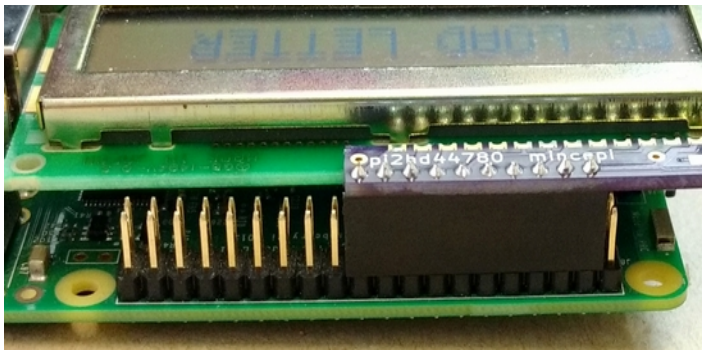
Seattle Raspberry Jam
A Raspberry Pi Meetup for Beginners to Experts
RASPBERRY JAM

Hardware Project #6:
16x1 LCD Character Display

Assembly



We will be using a 16x1 LCD character display for this project. These can be found on Aliexpress for just a couple of dollars (for example, bit.ly/2K8Hjr8). To connect it to the Raspberry Pi, we have designed a circuit board. Order it from OSH Park at LINK or download the design files from (bit.ly/2Ynf11a). The last part that we need is a 10-pin socket, which can also easily be found on Aliexpress (for example, bit.ly/2MpqQ4r). Solder it together as shown in the picture to the right. Finally, plug the pin socket into the Raspberry Pi's pin header on pins 10 to 28 as shown in the picture to the right. **WARNING! Incorrectly installing the display could damage the Raspberry Pi. Please have one of the organizers check that it is plugged in correctly before applying power.**

Preliminary Software Setup

While there are many ways of interfacing these displays with Raspberry Pis, we decided the simplest would be to use a kernel driver. To do this, we need to make some changes to */boot/config.txt*. Open config.txt in a text editor with this command:

Continued overleaf

How do 16x1 displays work?

The ubiquitous Liquid-Crystal-Display (LCD) is used in everything from microwave readouts to older monitors and TVs, but how do these light-emitting black boxes work? The simplest monochrome LCD is made up of four layers: a backlight, a vertical polarizing filter, a liquid crystal layer, and a horizontal polarizing filter. When the backlight is turned on, it emits incoherent light, or light that travels in many angles/directions. This light is then passed through the vertical filter, which only lets through the "vertical" light. Next, the light enters the liquid crystal, where it is either rotated 90 degrees – to horizontal – or left the same – vertical – depending on whether voltage is present. This special property of the liquid crystal is what makes these displays possible, and also explains the name. Finally, the light from the liquid crystal passes through a horizontal filter, which does the same as the vertical filter except that it only lets horizontal light through. This means that when voltage is applied to the crystal, light is let through, and when voltage is not present, no light is let through. All that is left to do is to control where the voltage is on the crystal, and this is done by a controller chip such as the HD44780. For more information on the display used in this project, see the HD44780 datasheet at bit.ly/2K7Y061 and the Wikipedia entry at bit.ly/2Ys76Qa.

```
sudo nano /boot/config.txt
```

And add the following lines to the bottom of the file:

```
dtoverlay=hd44780-lcd,pin_d4=18,pin_d5=23,pin_d6=24,pin_d7=25
dtparam=pin_rs=14,pin_en=15,display_width=8,display_height=2
```

The first line initializes the *hd44780-lcd* kernel driver to use GPIO pins 18, 23, 24 and 25 for the data lines. The second line sets GPIO 14 as the register select pin, GPIO 15 as the enable (start data read/write) pin, the display width to eight characters and the height to two lines. You might notice that these last two parameters seem strange; we are using a 16 characters 1 line display, and yet we are setting the driver to use an 8 character 2 line display. This quirk comes from the design of the display, where the manufacturers figured that they could use only one HD44780 chip, which is designed for 8x2 displays, to cut costs, instead of both an HD44780 chip and an expansion chip if they wanted 16x1s. Because of this, the first eight characters are written as if they are on the first line, and the next eight are written as if they are on the second line. Save and close the file with CTRL-X, then type Y and finally press Enter. Now run the following command to reboot the Raspberry Pi:

```
sudo reboot
```

Once the Raspberry Pi is back up and running, continue on to the next section.

Some Display Operations...

We can now display some text! First, we need root priviledges to run the display commands. Do this with the following command:

```
sudo su
```

Now run this command:

```
echo -ne "\e[2JPC LOAD\nLETTER" > /dev/lcd
```

“PC LOAD LETTER” should appear on the screen! (did you get the HP printer reference?). We are using the `echo` command to print formatted text. The flags “-n” and “-e” make `echo` remove the trailing newline character and take escape sequences. Next, note the “\e[2J” part in the text printed to the display; this is one of the aforementioned escape sequences; it clears the display and sets the cursor back to the first character. For further reference, another useful escape sequence is” \e[H,” which also sets the cursor to the first character but does not clear the display. The next part of interest is the “\n.” This is a newline character, and sets the display to the “first character on the second line” (a.k.a. the eighth character). Finally, there is a redirect operator, >, which directs the output of the `echo` command to the file device */dev/lcd*. Now that we know how to operate the display, it is time to do something useful with it!

Printing your IP address on boot

We figured that a good example would be to show the Raspberry Pi’s IP address on the screen while booting. To do this, open */etc/rc.local*, a script that automatically runs on boot, in a text editor:

```
sudo nano /etc/rc.local
```

Add the following line after the “ip=$(hostname -I)” line.

```
sudo echo -ne "\e[2J"${ip:0:8}"\n"${ip:8:7} > /dev/lcd
```

Now each time you start up the Raspberry Pi, the IP address will be printed to the screen, which is especially useful if you are running the Pi headless.

Challenge: Create a shell script that runs when the Raspberry Pi shuts down to display a message on the LCD display. For an easier challenge, change */etc/rc.local* to print “Finished Booting,” wait a few seconds, and then print the IP address. Hint: the command for waiting is `sleep SECONDS`