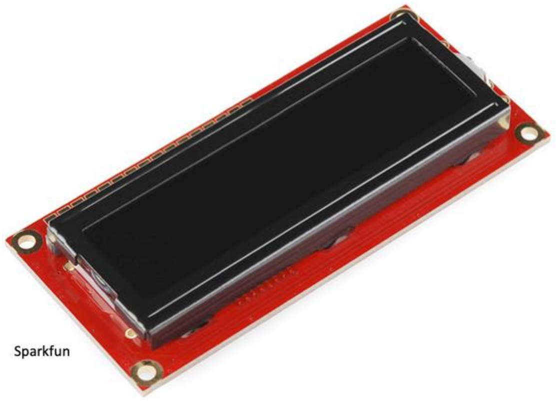
ECE 1001/1002 Introduction to Robotics
Lab #4: Display

Objectives

Introduction to the multiline display

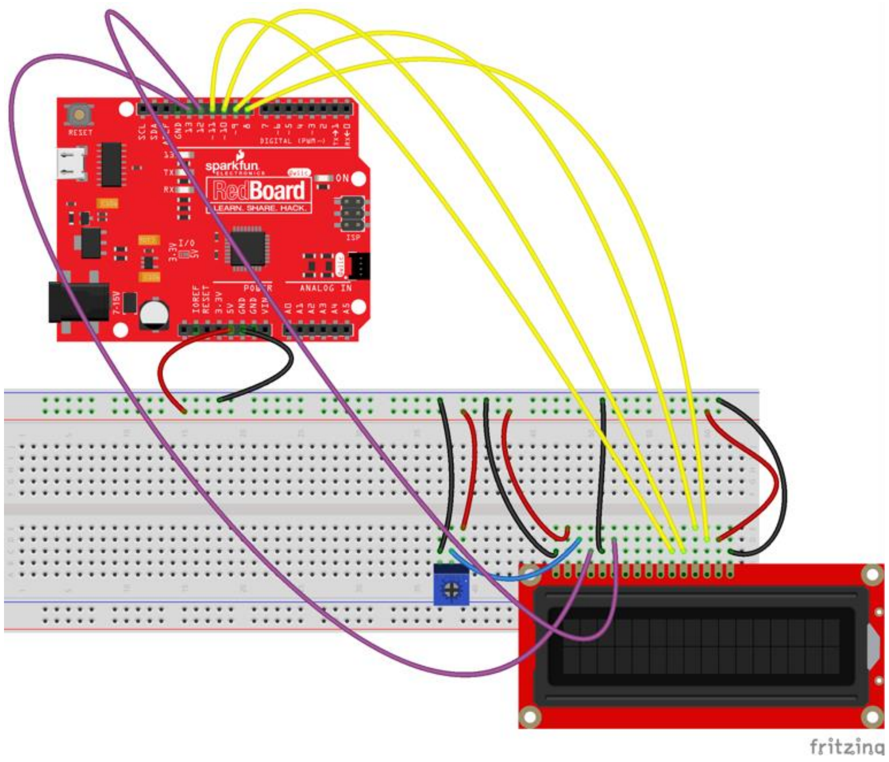
Requirements and Signoffs

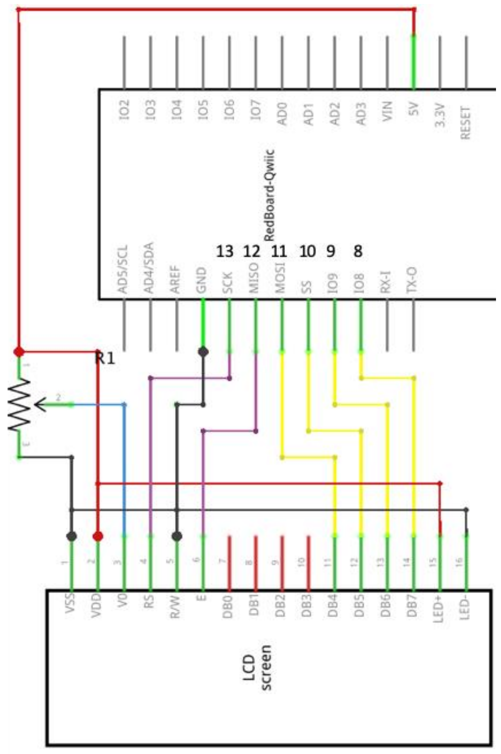
LEDs and Speakers off a way for the Arduino to make limited communications to you. Your kit also comes with a multiline display, which is far more flexible and useful.



Display Circuit

The multiline display has several inputs, and also takes up quite a bit of space. Build the circuit below.





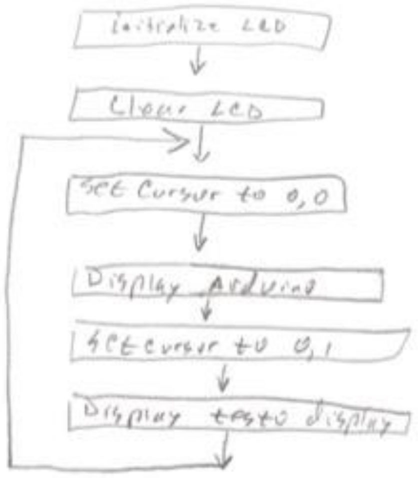
This is more complicated than previous circuits, and harder to see. Note in circuit schematics where wires cross they are connected if there is a circle at the intersection. You may need to enlarge the schematic to see that. The table below may be easier to use, it shows which wires to connect between the LCD display and the Arduino. Count pin numbers on the LCD from the upper right (opposite the metal tab connected to the display) towards the bottom

Arduino pin	Wire color suggestion	Display pin	display pin description
GND	black	1	GND
5V	red	2	5V
Connect to Pot Middle pin	blue	3	Contrast adj
13	purple	4	Register select
GND	black	5	R/W select
12	purple	6	Enable
11	yellow	11	Data d4
10	yellow	12	Data d5
9	yellow	13	Data d6
8	yellow	14	Data d7
5V	red	15	Backlight 5V
GND	black	16	Backlight GND

Double check your wiring a few times!

Simple text display

Open the lab4 simple text program, Lab4_simple_text_display.ino, in the Arduino IDE and look at it.



Upload it to your Arduino and have a look. If you can't see anything on your display, adjust the pot all the way in both directions to see if there is an optimal condition. If you still don't see anything, time to trouble-shoot!

Look some more at the program, near the top, there is an "include" statement:

```
#include <LiquidCrystal.h>
```

"include" makes a set of pre-written code for using displays available for use by your program. There are many pre-written libraries for doing various useful things, they can make your life easier, since there is no need to re-write code that somebody else has figured out. They can also make your life miserable, as sometimes there are mistakes, bugs, or different intended uses for the code in a library. The LCD display in your kit uses a very common chip for control, and this LiquidCrystal library works with many different displays (all using the same chip for control).

By including the library, you have expanded the commands at your disposal to do things (the keyword list is also expanded). These are the LCD commands used in this program; you can find more commands in the Arduino reference, but these are enough to do most things.

LiquidCrystal lcd(13, 12, 11, 10, 9, 8); defines which of the LCD pins are connected to which Arduino pins.

lcd.begin(16, 2); starts the Arduino communicating with the LCD, and sets the mode to 16 characters wide and 2 lines high

lcd.clear(); erases everything from the LCD screen

lcd.setCursor(0, 0); sets the cursor to the (x,y) location given. 0,0 is in the upper left, and 15,1 is in the lower right.

lcd.print("Arduino"); will diplay on the screen whatever is between the double quote marks

Note the use of the "." in all of these commands, such as "lcd.print" That is a sign the library is written in C^{++} which Arduino actually uses; C^{++} is an extension of the C language implementing modern, powerful programming techniques. This class won't go into much depth about this subject, we can do everything we need with just C, bringing a little C^{++} in here and there.

Now try and break (and fix) the display program, Lab4_simple_text_display.ino ! Write down these results on your sign-off sheet

Question 1) does it display as you expect? What's wrong?

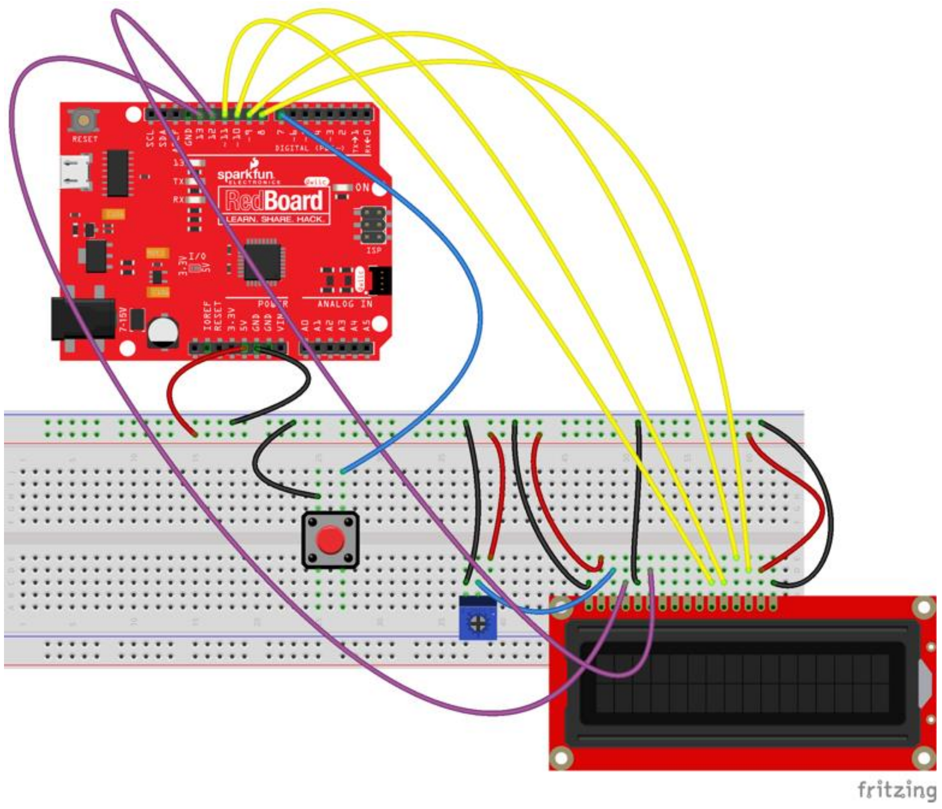
Question 2) Try entering text more than 16 characters long, write down what happens?

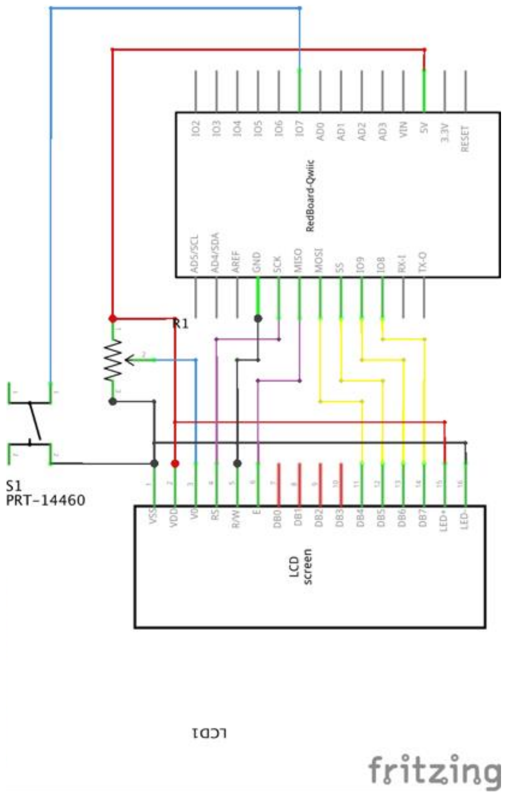
Question 3) Try displaying with a lcd.clear(); statement in the loop, does it have any effect? Why?

Question 4) Write your first name on the upper row, and last name on the lower row, describe what happens

The Lap-Watch Hardware

Your Arduino can display various useful things, and it has a built-in timer. One interesting thing to do is build a one-button stopwatch. Add a button to your circuit like this, it just takes a couple more wires

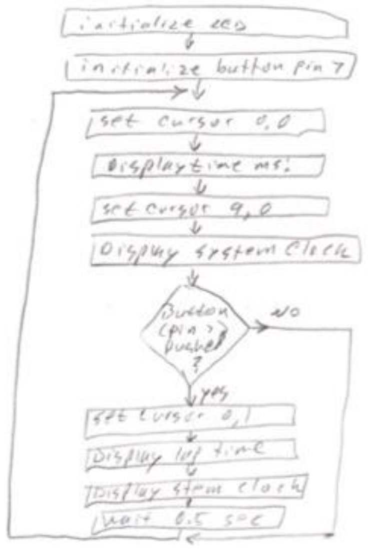




Actually, with the programming knowledge so far, we can easily implement a lap timer. This system will start with a reset (either software upload, power on, or a push of the little flat reset button on the corner of the Arduino). The number of milliseconds since the reset will be displayed on the top line. If you push the button (a red one in the image above) the timer will show the time the button was pressed since the reset, in the bottom row, and resume displaying the time after a half-second pause (the timer will not pause, just the display).

The Lap-Watch Software

Open the program, Lab4_stopwatch.ino, and have a look. A new line is needed in setup to use the switch connected to pin 7 with internal pullup resistors, and then some timing software. But not much software is needed!



The Arduino has a timer which counts the number of milliseconds since the Arduino was reset (or powered up, or software was uploaded). It can keep counting up for forty days or so before the count reaches a memory limit and will reset. So just display the millisecond count with millis() Here we want to display the value returned by millis() so we don't enclose it in double quotes:

lcd.print(millis());

The Arduino will loop continuously, displaying the time as it goes. If the red button is pressed, though, it will print the time of the press on the bottom line, and then wait for a half-second before continuing to display the time on the upper line. It's necessary to wait some time after a button press because mechanical switches like the button all "bounce" or "chatter" when pushed—the switch rapidly closes and opens for some time, and the electrical signal will bounce between High and Low. It shouldn't bounce for so long as 500 ms though, that's a bit excessive probably.

Using Lab4_stopwatch.ino answer the following questions.

Question 5) Try and break it. How short can you make the delay before it becomes unreliable?

Question 6) Try pushing the button quickly, what happens?

Question 7) Push reset quickly, what happens?

Question 8) What is the shortest time you can get to display on the lap? What if you adjust the delays in the program?

Show your results for these and the questions above (8 questions total) to your instructor, and demonstrate your Lap Timer for signoff

To receive credit for this lab, you *must* get the sign-off before the due date (see Canvas), during class time or with a TA during their hours. The instructor will not signoff outside of class hours. Partial credit is allowed.