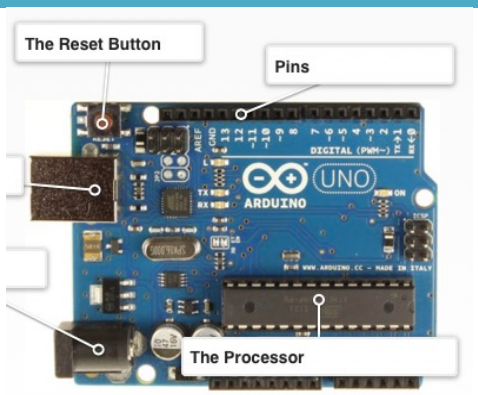
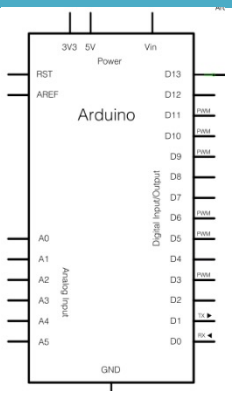
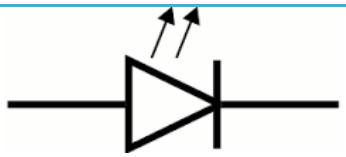
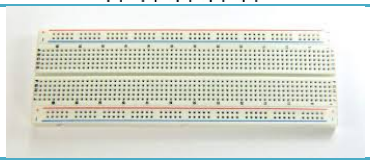
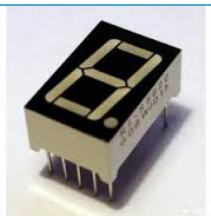
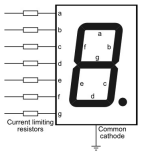
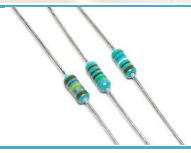
GETTING STARTED WITH ARDUINO

OBJECTIVE:

The aim of this lab is to learn basics on the use of the Arduino module and very common electronic components. You will learn how to set up Arduino to control a set of LEDs.

Please ask for teacher validation every time you plug in the Arduino module.

LIST OF MATERIALS REQUIRED:

Component	Photo	Schematic
Arduino module (Uno or Mega)		
Set of Leds		
Breadboard		
Wiring Cables		
Seven segments display		
Resistors		

TODO: Check that you have all required materials

ARDUINO BOARD DESCRIPTION

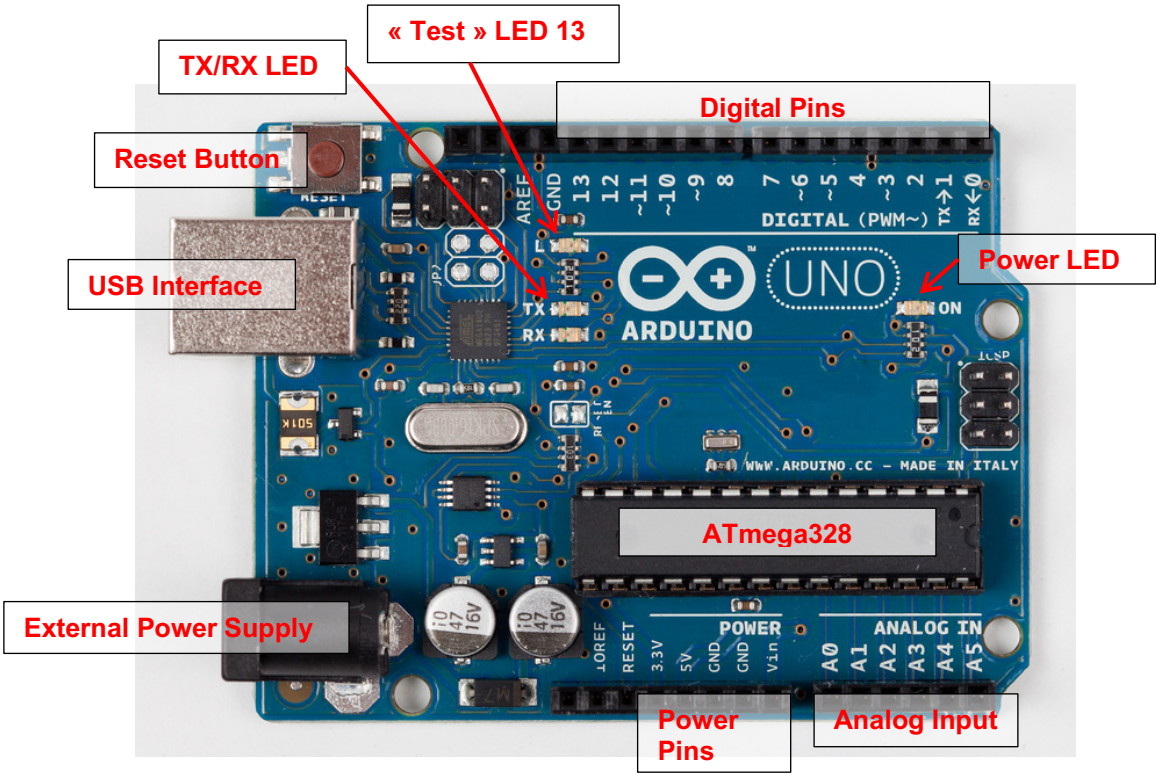


Figure 1 : Arduino Board

The Arduino Uno (see figure 1) is based on a Processor, the ATmega328 which need to be powered. The USB Interface is used to connect the Arduino Board to the computer in order to power and program the Processor. As the USB provides the power, you don't need an external Power Supply when you are programming the Arduino.
In this tutorial, you will use the USB Interface, the Digital Pins and the Power Pins 5V and GND.

1 SWITCHING ON AND OFF A LED WITH THE ARDUINO MODULE

Key steps:

1. Wiring properly the LEDs with the module
2. Writing a piece of code
3. Connecting the Arduino and uploading the code to the processor

1.1 THE WIRING DIAGRAM

The key component of the Arduino module is a programmable microprocessor. This processor can perform instructions as indicated by a program. An example of instructions may be to define an output voltage on a pin of the processor. Each pin of the processor is denoted by a number (that you can read on the card) and can be set up to **LOW** or **HIGH** levels (usually LOW refers to 0V and HIGH to 5V).

The first part of this work consists in setting up a controller pin to HIGH in order to switch on a LED ("Light Emitting Diode") that will be connected to this pin.

LEDs have an **anode** and a **cathode** pin as displayed below.

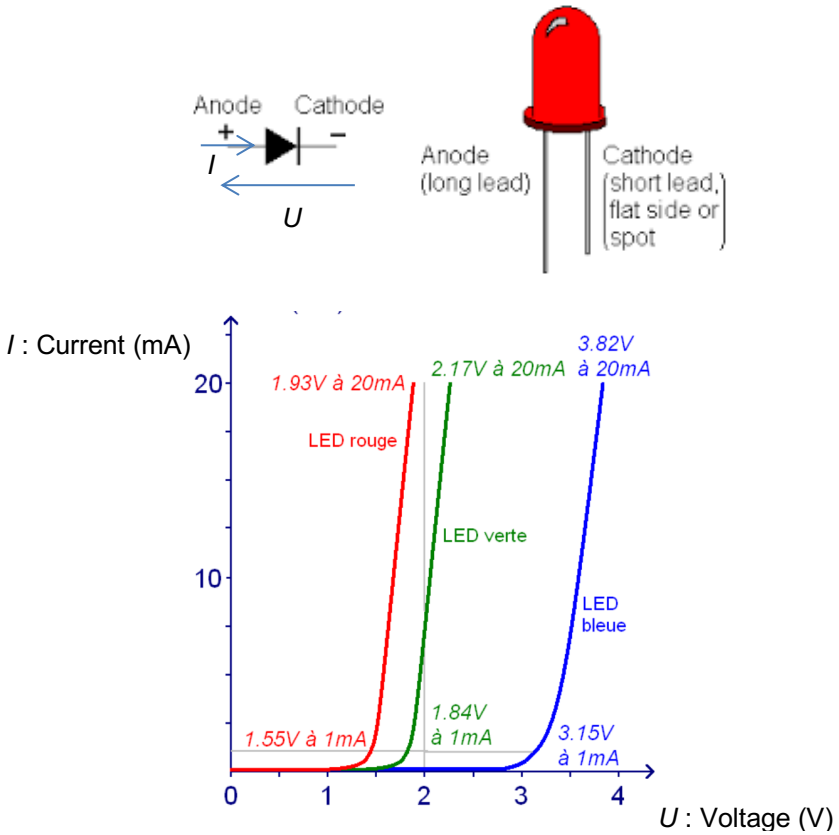


Figure 2: Behavior of LEDs

In a normal usage (called polarization) the tension at the anode should be higher than the tension at the cathode (as shown in the figure): under such configuration the LED emits light.

A LED has a given forward voltage (a voltage in which it works properly) that is indicated in its datasheet[1]. By reading the datasheet you will find that it is usually between 1.2V and 1.8V (3.2V for the blue LED) with an

[1]We recommend you to read datasheet of any component you plan to use. They indicate normal usage diagram, tensions, specificities of the component, etc.

associated current of 20mA. Since the Arduino HIGH state provides 5V (500ma) the LED **cannot** be directly connected to the 5V and the ground signals (LOW state). **This will certainly burn the LED!**

Figure 3 shows the diagram of wiring a led to the controller. A resistor is wired in serial (one after each other) with the LED in order to reduce the current but also to prevent shortcuts. A 220 Ohms resistor (minimum recommended) will be sufficient.

Definition: A resistor is an electrical component that limits or regulates the flow of electrical current in an electronic circuit. The current through a resistor is inversely proportional to its resistance, and directly proportional to the voltage across it. (Ohm's Law).

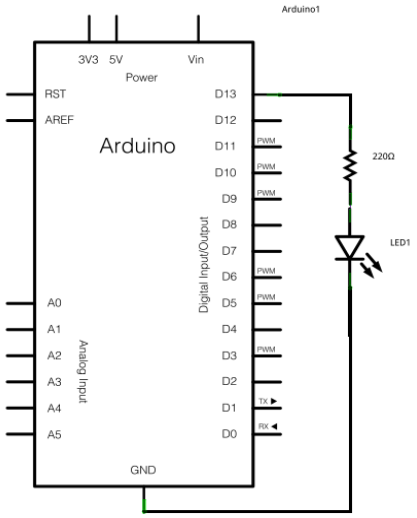
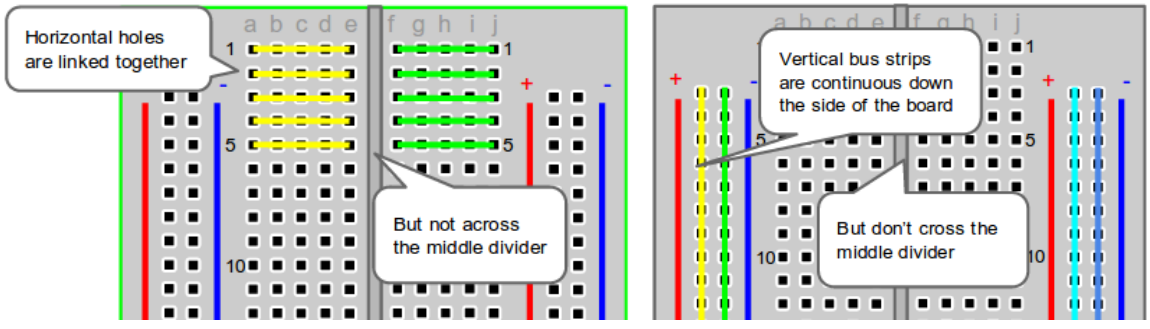


Figure 3 : Circuit diagram

The diagram of your first circuit is presented on the figure 3. Note that the components (a resistor and a LED) are wired via a breadboard. A breadboard allows connecting components and creating circuits without the need to solder.

How a breadboard works:



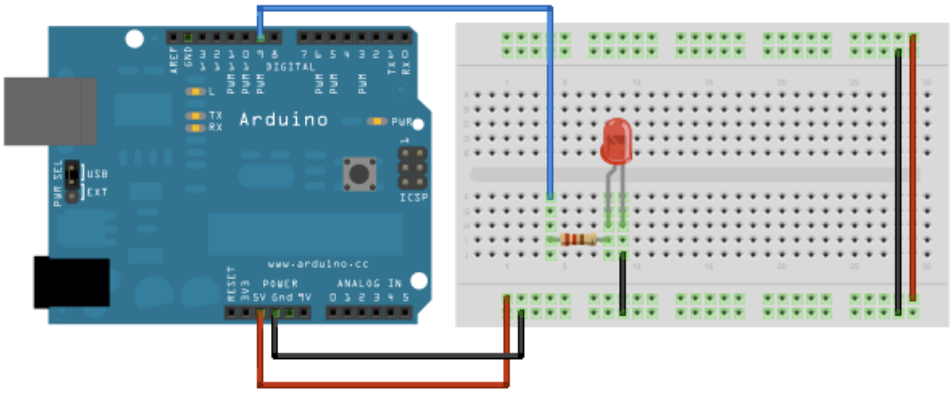


Figure 4 : Physical diagram of the circuit

By analyzing the diagram, we observe that PIN #9 of the controller is connected to the resistor that is connected to the Anode of the LED. The cathode of the LED is wired to the Gnd PIN (called Ground). The 5V and Ground are connected to the upper and lower sides of the breadboard.

TODO: Set up the wiring and valid with teacher.

The next step is to use the Arduino software to program the controller. The program should set up a HIGH level of tension on the PIN #9 in order to switch on the LED and a LOW level of tension in order to switch off the LED.

1.2 THE ARDUINO SOFTWARE

You will need a computer with Arduino software installed. You can find the software and installation instruction on the official website:

https://www.arduino.cc/en/Main/Software

TODO: Install the Arduino software

1.3 CONNECT BOARD AND SETUP COMMUNICATION

TODO: Connect the Arduino board to your computer using the USB cable.

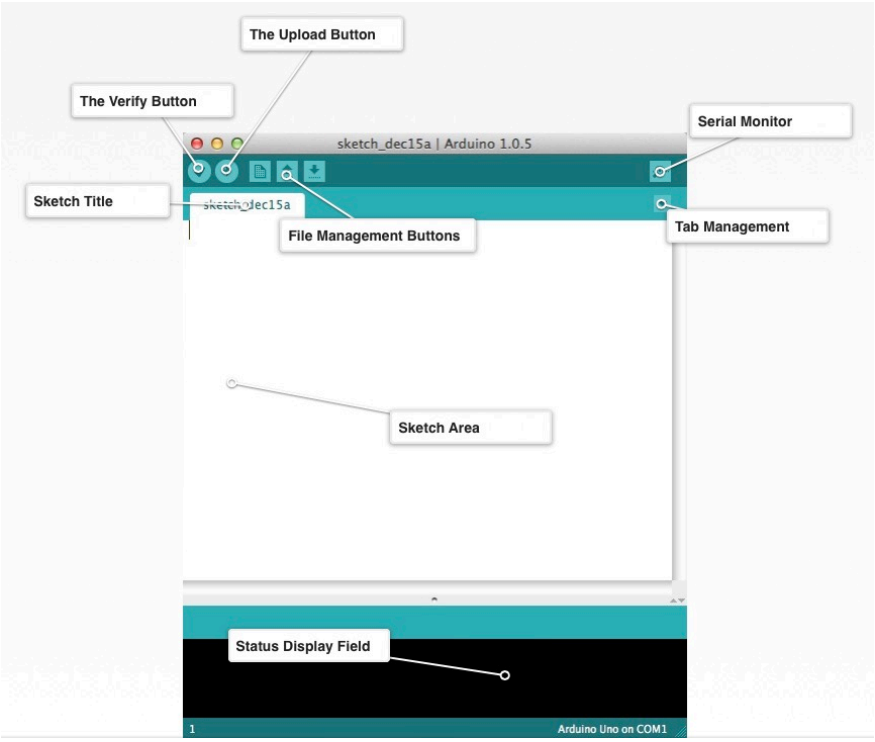
You may need additional drivers for setting up the connection between the Arduino and your computer. If drivers are not installed check up the driver section

https://www.arduino.cc/en/Guide/Windows or
https://www.arduino.cc/en/Guide/MacOSX

If you use a UNO R3 ship, you will need to install the driver for the WCH USB-SERIAL communication. You will find it on the following link:

http://www.wch.cn/download/CH341SER_MAC_ZIP.html
http://www.wch.cn/download/CH341SER_ZIP.html

TODO: Run the Arduino software



The software is very easy to use. The GUI is very simple with a limited number of features.

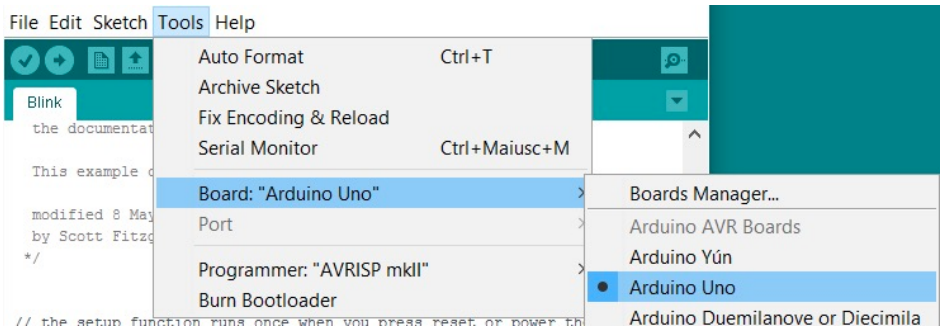
The **sketch area** contains the code you will write or copy that will be transferred to the microcontroller.

The **verify button** helps checking if the code is correct and could be understood by the controller.

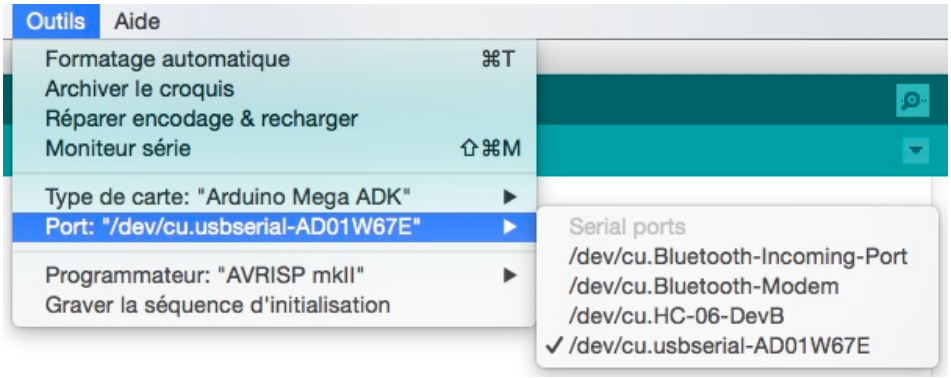
The **upload button** will load the instruction to the controller.

The **status display field** contains current status and error messages (the code cannot upload, errors exists in the code, etc.).

TODO: Set up the board type as indicated in the picture below



TODO: Select the right serial port (can be usbserial or COM3-4-5)



1.4 WRITING YOUR FIRST ARDUINO CODE

The sketch area contains instructions (code) for the actions you want the processor to perform. Arduino code is necessarily composed of two main parts (called functions) as indicated below:

```
void setup()
{
     // SOME CODE HERE
}

void loop()
{
  // SOME OTHER CODE HERE
}
```

TODO: Copy paste this code to a sketch area and check that it is valid.

The // symbols means the line is a comment and will be omitted when checking the validity of the code (it will not be taken into account by the processor). It is used for clarity and readability of the code you write.

The **setup** function is called when the program starts (only once) and is used for initialization. For example, we can indicate that the pin #9 of the Arduino will be used as an **output**[2].

```
void setup()
{
     pinMode(9,OUTPUT); //Pin #9 will be used as output
}
```

The instruction `pinMode(N,OUPUT)` tells the processor that we will work with the pin #9 and that we aim to setup ouput tension on the pin.

*At this point, nothing will happen when loading the software. In fact, we did not tell if the pin should be set to 5V or 0V. We will complete the program with such instruction in the **loop** function that allows the program to change and respond (by constantly looping through instructions it contains).*

In order to switch on the LED, we can add the `digitalWrite(9,HIGH)`; line. It indicates Arduino to send the signal HIGH (5V) to the pin #9. Whenever you would like to switch of the LED write LOW.

```
void setup()
{
     pinMode(9,OUTPUT);
}
```

[2] Note that some pins may be setup as inputs. In this case we will be able to read the tension of the pin of the arduino. It is used when sensors (microphone, temperature) are connected to pins, and we aim to identify the value they capture.

```
void loop()
{
    digitalWrite(9,HIGH);

}
```

TODO: Check and load to code to the module. Verify that the LED correctly switches on. Change the HIGH to LOW and check that the LED correctly switches OFF.

To go further in your understanding of Arduino language, you can open the example sketch >01.Basics > Blink. Try to understand the code before loading it. You may also try to work with multiple LEDs!

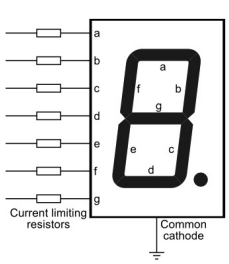
2 DISPLAY NUMBERS WITH A SEVEN SEGMENTS DIGIT

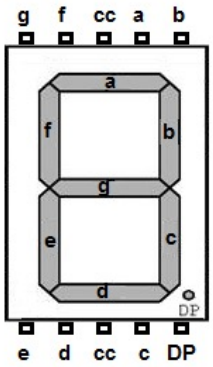
You may have already seen seven digits segments for example in alarm clocks. Such component is only a set of LEDs (one by segment) with particular shapes that allows displaying numbers. In this section, you will learn how to display particular number on such component.

2.1 WIRING THE COMPONENT

A seven segments digit is composed of seven LEDs connected together with a common anode or cathode.

The schematic below indicates a 7 segments digit with common cathodes. Under this assumption, in order to light a particular segment you must send a HIGH-level signal to its pin numbered from **a** to **g**. Each pin corresponds to a particular segment (a particular LED).





In order to display a '0', one must set every segment but **g** to a HIGH state. For displaying '1' the segments **b** and **c** must be HIGH and all others must be low. Since every segment is equivalent to a LED, resistors must be connected to each pin that corresponds to a LED anode. The common cathode can be connected to the ground.

Segment a	pin 2
Segment b	pin 3
Segment c	pin 4
Segment d	pin 5
Segment e	pin 6
Segment f	pin 7
Segment g	pin 8
Segment dp[3]	pin 9

TODO: Connect the 7 segments digit to the Arduino module. Do not forget resistors! Follow the table.

[3] The dp pin of the display refers to decimal point.

2.2 CODE TO DISPLAY A DIGIT

We will first identify each pin number with a given segment. For this purpose, we will use constants such as indicated below. A constant allow manipulating a name instead of a given value. It increases readability of the code.

```
/*Pin number 2 will now be called segment_a. This will make easier the rest of
the code, you will not need to remember the pin #.
*/
const int segment_a = 2;

void setup()
{
     pinMode(segment_a,OUTPUT);// segment_a (pin #2) is an output
}

void loop()
{
      digitalWrite(segment_a,HIGH);// segment_a is on!

}
```

We will now write a full code to display the number '0' to the digit.

```
const int segment_a = 2;
const int segment_b = 3;
const int segment_c = 4;
const int segment_d = 5;
const int segment_e = 6;
const int segment_f = 7;
const int segment_g = 8;
const int segment_dd = 9;

void setup()
{
     pinMode(segment_a,OUTPUT);
     pinMode(segment_b,OUTPUT);
     pinMode(segment_c,OUTPUT);
     pinMode(segment_d,OUTPUT);
     pinMode(segment_e,OUTPUT);
     pinMode(segment_f,OUTPUT);
     pinMode(segment_g,OUTPUT);
     pinMode(segment_dd,OUTPUT);
}

void loop()
{
      digitalWrite(segment_a,HIGH);
      digitalWrite(segment_b,HIGH);
      digitalWrite(segment_c,HIGH);
      digitalWrite(segment_d,HIGH);
      digitalWrite(segment_e,HIGH);
      digitalWrite(segment_f,HIGH);
      digitalWrite(segment_g,LOW);
      digitalWrite(segment_dd,LOW);

}
```

TODO: Find the code to display number 7 on the digit.

Note that this code can be simplified in order to not repeat same line multiple times. This requires a bit of more programming (a for loop) that you are not assumed to know at this point.

2.3 USING A PREDEFINED LIBRARY

Writing code for displaying every single digit may be a very long task. Furthermore, it is very likely that someone has already faced the same need. To save time and work, librairies (already developed functions) exist and can be used into our code to make it easier to work with the component. In this section we will see how to install, import and use a library to display whatever digit we want.

You will find detail on the seven Segment library on this page:
http://playground.arduino.cc/Main/SevenSegmentLibrary

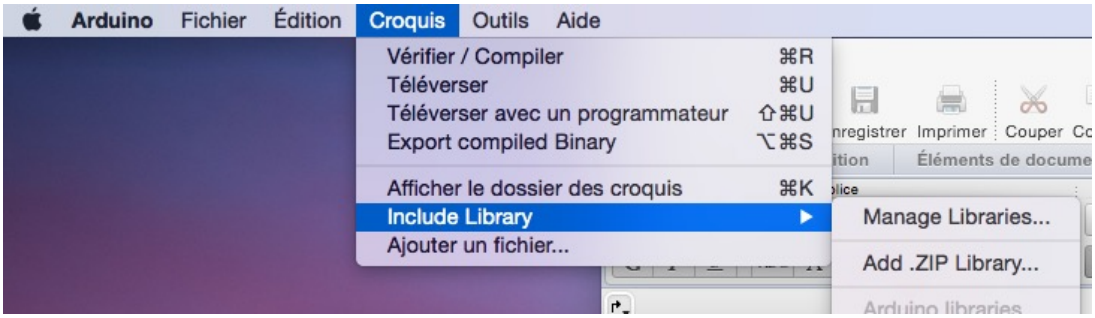
As already said, libraries are programs that have been written to produce some elementary tasks that are often needed. SevenSegmentLibrary provides solution to use a Seven Segment component easily.

Before using the library, you need to install it. For this purpose, copy the SevSeg folder into your Arduino `sketchbook\libraries` folder. More detailed instructions are available online:
http://arduino.cc/en/Guide/Libraries.

ToDo: Install the library

Any library will require the same steps for installation. Before starting to write complicated code for manipulating a particular component, think of existing libraries that could make your work easier!

From the Arduino software, you need to include the library. You can do this via the interface as shown below. Click on the seven segments library to import it.




```
//The next line indicates that the library will be used, it is automatically
//included in your sketch when you accomplish the upper step.
#include "SevSeg.h"

//This line will allow us to manipulate the functions of the //SevSeg using the
//sevseg variable.
SevSeg sevseg;

const int segment_a = 2;
const int segment_b = 3;
const int segment_c = 4;
const int segment_d = 5;
const int segment_e = 6;
const int segment_f = 7;
const int segment_g = 8;
const int segment_dd = 9;
```

A line of code appears at the top of the sketch indicating that you will work with a SevSeg library. The second line gives a name to the seven segment, this will allow to manipulate the component and ask for its services.

We will now have a look on the **setup** function of the sketch.

```
void setup() {
  byte numDigits = 1;
  byte digitPins[] = {10,11}; //caution to the wiring

//The common cathodes or anodes should be indicated here.
//They must be connected to the processor.
  byte segmentPins[] = {segment_a,segment_b,segment_c,
segment_d,segment_e,segment_f,segment_g,segment_dd};

  sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins);
  sevseg.setBrightness(90);
}
```

The most important part of this code is the **sevseg.begin** line (that uses every other upper instructions). It provides basic information of the wiring of the component. This is required by the library in order to interact properly with the component.

```
sevseg.begin(
COMMON_CATHODE, // Does the component have a common cathodes
or common_anodes

numDigits,  //Is it a single digit or a 4 digit display

digitPins,  //What pins corresponds to each digit, if only //one indicate only
the common anodes or cathodes pin.

segmentPins //The pins that corresponds to the segment from
//a to digital dot.
);
```

We now require only two more lines to display a given number in the loop function of the sketch.

```
void loop() {

  sevseg.setNumber(8,0);
  sevseg.refreshDisplay();
}
```

sevseg.setNumber(8,0); indicates to display the number 8 that has 0 decimals (integer).

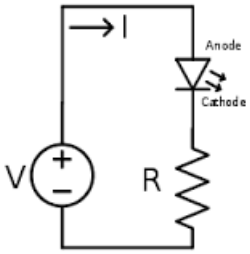
sevseg.refreshDisplay(); is required to send the signal to display.

ToDo: Test the code and check that you can display digits properly.

The following code will allow you to loop via all digits:

```
#include "SevSeg.h"

SevSeg sevseg; //Instantiate a seven segment
float deciSeconds;

void setup() {
  byte numDigits = 1;
  byte digitPins[] = {10,11};//caution to the wiring

  byte segmentPins[] = {segment_a,segment_b,segment_c,
segment_d,segment_e,segment_f,segment_g,segment_dd};

  deciSeconds=0;
  sevseg.begin(COMMON_CATHODE, numDigits, digitPins, segmentPins);
  sevseg.setBrightness(90);
}

void loop() {
  deciSeconds+=0.1;
  sevseg.setNumber(deciSeconds,0);
  if(deciSeconds>9)deciSeconds=0;
  sevseg.refreshDisplay(); // Must run repeatedly
}
```

Calculating the proper resistor to wire with LED



Objective: Set up a current and tension appropriate to the LED.

Example: Current: **20m** and a Tension: **1.5V**

Process:

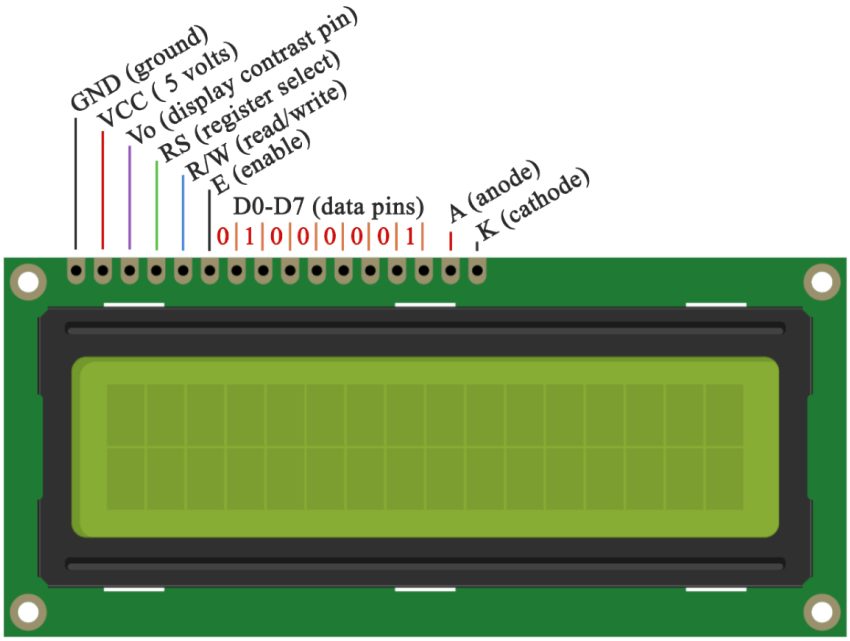
The tension at the resistor must equals: Ur = 5V (power supply) - 1.5V (needed for LED) = 3.5V

To identify the proper resistor, we can use the Ohm's law: U=R*I

Thus, I=U/R=3.5/0.02= **175 Ohms**

3 LCD DISPLAY

In this first part, we will only make sure that the LCD display works properly by wiring the power and ground signals. Connect the pins indicated by a *. By using the LCD datasheet complete the table by indicating the type of signal (what is it used for) for each of the LCD PIN.



LCD PIN	ARDUINO PIN	Type of signal
VSS	GROUND*	
VDD	5V*	
V0	GROUND (via resistors)	
RS	PIN 7	
RW	GROUND	
E	PIN 8	
D0		
D1		
D2		
D3		

D4	PIN 9	
D5	PIN 10	
D6	PIN 11	
D7	PIN 12	
A	5V*	
K	GROUND*	

TODO: Ask teacher validation and plug in the USB cable to test if LCD screen is on.

To adjust the contrast of the display you will plug the V0 pin of the LCD to the ground using in serial two resistors of 1K. You should now observe a first line of the LCD.

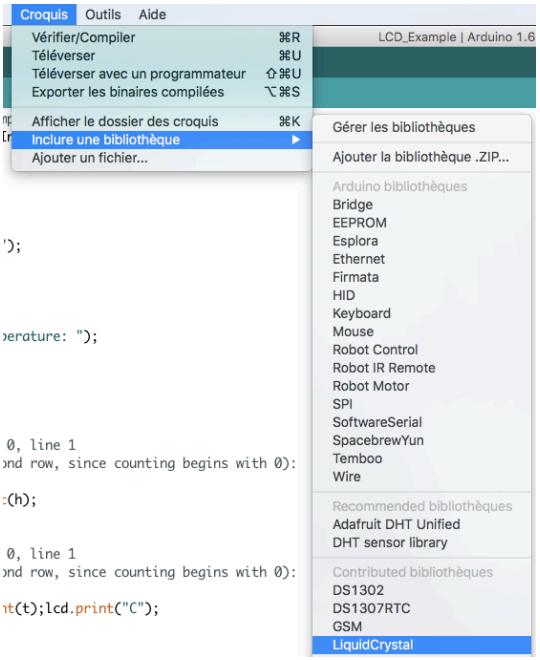
Now plug all data cables to the Arduino as indicated in the table.

3.1 SETTING UP THE LCD CODE TO DISPLAY HELLO WORLD

As you should have read on the LCD datasheet, the communication between the LCD and and the Arduino is not so simple. To avoid writing unnecessary code and allow anyone to use the LCD in a very simple manner, a library already exists.

In this section we will import the LiquidChristal library and use it to display a simple Hello WORLD ! message.

Import the library as indicated in the following picture.




```
#include <LiquidCrystal.h>

LiquidCrystal lcd(7,8,10,11,12); // Indicate the pin number to the lcd library

void setup()
{
     lcd.begin(16,2); // Tell the type of lcd 16 characters on 2 lines
```

```
     lcd.setCursor(0,0); // Character position 0 line 0 (first line first
character)
}

void loop()
{
      lcd.print("Hello world!");
}
```

TODO: Test the code and review the wiring if the LCD does not display the message.