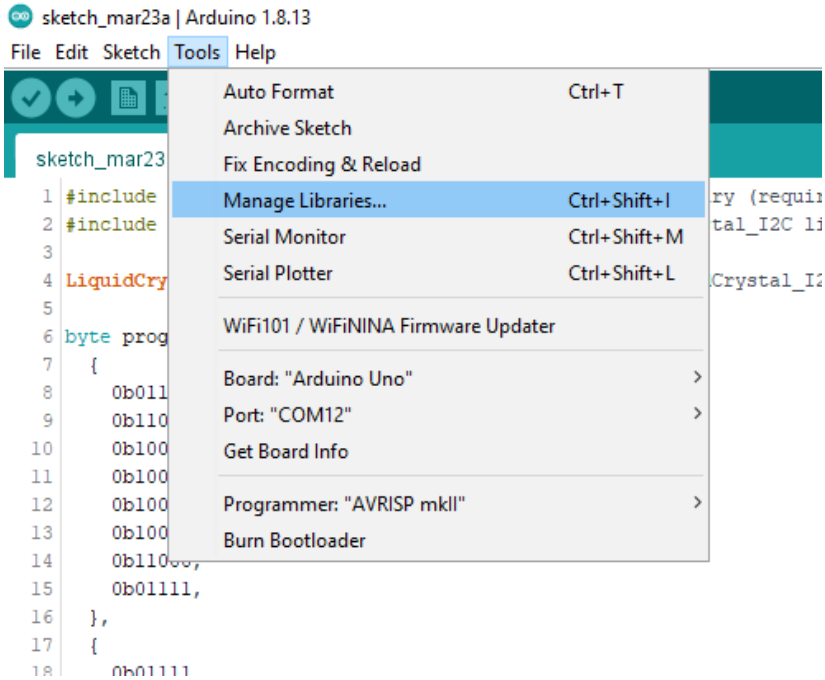
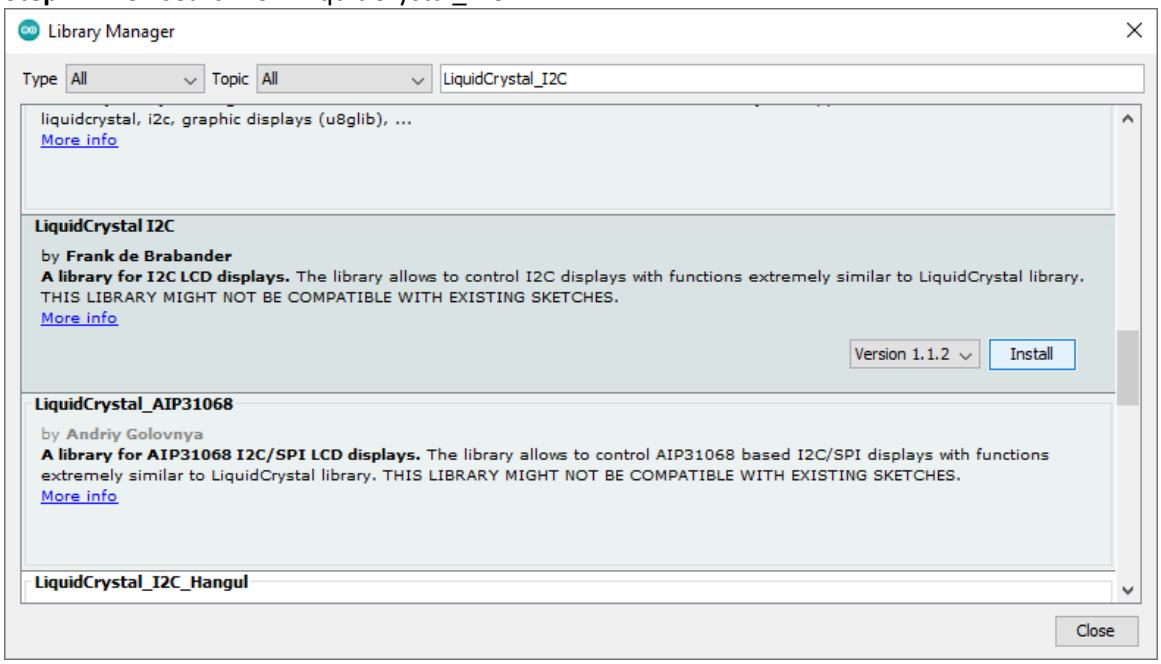
06.01 I2C LCD introduction

The aim of this exercise is to learn how to use the I2C LCD as an output. A useful link with lots of info: https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/
To use this LCD, we first need to install an additional library.

Step 1: Click on Tools > Manage libraries... in the menu.



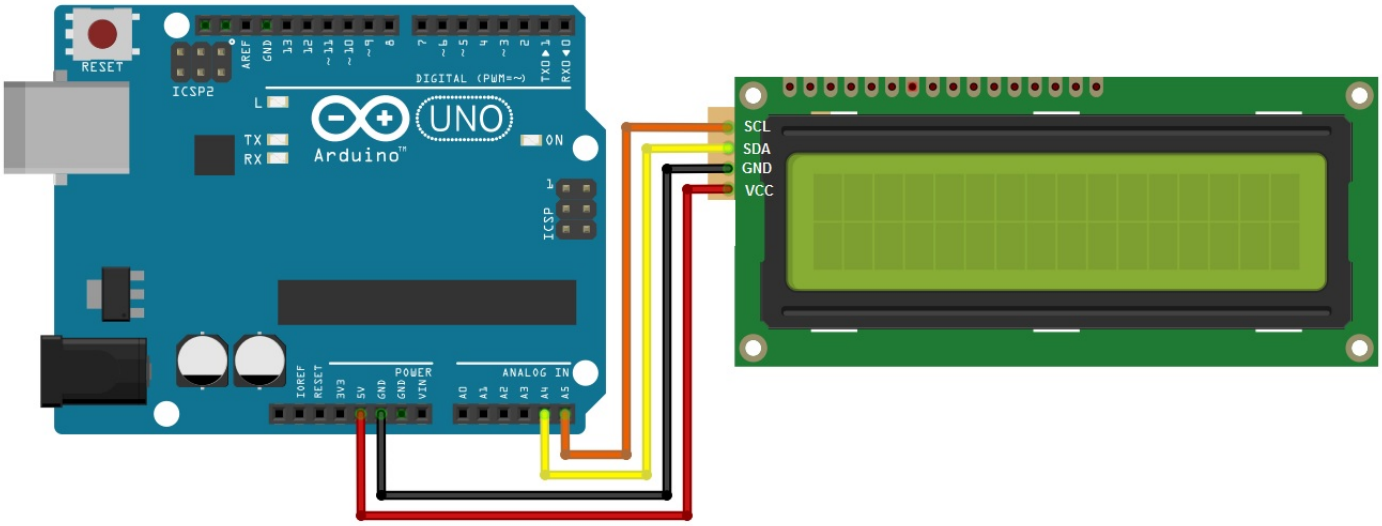
Step 2: Then search for "LiquidCrystal_I2C".



Step 3: Select the **Frank De Brabander version 1.1.2**
Step 4: Click on "Install" and wait until it is installed, then click on "Close".

06.01 exercise

Build the following circuit:

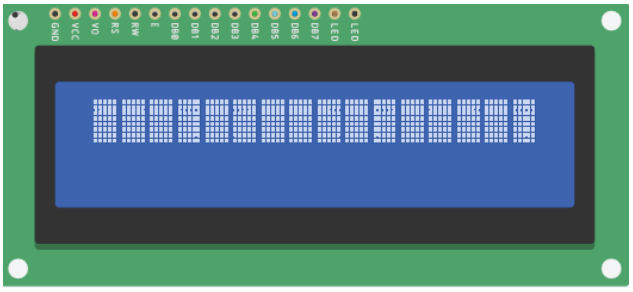


Download the following code from Canvas:

```
#include <LiquidCrystal_I2C.h>.

LiquidCrystal_I2C lcd(0x27,16,2);

void setup() {
 lcd.init(); //initialize LCD
 lcd.clear();
 lcd.backlight(); //enable backlight
 lcd.print("Hello world!");
}

int i = 0;

void loop() {
 lcd.setCursor(0, 1); //going to the beginning of the 2nd line
 lcd.print(i);
 delay(1000);
 i++;
 if(i>=11) {i = 0;}
}
```

Program this code into the controller. Turn the potentiometer until you see text.
If you do not see any text but only 16 blocks on the top line, you should check the connections and whether the correct code is in the controller.

Everything OK!



Best to check...

The code in the controller shows on the top line the text "Hello world!" and on the bottom line a counter from 0 to 10

BUT something is wrong! (This is a common error)

After counting to 10 for the first time, the counter on the display seems to count in steps of 10.
Please analyse and solve the error:

.....

.....

.....

.....

Create the following function yourself:

```
void printNum(int number, int places, char paddingChar)
```

This function must place a number (=first argument) on the LCD.
The second argument determines how many characters (minimum) will be used. Within these characters, the number must be aligned to the right. The third argument is the padding character. This character should be to the left of the number if the number is not large enough to fill all the characters.

Examples:

You call the function like this:	Then this should appear on the screen:
printNum(9, 4,'0');	0009
printNum(99, 4,'0');	0099
printNum(999, 3,'0');	999
printNum(9, 4,' ');	9
printNum(9999, 3,'0');	9999
printNum(123, 4,'#');	#123

Test this function. Think about how you can do this efficiently.

Create a new sketch.

The sketch should display all characters received through the serial monitor on the LCD.

The sketch should also be able to accommodate some of the limitations of the LCD:

The LCD has no automatic line-wrap. You must provide this in your code. In other words, when the line on the LCD is full, it should automatically move to the next line.

If you press enter, you must also move to the next line.

If you are on the 2nd line and the line is full or you press enter, the 2nd line must become the 1st line and you must continue from the beginning of the 2nd line.

Examples (each time from a blank screen, [enter] is a newline character):

You enter the following characters:	Then this should appear on the screen:
0123456789abcdefghijkl	0123546789abcdef ghijkl
Day[enter]World!	Day World!
Day[enter]James[enter]Bond	James Bond

What happens when you send a ~ (and why)?

.....

.....

.....