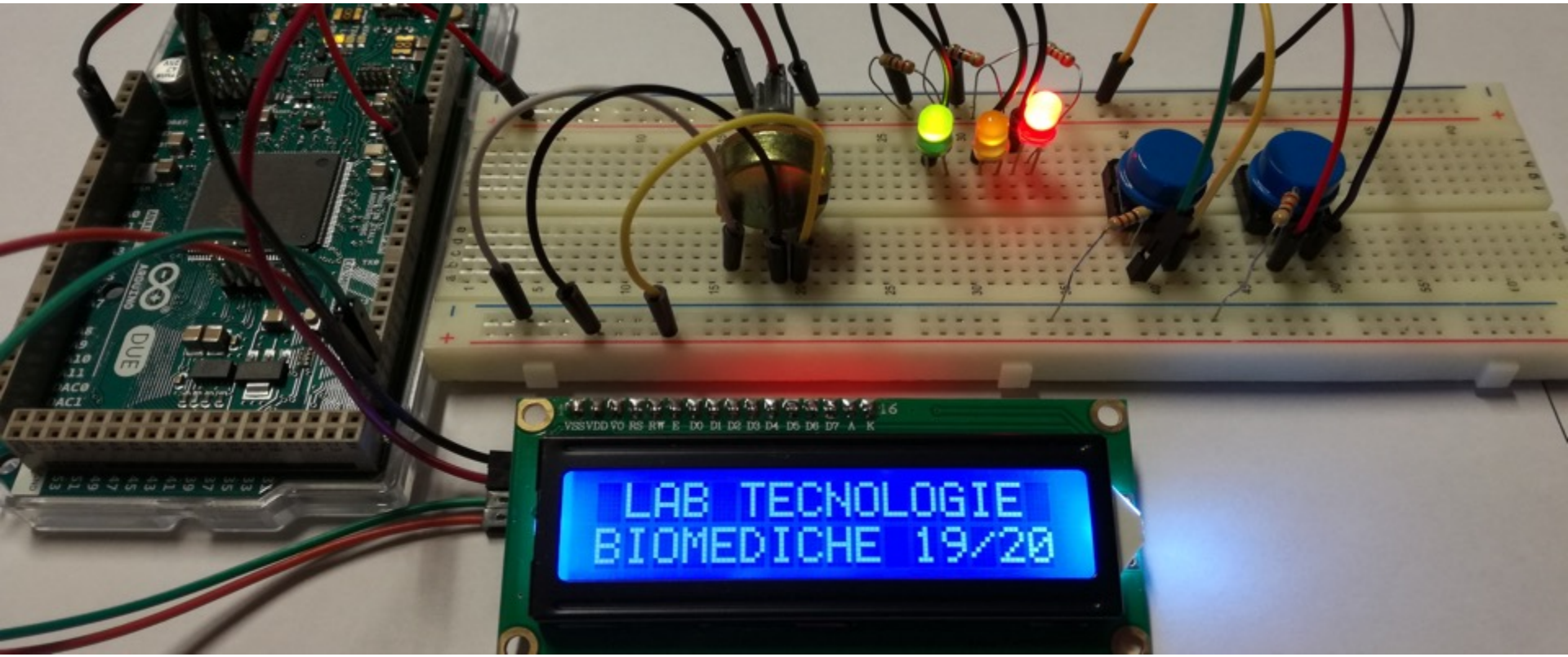
Electronic Prototyping

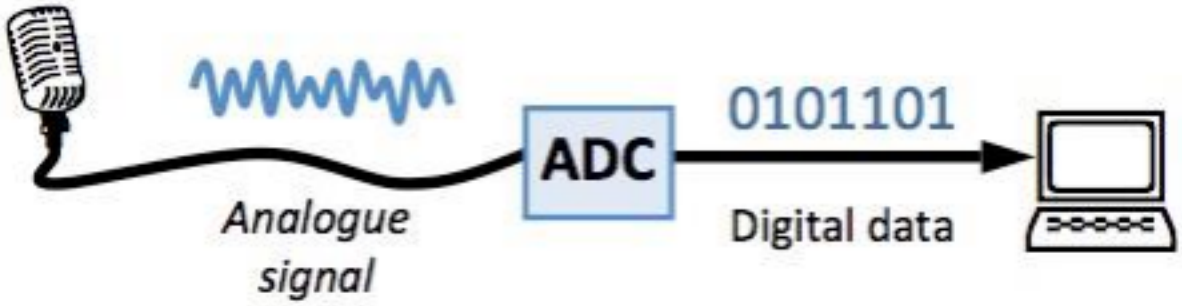
Analog to digital converters and libraries

Lesson 3



Analog to digital converters

Analog to digital converter



The Analog to Digital Conversion Process (1/2)

- **Sounds are analog - they are made of waves that travel through matter. People hear sounds when these waves physically vibrate their eardrums.**
- **Since Computers are digital devices, they cannot understand these continuous pressure variable analog signals, so they communicate digitally, using electrical impulses that represent 0s and 1s (i.e., through Binary).**

Binary Notations:

- **One binary digit (0 or 1) is referred to as a bit, which is short for binary digit. One bit can only be used to represent 2 different values: 0 and 1.**

The Analog to Digital Conversion Process (2/2)

- **To represent more than two values, we need to use multiple bits.**
- **Two bits combined can be used to represent 4 different values:**
  - 0 0, 0 1, 1 0, and 1 1.
- **Three bits can be used to represent 8 different values:**
  - 0 0 0, 0 0 1, 0 1 0, 1 0 0, 0 1 1, 1 0 1, 1 1 0 & 1 1 1.
- **In general, 'n' bits can be used to represent 2^n different values.**

2^0	1
2^1	2
2^2	4
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1024
2^{11}	2048
2^{12}	4096

Example of A/D Applications

- **Microphones**
  - Take your voice varying pressure in the air and convert them into varying electrical signals
- **Strain gages**
  - Determines the amount of strain (change in dimensions) when a stress is applied
- **Thermocouple**
  - Temperature measuring device converts thermal energy to electrical energy

What is an ADC?

An ADC (Analog-to-Digital-Converter) is a circuit which gets an analog voltage signal and provides (to software) a variable proportional to the input signal.

An ADC is characterised by:

- The **range** (in volts) of the input signal (typical [0,5V] or [0, 3.3V]).
- The **resolution** (in **bits**) of the converter

Example

- **Range** = [0,5V]
- **Resolution** = 12 bits

  Results in range [0, 2^{12}- 1] = [0, 4095]

  0V → 0 2.5V → 2048 5V → 4095

Analog → Digital Conversion
2 Step process:

- **Quantizing**
  - Breaking down analog value is a set of finite states
- **Encoding**
  - Assigning a digital word or number to each state and matching it to the input signal

Step 1: Quantizing (1/2)

Example

You have 0-10V signals. Separate them into a set of discrete states with 1.25V increments.

(How did we get 1.25V?

See next slide…)

Output States	Discrete Voltage Ranges (V)
0	0.00-1.25
1	1.25-2.50
2	2.50-3.75
3	3.75-5.00
4	5.00-6.25
5	6.25-7.50
6	7.50-8.75
7	8.75-10.0

Step 1: Quantizing (2/2)

The number of possible states that the converter can output is:

$$N=2^n$$

where n is the number of bits in the AD converter

Example: For a 3 bit A/D converter, $N=2^3=8$.

Analog quantization size:

$$Q=(V_{max}-V_{min})/N = (10V - 0V)/8 = 1.25V$$

Step 2 – Encoding

Here we assign the digital value (binary number) to each state for the computer to read

Output States	Output Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Accuracy of A/D Conversion

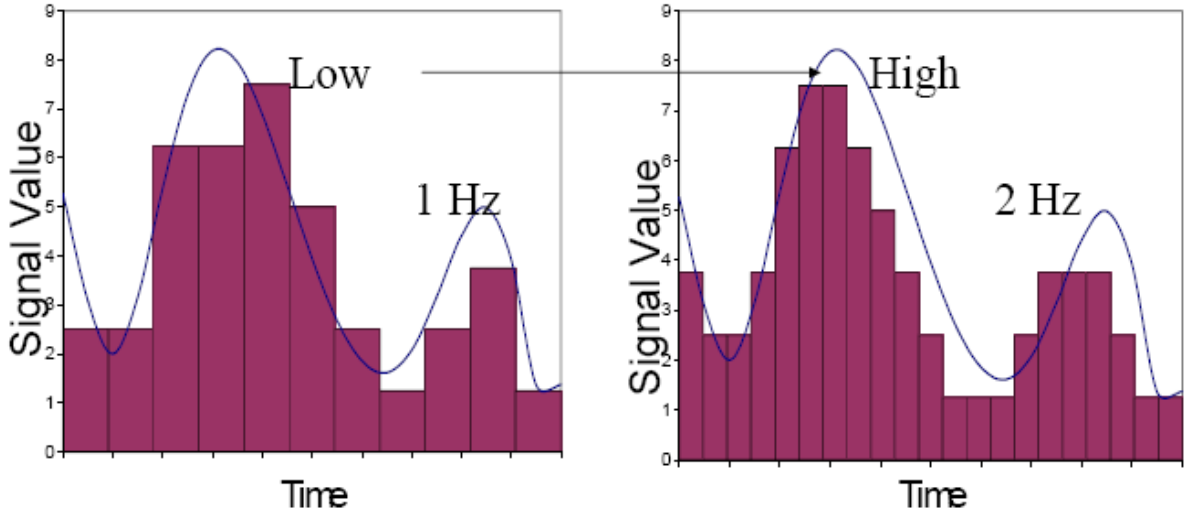
There are two ways to best improve accuracy of A/D conversion:

- increasing the resolution which improves the accuracy in measuring the amplitude of the analog signal.
- increasing the sampling rate which increases the maximum frequency that can be measured.

Resolution

- **Resolution**
  - (number of discrete values the converter can produce) = Analog Quantization size (Q)
  - (Q) = Vrange / 2^n, where Vrange is the range of analog voltages which can be represented
- **limited by signal-to-noise ratio (should be around 6dB)**
- **In our previous example:**
  - Q = 1.25V, this is a high resolution.
  - A lower resolution would be if we used a 2-bit converter, then the resolution would be $10/2^2 = 2.50$V.

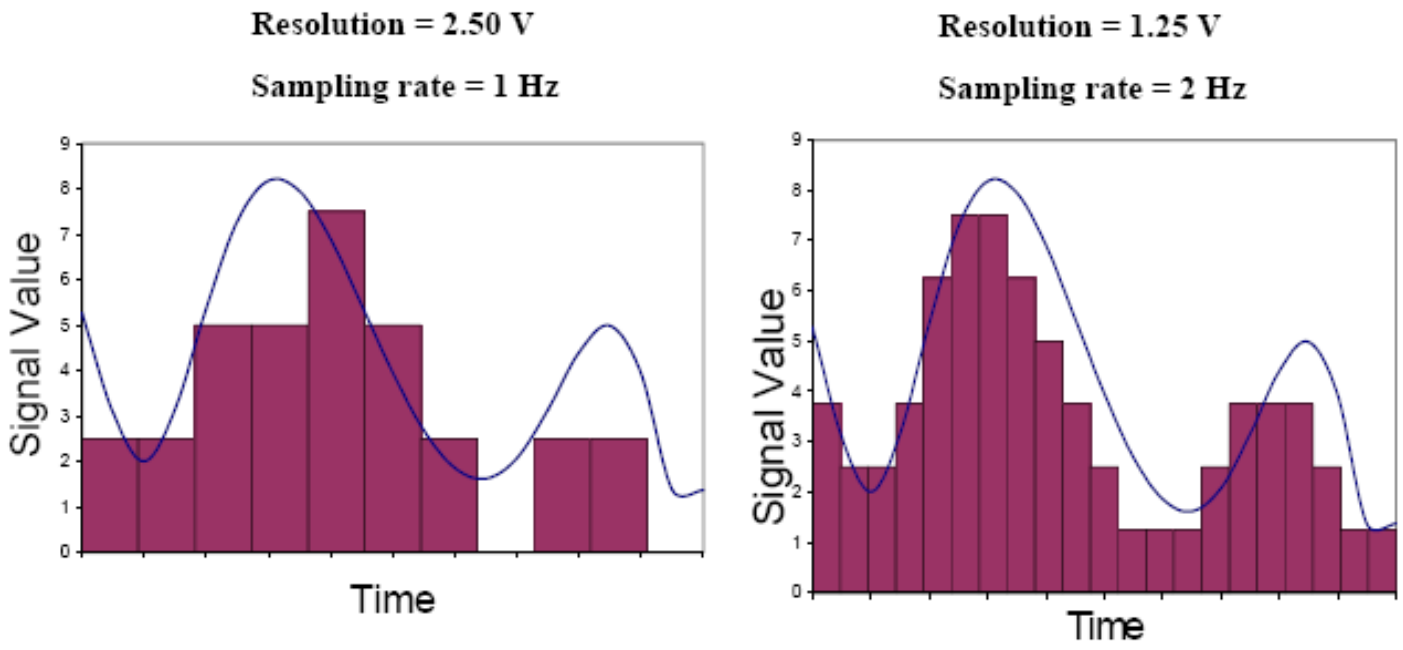
Sampling rate



Frequency at which ADC evaluates analog signal. As we see in the second picture, evaluating the signal more often more accurately depicts the ADC signal.

Overall Better Accuracy

Increasing both the sampling rate and the resolution you can obtain better accuracy in your AD signals.

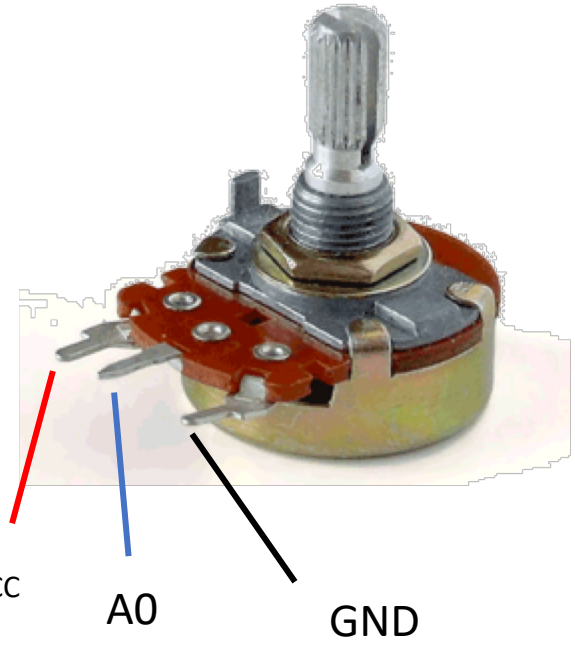


Arduino Due ADC

- **The Due has the following hardware capabilities:**
  - Resolution is defaults to 10 bits (returns values between 0-1023)
  - The Due boards have 12-bit ADC capabilities that can be accessed by changing the resolution to 12 [analogReadResolution(bits)]
    - This will return values from analogRead() between 0 and 4095.

Exercise 1: Potentiometer and LEDs

- **Make a traffic light using a potentiometer:**
  - turning the potentiometer, the leds will light up in sequence



Libraries

Arduino libraries

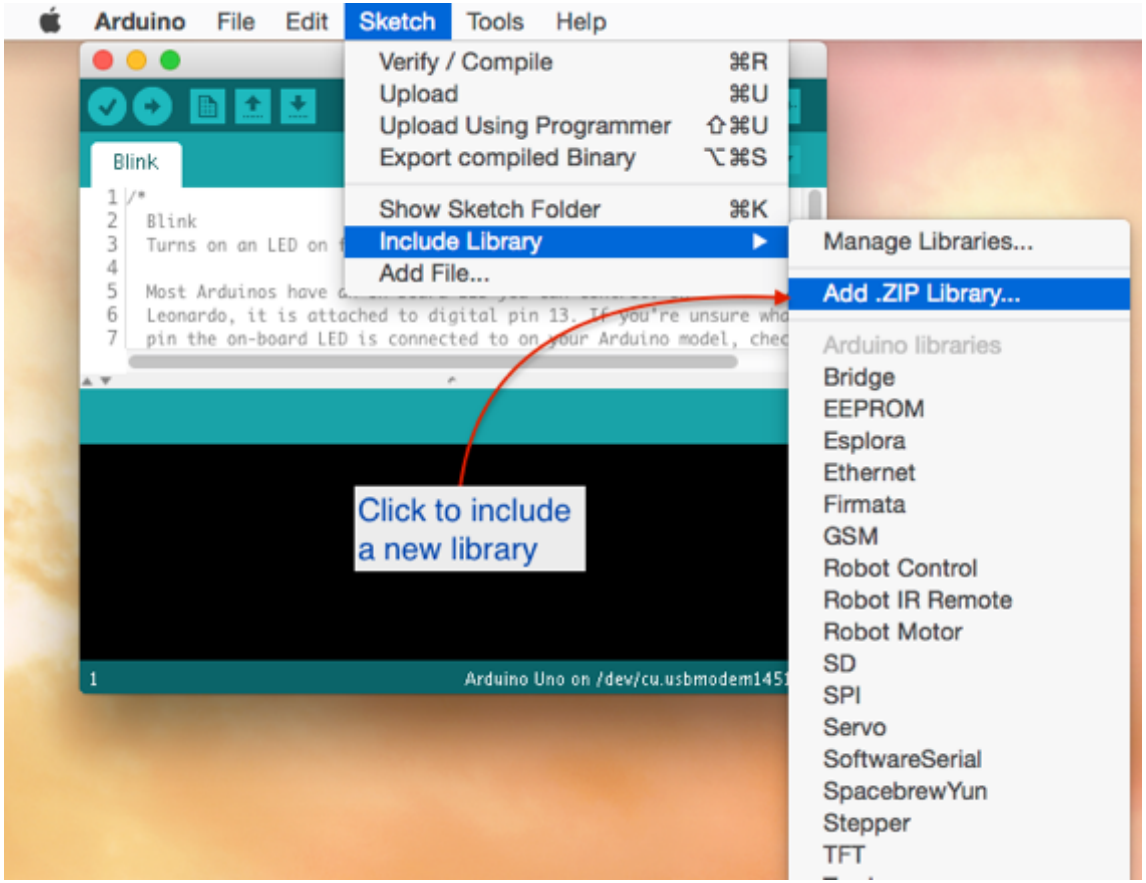
The Arduino environment can be extended through the use of libraries, just like most programming platforms.

Libraries provide extra functionality for use in sketches, e.g. working with hardware or manipulating data. To use a library in a sketch, select it from Sketch > #include Library.

Arduino libraries

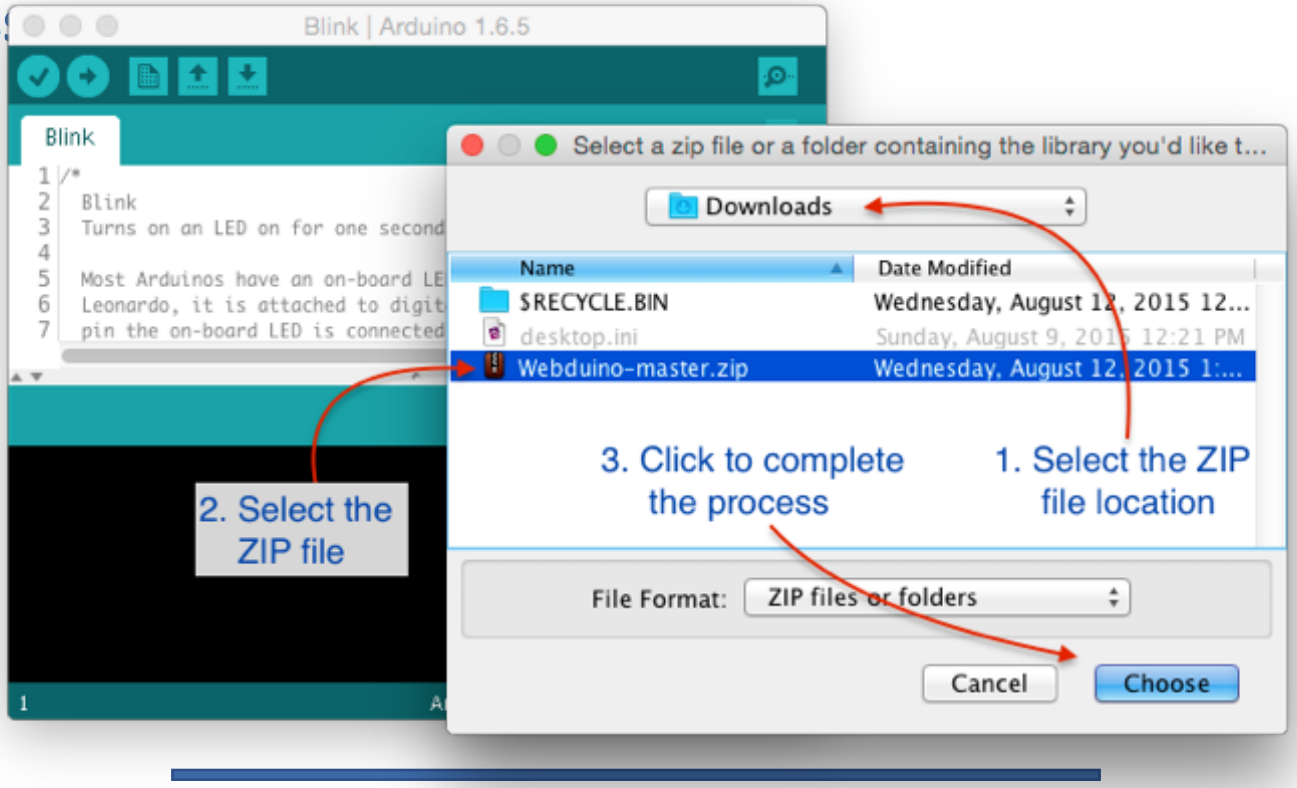
- **If there is a library that you need but is not included with the IDE, you can install it.**
- **Download the ZIP file on your computer. It doesn't matter what platform you are on; the libraries work the same regardless of whether you are on Windows, Mac or Linux.**
- **Also, do not worry about extracting the files from the ZIP archive. The newer versions of the Arduino IDE have an easy library installer that takes care of extracting the library from the ZIP file and copying the files to the right location.**
- **Assuming the library ZIP file is in your Downloads folder, start the Arduino IDE. Then click on "Sketch → Include Library → Add .ZIP Library…", like this:**

Add a new library



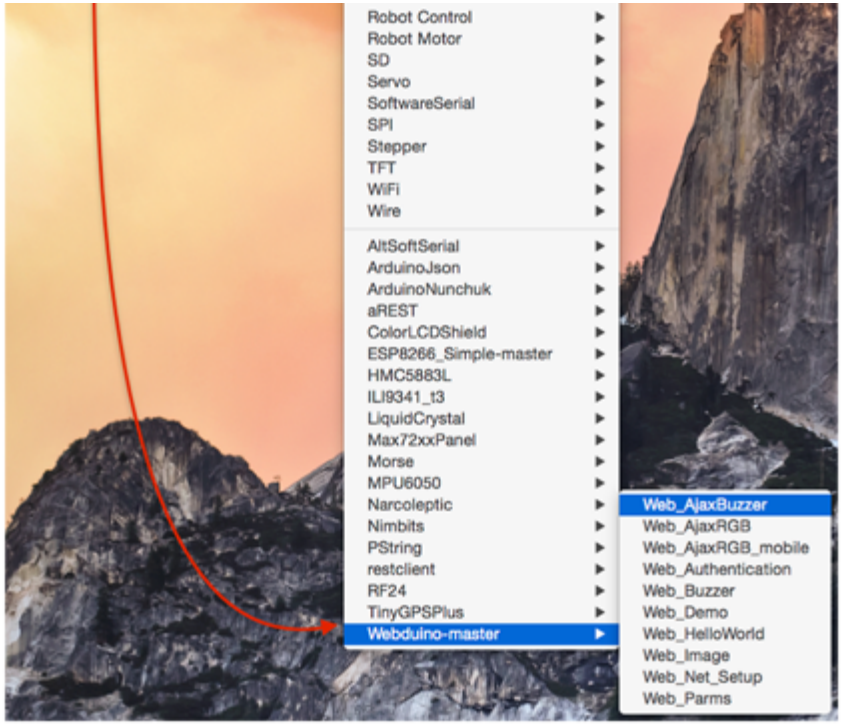
Add a new library

A new dialogue box will pop up. Browse to the location of the ZIP file, select it, and click on Choose to complete the proces



Add a new library

Go to File → Examples, and look at the bottom of the list for your new library:

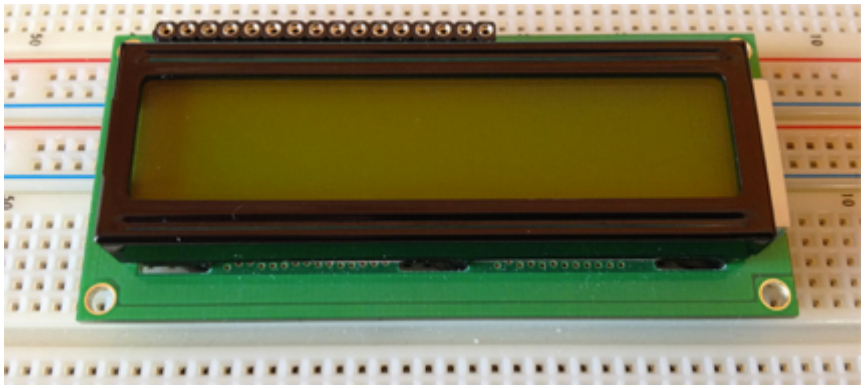


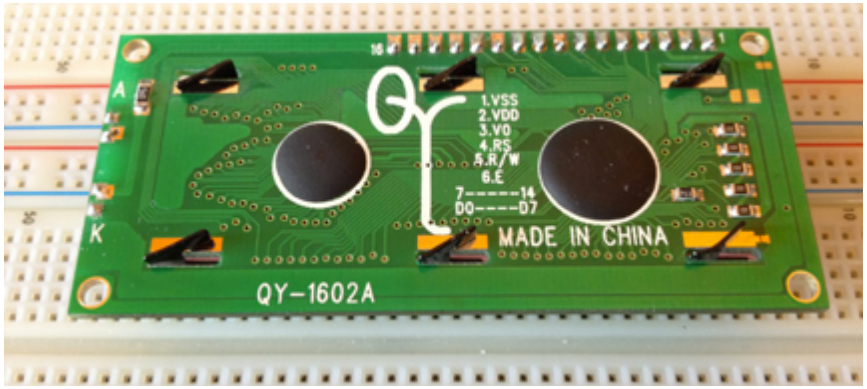
How to include a new library

#include <Name_of_library.h>

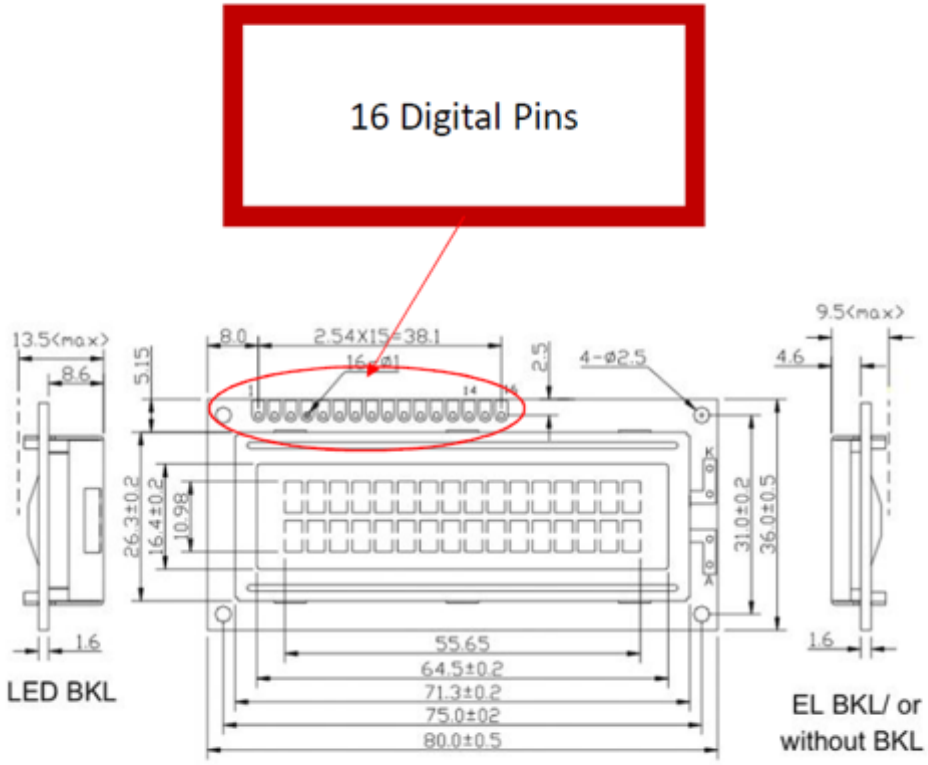
Monitor LCD

- **16 columns x 2 rows**



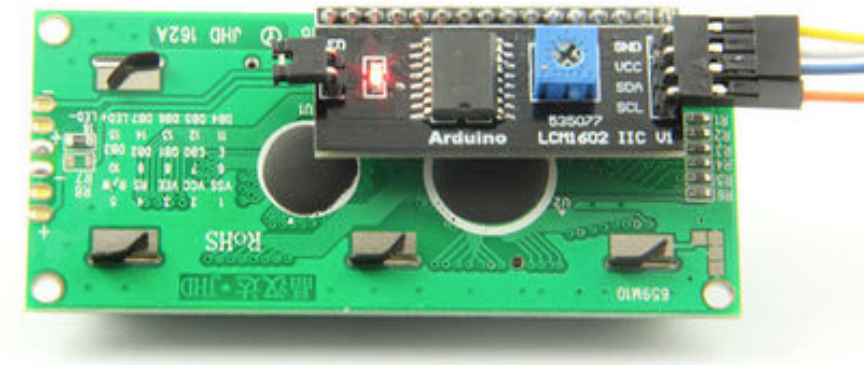


PIN description



PIN NO	Symbol	Fuction
1	VSS	GND
2	VDD	+5V
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	+4.2V for LED
16	K	Power supply for BKL(0V)

LCD monitor with I2C Driver



Board	I2C/TWI pins
Uno, Ethernet	A4 (SDA), A5 (SCL)
Mega2560	20 (SDA), 21 (SCL)
Leonardo	2 (SDA), 3 (SCL)
Due	20 (SDA), 21 (SCL), SDA1, SCL1

I2C communication

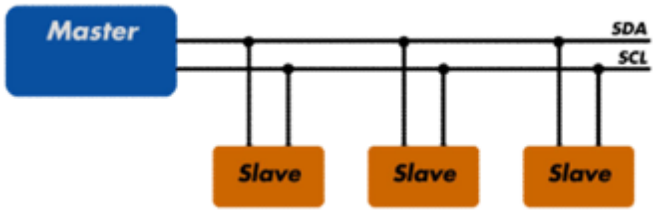
The I2C protocol involves using two lines to send and receive data:

- a **serial clock pin (SCL)** that the Arduino pulses at a regular interval,
- a **serial data pin (SDA)** over which *data is sent between the two devices*.

As the clock line changes from low to high (known as the rising edge of the clock pulse), a single bit of information - that w*ill form in sequence the address of a specific device and a command or data* - is transferred from the board to the I2C device over the SDA line.

***When this information is sent* - bit after bit -, the called upon device executes the request and transmits it's data back - if required - *to the board over the same line using the clock signal still generated by the Master on SCL as timing*.**

The *initial eight bits (i.e. eight clock pulses)* from the Master to Slaves contain the address of the device the Master wants data from. The bits after contain the memory address on the Slave that the Master wants to read data from or write data to, and the data to be written (if any).



Libraries

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

Exercise 2:
print “LAB TECNOLOGIE BIOMEDICHE 18/19” on a LCD display

Exercise 3:
print humidity and temperature from DHT11 sensor on a LCD display

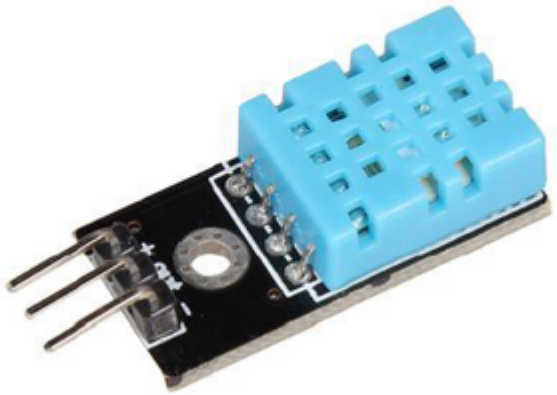
DHT11 sensor

The DHT11 is a relatively cheap sensor for measuring temperature and humidity.

The DHT11 has three lines:

- GND,
- +5V
- and a single data line.

Signal transmission range: 20m
Temperature range: 0-50°C
Humidity range: 20-90%RH



FIND THE DATASHEET

Libraries

```
#include <DHT11.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
```