Introduction to robotics

8th lab

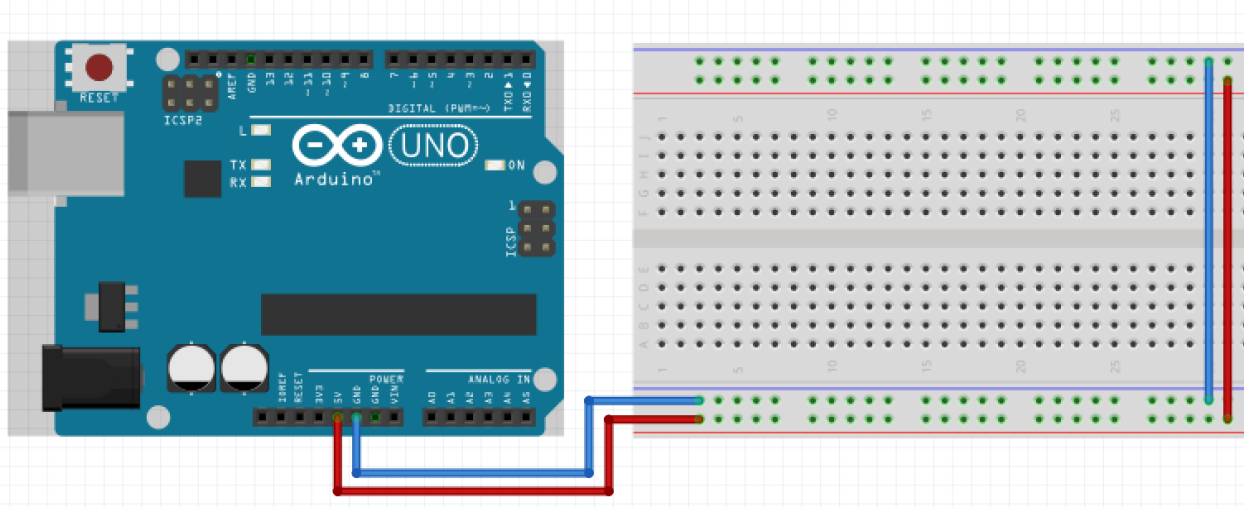
Remember, when possible, choose the wire color accordingly:

- **BLACK** for **GND (dark colors if not available)**
- **RED** for **POWER (3.3V / 5V / VIN) (bright colors if not available)**
- Bright Colored for read and write signal (use **red** when none available and **black** only as a last option)
- We know it is not always possible to respect this due to lack of wires, but the first rule is **DO NOT USE BLACK FOR POWER OR RED FOR GND!**

Now, let's pick it up where we left off…

Pull out your Arduino and breadboard and connect them like in the schematic. This is to "power up" the breadboard so we can easily have access to **5V** and **GND**.

Attention! Remember how the breadboard works. Use correct wire colors.



1. LCD Display

1.1 Introduction

Liquid Crystal Display(LCDs) provide a cost effective way to put a text output unit for a microcontroller. As we have seen in the previous tutorial, LEDs or 7 Segments do not have the flexibility to display informative messages. This display has 2 lines and can display 16 characters on each line. Nonetheless, when it is interfaced with the microcontroller, we can scroll the messages with software to display information which is more than 16 characters in length.

The LCD is a simple device to use but the internal details are complex. Most of the 16x2 LCDs use a **Hitachi HD44780** or a compatible controller. Yes, a microcontroller is present inside a Liquid crystal display as shown in figure 2.

The Display Controller takes commands and data from an external microcontroller and drives the LCD panel **(LCDP)**. It takes an ASCII value as input and generates a pattern for the dot matrix. E.g., to display letter 'B', it takes its value **0X42(hex)** or **66(dec)** decodes it into a dot matrix of 5x7 as shown in figure 1.

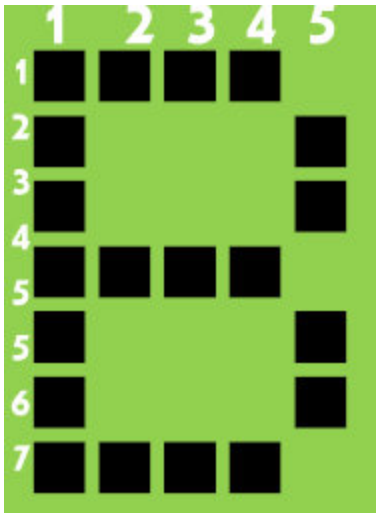


Fig 1: LCD Char 5x7 Matrix
(Each “cell” of the display)

(source:https://exploreembedded.com/wiki/LCD_16_x_2_Basics)

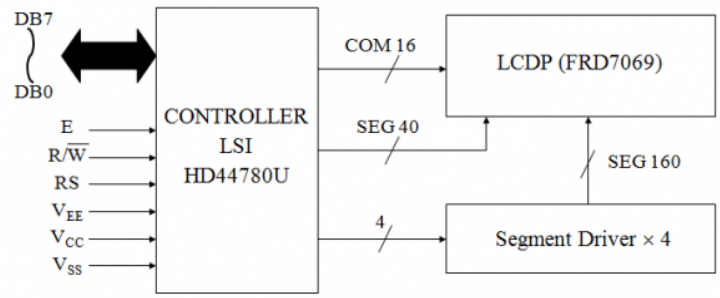
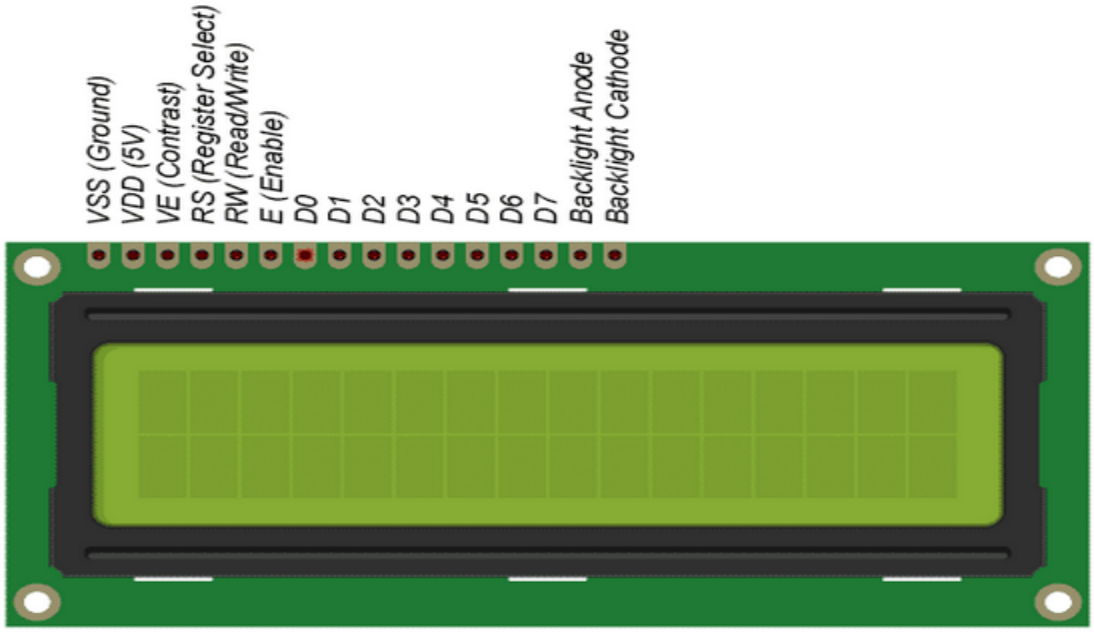
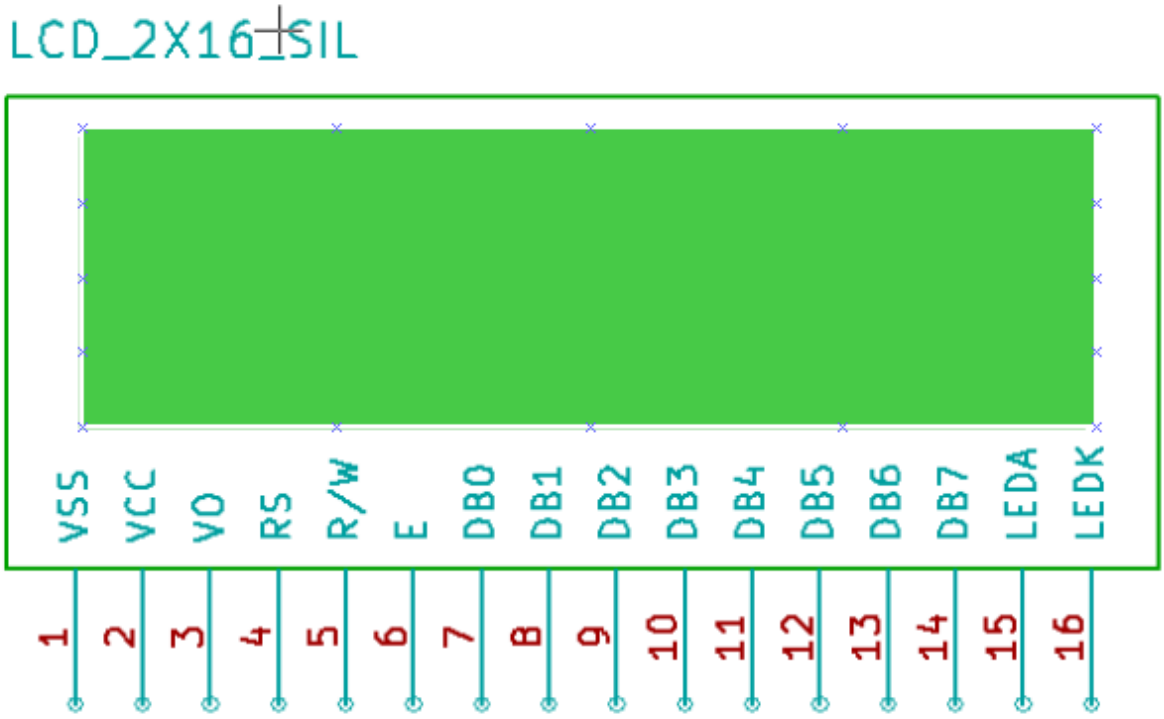


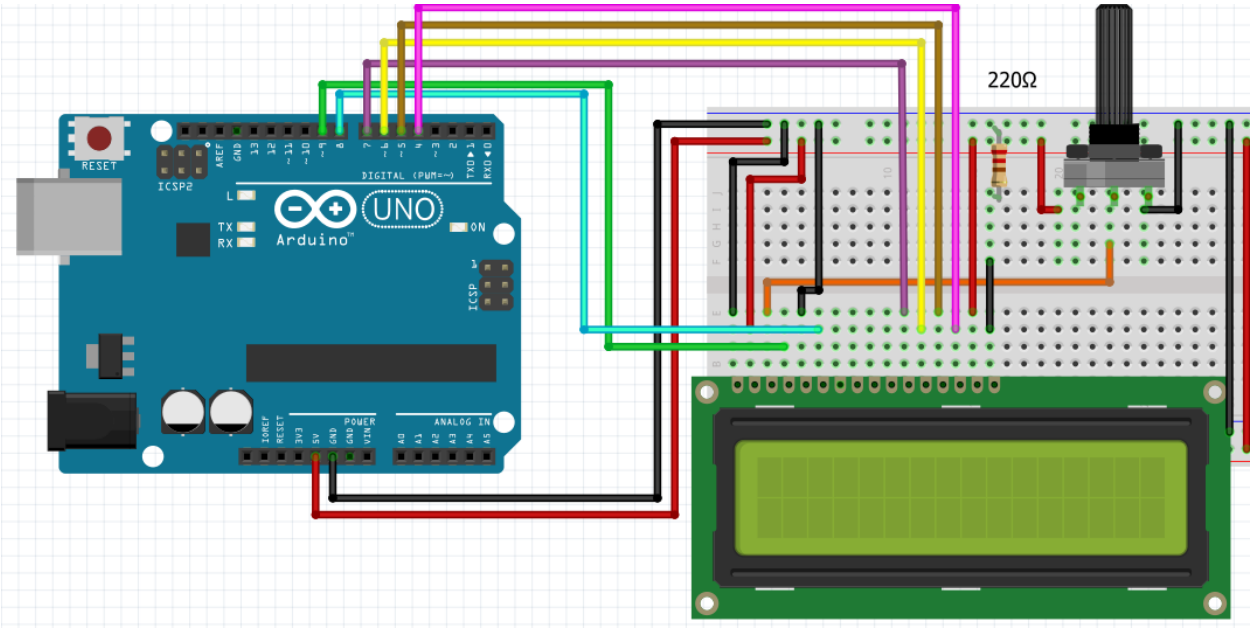
Fig 2: LCD Block diagram





All the pins are identically to the LCD internal controller discussed above

Display PIN	FUNCTION	Arduino PIN
VSS (1)	Ground	GND
VCC (2)	5V	5V
V0 (3)	Contrast adjustment	Potentiometer (analog). PWM after
RS (4)	Register Select. RS=0: Command, RS=1: Data	9
RW (5)	Read/Write (R/W). R/W=0: Write, R/W=1: Read	GND
E (6)	Clock (Enable). Falling edge triggered	8
D0 (7)	Bit 0 (Not used in 4-bit operation)	-
D1 (8)	Bit 1 (Not used in 4-bit operation)	-
D2 (9)	Bit 2 (Not used in 4-bit operation)	-
D3 (10)	Bit 3 (Not used in 4-bit operation)	-
D4 (11)	Bit 4	7
D5 (12)	Bit 5	6
D6 (13)	Bit 6	5
D7 (14)	Bit 7	4
A (15)	Back-light Anode(+)	5V (for direct control to pin 3)
K (16)	Back-Light Cathode(-)	GND (with 220+ ohm resistor)



Default setup. Resistor is about 220ohms, just like with a regular LED

LCD can be interfaced with the microcontroller in two modes, 8 bit and 4 bit. Today we'll interface with the 4bit. The tradeoff is speed, as we send half as much data in one go, but we gain a reduced number of wires.

Data Lines: In this mode, all of the 8 datalines DB0 to DB7 are connected from the microcontroller to an LCD module as shown in the schematic.
Control Lines:' The RS, RW and E are control lines, as discussed earlier.
Power & contrast: Apart from that the LCD should be powered with 5V between PIN 2(VCC) and PIN 1(gnd). PIN 3 is the contrast pin and is the output of the center terminal of potentiometer(voltage divider) which varies voltage between 0 to 5v to vary the contrast.
Back-light: The PIN 15 and 16 are used as backlights. The led backlight can be powered through a simple current limiting resistor as we do with normal leds.

1.2 Code Examples

Most of them are from https://docs.arduino.cc/learn/electronics/lcd-displays (be mindful of different wiring)

1.2.1 Hello world!

```
#include <LiquidCrystal.h>
const byte rs = 9;
const byte en = 8;
const byte d4 = 7;
const byte d5 = 6;
const byte d6 = 5;
const byte d7 = 4;

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

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);
}
```

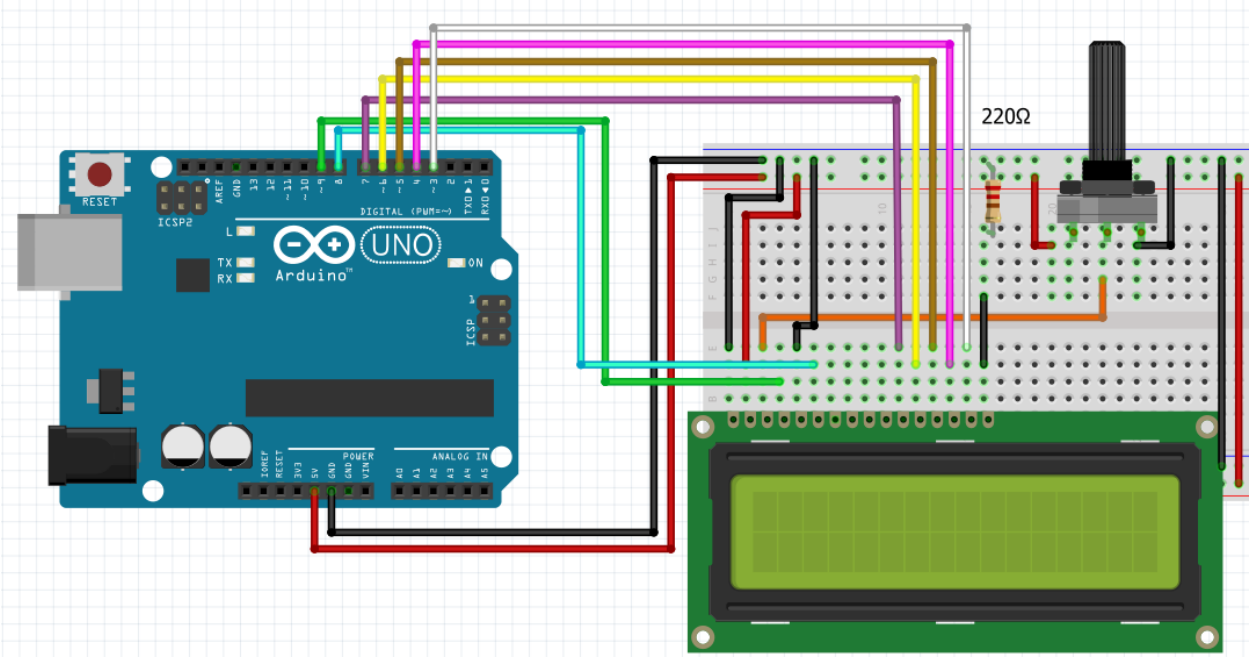
1.2.1 Try all the LCD functions in the guide

1. Go through all the functions in this guide: https://docs.arduino.cc/learn/electronics/lcd-displays
2. For each one, think of how you can use it in your homework
3. What could custom characters be used for?

2. Lab exercises

2.1 Controlling the LCD intensity

1. Connect the LCD Anode to Pin 3 and control it using analogWrite.
2. Read data from serial and use it to change the intensity of the LED
3. Save the value to eeprom



Setup with LCD LED connected to Arduino PWM pin

2.2 Create a custom character of your choosing and scroll it around

1. Create a custom character of choice
2. Scroll it on the display, playing with the settings

Resources:

1. https://docs.arduino.cc/learn/electronics/lcd-displays
2. https://docs.arduino.cc/learn/programming/eeprom-guide
3. https://www.arduino.cc/reference/en/language/functions/communication/serial/
4. https://docs.arduino.cc/learn/programming/memory-guide
5. https://omerk.github.io/lcdchargen/
6. https://electronoobs.com/eng_arduino_tut167.php
7. https://www.baldengineer.com/arduino-f-macro.html