Getting your hands dirty with Embedded: An introduction to Arduino

Table of Contents

References:

- http://forefront.io/a/beginners-guide-to-arduino

1. Birth of MCU/ Microprocessor

2. Introduction to Arduino

a. What is 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.

Source: http://www.arduino.cc/

b. What is microcontroller

A microcontroller is a compact microcomputer designed to govern the operation of embedded systems in motor vehicles, robots, office machines, complex medical devices, mobile radio transceivers, vending machines, home appliances, and various other devices. A typical microcontroller includes a processor, memory, and peripherals.

c. Which Arduino board we are going to use

The Arduino Duemilanove is a microcontroller board based on Atmel ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 Analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
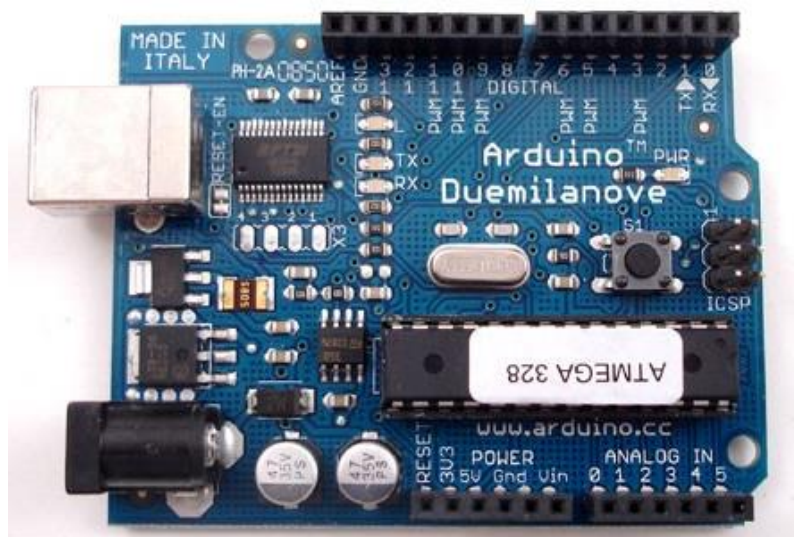
"Duemilanove" means 2009 in Italian and is named after the year of its release.

Summary of features provided by Duemilanove:

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
Communication Interface	UART (Serial RX/TX), SPI, I2C.
Programming Interface	USB (using bootloader), ICSP(SPI)
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 2 KB used by boot loader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power
The Arduino Duemilanove can be powered via the USB connection or with an external power supply.
The power   source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter or battery.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Memory
The ATmega328 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader).

Input and Output
Each of the 14 digital pins on the Duemilanove can be used as an input or output,
using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 K-Ohms.

In addition, some pins have specialized functions as mentioned below
1.   Serial: 0 (RX) and 1 (TX)
Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
2.   External Interrupts: 2 and 3
These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
3.   PWM: 3, 5, 6, 9, 10, and 11
Provide 8-bit PWM output with the analogWrite() function. These pins can also be used as GPIOs.
4.   SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)
These pins support SPI communication using the SPI library.
5.   LED: 13
There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Duemilanove has 6 Analog inputs, each of which provides 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function.

Additionally, some pins have specialized functionality:
1. I^2C: Analog input pins A4 (SDA) and A5 (SCL) Support I^2C (TWI) communication using the Wire library.
There are a couple of other pins on the board:
1.   AREF
This is reference voltage for the Analog inputs. Used with analogReference().
2.   Reset
Bring this line LOW to reset the microcontroller.

SETUP and LOOP brief Introduction

Source: http://arduino.cc/en/Reference/HomePage

Arduino Commands

Setup

Command	Explanation and usage
pinMode(pin , mode)	Used to direct given Arduino pin as Input or Output Ex : `pinMode(13, OUTPUT);` //would set Arduino pin 13 as output Ex : `pinMode(13, INPUT);` //would set Arduino pin 13 as input
Serial-UART	**Universal Asynchronous Receiver Transmitter**
Serial.begin(baudrate)	Used to initialize serial communication. Input is baud rate at which communication takes place. By default baud rate is configures to 9600.

Loop

Command	Explanation and usage
GPIO's	**General Purpose Input Output**
digitalWrite(pin, value)	Write a HIGH (+5V) or a LOW (+0V) value to a digital pin Ex: digitalWrite(13, HIGH); // would set the pin 13 as HIGH(5V)
digitalRead(pin)	Used to read the status of pin which is defined in input mode using pinMode command.
Delay	**Delay Functions**
delay(ms)	ms: the number of milliseconds to pause (*unsigned long*) Ex: delay(1000); // waits for 1000 millisecond
ADC	**Analog to Digital Convertor**
analogRead(pin)	Reads the value from the specified Analog pin and returns integer (0-1023) Ex: value = analogRead(3); // read the pin 3 and save the value in value
analogReference(type)	Configures the reference voltage used for Analog input (i.e. the value used as the top of the input range). The options are: - DEFAULT: the default Analog reference of 5 volts (on 5V Arduino boards) - EXTERNAL: the voltage applied to the AREF pin (0 to 5V only) is used as the reference.
PWM	**Pulse Width Modulation**
analogWrite(pin, value)	The frequency of the PWM signal on most pins is approximately 490 Hz Pin: the pin to write to. Value: the duty cycle: between 0 (always off) and 255 (always on). Ex: analogWrite(9, 255); // analogWrite values from 0 to 255
Serial –UART	
Serial.write(byte)	Byte: Character byte supposed to be sent through serial communication.

Arduino Duemilanove Pin out Diagram

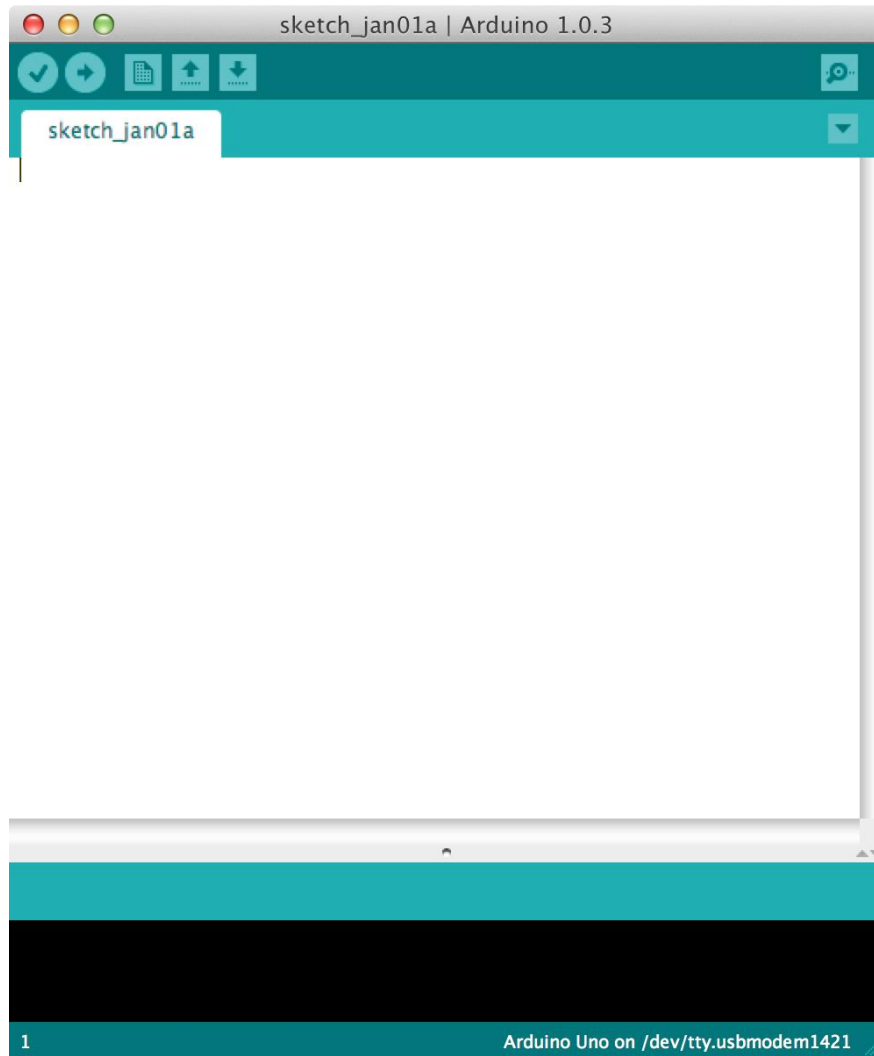Source: http://arduino.cc/en/Main/arduinoBoardDuemilanove

3.   Arduino sketches and internals

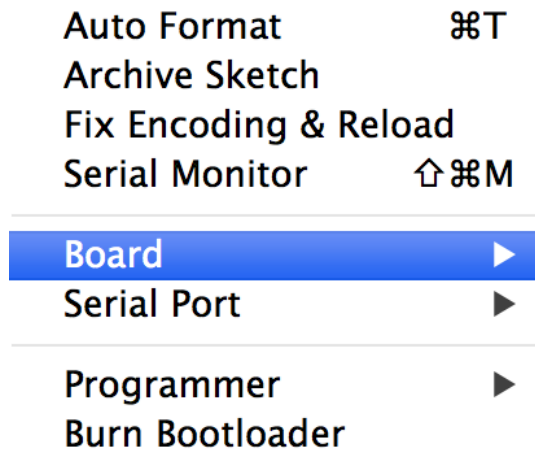Arduino IDE Version 0022 works smoothly with Arduino Duemilanove ([Link]).

Steps for using the Arduino IDE are mentioned below:
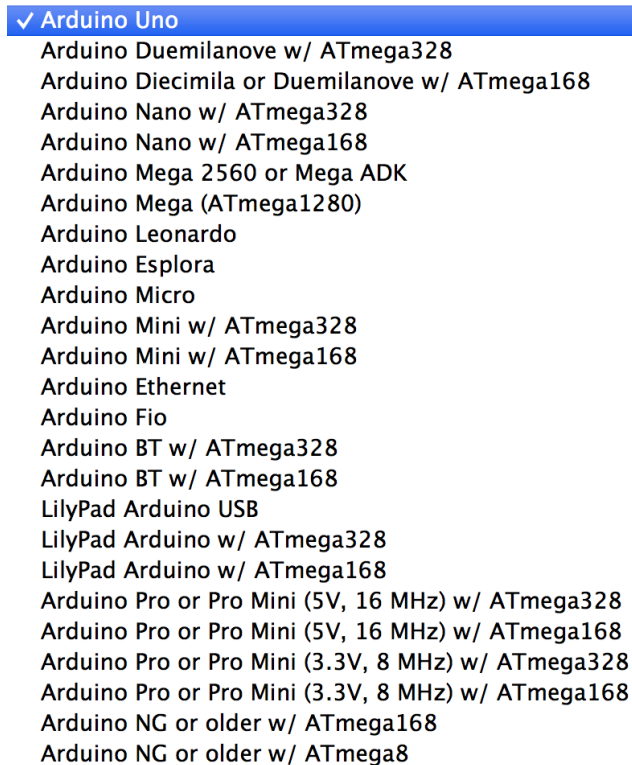
1. Start with new Arduino sketch

2. Initial setup:

Select the board Go to **Tools** menu and select **Board**

Auto Format   ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor   ⇧⌘M

Board   ▶
Serial Port   ▶

Programmer   ▶
Burn Bootloader

Then select the type of Arduino you want to program, in our case it's the **Arduino** Duemilanove.

✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8

4. The Code

The codes you write for your Arduino are known as **sketches**. Every sketch needs two *void type functions*, `setup()` and `loop()`. A void type function doesn't return any value. The `setup()` method is ran once at the just after the Arduino is powered up and the `loop()` method is ran continuously afterwards. The `setup()` is where you want to do any initialisation steps, and in `loop()` you want to run the code you want to run over and over again.

So, your basic sketch or program should look like this:

```
void setup()
{

}

void loop()
{

}
```

5.   Blinking LED example

The onboard LED we want to control is on pin 13. In our code above the `setup()` method let's create a variable called `ledPin`. In Arduino we need to state which type our variable is beforehand, in this case it's an integer, so it's of type `int`.

```
int ledPin = 13;

void setup()
{

}

void loop()
{

}
```

Each line is ended with a semicolon (`;`).

In the setup() method we want to set the ledPin to the output mode. We do this by calling a special function called pinMode() which takes two variables, the first the pin number, and second, whether it's an input or output pin. Since we're dealing with an output we need to set it to a constant called OUTPUT . If you were working with a sensor or input it would be INPUT .

```
int ledPin = 13;

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

void loop()
{

}
```

In our loop we are going to first switch off the LED to make sure our program is being transferred to the chip and overriding the default.

We do this by calling another special method called digitalWrite() . This also takes two values, the pin number and the level, HIGH or the on state or LOW the off state.
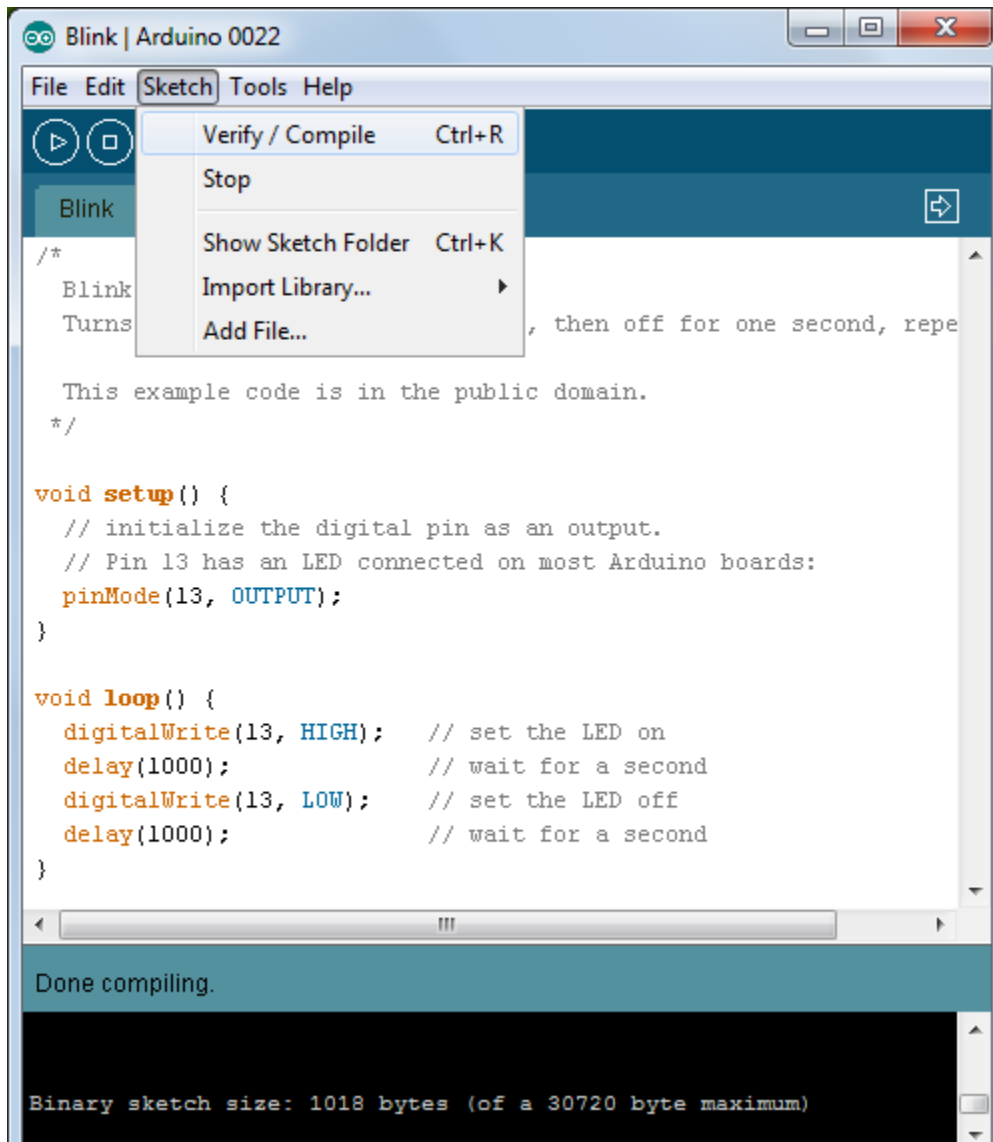
```
int ledPin = 13;

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

void loop()
{
        digitalWrite(ledPin, LOW);
}
```

Next we want to compile to machine code and deploy or *upload* it to the Arduino.
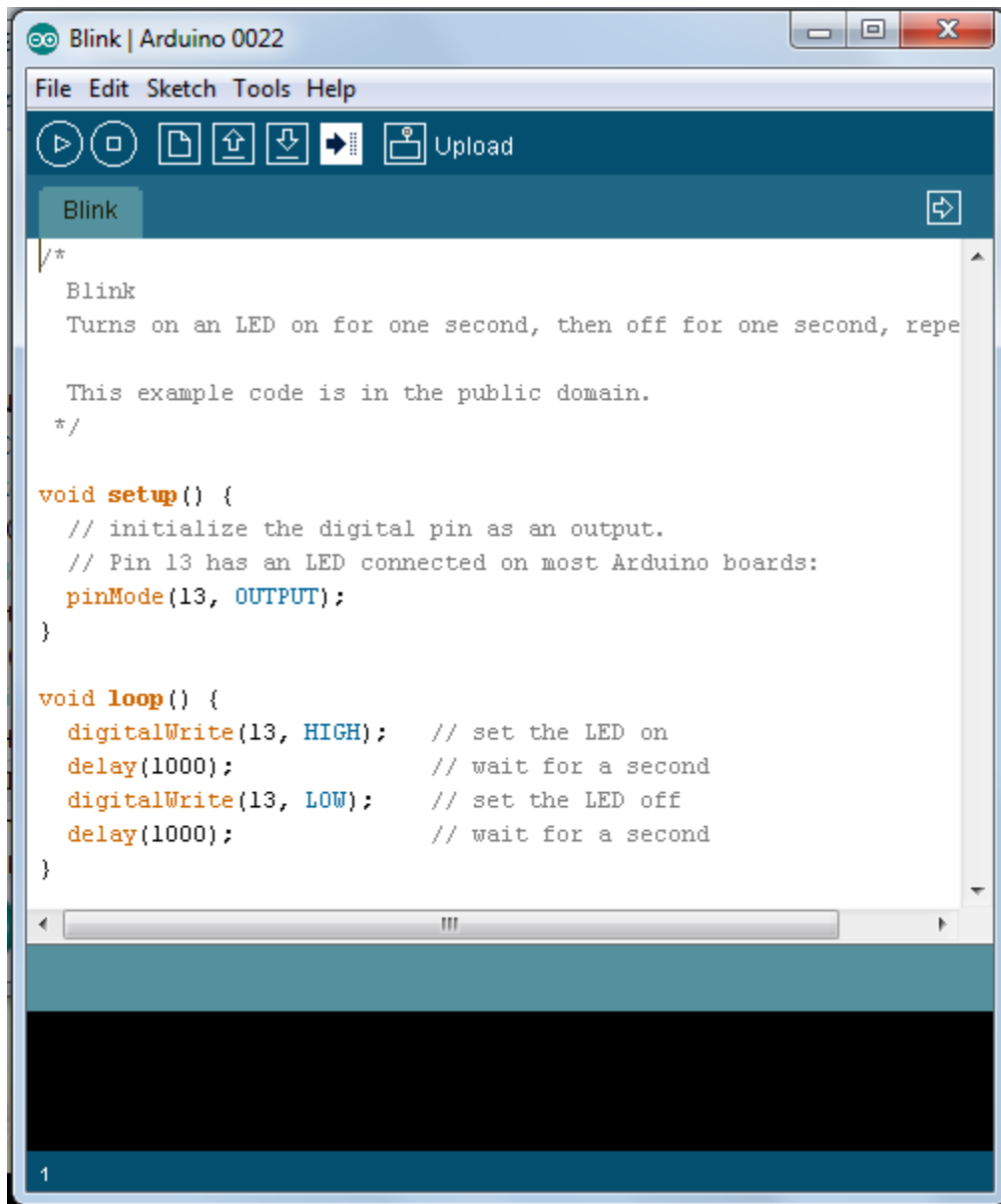
6.   Compiling the Code

Go to **Sketch** select **Verify/Compile** the sketch will start compiling

Selecting COM port for Arduino board:

Press upload button to load firmware (hex file) in to flash memory.


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repe

  This example code is in the public domain.
 */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);    // set the LED on
  delay(1000);               // wait for a second
  digitalWrite(13, LOW);     // set the LED off
  delay(1000);               // wait for a second
}
```

7.   Interfacing 16x2 Character LCD (Driver: HD44780 Hitachi)

Steps:

1. Connect the LCD as shown in snapshot below.
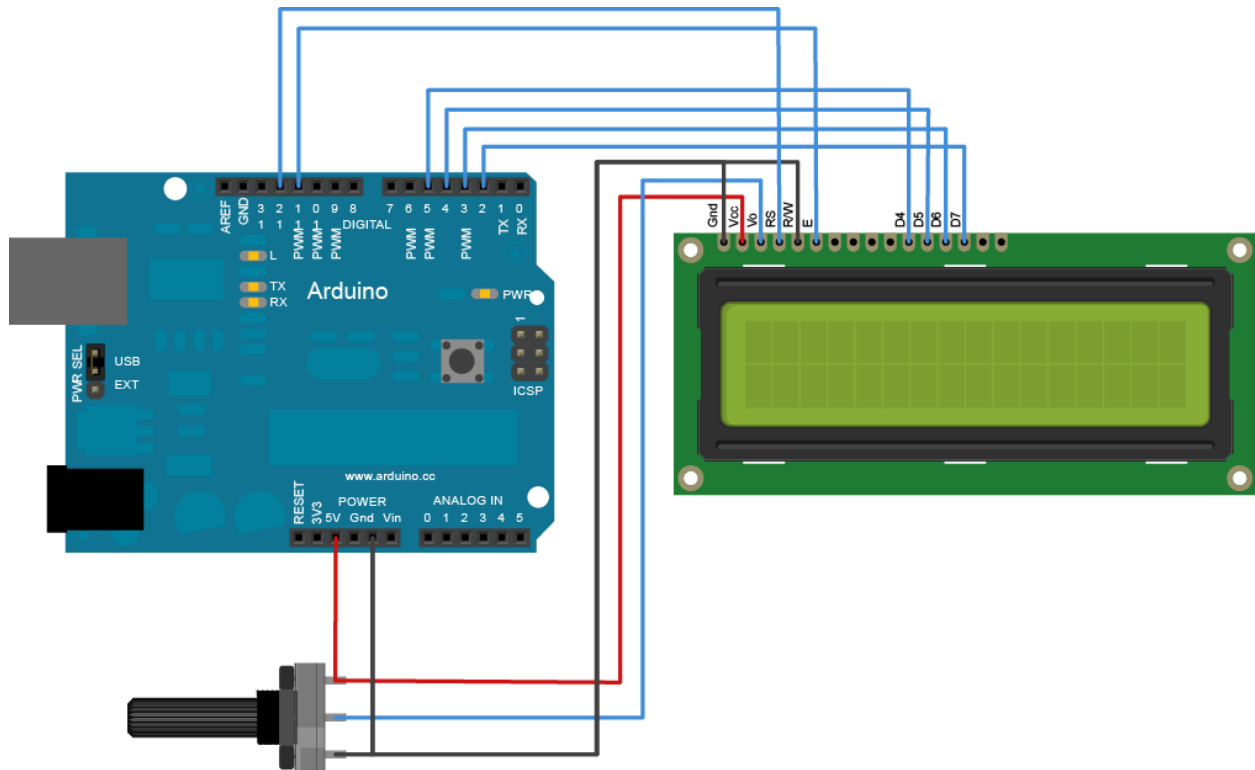
In a more detailed manner below:

tronixstuff.com / projects > Another LCD shield
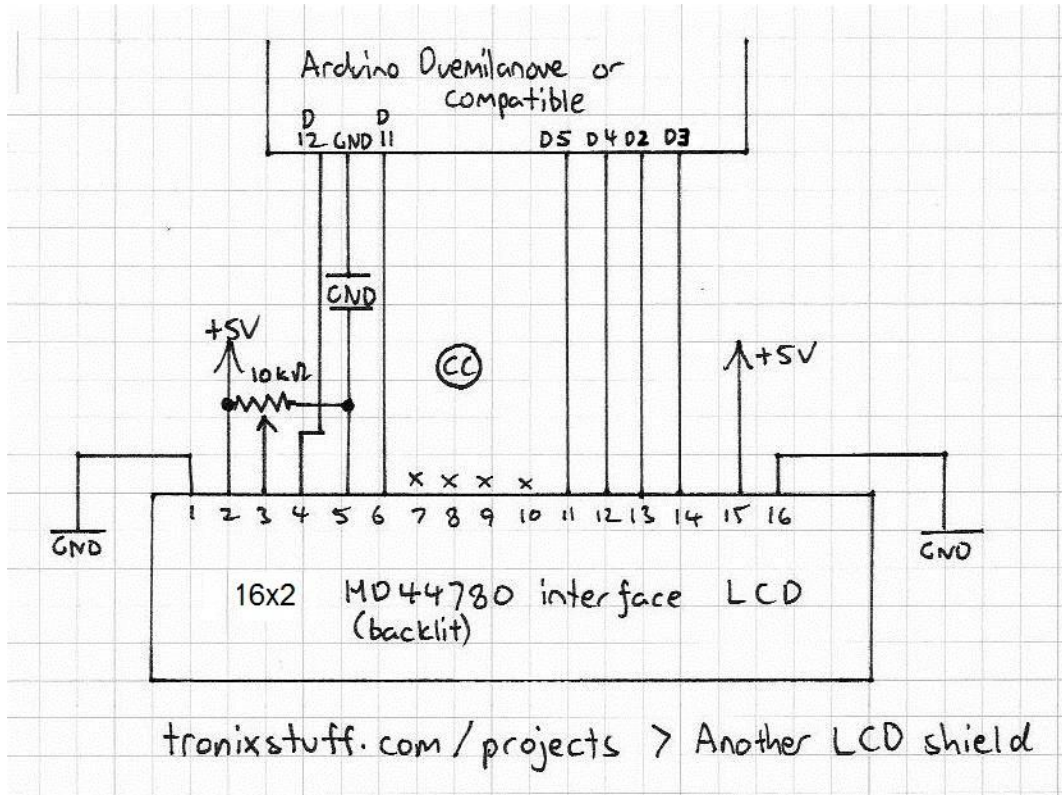
2. Example Code to display "Hello world"

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup()
    {
        // set up the LCD's number of columns and rows:
        lcd.begin(16, 2);
        // Print a message to the LCD.
        lcd.print("hello, world!");
    }
void loop()
    {
        // set the cursor to column 0, line 1
        // (note: line 1 is the second row, since counting begins
    with 0):
        lcd.setCursor(0, 1);
        // print the number of seconds since reset:
        lcd.print(millis()/1000);
    }
```

3. Functions for Character LCD

lcd.begin(cols, rows) :

Description: Initializes the interface to the LCD screen, and specifies the dimensions (width and height) of the display. begin() needs to be called before any other LCD library commands.

Syntax: *lcd*.begin(cols, rows)

Parameters

cols: the number of columns that the display has

rows: the number of rows that the display has

i.e. 16x2 or 20x4.

lcd.print(data) :

Description

Prints text to the LCD.

Syntax

lcd.print(data)

lcd.print(data, BASE)

Parameters

data: the data to print (char, byte, int, long, or string)

BASE (optional): the base in which to print numbers: BIN for binary (base 2), DEC for decimal (base 10), OCT for octal (base 8), HEX for hexadecimal (base 16).

lcd.setCursor(col, row) :

Description

Position the LCD cursor; that is, set the location at which subsequent text written to the LCD will be displayed.

Syntax

lcd.setCursor(col, row)

Parameters

col: the column at which to position the cursor (with 0 being the first column)

row: the row at which to position the cursor (with 0 being the first row)

Reference: http://arduino.cc/en/Reference/LiquidCrystal

8. Writing User Defined functions

There are two required functions in an Arduino sketch, setup() and loop(). Other functions must be created outside the brackets of those two functions. As an example, we will create a simple function to multiply two numbers.
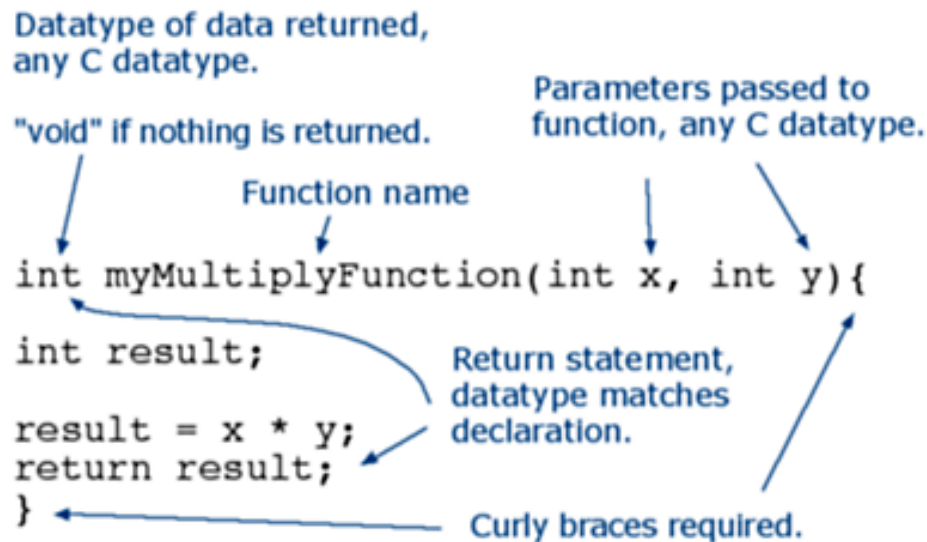
Reference for data types: look for Data type under variable http://arduino.cc/en/Reference/HomePage

User defined Function's need to declared outside any other function, and can go either above or below the "loop()" function.

To "call" user defined function, we pass it parameters of the data type that it is expecting:

Example of the above used defined function "myMultiplyFunction":

Source: http://arduino.cc/en/Reference/FunctionDeclaration

```
void setup()
{
  Serial.begin(9600); // Default baud rate
}

void loop()
{
  int i = 2;
  int j = 3;
  int k;
  k = myMultiplyFunction(i, j); // calling user defined function
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```