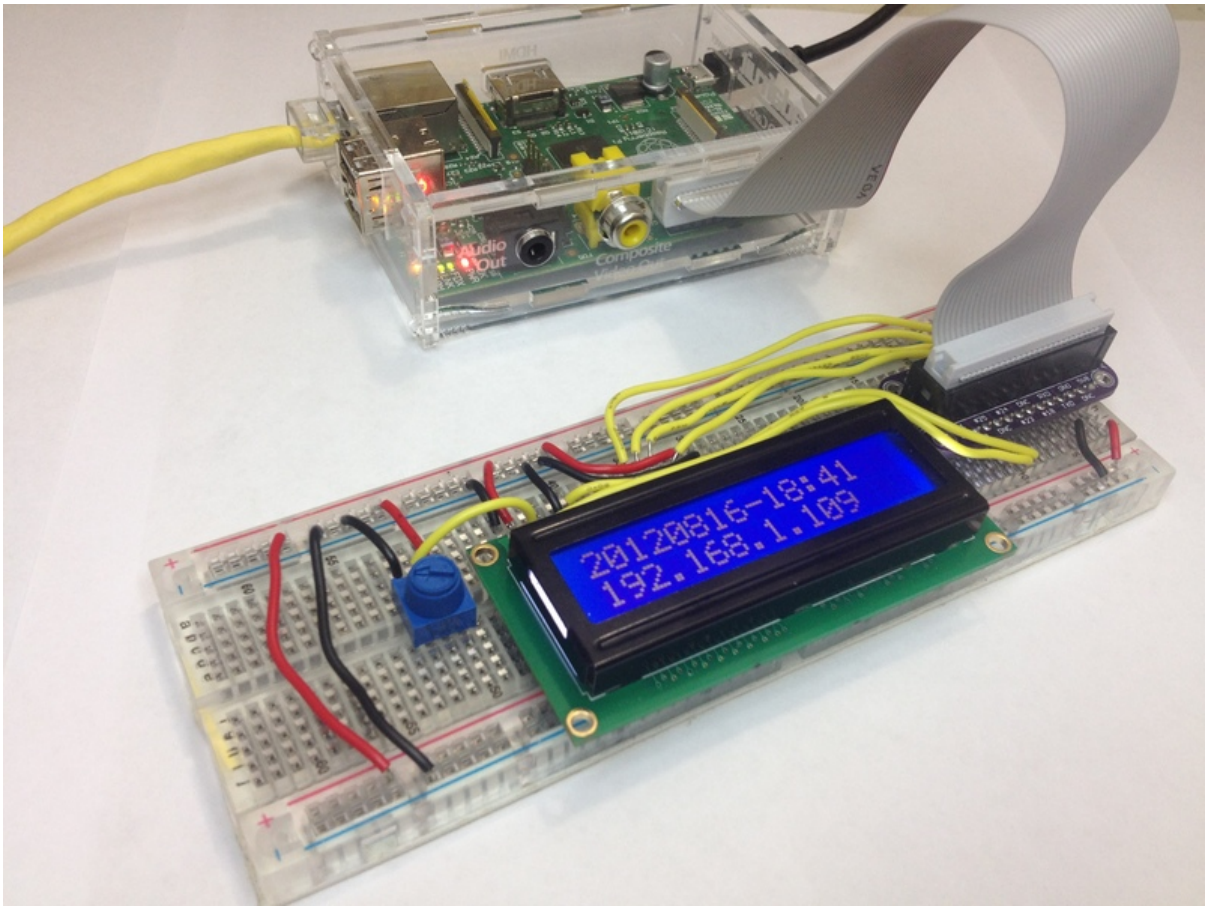
Drive a 16x2 LCD with the Raspberry Pi

Created by Michael Sklar

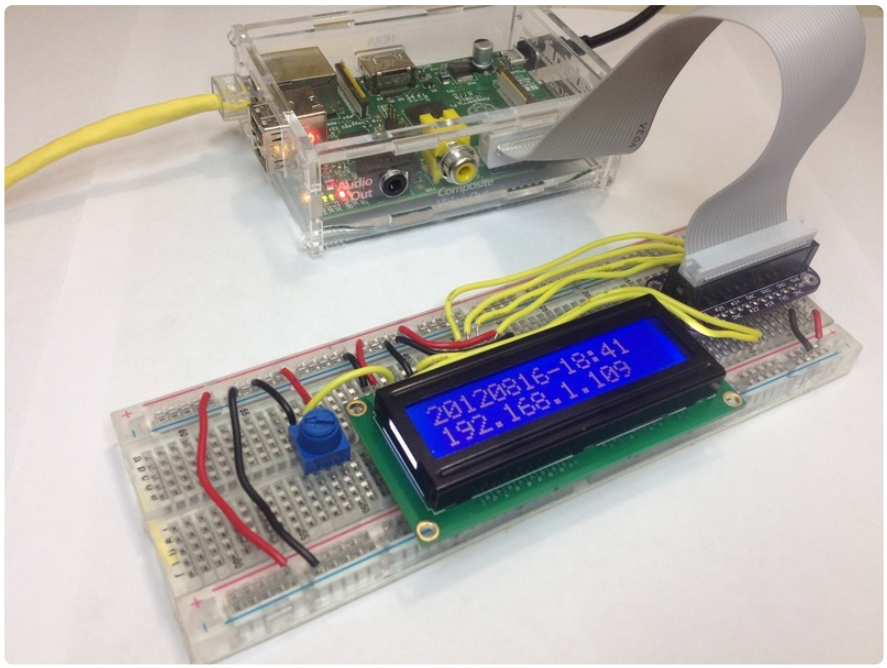


https://learn.adafruit.com/drive-a-16x2-lcd-directly-with-a-raspberry-pi

Last updated on 2024-03-08 01:44:04 PM EST

Table of Contents

Overview

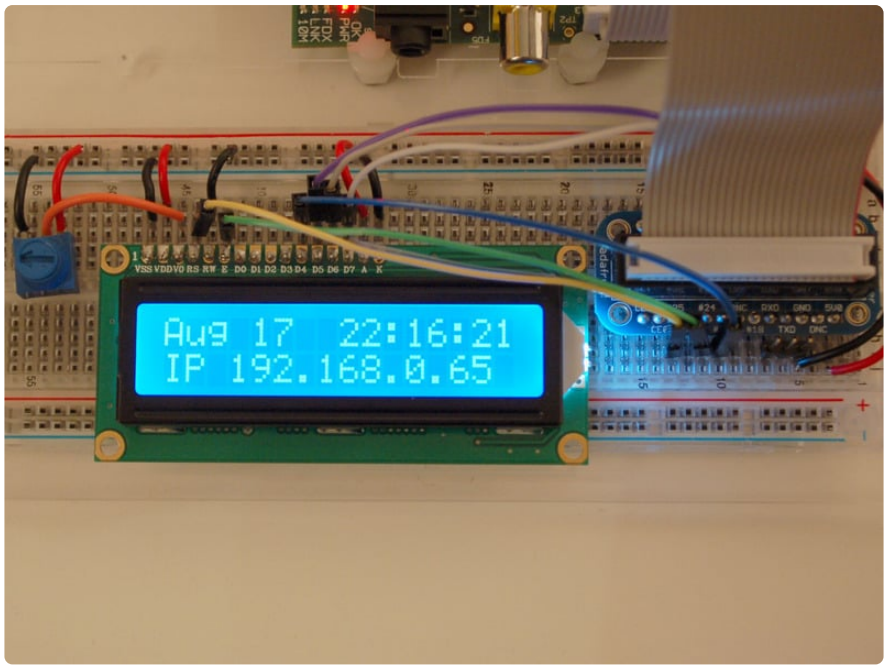


Adding a LCD to any project immediately kicks it up a notch. This tutorial explains how to connect an inexpensive HDD44780 compatible LCD to the Raspberry Pi using 6 GPIO pins. While there are other ways to connect using I2C or the UART, this is the most direct method that gets right down to the bare metal.

This technique:

- allows for **inexpensive LCDs** to be used
- does not require any **i2c drivers**
- won't steal the only **serial port** on the Pi.

The example Python code sends date, time, and the Pi's IP address to the display. If you are running a Pi in headless mode, being able to determine the IP address at a glance is really handy.



To Follow This Tutorial You Will Need

- Standard LCD 16x2 + extras (http://adafru.it/181)
- Pi T-Cobbler Plus, (http://adafru.it/2028) Pi Cobbler Plus for Model B+ / Pi 2 (http://adafru.it/2029) or the original Pi Cobbler (http://adafru.it/914)
- (2) Half size breadboards (http://adafru.it/64)
- Hook-up Wire (https://adafru.it/aM5)
- A Raspberry Pi (https://adafru.it/DBf) (compatible with all 26pin and 40pin Pi releases to date)

You can use nearly any character LCD with this tutorial - it will work with 16x1, 16x2, 20x2, 20x4 LCDs. It will not work with 40x4 LCDs

Wiring the Cobbler to the LCD

The LCD

Whenever you come across a LCD that looks like it has 16 connectors it is most likely using a HD44780 controller. These devices provide the same pinouts making them relatively easy to work with. The LCD uses a parallel interface meaning that we will need many pins from our raspberry pi to control it. In this tutorial we will use 4 data pins (4-bit mode) and two control pins.

The **data pins** are straightforward. They are sending data to the display (toggled high/low). We will only be using write mode and not reading any data.

The **register select** pin has two uses. When pulled low it can send commands to the LCD (like position to move to, or clear the screen). This is referred to as writing to the instruction or command register. When toggled the other way (1) the register select pin goes into a data mode and will be used to send data to the screen.

The **read/write** pin will be pulled low (write only) as we only want to write to the LCD based on this setup.

The **enable** pin will be toggled to write data to the registers.

LCD Pinout

1. Ground
2. VCC - **5v not 3.3v**
3. Contrast adjustment (VO) from potentiometer
4. Register Select (RS). RS=0: Command, RS=1: Data
5. Read/Write (R/W). R/W=0: Write, R/W=1: Read (**we won't use this pin**)
6. Clock (Enable). Falling edge triggered
7. Bit 0 **(Not used in 4-bit operation)**
8. Bit 1 **(Not used in 4-bit operation)**
9. Bit 2 **(Not used in 4-bit operation)**
10. Bit 3 **(Not used in 4-bit operation)**
11. Bit 4
12. Bit 5
13. Bit 6
14. Bit 7
15. Backlight LED Anode (+)
16. Backlight LED Cathode (-)

Before wiring, check that your LCD has an LED backlight, not an EL backlight. LED backlights use 10-40mA of power, EL backlights use 200+ma! EL backlights are often cheap to get but are not usable, make sure you don't use one or you will overload the Pi. Some cheap LCDs that have LED backlights do not include a resistor on the LCD module for the backlight, if you're not sure, connect a 1Kohm resistor between pin 15 and 5V instead of connecting directly. **All Adafruit LCDs have LED backlights with built in resistors so you do not need an extra resistor!**

5v LCD vs 3.3v Pi

The raspberry Pi GPIOs are designed for 3.3v, but our LCD is a 5v device. **It's fine to use a 5v display, but only if we are sending data out of the Pi**. We are not going to use the 3.3v power rail on the Cobbler, and we will tie the **RW** (read/write) pin of the display to GND as we do not want the display sending a 5v signal into the Pi.

Don't cross the streams!

Wiring Diagram

First, connect the Cobbler power pins to the breadboard power rail. +5.0V from the Cobbler goes to the red striped rail (red wire) and GND from the cobbler goes to the

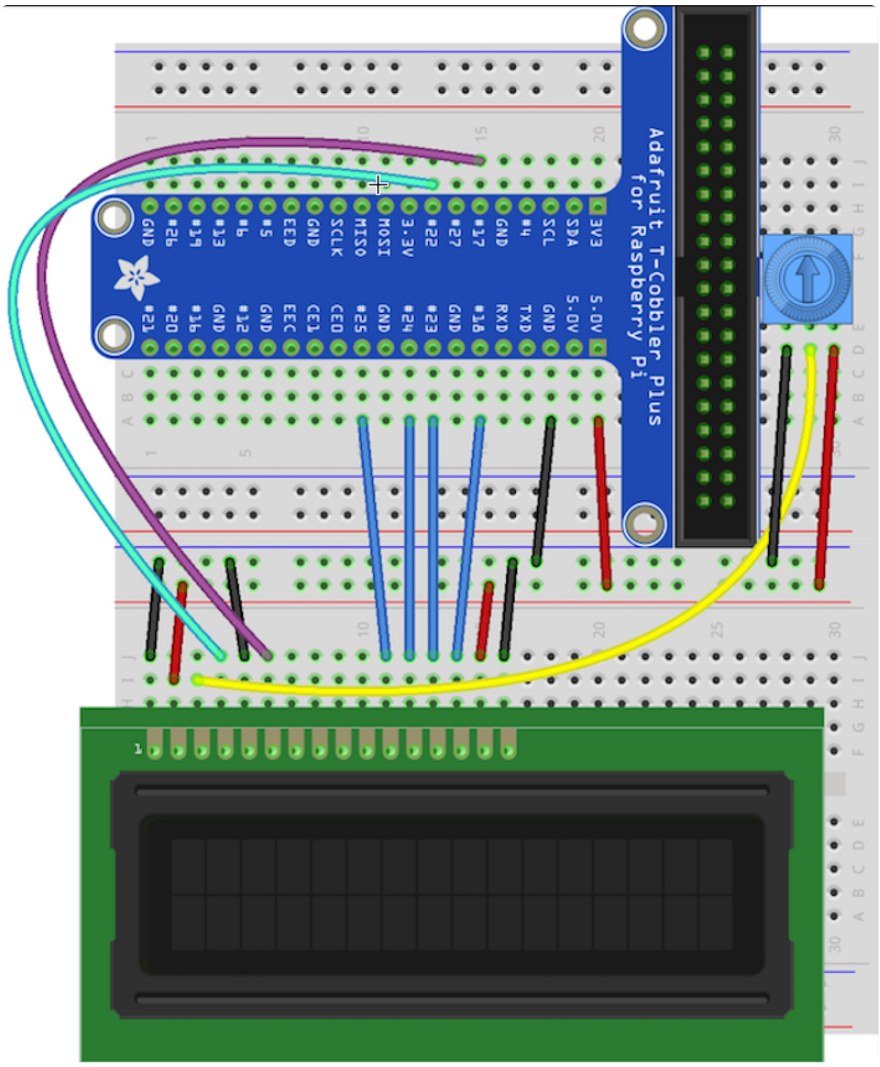
blue striped rail (black wire)

In order to send data to the LCD we are going to wire it up as follows

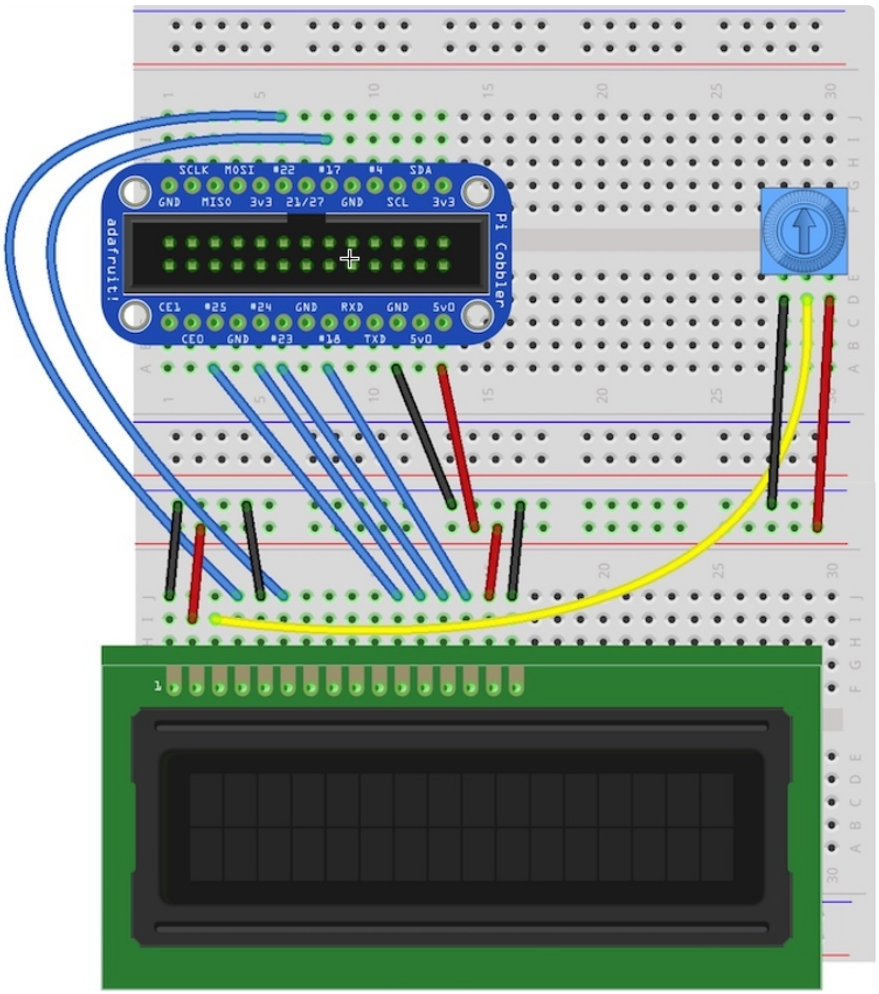
- Pin #1 of the LCD goes to ground
- Pin #2 of the LCD goes to +5V
- Pin #3 (Vo) connects to the middle of the potentiometer
- Pin #4 (RS) connects to the Cobbler #22
- Pin #5 (RW) goes to ground
- Pin #6 (EN) connects to Cobbler #17
- Skip LCD Pins #7, #8, #9 and #10
- Pin #11 (D4) connects to cobbler #25
- Pin #12 (D5) connects to Cobbler #24
- Pin #13 (D6) connects to Cobber #23
- Pin #14 (D7) connects to Cobber #18
- Pin #15 (LED +) goes to +5V (red wire)
- Pin #16 (LED -) goes to ground (black wire)

Then connect up the potentiometer, the left pin connects to ground (black wire) and the right pin connects to +5V (red wire)

Here's a sketch of the T-Cobbler Plus version:

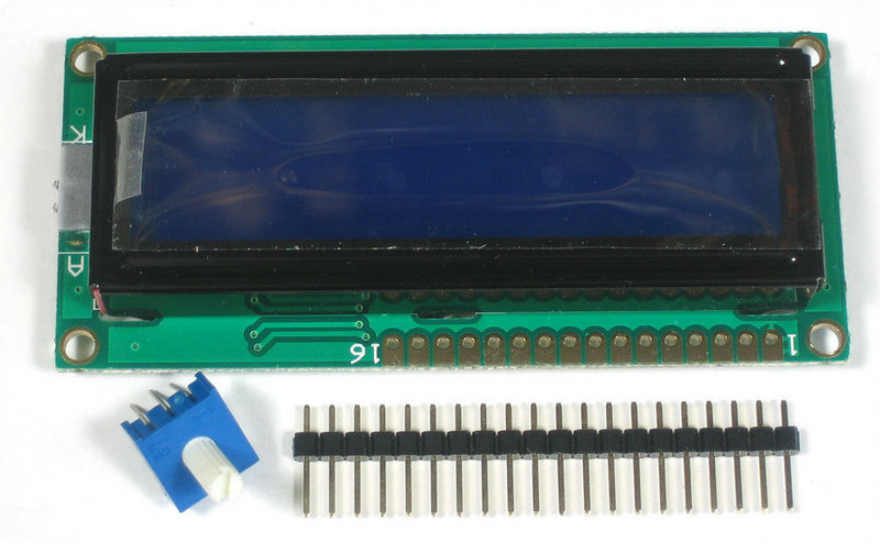


Sketch for a 26 pin Cobbler Raspberry Pi (v1, v2)

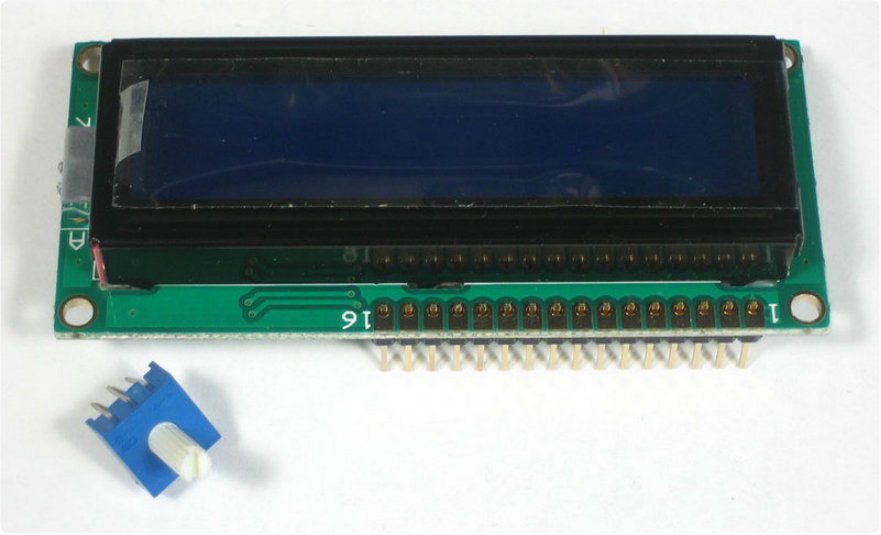


Preparing the LCD

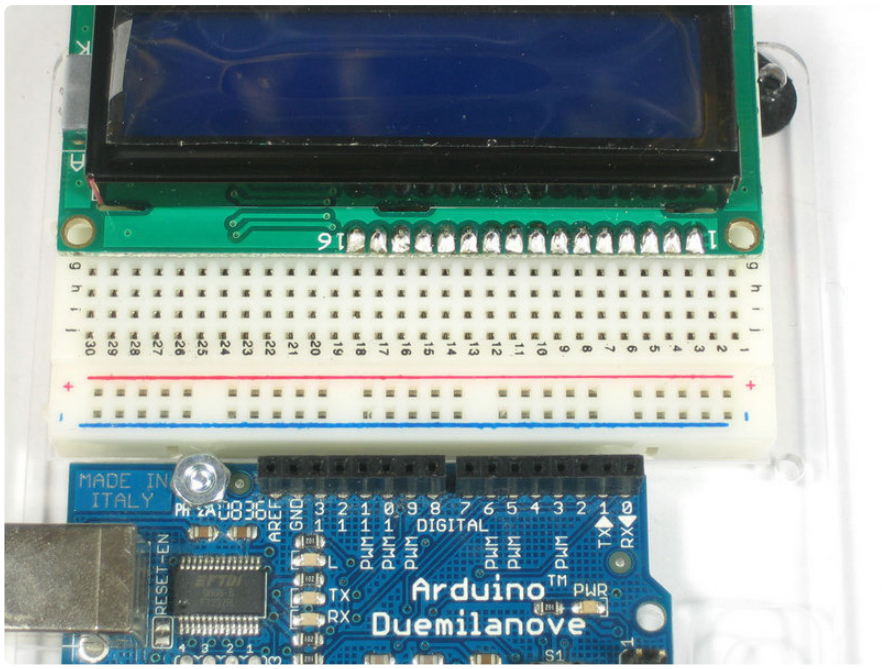
Before you start, make sure you have a strip of 0.1" male header and a 10K potentiometer. All Adafruit Character LCDs come with these parts so you should be good to go.



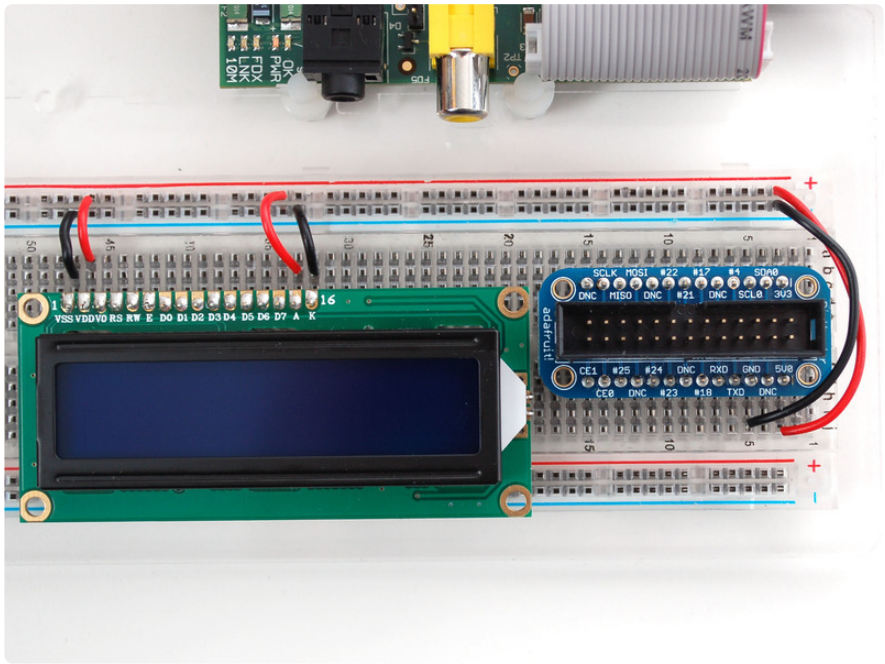
Most LCDs have a strip of 16 pins on the top, if the header is a little longer, just break it off until its the right length

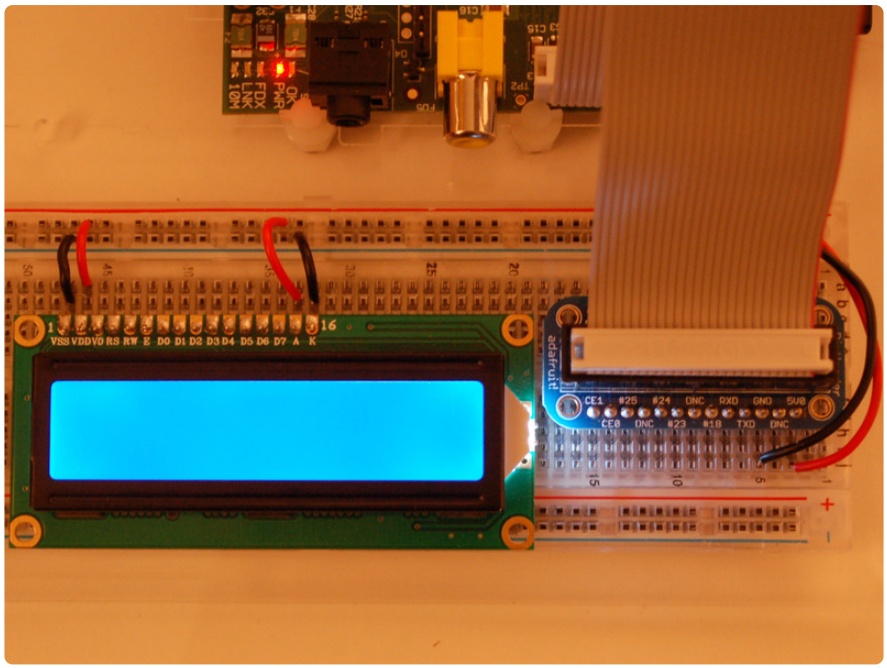


Next you'll need to solder the header to the LCD. **You must do this, it is not OK to just try to "press fit" the LCD!**



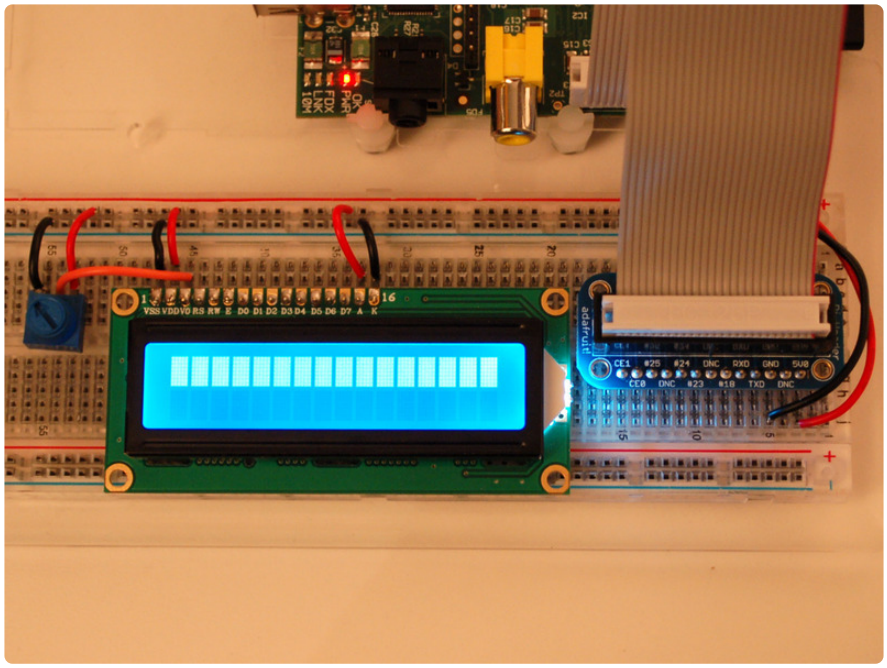
Start by connecting the **5V** and **GND** wires from the cobbler to the breadboard. Then connect pins #1, #2 and #15, #16 to the breadboard power rails as shown. The backlight should come on. If it doesn't, check the wiring!





Next, wire up the contrast potentiometer as shown above, with the middle pin connecting to LCD pin #3 and the other two pins going to 5V and ground.

Twist the potentiometer until you see the first line of the LCD fill with boxes. If you don't see the boxes, check your wiring!



Finish the wiring for the **RS, RW, EN, D4, D5, D6,** and **D7** pins as shown in the diagram up top.

That's it! Now you're ready to run the Python script to draw text on the screen!

Necessary Packages

Update Your Pi to the Latest Raspbian

Your Pi will need to be running the latest version of Raspbian. This tutorial was written using Raspbian Stretch (Nov. 2018). Checkout our guide for Preparing an SD Card for your Raspberry Pi (https://adafru.it/dDL) if you have not done so already. After the installation is complete be sure and run the following commands to make sure your installation packages are up to date.

```
sudo apt-get update -y
sudo apt-get upgrade -y
```

Install pip3

pip3 is already installed with a full Raspbian installation, but the Raspbian Lite does not include pip3 so it needs to be installed as shown below.

```
sudo apt-get install python3-pip
```

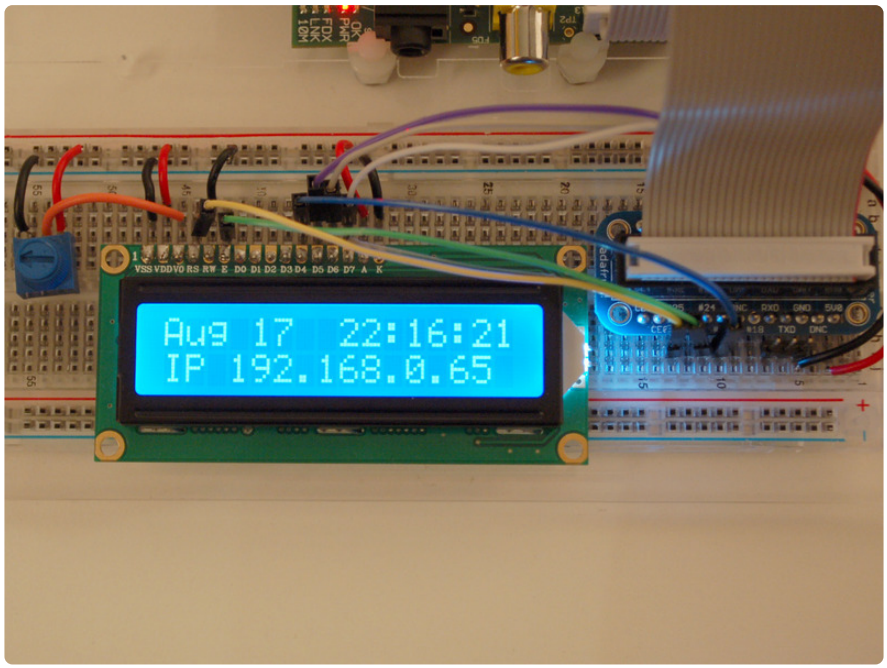
Install adafruit-blinka

```
sudo pip3 install adafruit-blinka
```

Install adafruit-circuitpython-charlcd

```
sudo pip3 install adafruit-circuitpython-charlcd
```

Python Script



The following code can be downloaded to your raspberry pi and run to get the date, time and IP address of your machine on the LCD display.

The Code

```
# SPDX-FileCopyrightText: 2018 Mikey Sklar for Adafruit Industries
#
# SPDX-License-Identifier: MIT
# Modified by Jonathan Seyfert, 2022-01-22
# to keep code from crashing when WiFi or IP is unavailable
from subprocess import Popen, PIPE
from time import sleep, perf_counter
from datetime import datetime
import board
import digitalio
import adafruit_character_lcd.character_lcd as characterlcd

# Modify this if you have a different sized character LCD
lcd_columns = 16
lcd_rows = 2

# compatible with all versions of RPI as of Jan. 2019
# v1 - v3B+
lcd_rs = digitalio.DigitalInOut(board.D22)
lcd_en = digitalio.DigitalInOut(board.D17)
lcd_d4 = digitalio.DigitalInOut(board.D25)
lcd_d5 = digitalio.DigitalInOut(board.D24)
lcd_d6 = digitalio.DigitalInOut(board.D23)
lcd_d7 = digitalio.DigitalInOut(board.D18)


# Initialise the lcd class
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
                                      lcd_d7, lcd_columns, lcd_rows)
```

```
# looking for an active Ethernet or WiFi device
def find_interface():
#    dev_name = 0 # sets dev_name so that function does not return Null and crash
code
    find_device = "ip addr show"
    interface_parse = run_cmd(find_device)
    for line in interface_parse.splitlines():
        if "state UP" in line:
            dev_name = line.split(':')[1]
            return dev_name
    return 1 # avoids returning Null if "state UP" doesn't exist

# find an active IP on the first LIVE network device
def parse_ip():
    if interface == 1: # if true, no device is in "state UP", skip IP check
        return "not assigned " # display "IP not assigned"
    ip = "0"
    find_ip = "ip addr show %s" % interface
    ip_parse = run_cmd(find_ip)
    for line in ip_parse.splitlines():
        if "inet " in line:
            ip = line.split(' ')[5]
            ip = ip.split('/')[0]
            return ip # returns IP address, if found
    return "pending      " # display "IP pending" when "state UP", but no IPv4
address yet

# run unix shell command, return as ASCII
def run_cmd(cmd):
    p = Popen(cmd, shell=True, stdout=PIPE)
    output = p.communicate()[0]
    return output.decode('ascii')

# wipe LCD screen before we start
lcd.clear()


# before we start the main loop - detect active network device and ip address
# set timer to = perf_counter(), for later use in IP update check
interface = find_interface()
ip_address = parse_ip()
timer = perf_counter()

while True:
    # check for new IP addresses, at a slower rate than updating the clock
    if perf_counter() - timer >= 15:
        interface = find_interface()
        ip_address = parse_ip()
        timer = perf_counter()

    # date and time
    lcd_line_1 = datetime.now().strftime('%b %d  %H:%M:%S\n')

    # current ip address
    lcd_line_2 = "IP " + ip_address

    # combine both lines into one update to the display
    lcd.message = lcd_line_1 + lcd_line_2

    sleep(1)
```

Download the Code

Let's put this file right in your home directory for simplicity. The wget command makes things easy.

```
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/
master/Drive_a_16x2_LCD_with_the_Raspberry_Pi/
Drive_a_16x2_LCD_with_the_Raspberry_Pi.py
```

Running the Code

The following command will start the program and you should see the LCD display come to life with date, time and IP address.

```
sudo python3 ./Drive_a_16x2_LCD_with_the_Raspberry_Pi.py
```

Display Time & IP on Every Boot

It's all fine and dandy to have a script which we can manually run, but wouldn't it be nice to have the time and ip address pop up on the display when the Raspberry Pi boots up? This is done using an init script which runs our Python code every time your Raspberry Pi boots up.

Download the service file

The following command will allow you to download the lcd.service file directly to your Pi.

```
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/
master/Drive_a_16x2_LCD_with_the_Raspberry_Pi/lcd.service
```

```
[Unit]
Description=LCD date|time|ip

[Service]
ExecStart=/usr/bin/python3 Drive_a_16x2_LCD_with_the_Raspberry_Pi.py
WorkingDirectory=/home/pi
StandardOutput=inherit
StandardError=inherit
Restart=always
User=pi

[Install]
WantedBy=multi-user.target
```

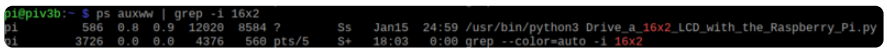
Place the lcd.service file

The lcd.service file needs to be copied into the correct location and the systemctl command can be used to start | stop | enable the service. It is a good idea to test this before enabling as there might be a minor path difference on your system.

```
sudo cp lcd.service /etc/systemd/system
```

Test the lcd.service

```
sudo systemctl daemon-reload
sudo systemctl start lcd.service
ps auxww | grep -i 16x2
```



The following commands read the updates to the service file, start the lcd.service and confirm that the process is running. If the script shows up in the 'ps' command output you have done everything correctly and can now enable the service and reboot. The service should activate upon bootup automatically.

Enable lcd.service

```
sudo systemctl enable lcd.service
```

Now on each boot the LCD will automatically show the date/time/ip address on startup. This means you will know when the Pi is reachable and what the ip address is without having to plug a monitor in.

Time Zone

Last, but not least: My Pi came configured with UT (Universal Time). I prefer to see time based on my time zone, which is Mountain. Here is how to configure time on the Pi for any location. This is a one time configuration setting that will survive between reboots.

We can use raspi-config to easy set our timezone. Choose the following:

- Localisation Options
- Change Timezone
- Continent / Country

- Time zone

Run raspi-config

```
sudo raspi-config
```