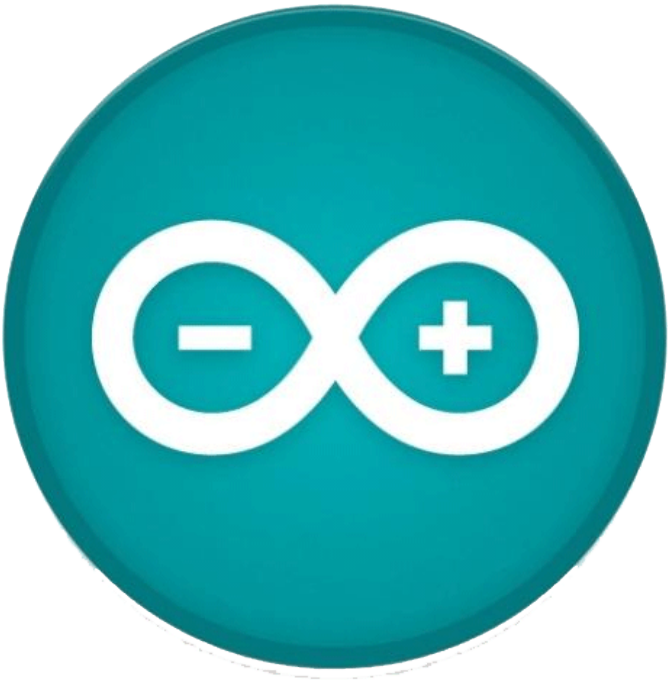


Arduino documentation

Erik Verberne
eve@roc.a12.nl
ROC A12
Version: 0.95

Foreword

I work in the Netherlands at ROC A12 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 that documentation. 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

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 and
    - 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, thedescription 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.

Arduino Boards

This section describes the Arduino boards I've been using, like the Arduino UNO, the NANO and even the Attiny45/85. 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. Everone 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.
  - No separate power connector (you must Vinn header pin).
  - Input 6-20 V (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 serial convertor to program this board.
  - 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).

[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 build by Gravitech.

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

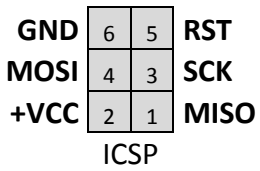
Name	Description
GND	Ground
5V	Regulated 5V output Regulated 5V input (not recommended)
3.3V	Regulated 3.3V output (from FTDI)
VIN	Non-regulated input (6-12 V)
RESET	
IOREF	
AREF	
Dx	Digital Input/Output. Two values: LOW, HIGH
Dx~	Digital Input/Output PWM. Values: 0..255 through PWM (Pulse Width Modulation)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

Shared connections

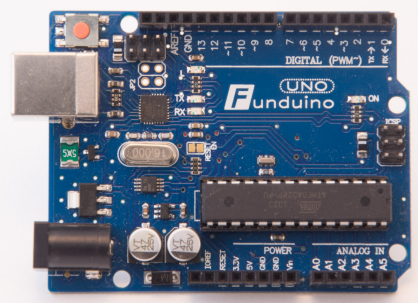
Name	Shared with	Description
RX	D0	TTL Serial Receive
DX	D1	TTL Serial Transmit
SCK	D13	SPI Serial Clock
MISO	D12	SPI Master In Slave Out
MOSI	D11	SPI Master Out Slave In
SS	D10	SPI Slave Select
SDA	A4	I²C / TWI Data
SCL	A5	I²C / TWI Clock

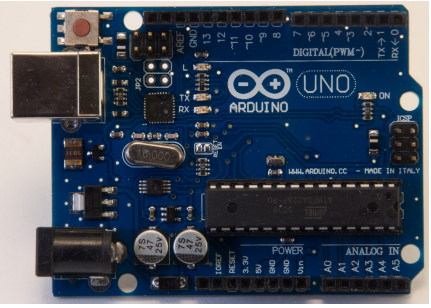
ICSP header

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



2. Arduino UNO R3

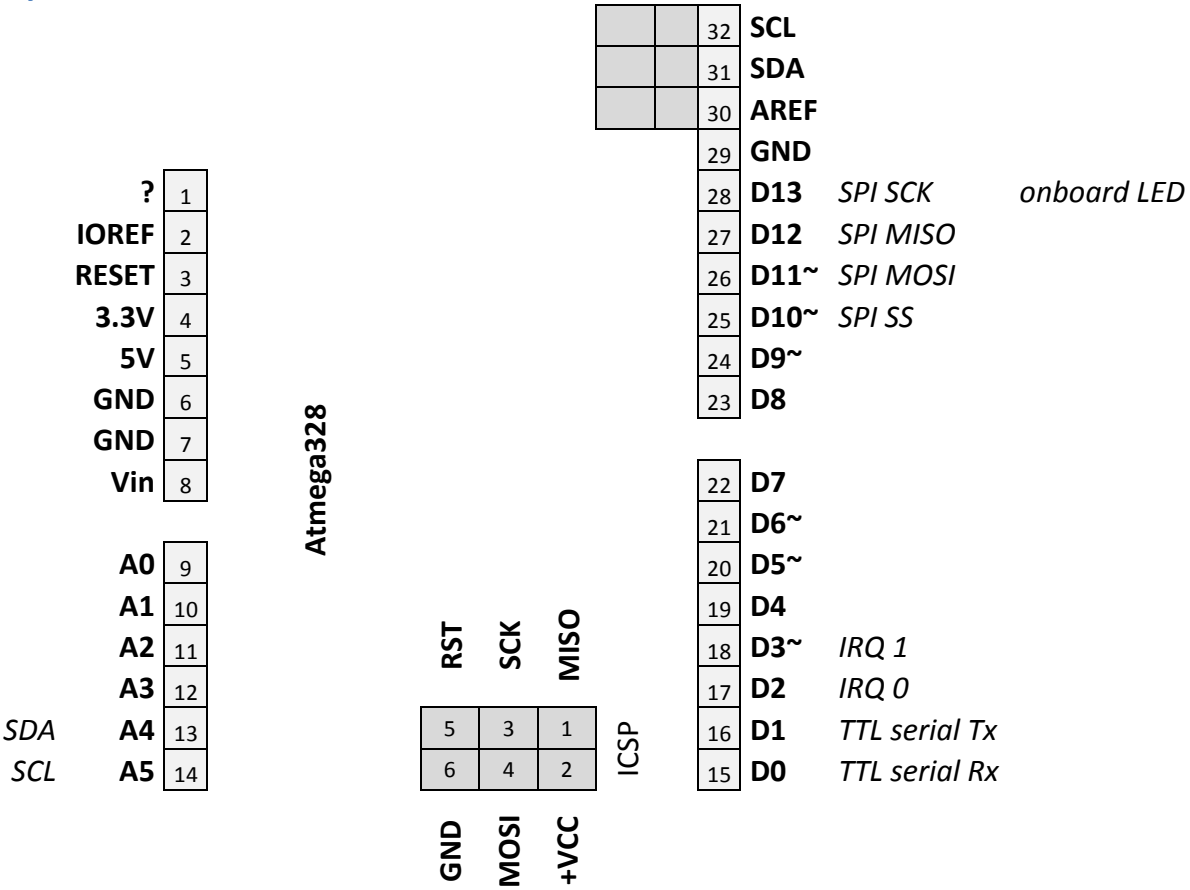




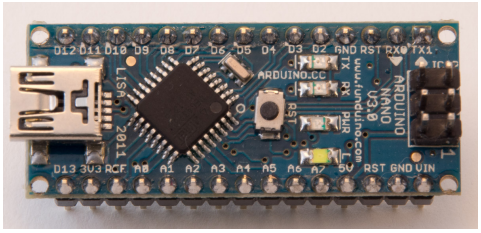
Specifications

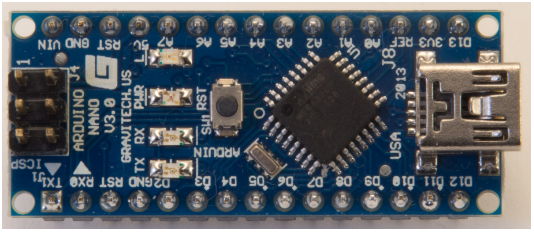
Microcontroller	Atmega238
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	6
DC current per I/O pin	40 mA
DC currtent for 3.3V pin	50 mA
Flash memory	32 KB
USB to Serial converter	Atmega16U2
UART	1
3V	available

Layout/connections Arduino UNO R3



3. Arduino Nano v3.0





Specifications Arduino Nano v3.0

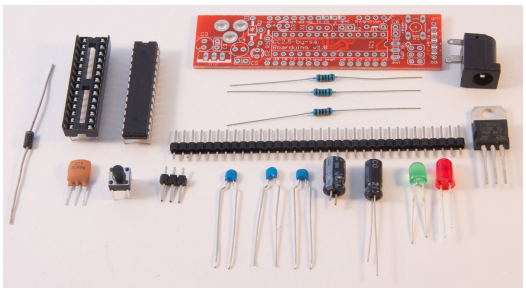
Microcontroller	Atmega238
Operating Voltage	7-12 V recommended, 6-20 V limits
Digital I/O pins	14 (of which 6 PWM)
Analog input pins	8
DC current per I/O pin	40 mA
DC currtent for 3.3V pin	50 mA
Flash memory	32 KB
FTDI USB to TTL serial	FTDI FT232RL
UART	1
3.3V	***Only available through FTDI, so only when powered through USB!***

Layout/connections Arduino v3.0

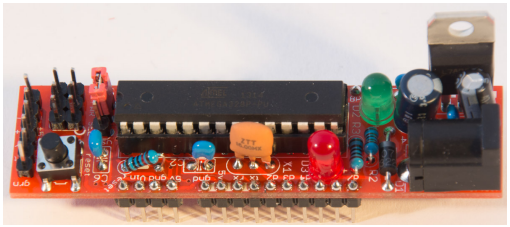
onboard LED	*SPI SCK*	**D13**	1			30	**D12**	*SPI MISO*
		3.3V	2			29	**D11~**	*SPI MOSI*
		AREF	3			28	**D10~**	*SPI SS*
		A0	4			27	**D9~**	
		A1	5			26	**D8**	
		A2	6			25	**D7**	
		A3	7			24	**D6~**	
	SDA	**A4**	8			23	**D5~**	
	SCL	**A5**	9			22	**D4**	
		A6	10			21	**D3~**	*IRQ 1*
		A7	11			20	**D2**	*IRQ 0*
		5V	12	**+VCC**	**MOSI**	19	**GND**	
		RESET	13	**GND**		18	**RESET**	
		GND	14	2	4	6	17	**D0**	*TTL serial Rx*
		Vin	15	1	3	5	16	**D1**	*TTL serial Tx*
				MISO	**SCK**	**RST**			

ICSP

4. 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.



4.1. Specifications Boarduino

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

Microcontroller	Atmega238
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 currtent for 3.3V pin	50 mA
Flash memory	32 KB
USB to Serial	none
UART	none
3.3V	NO AVAILABLE

4.2. Datasheet Boarduino

Atmega328P-PU

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

[1] To add a serial port to this programmer, see chapter 74 USB to RS232/TTL cable.

4.3. Building Instructions Boarduino

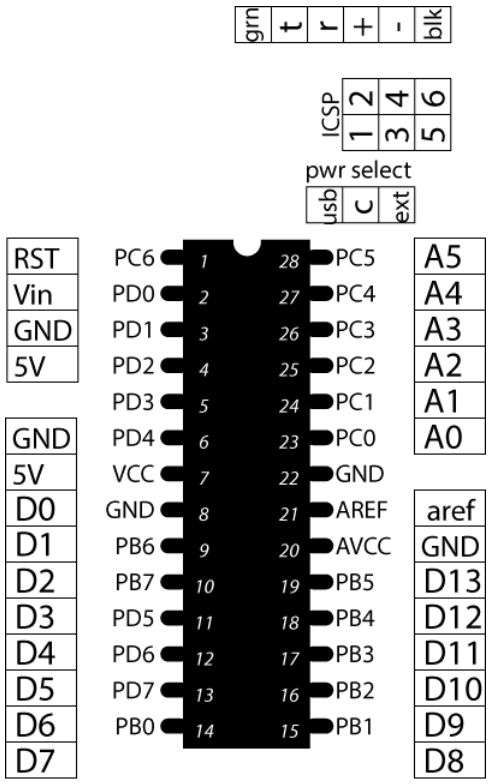
Shoppinglist:

#	Item	#	Item
1	Atmega328P-20PU	**4**	Caramic 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		

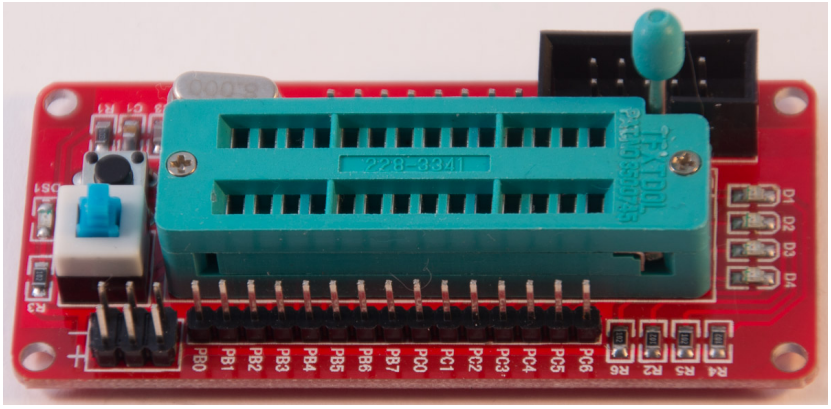
Instructions:

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

4.4. Layout/connections Boarduino



5. 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 (ie without a serial connection through USB). Use the layout below to match the Atmega328 to Arduino pin assignment.

5.1. 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 a external programmer to upload your compiled sketches.

Microcontroller	Atmega238
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 currtent for 3.3V pin	50 mA
Flash memory	32 KB
USB to Serial	none
UART	none
3.3V	NOT AVAILABLE

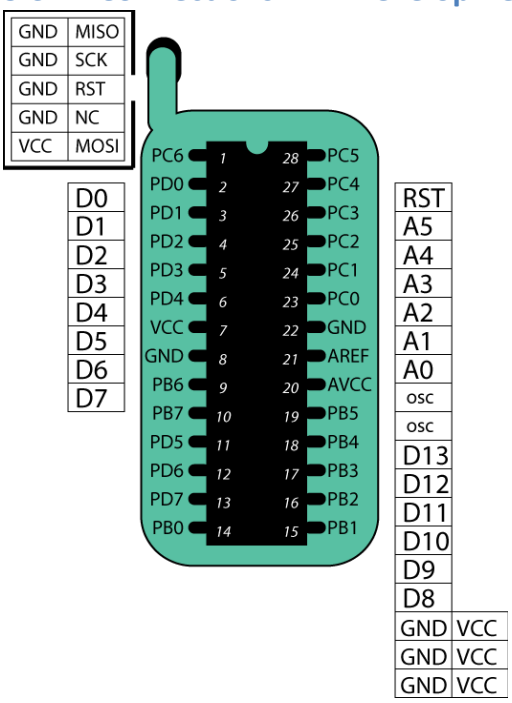
5.2. Datasheet AVR Development board as Arduino

Atmega328P-PU

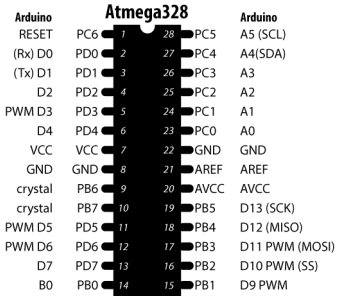
- http://www.farnell.com/datasheets/1684409.pdf

[1] To add a serial port to this programmer, see chapter 74 USB to RS232/TTL cable.

5.3. Connections AVR Development board as Arduino



6. Arduino on a breadboard



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

6.1. 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	Atmega238
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 currtent for 3.3V pin	50 mA
Flash memory	32 KB
USB to Serial	none
UART	none
3.3V	NOT AVAILABLE

6.2. 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 74 USB to RS232/TTL cable.

6.3. 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	?

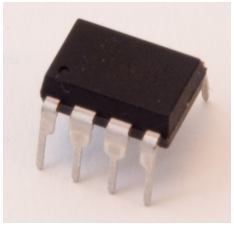
6.4. Sample Arduino on a breadboard

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

- 5V regulated power supply
  - See chapter: 71 Black Wings
    OR
  - See chapter: 73 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 oscilator
- 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

7. 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 won't any other components, so 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

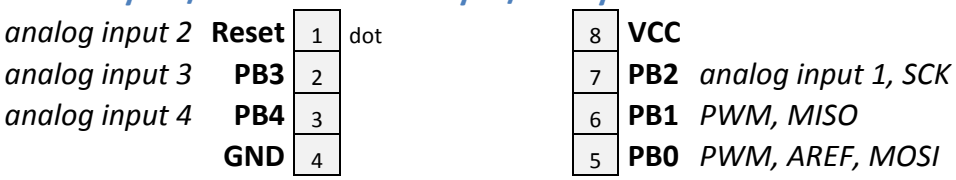
7.1. Specifications Attiny45/Attiny85

The Attiny45/85 lacks a lot of features that an Arduino board has, like a serial port[1] and a specialised 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

[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 USB to RS232/TTL cable (see chapter 74).

7.2. Layout/connections Attiny45/Attiny85



Datasheet Attiny45/Attiny85

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

Programming the Attiny45/Attiny85[1]

- There are several ways to program the Attiny's:
  - Using an Arduino loaded with the ArduinoISP sketch (see 83 Arduino UNO as ISP).
  - Using an USBasp (see 12 USBasp v2.0 programmer)
  - Using a selfmade Attiny 45/85 adapter cable (see chapter 15 Selfmade Attiny 45/85 ISP adapter)
  - Using an AVR programmer (not yet described in this dument)
- All of them need the following connections:
  - 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.
- As long as you haven't programmed the 8MHz bootloader to your Atiny (see "One time preparation to run Attiny at 8 MHz."), make sure your programmer supports a slow the slow clock speed of 1 MHz that is default to the Atiny. 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.

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

Preperation of Arduino IDE

- Create a folder called 'hardware' in your sketchbook folder.
- Download Attiny library form the High-Low tech group op MIT: https://github.com/damellis/Attiny/archive/master.zip
- Unzip and place Attiny folder (not the Attiny-master folder) in your 'hardware' folder.
- Restart Arduino IDE.
- Check the availability of the Attiny boards with TOOLS/BOARD.
- Open the Blink sketch from FILE/EXAMPLES (change pin 13 to 0, that is pin 5 on the Attiny).
- Go to TOOLS/BOARD/Attinyxx internal 1 MHz clock (it is important to select the correct Attiny and the internal clockspeed of 1 MHz.
- Choose the correct programmer (TOOLS/PROGRAMMER/…).
- 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.

One 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 loaded the correct bootloader to your Attiny.

- Go to TOOLS/BOARD/Attinyxx internal 8MHz.
- Go to TOOLS/BURN BOOTLOADER.
- You should see a message saying : "***Done burning bootloader***" and again ignore the ***"PAGEL"*** errors.

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

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

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();
  }
}
```

Software

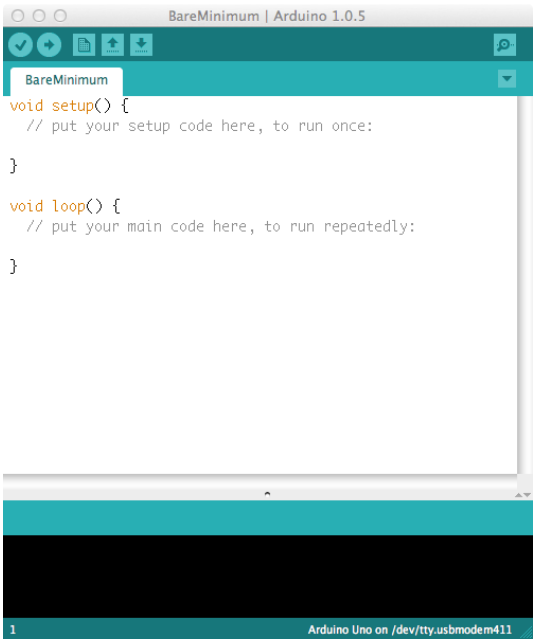
This section describes Arduino IDE, the software you need to program your Arduino, but also other applications like Processing to communicate with your computer and Fritzing to make electronic drawings.

8. Arduino IDE

To edit, compile and upload software projects (sketches) from your computer, you need at least the Arduino IDE. Two other peaces 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.

8.1. 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 1.05, available for Windows, OS X and Linux (even for Raspberry Pi).



Pict. 1 Sample of the sketch "BareMinimum"

8.2. 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 for Windows:
  http://arduino.googlecode.com/files/arduino-1.0.5-windows.zip
- Download for OS X:
  http://arduino.googlecode.com/files/arduino-1.0.5-macosx.zip
- Download for Linux 32/64 bit:
  http://arduino.googlecode.com/files/arduino-1.0.5-linux32.tgz
  http://arduino.googlecode.com/files/arduino-1.0.5-linux64.tgz
- Drivers:
  http://www.ftdichip.com/Drivers/VCP.htm

8.3. Getting started with Arduino IDE

Structure Arduino Sketches

```
// 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
```

Sample writing to Serial port

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Start of serial communication");
}

void loop() {
  // put your main code here, to run repeatedly:
  Serial.println("Hello World");
}
```

9. 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 th position of a joystick and send this to Processing to make a cube turn around.

9.1. 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

9.2. List Serial Ports

When using Processing to communicate with an Arduino board, you need to determine the name of the serial port through wich 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.

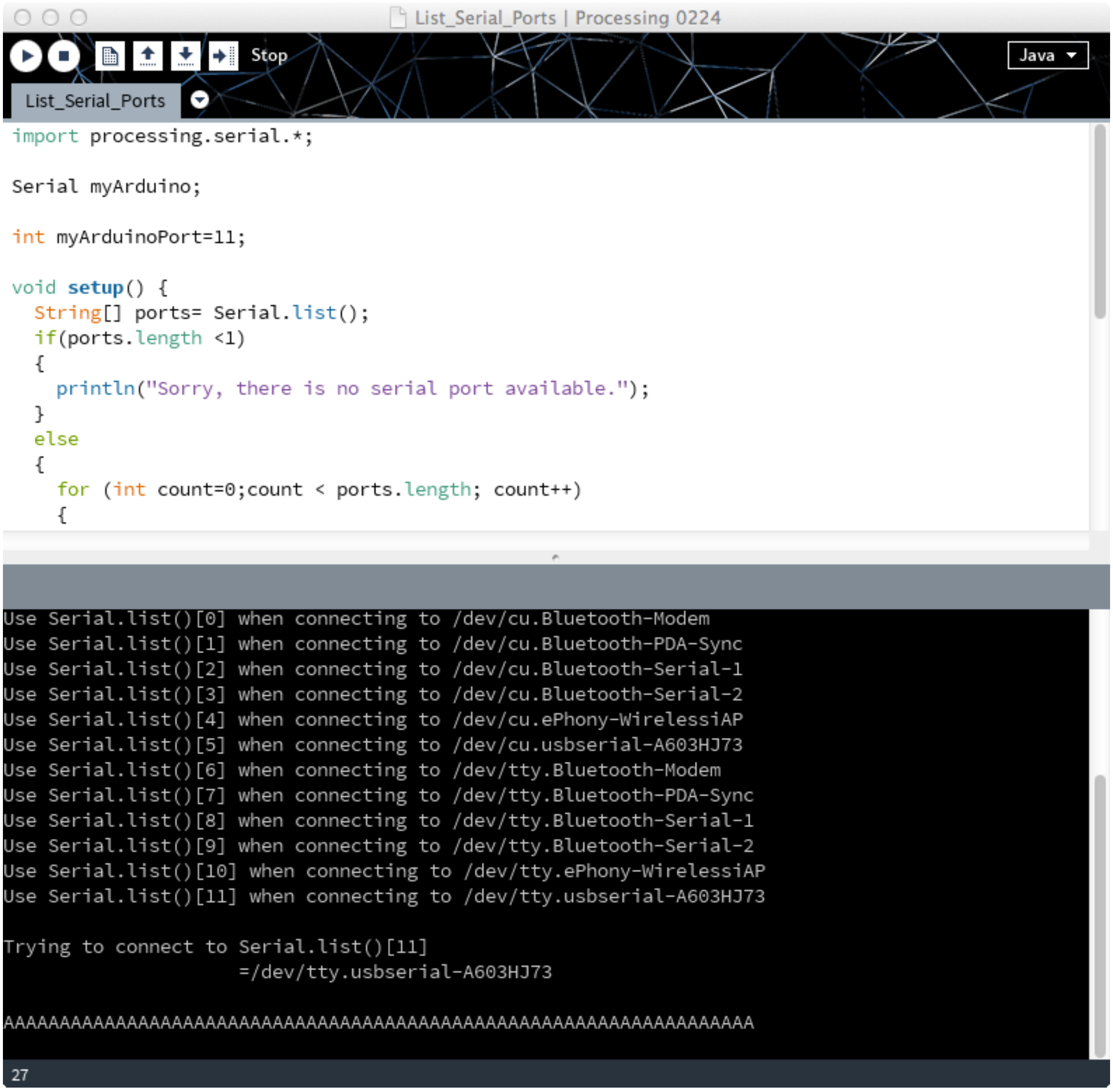
```
import processing.serial.*;

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 leter A to the serial port.

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

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

©Erik Verberne (erik@verberne.net)

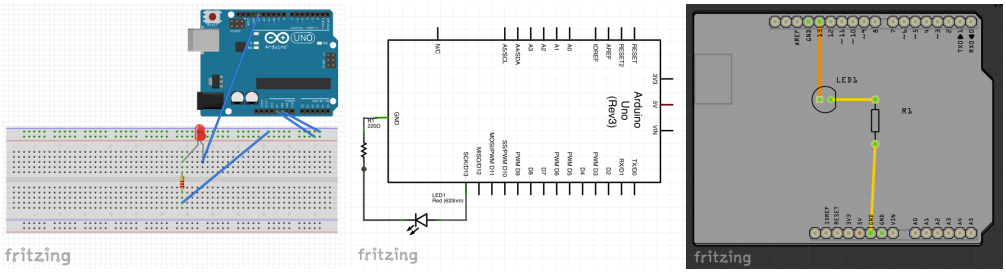
If you load both sketches (in Processing and Arduino) and if you've selected the right serial port, you will see the following output in Processing:



Pict. 2 Sample output from Processing

10. 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.

10.1. 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.

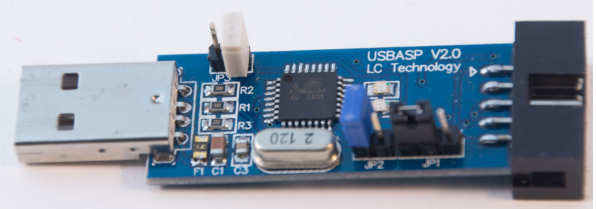
11. Programming an Arduino Board through USB

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

Programming a sketch sample

- Connect your Arduino to your computer using an 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.

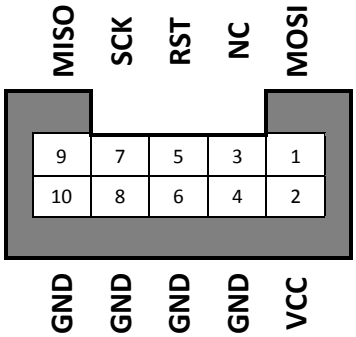
12. USBasp v2.0 programmer



12.1. Links USBasp v2.0 programmer

- Drivers Windows (none needed for Linux or 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/

12.2. Connections USBasp v2.0 programmer



12.3. 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.

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.

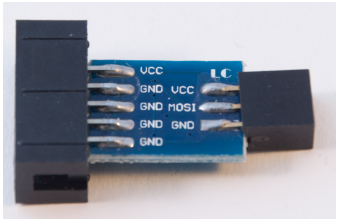
[1] This setup can be simplified by using an AVR ISP to ICSP adapter as described in the next chapter.

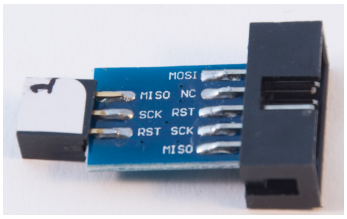
Programming a sketch sample

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.

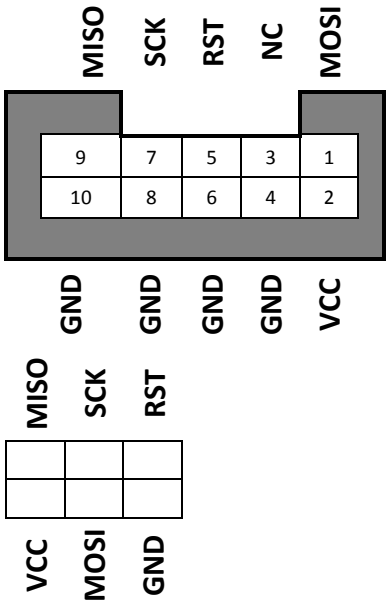
13. AVR ISP to ICSP Adapter





With this AVR ISP to ICSP adapter you can directly connect a USBasp programmer with the ICSP connector on a Arduino board.

13.1. Connections AVR ISP to ICSP Adapter

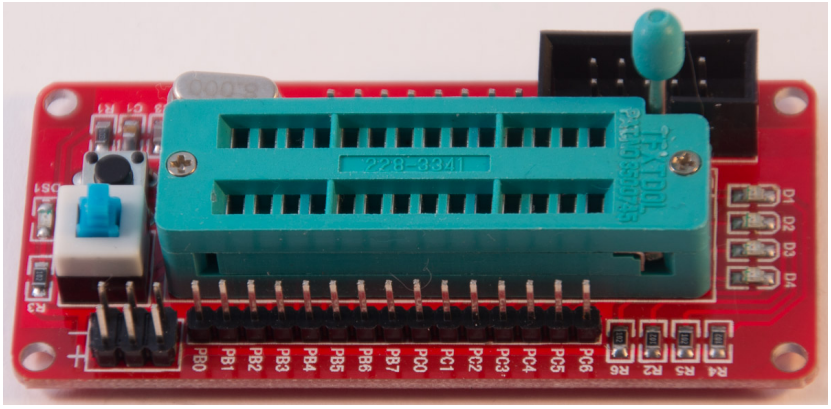


13.2. Sample Setup USBASP v2.0 programmer

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

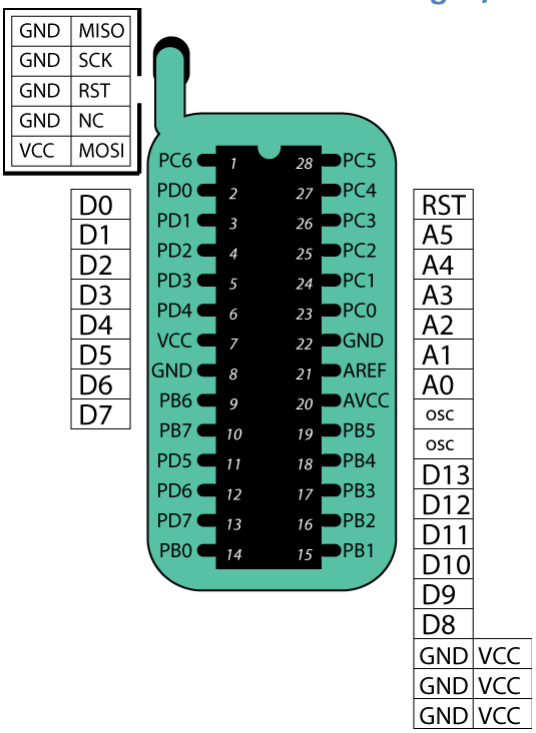
- Connect USBasp to a USB port.
- Connect AVR ISP to ICSP adapter with a flat cable to the USBasp.
- Connect other end of the AVR ISP to ICSP adapter to the ICSP headers on the Arduino UNO (with the GND pin facing the Atmega328).

14. 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 (ie without a serial connection through USB). Use the layout below to match the Atmega328 to Arduino pin assignment.

14.1. Connections AVR Atmega8/168/328 Development board



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

- Replace 8 MHz crystal with 16 MHz.
- Place Atmega328 in ZIF-socket.
- Connect flatcable 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 op with AVRdude errors.

15. Selfmade Attiny 45/85 ISP adapter

A selfmade Attiny45/85 – ISP adapter cable on a piece of breadboard.

15.1. Connections Selfmade Attiny 45/85 ISP adapter

15.2. Sample Programming a Attiny45

Place the Attiny on a solderless breadboard, place the double row of header pins of the Attiny45/85 adapter on the solderless breadboard with pin 1 alligned with pin 1 of the Attiny. Connect the 10 pins female connector in the ISP connector of an 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.

Sound

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

16. Buzzer

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

16.1. Specifications Buzzer

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

16.2. Connections Buzzer

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

16.3. Libraries needed for Buzzer

None needed.

16.4. Sample Buzzer

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

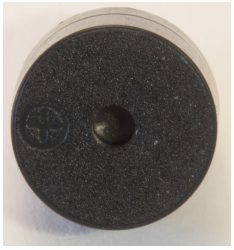
Sample Connections

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

Sample Sketch

```
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);
}
```

17. Piezo Speaker



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

17.1. Specifications Piezo Speaker

17.2. Connections Piezo Speaker

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

17.3. Libraries needed for Piezo Speaker

None needed.

17.4. Sample Piezo Speaker

The following sketch plays a simple melody.

Sample Connections

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

Sample Sketch

```
int speakerPin = 9;

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);
  }
}
```

LED (displays)

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

18. Onboard LED D13

18.1. Specifications Onboard LED D13

- Onboard, connected to D13.

18.2. Connections Onboard LED D13

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

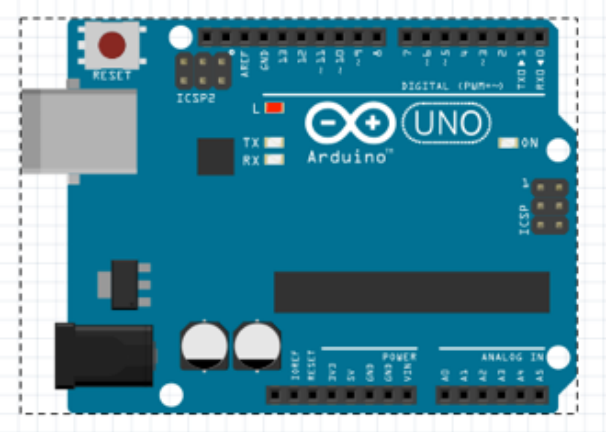
18.3. Libraries needed for Onboard LED D13

None needed.

18.4. Sample Onboard LED D13

Sample Connections

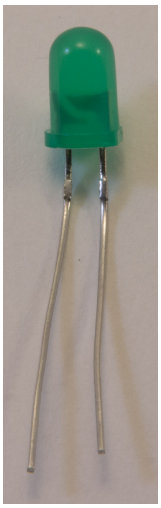
- No connections needed.



Sample Sketch

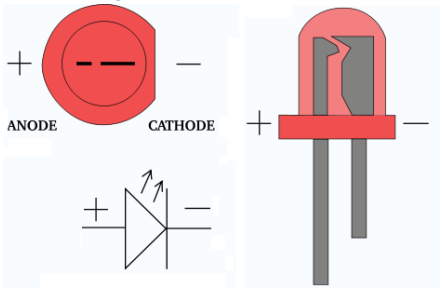
```
int led = 13;

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

void loop()
{
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

19. 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.

19.1. Specifications LED



Most LED have a maximum allowed Voltage of 1.8-2.0 V and a maximum allowed Current of 20 mA. The ouput 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

19.2. 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

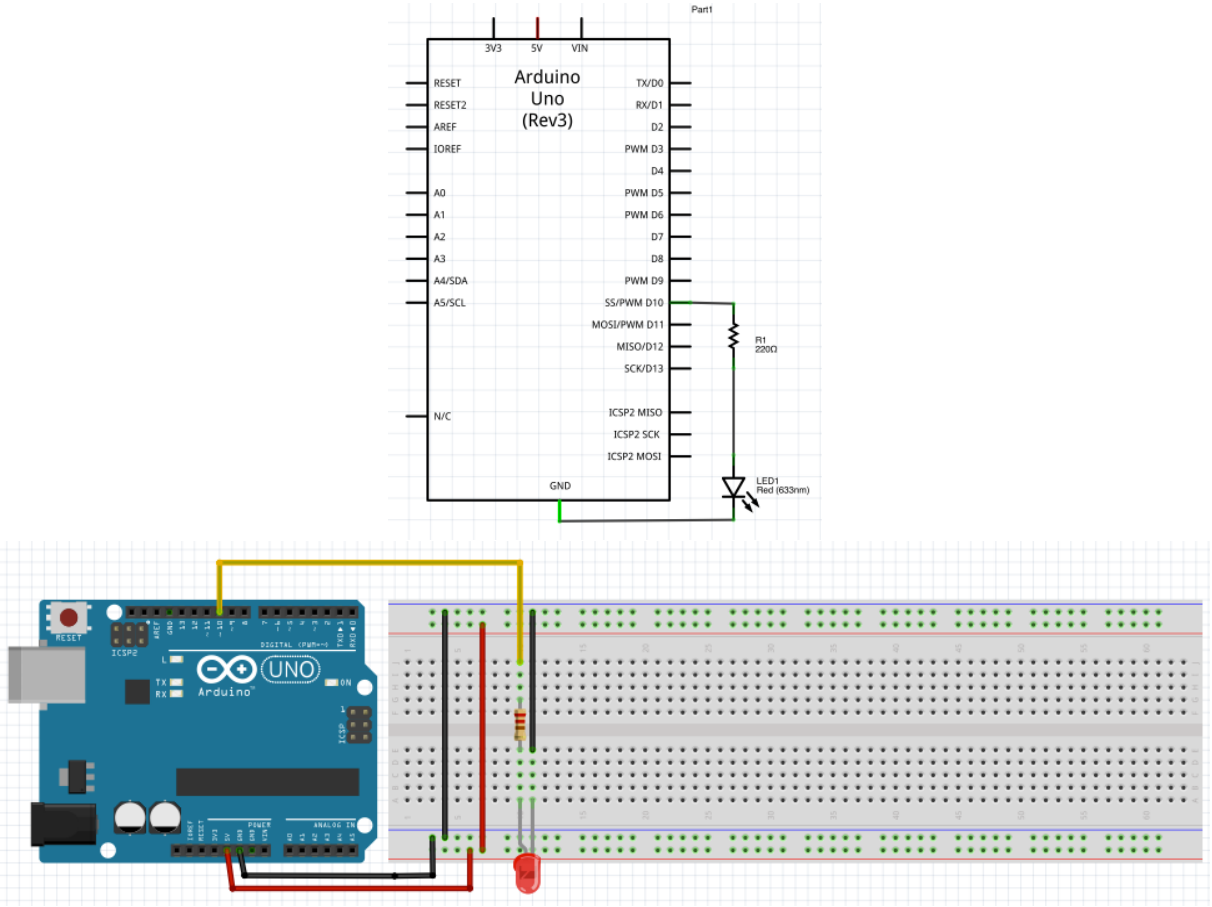
19.3. Libraries needed for LED

None needed.

19.4. Sample LED

Sample Connections

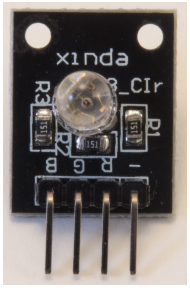
- Connect Cathode with GND.
- Connect Anode with one end of a Resistor of 220 Ohm.
- Connect other end of the Resistor with D10.



Sample Sketch

```
int led = 10;

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

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

20. RGB LED board



A 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.

20.1. 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. Its has a common Cathode.

20.2. Connections RGB LED board

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	-		

20.3. Libraries needed for RGB LED board

None needed.

20.4. Sample RGB LED board

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

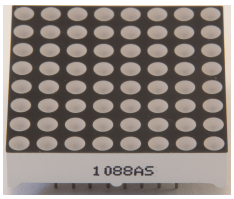
Sample Connections

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

Sample Sketch

```
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);
}
```

21. 8x8 DOT Matrix 1088AS

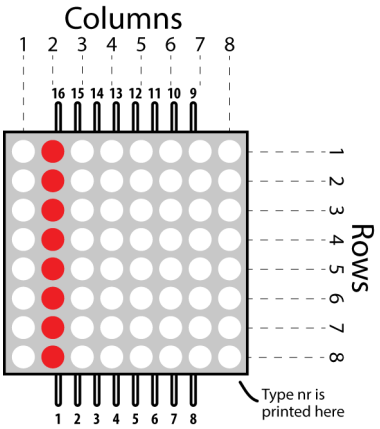


This 8x8 DOT Matrix is used in chapter ”23 8x8 DOT matrix with MAX7219 chip”.

21.1. Specifications 8x8 DOT Matrix 1088AS

- 64 RED LED’s 3mm.
- Row Cathode
- Column Anode

21.2. Connections 8x8 DOT Matrix 1088AS



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

21.3. Libraries needed for 8x8 DOT Matrix 1088AS

None needed.

21.4. 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.

This sketch will drive a whole column at a time.[1]

Sample Connections

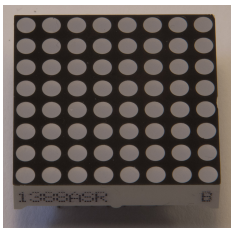
- Connect 1, 2, 5, 7, 8, 9, 12 and 14 to one end of a 220 ohm resistor each.
- Connect the other ends of the 8 resistors to GND.
- Connect 13 (Row 1) to D6
- Connect 3 (Row 2) to D7
- Connect 4 (Row 3) to D8
- Connect 10 (Row 4) to D9
- Connect 6 (Row 5) to D10
- Connect 11 (Row 6) to D11
- Connect 15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

Sample Sketch

```
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'l need some extra IC's, MAX7219 or 2 shift-registers (74HC595) and some transistors to deliver more current.

22. 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 ”23 8x8 DOT matrix with MAX7219 chip”.

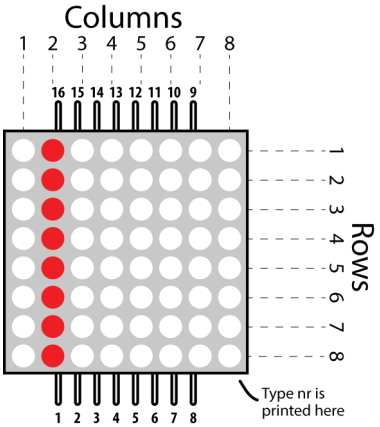
22.1. Specifications 8x8 DOT Matrix 1388ASR

- 64 RED LED’s 3mm.
- Row Anode
- Column Cathode
- V_f=1.8
- I_f=20 mA.

22.2. Datasheet 8x8 DOT Matrix 1388ASR

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

22.3. 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	

Pin nr	Row	Column
13		Cathode Column 1
14	Anode Row 2	
15		Cathode Column 7
16		Cathode Column 8

22.4. Libraries needed for 8x8 DOT Matrix 1388ASR

None needed.

22.5. 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.

This sketch will drive a whole row at a time.

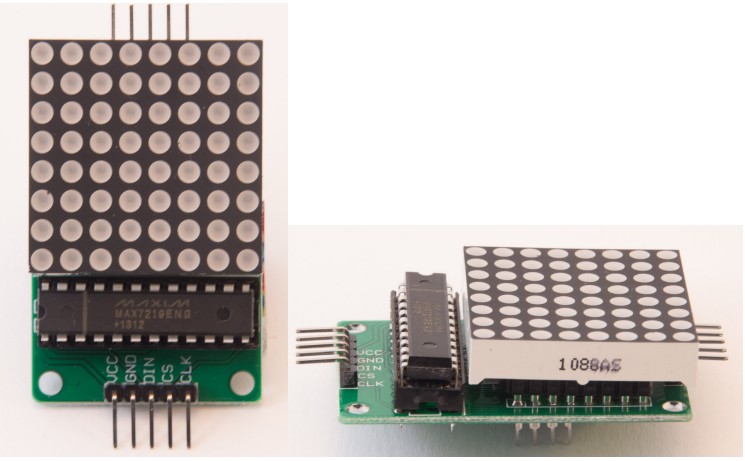
Sample Connections

- 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

Sample Sketch

```
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);
  }
}
```

23. 8x8 DOT matrix 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..

23.1. Specifications 8x8 DOT Matrix 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).

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

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

23.3. Connections 8x8 DOT Matrix 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

23.4. Libraries needed for 8x8 DOT Matrix with MAX7219 chip

I've been using the following 3 libraries

- Serial Peripheral Interface library included in the Arduino IDE.
- Max72xxPanel to address the Max7219 LED display driver. Created by Mark Ruys (mark@paracas.nl)
  https://github.com/markruys/arduino-Max72xxPanel
- AdafruitGFX to create graphics on all sorts of displays. Created by Adafruit.
  https://github.com/adafruit/Adafruit-GFX-Library

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 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 of. 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 cas, 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. 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

23.5. Sample 8x8 DOT Matrix 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).

Sample Sketch

```
#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);
    }
  }
}
```

24. Single Digit 7-Segment Display

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

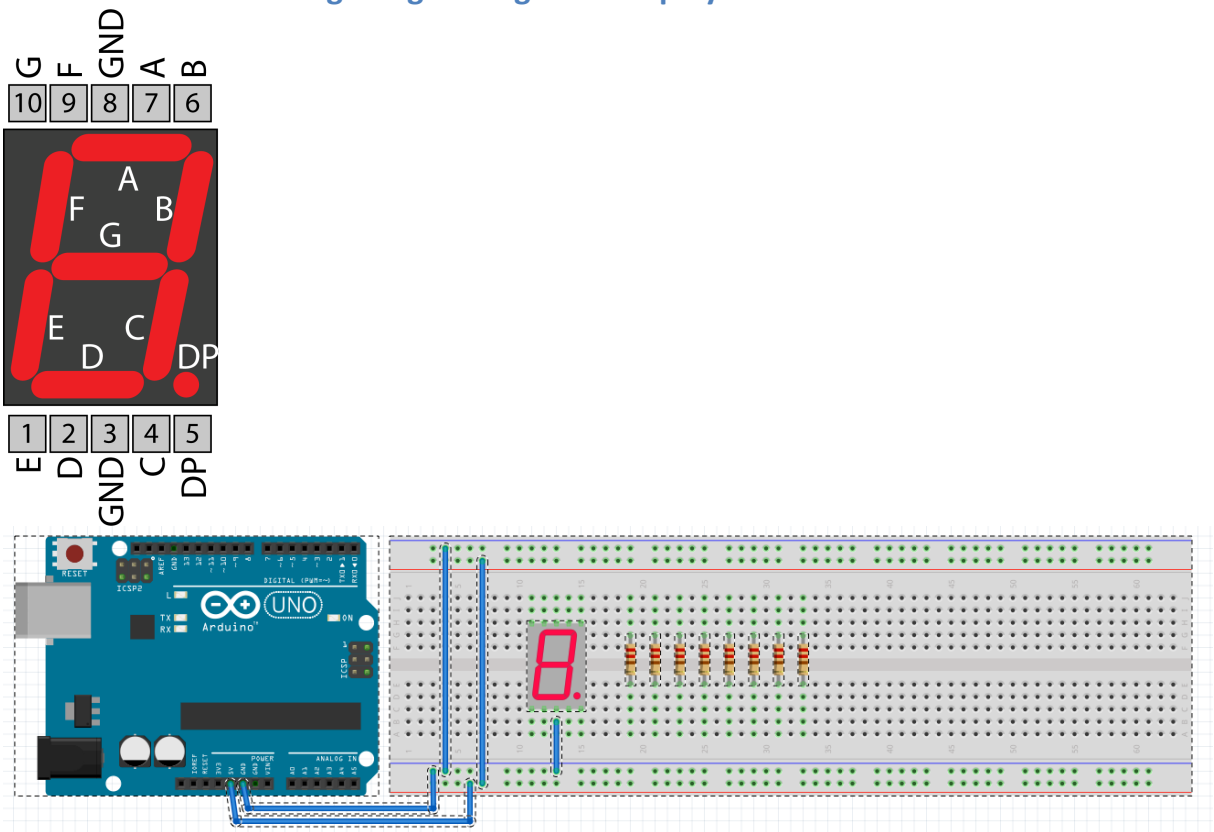
24.1. Specifications Single Digit 7-Segment Display

- 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).

24.2. Datasheet HS120561K Single Digit 7-Segment Display

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

24.3. 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

24.4. Libraries needed for Single Digit 7-Segment Display

- None needed.

Explanation

The following table shows wich 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

24.5. Sample Single Digit 7-Segment Display

The following sketch counts down from 9 to 0 repeatedly.

Sample Connections

- Connect 3 and 8 to GND.
- Connect one end of a 220 ohm resistor to pin 1 and the other end to D6.
- Connect one end of a 220 ohm resistor to pin 2 and the other end to D5.
- Connect one end of a 220 ohm resistor to pin 4 and the other end to D4.
- Connect one end of a 220 ohm resistor to pin 5 and the other end to D9.
- Connect one end of a 220 ohm resistor to pin 6 and the other end to D3.
- Connect one end of a 220 ohm resistor to pin 7 and the other end to D2.
- Connect one end of a 220 ohm resistor to pin 9 and the other end to D7.
- Connect one end of a 220 ohm resistor to pin 10 and the other end to D8.

Sample Sketch

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

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

25. 4 Digits 7-Segment Display

This 4 Digits display has 7 segments and 1 decimal point per digit.

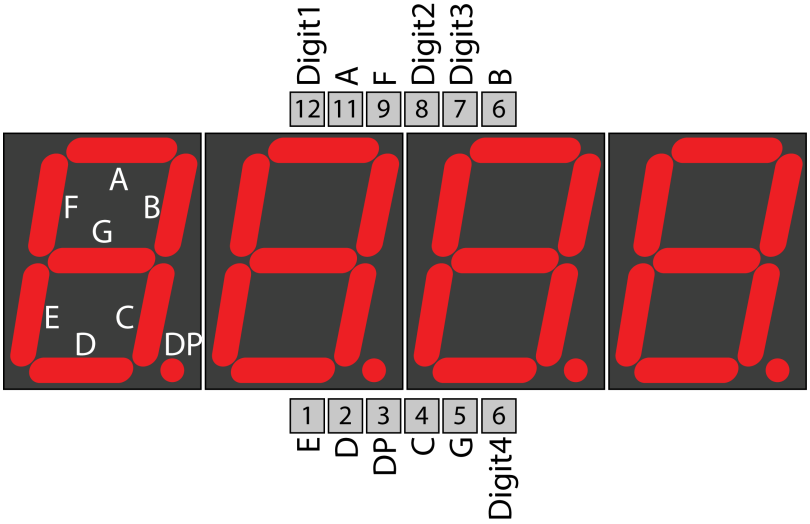
25.1. Specifications 4 Digits 7-Segment Display

- 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.

25.2. Datasheet HS420561K Single Digit 7-Segment Display

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

25.3. Connections 4 Digits 7-Segment Display



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

25.4. Libraries needed for 4 Digits 7-Segment Display

- Seven Segment Display Library from Dean Reading deanreading@hotmail.com. http://playground.arduino.cc/Main/SevenSegmentLibrary

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 comman 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() mus 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.

25.5. Sample 4 Digits 7-Segment Display

The following sketch counts from 0.00 to 99.99 seconds repeatedly.

Sample Connections

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

Sample Sketch

```
#include "SevSeg.h"

SevSeg sevseg;

unsigned long timer;
int CentSec=0;

void setup()
{
  sevseg.Begin(0,2,3,4,5,6,7,8,9,10,11,12,13);
  sevseg.Brightness(100);
  timer=millis();
}

void loop()
{
sevseg.PrintOutput();

  //Check if 10ms has elapsed
  unsigned long mils=millis();
  if (mils-timer>=10)
  {
    timer=mils;
    CentSec++;
    if (CentSec==10000)
    { // Reset to 0 after counting for 100 seconds.
      CentSec=0;
    }
    sevseg.NewNum(CentSec,(byte) 2);
  }
}
```

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

26. 8 Digits 7-Segment Display with TM1638 chip

8 Digits 7 SEG display

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

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

26.2. Datasheet TM1638 chip

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

26.3. 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.

26.4. Libraries needed for8 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 a 8x8 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.

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

This sketch repeatedly counts down from 10 to 0 (not very accurate), then blinks the text CAFE.

Sample connections

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

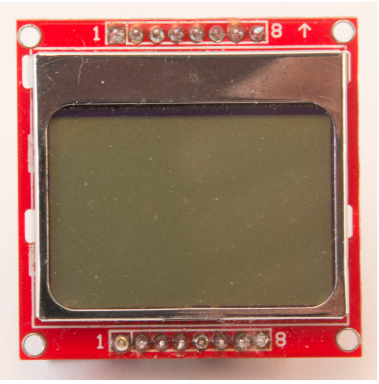
Sample sketch

```
#include <TM1638.h>

TM1638 module(8, 9, 7);

long value = 1000;

void setup()
{
  module.setDisplayToDecNumber(value, 4);
  delay(1000);
}

void loop()
{
 module.setDisplayToDecNumber(value, 4);
 delay(7);
 value--;
 if (value<0)
 {
   value=1000;
  for (int count=0;count<4;count++)
  {
    module.clearDisplay();
    delay(100);
    module.setDisplayToString("  CAFE  ");
    delay(100);
  }
  module.setDisplayToDecNumber(value,4);
  delay(1000);
 }
}
```

LCD displays

In this section you will find a couple of LCD displays.

27. 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 stil 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,-).

27.1. Specifications Nokia 5110/3310 LCD

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

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

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

27.3. Connections Nokia 5110/3310 LCD

Be carefull, 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

27.4. Libraries needed for Nokia 5110/3310 LCD

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

- AdafruitPCD8544 to address the PCD8544 display.
  https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library
- AdafruitGFX to create graphics on all sorts of displays, including the PCD8544 display.
  https://github.com/adafruit/Adafruit-GFX-Library
  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 <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 everytime you want the changes to mydisplay's buffer to be seen on your display.

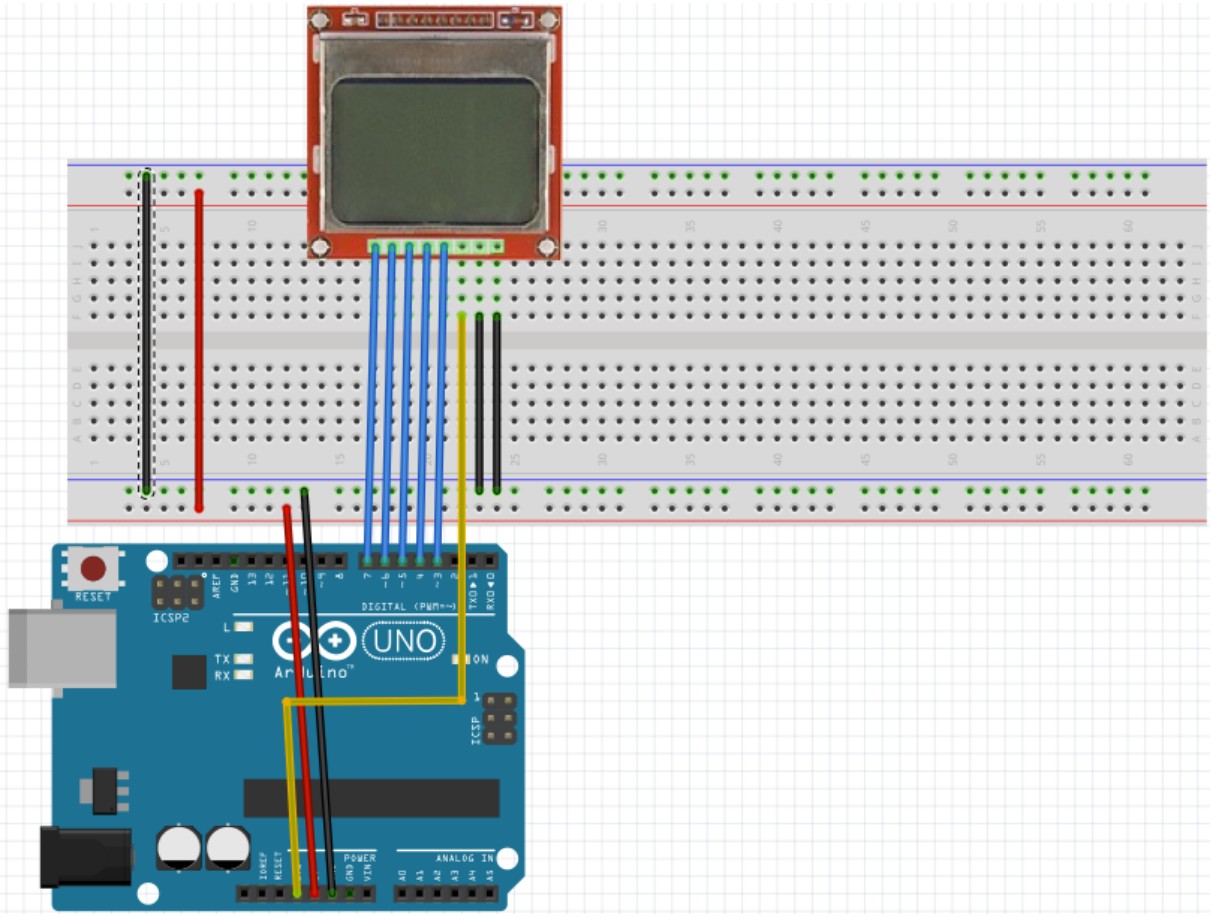
As with other libraries, information about other methods (functions) you can use, can be found in the corresponding library header files *.h in the library folder.

27.5. Sample Nokia 5110/3310 LCD display

This sample script displays a splash screen (AdaFruit logo) for 1 seconds and repeats to show the text “PCD8544 test” and a circle-outline.

Sample Connections

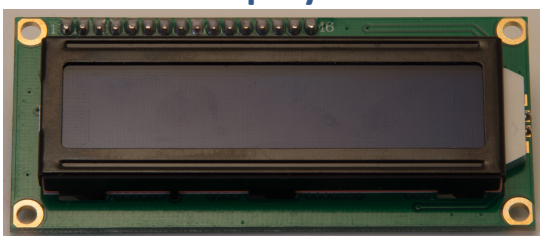
- Connect RST with D7.
- Connect (S)CE with D6.
- Connect DC with D5.
- Connect DIN with D4.
- Connect CLK with D3.
- Connect VCC with 3.3 V
- Connect Light with GND
- Connect GND with GND

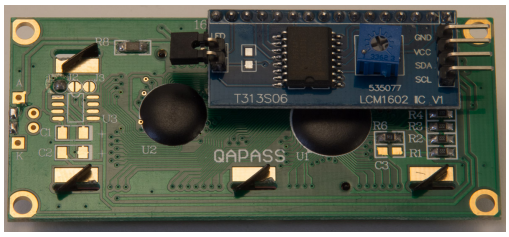


Sample Sketch

```
int CLK=3;
int DIN=4;
int DC=5;
int CE=6;
int RST=7;
#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);
}
```

28. 16x2 Display with LCM1602 chip





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 equiped with a backpack with a LCM1602 chip, so it can be controlled through I²C. I²C is a serial connection using 2 wires besides VCC and GND.

28.1. Specifications

- GDM1602X 16x2 display.
- Blue backlight.
- HD44780 parallel interface chipset.
- LCM1602 I²C.

28.2. Datasheets 16x2 display with LCM1602 chip

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

28.3. Connections 16x2 Display with LCM1602 chip

Pin nr	Name	Description	Arduino pin
1	SCL	I²C Clock	SCL (A5)
2	SDA	I²C Data	SDA (A4)
3	VCC	5 V	5V
4	GND	Ground	GND

28.4. Libraries need for 16x2 Display with LCM1602 chip

- Liquid Crystal I²C library from F. Malpartida (https://bitbucket.org/fmalpartida). https://bitbucket.org/fmalpartida/new-liquidcrystal/downloads
- 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-Intergrated Circuit (I²C) and Two Wire Interface (TWI) library.

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

Include the Liquid Crystal I²C library.

```
LiquidCrystal_I2C mylcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
```

Create 'mylcd' 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). I don't understand the other values, but hey it works!

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

Initialize your display as a 16 rows, 2 columns display and turn on backlight.

```
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 sketch like: http://blog.jacobean.net/?p=653.

28.5. Sample 16x2 Display with LCM1602 chip

The following sketch blinks the backlight 3 times, shows the text “Hello World!” on the first row, waits for 1 second and then shows 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.

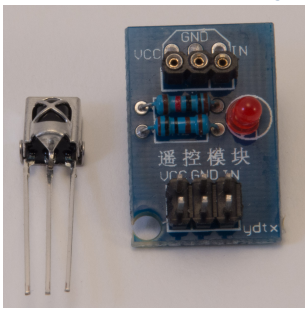
Sketch

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

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
void setup()
{
  lcd.begin(16,2);
  for(int i = 0; i< 3; i++)
  {
    lcd.backlight();
    delay(250);
    lcd.noBacklight();
    delay(250);
  }
  lcd.backlight();

  lcd.setCursor(0,0);
  lcd.print("Hello, world!");
  delay(1000);
  lcd.setCursor(0,1);
  lcd.print("Arduino is nice!");
}

void loop()
{
}
```

Wireless communication

In this section you will find several components used for wireless communication, using Infrared, Radio Frequencies and Bluetooth.

29. IR sensor (receive)



29.1. Specifications IR Sensor (receive)

- 1338B IR sensor

Backgroud 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

29.2. 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

29.3. Libraries needed for IR Sensor (receive)

- IRremote library created by Ken Shirriff.
  http://github.com/shirriff/Arduino-IRremote

Library use explanation

```
#include <IRremote.h>
```

Include the Infra Red remote library from Ken Shirrif.

```
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 reveived, then myresults.value will be printed to the serial port and the 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.

29.4. 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.

Sample Sketch

```
#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
  }
}
```

30. 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 (VCC and GND are swapped),

30.1. 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	D11
2	VCC	5 V	5 V
3	GND	Ground	GND

31. 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.

31.1. 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.
- Powered by 1 CR 2025 battery.

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

31.2. Sample IR remote control 'Car'

The following sample prints the name of the pressed button to the serial port.

Sample Connections

For connections see "29.2 Connections IR Sensor (receive)".

Sample Sketch

```
#define BAUDRATE 9600
#include <IRremote.h>
int RECV_PIN = 11;
IRrecv irrecv(RECV_PIN);
decode_results results;
int nobuttons=21;

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
};

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
  }
}
```

32. 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.

32.1. 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.

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

32.2. Sample IR remote control 'Keyes'

The following code is the array-declaration for this remote.

```
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
};
```

33. 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.

33.1. 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.

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

33.2. Sample IR Remote Control 3

The following code is the array-declaration for this remote.

```
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
};
```

34. IR LED (send)

34.1. Specifications IR LED (send)

34.2. Connections IR LED (send)

Pin nr	Name	Description	Arduino pin
1	Cathode		GND
2	Anode		Any Digital PWM port through 330 ohm resistor

34.3. Libraries needed for IR LED (send)

- Infrared remote library from??

```
#include <IRremote.h>
```

??

```
IRsend irsend; //Connect IR LED to D3!!
```

??

```
irsend.sendNEC(16720605, 32);
```

??

34.4. Sample IR LED (send)

This sketch uses a joystick to determine wich IR code is going to be send to another Arduino!

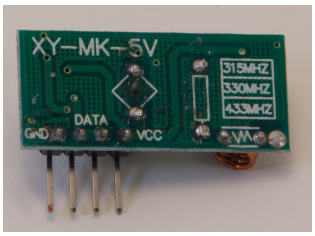
Sample Connections

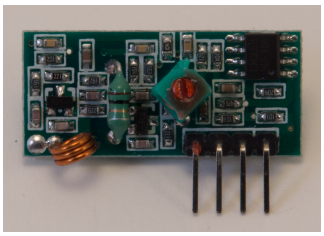
- Connect Cathode with GND.
- Connect Anode with one end of a Resistor of 330 Ohm.
- Connect other end of the Resistor with D10.

Sample Sketch

```
#include <IRremote.h>

IRsend irsend; //Connect IR LED to D3!!
long unsigned   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};

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[5], 32);
    if (sensorx<0) irsend.sendNEC(Button_DEC_code[3], 32);
    if (sensory>0) irsend.sendNEC(Button_DEC_code[1], 32);
    if (sensory<0) irsend.sendNEC(Button_DEC_code[7], 32);
    delay(100);
}
```

35. 315/330/433 MHz RF Receiver XY-MK-5V





Cheap 315/330/433 Mhz RF receiver.

35.1. Specifications 315/330/433 RF Receiver XY-MK-5V

- Voltage: 5V
- locked at 433.92 MHz

35.2. 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

35.3. Connections 315/330/433 RF Receiver XY-MK-5V

Pin nr	Name	Description	Arduino pin
1	VCC	VCC	5V
2	DATA	DATA	IRQ0 (D2) or IRQ1 (D3)
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)

35.4. Libraries needed for 315/330/433 MHz RF Receiver XY-MK-5V[1]

- Radio control library RC-Switch from Suat s@sui.li.
  https://code.google.com/p/rc-switch/

Library use explanation

```
#include <RCSwitch.h>
```

Include the Radio Control library from Suat s@sui.li

```
RCSwitch mySwitch = RCSwitch();
```

Create mySwitch, new instance of the object RCSwitch;

```
mySwitch.enableReceive(ReceiveIRQ);
```

Receiver data is connected to ReceiveIRQ (ReceiveIRQ=0 => D2, ReceiveIRQ=1 => D3).

```
mySwitch.available()
```

Boolean, true if mySwitch received a valid signal.

```
mySwitch.getReceivedValue()
```

Value received.

```
mySwitch.getReceivedBitlength()
```

Bitlength of received message.

[1] Some samples on the Internet use the Virtual Wire library.
http://www.pjrc.com/teensy/td_libs_VirtualWire.html

```
mySwitch.getReceivedProtocol()
```

Protocol of received message.

35.5. Sample 315/330/433 RF Receiver XY-MK-5V

The following sketch displays the received value, bitlength and protocol.

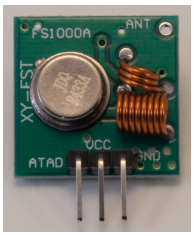
Sample Connections

- 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 datapins to D2 (IRQ0).
- Connect GND to GND.

Sample Sketch

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();

int ReceiveIRQ= 0; // IRQ=0 => D2, IRQ=1 => D3
void setup() {
  Serial.begin(9600);
  mySwitch.enableReceive(ReceiveIRQ);
}

void loop() {
  if (mySwitch.available()) {

    int value = mySwitch.getReceivedValue();

    if (value == 0) {
      Serial.print("Unknown encoding");
    } else {
      Serial.print("Received ");
      Serial.print( mySwitch.getReceivedValue() );
      Serial.print(" / ");
      Serial.print( mySwitch.getReceivedBitlength() );
      Serial.print("bit ");
      Serial.print("Protocol: ");
      Serial.println( mySwitch.getReceivedProtocol() );
    }

    mySwitch.resetAvailable();
  }
}
```

36. 433 MHz RF Transmitter FS1000A



Cheap 433 MHz RF transmitter.

36.1. Specifications 433 MHz RF Transmitter FS1000A

- Voltage: 3-12V
- 433,92 MHz
- > 500M (??)
- < 10Kbps
- Antenna 17 cm, 50 ohm single conductor

36.2. Datasheet 433 MHz RF Transmitter FS1000A

- http://www.567.dk/datasheet/fs1000a.zip
- http://www.electrodragon.com/w/index.php?title=433RF_module

36.3. 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

36.4. Libraries needed for 433 MHz RF Transmitter FS1000A[1]

- Radio control library RC-Switch from Suat s@sui.li.
  https://code.google.com/p/rc-switch/

Library use explanation

```
#include <RCSwitch.h>
```

Include the Radio Control library from Suat s@sui.li

```
RCSwitch mySwitch = RCSwitch();
```

Create mySwitch, new instance of the object RCSwitch;

```
  mySwitch.enableTransmit(Transmitpin);
```

Enable transmition. Transmitpin is an integer corresponding to the pin to which DATA is connected.

```
mySwitch.send(6000, 24);
```

Send the value 6000 with bitlength 24.

[1] Some samples on the Internet use the Virtual Wire library.
http://www.pjrc.com/teensy/td_libs_VirtualWire.html

36.5. Sample 433 MHz RF Transmitter FS1000A

The following sketch displays the received value, bitlength and protocol.

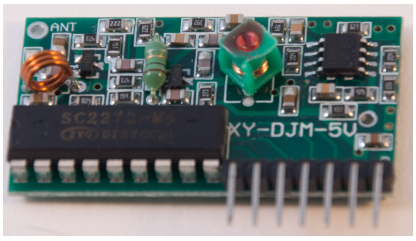
Sample Connections

- Connect ATAD to D10.
- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
#include <RCSwitch.h>

RCSwitch mySwitch = RCSwitch();
int Transmitpin = 10;

void setup() {

  Serial.begin(9600);
  mySwitch.enableTransmit(Transmitpin);
}

void loop() {
  mySwitch.send(6000, 24);
  delay(1000);
  mySwitch.send(5000, 24);
  delay(1000);
}
```

37. RF Wireless kit XY-DJM-5V



37.1. Specifications RF Wireless kit XY-DJM-5V

- Rx: SC2272-4M chip.
- Tx: SC2262 chip.

37.2. Datasheet RF 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

37.3. Connections RF 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

37.4. Libraries needed for RF Wireless kit XY-DJM-5V

None needed.

37.5. Sample RF Wireless kit XY-DJM-5V

The following sketch shows which button is pressed and lights the onboard LED if either button is pressed.

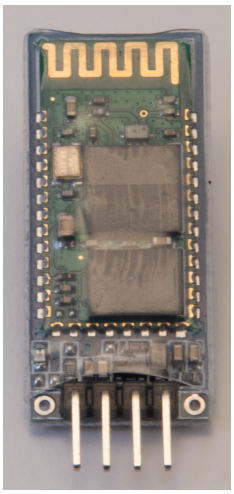
Sample Connections

- 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.

Sample Sketch

```
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);
}
```

38. Bluetooth Keyes BT_Board v2.0



38.1. Specifications Bluetooth Keyes BT_Board v2.0

- HC-06
- Slave-only

38.2. Datasheet Bluetooth Keyes BT_Board v2.0

38.3. Connections Bluetooth Keyes BT_Board v2.0

Pin nr	Name	Description	Arduino pin
1	RxD	Receive data	Tx (D1)[1]
2	TxD	Transmit data	Rx (D0)**Fout! Bladwijzer niet gedefinieerd.**
3	GND	Ground	GND
4	VCC	3.6 – 6 V	5V

38.4. Libraries needed for Bluetooth Keyes BT_Board

- None needed, if you hook up your Bluetooth to Tx (D1) and Rx (D0) the same ports you use to upload your sketches, so you will need to disconnect the Bluetooth board whenever you want to upload a sketch.
- SoftwareSerial libray included with Arduino IDE, if you want to use different pins for Tx and Rx.

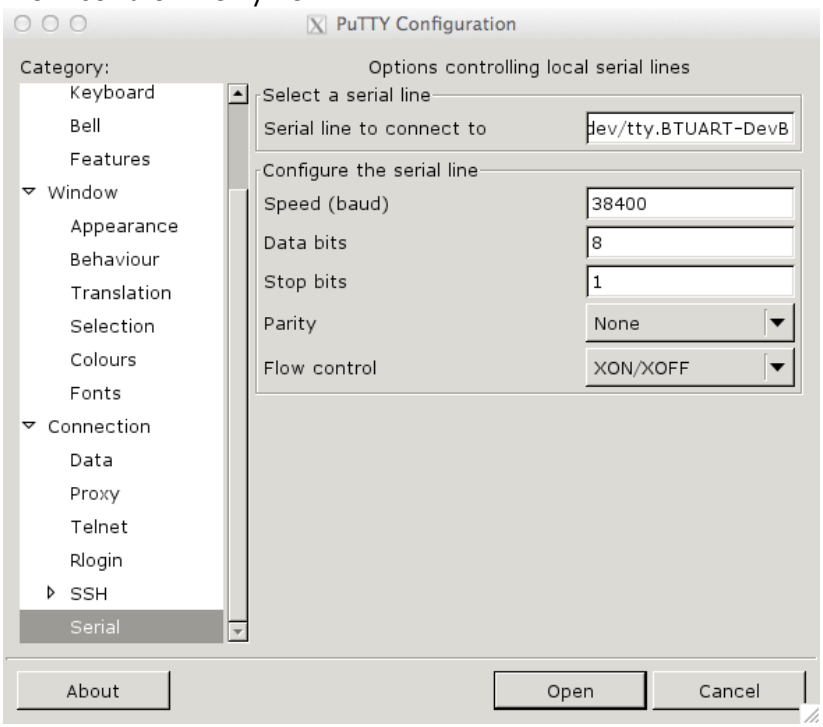
[1] Using D1 and D0, you'll need to unhook your Bluetooth everytime you upload a new sketch! You can use other pins if you use the SofwareSerial library.

38.5. Sample Bluetooth Keyes BT_Board v2.0

The following sketch sends the traditional "Hello world" through Bluetooth to another device.

Sample Connections

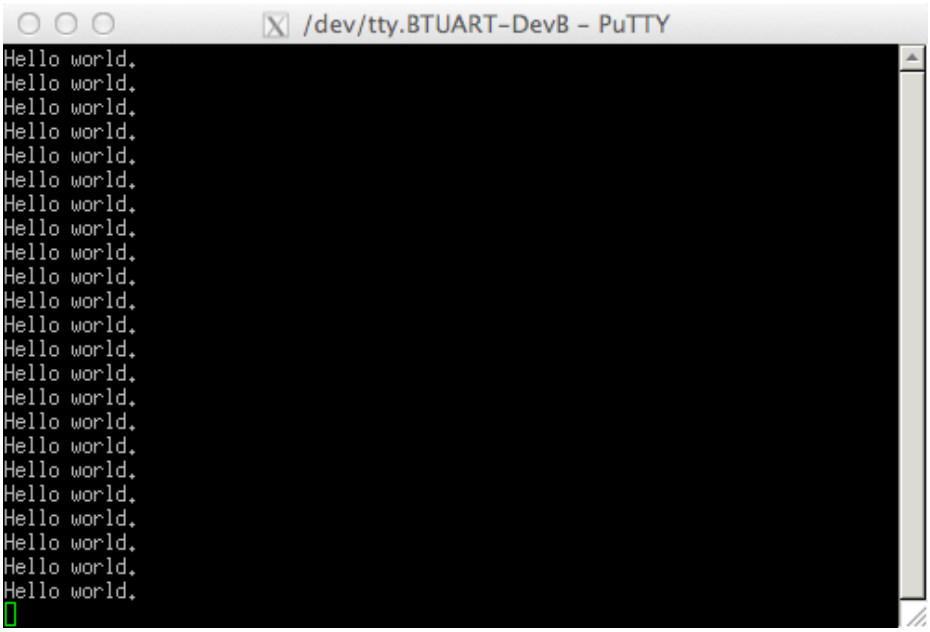
- Connect VCC to 5V.
- Connect GND to GND.
- Connect RxD to Tx (D1).
- Connect TxD to Dx (D0).
- Pair the bluetooth device named BT UART to your computer[1], the passkey is "1234". At this point there is no connection yet, so the red LED is still blinking.
- As soon as the sketch runs, you can make a terminal connection[2] with the BT UART device with the following parameters:
    - Serial line: for example COM3 (Windows) or /dev/tty.BTUART-DevB (Linux or OSx).
    - Speed: 57600 (could be anything from 9600-57600).
    - Data bits: 8.
    - Stop bits: 1.
    - Parity: None
    - Flow control: XON/XOFF



After making a terminal connection, the red light will lit continously and the text Hello World is repeated endlessly.

[1] This could be any desktop, laptop, Android tablet or Android smartphone. It is not possible to pair with an iOS based device such as iPhone, iPad and iPod. Apple will only allow pairing with certified Bluetooth chips!

[2] You could use PuTTY for this, a free and popular terminal connection application.

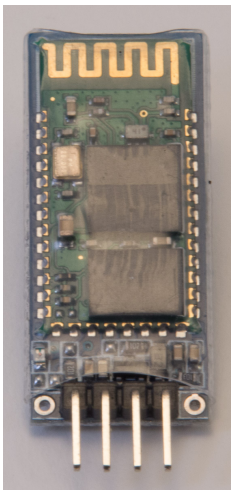


Sample Sketch

```
void setup()
{
  Serial.begin(9600);
  /*With this FIXED speed, your Arduino must communicates with the
BT_BOARD,
   this is not the same speed with wich your computer will communicate with
the BT UART device.*/
}

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

Notice that the serial connection between the Arduino board (Rx/Tx) with the BT_Board (RxD/TxD) uses a fixed speed of 9600, this is different to the speed (from 9600-57600) used by the terminal connection between the computer and the BT UART device (through bluetooth).

39. Bluetooth JY-MCU BT_Board v1.06



39.1. Specifications Bluetooth JY-MCU BT_Board v1.06

- HC-05
- Slave or master

39.2. Datasheet Bluetooth JY-MCU BT_Board v1.06

- AT command set HC-05
  http://www.instructables.com/files/orig/FOR/4FP2/HKZAVRT6/FOR4FP2HKZAVRT6.pdf[1]

39.3. Connections Bluetooth JY-MCU BT_Board v1.06

Pin nr	Name	Description	Arduino pin
1	State	?	?
2	RxD	Receive data	Tx (D1)[2]
3	TxD	Transmit data	Rx (D0)[2]
4	GND	Ground	GND
5	VCC	3.6 – 6 V	5V
6	KEY	Connected to pin 34 of the HC-05	?

39.4. Libraries needed for Bluetooth JY-MCU BT_Board v1.06

- None needed, if you hook up your Bluetooth to Tx (D1) and Rx (D0) the same ports you use to upload your sketches, so you will need to disconnect the Bluetooth board when you want to upload a sketch.
- SoftwareSerial libray included with Arduino IDE, if you want to use different pins for Tx and Rx.

[1] To enter AT commands, you'll need to enter Configuration mode. Instructions are given at the sample sketch.
[2] Using D1 and D0, you'll need to unhook your Bluetooth everytime you upload a new sketch! You can use other pins if you use the SofwareSerial library.

39.5. Sample Bluetooth JY-MCU BT_Board v1.06

The "Hello World" sample from the Bluetooth Keyes BT_Board v2.0, chapter 38, can also be used with the JY-MCU BT_Board v.1.06, but since this board uses the HC-05 chipset, you can also use the BT_Board in a master role, this way making connections to other BT_boards. To set the board in the master-role, you will have to use AT commands.

The following sketch can be used to send AT commands to the BT_BOARD. With these AT commands you can change the settings of the BT_Board, such as:

- AT
  This command doesn't change any settings. After entering this command, the HC-05 will just respond with OK. So this is just a way to test whether the HC-05 is accepting AT commands.
- AT+PSWD=0000
  to change the pass key t0 2987
  (default=1234).
  entering AT+PSWD? shows the current pass key!
- AT+NAME=BT2
  to change the bluetooth device name to BT2
  (default=HC-05).
- AT+UART=115200,1,0
  to change the serial connection parameters to 115200 bps, 1 stop bit, 0 parity bits.
  (default: 9600 bps, 0 stop bits, 0 parity bits).
- AT+ROLE=1
  to set the master role
  (default = 0 => slave role)
- AT+ORGL
  to return to factory settings
  - Role: slave
  - Serial parameters: 38400, 1 stop bit, 0 parity bits
  - passkey: 1234
  - Device name: H-C-2010-06-01

You'll need a the Serial monitor to communicate with the Arduino, so Rx (D0) and Tx (D1) are not available for the BT_Board. To connect the BT_board to your Arduino this sketch uses a SoftwareSerial connection at pins D11 (Rx) and D10 (Tx).

Sample Connections

- Don't connect VCC to 5V yet!!.
- Connect GND to GND.
- Connect RxD to D11 (SoftWareserial Tx).
- Connect TxD to D10 (SoftwareSerial Rx).
- Connect KEY (pin 34) to D9.
- Make sure the HC-05 device is not paired with any device!!
- Load the sketch and remove the USB cable.
- Reconnect the USB cable and after 2 seconds, connect VCC to 5V, this is a signal for the HC-05 to enter configuration mode. The red light will now blink every 2 seconds to show the HC-05 is now in configuration mode and ready to accept AT-commands.
- Open the serial monitor to your Arduino, type the command AT at the input line and press SEND. The Arduino will then send the AT test command to the HC-05 and prints the result in the serial monitor.

Sample Sketch[1]

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(10, 11); // RX | TX

void setup()
{
  pinMode(9, OUTPUT);  // this pin will pull the HC-05 pin 34 (key pin)
HIGH to switch module to AT mode
  digitalWrite(9, HIGH);
  Serial.begin(9600);
  Serial.println("Enter AT commands:");
  BTSerial.begin(38400);  // HC-05 default speed in AT command more
}

void loop()
{
     if (BTSerial.available())
    Serial.write(BTSerial.read());

  // Keep reading from Arduino Serial Monitor and send to HC-05
  if (Serial.available())
    BTSerial.write(Serial.read());
}
```

[1] This sketch and a full description can be found at: http://www.instructables.com/id/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/?ALLSTEPS.

Input sensors

In this section you will find several input sensors, switches, buttons, potentiometers even Nintendo's Nunchuk.

40. Swiches

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).

40.1. Specifications Switches

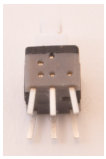
Push button NO

This pushbutton is normally off (NO). A connection will be made when the button is pushed.

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).



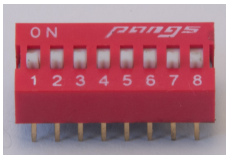
The switch above has 6 terminals that can be switched as 2 changeover switches (front row and back row). Used on a solder less breadboard only 1 switch can be used.

Toggle switch Changeover

Depending on the position of the switch, a connection is made or broken.

Dipswitches

This is a serie of multiple switches in a row, normally used to alter settings.



Tilt Switch Sensor

A tiltwitch is act like all other swithces, only the way to activate is differs. You must tilt the switch instead of push it.

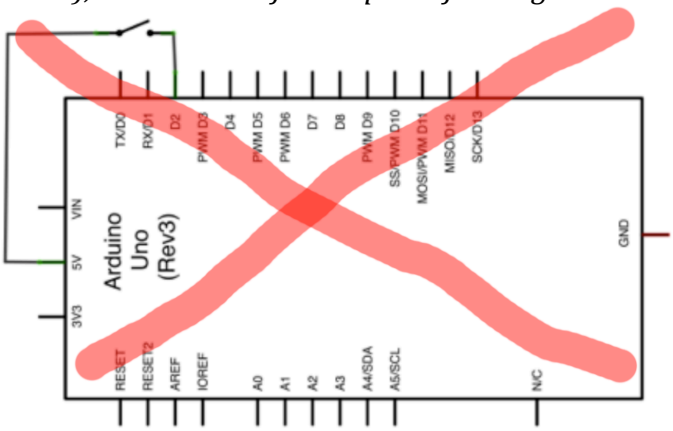
- Straight up, the legs are connected, 90 degrees tilted, the legs are disconnected.

40.2. Libraries needed for Switches

None needed.

40.3. 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.



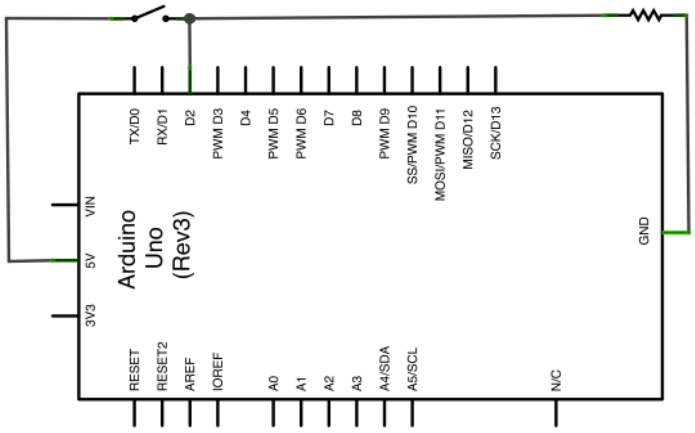
D2 is floating when switch is open and D2 is HIGH when switch is closed. Don't use these schematic!!

To prevent this floating, there are 3 solutions:

- Pulldown resistor of 10K ohm between Ground and the digital input.
- Pullup resistor of 10K ohm between 5V and the digital input.
- Activate an internal pullup resistor (20K ohm).

10K ohm Pulldown resistor

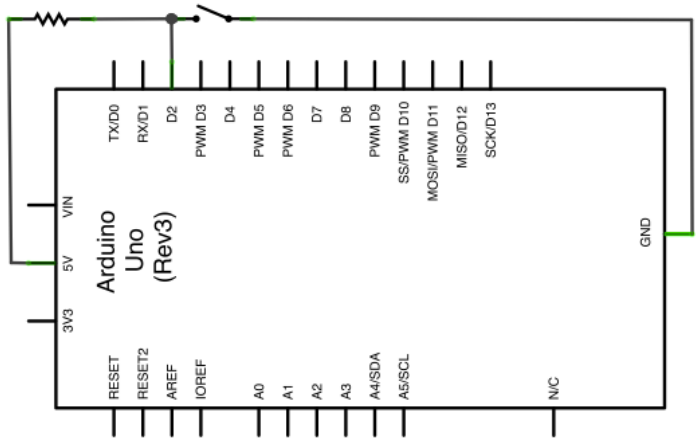
- Connect a 10K ohm resistor between the digital input and GND.
- Connect a switch between the digital input and 5V.
- Closing the switch makes the input HIGH.



D2 is LOW when switch is open and D2 is HIGH when switch is closed.

10K ohm Pullup resistor

- 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 Pulldown schematic, so you might have to change your sketch accordingly.



D2 is HIGH when switch is open and D2 is LOW when switch is closed.

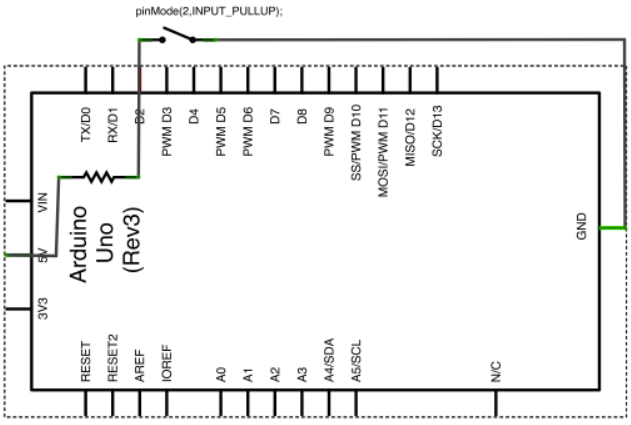
Internal Pullup resistor (20K ohm)

The MEGA328 has internal Pullup resistors of 20K ohm available for every digital I/O. Tot activate such an internal Pullup resistor you have the define the pinmode to INPUT_PULLUP instead of INPUT.

- So activate the internal Pullup resistor.

```
pinMode(2, INPUT_PULLUP);
```

- Connect the switch between the digital input and GND.



D2 is HIGH when switch is open and D2 is LOW when switch is closed.

Sample sketch for switches

```
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
  //pinMode(Button, INPUT_PULLUP_); // when using internal Pullup
}

void loop(){
  Buttonstate = digitalRead(Button);
  //If you change your schematic from Pulldown to Pullup (intern./extern.)
  //you should also alter the following line to 'if (Buttonstate == LOW)'
  if (Buttonstate == HIGH)
  {
    digitalWrite(Led, LOW);
  }
  else
  {
    digitalWrite(Led, HIGH);
  }
}
```

41. Optical Switch 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.

41.1. Specifications Optical Switch ITR8102

-

41.2. Datasheet Optical Switch ITR8102

- http://www.everlight.com/datasheets/ITR8102.pdf

41.3. Connections Optical Switch ITR8102

Pin nr	Name	Description	Arduino pin
1	Anode	+	5V
2	Cathode	E (emitter 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

41.4. Libraries needed for Optical Switch 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).

41.5. Sample Optical Switch ITR8102

The following sketch will switch of the LED on D13 when the light beam is interrupted.

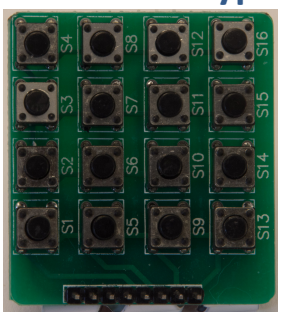
Sample Connections

- 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.

Sample Sketch

```
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);
  }
}
```

42. 4x4 Keypad



42.1. Specifications 4x4 Keypad

- 16 NO push buttons, connected through a 8 pin header row.
- Only one button at a time can be detected.

42.2. 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

42.3. Libraries needed for 4x4 Keypad

- Keypad library from Mark Stanley.
  http://playground.arduino.cc/Code/Keypad#Download

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.

42.4. Sample 4x4 Keypad

The following sketch prints the labels of the keys on the serial monitor.

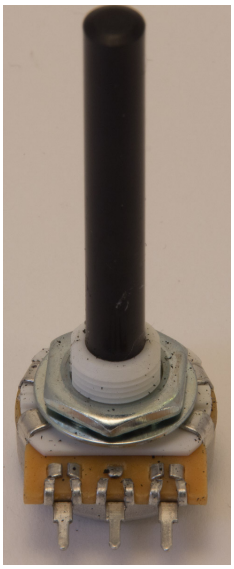
Sample Connections

- 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.

Sample Sketch

```
#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','=', '/'}
};

// Connect keypad ROW0, ROW1, ROW2 and ROW3 to these Arduino pins.
byte rowPins[ROWS] = { 2, 3, 4, 5 };
// Connect keypad COL0, COL1 and COL2 to these Arduino pins.
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);
  }
}
```

43. 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 a analog input device.

43.1. Specifications Potentiometer

In most projects the value of the potentiometer is not critical. A value of 1K to 10 K ohm is very common.

43.2. Connections Potentiometer

Pin nr	Name	Description	Arduino pin
1	left	Ground	GND
2	middle	Signal	Any Analog input port
3	right	VCC	5V

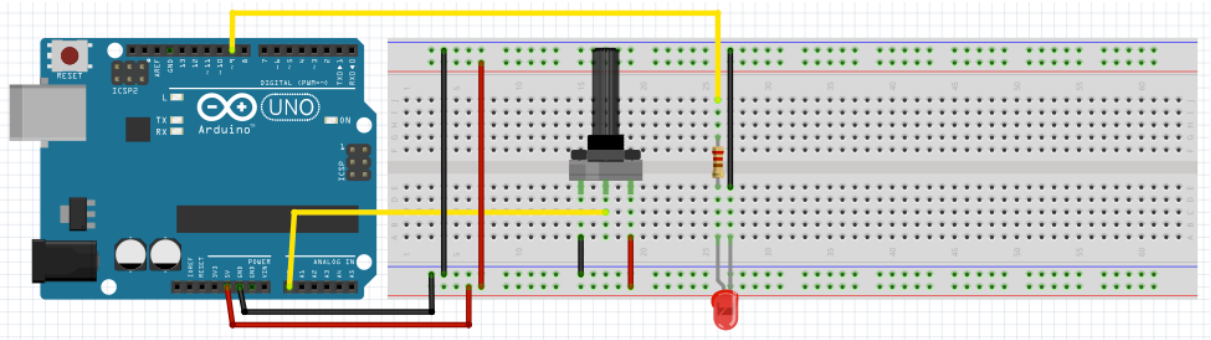
43.3. Libraries needed for Potentiometer

None needed.

43.4. Sample sketch Potentiometer

Connections

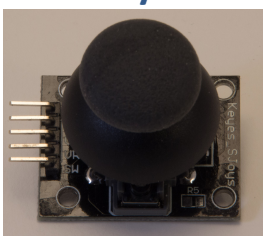
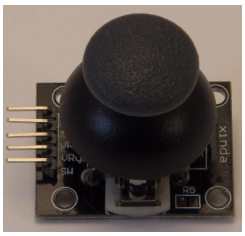
- Connect Left pin with GND.
- Connect Middle pin with A0.
- Connect Right pin with 5V.
- Connect one end of a 220 ohm resistor to D9 and the other end to the Anode of a LED.
- Connect the Cathode of the LED to GND.



Sketch

```
const int analogInPin = A0;
const int LED = 9;
int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup()
{
}

void loop()
{
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  analogWrite(LED, outputValue);
  delay(2);
}
```

44. 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 a 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).

44.1. 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 to PCB, the swith will be closed and connected to GND (LOW signal). SO you should use a pullup 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

44.2. 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

44.3. Libraries needed for Joystick

None needed.

44.4. 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

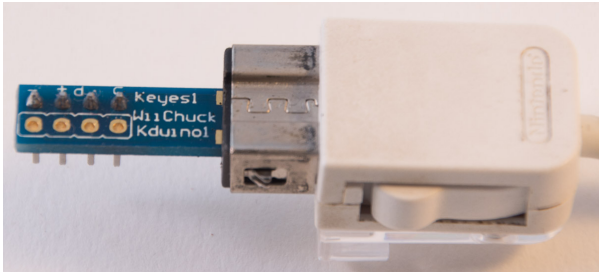
- Connect VCC to 5V.
- Connect GND to GND.
- Connect VRx to A0.
- Connect VRy to A1.
- Connect SW to D5.

Since SW switches D5 to groud, a Pullup resistor is needed. In the following sketch the internal Pullup resistor is activated.

Sample Sketch

```
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);
}
```

45. Nunchuk with connection board



45.1. Specifications Nunchuk with connection board

- 2-Way analog joystick
- 2 buttons
- 3 axis accelerometer
- Communication through Fast I²C at 400 kHz.

45.2. 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)

45.3. 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-Intergrated 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 baudrate for the serial port to something higher than the default 9600. Off course this is only needed if you want to print the reveived 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 nunchuksensors:

- *Joystick:*
  - *mynunchuk.analogX*
  - *mynunchuk.analogY*
- *Accelerometer:*
  - *mynunchuk.accelX*
  - *mynunchuk.accelY*
  - *mynunchuk.accelZ*
- *Z-button: mynunchuk.zButton*
- *C-button: mynunchuk.cButton*

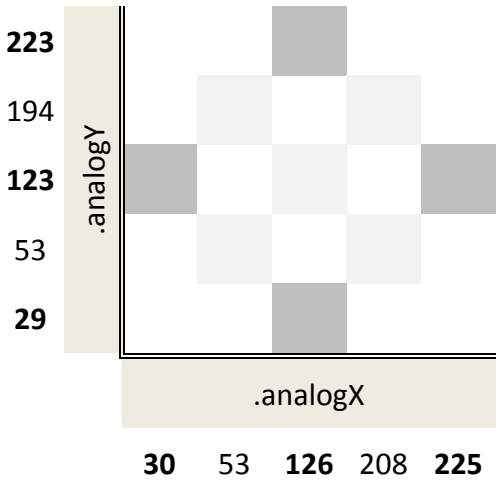
```
• Serial.print(mynunchuk.analogX, DEC);
• Serial.print(' ');
• Serial.print(mynunchuk.analogY, DEC);
• Serial.print(' ');
• Serial.print(mynunchuk.accelX, DEC);
• Serial.print(' ');
• Serial.print(mynunchuk.accelY, DEC);
• Serial.print(' ');
• Serial.print(mynunchuk.accelZ, DEC);
• Serial.print(' ');
• Serial.print(mynunchuk.zButton, DEC);
• Serial.print(' ');
• Serial.println(mynunchuk.cButton, DEC);
```

This code is often used to send the received value from the nunchuk to scripts on your computer (Python, Processing etc..). This way you can use a nunchuk as an input device on a Windows/Mac/Linux computer.

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)

45.4. 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).

Sketch

```
#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);
}
```

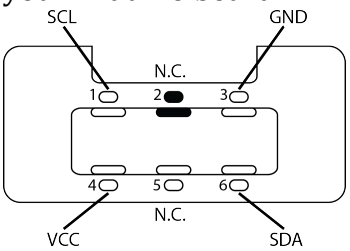
46. Nunchuk without connection board

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

46.1. Connections original connector

If you look inside the original 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)

46.2. 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.

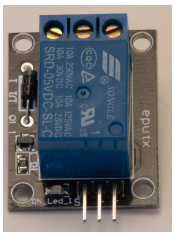
Cut-off the original connector and remove the outer isolation. Inside you'll find 5 isolated wires, of which you need only the following 4.

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

Isolation from higher voltages

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 need some special equipment and special care if it concerns mains electricity. This section describe the use of relays and optocouplers.

47. 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.

47.1. Specifications Relay 5V

- SRD-05VDC-SL-C
- Input: 3-48 V
- Load 10 A:
  - 250 VAC
  - 125 VAC
  - 30 VDC
  - 28 VDC
- 3 Terminals wich you can use as a Changeover switch or 1 NO (normally open) and 1 NC (normally closed) pair.

47.2. Datasheet Relay 5 V

http://www.parallax.com/sites/default/files/downloads/27115-Single-Relay-Board-Datasheet.pdf

47.3. 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 carefull!!!

Pin nr	Name	Description
1	NC	Normally closed
2	CO	Common
3	NO	Normally open

[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.

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.

47.4. Libraries needed for Relay 5V

None needed

47.5. 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 for ever.

Sample Connections[1]

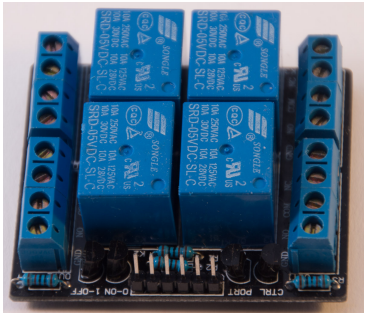
- Send your sketch to the Arduino
- Disconnect your laptop/computer from the Arduino (as a precaution for a faulty relay).Connect S to D12.
- Connect + to 5V.
- Connect - to GND.
- Remove your mains electricity connector from the wall outlet.
- Connect one end of the LIFE wire (coming from a light bulb) to screw 2. Remove just enough isolation, so there is not to 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 an external power source (battery pack).

Sample Sketch

```
int Relay = 12;

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

void loop()
{
     digitalWrite(Relay, HIGH);
     delay(3000);
     digitalWrite(Relay, LOW);
     delay(3000);
}
```

[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.

48. 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.

48.1. 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 wich 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.

48.2. Datasheet 4x Relay 5V Board

http://www.parallax.com/sites/default/files/downloads/27115-Single-Relay-Board-Datasheet.pdf

48.3. 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	K1	Relay K1	Any Digital port
3	K2	Relay K2	Any Digital port
4	K3	Relay K3	Any Digital port
5	K4	Relay 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 carefull!!!

Pin nr	Name	Description
1	GND	Ground
2	NO	Normally Open
3	COM	Common
4	NC	Normaly Closed

Use screw 4 and 3 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 2, 3, 4 or 5 will switch your device to ON on respectively K1, K2, K3 or K4.
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 2, 3, 4 or 5 wil switch your device to OFF on respectively K1, K2, K3 or K4.

48.4. Libraries needed for 4x Relay 5V Board

None needed.

48.5. 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 to D2.
- Connect K2 to D3.
- Connect K3 to D4.
- Connect K4 to 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 to 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 an external power source (battery pack).

Sample Sketch

```
int relay1=2;
int relay2=3;
int relay3=4;
int relay4=5;

void setup()
{
  pinMode(relay1,OUTPUT);
  pinMode(relay2,OUTPUT);
  pinMode(relay3,OUTPUT);
  pinMode(relay4,OUTPUT);
}

void loop()
{
  digitalWrite(relay1,HIGH);
  delay(500);
  digitalWrite(relay2,HIGH);
  delay(500);
  digitalWrite(relay3,HIGH);
  delay(500);
  digitalWrite(relay4,HIGH);
  delay(1500);
  digitalWrite(relay1,LOW);
  delay(500);
```

[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.

```
  digitalWrite(relay2,LOW);
  delay(500);
  digitalWrite(relay3,LOW);
  delay(500);
  digitalWrite(relay4,LOW);
  delay(1500);
}
```

49. Optocoupler MOC3023

This optocoupler isolates the output from your Arduino with the device you want to switch ON/OFF.

49.1. 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.

49.2. Datasheet Optocoupler MOC3023

- http://www.futurlec.com/LED/MOC3023.shtml

49.3. 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	Maint term.	Maint 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

49.4. Libraries needed for Optocoupler MOC3023

None needed.

[1] Calculated with the calculator on the following website: http://led.linear1.org/1led.wiz

49.5. Sample Optocoupler MOC3023

The following sketch turns on a photoflash after pressing the spacebar followed by the Enter key.

Sample Connections

- 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.

Sample Sketch

```
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");
    }
  }
}
```

Sensors

This section describes the use of different kind of sensors, temperature, humidity, distance and more.

50. Temperature Sensor LM35

This sensor measures the temperature in Centigrade.

50.1. Specifications Temperature Sensor LM35

- Linear scale +10 mV/ ^{0}C.
- 0.5 ^{0}C accuracy at +25 ^{0}C.
- Range: -55..+150 ^{0}C

50.2. Datasheet Temperature Sensor LM35

http://www.ti.com/lit/ds/symlink/lm35.pdf

50.3. Connections Temperature Sensor LM35

Pin nr	Name	Description	Arduino pin
1	+Vs	5V	5V
2	Vout	Output	Any analog port
3	GND	Ground	

50.4. Libraries needed for Temperature Sensor LM35

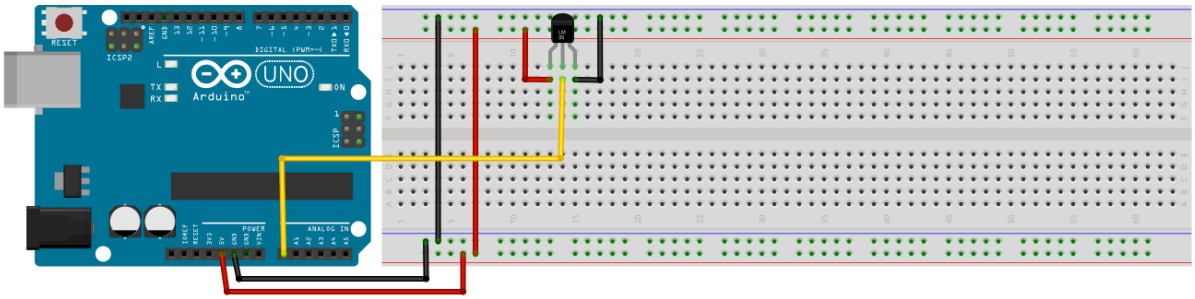
None needed.

50.5. 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);
```

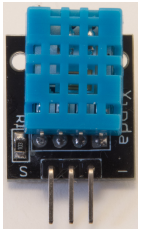
Now we have 1024 steps in 1.1 V, every 10.0 mV is 1 ^{0}C, so the equation should now be:

```
 tempC=(analogRead(tempPin)/(10.0/(1100/1024)));
```

```
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

51. Temperature and Humidity sensor board



This board measures the temperature and humidity.

51.1. Specifications Temperature and Humidity sensor board

- DHT11.
- 2 ^{0}C accuracy

51.2. Datasheet Temperature and Humidity sensor board

http://www.micro4you.com/files/sensor/DHT11.pdf

51.3. Connections Temperature and Humidity sensor

Pin nr	Name	Description	Arduino pin
1	S	Signal	Any analog port
2	?	5V	5V
3	-	Ground	GND

51.4. Libraries needed for Temperature and Humidity sensor board

- DHT11/ DHT21/DHT22 library from Rob Tillaart rob.tillaart@gmail.com
  http://playground.arduino.cc/Main/DHTLib

Library use explanation

```
#include <dht.h>
```

Include the DHT11/DHT21/DHT22 library from Rob Tillaart.

```
dht myDHT;
```

Create myDHT a new instance of the object type dht.

```
myDHT.read11(dht_dpin);
```

Read from analog pin 'dht_dpin'.

```
myDHT.humidity
```

This value is the humidity.

```
myDHT.temperature
```

This value is the measured temperature.

```
delay(800);
```

Delay for next reading. Should be at least 0.8 seconds.

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.

51.5. Sample Temperature and Humidity sensor board

The following sketch sketch measures the humidity and temperature every 0.8 seconds.

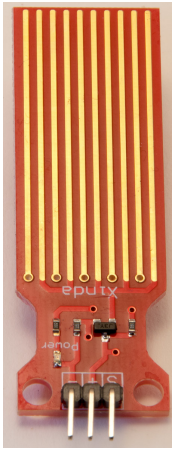
Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
#include <dht.h>

#define dht_dpin A0 //no ; here. Set equal to channel sensor is on

dht DHT;

void setup(){
  Serial.begin(9600);
  delay(300);//Let system settle
  Serial.println("Humidity and temperature\n\n");
  delay(700);//Wait rest of 1000ms recommended delay before
  //accessing sensor
}//end "setup()"

void loop(){
  //This is the "heart" of the program.
  DHT.read11(dht_dpin);

  Serial.print("Current humidity = ");
  Serial.print(DHT.humidity);
  Serial.print("%  ");
  Serial.print("temperature = ");
  Serial.print(DHT.temperature);
  Serial.println("C  ");
  delay(800);//Don't try to access too frequently... in theory
  //should be once per two seconds, fastest,
  //but seems to work after 0.8 second.
}// end loop()
```

52. Water sensor



This sensor can read the liquidlevel in a container.

52.1. Specifications Water sensor

- Sensing area (non-linear): 2,2x4cm

52.2. Connections Water sensor

Pin nr	Name	Description	Arduino pin
1	-	Ground	GND
2	+	5V	5V
3	S	Signal	Any analog port

52.3. Libraries needed for Water sensor

None needed.

52.4. 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.

Sample Sketch

```
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);
}
```

53. Distance Sensor

This module sends a TTL sound and measures the reflection time, so you can calculate the distance to an object.

53.1. Specifications Distance Sensor

- HC-SR04.
- Distance: 2-500 cm.
- Accuracy : 0,3 cm.
- Angle: 15 degrees.

53.2. 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

53.3. Connections for Distance Sensor

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

53.4. Libraries needed for Distance Sensor

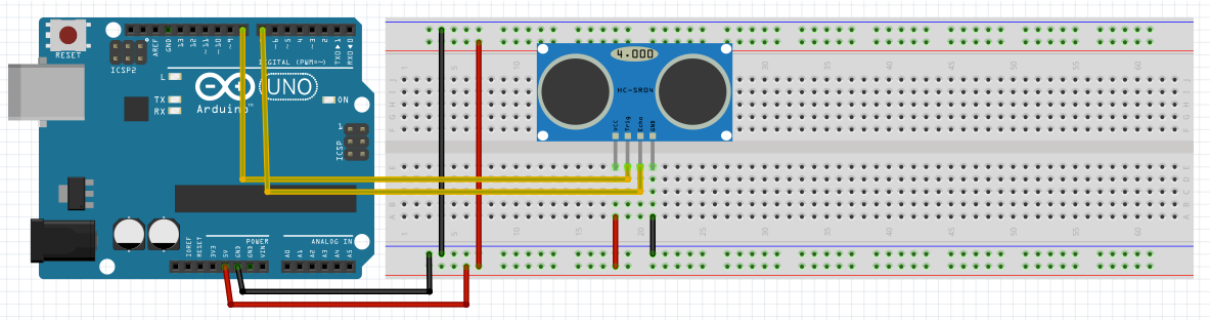
None needed.

53.5. Sample Distance Sensor

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 D8.
- Connect ECHO to D7.
- Connect GND to GND.



Sample sketch

```
#define echoPin 7 // Echo Pin
#define trigPin 8 // Trigger Pin
int maximumRange = 200;
int minimumRange = 0;
long duration, distance;

void setup()
{
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);

  distance = duration/58.2;

  if (distance >= maximumRange || distance <= minimumRange)
  {
    Serial.println("-1");
  }
  else
  {
    Serial.println(distance);
  }
  delay(50);
}
```

53.6. Processing sketch for the Distance Sensor

A nice Processing script to combine with the above Arduino sketch is listed below.

Be carefull to select the right serial port and to close the serial monitor in Arduino IDE, before you start the Processing script.

```
/* 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; // Number of squares to display on screen
int shapeSpeed = 2; // Speed at which the shapes move to new position
 // 2 = Fastest, Larger numbers are slower

//Global Variables
Square[] mySquares = new Square[numOfShapes];
int shapeSize, distance;
String comPortString;
Serial myPort;

/* ----------------------Setup ---------------------------*/
void setup(){
 size(400,400); //Use entire screen size.
 smooth(); // draws all shapes with smooth edges.

 /* Calculate the size of the squares and initialise the Squares array */
 shapeSize = (width/numOfShapes);
 for(int i = 0; i<numOfShapes; i++){
 mySquares[i]=new Square(int(shapeSize*i),height-40);
 }

 /*Open the serial port for communication with the Arduino
 Make sure the COM port is correct - I am using COM port 8 */
 myPort = new Serial(this, Serial.list()[5], 9600);
 myPort.bufferUntil(10);

 //myPort = new Serial(this, "COM8", 9600);
 //myPort.bufferUntil('\n'); // Trigger a SerialEvent on new line
}

/* ------------------------Draw ----------------------------*/
void draw(){
 background(0); //Make the background BLACK
 delay(50); //Delay used to refresh screen
 drawSquares(); //Draw the pattern of squares
}


/* ---------------------serialEvent ---------------------------*/
void serialEvent(Serial cPort){
 comPortString = cPort.readStringUntil('\n');
 if(comPortString != null) {
 comPortString=trim(comPortString);
 println(comPortString);
 /* Use the distance received by the Arduino to modify the y position
```

```
 of the first square (others will follow). Should match the
 code settings on the Arduino. In this case 200 is the maximum
 distance expected. The distance is then mapped to a value
 between 1 and the height of your screen */
 distance = int(map(Integer.parseInt(comPortString),1,50,1,height));
 if(distance<0){
 /*If computer receives a negative number (-1), then the
 sensor is reporting an "out of range" error. Convert all
 of these to a distance of 0. */
 distance = 0;
 }
 }
}


/* ---------------------drawSquares ---------------------------*/
void drawSquares(){
 int oldY, newY, targetY, redVal, blueVal;

 /* Set the Y position of the 1st square based on
 sensor value received */
 mySquares[0].setY((height-shapeSize)-distance);

 /* Update the position and colour of each of the squares */
 for(int i = numOfShapes-1; i>0; i--){
 /* Use the previous square's position as a target */
 targetY=mySquares[i-1].getY();
 oldY=mySquares[i].getY();

 if(abs(oldY-targetY)<2){
 newY=targetY; //This helps to line them up
 }else{
 //calculate the new position of the square
 newY=oldY-((oldY-targetY)/shapeSpeed);
 }
 //Set the new position of the square
 mySquares[i].setY(newY);

 /*Calculate the colour of the square based on its
 position on the screen */
 blueVal = int(map(newY,0,height,0,255));
 redVal = 255-blueVal;
 fill(redVal,0,blueVal);

 /* Draw the square on the screen */
 rect(mySquares[i].getX(), mySquares[i].getY(),shapeSize,shapeSize);
 }
}

/* ---------------------sketchFullScreen----------------------------*/
// This puts processing into Full Screen Mode
boolean sketchFullScreen() {
 return false;
}

/* ---------------------CLASS: Square ----------------------------*/
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;
 }
}
```

54. Photo resistor (LDR)

LDR stands for Light Dependent Resistor. The resistance of a LDR decreases when the light intensity increaces. The accuracy of these devices is very low, they are mainly used to detect is it light or dark.

54.1. Specifications Photo resistor (LDR)[1]

- Resitance range: 5M ohm (dark) – 100 ohm (directly under a bright light).

54.2. Connections Photo resistor (LDR)

Both ends of a LDR are equally the same.

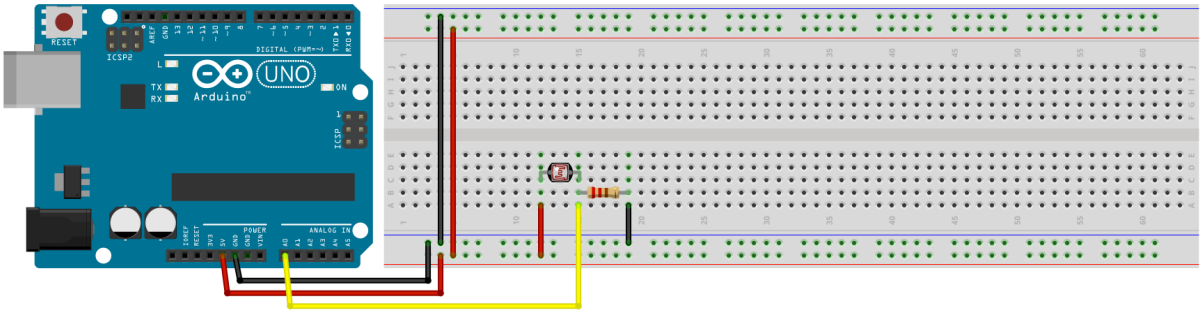
54.3. Libraries needed for Photo resistor (LDR)

None needed.

[1] Background information about LDR's can be found at:
http://learn.adafruit.com/photocells

54.4. 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 resistero to GND.

Sample Sketch

```
int sensorPin = A0;
unsigned int sensorValue = 0;

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

void loop()
{
  sensorValue = analogRead(sensorPin);
  Serial.println(sensorValue, DEC);
  delay(500);
}
```

55. 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 too bad also the IR frequencies that are part of some halogen lights.

55.1. Specifications Flame Sensor (IR photo transistor)

55.2. Datasheet Flame Sensor (IR photo transistor)

55.3. Connections Flame Sensor (IR photo transistor)

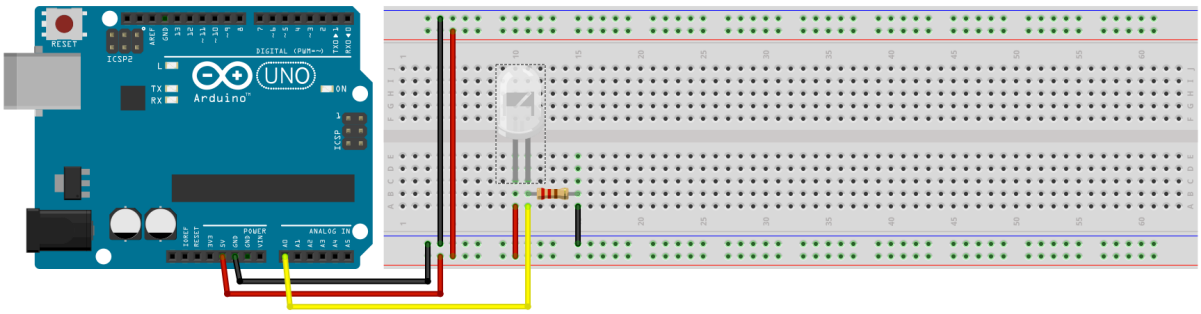
Pin nr	Name	Description	Arduino pin
1	Collector	shortest leg flat edge	5V
2	Emitter	longest leg rounded edge	Any Digital port and to GND through a 22K ohm resistor

55.4. Libraries needed for Flame Sensor (IR photo transistor)

None needed.

55.5. 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.

Sample Sketch

```
const int analogInPin = A0;

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

void loop()
{
  int sensorValue = analogRead(analogInPin);
  Serial.println(sensorValue);

  delay(200);
}
```

56. IR proximity sensor board

This IR Proximity sensor board can detect an object in its proximity.

56.1. Specifications IR proximity sensor board

- 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 through. 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.

56.2. 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

56.3. Libraries needed for IR proximity sensor board

None needed.

56.4. Sample IR proximity sensor board

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.

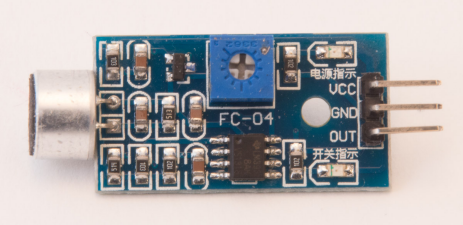
Sample Connections

- Connect G to GND.
- V+ to 5V.
- Connect to D13.

Sample sketch

```
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);
  }
}
```

57. Sound detection FC-04



This sensorboard detects the existence of sound (not the intensity).

57.1. 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.

57.2. 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

57.3. Libraries Sound detection

None needed.

57.4. Sample sketch Sound detection

The following sample sketch lights up the onboard LED on D13 for 1 second as sound as a sound was detected.

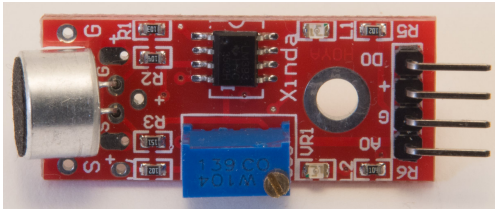
Sample connections

- Connect OUT to D2.
- Connect GND to GND.
- Connect VCC to 5V.

Sample sketch

```
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);
  }
}
```

58. Sound detection with digital and analog output



This sensorboard detects the existence and the intensity of sound.

58.1. 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.

58.2. 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

58.3. Libraries Sound detection

None needed.

58.4. Sample sketch Sound detection

The following sample sketch lights up the onboard LED on D13 for 1 second as sound as a sound was detected.

Sample connections

- Connect AO to A0.
- Connect G to GND.
- Connect + to 5V.
- Connect DO to D9.

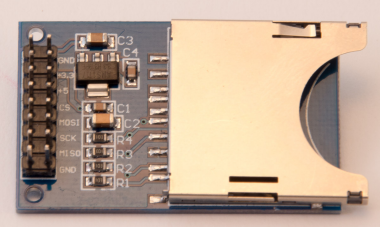
Sample sketch

```
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);
  }
}
```

Storage

This section describe the use of storage devices on your Arduino board, like an SD card, but also the use of RFID cards.

59. SD Card



59.1. Specifications SD Card

LC-TECH SD

59.2. Datasheet SD Card

59.3. Connections SC Card

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

59.4. Libraries needed for SD Card

- Secure Digital card library included with Arduino IDE.

Library use explanation

```
#include <SD.h>
```

Include the Secure Digital card libray 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 blocksize 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.

59.5. Sample SD Card

The following sketch shows the SD card type, the volume type and the volume size.

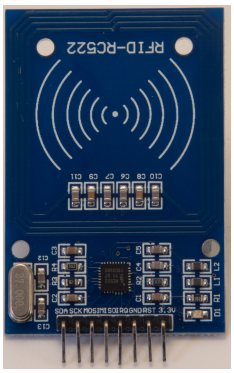
Sample Connections

- Connect GND to GND.
- Connect 3.3V to 3.3V.
- Connect CS to D4.
- Connect MOSI to D11.
- Connect SCK to D13.
- Connect MISO to D12.
- Leave 5V and second GND unconnected.

Sample Sketch

```
#include <SD.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)
{
}
```

60. Mifare RFID RC522



60.1. Specifications Mifare RFID RC522

- Contacless 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.

60.2. Datasheet Mifare RFID RC522

http://www.nxp.com/documents/data_sheet/MFRC522.pdf

60.3. 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	not connected
6	GND	Ground	GND
7	RST	Reset	Any Digital Port
8	3.3V	3.3 V	3.3V

60.4. Libraries needed for Mifare RFID RC522

- SPI library included in Arduino IDE: SPI.h.
- Mifare RC522 library from Miguel Balboa.
  https://github.com/miguelbalboa/rfid

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.

60.5. Sample Mifare RFID RC522

The following sketch

Connections

- Connect SDA to D10.
- Connect SCK to D13.
- Connect MOSI to D11.
- Connect MISO to D12.
- Connect GND to GND.
- Connect RST to D9.
- Connect 3.3V to 3.3V.

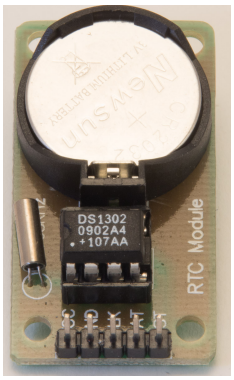
Sketch

```
#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);
}
```

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 looses power.

61. RTC module with DS1302 chip



This Real Time Clock module is equiped with a CR2032 battery so time will keep ticking after loosing power from the Arduino. It is also used in projects where timing must be accurate.

61.1. 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.

61.2. Datasheet DS1302 Trickle-Charge Timekeeping Chip

http://datasheets.maximintegrated.com/en/ds/DS1302.pdf

61.3. Connections 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

61.4. Libraries needed for module with DS1302 chip

- DS1302 RTC library from Henning Karlsen (http://www.henningkarlsen.com/electronics/index.php). http://www.henningkarlsen.com/electronics/library.php?id=5

LIBRARY use explanation

```
#include <DS1302.h>
```

```
DS1302 myrtc(CE, I/O, SLCK);
```
Create myrtc a new instance of the object type DS1302.

```
myrtc.halt(false);
```
Set the clock to run mode.

```
myrtc.writeProtect(false);
```
Disable write protection.

```
myrtc.setDOW(FRIDAY);
```
Set DOW (Day Of Week) to Friday (do this only once and everytime you need to adjust the time).

```
myrtc.setTime(20, 40, 0);
```
Set the time to 20:40 24 hour clock format (do this only once and everytime you need to adjust the time).

```
myrtc.setDate(28, 12, 2010);
```
Set the date to 28th of December 2013 (do this only once and everytime you need to adjust the time).

```
myrtc.getDOWStr()
```
This value contains the current day of the week (Monday .. Sonday).

```
Serial.print(myrtc.getDateStr());
```
This value contains the current date (dd.mm.yyyy).

```
myrtc.getTimeStr()
```
This value contains the current time (hh:mm:ss).

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.

61.5. Sample module with DS1302 chip

The following sketch

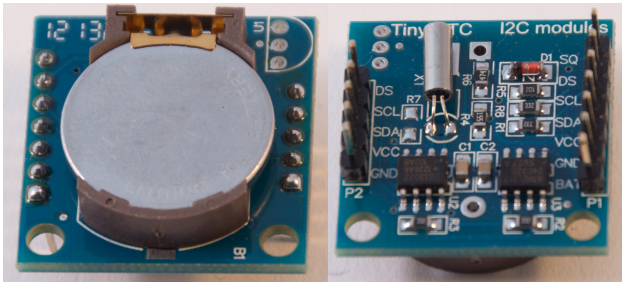
Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.
- Connect CLK to D4.
- Connect DAT to D3.
- Connect RST to D2.

Sample Sketch

```
#include <DS1302.h>
DS1302 rtc(2, 3, 4);

void setup()
{
rtc.halt(false);
  rtc.writeProtect(false);

  // Setup Serial connection
  Serial.begin(9600);

  //rtc.setDOW(SATURDAY);
  //rtc.setTime(20, 40, 0);
  //rtc.setDate(28, 12, 2013);
}

void loop()
{
  // Send Day-of-Week
  Serial.print(rtc.getDOWStr());
  Serial.print(" ");

  // Send date
  Serial.print(rtc.getDateStr());
  Serial.print(" -- ");

  // Send time
  Serial.println(rtc.getTimeStr());

  // Wait one second before repeating :)
  delay (1000);
}
```

62. Tiny RTC I²C module with DS1307 chip



This Real Time Clock module is equiped with a CR2032 battery so time will keep ticking after loosing power from the Arduino. It is also used in projects where timing must be accurate. It uses the I²C bus.

62.1. 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.

62.2. Datasheet Tiny RTC I²C module with DS1307 chip

- http://datasheets.maximintegrated.com/en/ds/DS1307.pdf

62.3. Connections Tiny RTC I²C module with DS1307 chip

P1: 7 pin header

Pin nr	Name	Description	Arduino pin
1	SQ	?	not needed
2	DS	?	not needed
3	SCL	I²C Clock	SDA (A4)
4	SDA	I²C Data	SCL (A5)
5	VCC	VCC	5V
6	GND	Ground	GND
7	BATT	Battery	not needed

P2: 5 pin header

Pin nr	Name	Description	Arduino pin
1	DS	?	not needed
2	SCL	I²C Clock	not needed
3	SDA	I²C Data	not needed
4	VCC	VDD	not needed
5	GND	Ground	not needed

62.4. Libraries needed for Tiny RTC I²C module with DS1307 chip

- Wire library included in Arduino IDE
- DS1307 Real Time Clock library from Adafruit
  https://github.com/adafruit/RTClib

Library use explanation

```
#include <Wire.h>
```
Include the Wire library included in Arduino IDE.

```
#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.

62.5. 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 A4
- Connect SDA to A5.
- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
#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 ain'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.

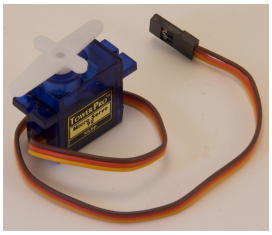
63. Standard Servo

Servo's are DC motors with a potentiometer connected to the motor shaft. This potentiometer tells the servo driver in which postion 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.

63.1. Specifications Standard Servo

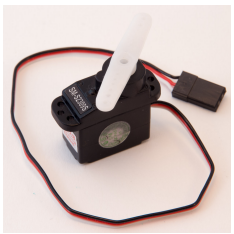
63.2. Connections Standard Servo

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

63.3. Libraries needed for Standard Servo

- Servo library included in Arduino IDE: Servo.h.

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 it's 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.

63.4. 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.

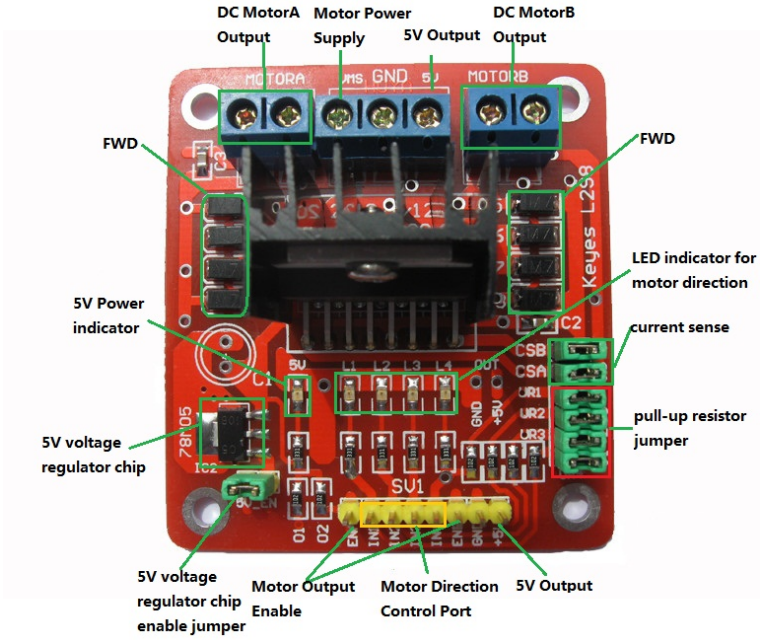
Connections

- Connect 1 to GND.
- Connect 2 to 5V.
- Connect 3 to D6.

Sketch

```
#include <Servo.h>
Servo myservo;

int pos = 0;

void setup()
{
  myservo.attach(11);
}


void loop()
{
  myservo.writeMicroseconds(600);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
  myservo.writeMicroseconds(2150);
  delay(2000);
  myservo.writeMicroseconds(1375);
  delay(2000);
}
```

64. 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

64.1. Connections Motor Driver Board

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

64.2. Library Motor Driver Board

None needed.

64.3. Datasheet L298N Motor Driver

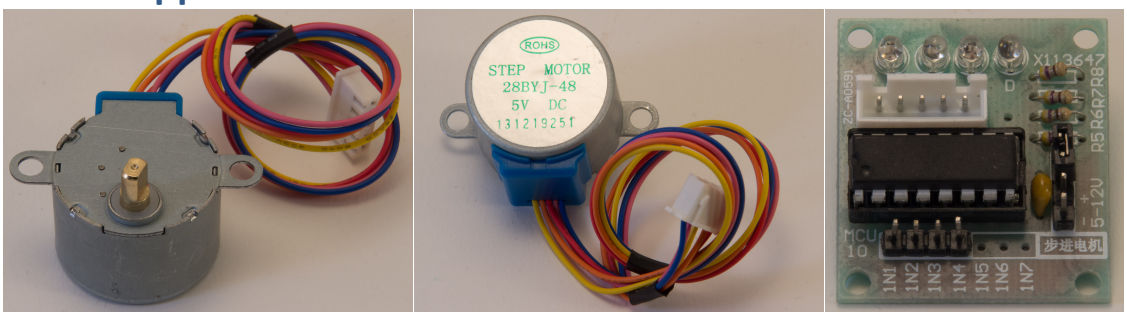
https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

64.4. Sample sketch Motor Driver Board

With the following sketch Motor A wil start turning clockwise (CW) at top speed for 2 seconds, stop for 2 seconds en repeat this.

```
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);
}
```

65. 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.

65.1. 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.

65.2. Datasheets Stepper Motor 28BYJ-48 5V with ULN2003 Interface board

Datasheet Stepper Motor 28BYJ-49 5V

http://robocraft.ru/files/datasheet/28BYJ-48.pdf

65.3. 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.

65.4. Libraries needed for Stepper Motor 28BYJ-48 5V

- Stepper motor driver included in Arduino IDE: Stepper.h.

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.

65.5. 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.

Sketch

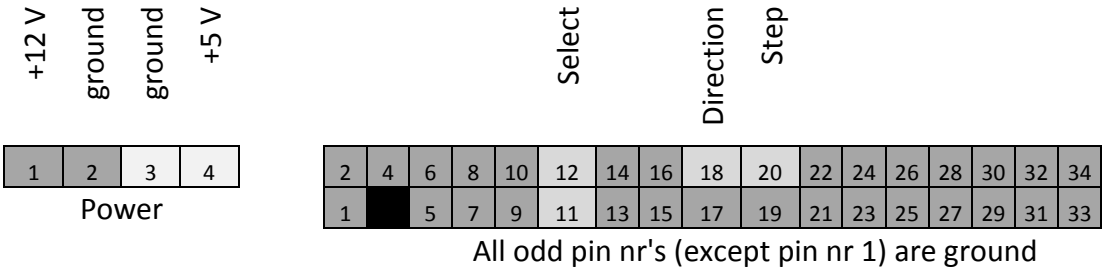
```
#include <Stepper.h>

const int stepsPerRevolution = 2040;

Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup()
{
  myStepper.setSpeed(14);
}

void loop()
{
  myStepper.step(stepsPerRevolution);
  delay(500);

  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

66. Floppy disc drive

Floppy disc drives (both 3.5 and 5.25 inch) contains a steppermotor 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 "72 External Power Supply").

66.1. 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:



66.2. Libraries needed for Floppy disc drive

None needed

66.3. Sample Floppy Disc Drive

The following sample sketch drives the read/write head repeatingly 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.

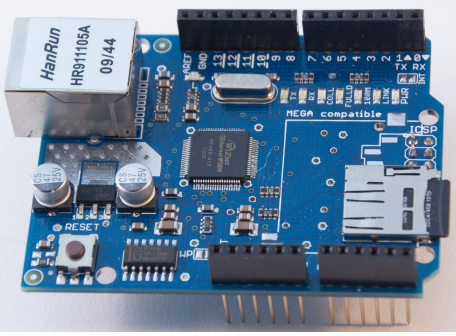
Sample Sketch

```
#define DELAY 1200  // sets speed
#define STEPS 200   // set nr of steps

// Floppy motor
int dir_pin = 12;
int step_pin = 13;

void setup()
{
  // Initial setup of pins
  pinMode(dir_pin, OUTPUT);
  pinMode(step_pin, OUTPUT);
}

void loop() {
  delay(1);
  digitalWrite(dir_pin, HIGH);  // clockwise
  delay(1);
  perform_step(STEPS);            // perform 200 steps
  delay(500);
  digitalWrite(dir_pin, LOW);   // anti-clockwise
  delay(1);
  perform_step(STEPS);            // perform 100 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);
  }
  // Set the pin low before we end
  digitalWrite(step_pin, LOW);
}
```

Shields

Shields are PCB's that can be placed directly on top of an Arduino UNO, so they are very easy to connect.

67. Ethershield



67.1. Specifications Ethershield

- Wiznet W5100 chip
- On-board micro-SD card slot

67.2. 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

67.3. Connections Xx

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

67.4. Libraries needed for Ethershield

- The Serial Peripheral Interface library included in the Arduino IDE.
- The Ethernet library included in the Arduino IDE.
- Secure Digital card library included in the Arduino IDE (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);
```

Webser 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.

67.5. Sample Ethershield

The following sketch starts a webserver that shows the value of the 6 analog ports in HTML.

Sample Connections

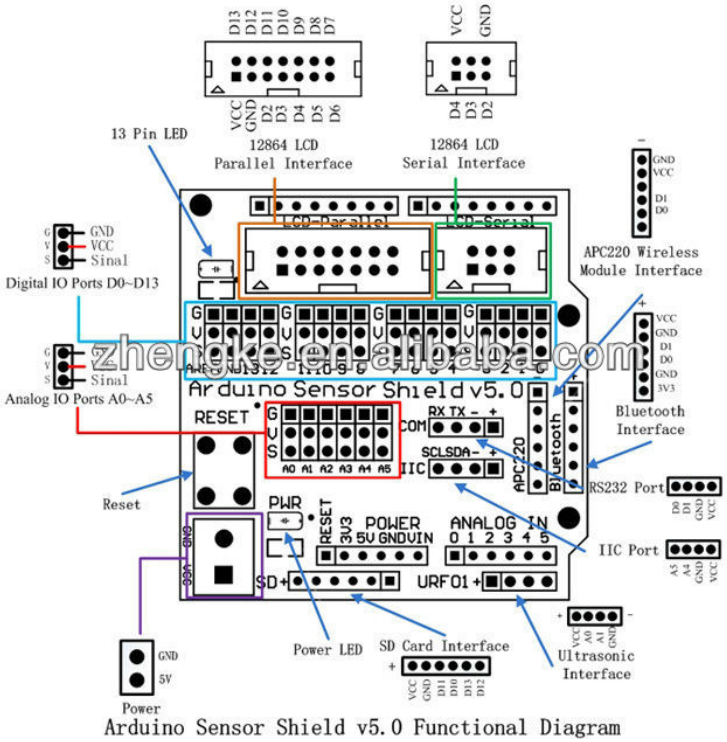
- Shield on top of Arduino UNO.

Sample Sketch

```
#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");
  }
}
```

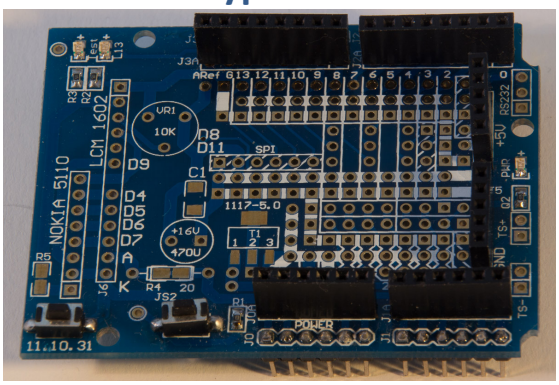
68. Arduino Sensor Shield v5.0

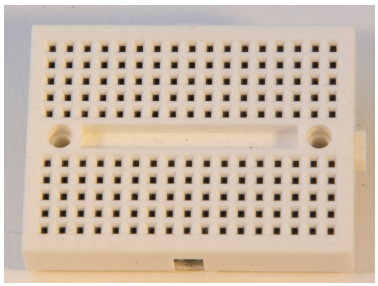


Arduino Sensor Shield v5.0 Functional Diagram

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.

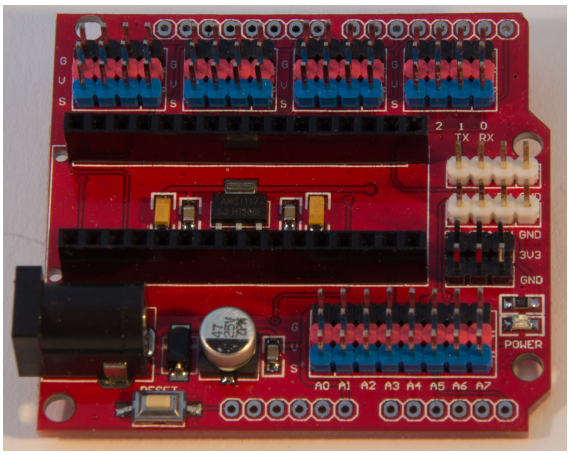
69. Proto type shield





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.

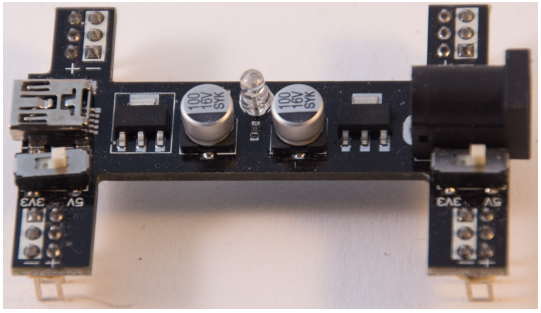
70. Nano sensor shield



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.

Power supplies

71. Black Wings



Black wings when placed on the bus strips 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).

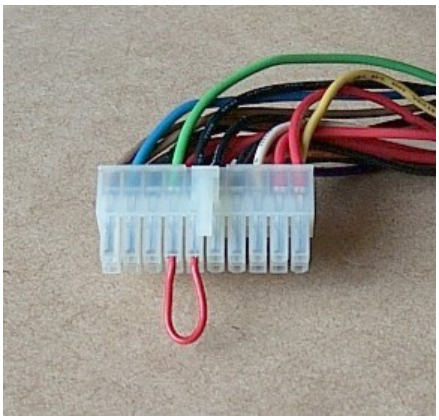
Both sides of the bus strips can be regulated to either 5V or 3.3V independently .

Each side of the Black Wings consist of 3 connectors for + (Vcc) and 3 connectors for – (GND) to give physical stability.

72. External Power Supply

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 computercase, you'll need a simple hack to switch on your PSU.

Locate the 20/24 pin ATX motherboard conector and connect the green cable (only one) with one of the black cables (ground). Like in the following picture. Be carefull, remove the Power cord (230 V cable) before you do this and only then connect the Power cord to mains again. If the PSU fan won't start, immediately remove the Power cord again and check you wiring!

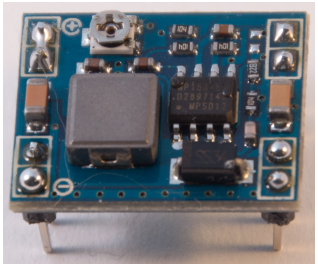


72.1. Connections

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)

73. 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 2 3,7 V 18650 LiOn batteries (2x3,7). Even when the voltage of your penlites drops to just above 6 V (2x3 V), the 5.0 V output voltage remains steady.

73.1. Specifications DC Step-Down Adjustable Power module

- Input Voltage: 4.5-28V
- Output Voltage: 0.8-20V
- Output Current: 3A (Max.)

73.2. Datasheet DC Step-Down Adjustable Power module

- http://www.electrodragon.com/w/images/d/d3/MP1584.pdf

73.3. 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

Miscellaneous

In this section you will find miscellaneous components not falling in the other categories.

74. USB to RS232/TTL cable

With this cable you can add a serial port connection to an Arduino board that lacks an UART, see chapters 4 “Boarduino”, 5 “AVR Development board as Arduino”, 6 “Arduino on a breadboard” and chapter 7 “Attiny45/Attiny85”.

74.1. Specifications USB to RS232/TTL cable

- FTDI
- PL203HX chip

74.2. Datasheet USB to RS232/TTL cable

- http://v-comp.kiev.ua/download/pl2303HX.pdf

74.3. Connections USB to RS232/TTL cable

Pin nr	Name	Description	Arduino pin
1	Red	VCC (5V)	5V
2	Black	Ground	GND
3	Green	TxD	D0 (Rx)
4	White	RxD	D1 (Tx)

74.4. Libraries needed for USB to RS232/TTL cable

None needed.

74.5. Sample USB to RS232/TTL cable

The following sketch prints the text “Hello World“ to the new serial port.

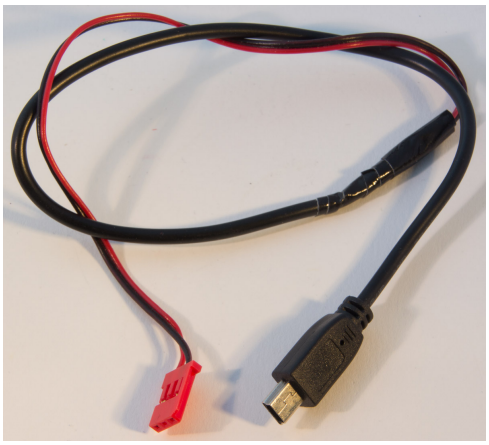
Sample Connections

- Connect Red to 5V.
- Connect Black to GND.
- Connect Green to D0 (Rx).
- Connect White to D1 (Tx).

Sample Sketch

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

void loop()
{
  Serial.println("Hello World");
}
```

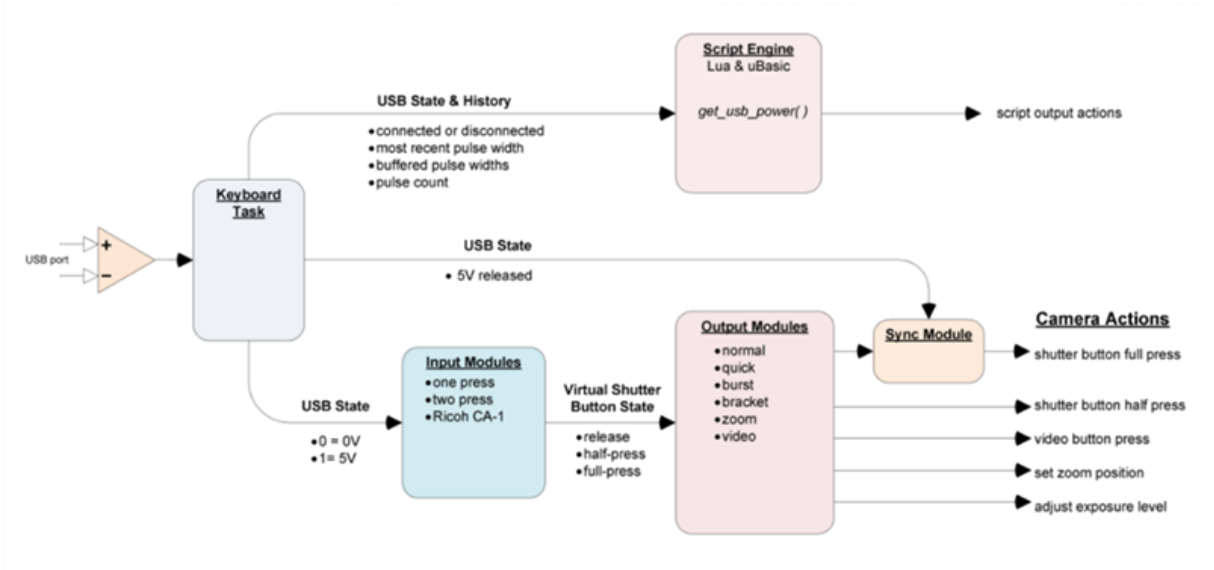
75. Selfmade Canon CHDK/SDM trigger cable



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.

75.1. Specifications Selfmade 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**

75.2. Connections Canon CHDK/SDM trigger cable

Pin nr	Name	Description	Arduino pin
1	S (red)	Signal	Any digital port
2	nc	not connected	nc
3	GND (black)	Ground	GND

75.3. Libraries needed for Selfmade Canon CHDK/SDM trigger cable

None needed.

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.
    - Downloadlink: 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

75.4. Sample Single trigger

The following script and sketch 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.

Sample Arduino sketch Single trigger

```
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);
}
```

Sample CHDK script Single trigger

```
@title Remote button

while 1
    wait_click 1
    if is_key "remote" then shoot
wend

end
```

75.5. Sample Multiple trigger events

By sending pulses ranging in length from 10 to 180 ms a gap in between pulses of at least 55ms, you can trigger different events. The pulses can be measured in a CHDK 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

This sketch sends 6 different pulses to the Canon camera with 10 seconds in between.

```
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);
}
```

Sample CHDK script Multiple trigger events

This script can distinguish 6 different pulse received from the Arduino board. Depending on the pulse length the camera will zoom to a specified level and would then take a picture.

```
@title Arduino_trigger
print "Start Arduino script"
z=0
while 1
   do
      a = get_usb_power
   until a>0
   if a >= 1 and  a <= 3 then gosub "Level2"
   if a >= 4 and  a <= 6 then gosub "Level5"
   if a >= 7 and  a <= 9 then gosub "Level8"
   if a >= 10 and a <= 12 then gosub "Level11"
   if a >= 13 and a <= 15 then gosub "Level14"
   if a >= 16 and a <= 18 then gosub "Level17"
   if a > 19 then print "error"
   wend
end

:Level2
     print "20 ms pulse"
     set_zoom 0
     shoot
     return

:Level5
     print "50 ms pulse"
     set_zoom 20
     shoot
     return

:Level8
     print "80 ms pulse"
     set_zoom 50
     shoot
     return

:Level11
     print "110 ms pulse"
     set_zoom 60
     shoot
     return

:Level14
     print "140 ms pulse"
     set_zoom 80
     shoot
     return

:Level17
     print "170 ms pulse"
     set_zoom 100
     shoot
     return
```

76. 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.

76.1. Specifications Resistor

The value of most resistors are color coded.

Color	1st band	2nd band	3rd band[1]	Last band Multiplier	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%

For example a resistor with the following colors:

Red	Red	Brown	Gold

This is a resistor with a 4 band code, so the column 3rd band is not used, but the column named last band is used instead.

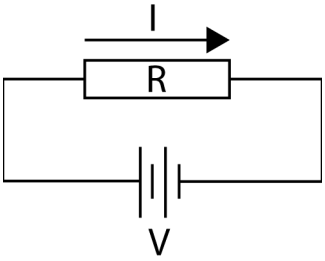
Red	2		
Red		2	
Brown			x10
Gold			± 5%

2 2 x10 = 22 x 10 = 220 ohm ± 5%

[1] Only used in 5 band code. When using a 4 band code, this column is not used, but the column named Last band is used instead!

76.2. 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:

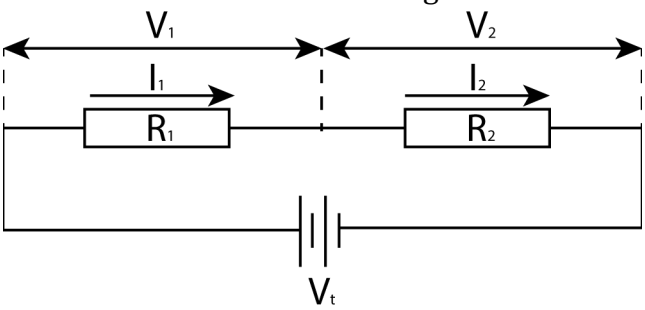
$$R = \frac{V}{I}$$

and

$$V = I \times R$$

76.3. Two resistors in series[1]

Usefull formula's when working with two resistors in series:



Replacement resistor for R1 and R2:

$$Rt = R1 + R2$$

$$It = I1 = I2 = \frac{Vt}{Rt}$$

$$Vt = V1 + V2$$

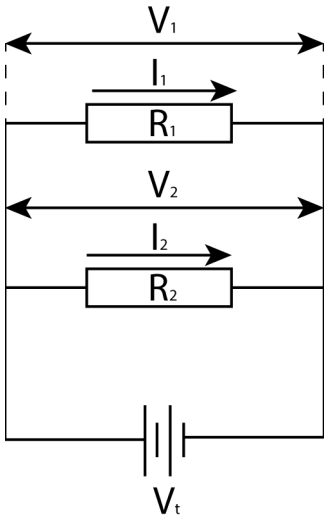
$$V1 = \frac{R1}{Rt} \times Vt = \frac{R1}{R1+R2} \times Vt$$

$$V2 = \frac{R2}{Rt} \times Vt = \frac{R2}{R1+R2} \times Vt$$

[1] You can also apply this to multiple resistors in series.

76.4. Two resistors parallel to each other:

Usefull 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 $\frac{1}{Rt} = \frac{1}{R1} + \frac{1}{R2} + \frac{1}{R3} \ldots\ldots +$

76.5. 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:

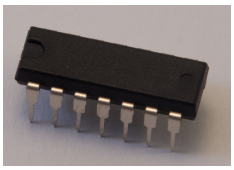
http://led.linear1.org/1led.wiz

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 forware voltage of 1.7 Volt and a diode forward current of 18mA!

77. 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 "Motor Driver board L298n". With that board IN1 and IN2 should always be each others 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 others inversion). This way you can save 2 digital ports when driving a robot (and some code lines as well).

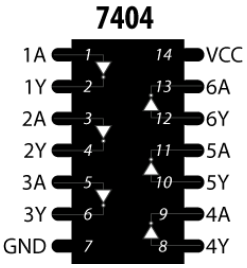
77.1. Specifications Inverter 7404

- 6 inputs and corresponding inverted output.

77.2. Datasheet Inverter 7404

- http://www.ti.com/lit/ds/symlink/sn74ls04.pdf

77.3. Connections Inverter 7404

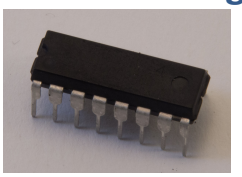


Pin nr	Name	Description	Arduino pin
1	1A	Input 1	Any Digital Output
2	1Y	Output 1 (Input 1 inverted)	To external device
3	2A	Input 2	Any Digital Output
4	2Y	Output 2 (Input 2 inverted)	To external device
5	3A	Input 3	Any Digital Output
6	3Y	Output 3 (Input 3 inverted)	To external device
7	GND	Ground	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

77.4. Libraries needed for Inverter 7404

None needed

78. 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.

78.1. 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

78.2. Datasheet Shift register 74HC595

- http://www.nxp.com/documents/data_sheet/74HC_HCT595.pdf

78.3. 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	
10	$MR^{inverted}$	Master Reset (active LOW)	
11	SHCP	Shift Register Clock input	
12	STCP	Storage Register Clock input	
13	$OE^{inverted}$	Output enable input (active LOW)	
14	DS	Serial data input	
15	Q0	Parallel data output 0	
16	VCC	Supply Voltage	5V

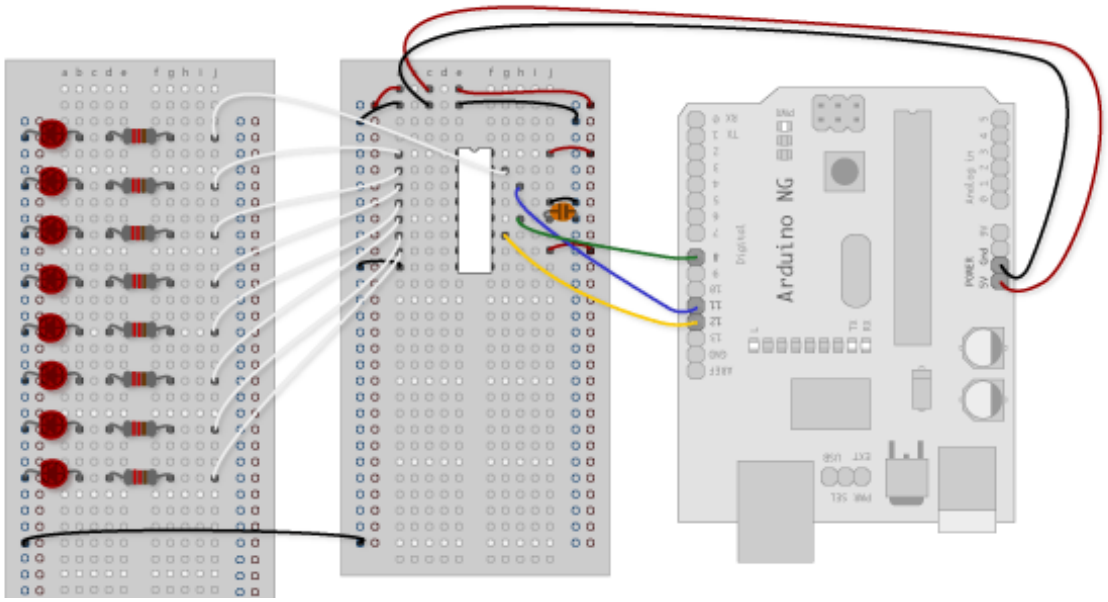
78.4. Libraries needed for Shift register 74HC595

None needed.

78.5. Sample Shift register 74HC595

The following sketch displays the binary values 0 .. 255 on 8 leds 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 us 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

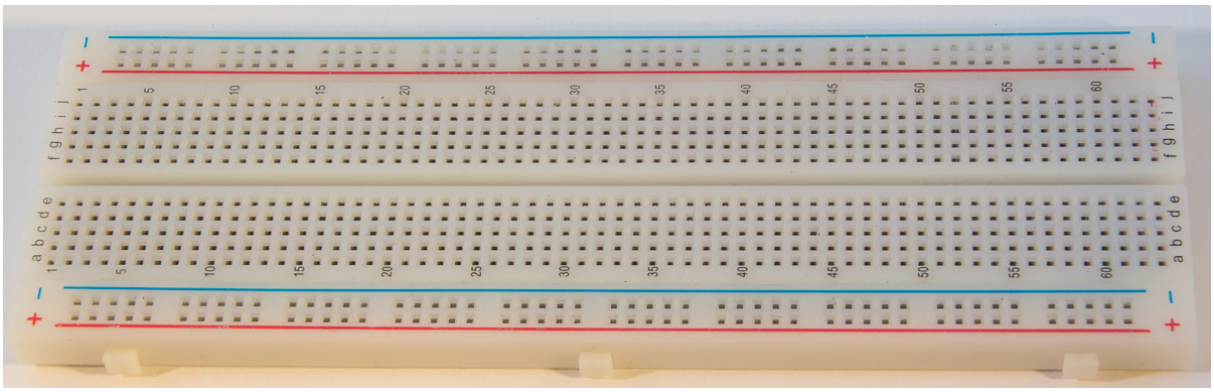


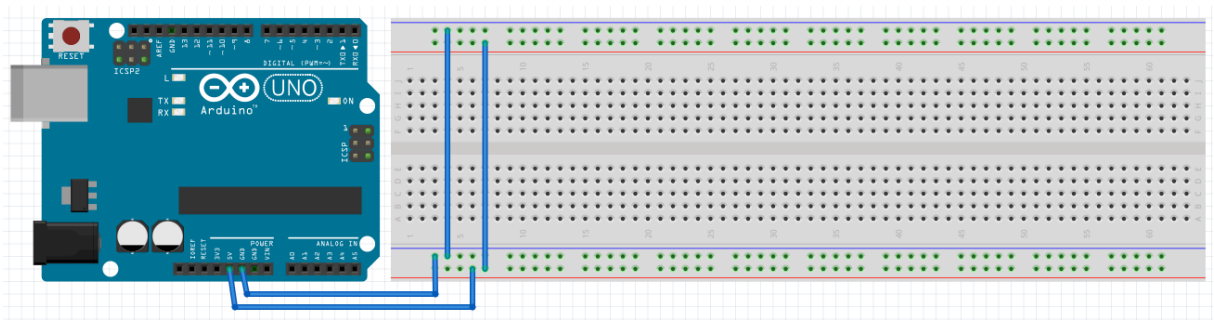
Pict. 2 Source: **http://arduino.cc/en/tutorial/ShiftOut#.UwO2cUJ5OhE**

Sample Sketch

```
//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);
  }
}
```

79. 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.

Projects

This section contains some projects that combine several sensors described in previous chapters.

80. 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.

80.1. 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 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: 57 Sound detection FC-04)
- Optocoupler (see: 49 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: 47 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.

80.2. 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

80.3. 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 pause then for 1 second for the next Flash, to prevent bouncing.

```
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 webshops.

81. Webshops

Company	URL	Description
Sparkfun electronics	https://www.sparkfun.com/	• Boards • Shields • Sensors • Kits • Programmers • Components
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
MicroController Pros LLC	http://microcontrollershop.com/	• Boards • Shields • Sensors • Kits • Programmers • Electronic components

82. 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)

To Do

In this section you will find components that need more time to finish, including fragments of information.

83. Arduino UNO as ISP

83.1. Specifications Arduino as ISP

83.2. Datasheet Arduino as ISP

83.3. Connections Arduino as ISP

Pin nr	Name	Description	Arduino pin
1			
2			
3			
4			

83.4. Sample Arduino as ISP

The following sketch

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
void setup()
{
}

void loop()
{
}
```

- Open the ArduinoISP sketch from FILE/EXAMPLES.
- Select your Arduino board and the corresponding serial port.
- Upload the ArduinoISP sketch to your Arduino.
- Place your Attiny on a breadboard.
- Connect a 10uF capacitor between Reset and GND pins on your Arduino. (negative side of the capacitor to GND).
- Connect D10 (SS) to Attiny pin 1.
- Connect D11 (MOSI) to Attiny pin 5.
- Connect D12 (MISO) to Attiny pin 6.
- Connect D13 (SCK) to Attiny pin 7.
- Connect GND to Attiny pin 4.
- Connect 5V to Attiny pin 8.

84. Template Xx

84.1. Specifications Xx

84.2. Datasheet Xx

84.3. Connections Xx

Pin nr	Name	Description	Arduino pin
1			
2			
3			
4			

84.4. Libraries needed for Xx

- 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.

84.5. Sample Xx

The following sketch

Sample Connections

- Connect VCC to 5V.
- Connect GND to GND.

Sample Sketch

```
void setup()
{
}

void loop()
{
}
```