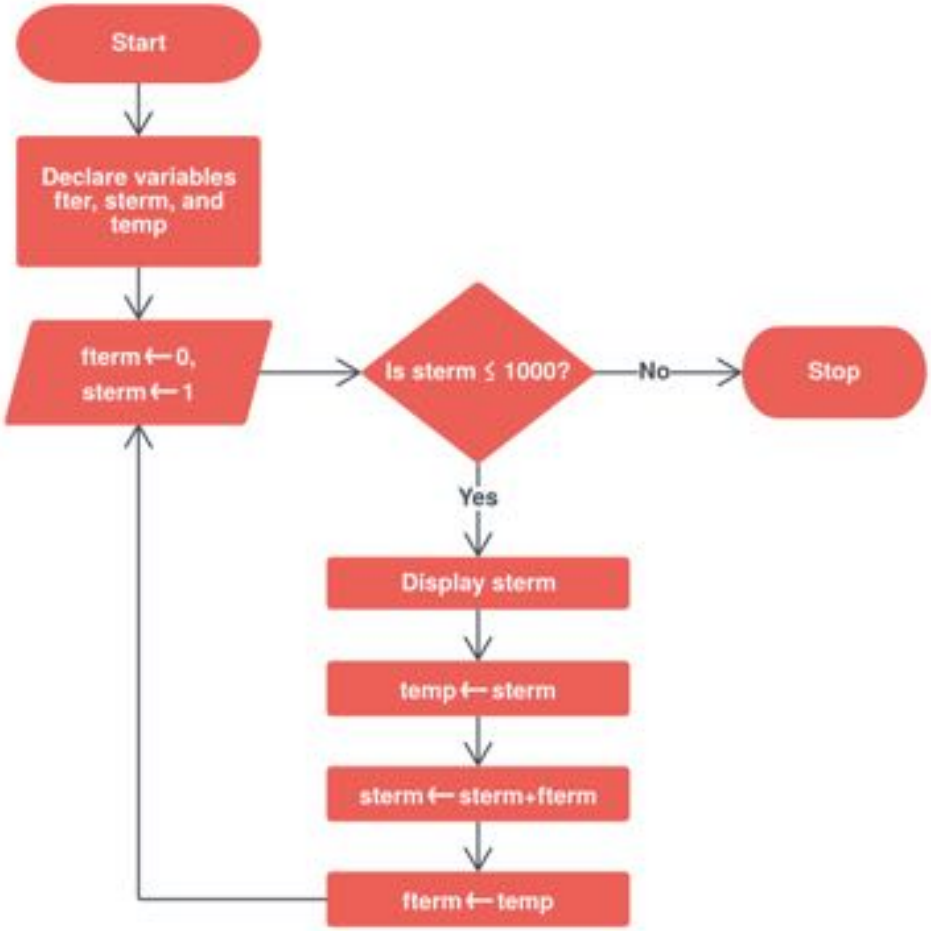
Electronic Prototyping

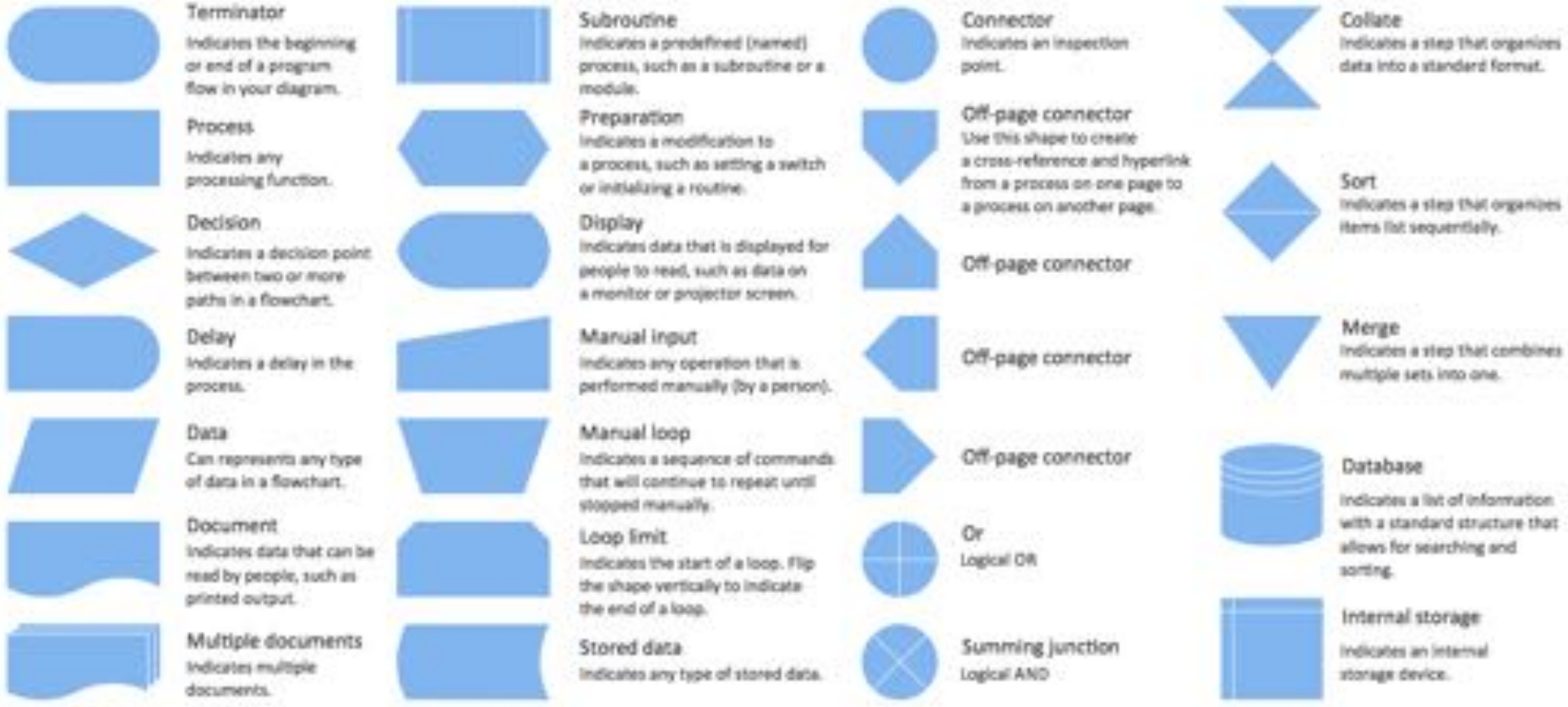
BUTTON and libraries

Lesson 2

Flow chart


Start
Declare variables fter, sterm, and temp
fterm ← 0, sterm ← 1
Is sterm ≤ 1000?
No
Stop
Yes
Display sterm
temp ← sterm
sterm ← sterm+fterm
fterm ← temp

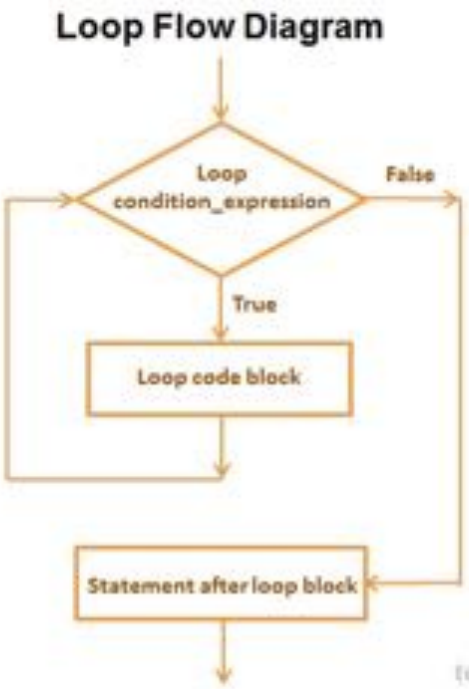
Flow Chart elements



Control structure

- Loops
  - For
  - While
  - Do … while
  - If … else

- https://www.arduino.cc/reference/en/#structure

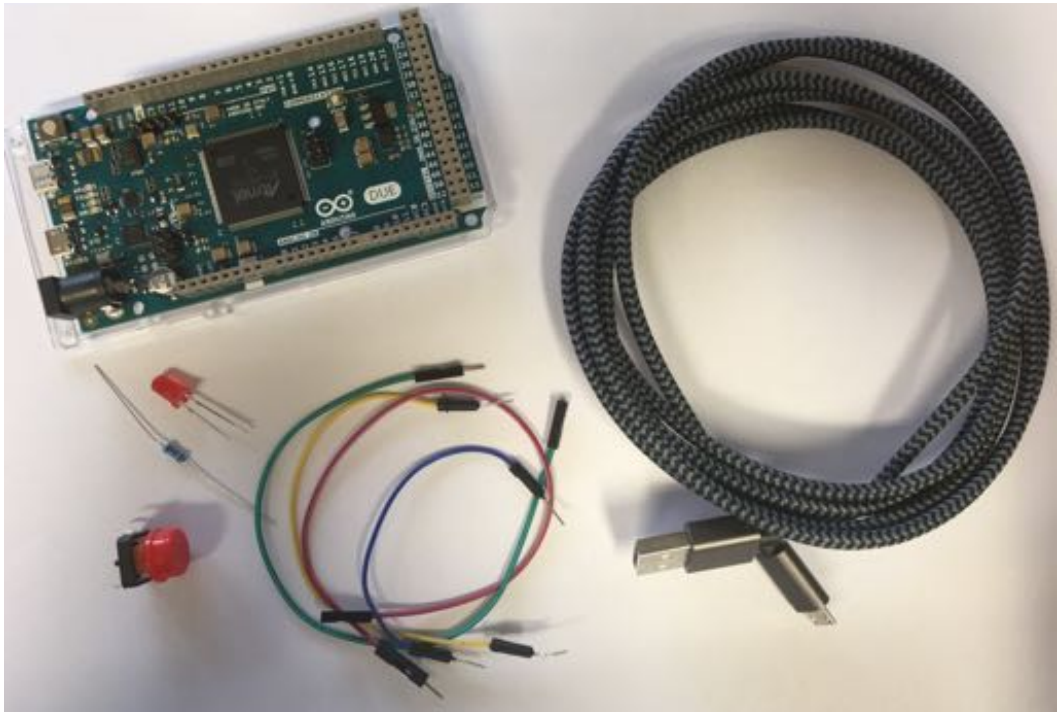



while (not edge) {
run();
}
do {
run();
} while (not edge);

Turn on a led with a button

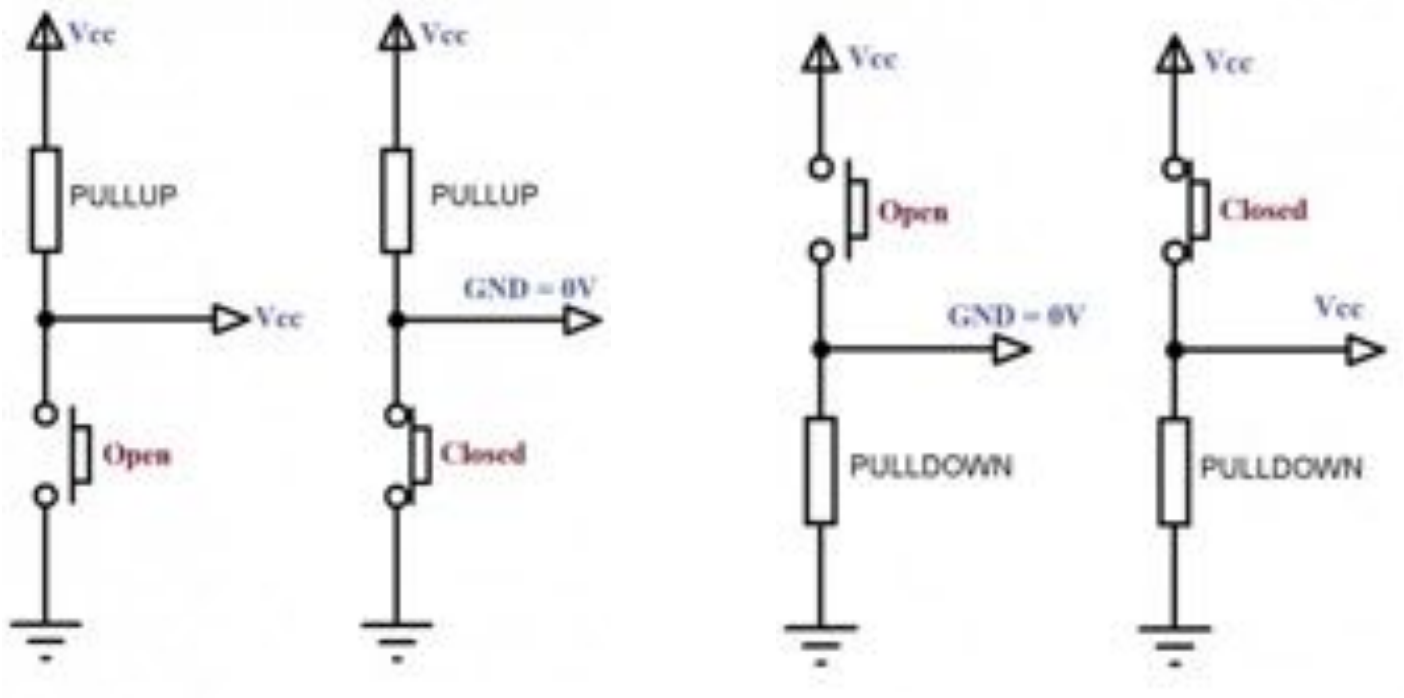
What are we going to use?

- Arduino DUE
- Breadboard
- USB Cable
- LED
- Button
- 10k resistor
- 220 resistor
- cables



How to connect a button?

- PULL-UP and PULL-DOWN Resistor



Internal PULL-UP Resistor

- Digital Pins of Arduino can be configured as
  - **OUTPUT**,
  - **INPUT** or
  - **INPUT_PULLUP**

  mode using **pinMode()** function.
- INPUT_PULLUP mode is used to enable the Internal PULL-UP Resistor.
- The value of Internal PULL-UP resistor of Arduino Uno is about 20-50kΩ
- The value of Internal PULL-UP resistor of Arduino DUE is about 100kΩ

Example 1: Arduino Code

- Turn on a led with a button:
  - If Button pressed → Turn on a LED
  - Else → Turn off a LED

Open Example 1 in the shared folder

Example 2: Arduino Code

- Use the button as a switch to turn on a LED
  - Button press → turn on the led keeping it on when it is released
  - Button press the second time → turn off the led

Open Example 2 in the shared folder

Example 2bis: Arduino Code

- Use the button as a switch to turn on a LED
  - Button press → turn on the led keeping it on when it is released
  - Button press the second time → turn off the led

Debouncing

https://www.youtube.com/watch?v=rvIRQ390mtc

Open Example 2bis in the shared folder

Exercise

Create a program that performs this function:

- **when a button is pressed the LED diode flashes,**
- **when the button is pressed a second time the LED stops blinking and it is always turned on**

Libraries

Arduino libraries

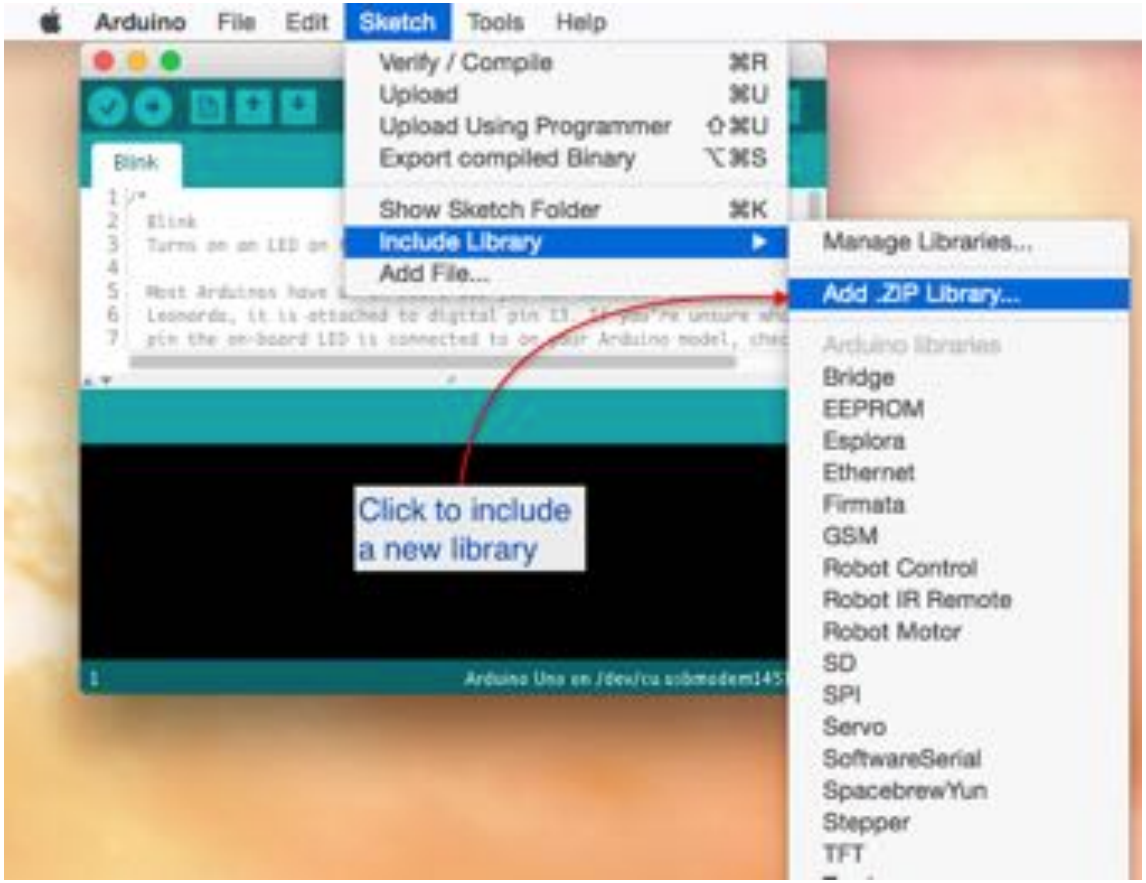
The Arduino environment can be extended through the use of libraries, just like most programming platforms.

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

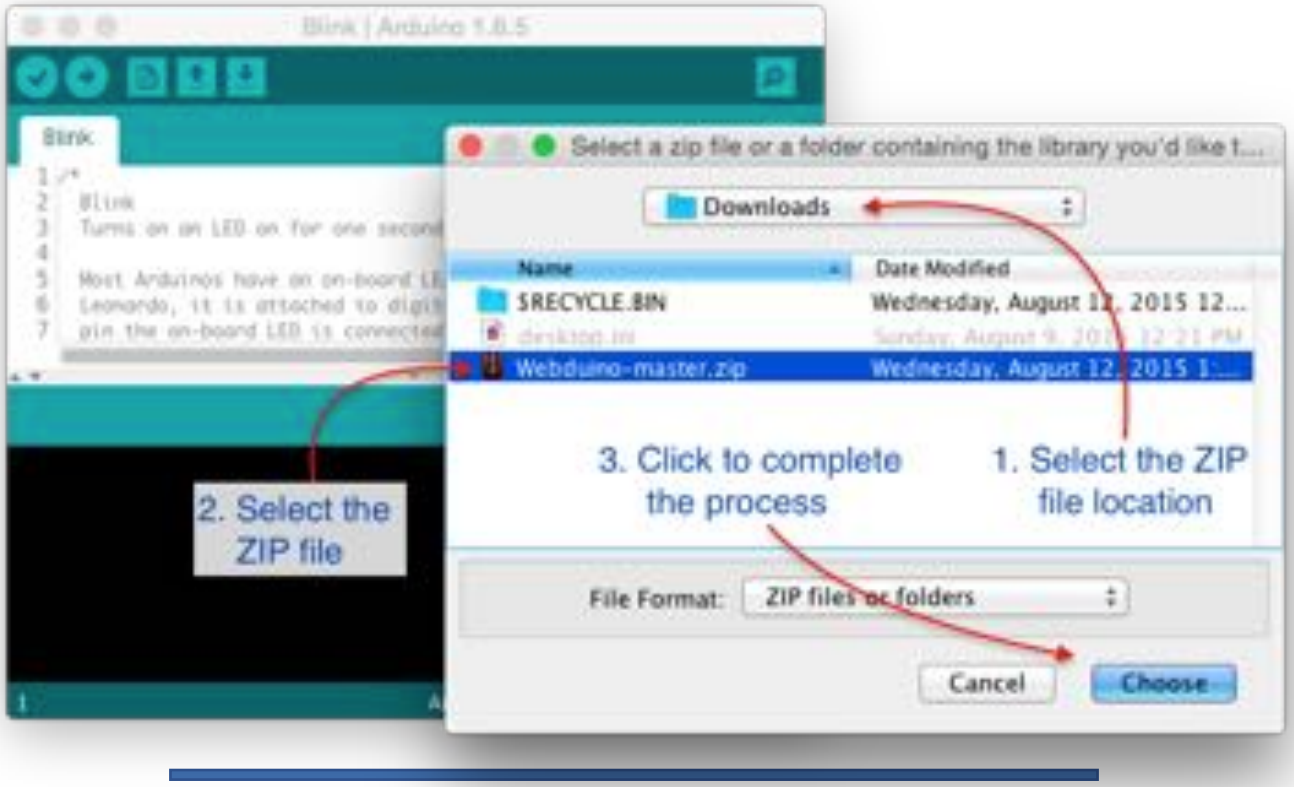
- If there is a library that you need but is not included in the IDE, you can install it.
- Download the ZIP file on your computer. It doesn't matter what platform you are on; the libraries work the same regardless of whether you are on Windows, Mac or Linux.
- Also, do not worry about extracting the 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



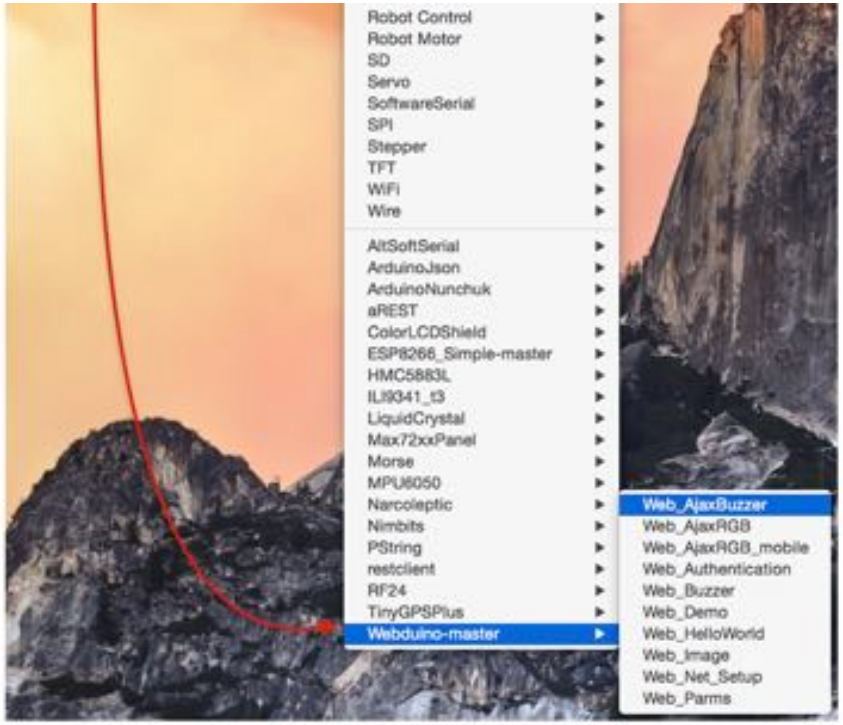
Add a new library

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



Add a new library

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



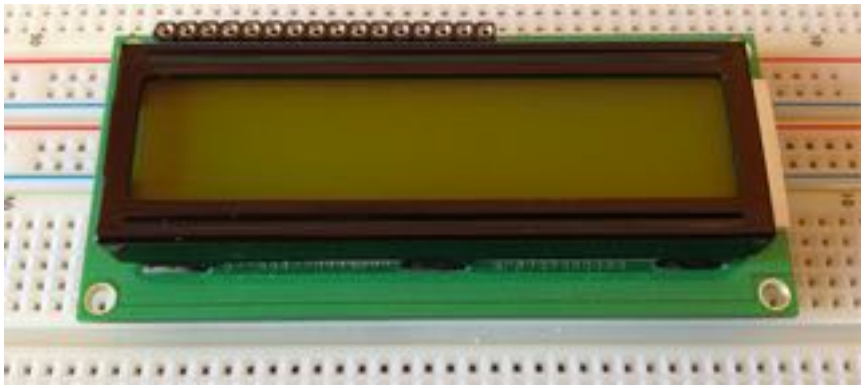
How to include a new library in your core

#include <Name_of_library.h>

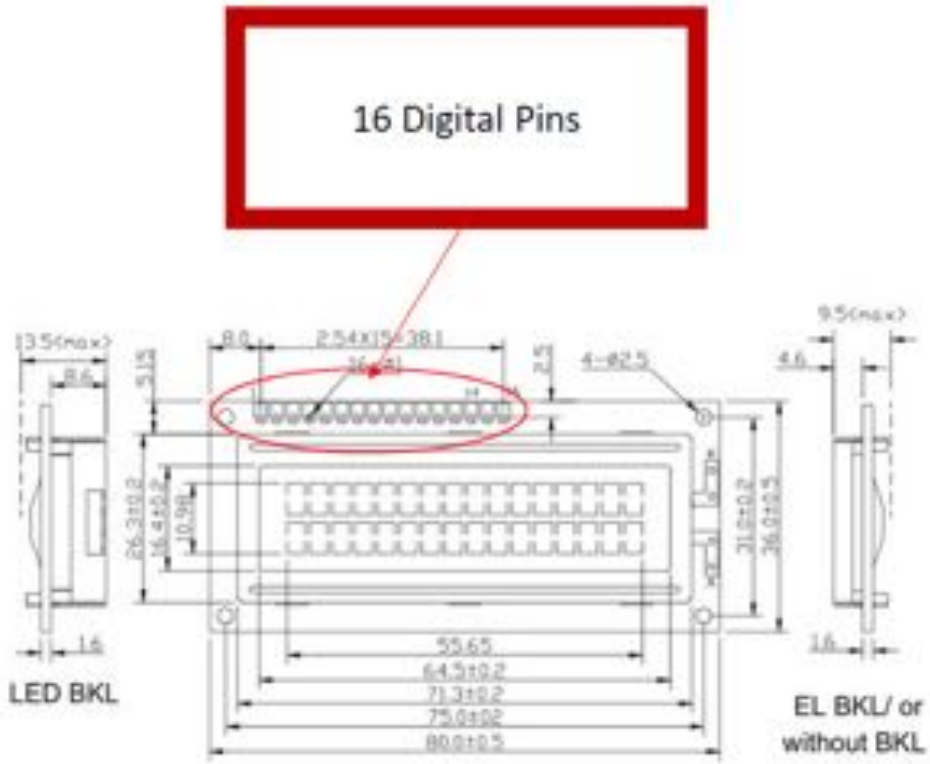
Exercise:
print “LAB TECNOLOGIE BIOMEDICHE 18/19” on a LCD display

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

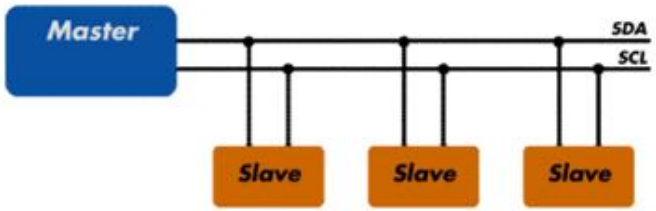
The I2C protocol involves using **two lines to send and receive data**:

- 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:
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>
```