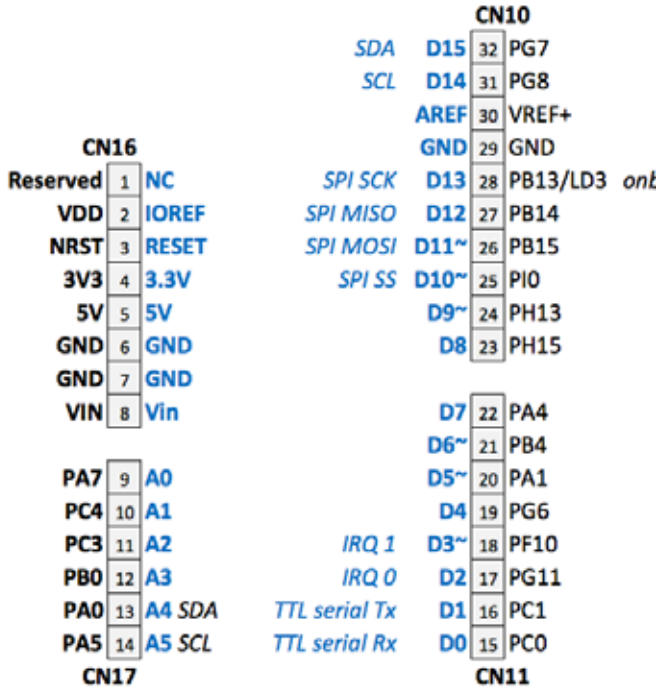
Name	Function	Description
B1	Reset	
B2	Joystick and select button	

254.6. Connectors STM32L4R9I-Discovery kit

Name	Function	Description
CN1	STMod+ male	Connector for the fan-out expansion board
CN2	Camera connector	
CN3	PMOD	
CN4	DSI V3	
CN5	SWD	
CN6	microSD	
CN¶	EXT_I2C	
CN8	TAG	
CN9	USB OTG full speed	
CN10..CN11	Arduino shield connectors	To place Arduino shields
CN12	Audio jack	
CN13	USB STLINK	
CN14..CN15	*not available*	-
CN16..CN17	Arduino shield connectors	To place Arduino shields

Arduino Shield

The following diagram show the relation between the Arduino GPIO's (in blue) and the STM32.



254.7. Jumpers STM32L4R9I-Discovery kit

Name	Function	Description	Default
JP1	IDD measurement	Current measurement internal (software) or external (ammeter)	2-1
JP2	*not available*	-	-
JP3	VDDA VDDUSB		1-2
JP4	Power header	Power source selector	(1)STLK
JP5	ARD_IN/OUT	5V power to or from Arduino CN16	2-1
JP6	*not available*	-	-
JP7	VDD	VDD powered from 1.8 or 3.3V regulator	3.3V
JP8	*not available*	-	-
JP9	+5V power LED	Power LED on or off	ON
JP10	VDD_STLK	Set VDD_STL power form STM32L4R9IDISCOVERY or from external through CN5	1-2

To prevent damage to your Discovery kit, please check paragraph 10.6 of the user manual when making changes to the power source you are going to use.
https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/ae/a3/af/94/27/d8/46/3f/DM00421316/files/DM00421316.pdf/jcr:content/translations/en.DM00421316.pdf

JP1 IDD measurement

With this jumper you can set how current consumption can be measured.

- Jumper on pin 2-1 (middle pin and IDD) (**default)**:
  Current can be measured by the MCU through a built-in ammeter circuit (MFX_V3) through software (60 nA to 50 mA).
- Jumper on pin 3-2 (VDD and middle pin)
  STM32RL4R9AI is powered by VDD, current measurement is not possible by the MCU.
- External ammeter on pin 3-2 (VDD and middle pin):
  Current can be measured through an external ammeter.

JP5 ARD_IN/OUT

- Jumper on 2-1 (**default**):
  STM32L4R9IDISCOVERY supplies 5V to the Arduino connector on CN16 (the Arduino shield is powered by the Discovery kit).
- Jumper on 3-2 together with JP4 on (4)ARD:
  STM32L4R9IDISCOVERY is supplied by 5 V from the Arduino connector on CN16 (the Discovery kit is powered by the Arduino shield).

254.8. Switches

Name	Function	Description
SW1	Debug/program enabler	ON: debug/program and power through CN13 (USB) OFF: no debug/program nor r power source (no debug/program)

254.9. Components STM32L4R9I-Discovery kit

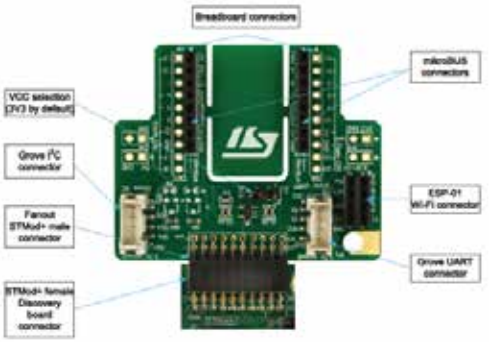
Name	Function	Description
U1	Microphones	
U2	Microphones	
U3..U5	*not available*	-
U6	PSRAM	
U7	STM32L4R9AII6	
U8..U24	*not available*	-
U24	Octo-SPI Flash memory	
U25	*not available*	-
U26	Audio codec	
U27..U28	*not available*	-
U29	MFX expander	
U30	STLINK MCU	

254.10. Sample code SW4STM32 for the STM32F103C8T6 Minimum Development Board

```
#include "stm32l4xx.h"
#include "stm32l4r9i_discovery.h"

#define ONBOARDLED_Pin GPIO_PIN_4
#define ONBOARDLED_GPIO_Port GPIOH

static void GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOH_CLK_ENABLE();
  /*            ^            */

  /*Configure GPIO pin */
  HAL_GPIO_WritePin(ONBOARDLED_GPIO_Port, ONBOARDLED_Pin, GPIO_PIN_RESET);
  GPIO_InitStruct.Pin = ONBOARDLED_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(ONBOARDLED_GPIO_Port, &GPIO_InitStruct);
}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1_BOOST);
  RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_HSI48|RCC_OSCILLATORTYPE_MSI;
  RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
  RCC_OscInitStruct.MSIState = RCC_MSI_ON;
  RCC_OscInitStruct.MSICalibrationValue = 0;
  RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 60;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV5;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  HAL_RCC_OscConfig(&RCC_OscInitStruct);
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);
  PeriphClkInit.PeriphClockSelection =
RCC_PERIPHCLK_USART2|RCC_PERIPHCLK_USART3
                              |RCC_PERIPHCLK_SAI1|RCC_PERIPHCLK_I2C1
                              |RCC_PERIPHCLK_USB|RCC_PERIPHCLK_SDMMC1;
  PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
  PeriphClkInit.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
  PeriphClkInit.I2c1ClockSelection = RCC_I2C1CLKSOURCE_PCLK1;
```

```
  PeriphClkInit.Sai1ClockSelection = RCC_SAI1CLKSOURCE_PLLSAI1;
  PeriphClkInit.UsbClockSelection = RCC_USBCLKSOURCE_HSI48;
  PeriphClkInit.Sdmmc1ClockSelection = RCC_SDMMC1CLKSOURCE_PLLP;
  PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC_PLLSOURCE_MSI;
  PeriphClkInit.PLLSAI1.PLLSAI1M = 1;
  PeriphClkInit.PLLSAI1.PLLSAI1N = 16;
  PeriphClkInit.PLLSAI1.PLLSAI1P = RCC_PLLP_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC_PLLQ_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1R = RCC_PLLR_DIV2;
  PeriphClkInit.PLLSAI1.PLLSAI1ClockOut = RCC_PLLSAI1_SAI1CLK;
  HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit);
}

int main(void)
{
 HAL_Init();
 SystemClock_Config();

 GPIO_Init();

 while (1)
 {
      HAL_GPIO_TogglePin(ONBOARDLED_GPIO_Port, ONBOARDLED_Pin);
      HAL_Delay(500);
 }
}
```

255. STM32L4R9I-Discovery kit fan-out expansion board



255.1. Specifications STM32L4R9I-Discovery kit fan-out expansion board

- Connectors:
  - STMod+ connector to Discovery board
  - ESP8266-ESP01
  - Grove UART
  - Grove I2C
  - mikroBUS
  - Breadboard friendly connector

255.2. Connectors STM32L4R9I-Discovery kit

Name	Function	Description
CN1	STMod+ female	Connector to connect the fan-out expansion board to the STM32L4R9I-Discovery kit
CN2	Seed Studio Grove: UART	
CN3	Seed Studio Grove: I2C	
CN4	ESP8266-ESP01	
CN5	Pin header 3.3V/GND	
CN6	Pin header 5V/GND	
CN7	Pin header 3.3V/GND	
CN8	Pin header 5V/GND	
CN9	Pin header for breadboard directly connected with STMOD+	
CN10..CN11	Microelectronic Click board	
CN12	Pin header for breadboard directly connected with STMOD+	

255.3. Jumpers STM32L4R9I-Discovery kit

Name	Function	Description
JP1	VCC selection	

Raspberry Pi

This section describes the Raspberry Pi. Using GPIO, interfacing with Arduino and running the Arduino IDE on it.

256. Raspberry Pi

Raspberry Pi is a mini computer with Arduino like input/output ports called GPIO. It has much more processor power than the Arduino, but the GPIO's are more delicate. It runs on Linux , so your I/O programs must be interpreted by the Operating System. Most of the I/O programs are written in Python.

The Raspberry Pi needs an operating system on an SD or Micro-SD card. Several Linux distributions are available for the Raspberry Pi. Microsoft has developed a special Internet of Things version of Windows 10 for the B2 model. This is not a desktop like OS, but mainly for handling sensors and actuators.

The Linux distribution used in this documents is Raspbian, as is recommended for normal use.
This chapter is a simplified introduction into the world of Pi.

256.1. Specifications Raspberry Pi Models

At this moment, there are several different models (for now I've omitted the Pi Zero and the Pi Compute Module). Not all of the models are clearly labeled. You can follow the instructions on the following link to determine your model.
http://www.raspberrypi-spy.co.uk/2012/09/checking-your-raspberry-pi-board-version/

	Pi 1 Model A	**Pi 1 Model B**	**Pi 1 Model A+**	**Pi 1 Model B+**	**Pi 2 Model B**	**Pi 3 Model B**
SoC	*Broadcom BCM2835 (CPU + GPU)*				*Broadcom BCM2836*	*Broadcom BCM2837*
CPU	*700 MHz ARM11*				*900 MHz quad-core ARM Cortex-A7*	*1.2 GHz quad-core Cortex-A53*
GPU	*Broadcom VideoCore IV, OpenGL, ES 2.0, OpenVG 1080p30 high-profile encode/decode*					
SDRAM[1]	*256MB*	*512MB*	*256MB*	*512MB*	*1024MB*	
USB 2.0	*1*	*2*	*1*	*4*		
Video outputs	*Composite RCA or HDMI*			*HDMI I (composite video requires a 4 pole 3.5 mm jack)*		
Audio	*4 pole 3.5 mm mini-jack (shared with composite video)*					
NIC	*none*	*10/100 Ethernet*	*none*	*10/100 Ethernet*		*10/100 Ethernet 802.11n WLAN*
Bluetooth						*Bluetooth 4.0 & Bluetooth Low Energy (BLE)*
Power Rating	*300mA*	*700 mA*	*600 mA-1.2A*	*650 mA*		*?*

[1] Earlier models (2012) had less memory. Model A was planned with 128 MB, but was upgraded to 256 MB on 29 Feb. 2012. Model B was upgraded from 256 MB to 512 MB on 15 Oct. 2012.

Power Source	*5 V DC via USB Micro type B or GPIO header*	
GPIO	*26 GPIO* *SPI, I2C, I2S* *UART*	*40 GPIO* *SPI, I2C, I2S, I2C IDC pins* *UART*
Storage	*SD-slot*	*Micro-SD slot*

256.2. Datasheet Raspberry Pi

- Official documentation by the Raspberry Pi Foundation: https://www.raspberrypi.org/documentation/
- http://elinux.org/RPi_Hardware
- http://elinux.org/RPi_Hardware_Basic_Setup.

256.3. Installing Raspbian OS

This chapter does not describe how to install an operating system on the Raspberry Pi. If you are new to Raspberry Pi, you can take a look at the following link to get started: https://www.raspberrypi.org/documentation/

- Format an 4 GB or larger (micro-) SD card as FAT.
- Download NOOBS from https://www.raspberrypi.org/downloads/noobs/ and extract the ZIP-file.
- Copy the extracted files onto the root of the SD card.
- Place SD card in RPI.
- Boot RPI from SD card, this will
  - create 2 partitions
  - start the NOOBs installer.
- On the bottom of the screen, select the correct language and keyboard.
- Select RASPBIAN and click on INSTALL.

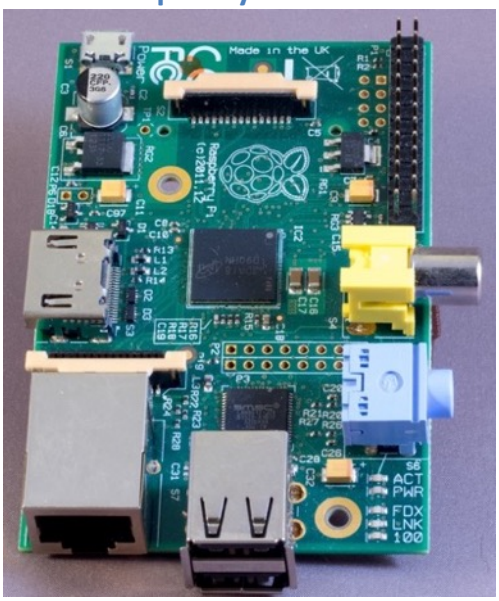
```
$ sudo raspi-config
```

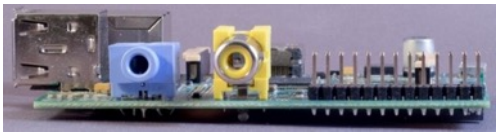
```
1 Expand Filesystem (if not NOOBS)
2 Change user (pi) password
3 Boot options
    B3 Desktop Gui requiring user to login
5 Internationalisation Options
    Change these settings to match yours
6 Enable camera (optional)
9 Advanced options
    A2 Hostname
    A4 SSH (on)
    A6 SPI (on)
    A7 I2C (on)
```

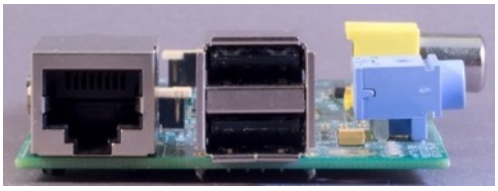
```
$ sudo shutdown -r now
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
$ sudo apt-get install wiringpi
$ sudo apt-get autoremove
$ sudo apt-get clean
```

WiringPi is probably already up to date.

256.4. Raspberry Pi 1 Model B



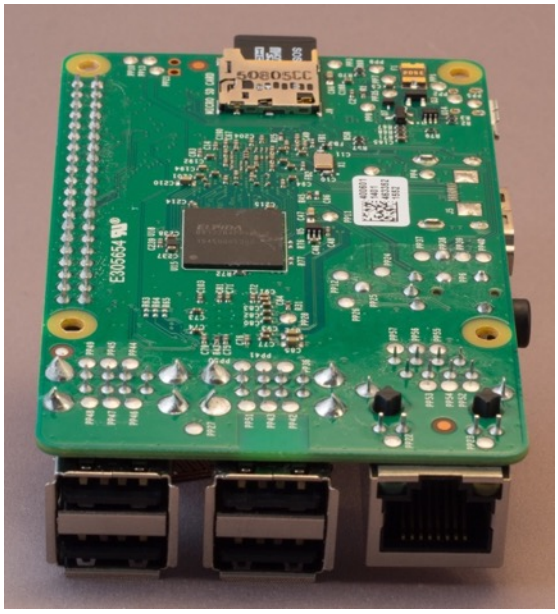


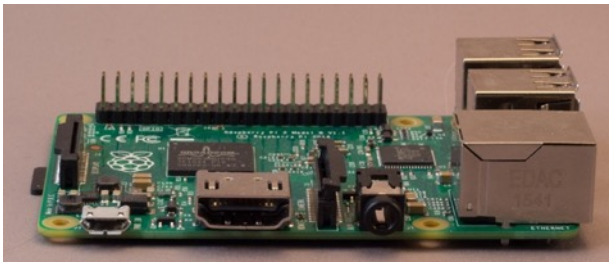


Specifications Raspberry Pi 1 Model B

- CPU: Broadcom BCM2835 (CPU + GPU), 700 MHz
- RAM: 512MB
- Storage: SD card slot
- Micro-USB power in (700 mA)
- 2x USB 2.0
- HDMI out
- Composite RCA Video out
- Mini Jack Audio out (4th pin is shared with Composite RCA Video out)
- 10/100 MB Ethernet
- 26 pin GPIO header
- CSI (Camera Serial Interface) connector
- DSI (Display Serial Interface) Connector

256.5. Raspberry Pi 2 Model B

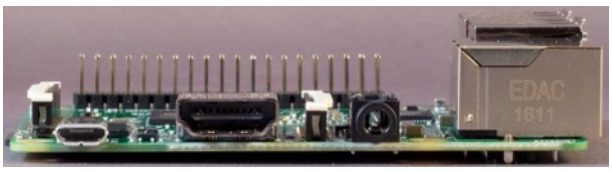




Specifications Raspberry Pi 2 Model B

- CPU: Broadcom BCM2836 quad-core ARM Cortex-A7 processor @ 900MHz
- RAM: 1024MB SDRAM
- Storage: Micro-SD card slot
- Micro-USB power in (650 mA)
- 4x USB 2.0
- HDMI out
- Mini Jack Audio out (4th pin is for Composite RCA Video out)
- 10/100 MB Ethernet
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector
- DSI (Display Serial Interface) Connector

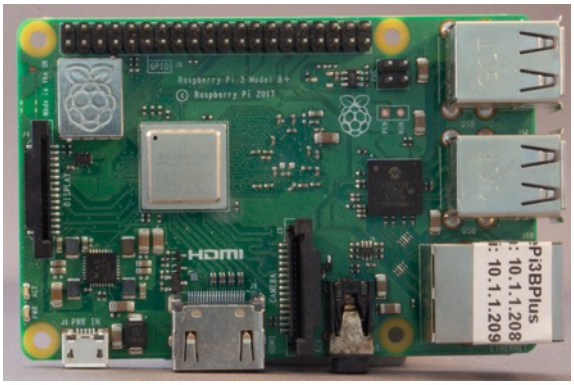
256.6. Raspberry Pi 3 Model B



Specifications Raspberry Pi 3 Model B

- CPU: Quad Core Broadcom BCM2837 64bit @ 1.2GHz
- RAM: 1024MB
- Storage: Micro-SD card slot
- Micro-USB power in
- 4x USB 2.0
- HDMI out
- Mini Jack Audio out (4th pin is for Composite RCA Video out)
- 10/100 MB Ethernet
- 802.11b/g/n WLAN
- Bluetooth 4.0 & BLE
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector
- DSI (Display Serial Interface) Connector

257. Raspberry Pi 3 Model B+



Specifications Raspberry Pi 3 Model B+

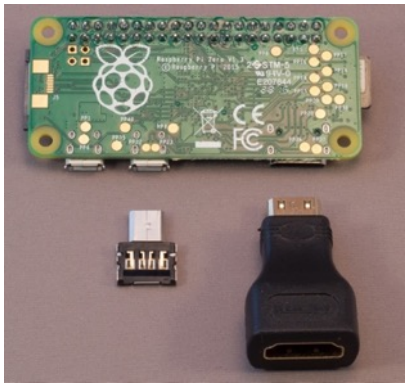
- CPU: Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
- GPU: Broadcom Videocore-IV
- RAM: 1GB LPDDR2 SDRAM
- Storage: Micro-SD card slot
- Micro-USB power in
- 4x USB 2.0
- HDMI out
- Mini Jack Audio out (4th pin is for Composite RCA Video out)
- 10/100/1000 MB Ethernet (Gigabit via USB channel)
- 802.11b/g/n/ac WLAN
- Bluetooth 4.2 & BLE
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector
- DSI (Display Serial Interface) Connector

258. Raspberry Pi 4, 4GB

Specifications Raspberry Pi 4, 4GB

- CPU: Specifications Raspberry Pi 3 Model B+
- CPU: Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- RAM: 4GB LPDDR4--2400 SDRAM
- Storage: Micro-SD card slot
- USB-C power in
- 2x USB 2.0
- 2x USB 3.0
- 2x micro-HDMI out (up to 4kp60)
- Mini Jack Audio out (4th pin is for Composite RCA Video out)
- 10/100/1000 MB Ethernet (Gigabit via USB channel)
- 802.11b/g/n/ac WLAN
- Bluetooth 5.0 & BLE
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector
- DSI (Display Serial Interface) Connector

258.1. Raspberry Pi Zero



Specifications Raspberry Pi Zero

- CPU: Broadcom BCM2835 @ 1GHz
- RAM: 512MB LPDDR2 SDRAM
- Storage: Micro-SD card slot
- Micro-USB power in
- 1x micro USB 2.0 OTG
- Mini HDMI out
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector

258.2. Raspberry Pi Zero W

Specifications Raspberry Pi Zero

- CPU: Broadcom BCM2835 @ 1GHz
- RAM: 512MB LPDDR2 SDRAM
- Storage: Micro-SD card slot
- Micro-USB power in
- 1x micro USB 2.0 OTG
- Mini HDMI out
- 802.11b/g/n WLAN
- Bluetooth 4.1 & BLE
- 40 pin GPIO header
- CSI (Camera Serial Interface) connector

259. Raspberry Pi GPIO

The Raspberry Pi is a mini computer with GPIO ports to communicate with sensors actuators, computers and microcontrollers. It has much more processing capabilities compared to the Arduino, but lacks protection on the GPIO ports. It is not recommended to connect your sensors and actuators directly to the GPIO ports without using a shield with buffers and level shifters (to 3.3 V).

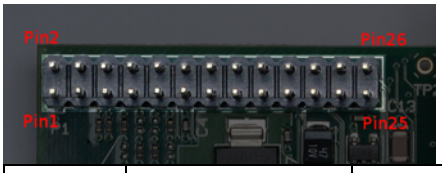
259.1. Pinout scheme according to the BCM chip

There is some confusion according to the GPIO numbers on the Pi. There are to standards. The pinout scheme according to the BCM chip (and most GPIO Adapter Expansion Board) and according to the WiringPi Python library. Both are combined in the scheme below. Off course, functions like MISO, MOSI, TxD, etc are the same in both schemes.

	WiringPi	BCM			BCM	WiringPi	
	3.3V	**3.3V**	1	2	**5V**	**5V**	
SDA1	**GPIO8**	**GPIO2**	3	4	**5V**	**5V**	
SCL1	**GPIO9**	**GPIO3**	5	6	**GND**	**GND**	
	GPIO7	**GPIO4**	7	8	**GPIO14**	**GPIO15**	*TxD0*
	GND	**GND**	9	10	**GPIO15**	**GPIO16**	*RxD0*
	GPIO0	**GPIO17**	11	12	**GPIO18**	**GPIO1**	
	GPIO2	**GPIO27**	13	14	**GND**	**GND**	
	GPIO3	**GPIO22**	15	16	**GPIO23**	**GPIO4**	
	3.3V	**3.3V**	17	18	**GPIO24**	**GPIO5**	
MOSI	**GPIO12**	**GPIO10**	19	20	**GND**	**GND**	
MISO	**GPIO13**	**GPIO9**	21	22	**GPIO25**	**GPIO6**	
SCLK	**GPIO14**	**GPIO11**	23	24	**GPIO8**	**GPIO10**	*CE0#*
	GND	**GND**	25	26	**GPIO7**	**GPIO11**	*CE1#*
		Raspberry Pi B2 and up					
ID_SD	-	**GPIO0**	27	28	**GPIO1**	-	*ID_SC*
	GPIO21	**GPIO5**	29	30	**GND**	**GND**	
	GPIO22	**GPIO6**	31	32	**GPIO12**	**GPIO26**	
	GPIO23	**GPIO13**	33	34	**GND**	**GND**	
MISO	**GPIO24**	**GPIO19**	35	36	**GPIO16**	**GPIO27**	*CE2#*
	GPIO25	**GPIO26**	37	38	**GPIO20**	**GPIO28**	*MOSI*
	GND	**GND**	39	40	**GPIO21**	**GPIO29**	*SCLK*

http://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins/

259.2. Connections Raspberry Pi GPIO Model A and B Original



Pin nr	Name BCM	Description	*WiringPi*
1	+3V3	3.3 Volt	*3.3 Volt*
3	GPIO2/SDA1	GPIO2 or I2C SDA	*GPIO8*
5	GPIO3/SCL1	GPIO3 or I2C SCL	*GPIO9*
7	GPIO4	GPIO4	*GPIO7*
9	GND	Ground	*Ground*
11	GPIO17	GPIO17	*GPIO0*
13	GPIO27	GPIO27	*GPIO2*
15	GPIO22	GPIO22	*GPIO3*
17	+3V3	3.3 Volt	*3.3 Volt*
19	GPIO10/MOSI	GPIO10 or SPI MOSI	*SPI MOSI or GPIO12*
21	GPIO9/MISO	GPIO9 or SPI MISO	*SPI MISO or GPIO13*
23	GPIO11/SCLK	GPIO11 or SPI SCLK	*SPI SCLK or GPIO14*
25	GND	Ground	*Ground*
2	+5V	5 Volt	*5 Volt*
4	+5V	5 Volt	*5 Volt*
6	GND	Ground	*Ground*
8	GPIO14/TxD0	GPIO14 or Serial Transmit UART	*GPIO15*
10	GPIO15/RxD0	GPIO15 or Serial Receive UART	*GPIO16*
12	GPIO18	GPIO18	*GPIO1*
14	GND	Ground	*Ground*
16	GPIO23	GPIO23	*GPIO4*
18	GPIO24	GPIO24	*GPIO5*
20	GND	Ground	*Ground*
22	GPIO25	GPIO25	*GPIO6*
24	GPIO8/CE0#	GPIO8 or CE0#	*GPIO10*
26	GPIO7/CE1#	GPIO7 or CE1#	*GPIO11*

A detailed description of these connections can be found at: http://elinux.org/RPi_Low-level_peripherals#P2_header.

GPIO voltage levels are 3.3 V and are not 5 V tolerant. There is no over-voltage protection on the board.

259.3. Raspberry Pi GPIO Model A+, B+, B2 and B3 (extra connections)

Pins 1 to 26 are the same as with the Models A and B. The following pins were added in the Models A+, B+, B2, B3.

Pin nr	Name	Description	***WiringPi***
27	GPIO0/ID_SD	GPIO0 or ID_SD	*ID_SD*
29	GPIO5	GPIO5	*GPIO21*
31	GPIO6	GPIO6	*GPIO22*
33	GPIO13	GPIO13	*GPIO23*
35	GPIO19/MISO	GPIO19 or SPI MISO	*GPIO24*
37	GPIO26	GPIO26	*GPIO25*
39	Ground	Ground	*Ground*
28	GPIO1/ID_SC	GPIO1 or ID_SC	*ID_SC*
30	Ground	Ground	*Ground*
32	GPIO12	GPIO12	*GPIO26*
34	Ground	Ground	*Ground*
36	GPIO16/CE2#	GPIO16 or CE2#	*GPIO27*
38	GPIO20/MOSI	GPIO20 or SPI MOSI	*GPIO28*
40	GPIO21/SCLK	GPIO21 or SPI SCLK	*GPIO29*

A detailed description of these connections can be found at: http://elinux.org/RPi_Low-level_peripherals#P2_header.

GPIO voltage levels are 3.3 V and are not 5 V tolerant. There is no over-voltage protection on the board.

260. Sample scripts Raspberry Pi

260.1. Hardware setup for the sample scripts

The following paragraphs describe how to program the Raspberry Pi to control 2 LED's with a switch.

- As long as the switch is open (not pressed), the first LED is ON at 95% and the second LED is OFF.
- As long as the switch is closed (pressed), the first LED is ON at 5% and the second LED blinks ON and OFF.

Sample connections

- Connect 1 LED through a 220 ohm resistor between GND and port 18.
- Connect another LED through a 220 ohm resistor between GND and port 23.
- Connect a switch between GND and port 22.

A more detailed description about this setup and about the scripts used in this chapter, can be found at: https://learn.sparkfun.com/tutorials/raspberry-gpio.

260.2. Control GPIO ports through Python

The Python scripting language is part of Raspbian, so you don't need to download anything to get started.

Brief description sample Python script

```
import RPi.GPIO as GPIO
```
Import the RPI.GPIO module included in Raspbian, so you can use GPIO functions.

```
include time
```
Include the time module so you can use time related stuff like delays.

```
GPIO.setmode(GPIO.BCM)
```
Set the GPIO pin number scheme to the Broadcom chip pin numbers.

```
GPIO.setup(18, GPIO.OUT)
```
Set GPIO18 to output.

```
pwm = GPIO.PWM(23, 50)
```
Create an instance pwm of the GPIO.pwm object. Set the frequency to 50 Hz.

```
GPIO.setup(22, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```
Set GPIO22 as input with internal pull-up resistor.

```
GPIO.output(18, GPIO.LOW)
```
Turn off the LED at GPIO18.

```
GPIO.output(18, GPIO.HIGH)
```
Turn on the LED at GPIO18

```
pwm.start(95)
```
Set the start duty cycle of pwm to 95.

```
if GPIO.input(22):
```
Check if button at GPIO22 is pressed.

```
pwm.ChangeDutyCycle(5)
```
Change duty cycle of pwm to 5.

```
time.sleep(0.075)
```
Sleep for 0.075 seconds.

```
except KeyboardInterrupt:
```
Check if CTRL-C has been pressed.

```
pwm.stop()
```
Turn of pwm.

```
GPIO.cleanup()
```
Clean up GPIO resources.

RPI_python_001_Blinker

```
import RPi.GPIO as GPIO
import time

pwmPin = 18
ledPin = 23
butPin = 22

dc = 95

GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(pwmPin, GPIO.OUT)
pwm = GPIO.PWM(pwmPin, 50)
GPIO.setup(butPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.output(ledPin, GPIO.LOW)
pwm.start(dc)

print("Here we go! Press CTRL+C to exit")
try:
    while 1:
        if GPIO.input(butPin):
            pwm.ChangeDutyCycle(dc)
            GPIO.output(ledPin, GPIO.LOW)
        else:
            pwm.ChangeDutyCycle(100-dc)
            GPIO.output(ledPin, GPIO.HIGH)
            time.sleep(0.075)
            GPIO.output(ledPin, GPIO.LOW)
            time.sleep(0.075)
except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

To run this script you need to issue the following command at a terminal:

```
$ sudo python blinker.py
```

260.3. Control GPIO ports through C (Wiring Pi)

If you don't like Python, or if you are more familiar to C-language, then you can install Wiring Pi. Wiring Pi is more than just a C library, it also includes a command-line utility.

Installation of Wiring Pi

```
$ git clone git://git.drogon.net/wiringPi
$ cd wiringPi
$ git pull origin
$ ./build
```

Test Wiring Pi

You can test Wiring Pi by issuing commands from the command line. Below you find a few examples.

The following command will set GPIO as output and turns the LED ON and then OFF.

```
$ gpio –g mode 18 output
$ gpio –g write 18 1
$ gpio –g write 18 0
```

The following commands will set GPIO as input with an internal pull-up resistor and will then read the status of the button (0 is pressed and 1 is not pressed).

```
$ gpio –g mode 22 up
$ gpio –g read 22
```

Brief description sample C script for Wiring Pi

```
#include <stdio.h>
```
Include the stdio.h library for the printf command.

```
#include <wiringPi.h>
```
Include the wiringPi.h library

```
wiringPiSetupGpio();
```
Initialize wiringPi using Broadcom pin numbers.

```
pinMode(18, PWM_OUTPUT);
```
Set GPIO18 to PWM_OUTPUT.

```
pinMode(22, OUTPUT);
```
Set GPIO23 to digital OUTPUT.

```
pinMode(butPin, INPUT);
```
Set GPIO22 to input.

```
pullUpDnControl(22, PUD_UP);
```
Set GPIO22 to low active using an internal pull-up resistor.

```
if (digitalRead(22))
```
Check if button at GPIO22 is pressed.

```
pwmWrite(18, 75);
```
Set pwm value of GPIO18 to 75. A low value is a high brightness. An high value (1024) is a low brightness.

```
digitalWrite(ledPin, LOW);
```
Turn OFF LED at GPIO23.

```
digitalWrite(ledPin, HIGH);
```
Turn ON LED at GPIO23

```
delay(75); // Wait 75ms
```
Wait 75 ms

RPI_C_001_Blinker.c

```
#include <stdio.h>
#include <wiringPi.h>

const int pwmPin = 18;
const int ledPin = 23;
const int butPin = 22;

const int pwmValue = 75;

int main(void)
{
  wiringPiSetupGpio();

  pinMode(pwmPin, PWM_OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(butPin, INPUT);
  pullUpDnControl(butPin, PUD_UP);

  printf("blinker is running! Press CTRL+C to quit.");

  // Loop (while(1)):
  while (1)
  {
    if (digitalRead(butPin))
    {
      pwmWrite(pwmPin, pwmValue);
      digitalWrite(ledPin, LOW);
    }
    else
    {
      pwmWrite(pwmPin, 1024 - pwmValue);
      digitalWrite(ledPin, HIGH);
      delay(75); // Wait 75ms
      digitalWrite(ledPin, LOW);
      delay(75); // Wait 75ms again
    }
  }

  return 0;
}
```

To run this program, you need to compile it first.

```
$ gcc –o blinker blinker.c –l wiringPi
$ sudo ./blinker
```

260.4. Control GPIO ports through an IDE (Python/C)

For most people it is more convenient to use an IDE. Geany is an example of an IDE that supports both Python as C.

```
sudo apt-get update
sudo apt-get install geany
```

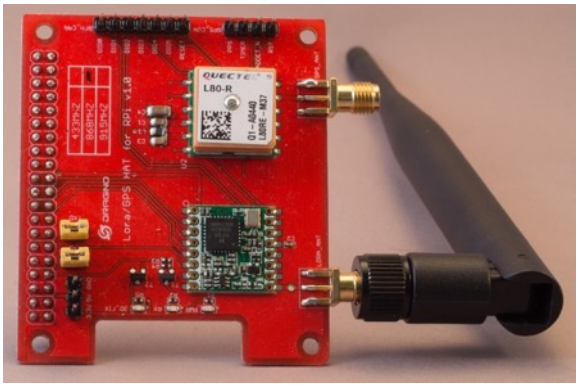
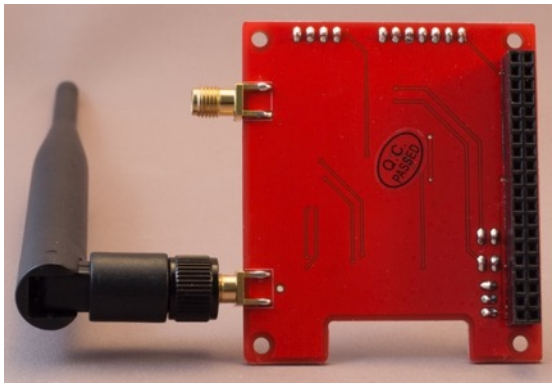
Depending on the version of Raspbian you are using, Geany is already installed and up to date.

To use Geany with C, you need to make the following changes to the build commands.

- Click on BUILD, BUILD COMMANDS
- Add –l wiringPi to the C command Compile.
- Add –l wiringPi to the C command Build as well.

Add sudo to the Execute commands.

261. HAT: Dragino Lora/GPS

This HAT, adds a Lora and GPS module to your RPI so you can use your RPI for Internet Of Things, either as a Single Channel Gateway or as a Node. Check the section on Internet Of Things for more information.

261.1. Specifications

- Lora module: RFM95W
  - Headers for DIO0 .. DIO5 & Reset
  - Antenna
- GPS module: L80RE-M37
  - Headers for: PPS, Timer, ADDET_N & RST
  - Antenna

261.2. Connections Dragino Lora/GPS HAT

Pin nr	Name	Description	Raspberry Pi pin nr	Wiring Pi name	BCM pin name
5	DIO0	Digital I/O software configured	7	GPIO7	GPIO4
6	MISO	SPI Data output	21	MISO	
7	MOSI	SPI Data input	19	MOSI	
8	SCK	SPI Clock input	23	CLK	
9	nSS	SPI Chip Select input	22	GPIO6	GPIO25
10	RST	Reset trigger input	11	GPIO0	GPIO17
11	3.3V	Power Supply	1	3.3V	
14	GND	Ground	6	GND	

261.3. Lora Sample

A description for using the Lora module on this HAT for Internet of Things can be found at:

261.4. GPS Sample Raspberry Pi 2 model B

This paragraph describes a sample on how to use the GPS on a Raspberry Pi 2 Model B. The procedure below is a simplified version of the procedure you can find at the following link. I found out, that it was not necessary to perform all steps.

http://wiki.dragino.com/index.php?title=Getting_GPS_to_work_on_Raspberry_Pi_2_Model_B

- Check the raw data output of your GPS module.

```
sudo cat /dev/ttyAMA0
```

```
$GPGSA,A,3,01,08,32,22,10,11,03,28,14,17,,,1.61,0.99,1.28*0C

$GPRMC,111830.000,A,5156.4208,N,00554.1896,E,0.20,170.62,061116
,,,A*6E

$GPZDA,111830.000,06,11,2016,,*5F
```

- Press CTRL-C to stop the raw data output.
- Install and configure the GPS daemon called gpsd, its clients and the gps python library.

```
sudo apt-get install gpsd gpsd-clients python-gps
sudo nano /etc/default/gpsd
```

```
START_DAEMON="true”
GPSD_OPTIONS="-n"
DEVICES="/dev/ttyS0"
USBAUTO="false"
GPSD_SOCKET="/var/run/gpsd.sock"
```

- Test the GPS module with a python script, showing the current coordinates.

```
nano gpstest.py
```

RPI_python_002_GPS.py

```
import os
from gps import *
from time import *
import time
import threading

gpsd = None #seting the global variable

class GpsPoller(threading.Thread):
  def __init__(self):
    threading.Thread.__init__(self)
    global gpsd #bring it in scope
    gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
    self.current_value = None
    self.running = True #setting the thread running to true

  def run(self):
    global gpsd
    while gpsp.running:
      gpsd.next() #this will continue to loop and grab EACH set
of gpsd info to clear the buffer

if __name__ == '__main__':
  gpsp = GpsPoller() # create the thread
  try:
    gpsp.start() # start it up
    while True:

      print gpsd.fix.latitude, ",", gpsd.fix.longitude

      time.sleep(5) #set to whatever

  except (KeyboardInterrupt, SystemExit): #when you press
ctrl+c
    print "\nKilling Thread..."
    gpsp.running = False
    gpsp.join() # wait for the thread to finish what it's doing
  print "Done.\nExiting."
```

```
Python gpstest.py
```

```
0.0 , 0.0
51.940893333 , 5.903438333
51.940871667 , 5.903433333
51.940405 , 5.903213333
```

More about gpsd can be found at:

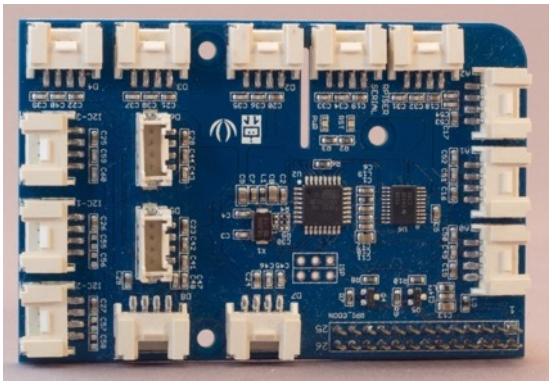
- http://www.catb.org/gpsd/
- http://www.catb.org/gpsd/client-howto.html

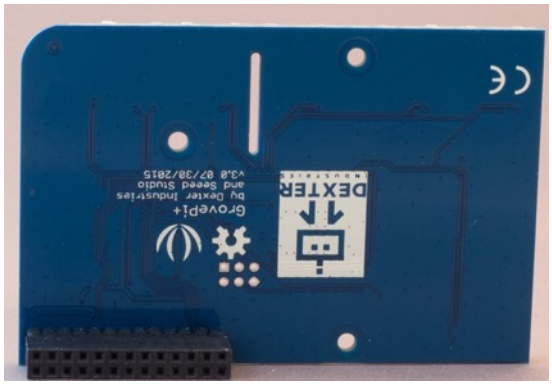
261.5. GPS Sample Raspberry Pi 2 model B

Unfortunately the above procedure does not work for the Pi 3, because of a conflict between the onboard Bluetooth module and the Dragino GPS. There is a workaround, but I haven't tested it yet.

http://wiki.dragino.com/index.php?title=Getting_GPS_to_work_on_Raspberry_Pi_3_Model_B

262. HAT: GrovePi+





This HAT was developed by Seeed studio to connect Grove modules to a Raspberry Pi (see Grove section). It contains an ATMEGA328, the same MCU that is used in some Arduino boards. The Grove modules communicate with the ATMEGA328. The Raspberry Pi does not communicate with the Grove modules, but sends commands to the ATMEGA328.

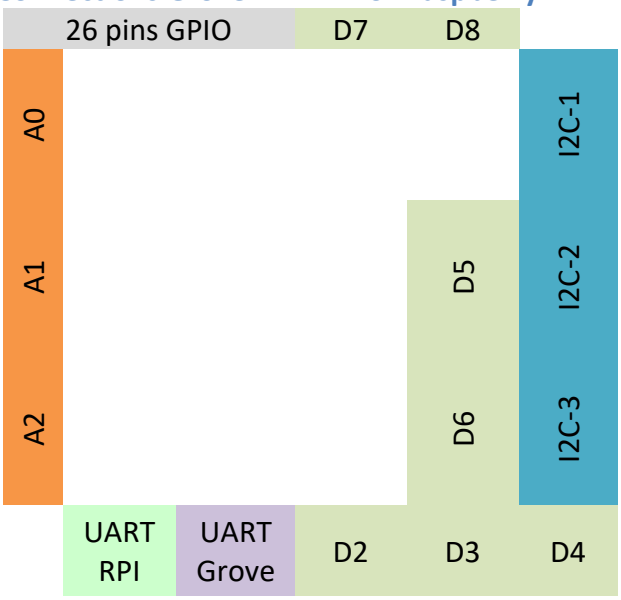
262.1. Specifications GrovePi+ HAT for Raspberry Pi

- ATMEGA328 microcontroller
- 1 Serial port connected to GrovePi
- 1 Serial port connected to Raspberry Pi
- Grove header VCC output voltage: 5V
- 7 Digital Port (6 x PWM)
- 3 Analog Ports
- 3 I2C connectors (SCL and SDA)
- The first 26 connectors of the Raspberry Pi is available on the header pins
- Camera cable outlet hole.

262.2. Datasheet GrovePi+ HAT for Raspberry Pi

http://wiki.seeedstudio.com/wiki/GrovePi+

Connections GrovePi+ HAT for Raspberry Pi



262.3. Libraries needed for GrovePi+ HAT for Raspberry Pi

The libraries and sample scripts are available on GitHub. Following the instructions below to install the GrovePi software.

- Turn off your RPI and place the GrovePi+ HAT on top of your RPI
- Clone files from github to ~/Desktop/GrovePi

```
cd ~/Desktop
sudo git clone https://github.com/DexterInd/GrovePi
```

- Run the install script

```
cd ~/Desktop/GrovePi/Script
sudo chmod +x install.sh
sudo ./install.sh
```

- Cleanup

```
sudo apt-get autoremove
```

- Check your GrovePi + HAT by typing the following command:

```
i2cdetect –y 1[1]
```

This will show that there is an I2C device connected with ID=04.

- Install Python libraries

```
cd ~/Desktop/GrovePi/Software/Python
sudo python setup.py install
```

- Update firmware on the GrovePi's ATMEGA328

```
cd ~/Desktop/GrovePi/Firmware
sudo chmod +x firmware_update.sh
sudo ./firmware_update.sh
```

- Create folder for your GrovePi Projects

```
mkdir ~/Desktop/GrovePiProjects
```

- Make a template file for your Python scripts

```
cd ~/Desktop/GrovePiProjects
nano template.py
```

[1] On some older RPI's the GrovePi+ is connected at I2C 0, in that case you must type i2cdetect –y 0.

- Type the following code, make sure to mimic the indentation exactly. You can use either spaces or TAB's, but be consistent. Save the file as template.py.

RPI_python_003_template.py (template.py)

```
import time
from grovepi import *

while True:
    try:
        #Place at least one instruction here

    except KeyboardInterrupt:
        # Turn off LED's etc before stopping
        break

    except IOError:
        print ("Error")
```

- You must copy the content of this file for every new Python script you want to create.

263. HAT: Raspberry pi to Arduino Shields Connection Bridge v1

This HAT from Cooking Hacks (cooking-hacks.com) is a connection bridge between a Raspberry Pi (B+, 2 and 3) and Arduino shields, enabling a wide range of Arduino Shields combined with the power of the Raspberry Pi.

263.1. Specifications Raspberry pi to Arduino Shields Connection Bridge v1

- Socket for wireless modules (XBEE socket), like the Xbee 802.15.4/Xbee ZigBee, RFID, NFC, Bluetooth, Bluetooth Pro, WiFi, GPRS, 3G etc..
- 8 Digital pins (Arduino D2..D9)
- Rx/Tx pins (Arduino D0, D1)
- I2C pins (SDA/SCL) (Arduino A4, A5)
- SPI pins (SCK, MISO, MOSI, CS) can also be used as GPIO
- 8 channel analog to digital converter
- Arduino A4..A7 also available
- Switch to enable external power supply

263.2. Datasheet Raspberry pi to Arduino Shields Connection Bridge v1

- Documentation https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge/
- Schematics: https://www.cooking-hacks.com/media/cooking/images/documentation/raspberry_arduino_shield/arduino2raspberryPi_sch.pdf

263.3. Libraries needed for Raspberry pi to Arduino Shields Connection Bridge v1

arduPi is a C++ library (tested on Raspbian) that lets you write programs for Raspberry as if you were writing an Arduino program.
Besides this you also need the libraries for the shields you are using (if any).

Raspberry Pi B+

```
wget http://www.cooking-
hacks.com/media/cooking/images/documentation/raspberry_arduino_shield/raspb
errypi.zip && unzip raspberrypi.zip && cd cooking/arduPi && chmod +x
install_arduPi && ./install_arduPi && rm install_arduPi && cd ../..
```

Raspberry Pi 2 and 3

```
wget http://www.cooking-
hacks.com/media/cooking/images/documentation/raspberry_arduino_shield/raspb
```

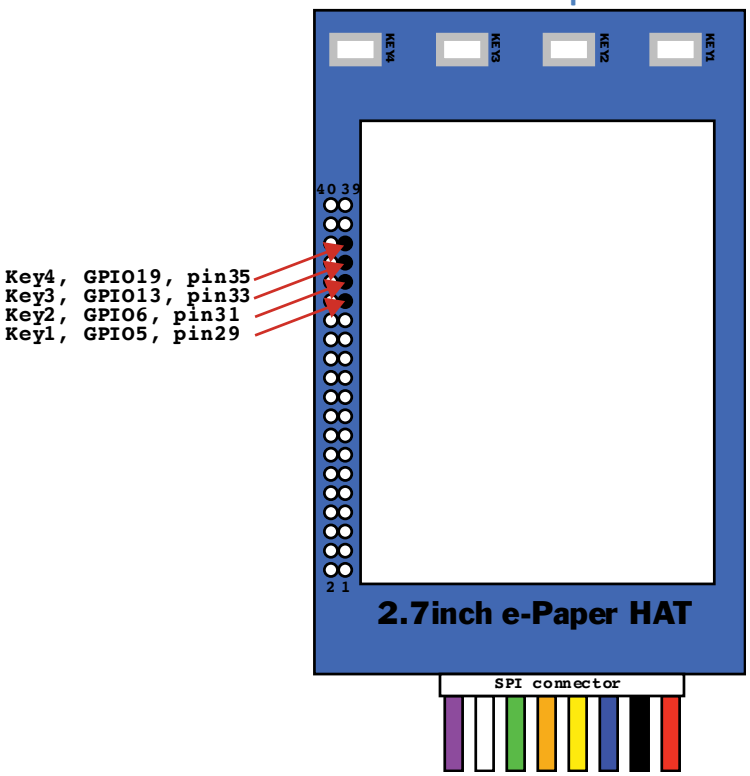
```
errypi2.zip && unzip raspberrypi2.zip && cd cooking/arduPi && chmod +x
install_arduPi && ./install_arduPi && rm install_arduPi && cd ../..
```

More information can be found at:
https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge/#step3

264. Waveshare 2.7 e-Paper HAT


Three color 2.7 inch e-paper display HAT designed for Raspberry Pi, but can also be used on other Micro computers and Micro Controllers like Arduino, ESP8266 and STM32 devices, through an 8 pin SPI connection. This chapter describes the combination of this HAT with Raspberry Pi.

264.1. Connections Waveshare 2.7 e-Paper HAT



GPIO header

Name	Description	GPIO pin	GPIO port BCM	GPIO port WiringPi
VCC	Power Supply	1	3.3V	3.3V
GND	Ground	6	GND	GND
DIN	SPI Mosi	19	GPIO10	GPIO12
CLK	SPI SCK	23	GPIO11	GPIO14
CS	SPI chip selection	24	GPIO8	GPIO10

DC	Data/Command Selection	22	GPIO25	GPIO6
RST	Reset	11	GPIO17	GPIO0
Busy	Busy	18	GPIO24	GPIO5
Key1	Key 1	pin 29	GPIO 5	GPIO 21
Key2	Key 2	pin 31	GPIO 6	GPIO 22
Key3	Key 3	pin 33	GPIO 13	GPIO 23
Key4	Key 4	pin 35	GPIO 19	GPIO 24

264.2. Specifications Waveshare 2.7 e-Paper HAT

- SKU: 13357
- Interfaces:
  - Standard Raspberry Pi 40 pin GPIO extension header
  - 8 pin SPI connector (3-wire or 4-wire SPI)
- Display:
  - 264 x 176 pixels
  - Three-color display: red, black and white, 2 gray levels
  - Dot pitch: 0.217 x 0.217
  - 57.288 x 38..192mm
  - Full refresh time: 15 seconds
  - Viewing angle: > 170
- 4 onboard keys
- Onboard voltage translator, compatible with 3.3V/5V MCU's
- Dimensions: 85 x 56mm.
- Standby power: < 0.017mW

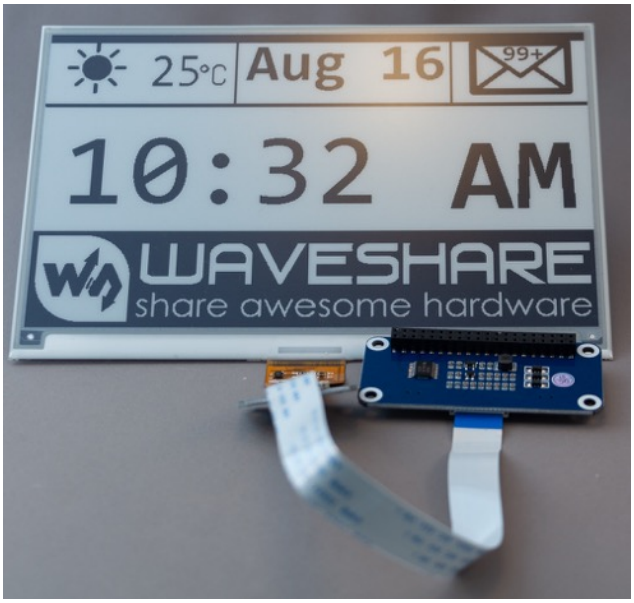
264.3. Datasheet Waveshare 2.7 e-Paper HAT

- Datasheet: https://www.waveshare.com/w/upload/d/d8/2.7inch-e-paper-b-specification.pdf
- Manual https://www.waveshare.com/w/upload/3/31/2.7inch_e-paper_hat_user_manual_en.pdf
- Wiki https://www.waveshare.com/wiki/2.7inch_e-Paper_HAT_(B)

264.4. Libraries and samples for Waveshare 2.7 e-Paper HAT

- git clone https://github.com/waveshare/e-Paper
- cd ~/e-Paper/RaspberryPi\&JetsonNano/python/examples
  - python epd_2in7b_test.py (for 2.7 inch three color)

265. Waveshare 7.5 e-Paper + HAT



Two color 7.5 inch e-paper display + HAT designed for Raspberry Pi, but can also be used on other Micro computers and Micro Controllers like Arduino, ESP8266 and STM32 devices, through an 8 pin SPI connection. This chapter describes the combination of this HAT with Raspberry Pi.

265.1. Specifications Waveshare 7.5 e-Paper + HAT

- SKU: 13504
- Interfaces:
  - Standard Raspberry Pi 40 pin GPIO extension header
  - 8 pin SPI connector (3-wire or 4-wire SPI)
- Display:
  - 640 x 384 pixels
  - Two-color display: black and white, 2 gray levels
  - Dot pitch: 0.255 x 0.255
  - 163.2 x 97.92 mm
  - Full refresh time: 6 seconds
  - Viewing angle: > 170
- Onboard voltage translator, compatible with 3.3V/5V MCU's
- Dimensions: outline: 170.2 x 111.2 mm
- Standby power: < 0.017mW

265.2. Datasheet Waveshare 7.5 e-Paper + HAT

- Datasheet
  https://www.waveshare.com/wiki/File:7.5inch-e-paper-specification.pdf
- Manual
  https://www.waveshare.com/w/upload/7/74/7.5inch-e-paper-hat-user-manual-en.pdf
- Wiki
  https://www.waveshare.com/wiki/7.5inch_e-Paper_HAT

265.3. Libraries and samples for Waveshare 7.5 e-Paper + HAT

- git clone https://github.com/waveshare/e-Paper
- cd ~/e-Paper/RaspberryPi\&JetsonNano/python/examples
- python epd_7in5_test.py (for 7.5 inch two color)

266. Add-ons for Raspberry Pi

This chapter contains a few of the many available add-ons for the Raspberry Pi.

266.1. 7" Touch Display v1.1


This display is sold in a kit with the following content:

- 7" Touchscreen Display
- Adapter Board
- DSi ribbon flex cable
- 4x stand-offs and screws
- 4x jumper wires

Specifications 7" Touch Display v1.1

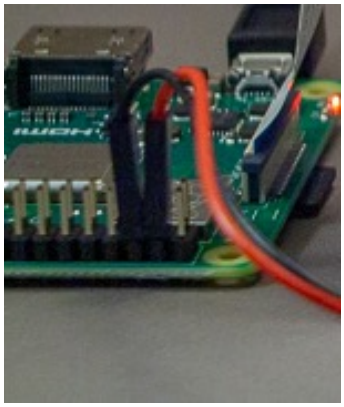
- Adapter board connected to the display (situated at the back of the display)
- Connector: 15 pin ribbon cable to DSi
- 800x480 pixels
- 10 finger capacitive touch
- 194x110x20 mm

Installation 7" Touch Display v1.1

This installation is derived from the following URL:
https://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display#installI

- Connect the ribbon cable to the adapter board.
- Screw the 4 stand-offs to the adapter board
- Connect the red jumper cable to the 5V connector on the adapter board
- Connect the black jumper cable to the GND connector on the adapter board
- Place the Raspberry Pi on the adapter board and use the 4 screws to secure it.
- Connect the other end of the ribbon cable to the Pi.
- Connect the other end of the red adapter cable to GPIO header pin 4 (5V)

- Connect the other end of the black adapter cable to GPIO header pin 6 (Ground)



- By connecting the USB power to the Pi, both the Pi and the display will be powered.

Rotation 7" Touch Display v1.1

In some cases it is necessary to rotate the display (including the touch functionality), this can be achieved by adding one of the following lines to the end of the file /boot/config.txt.

Rotation	**Line** *make sure not to add spaces to this line*
0 deg.	`lcd_rotate=0`
90 deg.	`lcd_rotate=1`
180 deg.	`lcd_rotate=2`
270 deg.	`lcd_rotate=3`

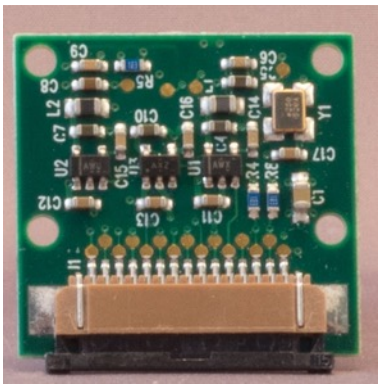
Installation onscreen keyboard

With the following command you can install an onscreen keyboard.

```
sudo apt-get install matchbox-keyboard
```

You can find this onscreen keyboard by clicking MENU/ACCESSORIES/KEYBOARD.

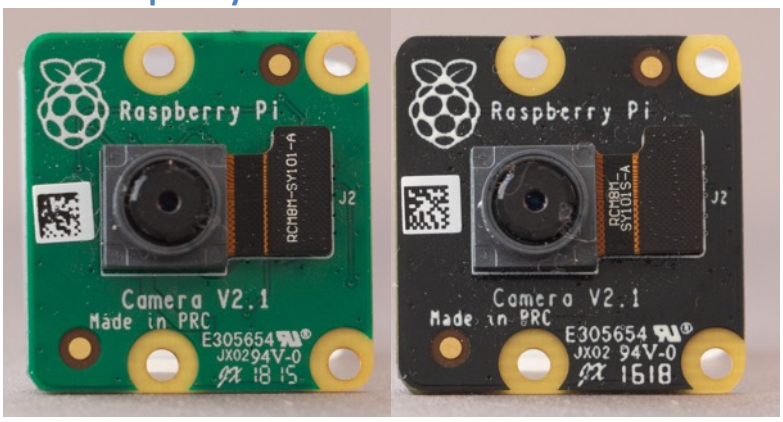
266.2. Raspberry Pi Camera Rev 1.3



Specifications

- Raspberry Pi model 1 A and up.
- Connector: 15 pin ribbon cable to MIPi CSI (Camera Serial Interface)
- Photo: 5MP (2592x1944 pixels)
- Video:
    - 1080p HD, 30 fps
    - 720p 60 fps
    - 640x480p 60 fps
    - 640x480 90 fps
- Omnivision 5647 sensor fixed focus

266.3. Raspberry Pi Camera v2.1



Specifications

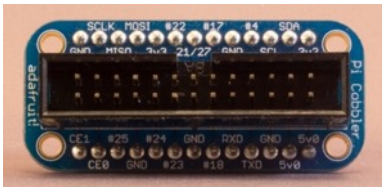
- Connector: 15 pin ribbon cable to MIPi CSI (Camera Serial Interface)
- Photo: 8MP (3280x2464 pixels)
- Video:
    - 1080p HD, 30 fps
    - 720p 60 fps
    - 640x480 90 fps
- Sony IMX219 CCD
- Fixed Focus

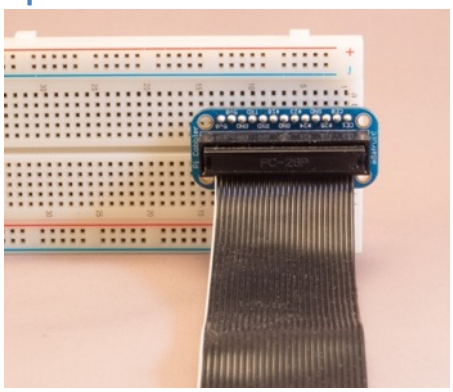
266.4. Raspberry Pi Camera NoIR v2.1

Specifications

Same specs as the Raspberry Pi Camera v2.1, but the NoIR has no infra red filter on the sensor, giving a great response in low light situations or when you want to use "invisible" IR-illumination for survaillance or other night shooting purposes.

266.5. Adafruit Pi cobbler 26 pins

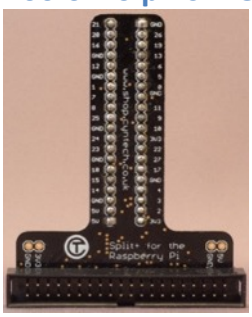




Specifications

- Raspberry Pi model 1 A and up
- 26 pins flat cable
- Pin names according to BCM

266.6. 40 pins T-Cobbler breakout board for Raspberry Pi



Specifications

- Raspberry Pi model 2 and up.
- 40 pins flat cable
- Pin names according to BCM

Mindstorms NXT

LEGO's Mindstorms NXT[1] was the predecessor of the LEGO Mindstorms EVT. It is a Microcontroller that can be connected to several proprietary LEGO sensors and motors. Since the NXT (and also the EVT) works with the I2C protocol, it is possible to combine the NXT with Arduino, bringing a whole lot of universal sensors and actuators to this beautiful platform.

[1] I'm not sure if everything in this chapter also applies to the newer EVT brick. I think so, but I'm not able to test it.

267. Mindstorms NXT-Brick

Mindstorms is a microcontroller from LEGO. It consist of a large brick witch 3 connections for motors (A..C) and 4 for sensors (1..4). The LEGO NXT software is a GUI drag & drop modular programming system, developed for easy programming your projects. You can also use other languages, some working with the original LEGO firmware, others needing a customized firmware. This chapter describes the use of the No eXactly C-language called NXC, in combination with the tool NBC. This combination is free and can be used with the original LEGO firmware.

267.1. Specifications Mindstorms NXT

- 48 MHz 32 bit Atmel ARM processor
- 256 KB Flash
- 64 KB RAM
- 4 sensor ports Analog/Digital (9600 bit/s) I2C
- 3 Motor ports with encoders
- USB 12 Mbit/s
- 4 buttons.
- LCD matrix monochrome 100x64.
- Bluetooth (not certified, so only Android and no iPhone support)
- NXT 2.0 software supports:
  - Windows
  - OSx up until 10.9.x (no support for 10.10 Yosemite and above!)

267.2. NXT sensor interface pinout

Pin nr	Name	Description	Arduino pin
1	Analog	Analog interface, +9V supply	White
2	GND	Ground	Black
3	Ground	Ground	Red
4	IPOWERA	+4.3 Vout	Green
5	DIGIAI0	I2C Clock (SCL), RS-485B	Yellow
6	DIGIAI1	I2C Data (SDA), RS-485B	Blue

268. Programming with NXC/NBC

The Graphical programming environment from LEGO is limited to LEGO components. If you want to use the full power of your NXT-brick, you'll need another programming language and another environment. There are several solutions available, some commercial others free. In this document I've described the free combination of the language NXC with the compiler NBC. Both are available for Windows and OSx. For Windows there is even a graphical IDE called Bricxcc.

268.1. NXC/NBC on Windows

Before you can write your first program for the NXT-brick on Windows, you'll need to download and install NBC and the Bricxcc IDE. The Bricxcc IDE gives you a GUI to compile and download[1] your compiled software to the NXT-brick.

Installing nbc on Windows

- Download the drivers for the NXT-brick[2]:
  - http://downloads.robotc.net/drivers/NXTUSBDriver_32bit.zip or http://downloads.robotc.net/drivers/NXTUSBDriver_64bit.zip
  - Extract this zip file and start the setup of the driver by double clicking on LegoMinstormsNXTdriverxx.msi and follow the instructions.
  - Connect your NXT-brick so the driver will be installed.
- Download and install the Bricxcc IDE (including nbc): http://bricxcc.sourceforge.net/test_releases/bricxcc_setup_33810_20130220.exe
- [3]Extract this file and start the setup file.
- The Bricxcc IDE is now installed
- To use NBC from the command prompt, you'll need to add "C:\Program Files (x86)\Bricxcc" to the search path.
  - Right click in the Explorer on your Computer.
  - Choose PROPERTIES.
  - Select ADVANCED SYSTEM SETTINGS.
  - Choose ENVIRONMENT VARIABLES.
  - Select the SYSTEM VARIABLE named PATH.
  - Click EDIT
  - Go to the end of that line, place a semi colon ";" as delimiter and then the path to nbc: C:\Program Files (x86)\Bricxcc.
  - Press OK several times to close all dialog boxes.
  - Open a command prompt and check whether nbc.exe can be started.

[1] LEGO uses the word Download (instead of Upload) for sending a file from a computer to the NXT-brick.

[2] This is not needed, if you've already installed the original NXT Software.

[3] If you only need NBC, you can download the following file: http://downloads.sourceforge.net/bricxcc/nbc-1.2.1.r4.zip

Compiling code on Windows

- Open Bricx Command Center, choose NXT as you target device and choose AUTOMATIC for your connection.
- Create a new file with the following content:

LEGONXT_C_001_HelloWorld.nxc

```
task main()
{
    TextOut(0, 0, "Hello World!");
    Wait(2000);
}
```

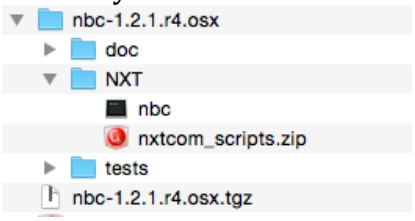
- Save this file (extension nxc).
- Click on Download  to compile this program and send it to the NXT-brick.

268.2. Programming with NXC/NBC on OSx

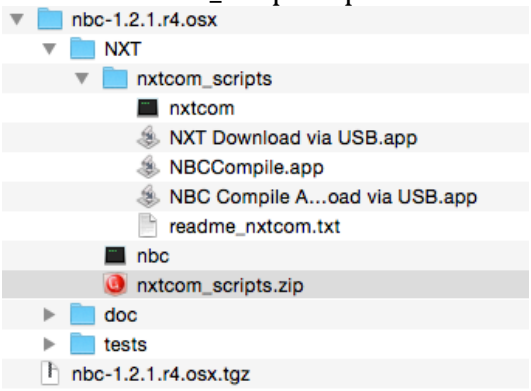
Before you can write your first program for the NXT-brick on OSx, you'll need to download and install NBC.

Installing nbc on OSx

- Download the following file: http://downloads.sourceforge.net/bricxcc/nbc-1.2.1.r4.osx.tgz
- Extract this file (by double clicking on it in the Finder). The result is placed in a directory with the same name.



- Extract nxtcom_scripts.zip as well.



- The following step is needed if you want to compile using the command prompt:
  - Copy both nxtcom and nbc to /usr/bin.
- The next three steps are needed if you want to compile by dragging and dropping your source files.
  - Create the following directory structure: /Applications/Mindstorms/nxt
  - Copy the 3 .app files from nxt_com_scripts to /Applications/Mindstorms.
  - Copy both nxtcom and nbc to /Applications/Mindstorms/nxt.

Compiling on OSx

- Use a Text editor like TextAdit.app to write your NXC code.
- Save your source code with the .nxc extension.
- If you want to use the command line for compiling use the following procedure:
  - Open a Terminal window
  - Go to the folder with your source code.
  - If there are no errors, then you can download[1] the .rxe file to your NXT-brick.
  - Type the following command to compile your source:

```
nbc <sourcename>.nxc -O=<compiledname>.rxe
```

  - If there are no errors, then you can download[2] the .rxe file to your NXT-brick with the following command:

```
nxtcom <compiledname>.rxe
```

- If you don't like the command prompt, then you can also drag and drop your .nxt source code or you .rxe on the apps you have installed in the previous paragraph.

268.3. NXC/NBC language

Next Byte Codes (NBC) is both a simple language, with an assembly language syntax and a compiler for NXC. Not eXactly C is a high level language similar to C.

Guides for these languages can be found at:

- http://bricxcc.sourceforge.net/nbc/doc/NBC_Guide.pdf
- http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf

To get you start I've listed a few simple programs below.

Motor

The following sample drives the LEGO motor connected to port A. 4 seconds forward, speed set at 75%, then 4 seconds backwards before it stops.

LEGONXT_C_002_Motor.nxc

```
task main()
{
 OnFwd(OUT_A, 75);
 Wait(4000);
 OnRev(OUT_A, 75);
 Wait(4000);
 Off(OUT_A);
}
```

[1] LEGO uses the word Download (instead of Upload) for sending a file from a computer to the NXT-brick.
[2] LEGO uses the word Download (instead of Upload) for sending a file from a computer to the NXT-brick.

Music

The following sample plays a simple melody continuously.

LEGONXT_C_003_Melody.nxc

```
task main()
{
 while (true)
 {
   PlayTone(262,400); Wait(500);
   PlayTone(294,400); Wait(500);
   PlayTone(330,400); Wait(500);
 }
}
```

269. Mindstorms and Arduino

In this chapter, I will describe how to connect an Arduino to the NXT brick, so you are free in choosing your "cheap" sensors and motors as are described in this document. This is not possible with the original NX Software, so we'll be using NXC/NBC as programming environment. The installation of this combination is described in the previous chapter.

269.1. Libraries needed for Mindstorms and Arduino

- Built-in Wire library for I2C.

269.2. Sample Mindstorms and Arduino

In this sample the NXT is sending commands to the Arduino and the Arduino responds with a string saying whether the LED is ON or OFF.

Sample Connections Mindstorms and Arduino

To connect your Arduino to the NXT-brick you need to make a special cable. On one side of this cable, you must have an NXT-Brick connector. This connector looks like a telephone connector RJ-12, with one important difference: the latch of the NXT-Brick is not in the middle, but is situated on the right hand side.

- You have several options for this proprietary connector:
  - Order NXT compatible male plugs at mindsensors.com: http://www.mindsensors.com/ev3-and-nxt/115-nxt-compatible-male-plugs-10-pack but then you also need to alter your crimping tool.
  - Order the Breadboard Connector Kit for NXT or EV3 also at mindsensors.com: http://www.mindsensors.com/51-cables-connectors.
  - Use one of your original NXT cables and order NXT compatible female sockets at mindsensors.com: http://www.mindsensors.com/ev3-and-nxt/117-nxt-compatible-female-sockets-5-pack.
  - Use one of your original NXT-cables and cut this in half. Now you can build 2 NXT-Arduino cables.
  - Crimp an normal RJ-12 connector on a length of flat telephone cable (it's important to do this first!). Remove the latch, modify it and glue it back on the connector. A detailed description can be found at: http://www.philohome.com/nxtplug/nxtplug.htm.

- Connect NXT cable to port 1 on Mindstorms NXT block
- ~~Connect VCC to 5V~~. You don't need to connect VCC
- Connect GND to GND.
- Connect Yellow (SCL) to A5
- Connect Blue (SDA) to A4
- Connect a 82 Kohm resistor between A5 and Green (4.7V)
- Connect a 82 Kohm resistor between A4 and Green (4.7V)

- You can power the Arduino through the NXT-Brick, by connecting Green (4.7V) to Vin. Make sure not to overload the NXT with more than mA current.
- Upload the Arduino sketch to your Arduino
- Download the NXT Program to your NXT-Brick

125_LEGONXT.ino

```
//A full description of this module (and many others) can be downloaded at:
https://eve_arduino
#include <Wire.h>
#include <string.h>

#define ADDRESS 0x43
char readfromnxt;
int LED = 13;
boolean LEDStatus = false;

void setup()
{
  Serial.begin(9600);
  Wire.begin(ADDRESS);
  Wire.onReceive(receiveEvent);
  Wire.onRequest(requestEvent);
  pinMode(LED, OUTPUT);
}

void loop()
{
  digitalWrite(LED, !digitalRead(LED));
  delay(3000);
}


//---------------------------------I2C Events------------------------------
-//

void receiveEvent(int howMany)
{
  if (Wire.available() > 0 ) //&& index <15)
  {
    readfromnxt = Wire.read(); // receive byte as an integer
  }
}

void requestEvent()
{
  if (readfromnxt == 0x42)
  {
    if (digitalRead(LED))
    {
      Wire.write("LED is ON ", 15);
    }
    else
    {
      Wire.write("LED is OFF", 15);
    }
  }
}
```

LEGONXT_C_004_Arduino.nxc

```
#define ARDUINO_ADDRESS 0x43
#define ADDRESS_SEND (ARDUINO_ADDRESS << 1)
#define ADDRESS_RECV (ADDRESS_SEND + 1)
#define ARDUINO_PORT  IN_1

string i2cReadString(byte port, byte adr, byte reg, byte cnt)
{
  string temp;
  byte outbuf[];
  byte cmdbuf[];
  ArrayBuild(cmdbuf, adr, reg);
  byte nByteReady = 0;
  while (I2CStatus(port, nByteReady) ==  STAT_COMM_PENDING)
  {
  }
  Wait (100);
  if (I2CBytes(port, cmdbuf, cnt, outbuf))
  {
    temp = ByteArrayToStr(outbuf);
    //TextOut(0, LCD_LINE6, temp2, false);
  }
  return temp;
}


task main()
{
  ClearScreen();
  SetSensorLowspeed(ARDUINO_PORT);
  while (true)
  {
    Wait (100);
    TextOut(0, LCD_LINE1, i2cReadString(ARDUINO_PORT, ADDRESS_SEND, 0x42,
15), false);
  }
}
```

270. Mindstorms sensors with Arduino

I planned a chapter about connecting Mindstorms components like the motor, and the various sensors to Arduino. At first I thought that they all would talk the I2C protocol. That proofed wrong. Every component has its own way of communicating and only the ultrasonic rangefinder uses a modified I2C protocol, not supported by the I2C Wire protocol. I found an article at arduino.cc about a home-made shield so you could connect the Mindstorms Motors to Arduino:

- http://arduino.cc/forum/index.php/topic,65022.msg475002.html#msg475002.

An overview of the Mindstorms sensors in combination with Arduino and some pseudo code, can be found at:

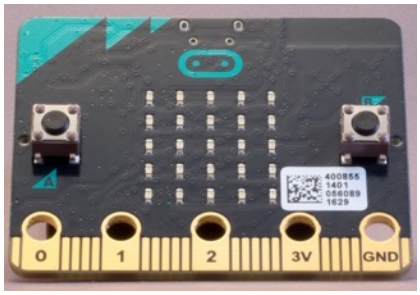
- http://www.instructables.com/id/How-to-use-LEGO-NXT-sensors-and-motors-with-a-non-/

I think this is too complicated at this moment. Since the NX-Brick I used to write this section, was not mine, I decided not to continue with Mindstorms sensors with Arduino.

Non Arduino MCU Boards

This sections covers MCU boards that are not compatible with the Arduino IDE, but interesting enough to describe.

271. BBC Micro:bit

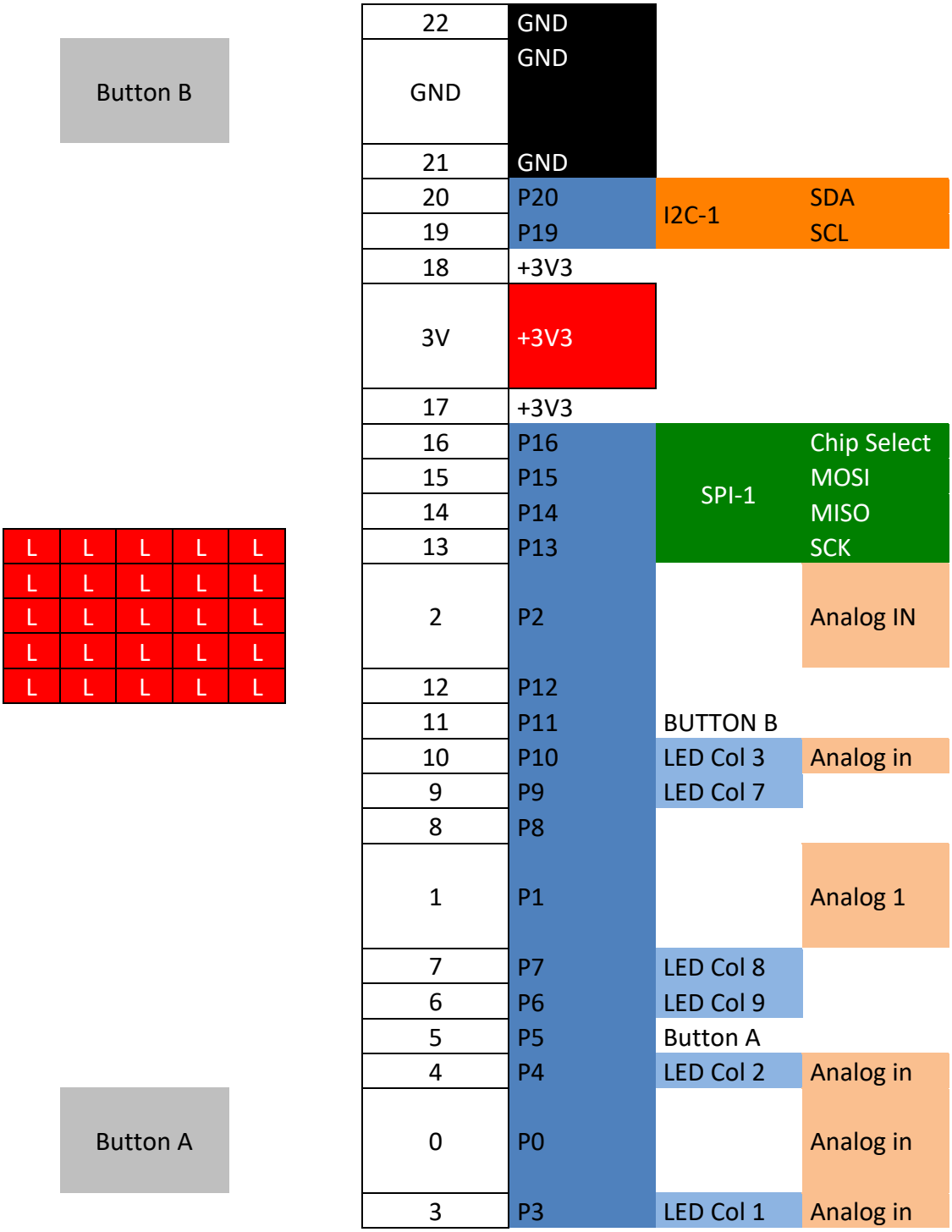


The Micro:bit is an education project of the BBC. This little MCU was given to all 7th graders (11 and 12-year old) in Britain to teach yougsters how to program.

271.1. Specifications BBC

Microcontroller	Nordic nRF51822 ARM Corex-M0 16 MHz
Operating Voltage	3 V (2x1,5V alkaline) or USB 5V
Flash memory	256 KB
RAM	16 KB
Bluetooth	BLE
Accelerometer	NXP/Freescale MMA8652 3-axis via I2C
Magnetometer	NXP/Freescale MAG3110 3-axis via I2C (to act as compass and metal detector)
Connectors	Micro-USB, battery, 20 pin edge connector
Display	25 addressable LEDS in 5x5 matrix
Buttons	Three tactile pushbuttons (1 for reset, 2 programmable)
Edge ring connectors (for banana plug and crocodile clips)	3 for IO 2 for power (GND & 3.3V)
Edge connectors (backside is not connected!)	17 GPIO pins, 6 analog inputs, 2 or 3 PWM, serial I/O, SPI and I2C.

271.2. Layout/connections BBC Micro:bit



Large pins

You can connect to these pins through a banana plug or a crocodile clip.

Pin	Description
GND	Ground
+3V	Output to peripherals Input in case there is no battery nor USB connected
pin 0	GPIO with ADC (analog in), touch sensitive
pin 1	GPIO with ADC (analog in), touch sensitive
pin 2	GPIO with ADC (analog in), touch sensitive

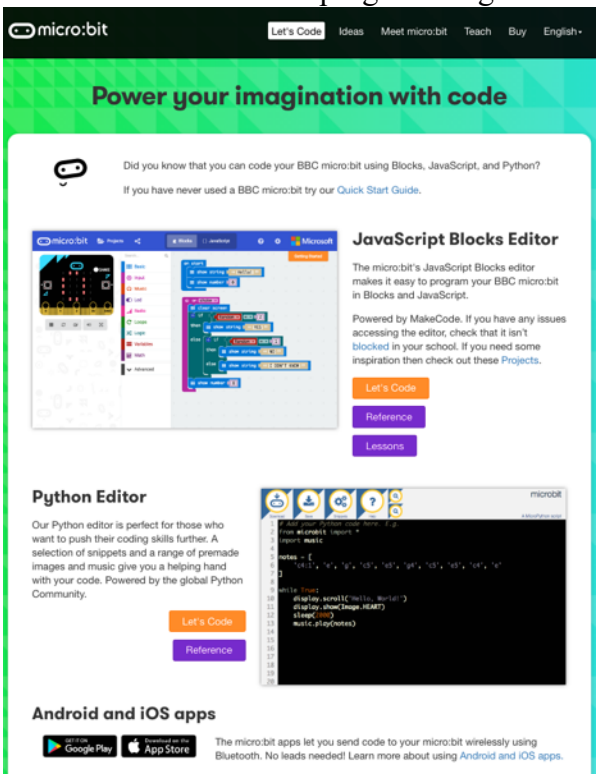
Small pins

You can connect to these pins though an 0.05 inch spaced edge connector.

Pin	Description
pin 3	LED Col 1 GPIO with ADC, when the LED screen is turned off
pin 4	LED Col 2 GPIO with ADC, when the LED screen is turned off
pin 5	Button A (with pull-up resistor) GPIO
pin 6	LED Col 9 GPIO, when the LED screen is turned off
pin 7	LED Col 8 GPIO, when the LED screen is turned off
pin 8	GPIO
pin 9	LED Col 7 GPIO, when the LED screen is turned off
pin 10	LED Col 3 GPIO with ADC, when the LED screen is turned off
pin 11	Button A (with pull-up resistor) GPIO
pin 12	GPIO
pin 13	GPIO for SPI SCK
pin 14	GPIO for SPI MISO
pin 15	GPIO for SPI MOSI
pin 16	GPIO SPI Chip Select
pin 17, 18	Output to peripherals Input in case there is no battery nor USB connected
pin 19	I2C SCL
pin 20	I2C SDA
pin 21,22	Ground

271.3. How to program the Micro:bit

There are several online programming environments in which you can code your Micro:bit.



- Microsoft's JavaScript Block Editor
  A visual editor in which you can drag and drop blocks to build your program, but you can also use JavaScript, you can even switch between these two.
- Python Editor
  A text-based editor based on Python.
- C/C++ with Arduino IDE.

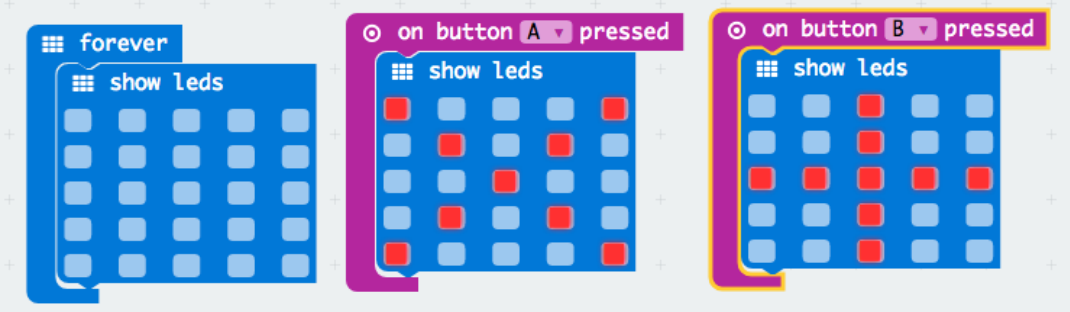
The first two IDE's can be found at: http://microbit.org/code/. The procedure about how to install the board definition for the Arduino IDE is describer further on in this chapter.

I'm only giving one example for a simple program in the Block Editor.

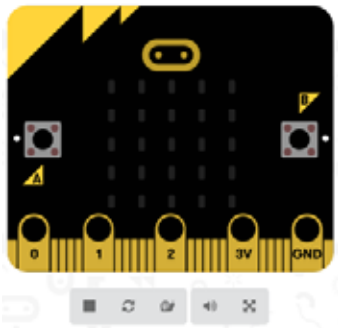
Sample Block Editor

- Connect the Micro:bit to the USB port of your computer.
- Press the RESET button at the back of the Micro:bit, your Micro:bit should now be visible as an external drive called MICROBIT.
- Create the program below, by dragging and dropping the correct blocks.

- Go to: https://makecode.microbit.org/# and edit the program to resemble the code below.



- You can check your program in the simulator in de left sidebar:



- Press the A button briefly, to check the pattern.
- Do the same for the B button.
- You can even take a look at the text-code behind your program, by pressing the {} JAVASCRIPT button:

Blocks { } JavaScript Microsoft

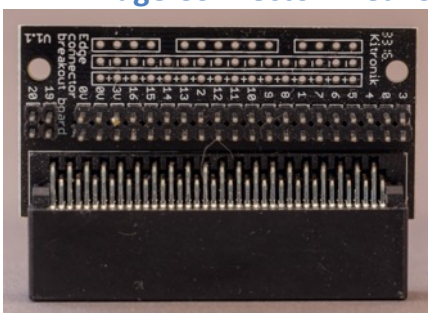
```
input.onButtonPressed(Button.A, () => {
    basic.showLeds(`
        # . . . #
        . # . # .
        . . # . .
        . # . # .
        # . . . #
        `)
})
input.onButtonPressed(Button.B, () => {
    basic.showLeds(`
        . . # . .
        . . # . .
        # # # # #
        . . # . .
        . . # . .
        `)
})
basic.forever(() => {
    basic.showLeds(`
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        `)
})
```

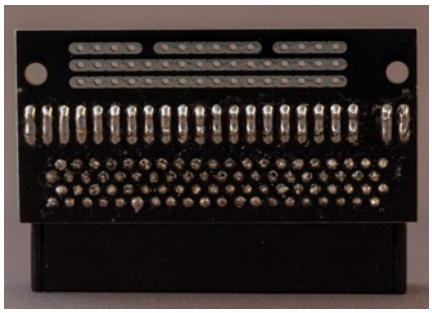
- When you are satisfied, you can compile and place your code on the Micro:bit by pressing the DOWNLOAD button and saving the compiled HEX file to the

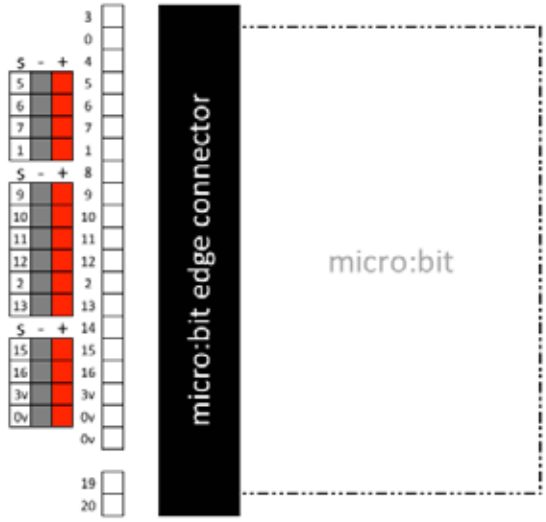
MICROBIT external drive. If the external drive MICROBIT was not visible, you have probably forgotten to press the RESET button in the second step of this small sample.

- Test the result by briefly pressing either the A or the B button.

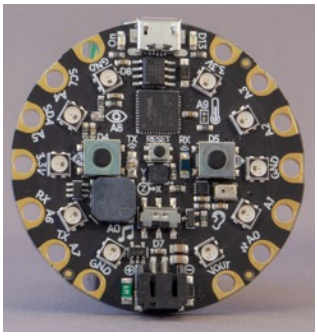
271.4. Edge Connector Breakout board for micro:bit

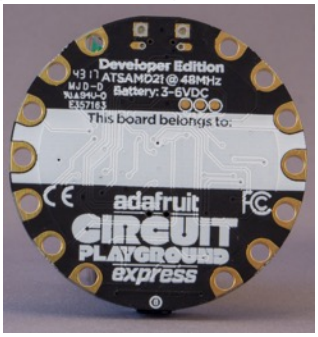






272. Adafruit Playground Express





The Adafruit Playground Express is the succeeder of the Playground Classic. It looks like Adafruit's response to the Micro:bit. Just like the Micro:bit this little board is equipped with loads of on-board sensors an actuators and it is aimed for educational purposes.

272.1. Specifications Adafruit Playground Express

Microcontroller	**ATSAMD21** ARM Cortex M0 Processor, running at 3.3V and 48 MHz
Power Supply	USB: 5V JST connector:
SRAM	256 KB
Flash	• 32 KB Flash • 2 MB SPI Flash used primarily with CircuitPython to store code and libraries, also available for Arduino IDE to store files
I/O	• UART • I2C • SPI • 8 analog input/ PWM output (7 can act as capacitive touch inputs) • Micro USB (can act as HID such as keyboard, mouse, joystick or MIDI)
LED's	• 10 mini NeoPixels i.e. very bright addressable RGB LEDs (WS2812) • 1 RED LED on D13 • 1 GREEN "POWER ON" LED
Motion sensor	**LIS3DH** • triple axis accelerometer • tap detection • free-fall detection
Temperature sensor	NTC thermistor: Murata **NCP15XH103F03RC**
Light sensor (phototransistor)	Photo transistor: **ALS-PT19** • Light level sensor • Color sensor • Pulse sensor
Sound sensor	MEMS microphone
Speaker	Mini speaker/buzzer with class D amplifier
Buttons / switches	• 2 push buttons (A=> D4 & B=> D5) • 1 slide switch (D7)

	• RESET button
Infrared Transmitter	• Transmit remote control codes • Send messages to other Playground Expresses
Infrared Receiver	• Receive remote control codes • Receive message from other Playground Expresses • Proximity sensing

272.2. Connections Adafruit Playground Express

The Playground Express has 14 connection pad. You can use these with alligator clips, directly solder to them, use conductive thread or use 3mm bolts (and nut).

Pin name	External	Internal	
GND	3 pads		Ground
3.3V	2 pads		3.3V output, connected to the output of the on-board regulator (300-450 mA depending on the use of the NeoPixels)
Vout	1 pad		Output voltage equal to USB / Battery input (whichever is the highest), not connected to the on-board regulator (resettable fuse at 500 mA continuous or 1 A peak)
USB	Micro USB		Used to power and program the Playground Express
Battery	JST connector		Used to power the Playground Express through a battery 3.3 – 6 V connected to the input of the on-board regulator
A0	1 pad	x	• Analog I/O • Digital I/O • Internally connected to the built-in speaker
A1 / D6 A2 / D9 A3 / D10	x		• Analog input • Digital I/O (PWM capable) • Capacitive touch sensor
A4 / D3 A5 / D2	x		• Analog input • Digital I/O (PWM capable) • Capacitive touch sensor • A4 = I2C SCL, A5 = I2C SDA
A6 / D0 A7 / D1	x		• Analog input • Digital I/O (PWM capable) • Capacitive touch sensor • D0 = Serial Rx, D1 = Serial Tx
D4		x	Left button A
D5		x	Right button B
D7		x	Slide Switch
D8		x	Built-in NeoPixels
D13		x	Red LED
D27		x	Accelerometer interrupt
D29		x	IR transmitter
D39		x	IR Receiver
A8		x	Light sensor
A9		x	Temperature sensor

Pin name	External	Internal	
A10		x	IR proximity sensor
D28		x	Internal I2C SDA (access with Wire1)
D29		x	Internal I2C SCL (access with Wire1)
D30		x	SPI-MISO
D31		x	SPI-SCK
D32		x	SPI-MOSI
D33		x	SPI-CS

272.3. Datasheets and documentation Adafruit Playground Express

- Lots of information and a step by step tutorial to get started, can be found at Adafruit's website.
  https://learn.adafruit.com/adafruit-circuit-playground-express/overview
- ATSAMD21G18 ARM Cortex M0 Processor
  http://ww1.microchip.com/downloads/en/DeviceDoc/40001882A.pdf
- LIS3DH digital output motion sensor
  https://cdn-shop.adafruit.com/datasheets/LIS3DH.pdf
- Murata NCP15XH103F03RC thermistor
  https://www.murata.com/en-sg/api/pdfdownloadapi?cate=&partno=NCP15XH103F03RC
- ALS-PT19 phototransistor
  https://cdn-shop.adafruit.com/product-files/2748/2748+datasheet.pdf

272.4. Drivers Adafruit Playground Express

You don't need any drivers on OSx nor on Linux, but you do need to install drivers on some versions of Windows.

- You can download the driver installer at:
  https://github.com/adafruit/Adafruit_Windows_Drivers/releases/download/2.2.0/adafruit_drivers_2.2.0.0.exe
  Adafruit advises to install all drivers in this installer.
- Or you can download the driver files (inf/cat) at:
  https://github.com/adafruit/Adafruit_Windows_Drivers/releases/tag/2.2.0

272.5. How to program the Adafruit Playground Express

There are three ways to program the Playground Express:

- MakeCode
  A visual editor in which you can drag and drop blocks to build your program, but you can also use JavaScript, you can even switch between these two. It is the same as the Microsoft's JavaScript Block Editor for the Micro:bit.
- CircuitPython
  A Python interpreter/compiler loaded on the Playground Express.
- C/C++ with Arduino IDE. The procedure about how to install the board definition for the Arduino IDE is describer further on in this chapter.

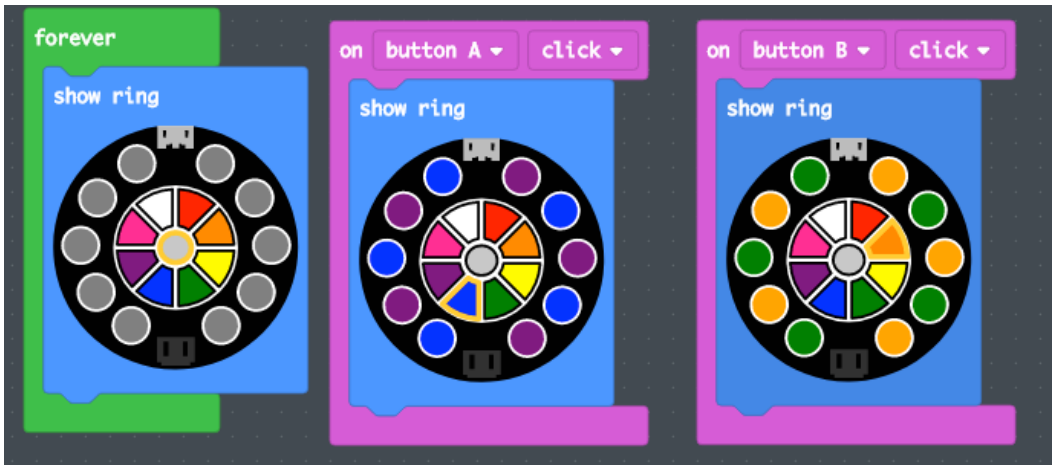
Playground Express with MakeCode

MakeCode is a web based graphical programming interface from Microsoft, using drag-and-drop blocks like with the Micro:bit.

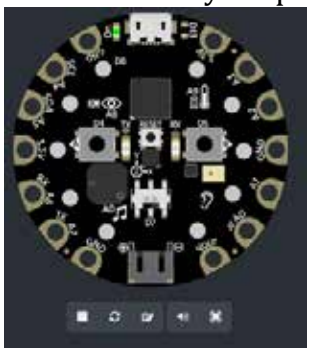
Sample MakeCode PlaygroundExpress_001_LEDButton (CURRENT.U2F)

Since this document is about Arduino, I will only briefly show how to use MakeCode.

- Connect the Playground to the USB port of your computer and then press the RESET button. If you have installed the correct drivers, your Playground Express should be visible as an external drive called CPLAYBOOT, the NeoPixels should be lit green. Sometimes, you need to press the RESET button a second time to get the green circle of LED lights.
- Got to https://makecode.adafruit.com/ and create the program below, by dragging and dropping the correct blocks.
- Edit the blocks to resemble the code below.

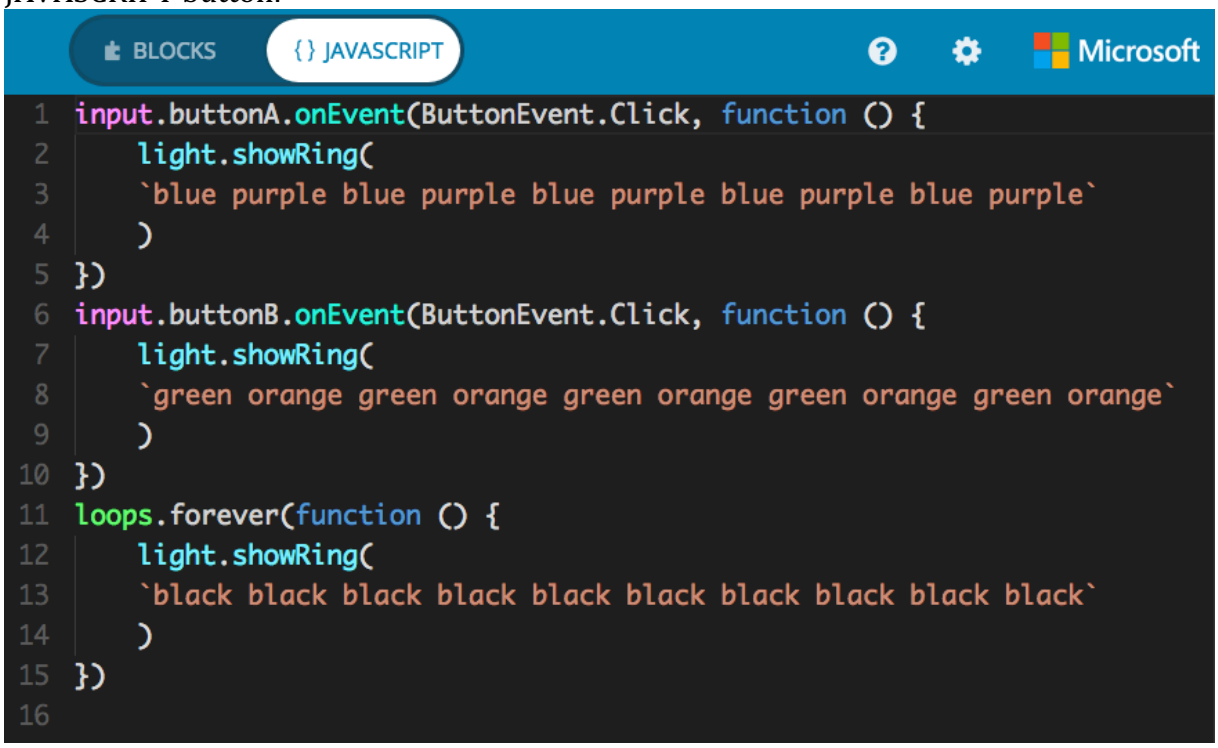


- You can check your program in the simulator in de left sidebar:



- Press the A button briefly, to check the pattern.
- Do the same for the B button.

- You can even take a look at the text-code behind your program, by pressing the {} JAVASCRIPT button:



- When you are satisfied, you can compile and place your code on the Playground Express by pressing the DOWNLOAD button and saving the compiled UF2 file to the CPLAYBOOT external drive. If the external drive CPLAYBOOT was not visible, you have probably forgotten to press the RESET button in the second step of this small sample.
- A very nice feature of this UF2 file is, that the source code is also embedded in this file. So you can download the running UF2 (CURRENT.UF2) file from any Playground Express and open it in the MakeCode editor, so you can continue working on a copy of the original source code (both graphical as JavaScript).
- Test the result by briefly pressing either the A or the B button.

Playground Express with CircuitPython

CircuitPython is a text based programming environment based on Python and should be installed on the Flash memory of the Playground Express itself. Source code, libraries and other files, should be installed on the 2MB SPI Flash that will be available, as soon as CircuitPython is installed.

Preparing Playground Express for CircuitPython

- Connect the Playground to the USB port of your computer and then press the RESET button. If you have installed the correct drivers, your Playground Express should be visible as an external drive called CPLAYBOOT, the NeoPixels should be lit green. Sometimes, you need to press the RESET button a second time to get the green circle of LED lights.
- Download CircuitPython and place it at the CPLAYBOOT drive. https://github.com/adafruit/circuitpython/releases/download/2.2.3/adafruit-circuitpython-circuitplayground_express-2.2.3.uf2
- After placing CircuitPython on the CPLAYBOOT drive, this drive will disappear and will be replaced by a drive called CIRCUITPY.
- The next step is to download the Adafruit CircuitPython Library bundle and unzip it. Copy the unzipped lib folder onto the CIRCUITPY drive. https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases/download/20180213/adafruit-circuitpython-bundle-2.2.1-mpy-20180213.zip
- You are now ready to program with CircuitPython.
- More information about installing CircuitPython can be found at: https://learn.adafruit.com/welcome-to-circuitpython/installing-circuitpython
- If you ever need to remove CircuitPython (for example because you want to return to MakeCode or to continue with Arduino IDE), you can start the boot load mode by double pressing the RESET button. After that the circle of green LED's will appear and you can upload either a UF2 file or a sketch from the Arduino IDE.
- As soon as you save a file called **code.py** on the drive CIRCUITPY, it will run on the Playground Express. For this reason it is important that your editor will save the code.py file completely instead of keeping it in cache. This is not the case with all editors (nano, notepad, notepad++ are NOT suitable for working with CircuitPython). Adafruit recommends the editor called MU, which is a nice editor with support for both Playground Express and also for the Micro:bit. Mu also supports REPL a form of Serial Monitor like in the Arduino IDE. (https://learn.adafruit.com/welcome-to-circuitpython/installing-mu-editor).

Sample with CircuitPython

In the following sample all LED's will turn BLUE as long as you press button A and RED as long as you press button B. Be very precise with the indentation. Python uses indentation instead of brackets {}.

Circuitpython_001_LEDButton.py (save as code.py on the Playground Express)

```
from digitalio import DigitalInOut, Direction, Pull
import board
import neopixel
import time

pixels = neopixel.NeoPixel(board.NEOPIXEL, 10, brightness=.2,
auto_write=False)
pixels.fill((0, 0, 0))
pixels.show()

buttonA = DigitalInOut(board.BUTTON_A)
buttonA.direction = Direction.INPUT
buttonA.pull = Pull.DOWN

buttonB = DigitalInOut(board.BUTTON_B)
buttonB.direction = Direction.INPUT
buttonB.pull = Pull.DOWN

while True:
    RED = (0x10, 0, 0)
    BLUE = (0, 0, 0x10)
    BLACK = (0, 0, 0)

    if buttonA.value is True:
        for i in range(len(pixels)):
            pixels[i] = BLUE
        pixels.show()
        time.sleep(1)
    else:
        if buttonB.value is True:
            for i in range(len(pixels)):
                pixels[i] = RED
            pixels.show()
            time.sleep(1)
        else:

            for i in range(len(pixels)):
                pixels[i] = BLACK
            pixels.show()
            time.sleep(1)
```

Playground Express with Arduino IDE

Is it also possible to install the board definition for the playground express in the Arduino IDE. This will give you lots of programming samples and support for a ton of sensors and actuators.

Preparing Arduino IDE for Playground Express

The processor used in the Playground Express is a SAMD chip. Arduino supports SAMD chip boards with a standard board definition. You only need to install it with the Boards Manager.

- Go to: TOOLS, BOARD:.., BOARDS MANAGER.
- In the search bar type Playground Express.
- Select the board: ARDUINO SAMD BOARDS (32-BITS ARM CORTEX MO+) by Arduino.
- Click on INSTALL.
- Quit and restart the Arduino IDE so the new boards definition will be loaded.
- Before you can flash a sketch, you'll first need to select the Playground Express board and the correct settings.
    - Go to: TOOLS, BOARD:..., Adafruit Circuit Playground Express
    - Go to: TOOLS, PORT, select the serial port assigned to the Playground Express
- You can program the Playground Express in the same way as any other Arduino board, but there are some exeptions (https://learn.adafruit.com/adafruit-circuit-playground-express/adapting-sketches-to-m0). This is beyond the scope of this document.
- Another way to use the Arduino IDE, is to use Adafruit's CircuitPlayground library Below is a short introduction to this library and a sample sketcht. More information can be found at: https://learn.adafruit.com/circuit-playground-lesson-number-0/basic-leds

```
#include <Adafruit_CircuitPlayground.h>
```

Include Adafruit's CircuitPlayground library (included in the Playground board definitions.

```
CircuitPlayground.begin();
```

Initialize the CircuitPlayground library.

```
CircuitPlayground.clearPixels();
```

Clear all NeoPixels at once.

```
CircuitPlayground.leftButton()
```

Results in the status of the left button.

```
CircuitPlayground.rightButton()
```

Results in the status of the right button.

```
CircuitPlayground.redLED(HIGH);
```

Turns on LED on D13 (red LED).

```
CircuitPlayground.setPixelColor(LEDNR, RED, GREEN, BLUE);
```

Set the color for a specific LED (this happens instantly).

048_PlaygroundExpress.ino

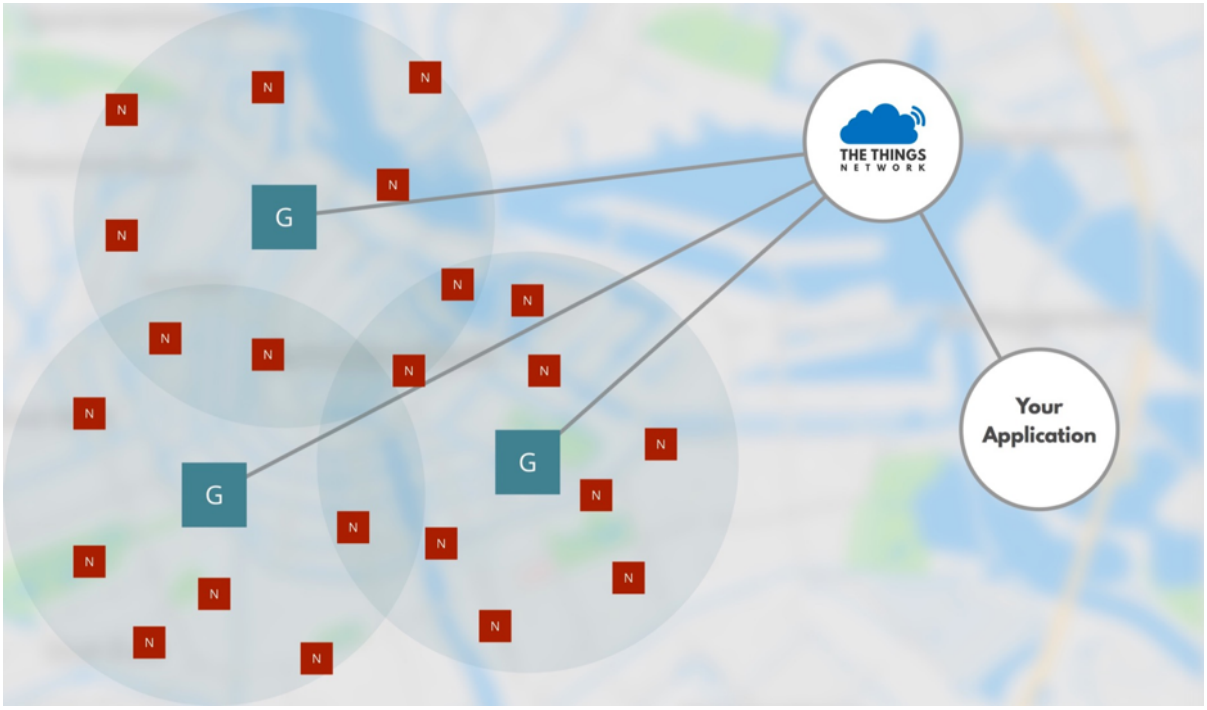
With this sketch the NeoPixel ring will light green when the left button is pressed and red when the right button is pressed. Both buttons will toggle the status of the red LED on D13.

```
#include <Adafruit_CircuitPlayground.h>

boolean LEDSTATUS = false;

void setup()
{
  CircuitPlayground.begin();
}

void loop()
{
  CircuitPlayground.clearPixels();
  if (CircuitPlayground.leftButton())
  {
    CircuitPlayground.redLED(LEDSTATUS = !LEDSTATUS);
    for (int nr = 0; nr < 10; nr++)
    {
      CircuitPlayground.setPixelColor(nr, 0, 255, 0);
    }
  }
  if (CircuitPlayground.rightButton())
  {
    CircuitPlayground.redLED(LEDSTATUS = !LEDSTATUS);
    for (int nr = 0; nr < 10; nr++)
    {
      CircuitPlayground.setPixelColor(nr, 255, 0, 0);
    }
  }
  delay(100);
}
```

The Things Network

This section describes the steps you need to take to get started with the Internet of Things (IoT) at The Things Network. This section contains samples for working with The Things Network, using Arduino and LoRa modules for the nodes and optionally a Raspberry Pi as a single channel gateway.

273. Basics The Things Network[1]



The picture above shows the typical The Things Network model with 4 basic components, providing low power wireless connectivity over a long range.

- N: Nodes, are the devices that collects and broadcasts data. The nodes in this document are built around Arduino. They use Lora (Low Power, Long Range) modules to transmit their data. Nodes only broadcasts and receives only very small messages periodically. Very often those messages are only 1-4 bytes in size and they are send only once every few minutes to once every hour. Fair Uses states you are only allowed to use 1% of the bandwidth.
- G: Gateways, are devices that bridges the data received on the LoRaWAN and sends it to the Backend of the TTN network.
- The Backend routes messages from the nodes to the right application.
- The applications receives and processes the messages from the nodes.

273.1. Setup The Things Network

The setup of TTN consists of three major steps:

- Configuration of a (Single Channel) Gateway in case there is no TTN Gateway in your neighborhood, otherwise this step is optional.
- Configuration of your application at TTN.
- Configuration of your device(s) in your application at TTN.
- Configuration of your physical nodes (in the next chapter).
-

[1] This picture is from the website of The Things Network.

274. Single Channel Gateway on Raspberry Pi (J.F. Telkamp)

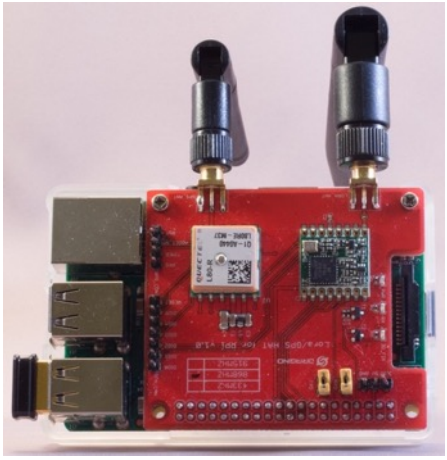
In case there is no TTN Gateway in your neighborhood, you can either buy a multi-channel Gateway and configure this for TTN (not covered in this document), or connect a Raspberry Pi with Lora module to internet and install a Single Channel Gateway on it.

This paragraph describes the configuration of a Raspberry Pi with either a Dragino Lora/GPS HAT or a HopeRF board and RFM95W module. *The specifications of the Dragino HAT and the HopeRF board are described elsewhere in this document.*

Some differences between a Single Channel Gateway and a full LoRaWAN gateway:

- This single Channel Gateway only listens on 1 channel and one Spreading Factor. Most nodes will rotate through all the channels (randomly?), so not all the messages (98%) from your nodes will be retrieved. Using LoRa modules like the SX127x or RFM9xW, you can alter your sketch, so only one channel/spreading factor is used. This is not possible with the RN2483 modules.
- This Single Channel Gateway can't send messages back to the nodes (no Download and no Acknowledgements).
- This Single Channel Gateway cannot service OTAA authentication, because for that the Download feature is needed.
- Check the next chapter with Jaap Braam's Single Channel Gateway if you want to listen on multiple Spreading Factors, if you need the Download feature and or need OTAA.
- The range of a Single Channel Gateway, even with a perfect Line of Sight is very small.
- It is not possible to use the RN2483 module in any Gateway, you must use LoRa modules like the SX127x or RFM9xW.

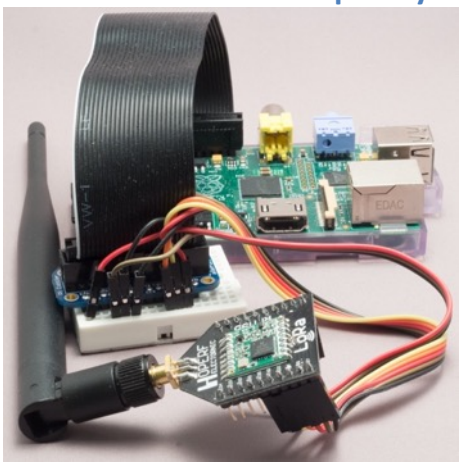
274.1. Connections Raspberry Pi with Dragino Lora/GPS HAT



Place the Dragino Lora/GPS on top of your Raspberry Pi (2 and up). Nss is now connected to GPIO6, dio0 to GPIO7 and RST to GPIO0 (all GPIO port names according to WiringPi).

Skip the next paragraph and continue with the Configuration of the Single Channel Gateway.

274.2. Connections Raspberry Pi with HopeRF adapter and RFM95W module



Connect the HopeRF board according to the following table.

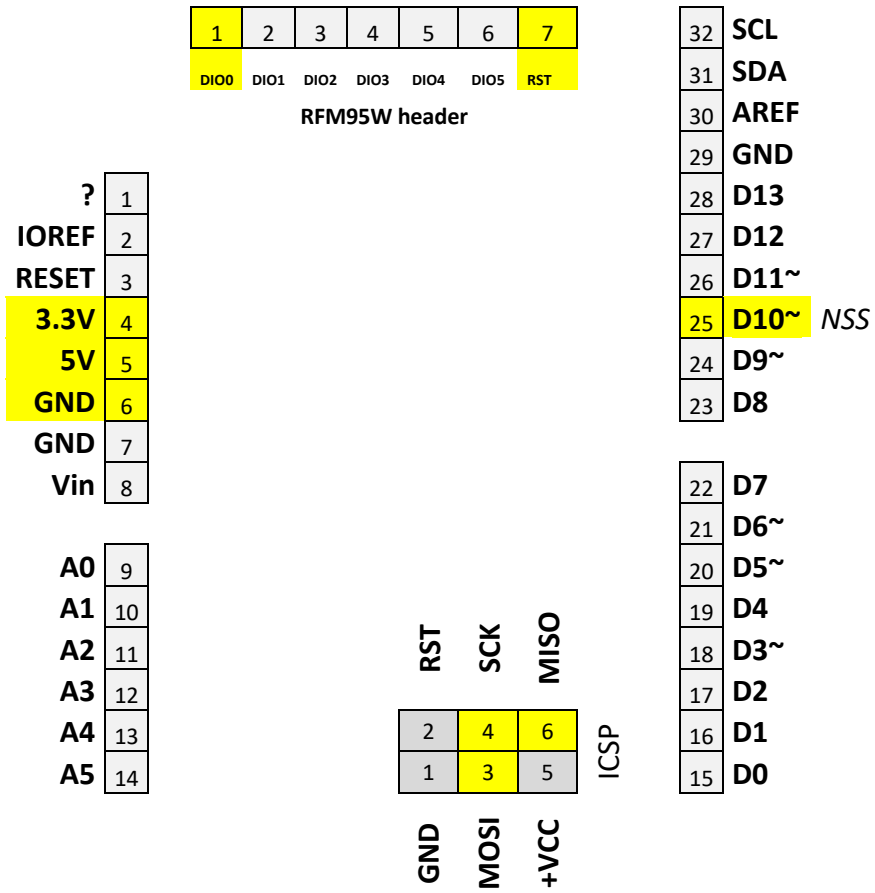
Pin nr	Name	Description	Raspberry Pi pin nr	Wiring Pi name	BCM pin name
5	DIO0	Digital I/O software configured	7	GPIO7	GPIO4
6	MISO	SPI Data output	21	MISO	
7	MOSI	SPI Data input	19	MOSI	
8	SCK	SPI Clock input	23	CLK	
9	nSS	SPI Chip Select input	22	GPIO6	GPIO25
10	RST	Reset trigger input	11	GPIO0	GPIO17
11	3.3V	Power Supply	1	3.3V	
14	GND	Ground	6	GND	

In this table, I've used the same pins as with the Dragino Board.

Continue with the paragraph about the Configuration of the T.F. Telkamp Single Channel Gateway.

274.3. Connections Raspberry Pi with Dragino (Arduino) Shield

The Dragino Shield is designed for the Arduino, but by using jumper cables you can also use this shield on the Raspberry PI.



Connect the Dragino (Arduino) Shield according to the following table.

Pin nr	Name	Raspberry Pi pin nr	Wiring Pi name	BCM pin name
ICSP Header: pin 1	MISO	21	MISO	
ICSP Header: pin 3	SCK	23	CLK	
ICSP Header: pin 4	MOSI	19	MOSI	
RFM Header: pin 1	DIO0	7	GPIO7	GPIO4
RFM Header: pin 7	RST	11	GPIO0	GPIO17
D10	nSS	22	GPIO6	GPIO25
3.3V	3.3V	1	3.3V	
5V	5V	2		
GND	GND	6	GND	

Notes:

- The Dragino Shield was designed to be used on a 5V Arduino, so most pins are connected to level shifters. To get these level shifters working you need both 3.3V as well as 5V connections between the shield and the Raspberry Pi.
- On an Arduino you'll find MOSI, MISO and SCK on D11, D12 & D13, but that didn't work in my setup, so I used MOSI, MISO and SCK on the ICSP header pins 4, 1 & 3 instead.
- On an Arduino this shield connects DIO0 and RST to D2 and D9, but in my setup this didn't work, so I used the RFM95W header pins 1 and 7 instead.
-

In this table, I've used the same pins as with the Dragino.

Continue with the next paragraph about the Configuration of the T.F. Telkamp Single Channel Gateway.

274.4. Configuration of T.F. Telkamp's Single Channel Gateway

After installing the LoRa module you must download configure and compile the Single Channel Gateway software.

```
git clone https://github.com/tftelkamp/single_chan_pkt_fwd ~/Desktop/IOT
cd ~/Desktop/IOT
sudo nano main.cpp
```

```
// SX 1272 – Raspberry connections
int ssPin =6
int dioO = 7
int RST = 0

// Set location
float lat=0.0;
float lon=0.0;
int alt=0;

/* Informal status fields */
static char platform[24]    = “”
static char email[40]       = “”
static char description[6]  = “”

//define servers
#define SERVER1 “52.169.76.203”  // router.eu.thethings.network
```

The settings for the LoRa module already matches those chosen in the previous paragraphs.

Edit the Information Status fields and the coordinates of your gateway at the Set location section.

Change the IP address to 52.169.76.203[1]

Save the file and continue with the following commands:

```
sudo make
sudo ./single_chan_pkt_fwd
```

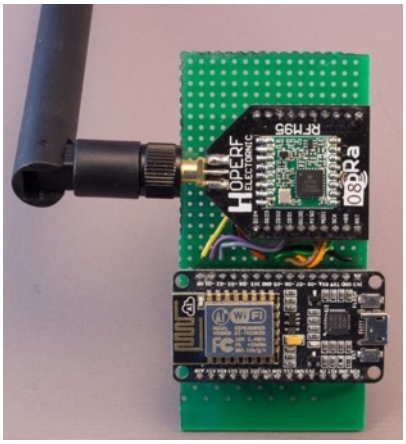
This will result in the following output on your screen, containing `stat update` every 30 seconds and `rxpk update` every time a message from a node was received.

```
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:bf:d5:38
Listening at SF7 on 868.100000 Mhz.
------------------
stat update: {"stat":{"time":"2016-11-06 18:42:20
GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":1,"rxok":1,"rxfw":0,"ack
r":0.0,"dwnb":0,"txnb":0,"pfrm":"Single Channel
Gateway","mail":"","desc":""}}
stat update: {"stat":{"time":"2016-11-06 18:42:50
GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ack
r":0.0,"dwnb":0,"txnb":0,"pfrm":"Single Channel
Gateway","mail":"","desc":""}}
```

[1] This address was found by resolving router.eu.thethings.network (52.169.76.203) or router.us.thethings.network (13.66.213.36).

```
Packet RSSI: -42, RSSI: -109, SNR: 9, Length: 22
rxpk update:
{"rxpk":[{"tmst":1191593527,"chan":0,"rfch":0,"freq":868.100000,"stat":1,"m
odu":"LORA","datr":"SF7BW125","codr":"4/5","lsnr":9,"rssi":-
42,"size":22,"data":"QCADAgGAAQAB5+ZysTEehl+tk1cB9w=="}]}
stat update: {"stat":{"time":"2016-11-06 18:43:20
GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":1,"rxok":1,"rxfw":0,"ack
r":0.0,"dwnb":0,"txnb":0,"pfrm":"Single Channel
Gateway","mail":"","desc":""}}
```

275. Single Channel Gateway on ESP8266 (Jaap Braam)



This Single Channel Gateway is written in LUA and is based on the popular ESP8266 module. It has some extra features compared to the T.F. Telkamp's Single Channel Gateway from the previous chapter.

- It listens on all Spreading Factors.
- It can send downlink messages (on all channels?).
- It supports OTAA (because of the downlink feature).
- It supports WiFi, but it lacks Ethernet.
- The ESP8266 is way cheaper than the Raspberry.

This paragraph describes the configuration of the ESP8266 E12 Node MCU module with an HopeRF board and RFM95W module. *The specifications of the HopeRF board are described elsewhere in this document.*

Some differences between a Single Channel Gateway and a full LoRaWAN gateway:

- This single Channel Gateway only listens on 1 channel and one Spreading Factor. Most nodes will rotate through all the channels (randomly?), so not all the messages from your nodes will be retrieved. Using LoRa modules like the SX127x or RFM9xW, you can alter your sketch, so only one channel/spreading factor is used. This is not possible with the RN2483 modules.
- The range of a Single Channel Gateway, even with a perfect Line of Sight is very small.
- It is not possible to use the RN2483 module in any Gateway, you must use LoRa modules like the SX127x or RFM9xW.

275.1. Connections ESP8266 E12 Node MCU with HopeRF adapter and RFM95W module

Connect the HopeRF board according to the following table.

Pin nr	Name	Description	ESP8266 E12 Node MCU pin
4	DIO1	Digital I/O	D2
5	DIO0	Digital I/O	D1
6	MISO	SPI Data output	D6
7	MOSI	SPI Data input	D7
8	SCK	SPI Clock input	D5
9	nSS	SPI Chip Select input	D0

11	3.3V	Power Supply	3.3V
14	GND	Ground	GND

Continue with the next paragraph about the Configuration of Jaap Braam's Single Channel Gateway.

275.2. Configuration of Jaap Braam's Single Channel Gateway

After connecting a SX1276 or RFM95W module, the configuration of Jaap Braam's Single Channel Gateway consists of the following steps.

- Download the source code: https://github.com/JaapBraam/LoRaWanGateway
- Get a modified version of the NodeMCU firmware at http://nodemcu-build.com/index.php
  - Select the dev branch
  - Select the following module: bit, cjson, encoder, file, gpio, net, node, rtctime, sntp, spi, tmr, uart and wifi
  - Download the integer version of this custom build firmware.
- Flash the custom build firmware to the ESP8266 module.
- Format the ESP8266 filesystem.
- Connect to your WiFi (you'll need to do this only once after flashing the ESP8266).
  - wifi.setmode(wifi.STATION)
  - wifi.sta.config("<SSID>","<WPA2 key>")
  - wifi.sta.autoconnect(1)
  - wifi.sta.connect()
- Upload the three lua files from Jaap Braam's sourcecode.
  - init.lua
  - LoRaWanGW.lua
  - SX1276.lua
- Restart your ESP8266
- Use some terminal program (like Serial Monitor in Arduino IDE, or PuTTY) at 115200 baud.

This will result in the following output on your screen, containing `rxpk` messages every time a message from a node was received and `txpk` every time a download message was received from TTN and a `transmitPkt` when that downloaded message was send to the node.

```
NodeMCU custom build by frightanic.com
     branch: dev
     commit: 378e5eb0ad04a555da4a9e4939d88fc771d1f13f
     SSL: false
     modules:
bit,cjson,encoder,file,gpio,net,node,rtctime,sntp,spi,tmr,uart,wifi
 build    built on: 2017-01-09 19:39
 powered by Lua 5.1.4 on SDK 2.0.0(656edbf)
> got ip  10.1.1.124    255.255.255.0 10.1.1.254
Gateway ID    5CCF7FF42F0674FB
ntp synced using 178.239.61.38
2017-01-10 21:29:41 GMT
router ip:    52.169.76.203
rx timeout    7    0    rssi 43
rxpk 01cd9b005ccf7ff42f0674fb    message   {"rxpk":[{"rssi":-
36,"stat":1,"modu":"LORA","rfch":0,"tmst":49943085,"datr":"SF7BW125","lsnr"
:10,"time":"2017-01-
10T21:30:23.849586Z","codr":"4/5","data":"QOOq8VkAtQQB44BCOtzJG0pV5Q==","fr
eq":868.100,"chan":0,"size":19}]}length    233
rx timeout    7    0    rssi 42
rx timeout    7    0    rssi 73
rx timeout    7    0    rssi 43
rx timeout    7    0    rssi 41
```

```
rxpk 018c56005ccf7ff42f0674fb    message   {"rxpk":[{"rssi":-
36,"stat":1,"modu":"LORA","rfch":0,"tmst":66418322,"datr":"SF7BW125","lsnr"
:10,"time":"2017-01-
10T21:30:40.325407Z","codr":"4/5","data":"QOOq8VkAuAQBkd5Y0TwfzoSJLA==","fr
eq":868.100,"chan":0,"size":19}]}length   233
rxpk 01034a005ccf7ff42f0674fb    message   {"rxpk":[{"rssi":-
37,"stat":1,"modu":"LORA","rfch":0,"tmst":81849379,"datr":"SF7BW125","lsnr"
:9,"time":"2017-01-
10T21:30:55.755852Z","codr":"4/5","data":"QOOq8VkAuwQBIFx7Jx0mvRPNlQ==","fr
eq":868.100,"chan":0,"size":19}]}length   232
rx timeout    7    0    rssi 43
rx timeout    7    0    rssi 42
rxpk 01ba39005ccf7ff42f0674fb    message   {"rxpk":[{"rssi":-
36,"stat":1,"modu":"LORA","rfch":0,"tmst":98326334,"datr":"SF7BW125","lsnr"
:9,"time":"2017-01-
10T21:31:12.232594Z","codr":"4/5","data":"QOOq8VkAvgQBikbZTbwUEeCVqA==","fr
eq":868.100,"chan":0,"size":19}]}length   232
rx timeout    7    0    rssi 43
txpk
    {"txpk":{"imme":false,"tmst":99326334,"freq":868.1,"rfch":0,"powe":14,
"modu":"LORA","datr":"SF7BW125","codr":"4/5","ipol":true,"size":15,"data":"
YOOq8VkABAABH58ijZv5"}}
txpk_ack  {"txpk_ack":{"error":"NONE"}}
transmitPkt   8739 5120 -2   868100000 112  112  2    64   14 15
rx timeout    7    0    rssi 42
rx timeout    7    0    rssi 43
```

276. Configuration of applications and devices at TTN

The following steps describe how to get started with IOT at The Things Network. A complete Quick Start can be found at:
https://www.thethingsnetwork.org/docs/current/cli/

Starting at December 14th 2016, TTN made a transition from the Staging (v1) environment to the Production (v2) environment. This document describes the procedures and software you'll need to work with the Production environment.

- Download The Things Network command line interface software called ttnctl
  - MAC: https://ttnreleases.blob.core.windows.net/release/master/ttnctl-darwin-amd64.zip
  - Linux (64 bits): https://ttnreleases.blob.core.windows.net/release/master/ttnctl-linux-amd64.zip
  - Windows (64 bits): https://ttnreleases.blob.core.windows.net/release/master/ttnctl-windows-amd64.exe.zip
- Extract the zip-file to a convenient location and rename the executable to ttnctl.
- Create an account at https://account.thethingsnetwork.org/users/login
- Confirm this account by clicking in the activation link in your mail.
- Update ttnctl to the latest version.

```
ttnctl selfupdate
```

- Go to https://account.thethingsnetwork.org/
- Login with your new account and click on TTNCTL ACCESS CODE
- Left click on this code so the code will be selected and then copy the selected code.
- Within 5 minutes after this, you'll need to type the following command:

```
ttnctl user login <ttnctl-access-code>
```

  At this point you will be asked for your password. It looks like that your credentials are stored in a local config file. So you only have to login once per computer.

- Create your first application

```
ttnctl applications add <application-id> "<application-title>"
```

  You are free to choose a descriptive <application-id> using only lower case, numbers, score (-) and underscore (_).
  You are also free to choose an <application-title> using upper case, lower case, numbers, spaces and some other characters.

- Examine the list of available applications (1 at this point).

```
ttnctl applications list
```

  This can also be found at https://console.thethingsnetwork.org/applications.

- Connect to this new application.
```
ttnctl applications select
```
If there are more than 1 applications, a list will be shown, so you can select the correct application, but at this point there is only 1 application available. So that application will be selected automatically.

- You must now register this applications to a nearby handler.
```
ttnctl applications register <handler>
```
<handler> is the name of the nearby handler (example: ttn-handler-eu)
You can also omit <handler> to let ttnctl decide the name of the handler.

- Create your first device (node) for this application. You can choose between an Over-The-Air activated device (OTAA) or a personalized device (ABP). In this sample I use the ABP type.[1]
```
ttnctl devices register <device-id>
```
You are free to choose a descriptive <device-id> using only lower case, numbers, score (-) and underscore (_).

```
ttnctl devices personalize <device-id>
```

- A list of all available devices in your selected application can be found through:
```
ttnctl devices list
```

- Examine the information like the DevAddr, NwsKey and AppSkey for your device. These three strings are needed in your nodes. See section The Things Network Nodes in this document!
```
ttnctl devices info <device-id>
```

- You can also check this information (in different formats) through: https://console.thethingsnetwork.org/applications
    - Click on the correct application.
    - Click in the section DEVICES on MANAGE DEVICES
    - Click on the correct DEVICE
    - You'll find the information you'll need
        - Device Address
        - Network Session Key
        - App Session Key

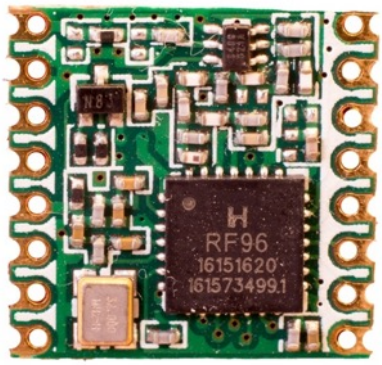
At this point you have at least one device and one application. Continue with the section The Things Network Nodes and after configuring your node, continue at the section The Things Network Data Handling.

[1] The main reason why I choose to use ABP over OTAA is that there is only a Single Channel Gateway in my neighborhood, so OTAA is not possible. (My) Single Channel Gateway does not support Download links, which is needed for the OTAA authentication process.

The Things Network Nodes

LoRa is an abbreviation of Low power/Long Range transceivers. It can be used in The Things Network and other Internet of Things networks.

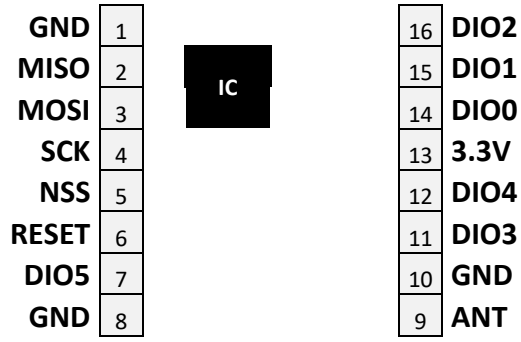
277. LoRa: RFM95W module



This LoRa modem is used for Internet of Things applications. It is a Low Power Long Range Transceiver. For a typical TTN application, it needs to connect to a TTN Gateway. Depending of the Line of Sight this can be at a distance of up to 10KM. It is meant to be used for very slow connections.

The RFM95W module is not breadboard friendly. The distance between the connectors is way smaller than the default of 0.1 inch. You must solder directly to the connectors or use some sort of adapter board, like the 278 LoRa: HopeRF Adapter board for RFM95W.

The RFM95W is compatible with the SX1276 LoRa module. Note that the name of the IC on top of the RFM95W module is RF9**6** to show that this SX127**6** compatible (the RFM98W contains a RF9**8** chip, showing that it is SX127**8** compatible).



The RFM95W module does only contain the LoRa radio module and. The LoRaWAN protocol stack must be programmed in the MCU (Arduino). Because of this, the sketches are very large, but on the other hand, you'll be able to do things that are not "allowed" in the LoRaWAN description. This makes it possible to build a Single Channel Node to communicate with your own Single Channel Gateway. Nice during development and testing, but not so good for production!

277.1. Specifications LoRa: RFM95W module

- Low Power Long Range
- Module Size16x16 mm
- LoRa modem
- Frequency: 868/915 MHz
- Spreading :Factor 6-12
- Bandwidth: 7.8-500 kHz
- Effective Bitrate: 0.018 - 37.5 kbps
- Est. Sensitivity: -111 - -148 dBm
- VCC: 3.3V

277.2. Datasheet LoRa: RFM95W module

- http://webshop.ideetron.nl/Files/3/1000/1211/Attachments/Product/38C8M4K5j578I7Go79331xub2L8Dr2Al.pdf

277.3. Connections LoRa: RFM95W module

Pin nr	Name	Description	Arduino pin
1	GND	Ground	GND
2	MISO	SPI Data output	D12
3	MOSI	SPI Data input	D11 through 5v-3.3V level shifter
4	SCK	SPI Clock input	D13 through 5v-3.3V level shifter
5	NSS	SPI Chip Select input	D10 through 5v-3.3V level shifter
6	RESET	Reset trigger input	Any Digital port through 5v-3.3V level shifter
7	DIO5	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
8	GND	Ground	GND
9	ANT	Antenna	-
10	GND	Ground	GND
11	DIO3	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
12	DIO4	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
13	3.3V	Voltage Supply	3.3V
14	DIO0	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
15	DIO1	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
16	DIO2	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input

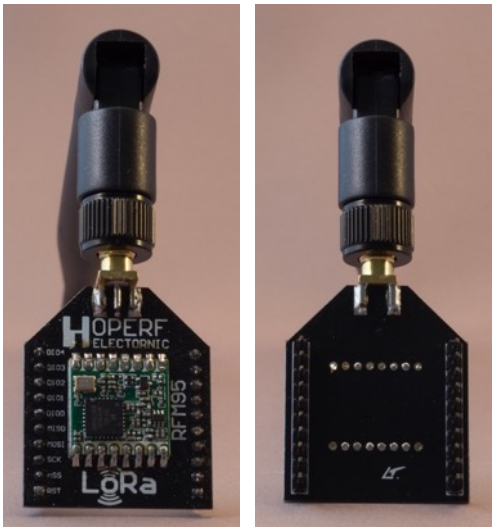
277.4. Libraries and sample LoRa: RFM95W module

I have never used this without some kind of board and shield, so check the appropriate chapters for libraries and a sample sketch.

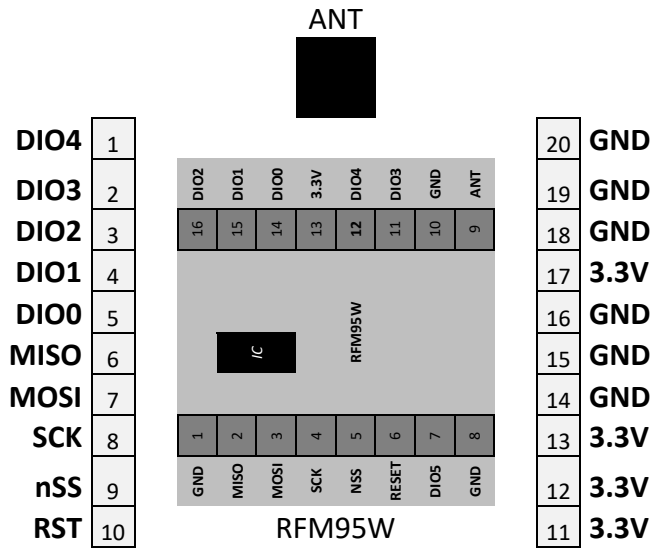
Check for a description of an adapter board at:

- "278 LoRa: HopeRF Adapter board for RFM95W" in the next chapter.

278. LoRa: HopeRF Adapter board for RFM95W



This adapter board is a 0.1 inch spaced PCB for the RFM95W module. Although it is spaced 0.1 inch, it is not breadboard friendly, because it is 1 pin to wide. Below is the diagram of the HopeRF Adapter board. Normally this board is sold without the RFM95W module. You have to buy the RFM95 separate and solder it yourself.



278.1. Specifications HopeRF Adapter board for RFM95W

- 0.1 inch spaced
- SMA connector for 868 MHz Antenna
- DIO5 of the RFM95W module is not connected to any of the header pins.

278.2. Connections HopeRF Adapter board for RFM95W

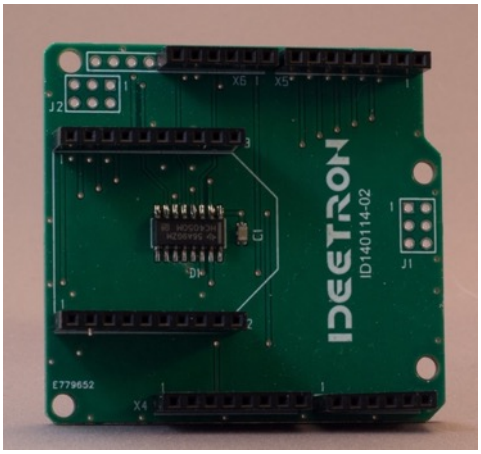
Pin nr	Name	Description	Arduino pin
1	DIO4	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
2	DIO3	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
3	DIO2	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
4	DIO1	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
5	DIO0	Digital I/O software configured	Any Digital port possibly through 5v-3.3V level shifter when used as input
6	MISO	SPI Data output	D12
7	MOSI	SPI Data input	D11 through 5V-3.3V level shifter
8	SCK	SPI Clock input	D13 through 5V-3.3V level shifter
9	nSS	SPI Chip Select input	D10 through 5V-3.3V level shifter
10	RST	Reset trigger input	Any Digital port through 5V-3.3V level shifter
11-13	3.3V	Power Supply	3.3V
14-16	GND	Ground	GND
17	3.3V	Power Supply	3.3V
18-20	GND	Ground	GND

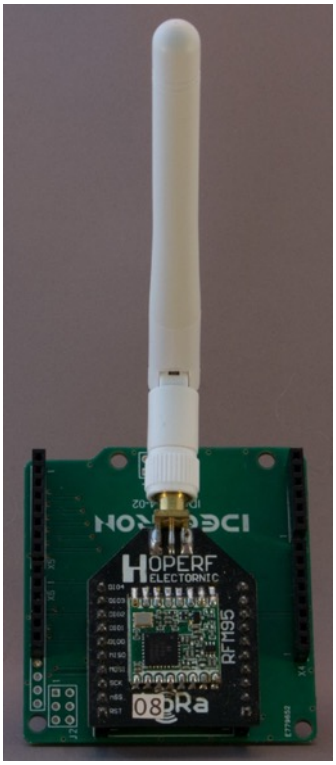
278.3. Libraries and sample HopeRF Adapter board for RFM95W

I have never used this without some kind of shield, so check the appropriate chapters for libraries and a sample sketch.

Check for a description of an appropriate shields on which you can place this adapter board at the next chapter.

279. LoRa: Shield for HopeRF board





This shield was designed by Bart Hiddink and Geert den Hartog from Ideetron.nl to connect the HopeRF adapter board on an Arduino Uno. It includes a Level shifter on all RFM inputs.

279.1. Specifications LoRa: Shield for HopeRF board

- Onboard 5V-3.3V level shifter 74HC4050M, connected to all RFM inputs (MOSI. SCK. nSS, and RST).

279.2. Datasheet LoRa: Shield for HopeRF board

- http://webshop.ideetron.nl/Files/3/1000/1211/Attachments/Product/iLE63ZN11M974649LW8SLY602R4u73YP.pdf

279.3. Connections Shield for HopeRF RFM Adapter boards

Pin nr	Name	Description	Arduino pin
1	DIO4	Digital I/O software configured	D6
2	DIO3	Digital I/O software configured	D5
3	DIO2	Digital I/O software configured	D4
4	DIO1	Digital I/O software configured	D3
5	DIO0	Digital I/O software configured	D2
6	MISO	SPI Data output	D12
7	MOSI	SPI Data input	D11 through onboard 5V-3.3V level shifter
8	SCK	SPI Clock input	D13 through onboard 5V-3.3V level shifter
9	nSS	SPI Chip Select input	D10 through onboard 5V-3.3V level shifter
10	RST	Reset trigger input	D8 through onboard 5V-3.3V level shifter
11-13	3.3V	Power Supply	3.3V
14-16	GND	Ground	GND
17	3.3V	Power Supply	3.3V
18-20	GND	Ground	GND

279.4. Libraries needed for LoRa: Shield for HopeRF board

- IBM LMIC (LoraMAC-in-C) modified by Matthijs Kooijman.
  https://github.com/matthijskooijman/arduino-lmic

Library use explanation

```
static const PROGMEM u1_t NWKSKEY[16] =  { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the Network Session Key from your application at The Things Network.

```
static const u1_t PROGMEM APPSKEY[16] = { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the App Session Key from your application at The Things Network.

```
static const u4_t DEVADDR = 0x..... ;
```

Copy the HEX value of the Dev Address from your application at The Things Network.

```
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    .dio = {2, 3, 4},
};
```

These values are specifically for the Shield for HopeRF RFM Adapter boards.

```
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),BAND_CENTI);
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
…
…
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK), BAND_MILLI);
```

This will configure your node to change channels for every transmission. If you are using a Single Channel Gateway it's advised to remove all other channels from your code otherwise you will only receive one payload in every 9 transmissions. Very often only channel 0 on frequency 868100000 kHz (=868.1 MHz) will be used in that case. Make sure your transmission frequencies matches the off your Gateway.

279.5. Sample LoRa: Shield for HopeRF board

Copy the sample sketch ***ttn_abp.ino*** from the LMIC library from Matthijs Kooijman and edit the lines that are listed in the code below.

- For more information about the values for NwsKey, AppsKey, DevAddr and the frequency to be used, check the chapter "276 Configuration of applications and devices at TTN".
- For more information about processing the data transmitted by your node, check the next section "The Things Network Data Handling".

Sample Connections

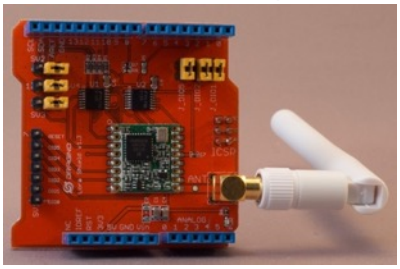
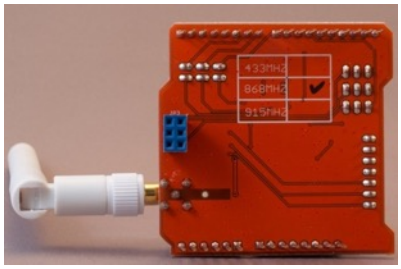
- Place the HopeRF board on top of the shield.

141_Lora_HopeRf.ino

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

static const PROGMEM u1_t NWKSKEY[16] = { 0xB6, 0x61, 0xF4, 0x35, 0x3E,
0x2C, 0x5D, 0x13, 0x91, 0x6E, 0x93, 0x3C, 0xB9, 0x1B, 0xEF, 0x0C };

static const u1_t PROGMEM APPSKEY[16] = { 0x44, 0x09, 0xB2, 0xA0, 0xF9,
0x95, 0x5B, 0xB6, 0x76, 0xEB, 0xF1, 0xC9, 0xE7, 0x73, 0x72, 0xD9 };

static const u4_t DEVADDR = 0x260117F3;
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

const unsigned TX_INTERVAL = 60;

const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    .dio = {2, 3, 4},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              Serial.println(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_RXCOMPLETE:
```

```
            Serial.println(F("EV_RXCOMPLETE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j){
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));
    os_init();
    LMIC_reset();
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);

    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(1, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(2, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band

//    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
BAND_MILLI);      // g2-band
    LMIC_setLinkCheckMode(0);
    LMIC.dn2Dr = DR_SF9;
    LMIC_setDrTxpow(DR_SF7,14);
    do_send(&sendjob);
}

void loop() {
    os_runloop_once();
}
```

280. LoRa: Dragino-shield

This shield was designed around the RFM95W modules

280.1. Specifications for Lora: Dragino-shield

- Compatible with 3.3v or 5v Arduino boards Leonardo, Uno, Mega and DUE.
- Frequency Band pre-configured in factory (433, 868, 915 MHz).
- Low power consumption.
- External Antenna via I-Pex connector.

280.2. Datasheet for Lora: Dragino-shield

- http://www.instructables.com/id/Use-Lora-Shield-and-RPi-to-Build-a-LoRaWAN-Gateway/
- http://wiki.dragino.com/index.php?title=Lora_Shield

280.3. Connections for Dragino Lora-shield

Pin nr	Name	Description	Arduino pin
	DIO4	Digital I/O software configured	?
	DIO3	Digital I/O software configured	?
	DIO2	Digital I/O software configured	D6
	DIO1	Digital I/O software configured	D5
	DIO0	Digital I/O software configured	D2
	MISO	SPI Data output	D12
	MOSI	SPI Data input	D11 through onboard 5V-3.3V level shifter
	SCK	SPI Clock input	D13 through onboard 5V-3.3V level shifter
	nSS	SPI Chip Select input	D10 through onboard 5V-3.3V level shifter
	RST	Reset trigger input	D9 through onboard 5V-3.3V level shifter
	3.3V	Power Supply	3.3V

	GND	Ground	GND
	3.3V	Power Supply	3.3V
	GND	Ground	GND

280.4. Libraries needed for Lora: Dragino-shield 868 MHz

- IBM LMIC (LoraMAC-in-C) modified by Matthijs Kooijman.
  https://github.com/matthijskooijman/arduino-lmic

Library use explanation

```
static const PROGMEM u1_t NWKSKEY[16] =  { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the Network Session Key from your application at The Things Network.

```
static const u1_t PROGMEM APPSKEY[16] = { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the App Session Key from your application at The Things Network.

```
static const u4_t DEVADDR = 0x..... ;
```

Copy the HEX value of the Dev Address from your application at The Things Network.

```
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 5, 6},
};
```

These values are specifically for the Shield for Dragino-shield 868 MHz

```
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),BAND_CENTI);
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
…
…
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK), BAND_MILLI);
```

This will configure your node to change channels for every transmission. If you are using a Single Channel Gateway it's advised to remove all other channels from your code otherwise you will only receive one payload in every 9 transmissions. Very often only channel 0 on frequency 868100000 kHz (=868.1 MHz) will be used in that case. Make sure your transmission frequencies matches the off your Gateway.

280.5. Sample Lora: Dragino-shield

Copy the sample sketch ***ttn_abp.ino*** from the LMIC library from Matthijs Kooijman and edit the lines that are listed in the code below.

- For more information about the values for NwsKey, AppsKey, DevAddr and the frequency to be used, check the chapter "276 Configuration of applications and devices at TTN".
- For more information about processing the data transmitted by your node, check the next section "The Things Network Data Handling".

Sample Connections

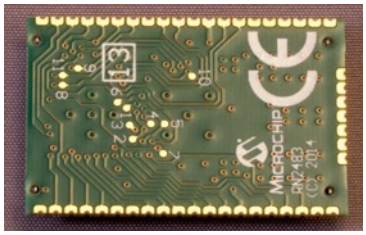
- Place the Dragino Lora-shield on top of the Arduino.

140_LoraDragino.ino

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

static const PROGMEM u1_t NWKSKEY[16] = { 0xB6, 0x61, 0xF4, 0x35, 0x3E,
0x2C, 0x5D, 0x13, 0x91, 0x6E, 0x93, 0x3C, 0xB9, 0x1B, 0xEF, 0x0C };

static const u1_t PROGMEM APPSKEY[16] = { 0x44, 0x09, 0xB2, 0xA0, 0xF9,
0x95, 0x5B, 0xB6, 0x76, 0xEB, 0xF1, 0xC9, 0xE7, 0x73, 0x72, 0xD9 };

static const u4_t DEVADDR = 0x260117F3;
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

const unsigned TX_INTERVAL = 60;

const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 5, 6},
};

void onEvent (ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_TXCOMPLETE:
            Serial.println(F("EV_TXCOMPLETE (includes waiting for RX
windows)"));
            if (LMIC.txrxFlags & TXRX_ACK)
              Serial.println(F("Received ack"));
            if (LMIC.dataLen) {
              Serial.println(F("Received "));
              Serial.println(LMIC.dataLen);
              Serial.println(F(" bytes of payload"));
            }
            os_setTimedCallback(&sendjob,
os_getTime()+sec2osticks(TX_INTERVAL), do_send);
            break;
        case EV_RXCOMPLETE:
```

```
            Serial.println(F("EV_RXCOMPLETE"));
            break;
         default:
            Serial.println(F("Unknown event"));
            break;
    }
}

void do_send(osjob_t* j){
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        Serial.println(F("Packet queued"));
    }
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting"));
    os_init();
    LMIC_reset();
    uint8_t appskey[sizeof(APPSKEY)];
    uint8_t nwkskey[sizeof(NWKSKEY)];
    memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
    memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
    LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);

    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(1, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
    LMIC_setupChannel(2, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band

//    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
//    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
BAND_MILLI);      // g2-band
    LMIC_setLinkCheckMode(0);
    LMIC.dn2Dr = DR_SF9;
    LMIC_setDrTxpow(DR_SF7,14);
    do_send(&sendjob);
}

void loop() {
    os_runloop_once();
}
```

281. LoRa: RN2483 module



281.1. Specifications LoRa: RN2483 module

- On-board LoRaWAN Class A protocol stack
- Low-Power Long Range transceiver in the 433 and 868 MHz frequency bands
- >5 km coverage at urban area
- >15 km coverage at suburban area
- Operating voltage 2.1 to 3.6V (typical 3.3V)
- ASCII command interface over UART
- Device Firmware Upgrade (DFU) over UART (SPI)
    - 57600 bps, 8 bits, no parity, 1 stop bit, no hardware flow control
- 14 GPIO for control, status and ADC
- SMT pads, so not breadboard friendly, but since you only need 5 of them, you could solder stranded wires to these pads (UART_RX, UART_TX, not-RESET, VDD and GND)
- 50 ohm impedance for Antenna, SMA connector to RFL (433 MHz) or RFH (868 MHz) and GND
- For short ranges you can use a solid wire:
    - 433 MHz to RFL (25)
        - 1/4 wave = 164.7mm
        - 1/2 wave = 329.4mm
        - full wave = 692.7mm
    - 868 MHz to RFH (23)
        - 1/4 wave = 82.2mm
        - 1/2 wave = 164.3mm
        - full wave = 345.5mm
        -

The RN2483 firmware contains the complete LoRaWAN protocol stack. Because of this, sketches can be very small, but on the other hand it is not possible to bypass the rules in the LoRaWAN description, making it impossible to build a Single Channel Node to communicate with your own Single Channel Gateway. You can limit the number of channels to three, so working with a Single Channel Gateway is still possible, but not ideal.

281.2. Datasheet LoRa: RN2483 module

- It is difficult to find a shop where they sell these modules, so I included the following search tool link (can also be used for finding other chips): http://www.findchips.com/search/rn2483
- A nice "How to get started" tutorial can be found at: https://www.thethingsnetwork.org/forum/t/how-to-build-your-first-ttn-node-arduino-rn2483/1574

- De datasheet for the RN2483 can be found at: http://www.farnell.com/datasheets/1947946.pdf?_ga=1.80085719.1148387527.1482060915
- Command reference guide: http://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf
- Several other interesting documents can be found at: http://www.microchip.com/wwwproducts/en/RN2483

281.3. Connections LoRa: RN2483 module

Only 5 pins are needed to communicate between an Arduino and the RN2483.

Pin nr	Name	Description	Arduino pin
1	GND	Ground	-
2	UART_RTS	UART RTS signal	-
3	UART_CTS	UART CTS signal	-
4..5	RESERVED	DO NOT CONNECT	-
6	UART_TX	UART Transmit (Tx)	Any digital pin (software serial Rx)
7	UART_RX	UART Receive (Rx)	Any digital pin (software serial Tx)
8	GND	Ground	-
9..10	GPIO13..12	General Purpose I/O	-
11	GND	Ground	-
12	VDD	3.3V Power supply	-
13..14	GPIO10..11	General Purpose I/O	-
15..19	NC	-	-
20..22	GND	Ground	-
23	RFH	RF Analog for high band Antenna for 868 MHz	-
24	GND	Ground	-
25	RFL	RF analog for low band Antenna for 433 MHz	-
26..28	GND	Ground	-
29	NC	-	-
30..31	TESTx	DO NOT CONNECT	-
32	not RESET	Active low Reset input	Any digital pin
33	GND	Ground	GND
34	VDD	3.3V Power supply	3.3V
35..40	GPIO00..5	General Purpose I/O	-
41	GND	Ground	-
42	NC	-	-
43..46	GPIO6..9	General Purpose I/O	-
47	GND	Ground	-

281.4. Libraries needed for LoRa: RN2483 module

- The rn2xx3 library from JP Meijers through the Library Manager.

Library use explanation

```
#include <rn2xx3.h>
```
Include the rn2xxr library from JP Meijers.

```
#include <SoftwareSerial.h>
```
Include SoftwareSerial for the communication between Arduino and the RN2483.

```
SoftwareSerial Serial1(RX_pin, TX_pin);
```
Initialize SoftwareSerial with Rx_pin being the pin to which UART_TX (6) is connected and Tx_pin being the pin to which UART_RX (7) is connected.

```
rn2xx3 myLora(mySerial);
```

```
Serial.begin(57600); //serial port to computer
```
Initialize the hardware Serial at 57600 bps.

```
mySerial.begin(9600); //serial port to RN
```
Initialize the SoftwareSerial at 9600 bps.

```
pinMode(12, OUTPUT);
digitalWrite(12, LOW);
delay(500);
digitalWrite(12, HIGH);
delay(100);
```
This will initialize the RN2483 module.

```
mySerial.flush();
```
Flush all buffers from mySerial.

```
myLora.autobaud();
```
Set the baud rate of myLora automatically.

```
Serial.println(myLora.sysver());
```
Print the firmware version of the RN2483 module

```
const char *devAddr = "..";
const char *nwkSKey = "..";
const char *appSKey = "..";
```
You can copy this 3 lines from your device at the section EXAMPLE CODE.

```
join_result = myLora.initABP(devAddr, appSKey, nwkSKey);
```
Set connect settings to the chosen devAddr, appsKey and nwkSKey.

```
byte payload[]="Hello";
```
This defines the payload to be send.

```
myLora.txBytes(payload, sizeof(payload));
```
Sends the payload without asking for an acknowledgement.

281.5. Sample LoRa: RN2483 module

The following sketch sends 1 byte of data to an application at The Things Network. Edit the devAddr, nwkSkey and appSkey lines in the code below.

- For more information about the values for NwsKey, AppsKey and DevAddr to be used, check chapter "276 Configuration of applications and devices at TTN".
- For more information about processing the data transmitted by your node, check the next section "The Things Network Data Handling".

Sample Connections

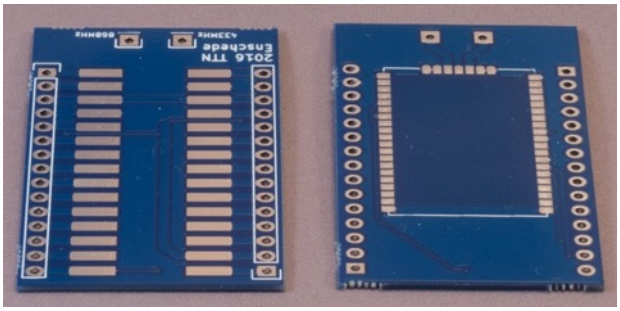
- Connect UART_TX (6) to D10
- Connect UART_RX (7) to D11
- Connect not-RESET (32) to D12
- Connect GND (33) to GND.
- Connect VDD (34) to 3.3V.

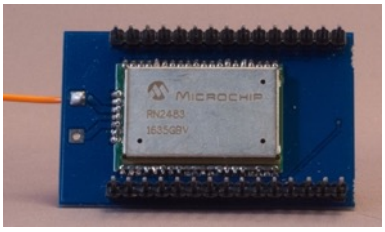
126_LoRa_RN2483.ino

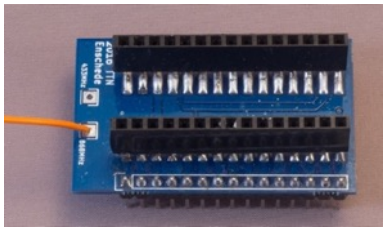
```
#include <rn2xx3.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

rn2xx3 myLora(mySerial);

const int LED_PIN = 13;

void setup()
{
  pinMode(13, OUTPUT);
  digitalWrite(LED_PIN, 1);
  Serial.begin(57600); //serial port to computer
  mySerial.begin(9600); //serial port to radio
  Serial.println("Startup");
  initialize_radio();
  digitalWrite(LED_PIN, 0);
  delay(2000);
}

void initialize_radio()
{
  pinMode(12, OUTPUT);
  digitalWrite(12, LOW);
  delay(500);
  digitalWrite(12, HIGH);
  delay(100); //wait for the RN2xx3's startup message
  mySerial.flush();
  myLora.autobaud();
  String hweui = myLora.hweui();
  Serial.println(myLora.hweui());
  Serial.println("RN2xx3 firmware version:");
  Serial.println(myLora.sysver());
  Serial.println("Trying to join TTN");
  bool join_result = false;
  const char *devAddr = "260117F3";
  const char *appSKey = "4409B2A0F9955BB676EBF1C9E77372D9";
  const char *nwkSKey = "B661F4353E2C5D13916E933CB91BEF0C";
  join_result = myLora.initABP(devAddr, appSKey, nwkSKey);
```

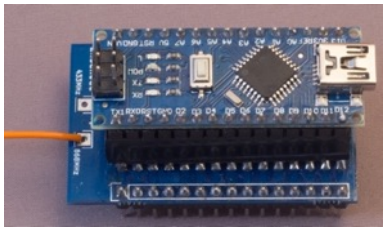
```
  Serial.print("devAddr=");
  Serial.println(devAddr);
}

void loop()
{
  digitalWrite(LED_PIN, 1);
  Serial.println("TXing");
  myLora.tx("AB");
  //  byte payload[] = "Hello";
  //  myLora.txBytes(payload, sizeof(payload));
  digitalWrite(LED_PIN, 0);
  delay(200);
}
```

282. LoRa: RN2483 Enschede Nano breakout board









This board was designed by JM Meijers and was used in May 2016 in Enschede The Netherlands at a LoRa meeting.

282.1. Specifications LoRa: RN2483 Enschede Nano breakout board

You can use this board as a breakout board for the following pins:

- UART_TX (6)
- UART_RX (7)
- not_RESET (32)
- GND (33)
- VDD (34)

This board was designed work with an Arduino Nano soldered on top (surface mount).

282.2. Ordering LoRa: RN2483 Enschede Nano breakout board

You can order the PCB directly at OSH Park in the USA:

- https://oshpark.com/shared_projects/yhetNM7W

The assembled Nano breakout can be ordered at tindie.com. This means a breakout board with the RN2483. The Nano and the header pins are not included. They will be soldered and delivered by JM Meijers himself:

- https://www.tindie.com/products/jpmeijers/rn2483-ttn-enschede/?pt=full_prod_search

JM Meijers has also developed an Arduino UNO shield. That PCB can also be ordered at OSH Park: https://oshpark.com/shared_projects/8QkQGKwY

282.3. Connections LoRa: RN2483 Enschede Nano breakout board

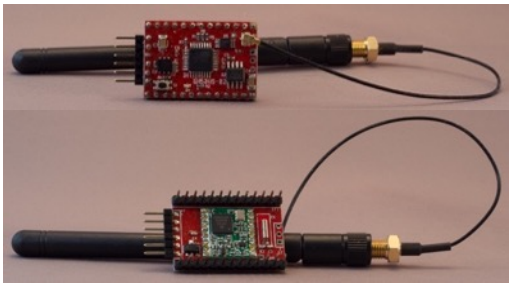
Pin nr breakout	Name RN2483 pin	Description	Arduino NANO pin
2	VDD (34)	3.3V Power	3.3V
29	GND (33)	Ground	GND
58	UART_TX(6)	Transmit pin	D10
59	UART_RX(7)	Receive pin	D11
28	not_RESET (32)	Low-active reset pin	D12

The other pins are not connected to the RN2483 but are connected to the pins on the Arduino Nano. Pin 1 on the breakout board is connected to pin 1 on the Arduino Nano, pin 2 to pin 2 etc….

282.4. Sample LoRa: RN2483 module

With an Arduino Nano, the connection on this breakout board are the same as within the sample of the previous chapter, so you can use the same sample sketch as in the previous chapter.

283. LoRa: Nexus as a TTN node



This specialized board was developed by Ideetron in The Netherlands and is based on an Arduino Mini and a RFM95W or RFM98W LoRa module. More information of this specialized Arduino board, can be found in the corresponding chapter in the Arduino Boards section. More information about the RFM95W module can be found in one of the next chapters.

283.1. Library needed to use Nexus as an TTN node

- IBM LMIC (LoraMAC-in-C) modified by Matthijs Kooijman.
  https://github.com/matthijskooijman/arduino-lmic

Library use explanation

```
static const PROGMEM u1_t NWKSKEY[16] =  { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the Network Session Key from your application at The Things Network.

```
static const u1_t PROGMEM APPSKEY[16] = { 0x.., 0x.., 0x.., 0x.., 0x..,
0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.., 0x.. };
```

Copy the MSB value of the App Session Key from your application at The Things Network.

```
static const u4_t DEVADDR = 0x..... ;
```

Copy the HEX value of the Dev Address from your application at The Things Network.

```
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    .dio = {4, 5, 7},
};
```

These values are specifically for the Nexus.

```
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),BAND_CENTI);
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
…
…
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK), BAND_MILLI);
```

This will configure your node to change channels for every transmission. If you are using a Single Channel Gateway it's advised to remove all other channels from your code otherwise you will only receive one payload in every 9 transmissions. Very often only channel 0 on frequency 868100000 kHz (=868.1 MHz) will be used in that case. Make sure your transmission frequencies matches the off your Gateway.

283.2. Sample Nexus

Copy the sample sketch ***ttn_abp.ino*** from the LMIC library from Matthijs Kooijman and edit the lines that are listed in the code below.

- For more information about the values for NwsKey, AppsKey, DevAddr and the frequency to be used, check the chapter "276 Configuration of applications and devices at TTN".
- For more information about processing the data transmitted by your node, check the next section "The Things Network Data Handling".

Sample Connections

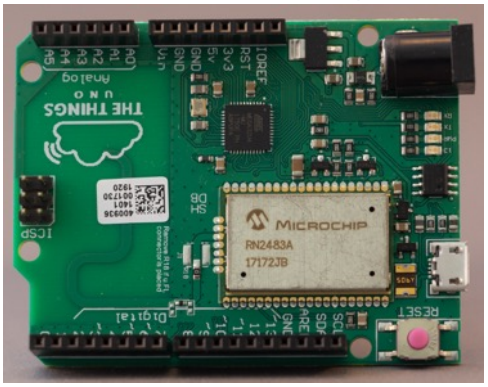
- Since the Nexus has an presoldered RFM95W module, no connections are needed for this sample.

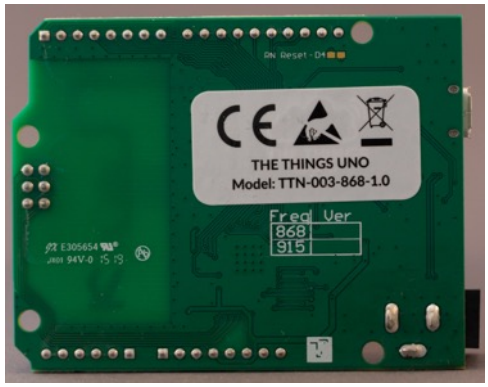
142_Lora_Nexus.ino

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

static const PROGMEM u1_t NWKSKEY[16] = { 0xB6, 0x61, 0xF4, 0x35, 0x3E,
0x2C, 0x5D, 0x13, 0x91, 0x6E, 0x93, 0x3C, 0xB9, 0x1B, 0xEF, 0x0C };

static const u1_t PROGMEM APPSKEY[16] = { 0x44, 0x09, 0xB2, 0xA0, 0xF9,
0x95, 0x5B, 0xB6, 0x76, 0xEB, 0xF1, 0xC9, 0xE7, 0x73, 0x72, 0xD9 };

static const u4_t DEVADDR = 0x260117F3;
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

const unsigned TX_INTERVAL = 60;

const lmic_pinmap lmic_pins = {
  .nss = 10,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = LMIC_UNUSED_PIN,
  .dio = {4, 5, 7},
};

void onEvent (ev_t ev) {
  Serial.print(os_getTime());
  Serial.print(": ");
  switch (ev) {
    case EV_TXCOMPLETE:
      Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
      if (LMIC.txrxFlags & TXRX_ACK)
        Serial.println(F("Received ack"));
      if (LMIC.dataLen) {
        Serial.println(F("Received "));
        Serial.println(LMIC.dataLen);
        Serial.println(F(" bytes of payload"));
      }
      os_setTimedCallback(&sendjob, os_getTime() +
sec2osticks(TX_INTERVAL), do_send);
      break;
    case EV_RXCOMPLETE:
```

```
      Serial.println(F("EV_RXCOMPLETE"));
      break;
    default:
      Serial.println(F("Unknown event"));
      break;
  }
}

void do_send(osjob_t* j) {
  if (LMIC.opmode & OP_TXRXPEND) {
    Serial.println(F("OP_TXRXPEND, not sending"));
  } else {
    LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 0);
    Serial.println(F("Packet queued"));
  }
}

void setup() {
  Serial.begin(115200);
  Serial.println(F("Starting"));
  os_init();
  LMIC_reset();
  uint8_t appskey[sizeof(APPSKEY)];
  uint8_t nwkskey[sizeof(NWKSKEY)];
  memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
  memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
  LMIC_setSession (0x1, DEVADDR, nwkskey, appskey);

  LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  LMIC_setupChannel(1, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  LMIC_setupChannel(2, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band

  //    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),
BAND_CENTI);      // g-band
  //    LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,  DR_FSK),
BAND_MILLI);      // g2-band
  LMIC_setLinkCheckMode(0);
  LMIC.dn2Dr = DR_SF9;
  LMIC_setDrTxpow(DR_SF7, 14);
  do_send(&sendjob);
}

void loop() {
  os_runloop_once();
}
```

284. Lora: The Things Uno as TTN node





The Things Uno is based on the Arduino Leonardo (not the Arduino Uno), with an added Microchip LoRaWAN module (RN2483A). More of this specialized Arduino board can be found in the corresponding chapter in the Arduino Boards section. More information about the RN2483A module can be found in one of the next chapters.

284.1. Library needed to use The Things Uno as TTN node

- TheThingsNetwork library from The Things Network, through the Library Manager.

Library use explanation

```
#include <TheThingsNetwork.h>
```

Include the TheThingsNetwork from The Things Network.

```
#define loraSerial Serial1
```

The name 'loraSerial' will be used to point to UART Serial1.

```
#define debugSerial Serial
```

The name 'debugSerial' will be used to point to UART Serial

```
#define freqPlan TTN_FP_EU868
```

Set the frequency to 868MHz (Europe).

```
const char *appEui = "....";
const char *appKey = "....";
```

These values must be obtained from the device overview in your application at The Things Network.

```
TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);
```

Make an instance called ttn from the class TheThingsNetwork, use loraSerial for the connection to the radio, debugSerial for output to Serial Monitor and freqPlan as the frequency-plan you are going to use.

```
loraSerial.begin(57600);
```

Initialize the connection to the radio at 57600 bps.

```
debugSerial.begin(9600);
```

Initialize the connection with Serial Monitor at 9600 bps.

```
delay(3000);
```

Wait some time to make sure debugSerial (Serial) is up and running.

```
ttn.showStatus();
```
Show some settings of your radio (such as EUI, Battery, AppEUI and DevEUI).

```
ttn.join(appEui, appKey);
```
Join with The Things Network.

```
byte payload[1];
```
Define length of the payload to be sent.

```
payload[0] = map(analogRead(A0),0,1023,0,255);
```
Set the payload to the converted value at A0 (map value at A0 from 0-1023 to 0-255).

```
ttn.sendBytes(payload, sizeof(payload));
```
Send the payload to The Things Network.

```
delay(60000);
```
Wait for 60 seconds before sending the next payload.

284.2. Sample The Things Uno

The following sketch reads the value at A0, converts it to a 1 byte value and0 sends it to The Things Network.

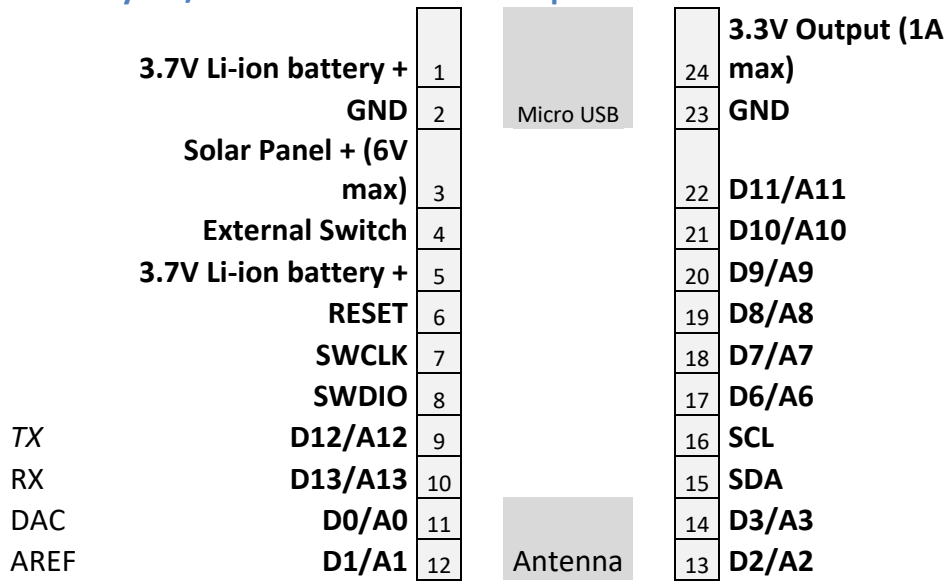
173_TheThingsUno.ino

```
#include <TheThingsNetwork.h>

#define loraSerial Serial1
#define debugSerial Serial
#define freqPlan TTN_FP_EU868

const char *appEui = "70B3D57ED0027053";
const char *appKey = "F8DBA86EC0380258A959DF7BD8826249";

TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);

void setup()
{
  loraSerial.begin(57600);
  debugSerial.begin(9600);
  delay(3000);
  debugSerial.println("-- STATUS");
  ttn.showStatus();
  debugSerial.println("-- JOIN");
  ttn.join(appEui, appKey);
}

void loop()
{
  debugSerial.println("-- LOOP");
  byte payload[1];
  byte payload[1];
  payload[0] = map(analogRead(A0),0,1023,0,255);
  ttn.sendBytes(payload, sizeof(payload));
  delay(60000);
}
```

285. LoRa: Sodaq One as a TTN node

This board is based on Sodaq's Autonomo 32 bits Arduino compatible platform. More information of this specialized Arduino board, can be found in the corresponding chapter in the Arduino Boards section.

285.1. Layout/connections LoRa: Sodaq One



Next these external pins, the Sodaq One has 8 extra internal I/O ports

Name	Internal Pin/Port	Definition in Arduino IDE	Description
Red LED	D14 *OUTPUT*	LED_RED	Set this pin high to turn on the RED LED
Green LED	D15 *OUTPUT*	LED_GREEN	Set this pin high to turn on the Green LED
Blue LED	D16 *OUTPUT*	LED_BLUE	Set this pin high to turn on the Blue LED
GPS Time pulse	D17 *INPUT*	GPS_TIMEPULSE	
GPS Switch	D18 *OUTPUT*	GPS_ENABLE	Set this pin high to turn on the GPS module
User Button	D19 *INPUT*	BUTTON	High when not pressed, low when pressed
Power Enable	D22 *OUTPUT*	ENABLE_PIN_IO	
External Switch	D23 *INPUT*	SWITCH_SENSE	Used to sense the position of an external switch connected between the External Switch pin and the battery.

285.2. Sample Sodaq One: Tracker Software

Sodaq has developed Tracker Software for the Sodaq One. This software is preinstalled on the Sodaq One.

The latest version of the Sodaq One Universal Tracker, can be found at:
https://github.com/SodaqMoja/SodaqOne-UniversalTracker

163_SodaqOneTracker.ino

Version 3.0 of this Tracker software is also in my samples folder.

Version 3.0 has the following functionality:

- Customizable parameters (see table)
- ABP or OTAA
- Sending the following information to your Application at TTN
    - Battery
    - Course
    - Epoch
    - Latitude. Longitude
    - Time to GPS Fix
    - Number of satellites
    - Speed
    - Board temperature

Parameter	Command	Default value
GPS	gps=	1
Fix Interval (min)	fi=	15
Alt. Fix Interval (min)	afi=	0
Alt. Fix From (HH)	afhh=	0
Alt. Fix From (MM)	afmm=	0
Alt. Fix To (HH)	afth=	0
Alt. Fix To (MM)	aftm=	0
GPS Fix Timeout (sec)	gft=	120
Minimum satellite count	sat=	4
OTAA mode (1= OTAA, 0=ABP)	otaa=	0
Retry connection	retry=	0
Automatic Data Rate (ADR)	adr=	1
Acknowledgements (ACK)	ack=	0
Spreading Factor	sf=	7
Output Index	pwr=	1
DevAddr / DevEUI	dev=	0004A30B001A1FD4
AppSKey / AppEUI	app=	000……..000
NWSKey / AppKey	key=	000……..000
Number of coordinates to upload	num=	1
Repeat count	rep=	0
Status LED	led=	0
Debug mode	dbg=	0

Working with the Universal Tracker

This paragraph describes how to get started with the Sodaq Universal Tracker 3.0.

Prerequisites Universal Tracker

You need the following:

- Arduino IDE for Serial Monitor
- Sodaq One and USB cable
- Device parameters ABP or OTAA
    - ABP parameters
        - DevAddr
        - AppSKey
        - NWSKey
    - OTAA parameters
        - DevEUI
        - AppEUI
        - AppKey

Initialize Universal Tracker

- After connecting the Sodaq, you will only have 30 seconds to enter your ABP or OTAA parameters, so it is a good idea to prepare a text file with the following commands, so you can copy/paste them to the Serial Monitor:
- ABP

```
otaa=0
dev=<DevAddr>
app=<AppsKey>
key=<<NWSKey>
fi=1
rep=5
led=1
dbg=0
```

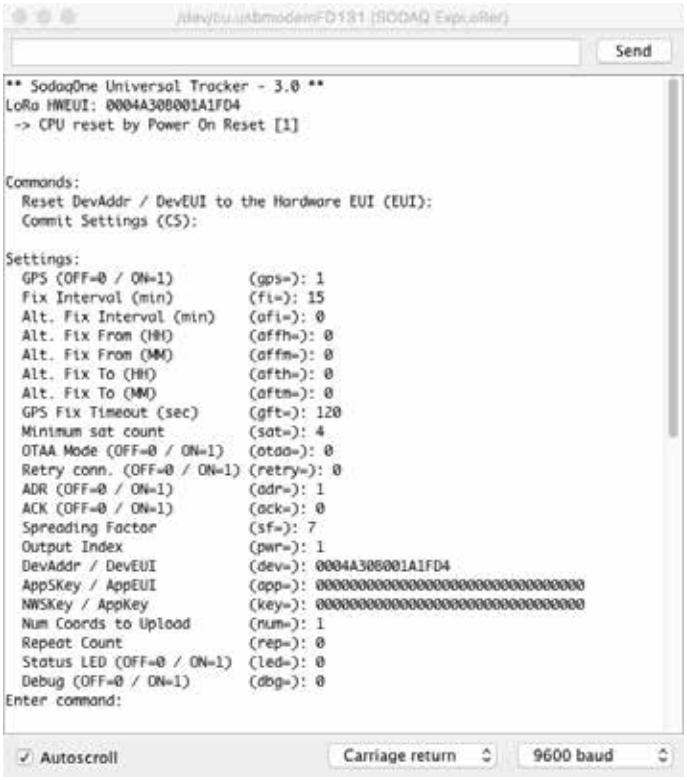
- OTAA

```
otaa=1
dev=<DevEUI>
app=<AppEUI>
key=<<AppKey>
fi=1
rep=5
led=1
dbg=0
```

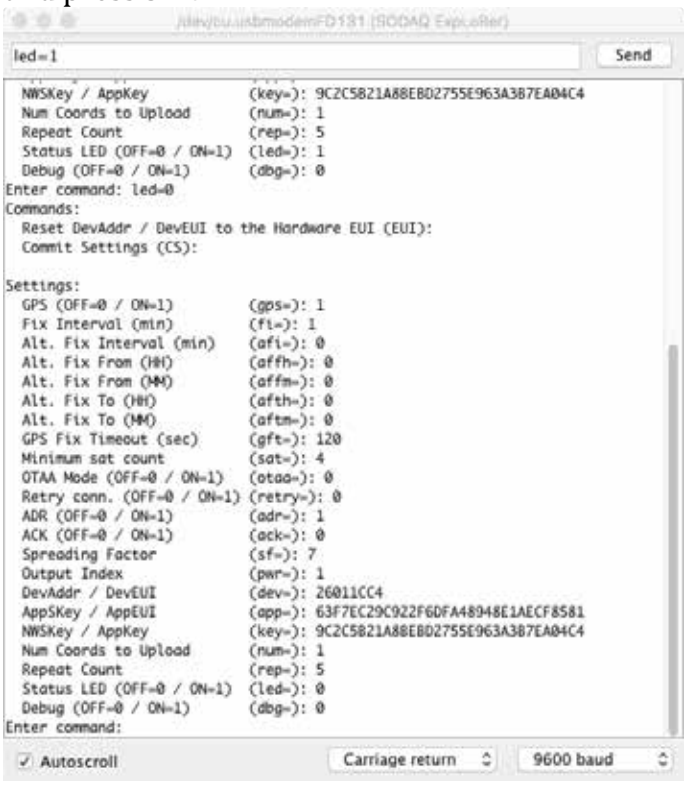
  I've not been able to test this, because I'm not within reach of a Full channel Gateway and my Single Channel Gateway does not support Download links. The latter is needed for OTAA authentication.

- Start the Arduino IDE.
- Connect the Sodaq One with your computer and immediately open the Serial Monitor.
  During the first 30 seconds after being connected, the Boot menu will be shown. After those first 30 seconds, USB will be disabled to safe energy.

- After opening Serial Monitor, you should see something like this:

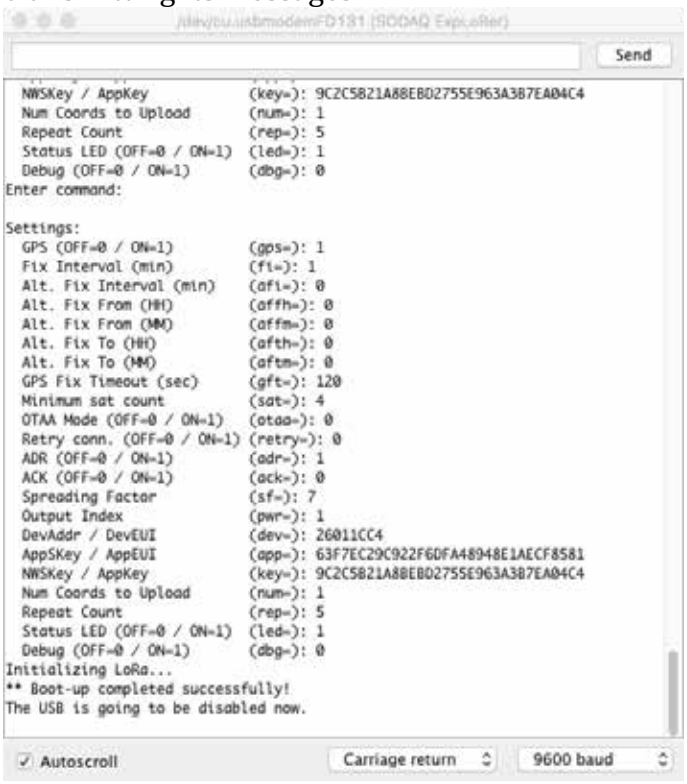


- One by one copy the commando's from your text file, paste them in the SEND box and press SEND.



- The Boot Menu will respond by displaying the changed boot menu, directly after you've pressed SEND.

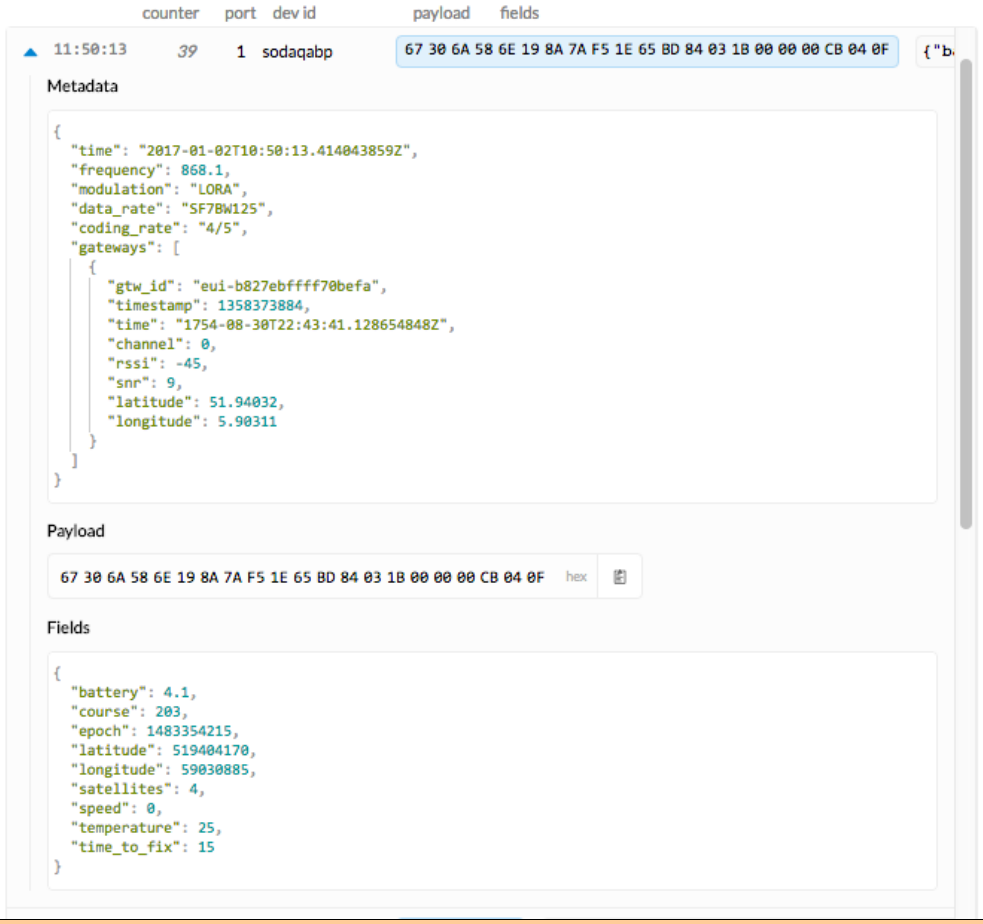
- After 30 seconds the USB port will be disabled and the Sodaq will start transmitting its messages.



Data Handling Universal Tracker

You can use the following Decoder Payload functions to decode the data received from your Sodaq Universal Tracker. Check the next section how to use this function.

```
function Decoder(bytes, port)
{
  var epoch = (bytes[3] << 24) | (bytes[2] << 16) | (bytes[1] << 8) | bytes[0];
  var batt = (3000+10*bytes[4])/1000;
  var temp = bytes[5];
  var lat = (bytes[9] << 24) | (bytes[8] << 16) | (bytes[7] << 8) | bytes[6];
  var lon = (bytes[13] << 24) | (bytes[12] << 16) | (bytes[11] << 8) | bytes[10];
  var alt = (bytes[15] << 8) | bytes[14];
  var speed = (bytes[17] << 8) | bytes[16];
  var course = bytes[18];
  var sats = bytes[19];
  var ttf = bytes[20];
  return {
    course: course,
    satellites: sats,
    time_to_fix: ttf,
    latitude: lat,
    longitude: lon,
    epoch: epoch,
    battery: batt,
    speed: speed,
    temperature: temp
  };
}
```



286. LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI as a TTN Node

The Dragino LoRaWAN GPS Tracker LGT-92-LI is an open source GPS tracker based on Ultra Low Power STM32L072 and SX1278 LoRa module.

286.1. Specifications LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI as a TTN Node

- MCU: STM32L072CZT6
  - Flash: 192 kB
  - RAM: 20 kB
  - EEPROM: 6 KB
- GPS L70R
- 9-axis accelerometer MPU9250
  - 3-axis MEMS gyroscope
  - 3-axis MEMS accelerometer
  - 3-axis MEMS magnetometer
- Firmware 1.4
- LoRaWAN 1.0.3
- Motion sensing
- Power monitoring
- Tri-colored LED
- Alarm buttoon
- 1000 mA Li-on battery
- Charging circuit with USB port
- Power consumption
  - Sleeping mode: 77uA
  - Tracking: max 38 mA
  - LoRa transmit: 24-150 mA
- Weight: 55g
- Comes with a micro-USB to 4 wire dupont cable, so you can connect the Tracker to an USB-Serial adapter of your choice.

286.2. Layout/connections LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI as a TTN Node

To configure the preloaded software on your Dragino Tracker you must use the Micro-USB to dupont cable and a USB-serial adapter.

Description	Micro-USB to dupont	USB to Serial adapter
Transmit data from Tracker	Green	Rx
Receive data from PC	White	Tx
Ground	Black	GND
Vcc	Red	n.c. (not needed)

286.3. Documents LoRa: Dragino LoRaWAN GPS Tracker LGT-92-LI as a TTN Node

Datasheet:

- https://www.dragino.com/downloads/index.php?dir=LGT_92/&file=Datasheet_LGT-92.pdf

AT-Commands 1.4

- https://www.dragino.com/downloads/downloads/LGT_92/DRAGINO_LGT92_AT_Commands_v1.4.0.pdf

User manual 1.4

- https://www.dragino.com/downloads/index.php?dir=LGT_92/&file=LGT-92_LoRa_GPS_Tracker_UserManual_v1.4.3.pdf

Firmware

This tracker is preloaded with tracker software firmware. The latest version of this firmware can be found at:

- https://www.dragino.com/downloads/index.php?dir=LGT_92/firmware/

You'll need an "36 ST Link V2" to reprogram this Tracker.

286.4. Register device to TTN with the original keys

Before you create this device on The Things Network you'll need the keys that are printed on a label that came with the Tracker. It looks like:

I forgot to record the steps I took to create this device at The Things Network and since you can only use the preinstalled keys once, I can't create this device again. I've tried to reproduce the steps I took the first time, but I wasn't able to reproduce the finish it, because the keys

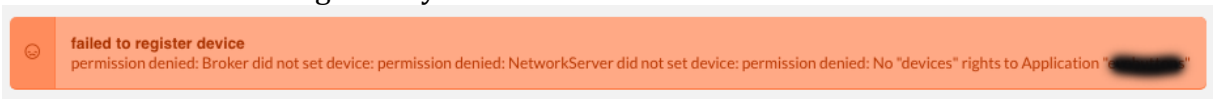
were already used before. I is possible though to recreate the device with different keys, this is described in the next paragraph.

Create an application at TTN

- Login to TTN, go to your console and choose for APPLICATIONS
- Click on ADD APPLICATION
- Create an application in Console Applications
- Type a name for your application at APPLICATION ID
    - You can only use lowercase characters (a..z) and score/underscore (-_) and can't start with score/underscore
    - This name should be unique for The Things Network and once used it can never be re-used again, not even after deleting this application.
    - After setting the Application ID, you can't change it anymore.
- Type a description ad DESCRIPTION
- Don't type anything at APPLICATION EUI, it will be generated by TTN. You can add the APP EUI listed on your Tracker's label in the application settings page later.
- Choose the correct handler at HANDLER REGISTRATION
- Now click on the ADD APPLICATION button, the application is now created, you will first have to add the APP EUI listed on your Tracker's label and then add your device to this new application.
- Click on MANAGE EUIS, right next to the header APPLICATION EUIS
- Now click on ADD EUI right next to the header EUIS and then click on the pencil and enter the APP EUI as listed on your Tracker's label.

Create a device for your application

- To register a device in this application, click on DEVICES and then click on REGISTER DEVICE
- Type a name for your device at DEVICE ID
    - You can only use lowercase characters (a..z) and score/underscore (-_) and can't start with score/underscore
    - This name should be unique for The Things Network and once used it can never be re-used again, not even after deleting this application.
    - After setting the Device ID, you can't change it anymore.
- Enter the device EUI from your Tracker's label at DEVICE EUI
- Click on the pencil at APP KEY and type the APP KEY from your Tracker's label.
- Last select the correct App EUI (the one from your Tracker's label).
- Finally click on REGISTER (this failed with me, because I had already registered this device with the original keys.



286.5. Register device to TTN with custom keys

I'm going to use this device with students and they need to register the tracker again and again. This is not possible with the original keys listed on the Tracker's label, so this paragraph describes how to register this Tracker with custom (not original) keys.

Create an application at TTN

- Login to TTN, go to your console and choose for APPLICATIONS

- Click on ADD APPLICATION
- Create an application in Console Applications
- Type a name for your application at APPLICATION ID
  - You can only use lowercase characters (a..z) and score/underscore (-_) and can't start with score/underscore
  - This name should be unique for The Things Network and once used it can never be re-used again, not even after deleting this application.
  - After setting the Application ID, you can't change it anymore.
- Type a description ad DESCRIPTION
- Don't type anything at APPLICATION EUI, it will be generated by TTN
- Choose the correct handler at HANDLER REGISTRATION
- Now click on the ADD APPLICATION button, the application is now created.

Create a device for your application

- To register a device in this application, click on DEVICES and then click on REGISTER DEVICE
- Type a name for your device at DEVICE ID
  - You can only use lowercase characters (a..z) and score/underscore (-_) and can't start with score/underscore
  - This name should be unique for The Things Network and once used it can never be re-used again, not even after deleting this application.
  - After setting the Device ID, you can't change it anymore.
- Click on the GENERATE button at DEVICE EUI
- Don't type anything at DEVICE EUI nor at APP KEY, those fields will be generated by TTN
- Finally click on REGISTER
- You device is now registered, but you'll tell the Tracker to use the same KEYs.

Change keys at your Tracker to match those at TTN

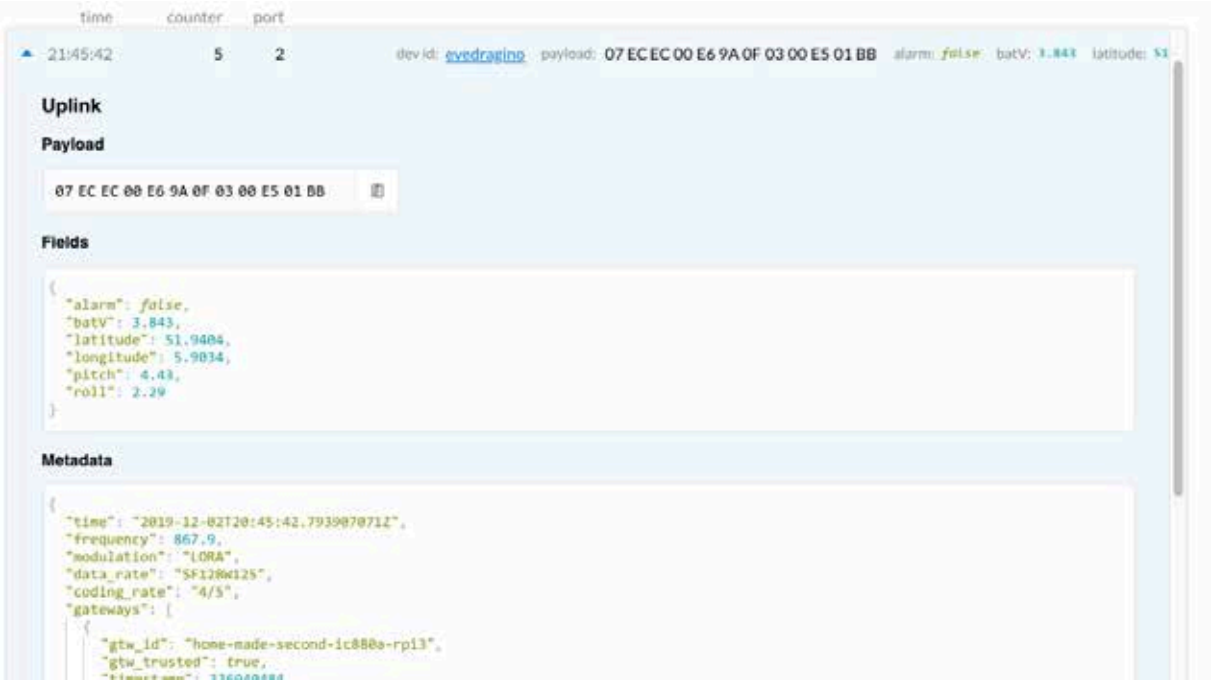
- Collect the following information at TTN:
  - Device EUI:
  - Application EUI:
  - App Key
- Connect the micro-USB to dupont cable between the Tracker and an USB-serial adapter (for example an FTDI)
  - connect the black dupont connector to GND on the USB-serial
  - connect white to Tx
  - connect green to Rx
  - don't connect the red dupont connector
- In Arduino IDE select the correct serial port and open Serial Monitor
- Choose 9600 bps and choose 'Both NL & CR'
- Type AT and click on SEND (or press ENTER)
- The tracker should respond with OK
- AT+FDR
- ATZ
- AT+DEUI= 00 19 06 4E 8D CC DF 9F
- AT+APPEUI= 70 B3 D5 7E D0 02 A9 94
- AT+APPKEY= 4D 50 6F 0C 54 16 AE B9 09 96 ED D6 E1 55 95 C3
- ATZ

286.6. Data Handling Universal Tracker

You can use the following Decoder Payload functions to decode the data received from your Dragino LoRaWAN GPS Tracker LGT-92-LI.

```
function Decoder(bytes, port)
{
    // Decode an uplink message from a buffer
    // (array) of bytes to an object of fields.

    //Alarm status
    var alarm=(bytes[6] & 0x40)?true:false;
    value=((bytes[6] & 0x3f) <<8) | bytes[7];

    //Battery,units:Volts
    var batV=value/1000;
    value=bytes[8]<<8 | bytes[9];
    if(bytes[8] & 0x80)
    {
        value |=0xFFFF0000;
    }

    //roll,units: °
    var roll=value/100;
    value=bytes[10]<<8 | bytes[11];
    if(bytes[10] & 0x80)
    {
        value |=0xFFFF0000;
    }
    //pitch,units: °
    var pitch=value/100;
    var json=
    {
        roll:roll,
        pitch:pitch,
        batV:batV,
        alarm:alarm
    };
    var value=bytes[0]<<16 | bytes[1]<<8 | bytes[2];
    if(bytes[0] & 0x80)
    {
        value |=0xFFFFFF000000;
    }
    var value2=bytes[3]<<16 | bytes[4]<<8 | bytes[5];
    if(bytes[3] & 0x80)
    {
        value2 |=0xFFFFFF000000;
    }
    if (value == 0x0FFFFF && value2 == 0x0FFFFF)
    {
        //gps disabled (low battery)
    }
    else if (value === 0 && value2 === 0)
    {
        //gps no position yet
    }
    else
    {
        json.latitude=value/10000;//gps latitude,units: °
        json.longitude=value2/10000;//gps longitude,units: °
    }
    return json;
```

```
}
```


time
counter
port
21:45:42
5
2
dev id: evedraginо
payload: 07 EC EC 00 E6 9A 0F 03 00 E5 01 BB
Uplink
Payload
07 EC EC 00 E6 9A 0F 03 00 E5 01 BB
Fields
{
"alarm": false,
"batV": 3.843,
"latitude": 51.9404,
"longitude": 5.9034,
"pitch": 4.43,
"roll": 2.29
}
Metadata
{
"time": "2019-12-02T20:45:42.793907071Z",
"frequency": 867.9,
"modulation": "LORA",
"data_rate": "SF12BW125",
"coding_rate": "4/5",
"gateways": [
{
"gtw_id": "home-made-second-ic880a-rpi3",
"gtw_trusted": true,

287. Grove Lora Radio 868 as a TTN Node (failure)

This module is a LoRa radio for long range, low power communication. There is a RFM95W LoRa 868 MHz radio module and a ATMega168 MCU. The ATMega168 MCU serves as an interface between your Arduino (serial connection) and the RFM95W LoRa radio module through SPI.

Note: I haven't found a library (yet) for this module to communicate with LoRaWAN networks like The Things Network!!

The Things Network Data Handling

There are several ways to check the data your nodes are sending to TTN. You must understand that the data arriving at TTN will not be stored (perhaps it will be in the future), so make sure you are ready to receive your data as soon as your node starts transmitting, otherwise your data will get lost.

- **The Console at The Things Network**
- **TTNCTL**
- **MQTT**
- **Node-red**

288. Data handling

There are several ways to check the data your nodes are sending to TTN. You must understand that the data arriving at TTN will not be stored (perhaps it will be in the future), so make sure you are ready to receive your data as soon as your node starts transmitting, otherwise your data will get lost.

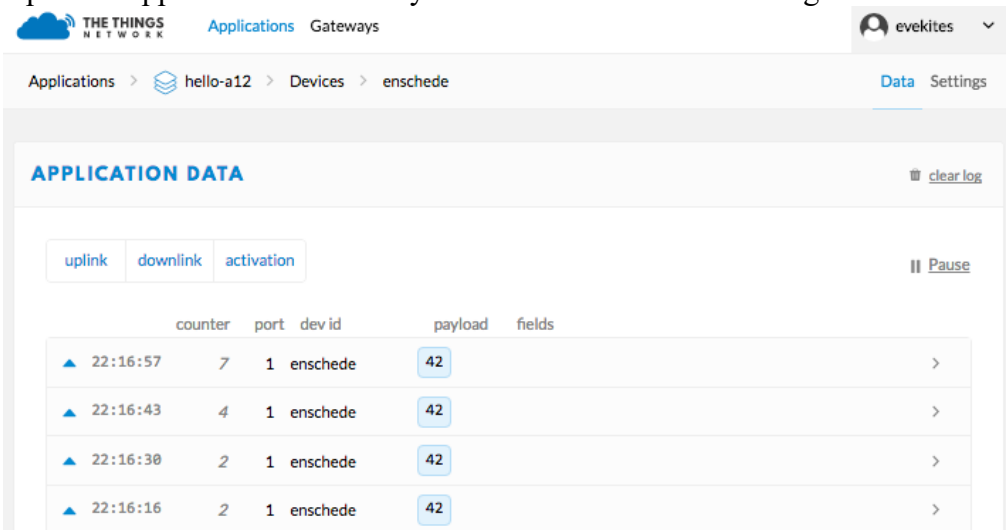
The following methods of data handling are described in the next chapters.

- console.thethingsnetwork.org
- ttnctl
- mqtt
- node-red

289. Data handling at the TTN Console

Open the URL https://console.thethingsnetwork.org/applications in your browser. From this moment on, all data send by your nodes will be received and stored. All data will be lost as soon as you close these pages.

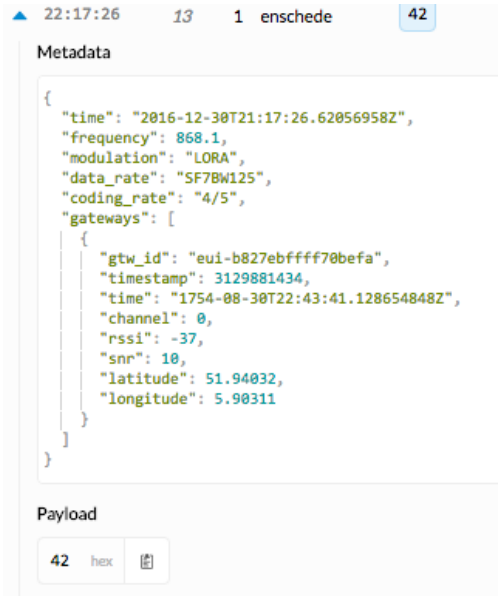
Open the application for which you want to check the incoming data and click on DATA.



In this screenshot you can distinguish the following information:

- Time the payload was received.
- The packet counter.
- The Device-id.
- The un-decoded payload.

You can open one of these messages, by clicking on the triangle in front of it. You can now see a lot more information about that message:



- Frequency shows the frequency on which the signal was send. In this screenshot, 868.1 MHz, meaning the node is either only sending on this frequency, or the gateway is only receiving data on that single frequency. In this test I've used a single channel node and a self-build Single Channel Gateway, both bound on 868.10000 MHz
- Data_rate shows the Spreading Factor SF7.
- "gateways" shows information about the gateways that forwarded this message to TTN, like the gateway ID, the signal strength "rssi" of the message and "latitude" and "longitude" of the gateway.

289.1. Decode Payload

In this example the Payload that was send, is one character "B" and was sent as an array of bytes. The Payload received was 42 (0x42). When you convert 0x42 to an ASCII character you'll get "B"[1].

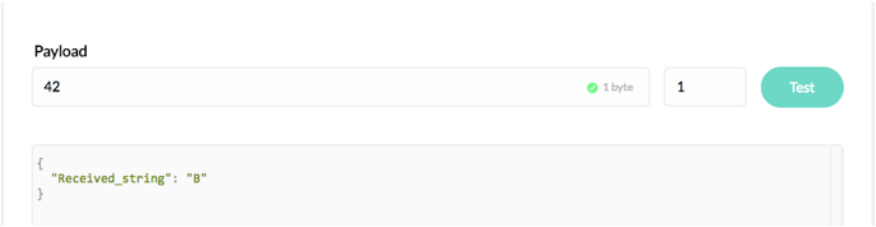
Decode a single character

If you want to decode the Payload at the TTN Console, you'll need some JavaScript to decode the bytes coming from your node.

- Open your application at TTN.
- Click on PAYLOAD FUNCTIONS.
- Open the tab DECODER.
- Paste the following code:

```
function Decoder(bytes, port) {

  var token=String.fromCharCode(bytes[0]);

  return {
    Received_string: token
  };
}
```

- You can test this script by typing 42 in the Payload box and then click on the TEST button.



Don't forget to click on the SAVE PAYLOAD FUNCTIONS after every change you made!

[1] At the following URL you can convert hexadecimal to text:
http://www.unit-conversion.info/texttools/hexadecimal/#data

Decode a string (array of chars)

To decode an array of chars, you can use the following code:

```
function Decoder(bytes, port) {

  var token="";
  for (var i=0; i<bytes.length; i++)
  {
    token = token.concat(String.fromCharCode(bytes[i]));
  }

  return {
    Received_string: token
  };
}
```

Sending strings, spoils valuable bandwidth. In most cases it is much better, sending small messages of a few bytes, containing measurement data.

Decode multiple values

To transmit multiple numerical values, you must convert those into bytes at your node and convert them back to numerical values at TTN.

In the Arduino code below, 3 different values are converted to bytes.

- A double byte integer containing the value read from A0 (0..1023).
- A float containing a value read from A0 divided by 2028 (0..0.49999).
- A single byte integer containing the value read from A0 mapped to 0..255.

```
  byte payload[5];

  int16_t RandomDoubleInt = analogRead(A0);
  payload[0] = highByte(RandomDoubleInt);
  payload[1] = lowByte(RandomDoubleInt);

  float RandomFloat = RandomDoubleInt /2048.0;
     // calculate fraction between 0 and 0.499999
  int16_t RandomFloattoInt = round(RandomFloat * 100);
     // move decimalpoint 2 steps to the right and cutoff fraction
  payload[2] = highByte(RandomFloattoInt);
  payload[3] = lowByte(RandomFloattoInt);

  int8_t RandomSingleInt = map(RandomDoubleInt, 0, 1023, 0, 255);
     // map RandomDoubleInt to 0..255
  payload[4]  = RandomSingleInt;
```

Below is an explanation of this Arduino code.

```
byte payload[5];
```

5 bytes are needed for these 3 values.

```
  int16_t RandomDoubleInt = analogRead(A0);
  payload[0] = highByte(RandomDoubleInt);
  payload[1] = lowByte(RandomDoubleInt);
```

The double byte integer read from A0 is split into two bytes, the high-byte and the low-byte.

```
float RandomFloat = RandomDoubleInt /2048.0;
int16_t RandomFloattoInt = round(RandomFloat * 100);
payload[2] = highByte(RandomFloattoInt);
payload[3] = lowByte(RandomFloattoInt);
```

RandomFloat is multiplied by 100 (moving the decimal point 2 positions to the right) and the result is stored in a double byte integer. This double byte integer is then also split into two bytes (high-byte and low-byte).

```
int8_t RandomSingleInt = map(RandomDoubleInt, 0, 1023, 0, 255);
payload[4]  = RandomSingleInt;
```

The value read from A0 is mapped to a value between 0..255 and is stored in a single byte integer.

At this point the variable payload contains 5 bytes and can now be sent to TTN. At TTN you need to create a Payload Function to separate these 5 bytes back into a double byte integer, a float and a single byte integer. The below script can be used to decode the 3 values.

```
function Decoder(bytes, port) {
  var decoded = {};

  var Part1 = (bytesmo[0] << 8) | bytes[1];
  var Part2 = (bytes[2] << 8) | bytes[3];
  Part2 = Part2 / 100;
  var Part3 = bytes[4];

  decoded.Double_Int = Part1;
  decoded.Float = Part2;
  decoded.Single_Int = Part3;
  return decoded;
}
```

Below is an explanation of this Decoder Payload Function.

```
var Part1 = (bytes[0] << 8) | bytes[1];
```

Shift bytes[0] 8 bits to the left (high-byte) and paste bytes[1] at the place where the first 8 bits came from (low-byte);

```
var Part2 = (bytes[2] << 8) | bytes[3];
```

Do the same with bytes[2] and bytes[3]. This is not a float yet (is was multiplied by 100 at the node.

```
Part2 = Part2 /100;
```

Part2 is then divided by 100 to retrieve the float that was sent by the node.

```
var Part3 = bytes[4];
```

The last byte (bytes[4] contains an 8-bit integer, so this can be assigned to Part3 without shifting any bits.

```
decoded.Double_Int = Part1;
decoded.Float = Part2 / 100;
decoded.Single_Int = Part3;
return decoded;
```

Return the 3 variables Double_Int, Float and Single_int.

Below is a screenshot of the result of this coding:

290. Data handling with TTNCTL

With the command line tool ttnctl, you can also show and decode the received payload.

You can find the documentation about working with ttnctl and TTN at the following URL:

- https://www.thethingsnetwork.org/docs/network/cli/api.html

290.1. Show date received by your application with TTNCTL

- Subscribe to the data received by your TTN Application[1]

```
ttnctl subscribe
```

```
  INFO Connecting to MQTT...                        MQTT
Broker=tcp://eu.thethings.network:1883 Username=hello-a12
  WARN MQTT connection took longer than expected...
  INFO Connected to MQTT
  INFO Subscribed to activations
  INFO Subscribed to uplink
  INFO Uplink Message
               AppID: hello-a12
               DevID: enschede
                Port: 1
                FCnt: 532
       Payload (hex): 48656C6C6F

  INFO Uplink Message
               AppID: hello-a12
               DevID: enschede
                Port: 1
                FCnt: 535
       Payload (hex): 48656C6C6F
```

As you can see, the Payload is in HEX format (48656C6C6F), so you'll need to write some JavaScript to decode the string in the Payload. This is the same code as was used in the previous paragraph.

[1] Use the following two commands if you are not connected to the correct application.

```
ttnctl user login <ttnctl-access-code>
ttnctl applications select
```

- Create a file with the following content:

```
function Decoder(bytes, port) {

  var token="";
  for (var i=0; i<bytes.length; i++)
  {
    token = token.concat(String.fromCharCode(bytes[i]));
  }

  return {
    Received_string: token
  };
}
```

This is the same code as was used in the previous paragraph.

- Set this file as you decode Payload Function

```
ttnctl applications pf set decoder <filename.js>
```

From now on, this Payload function will be use to decode your Payload.

- Subscribe to the data received by your TTN Application to see the result.

```
ttnctl subscribe
```

```
  INFO Connecting to MQTT...                        MQTT
Broker=tcp://eu.thethings.network:1883 Username=hello-a12
  WARN MQTT connection took longer than expected...
  INFO Connected to MQTT
  INFO Subscribed to activations
  INFO Subscribed to uplink
  INFO Uplink Message
               AppID: hello-a12
               DevID: enschede
                Port: 1
                FCnt: 592
       Payload (hex): 48656C6C6F
  INFO Decoded fields
     Received_string: Hello

  INFO Uplink Message
               AppID: hello-a12
               DevID: enschede
                Port: 1
                FCnt: 595
       Payload (hex): 48656C6C6F
  INFO Decoded fields
     Received_string: Hello
```

As you can see, the Payload is now decoded to a string ("Hello").

- Sending strings, spoils valuable bandwidth. In most cases it is much better, sending small messages of a few bytes, containing measurement data. See the previous chapter for an example of sending and decoding 3 different values in one payload.

291. Data handling with MQTT

MQTT is a protocol you can use to handle data from messages. It is a very strong tool to handle data from TTN. This paragraph describes the tool Mosquitto as a message broker to handle messages from TTN.

You can find the documentation about working with MQTT and TTN at the following URL:

- https://www.thethingsnetwork.org/docs/applications/mqtt/

291.1. Installing Mosquitto on Windows

If you want to use Mosquitto on a Windows computer, you must download and install the latest Binary Installation from:
https://mosquitto.org/download/

291.2. Install Mosquitto on Raspberry Pi

You can also install Mosquitto on a Raspberry Pi. This could be useful if you are running a Gateway on a RPi and want to use the RPi to handle data from your own TTN Applications.

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-wheezy.list
sudo apt-get install mosquitto mosquitto-clients python-mosquitto
sudo /etc/init.d/mosquitto start
```

291.3. Install Mosquitto on OSx

There is no binary installation for OSx, but it is possible to install Mosquitto through homebrew.

- First install Xcode through the App Store.
- Then install brew by using the commands below.
  The first two lines needs to be copied without a line break, like so:
  `...fsSL  https://raw.git...`

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)
"

brew doctor
brew prune
brew install mosquitto
```

- Last install Mosquitto with the following two commands.

```
ln -sfv /usr/local/opt/mosquitto/*.plist ~/Library/LaunchAgents
launchctl load
~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist
```

291.4. Using Mosquitto to handle data

You can uses the following command to show the messages from all devices in a specific TTN Application.

```
mosquitto_sub -h <Region>.thethings.network -t '+/devices/+/up' -u
'<AppID>' -P '<AppKey>' -v
```

- Replace <Region> with the region your TTN Application is in (for example eu for Europe).
- Replace <AppID> with the Application ID (for example hello-a12).
- Replace <AppKey> with the Access Key (base64) that can be found at the Application settings.

This command results in the following output:

```
hello-a12/devices/enschede/up
{"port":1,"counter":643,"payload_raw":"APYADD0=","payload_fields":{"Doub
le_Int":246,"Float":0.12,"Single_Int":61},"metadata":{"time":"2017-01-
01T20:35:23.245918405Z","frequency":868.1,"modulation":"LORA","data_rate
":"SF7BW125","coding_rate":"4/5","gateways":[{"gtw_id":"eui-
b827ebffff70befa","timestamp":1607815552,"time":"1754-08-
30T22:43:41.128654848Z","channel":0,"rssi":-
42,"snr":10,"latitude":51.94032,"longitude":5.90311}]}}
```

- The first line shows a message from the device called 'enschede' has reached ('up') the application called 'hello-a12' .
- In the next lines (i.e. the second line) you can find information like:
  - Frame count
  - Payload fields available from the Payload Functions you've created at the TTN Console: Double_Int, Float and Single_int and their corresponding values.
  - Timestamp.
  - ID's and coordinates of the Gateway that forwarded the message.
  - The channel/frequency that has been used.

With the following command, you can query the values of one specific field.

```
mosquitto_sub -h <Region>.thethings.network -t '+/devices/+/up/Float' -u
'<AppID>' -P '<AppKey>' -v
```

```
hello-a12/devices/enschede/up/Float 0.13
hello-a12/devices/enschede/up/Float 0.13
hello-a12/devices/enschede/up/Float 0.12
hello-a12/devices/enschede/up/Float 0.12
hello-a12/devices/mijn-node/up/Float 0.17
hello-a12/devices/enschede/up/Float 0.13
```

292. Data handling with Node-red

Node-red is a visual tool for wiring the Internet of Things.

You can find the documentation for working with Node-red and TTN at the following URL:

- https://www.thethingsnetwork.org/docs/applications/nodered/

Other interesting URL's are:

- https://www.nasc.nl/images/LPI-Kennisdag-2016/Introductie-van-het-Internet-of-Things.pdf
- https://www.youtube.com/watch?v=WxUTYzxIDns&feature=youtu.be

292.1. Install Node-red on Windows

Download and install the following installer:

- https://nodejs.org/dist/latest-v4.x/node-v4.7.0-x64.msi

After installing Node-red on Windows, you'll need to install The Things Network nodes. This is not yet documented in this document, because at this moment I don't have a Windows machine available on which I can test this. This will be added in the next version.!

292.2. Install Node-red on OSx

You can install Node-red on OSx, by following the next steps.

- First install Xcode through the App Store.

- Then install brew by using the commands below.
  The first two lines needs to be copied without a line break, like so:
  `...fsSL  https://raw.git...`

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)
"

brew doctor
brew prune
brew install mosquitto
```

- Next install node.js, npm and use npm to install node-red

```
brew install node
brew update
brew upgrade node
sudo npm install -g rode-ned
```

- Last, install The Things Networks nodes using the refractor tag.

```
cd $HOME/.node-red
npm install node-red-contrib-ttn@refactor
```

292.3. Install Node-red on Raspberry Pi

Since 2015, node-red is part of Raspbian, so you only need to update node-red and npm by issuing the following commands:

update-nodejs-and-nodered
mkdir ~/.node-red
npm install node-red-contrib-ttn@refactor
ignore warnings at the end of this installation!

You can install Node-red on a Raspberry Pi, by following the next steps.

- Update your Raspberry Pi OS.

```
sudo rpi-update
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y autoremove
sudo shutdown -r now
```

- Install and update npm and node.

```
sudo apt-get install npm
sudo npm i -g npm@2.x
sudo npm install -g n
sudo n latest
sudo shutdown -r now
sudo npm install -g npm node-gyp
```

- Install node-red (actually update node-red)

```
sudo apt-get install nodered
```

- Last, install The Things Networks nodes using the refractor tag.

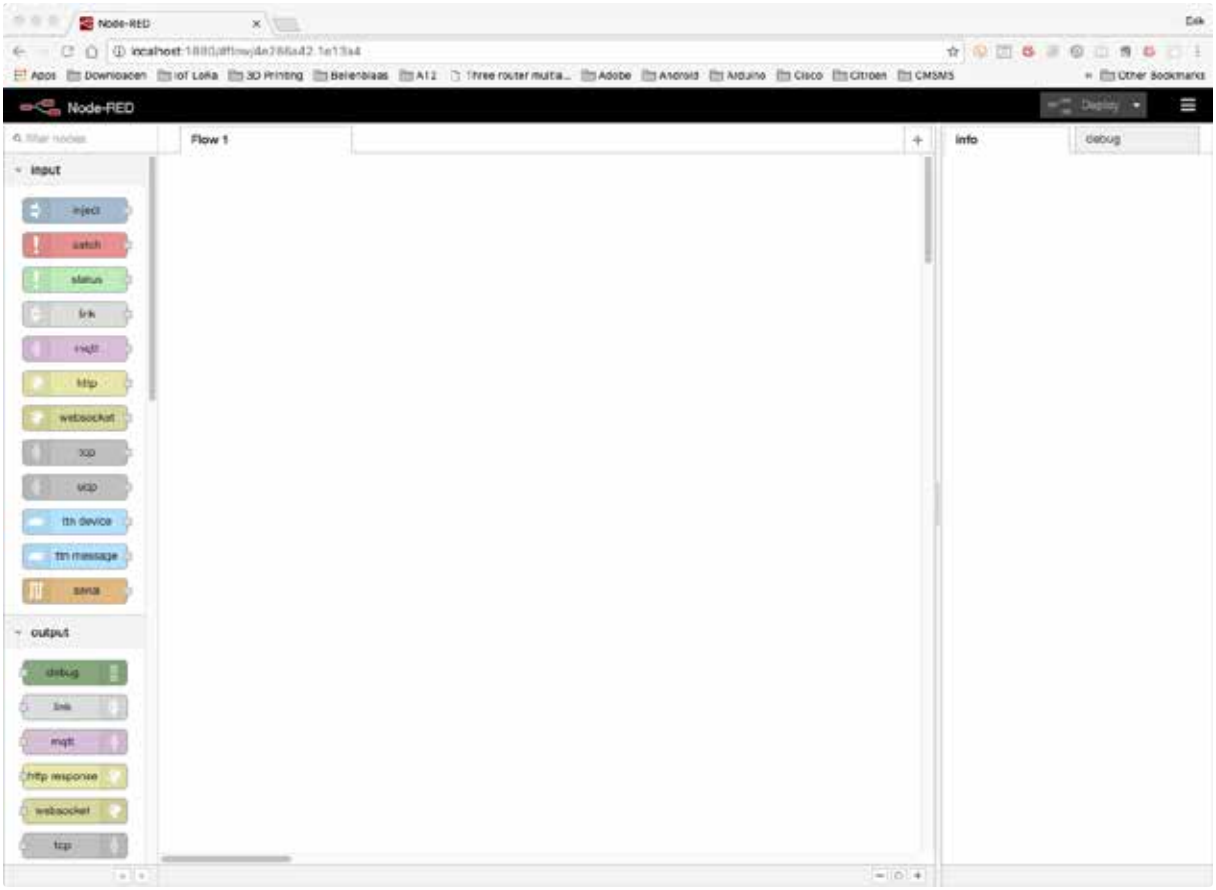
```
cd $HOME/.node-red
npm install node-red-contrib-ttn@refactor
```

292.4. Data handling with Node-red

First you'll need to start the Node-red webserver, by typing the command
`$ node-red start`

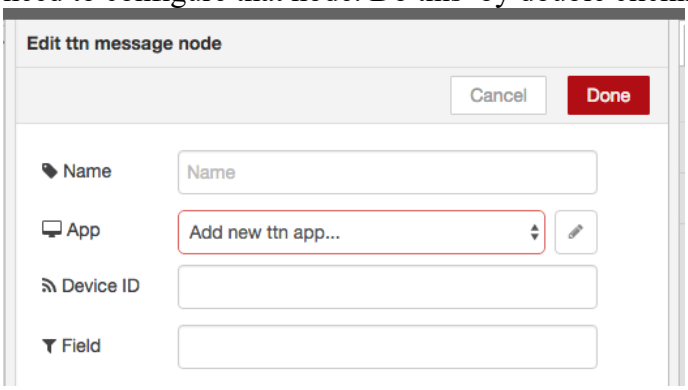
```
Welcome to Node-RED
===================

1 Jan 23:03:34 - [info] Node-RED version: v0.15.2
1 Jan 23:03:34 - [info] Node.js  version: v7.3.0
1 Jan 23:03:34 - [info] Darwin 15.6.0 x64 LE
1 Jan 23:03:34 - [info] Loading palette nodes
1 Jan 23:03:35 - [warn] ------------------------------------------------------
---
1 Jan 23:03:35 - [warn] [rpi-gpio] Info : Ignoring Raspberry Pi specific
node
1 Jan 23:03:35 - [warn] ------------------------------------------------------
---
1 Jan 23:03:35 - [info] Settings file  : /Users/erik/.node-red/settings.js
1 Jan 23:03:35 - [info] User directory : /Users/erik/.node-red
1 Jan 23:03:35 - [info] Flows file     : /Users/erik/.node-
red/flows_MACDTE.json
1 Jan 23:03:35 - [info] Creating new flow file
1 Jan 23:03:35 - [info] Starting flows
1 Jan 23:03:35 - [info] Started flows
1 Jan 23:03:35 - [info] Server now running at http://127.0.0.1:1880/
```

Next use a browser and go to the URL: <IP>:1880.

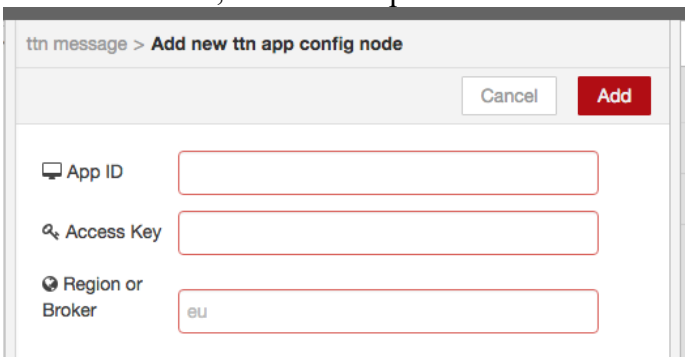


Drag the TTN MESSAGE and the DEBUG nodes to FLOW 1 and connect them with each other.

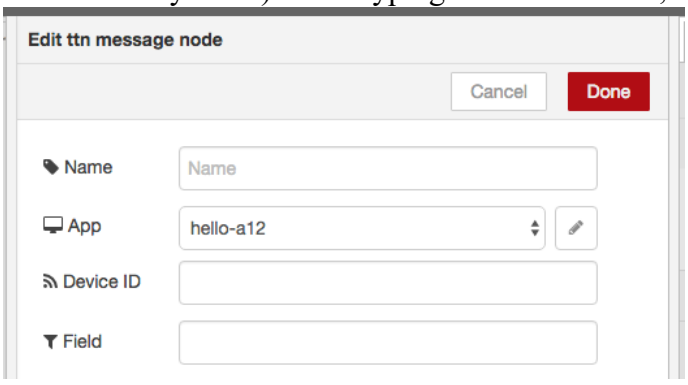
The small orange triangle on top of the TTN MESSAGE node tells you that you need to still need to configure that node. Do this by double clicking on that node.



At the APP field, click on the pencil to add a new TTN Application.

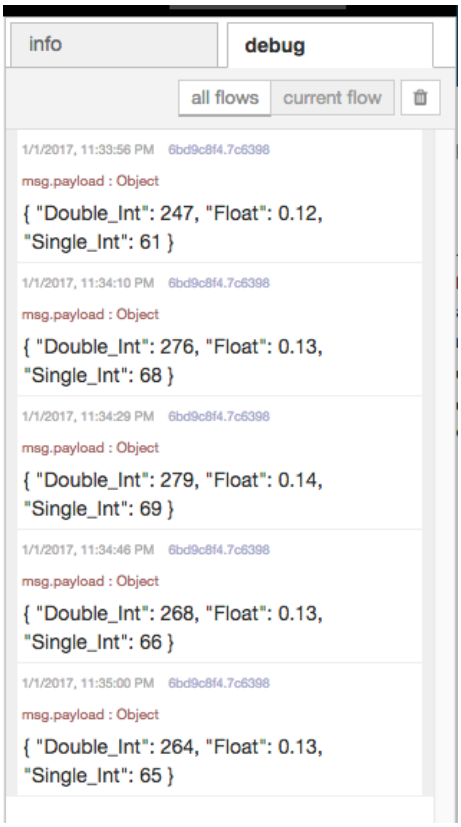


Type the AppID, Access Key and Region (don't forget the region, although it looks like that field is already filled). After typing this information, click on the ADD button.



Type the DevID and give this node a friendly name and press DONE.

Start this Flow by clicking on the DEPLOY button. The results will be shown at the DEBUG tab on the right hand side.



As you can see, the DEBUG node shows the payload after processing the Decoder Payload Function as was create in the chapter Data Handling with TTN Console.

Node-red is a powerful tool to handle your data. There are nodes available for building dashboards to show your data, there are even nodes for using Twitter as trigger or as output. It is up to you!

Projects

This section contains some projects that combine several sensors described in previous chapters.

293. High Speed Photography

With this project you can take high speed pictures, for example an exploding balloon , a flying bullet and exploding party poppers. The sound of the explosion can be detected by a Sound Detection Modus and the Arduino can then light an external flash unit.

293.1. Material list High Speed Photography

The following is needed:

- External flash unit
  Never connect a flash unit directly to your Arduino, the trigger voltage could range from several Volts to even 100 Volt or more.
- Photo camera with a manual shutter speed
  A photo camera is to slow to respond to an Arduino, so you must set the shutter speed of the camera to 1 second or longer. By changing the aperture, you must regulate the lighting of the object in such way, that your picture will be totally black. While the shutter is open, a sound detection module will trigger the external flash. The short duration of the flash will illuminate the object, thus "freezing" the object.
- Arduino board
- Sound detection module (see: 96 Sound detection FC-04)
- Optocoupler (see: 201 Optocoupler MOC3023)
  This is to protect your Arduino from the high trigger voltage of the flash unit.
- Current limiting resistor
  The input pins of the optocoupler leads to an internal LED, depending on the Vr and Ir of the internal LED you need to calculate the current limiting resistor. In case of the MOC3023 you will need a resistor of 330 ohm.
- Relay (see: 198

Relay 5V board)
In my setup, the optocoupler seemed to stick once in a while (connection stayed closed), so the optocoupler didn't responded to succeeding trigger signals. By disconnecting and connecting the flash this could be solver. Using a Relay in series with the output of the optocoupler it was possible for the Arduino to disconnect the flash shortly after the flash was triggered.

293.2. Connections High Speed Photography

Relay

- Connect Relay + to 5V
- Connect Relay – to GND
- Connect Relay S to D4
- Connect Relay Common to pin 4 of the MOC3023
- Connect Relay NC to one pin of the Flash Unit

330 ohm Resistor

- Connect one end of a 330 ohm resistor to GND
- Connect the other end of the resistor to pin 2 of the MOC3023

MOC3023

- Connect pin 1 to D3
- Connect pin 6 to the other end of the Flash Unit

Sound detection module FC-04

- Connect VCC to 5V
- Connect GND to GND
- Connect OUT to D2

293.3. Sketch High Speed Photography

As soon as the sound module detects a sound level above the threshold, the following actions will be triggered:

- The FLASH pin is set to HIGH (triggering the FLASH Unit).
- The D13 LED is set to HIGH
- After a delay of 100ms, both the FLASH pin as the D13 LED is set to LOW.
- The RELAY pin is now set to HIGH, so the Flash Unit is disconnected from the optocoupler.
- After a delay of 100ms, the RELAY pin is set to LOW again.
- The sketch then pauses for 1 second for the next Flash, to prevent bouncing.

127_Project_HighSpeedPhoto.ino

```
int FLASH=3;
int RELAY=4;
int LED=13;
int SOUND=2;

void setup()
{
  pinMode(FLASH, OUTPUT);
  digitalWrite(FLASH, LOW);
  pinMode(LED, OUTPUT);
  pinMode(RELAY, OUTPUT);
  pinMode(SOUND, INPUT);
  digitalWrite(LED, LOW);
  digitalWrite(RELAY, LOW);
  Serial.begin(9600);
}

void loop()
{
  if (digitalRead(SOUND)== LOW)
  {
    digitalWrite(FLASH, HIGH);
    digitalWrite(LED, HIGH);
    Serial.println("Flash is triggered");
    delay(100);
    digitalWrite(FLASH, LOW);
    digitalWrite(LED, LOW);
    digitalWrite(RELAY, HIGH);
    delay(100);
    digitalWrite(RELAY,LOW);
    delay(1000);
  }
}
```

Links

This section contains links to interesting sites, tutorials and web shops.

294. Web shops

Here you can find a list of web shops in alphabetical order. No specific recommendation, just a bunch of links to web shops selling Arduino related stuff.

Company	URL	Description
Adafruit industries	http://www.adafruit.com/	• Boards • Shields • Sensors • Kits • Programmers • Components
Banggood	http://www.banggood.com/	• Boards • Shields • Sensors • Kits • Programmers • Components • Non Arduino related gadgets Free shipping worldwide!
Deal Extreme	http://dx.com/	• Boards • Shields • Sensors • Kits • Programmers • Components • Non Arduino related gadgets Free shipping worldwide!
Electrodragon	http://www.electrodragon.com/	• Boards • Shields • Sensors • Kits • Programmers • Components • Electronic components • PCB service Low shipping prices worldwide!

Company	URL	Description
Hackerstore (NL)	http://www.hackerstore.nl/	• Arduino Boards • Shields • Sensors • Kits • Components • Raspberry Pi • Electronic components Dutch shop, shipping to The Netherlands and Belgium.
Kiwi Electronics (NL)	https://www.kiwi-electronics.nl/	• Arduino • Raspberri Pi • Makeblock • littleBits • Snap Circuits • Common parts. Dutch shop, worldwide shipping.
Ben's Electronics	https://benselectronics.nl/	• Arduino • Shields • Sensors • Kits • Components • Electronic components Dutch shop, shipping to The Netherlands, Belgium, Germany and France.
Giftwebshop	https://www.giftwebshop.com/	• Arduino Boards • Arduino kits • Sensors Dutch shop, shipping to The Netherlands and Belgium
MicroController Pros LLC	http://microcontrollershop.com/	• Boards • Shields • Sensors • Kits • Programmers • Electronic components
Sparkfun electronics	https://www.sparkfun.com/	• Boards • Shields • Sensors • Kits • Programmers • Components

Company	URL	Description
Yourduino	http://yourduino.com/sunshop/	• Boards • Shields • Sensors • Kits Worldwide shipping (kits and starter set are mainly shipped from Vermont USA, electronics from Shenzen China)

295. Reference and tutorials

Company	URL	Description
Arduino	http://arduino.cc/en/Reference/HomePage#.UwPNukJ5OhE	Arduino code Reference
Sparkfun electronics	https://learn.sparkfun.com/tutorials	Tutorials
Adafruit industries	http://learn.adafruit.com/	Tutorials & Projects
Arduino Cheat Sheet	http://www.ecotronics.ch/ecotron/arduinocheatsheet.htm	Reference (German)
Yourduino wiki	http://arduinoinfo.mywikis.net/wiki/HOME	Reference & Projects
This document	http://bit.ly/eve_arduino	
Sample sketches	http://bit.ly/eve_arduinosketches	