adafruit learning system

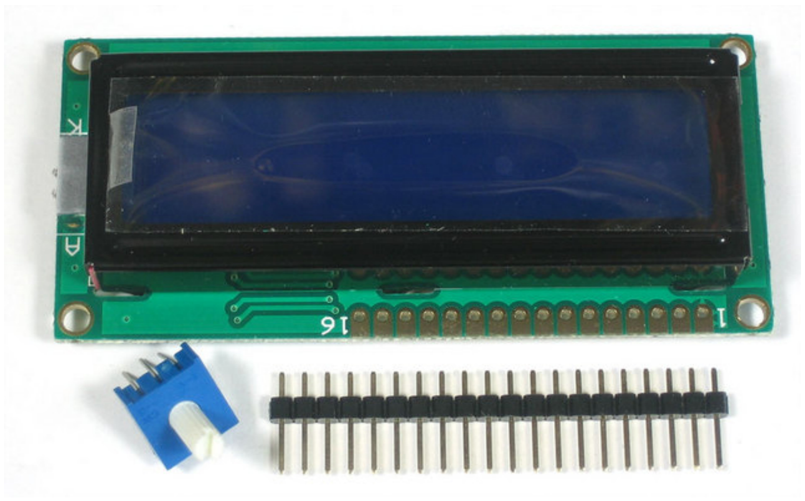
Character LCDs

Created by Ladyada

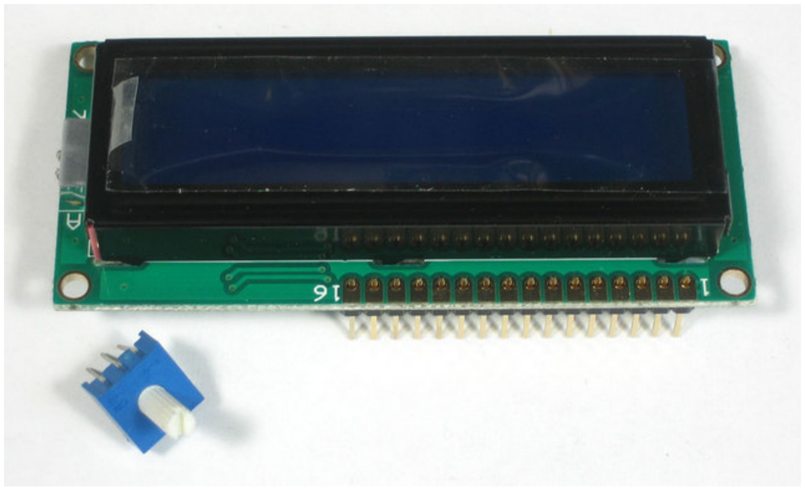
Last updated on 2013-07-26 02:45:29 PM EDT

Wiring a Character LCD

Installing the Header Pins



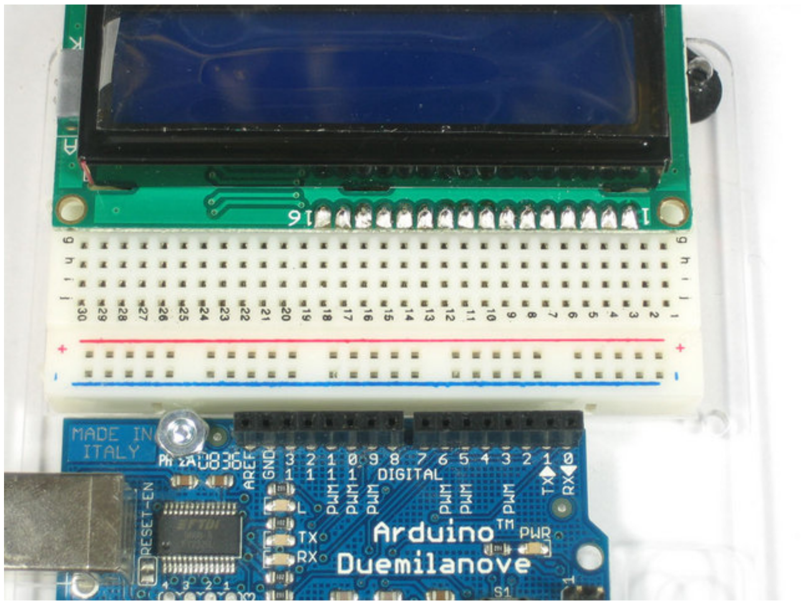
OK now you've got your LCD, you'll also need a couple other things. First is a 10K potentiometer. This will let you adjust the contrast. Each LCD will have slightly different contrast settings so you should try to get some sort of trimmer. You'll also need some 0.1" header - 16 pins long.



If the header is too long, just cut/snap it short!

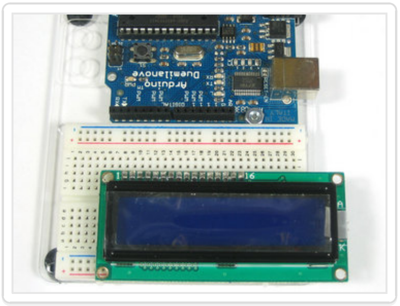
Next you'll need to solder the header to the LCD.**You must do this, it is not OK to just try to 'press fit' the LCD!**

Also watch out not to apply too much heat, or you may melt the underlying breadboard. You can try 'tacking' pin 1 and pin 16 and then removing from the breadboard to finish the remaining solder points

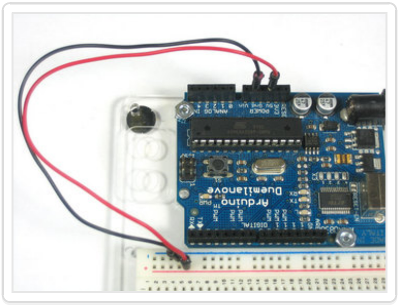


The easiest way we know of doing this is sticking the header into a breadboard and then sitting the LCD on top while soldering. this keeps it steady.

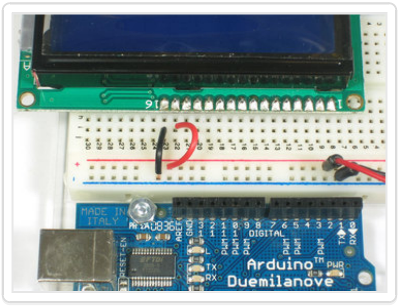
Power and Backlight



K now we're onto the interesting stuff! Get your LCD plugged into the breadboard.

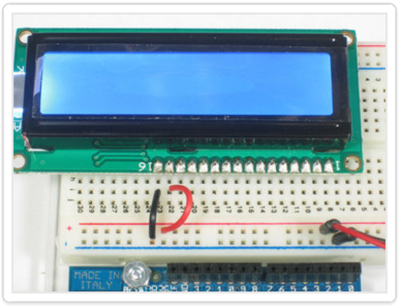


Now we'll provide power to the breadboard. Connect +5V to the red rail, and Ground to the blue rail.



Next we'll connect up the backlight for the LCD. Connect pin 16 to ground and pin 15 to +5V. On the vast majority of LCDs (including ones from Adafruit) the LCD includes a series resistor for the LED backlight.

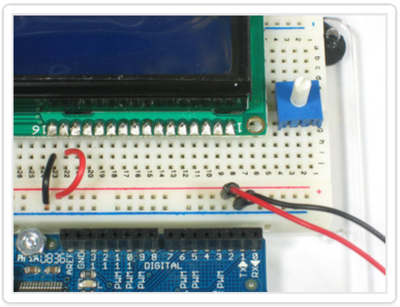
If you happen to have one that does not include a resistor, you'll need to add one between 5V and pin 15. To calculate the value of the series resistor, look up the maximum backlight current and the typical backlight voltage drop from the data sheet. Subtract the voltage drop from 5 volts, then divide by the maximum current, then round up to the next standard resistor value. For example, if the backlight voltage drop is 3.5v typical and the rated current is 16mA, then the resistor should be (5 - 3.5)/0.016 = 93.75 ohms, or 100 ohms when rounded up to a standard value. If you can't find the data sheet, then it should be safe to use a 220 ohm resistor, although a value this high may make the backlight rather dim.



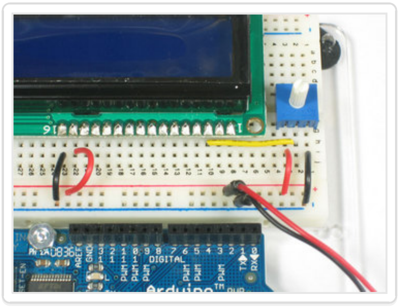
Connect the Arduino up to power, you'll notice the backlight lights up.

Note that some low-cost LCDs dont come with a backlight. Obviously in this case you should just keep going.

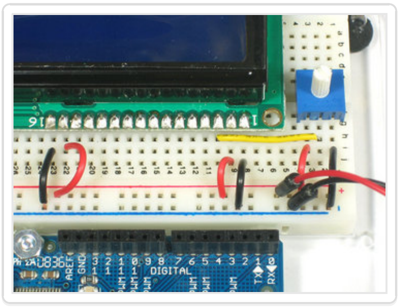
Contrast Circuit



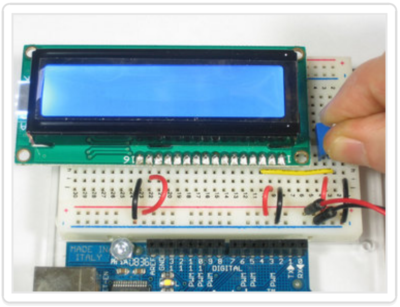
Next, lets place the contrast pot, it goes on the side near pin 1.

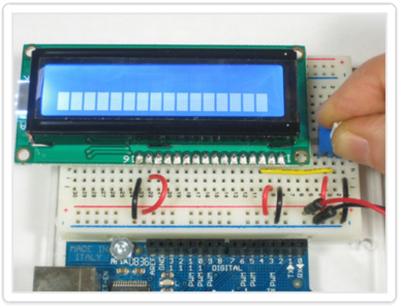


Connect one side of the pot to +5V and the other to Ground (it doesn't matter which goes on what side). The middle of the pot (wiper) connects to pin 3 of the LCD.



Now we'll wire up the logic of the LCD - this is seperate from the backlight! Pin 1 is ground and pin 2 is +5V.





Now turn on the Arduino, you'll see the backlight light up (if there is one), and you can also twist the pot to see the first line of rectangles appear.

This means you've got the logic, backlight and contrast all worked out. Don't keep going unless you've got this figured out!

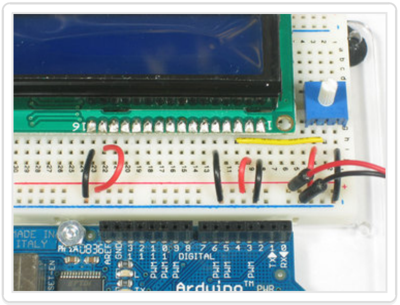
Bus Wiring

Now we'll finish up the wiring by connecting the data lines. There are 11 bus lines: **D0** through **D7** (8 data lines) and **RS**, **EN**, and **RW**. D0-D7 are the pins that have the raw data we send to the display. The**RS** pin lets the microcontroller tell the LCD whether it wants to display that data (as in, an ASCII character) or whether it is a command byte (like, change posistion of the cursor). The **EN** pin is the 'enable' line we use this to tell the LCD when data is ready for reading. The **RW** pin is used to set the direction - whether we want to write to the display (common) or read from it (less common)

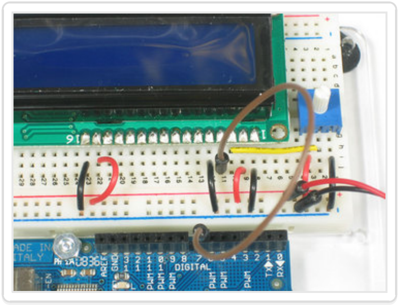
The good news is that not all these pins are necessary for us to connect to the microcontroller (Arduino). **RW** for example, is not needed if we're only writing to the display (which is the most common thing to do anyways) so we can 'tie' it to ground. There is also a way to talk to the LCD using only 4 data pins instead of 8. This saves us 4 pins! Why would you ever want to use 8 when you could use 4? We're not 100% sure but we think that in some cases its faster to use 8 - it takes twice as long to use 4 - and that speed is important. For us, the speed isn't so important so we'll save some pins!

So to recap, we need 6 pins: **RS, EN, D7, D6, D5,** and **D4** to talk to the LCD.

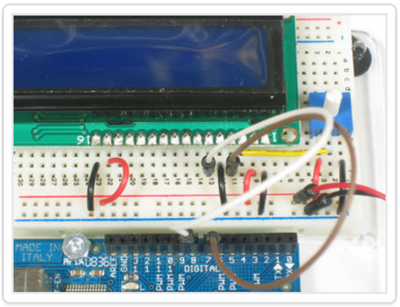
We'll be using the **LiquidCrystal** library to talk to the LCD so a lot of the annoying work of setting pins and such is taken care of. Another nice thing about this library is that you can use **any** Arduino pin to connect to the LCD pins. So after you go through this guide, you'll find it easy to swap around the pins if necessary



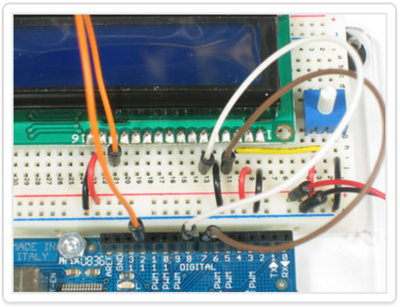
As mentioned, we'll not be using the **RW** pin, so we can tie it go ground. That's pin 5 as shown here.



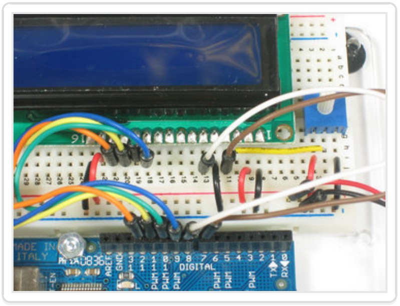
Next is the **RS** pin #4. We'll use a brown wire to connect it to Arduino's digital pin **#7.**



Next is the **EN** pin #6, we'll use a white wire to connect it to Arduino digital #**8.**

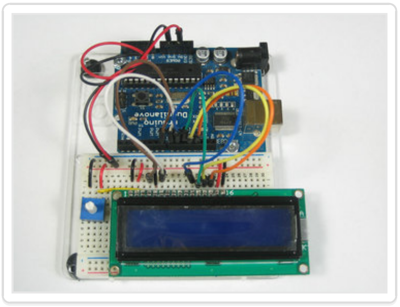


Now we will wire up the data pins. **DB7** is pin #14 on the LCD, and it connects with an orange wire to Arduino #**12.**



Next are the remaining 3 data lines, **DB6** (pin #13 yellow) **DB5** (pin #12 green) and **DB4** (pin #11 blue) which we connect to Arduino #**11, 10** and **9.**

You should have four 'gap' pins on the LCD between the 4 data bus wires and the control wires.



This is what you'll have on your desk.

Using a Character LCD

Now we must upload some sketch to the Arduino to talk to the LCD. Luckily the **LiquidCrystal** library is already built in. So we just need to load one of the examples and modify it for the pins we used.

If you've changed the pins, you'll want to make a handy table so you can update the sketch properly.

LCD pin name	RS	EN	DB4	DB5	DB6	DB7
Arduino pin #	7	8	9	10	11	12

Open up the **File→Examples→LiquidCrystal→HelloWorld** example sketch

Now we'll need to update the pins. Look for this line:

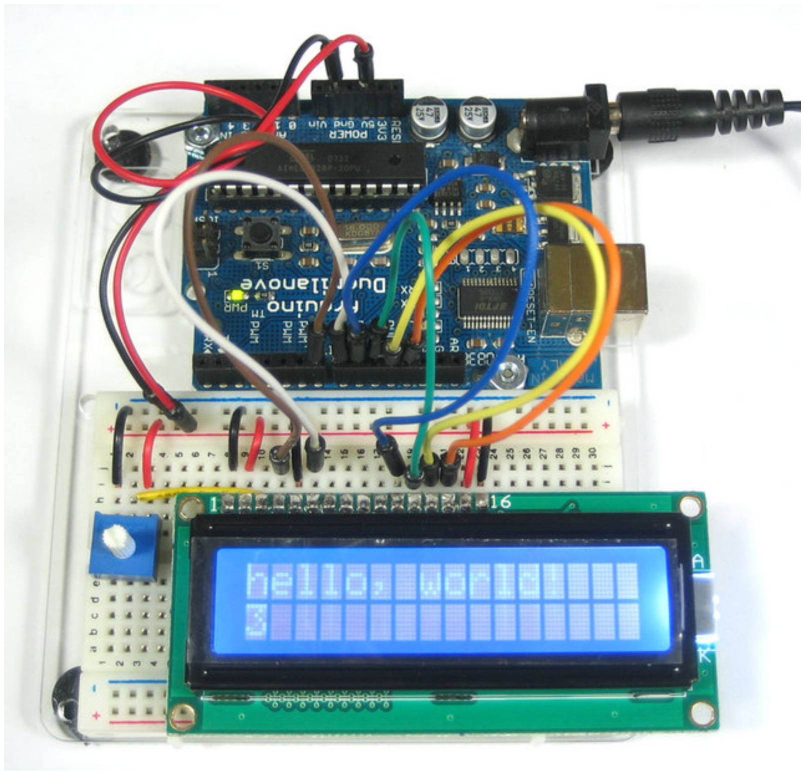
```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

And change it to:

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

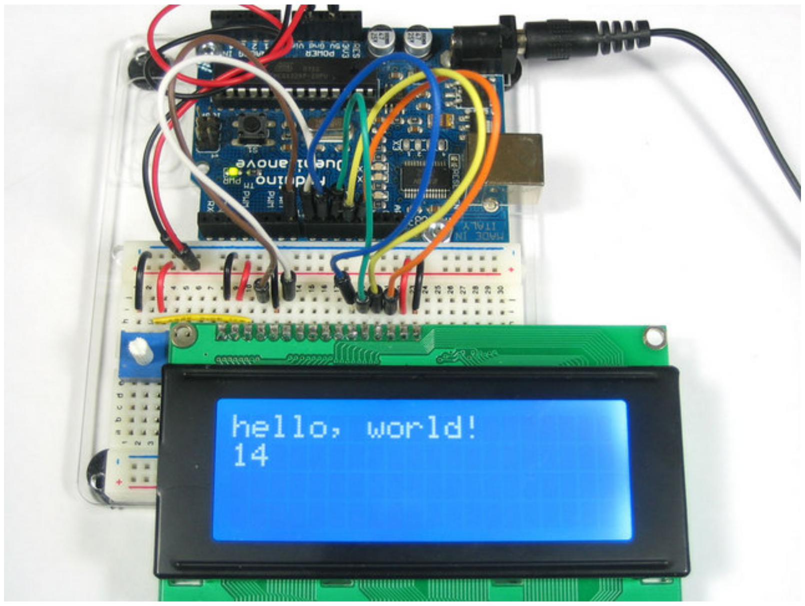
To match the pin table we just made.

Now you can compile and upload the sketch.

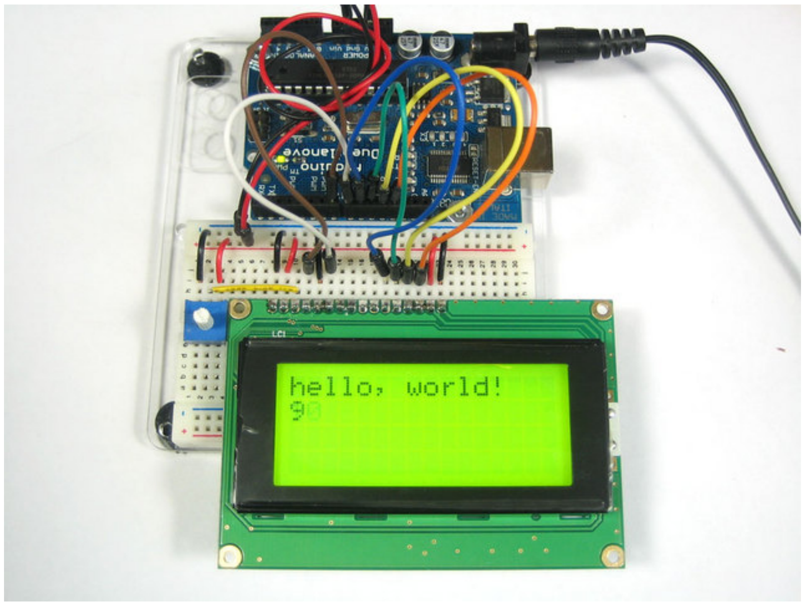


Adjust the contrast if necessary

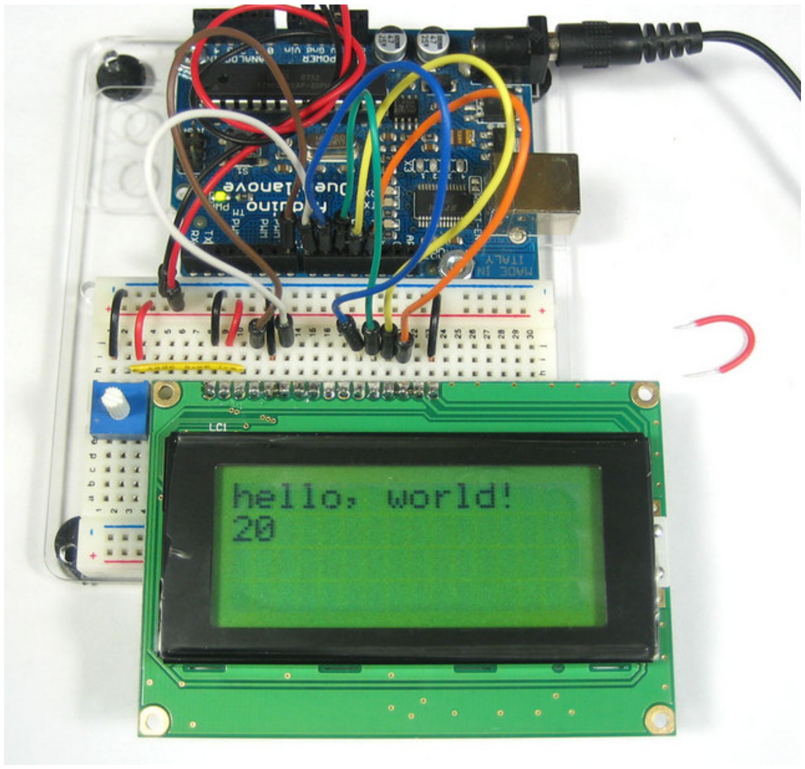
You can of course use any size or color LCD, such as a 20x4 LCD



Or a black on green



The nice thing about the black on green ones is you can remove the backlight. Sometimes they dont come with one!



Multiple Lines

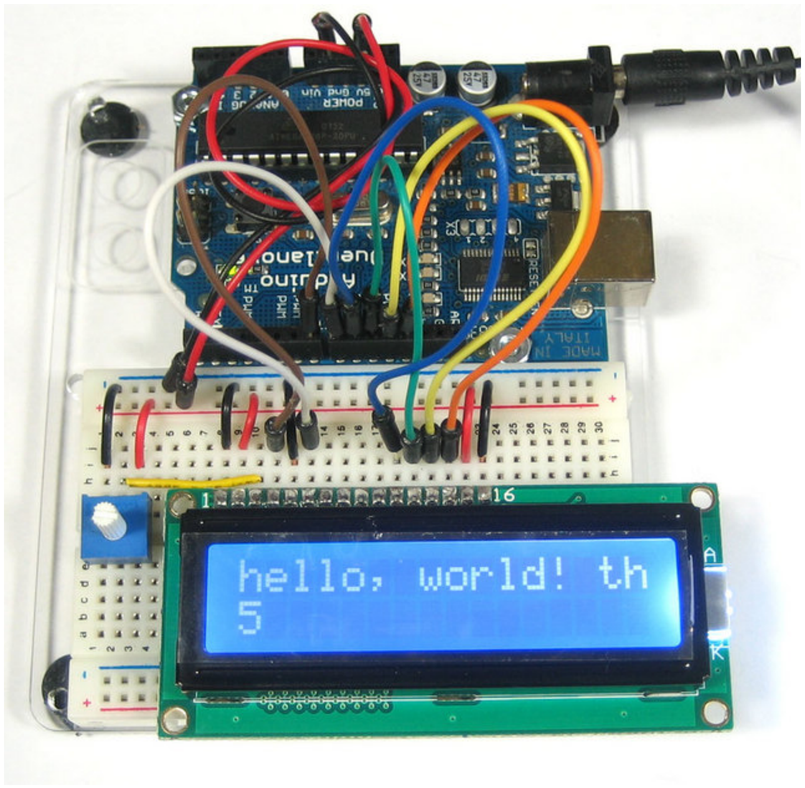
One thing you'll want to watch for is how the LCD handles large messages and multiple lines. For example if you changed this line:

```
lcd.print("hello, world!");
```

To this:

```
lcd.print("hello, world! this is a long long message");
```

The 16x2 LCD will cut off anything past the 16th character:



But the 20x4 LCD will 'wrap' the first line to the **third** line! (Likewise the 2nd line runs into the 4th) This seems really bizarre but its how the LCD memory configured on the inside. This probably should have been done differently but hey that's what we have to live with. Hopefully we'll have a future LCD library that is very smart and wraps lines but for now we are stuck. So when writing long lines to the LCD count your characters and make sure that you dont accidentally overrun the lines!