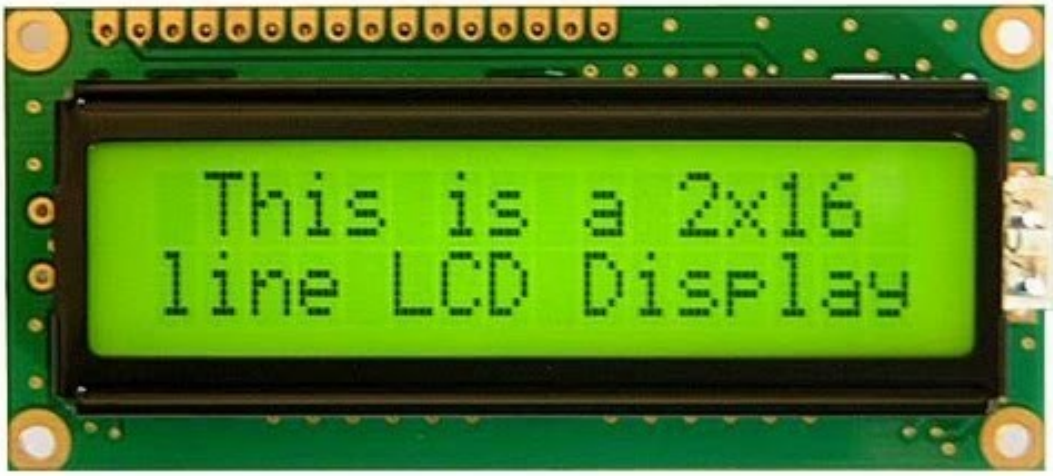
Use Your Pi!

…. with a 16x2 LCD



Contents:

1. Introduction

The Raspberry Pi has been around in various form-factors and iterations since 2012, and can be used for so many purposes that they are too numerous to mention here.

The uses which are described here have been tested on a Pi Zero W, but may well be equally successful on any version from Raspberry Pi 2 upwards – it is up to you to try them out!

The lines of code used in this book are a combination of adapted sources, which are cited where used, and my own invention. At this point I feel I should say that I have no particular expertise or diplomas in coding, so use what I present whilst understanding that I am trying to enable those who are interested to extend their knowledge, without having to spend hours scouring the internet for solutions simply to "make things work". I can offer no cast-iron guarantees of success so, for that reason, I have to state the customary warning that anything you try is at your own risk.

I hope that what I present will inspire you to experiment, and that you will be able to put any knowledge that you gain to further use, improving on my source adaptations, methods and coding.

SEDET HUMEROS GIGANTES NANUM SUM

2. Components and Tools

- A Raspberry Pi – Any model with 40 GPIO pins and wi-fi on board. A Pi Zero WH is ideal for those who do not wish to solder GPIO pins to their Pi Zero. And, of course, you'll need a power source for the Pi (the official ones are best). You may also need a monitor and HDMI cable for your Pi (although it can be run "headless").
- A Micro SD Card for the PI – Class 10 is best, of any capacity above 8GB. You do not need a large capacity card, even though the Pi may be capable of using sizes larger than 128GB.
  These days 16GB cards are becoming more difficult to find, but either that or 32GB would probably be ideal, depending on how much you want to spend. Personally, I have never had a problem using this brand:

  You can probably find one on Amazon or eBay for under £5.00

- A PC or laptop. This could be Windows, Chromium or Linux but, for the sake of simplicity, any further references to a PC should be considered as meaning "a PC running Windows 10". This will be needed for downloading and installing the Raspberry Pi Operating System on the SD card.
- A USB Keyboard and a USB Mouse for the Pi (unless you are running it "headless").
- A 16x2 / 1602 LCD display. A cheap one will do, so long as it has holes for a 16 pin header. Mine was from Amazon.co.uk and cost £3.50:

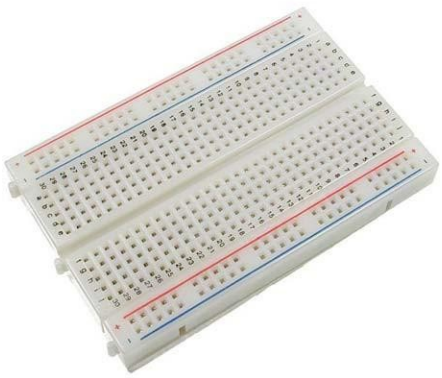
https://www.amazon.co.uk/gp/product/B00N8K2BYM/ref=ppx_yo_dt_b_asin_title_o03_s00?ie=UTF8&psc=1

- A length of male header pins (at least 16 pins long). It's unlikely you'll find a length of exactly 16, but they are easily cut to the length required. You can also find these on Amazon, amongst other places:

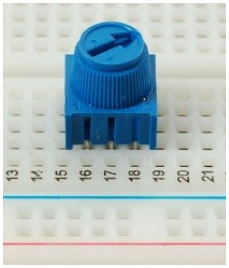
https://www.amazon.co.uk/SODIAL-2-54mm-Straight-Connector-Arduino/dp/B00K67XSCY/ref=sr_1_15?dchild=1&keywords=Header+pins&qid=1609632862&sr=8-15

- A small soldering iron and solder (unless you can find a display with a pre-soldered header, and are prepared to wait for its delivery)

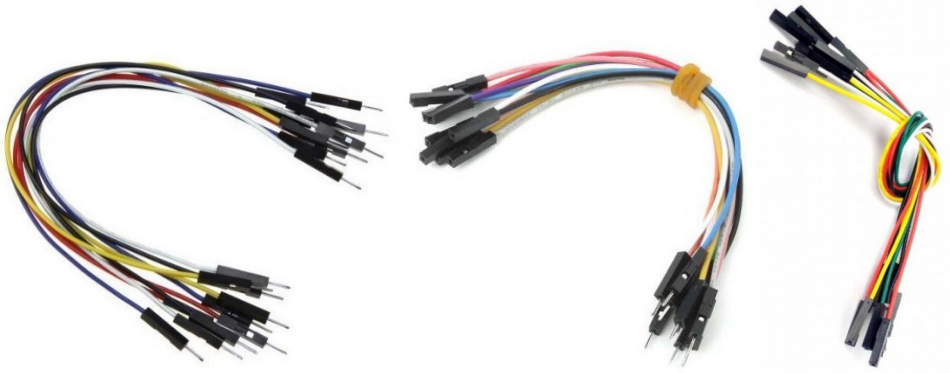
- A "breadboard". These are available from The Pi Hut: https://thepihut.com/



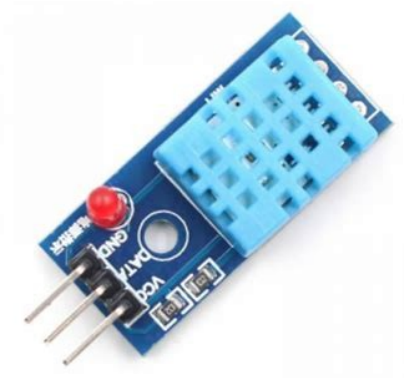
- A 10K-Ohm breadboard trimmer potentiometer (trim-pot) to control the brightness of the LCD. These are cheaply available from The Pi Hut, Amazon and many other online stores.



- Some jump wires. It is advisable to buy a collection of male-male, male-female and female-female wires. The number of each type you will need will vary according to how many you connect directly between the Pi and the display, and how many you use to connect these to the breadboard. The DuPont type are ideal, and they too are available for purchase at The Pi Hut, Amazon and at many other places online.



- Optionally a DHT11 Humidity and Temperature sensor module. Try to get the type that is mounted on a small printed circuit board and has a red led and just 3 pins. This type works when connected in the way shown in the Fritzing diagram in Chapter 5, without the need for an additional resistor and capacitor. Unfortunately, I could not find the modular version, pictured below, in the Fritzing components library, although it should be understood that the type shown in the Fritzing diagram should be taken to represent the modular version below. The actual sensor unit in the diagram, with four pins and without an led, will almost certainly need the additional resistor and capacitor; the module in the picture below already has them on its circuit board, to the right of the hole.



3. Set up your Pi

N.B. If you are already familiar with this procedure, you can skip to Chapter 4.

1. Using your PC, browse to https://www.raspberrypi.org/software/ and scroll down the page to: **Install Raspberry Pi OS using Raspberry Pi Imager**
2. Select the version of the **Imager** to download according to the Operating System (OS) of your PC. This will NOT install the Raspberry Pi OS on your PC; it will simply download an app which you can then use to write the Raspberry Pi OS to your SD card, whilst optimising the card for use.
3. If, like me, you select the Download for Windows choice, a box will pop up asking what you want to do with this file. Choose the "Save" option. You can then choose where to save it to (the usual default folder is "Downloads", which is fine).
4. Depending on the browser you use on your PC (I prefer Firefox, because it makes download progress more visible than Chrome or Edge), the download will progress fairly quickly. Open the file and allow it to install the Imager. You now have the means to write the Raspberry Pi OS to an SD card …. but you do not yet have the image of the OS to write!
5. Put your SD card either directly into your PC (or into a USB SD card reader which you then put into your PC), and cancel any messages Windows may give you. You will not need to format it, as the Imager app takes care of that. If you use a card which is 64GB or larger, however, you will need to ensure that it is formatted to FAT32 first.
6. Open the Raspberry Pi Imager app, and allow it "to make changes to your device" in Windows.
7. The Imager will then present you with a list of versions of the Raspberry Pi OS from which to choose an installation, and an SD card option. The 32-bit version of the OS is the recommended and best option for our needs, so select that, then select the correct SD card. It should detect your micro SD card by its drive letter and size, and indicate the card as the destination to which it will write the OS. Above all, at this point, do not allow it to write the image to a drive which does not have the drive letter or size you expect for your SD card. Hit the "write" button, and let the app do its work.
8. If you have any difficulties with the Imager app, try downloading and using **Etcher** instead. Note that you will also need to download manually an image of Raspberry Pi OS (from the software page in Step 1) to use with Etcher, so that you can use Etcher to browse to the image and then write it to your SD card. If you use a card which is 64GB or larger, you will probably need to format it to FAT32 first.
9. *Optional:* To set up your Pi to run "Headless" on first boot (without a monitor, and controlled via SSH from another computer), navigate to your SD card in your (Windows) PC **before booting it in your Pi**, and in the Boot folder create a blank file named *ssh*. Then create another file named *wpa_supplicant.conf,* open it with Notepad and in it save the following lines:

```
country=gb                    (Here enter your country's international identity code)
update_config=1
ctrl_interface=/var/run/wpa_supplicant
```

```
network={
 scan_ssid=1
 ssid="MyNetworkSSID"      (Here enter the name of your network within the quotes)
 psk="Pa55w0rd1234"        (Here enter your network's access password within the quotes)
}                          (Don't forget this curly bracket!)
```

10. Eject the SD card from your PC and put it in your Pi. Connect the Pi to its mouse, keyboard (and monitor, if not headless) then plug it in. Allow it a couple of minutes to boot (and, if you followed step 9, find the network). You should now be able to log in – you may need to try a couple of times - or use ssh to do this (Username: pi, Password: raspberry) and configure the OS / create users / change passwords etc. The first thing you should do is update and upgrade the Operating System. Open the terminal (a black command-line screen) and then type:

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

hit ENTER, and allow the system to update. It may take a while.
Then proceed with the next step:

11. Open the terminal again and type:

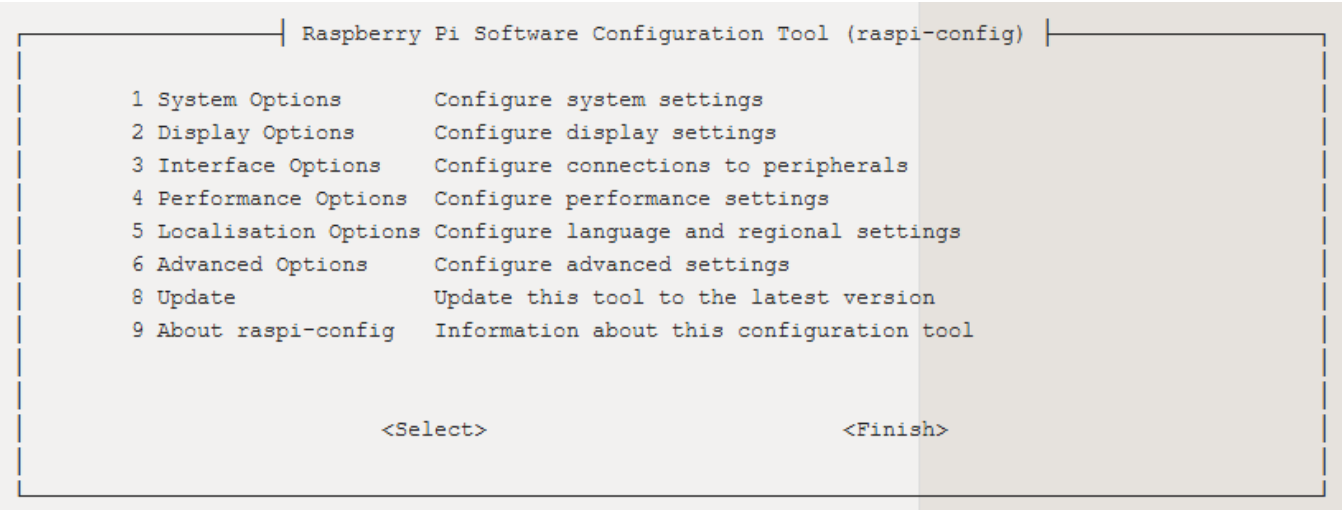
```
sudo raspi-config
```
(+ hit **Enter**)

This will open configuration settings.

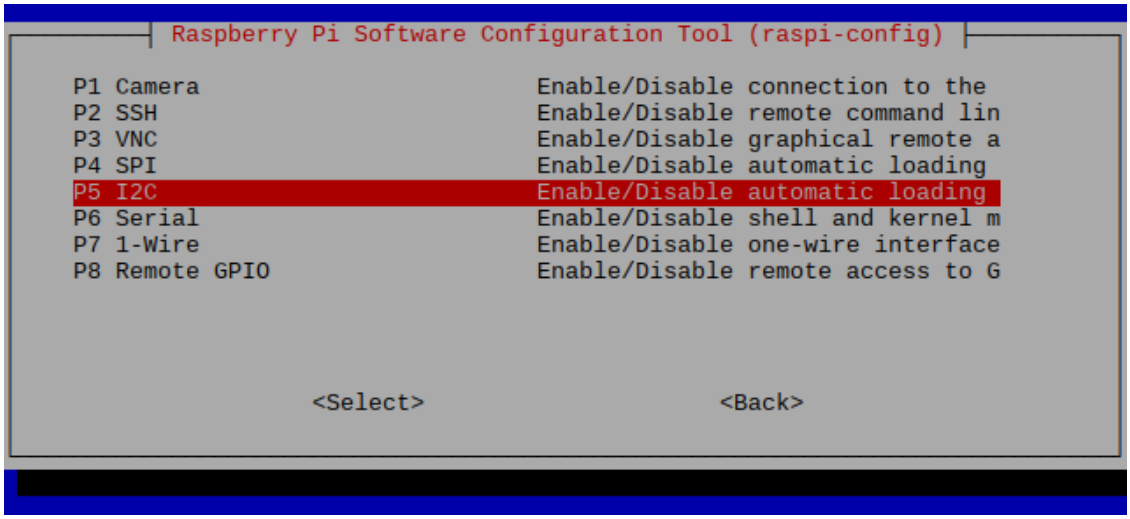
12. During configuration, enable as many of the interfaces as you need. I would certainly enable SSH to be able to control the Pi from its terminal screen on another computer, and you may wish to enable VNC to be able to do the same thing using the Graphical User Interface (GUI). Other choices will depend upon what else, by way of sensors, you intend to connect to the GPIO pins. You will probably not need to attach a camera, as suggested by the third of the images below, but you *may* need to enable the SPI, I2C, Serial or 1-Wire interfaces. Check what your components require first.

Raspi-config during initial setup. The first setup screen - explore the options:

(N.B. This screen will appear with a blue background, similar to the second image)



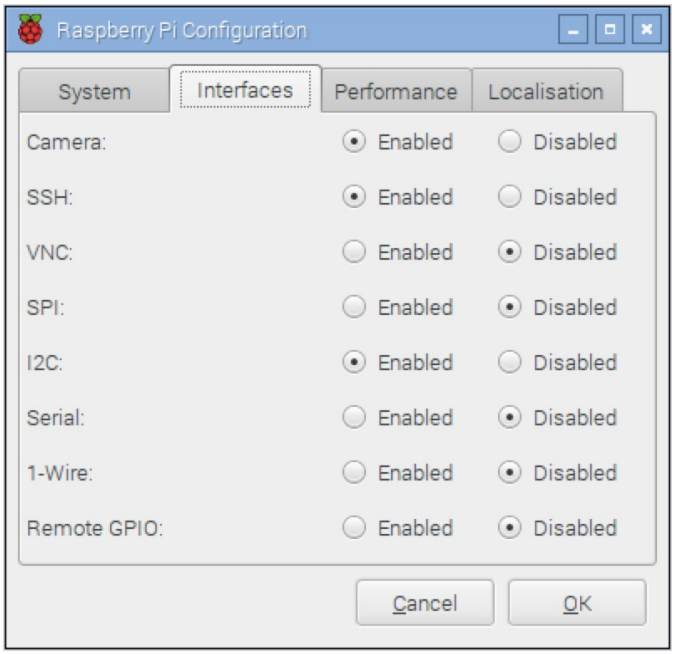
You will need to go to option 1 (above) to configure network and internet access, and option 3 from the above screen will give you the screen below to configure connections to devices:



It would also be wise to go to no. 6 Advanced Options on the first screen, and from there select the Expand Filesystem option. This will enable you to make full use of your SD card. The system will then need to reboot.

After setup you can go to Preferences >> Raspberry Pi Configuration to fine-tune or amend your choices.

Use the options under the “System” tab to connect to your local router / network:



To learn how to control your Pi using SSH (Secure Shell) from a PC, or to set up other forms of remote access, see this guide:

https://www.raspberrypi.org/documentation/remote-access/

4. Assembly of the LCD

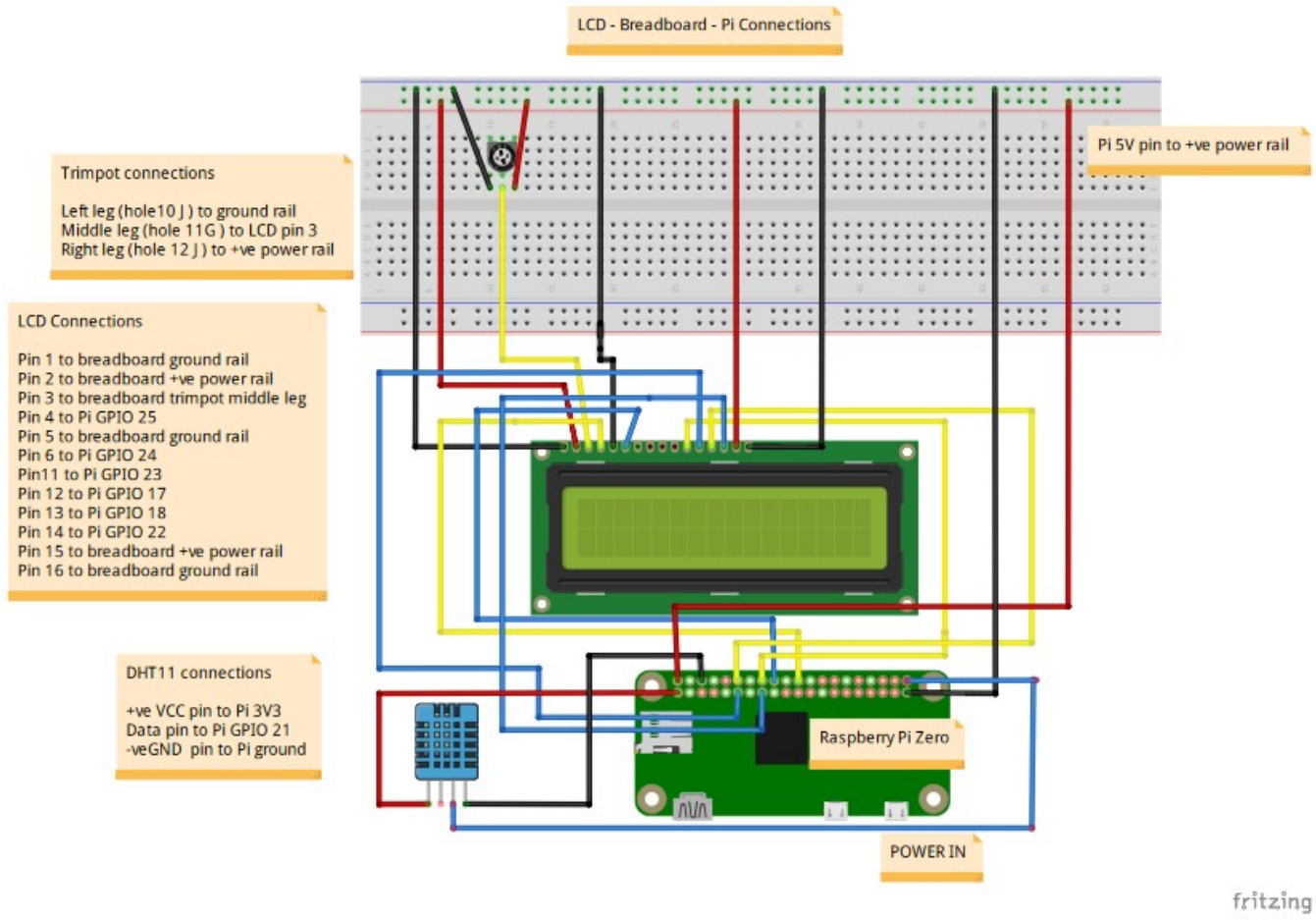
There are several varieties of 1602 LCD screens on the market. Some have pins (or holes for pins) on the end of the screen's circuit board for connecting jump wires to the I2C pins on the Raspberry Pi, whilst many have a row of 16 holes along one edge for attaching a header to make power and data connections. It is this second type which is described in this chapter.

- First of all, you need to cut the header to length. A junior hacksaw will do this nicely.
- Then decide whether you are going to mount the LCD directly onto the breadboard. If so, you will need to solder the 16-pin header onto the LCD **with the shorter pins protruding through to the screen side,** which is where you will solder them. If you decide to connect the LCD to the breadboard and the Pi using jump wires, then you should mount the header so that the short pins protrude through to the back of the LCD and solder them there.
- Using a small electric soldering iron (and perhaps some Blu-Tack to support the LCD), carefully solder the 16-pin header to the LCD's circuit board. I stuck the header into the breadboard, then supported the Pi Zero by placing books beneath it while soldering.
- Once assembled, check that the soldered joints have enough solder on the contacts, and that the solder has not run to make a short circuit between two contacts.

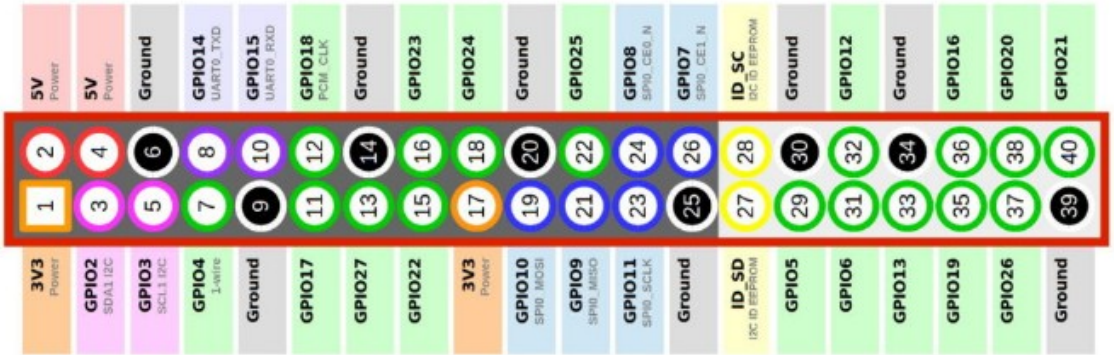
  You can now move on to connecting everything up, and trying it out!

5. Connections

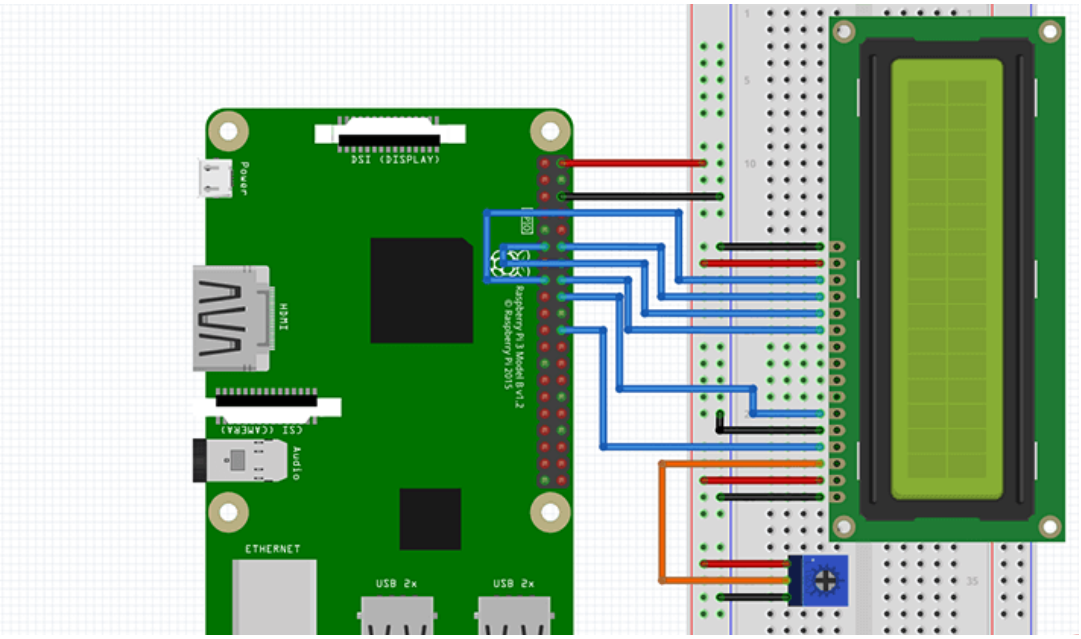
Use the Fritzing diagram below to connect everything as shown. It is not necessary to use so many jumper wires if you have soldered the header facing downwards so that you can mount the LCD directly onto the breadboard. If it is difficult to see where to make a connection, follow the description.



Pi G

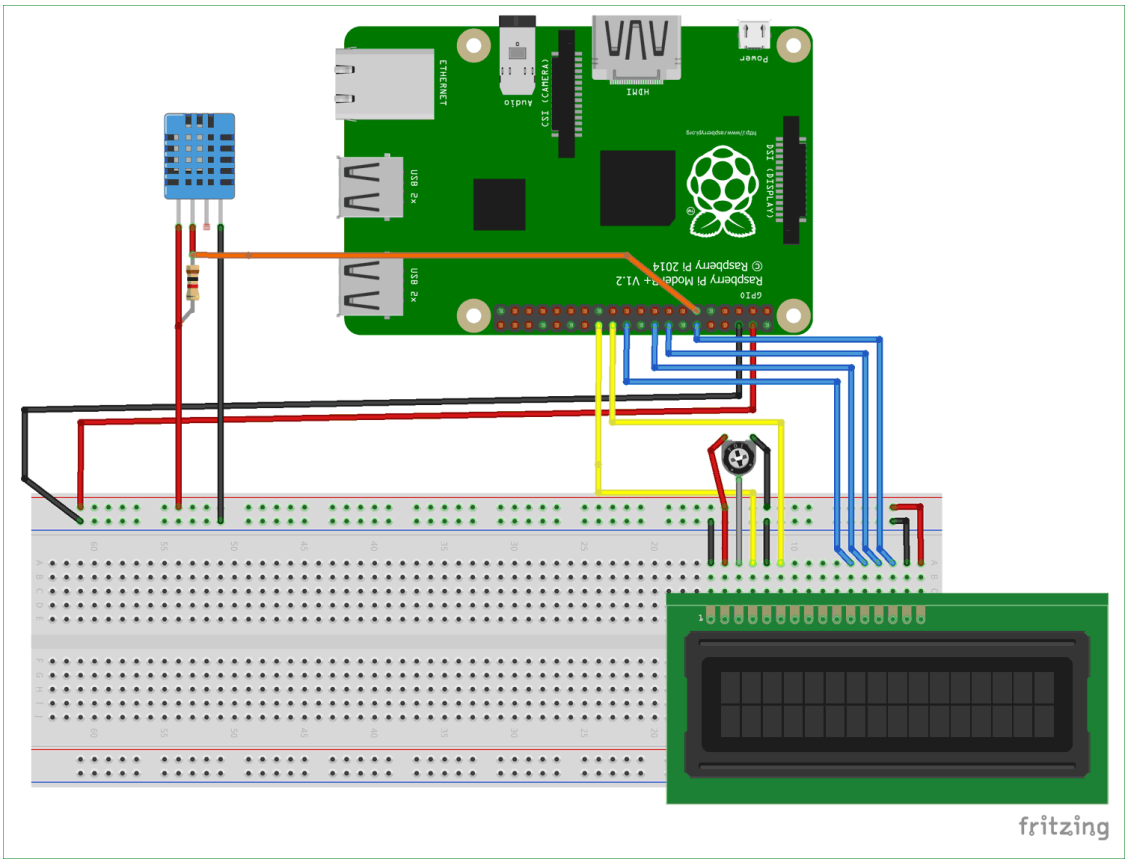


Rasp



Pi, LCD on breadboard, and trimpot without DHT11 sensor module

If you choose to mount your Pi on the breadboard and have the DHT sensor type with four pins and without an LED, you could connect them in the following way:



Note the use of a pull up resistor of value 1k Ohm on the data pin of the DHT11 sensor.

The type of Pi used does not matter, so long as it has 40 GPIO pins but, if you choose to use the pins shown in this diagram, where the Pi is turned around, you will need to number them correctly when you set up the Python software.

6. Use Your Pi LCD as a Temperature and Humidity display

To set up our Pi and LCD to display ambient room temperature and humidity, we will first need to download and install updates to the Raspberry Pi Operating System (OS). This procedure is described on page 5.

Then we need to install the correct software libraries to the correct locations (directories) on the Pi to enable us to obtain data from the sensor and display it on the LCD.

First, ensure that the dependencies for the main LCD library from Adafruit are installed. Using the terminal screen, type the following and press ENTER:

```
sudo apt-get install build-essential python-dev python-smbus python-pip
```

Then:

```
sudo python -m pip install --upgrade pip setuptools wheel
```

```
sudo pip3 install adafruit-circuitpython-dht
```

Then:

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

and:

```
sudo apt install libgpiod2
```

Next, enter the following command to open a new file with a text editor.

```
nano LCD_DHT11.py
```

This file is going to store the software of the program we are going to use. Give it a name you will remember. Here, we have called it: `LCD_DHT11.py`

An empty file will open.

You should now copy and paste the code given on the next page into the terminal (after copying, simply right-click on the black terminal screen to paste):

```
# Working Python3 Digital Version of DHT11 Software
# Program to read the values of Temp and Hum from the DHT11 sensor and display them on the LCD
# The LCD backlight is not controlled in this program
# Ensure you have installed the adafruit_character_lcd library
# Ensure you have installed the Adafruit_dht library
# Ensure that you have installed digitalio library
# Ensure that the pin connections on your RPi are correct
# Start with command: python3 DHT11_LCD.py

#!/usr/bin/python3

import os
import sys
import time # For 'sleep' commands
import adafruit_character_lcd.character_lcd as characterlcd
import adafruit_dht # Sensor software
import board # LCD Hardware detection
import digitalio # LCD Software
lcd_rs = digitalio.DigitalInOut(board.D25) #RS of LCD is connected to GPIO 25 on Pi
lcd_en = digitalio.DigitalInOut(board.D24) #EN of LCD is connected to GPIO 24 on Pi
lcd_d7 = digitalio.DigitalInOut(board.D22) #D7 of LCD is connected to GPIO 22 on Pi
lcd_d6 = digitalio.DigitalInOut(board.D18) #D6 of LCD is connected to GPIO 18 on Pi
lcd_d5 = digitalio.DigitalInOut(board.D17) #D5 of LCD is connected to GPIO 17 on Pi
lcd_d4 = digitalio.DigitalInOut(board.D23) #D4 of LCD is connected to GPIO 23 on Pi
#lcd_backlight = 2 #If LCD is mono, i.e. colours cannot be changed, do not assign a GPIO pin
to backlight pin

dht = adafruit_dht.DHT11(board.D21)

lcd_columns = 16  # for 16*2 LCD
lcd_rows    = 2  # for 16*2 LCD
```

```
# Define and activate lcd and its pins
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_rows)

lcd.message = 'DHT11 with RPi \n- by Mark et al.' # Give an intro message
print ('DHT11 with RPi - Temperature and Humidity Sensor') # Intro on monitor

time.sleep(6.0) # wait for X secs, ideally more than 4

try:
    while 1: #Infinite Loop
        #humidity, temperature = Adafruit_DHT.read_retry(sensor_name, sensor_pin) #read from
sensor and save respective values in temperature and humidity variable
        try:
            lcd.clear() # Clear the LCD screen
            time.sleep(8.0) # Gives a break with a clear screen between readings

            # Display the value of temperature in Celsius and humidity as a percentage
            temperature = dht.temperature
            humidity = dht.humidity
            lcd.message = "Temp = {:.1f} C".format(temperature)
            print("Temperature = {:.1f} C".format(temperature))
            lcd.message = "\nHum = {}%".format(humidity)
            print("Humidity = {}%".format(humidity))

            time.sleep(8.0) # Wait for X sec then update the values (at least 4 seconds is
recommended for consistent returns)

        except RuntimeError as e:
            # Reading doesn't always work! Just print error and we'll try again
            print("Reading from DHT failure: ", e.args)


except KeyboardInterrupt: # If there is a KeyboardInterrupt when you press Ctrl+C, exit the
program and clean up
        print("Cleaning up LCD!")
        lcd.clear()
        lcd.message = "Cleaning Up!"
        time.sleep(3.0)
        lcd.clear()
        exit() # Comment out this command for reboot to work
        cmd = 'sudo reboot -h now' # For a reboot
        os.system(cmd) # For a reboot
```

Then hold CTRL and press X. Save the modified buffer by pressing Y. Hit ENTER to agree to the name you gave it.

You should now be able to issue the command (followed by ENTER) in the terminal, and see the LCD working, displaying the sensor module's readings:

```
Python3 LCD_DHT11.py
```

Please note that Adafruit's latest CircuitPython software for DHT sensors in Python3 does not produce such consistent results as their old software for Python2 used to. Do not be surprised to see error messages at first, such as: `('A full buffer was not returned. Try again.',)`, or: `('DHT sensor not found, check wiring',)` nevertheless allow the software to run for several minutes to allow it to "settle down".

If the LCD does not light up, check all your connections and solder points. If it lights up but seems to fail to display anything, then turn the screw on the trimpot gradually clockwise to increase the contrast. Readings should then start to appear. Be conservative in adjusting the trimpot screw, as most of these will tolerate being turned only 8 to 20 times backwards and forwards before they break.

To exit from the program, press and hold CTRL while hitting C on your keyboard.

To switch off your Pi, use the following command, and wait for the green light on the Pi to flash 10 times and go out before unplugging:

```
sudo shutdown –h now
```

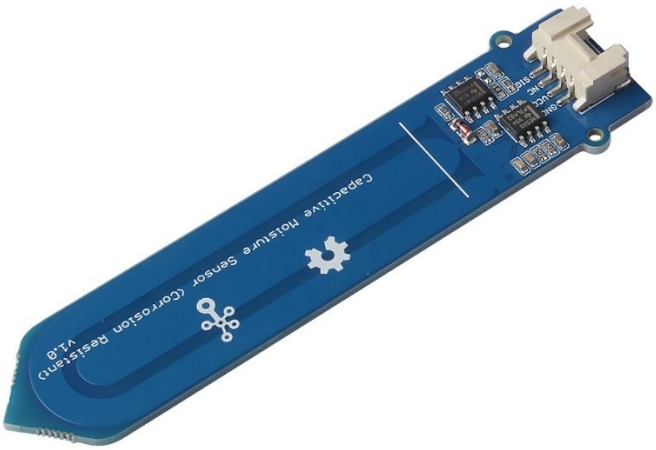
If you wish to adjust the settings in the DHT code above, then open the terminal editor and have fun experimenting!

```
nano LCD_DHT11.py
```
(+**ENTER**)

N.B. Anything written after a # symbol is a note to you, and is not part of the code.

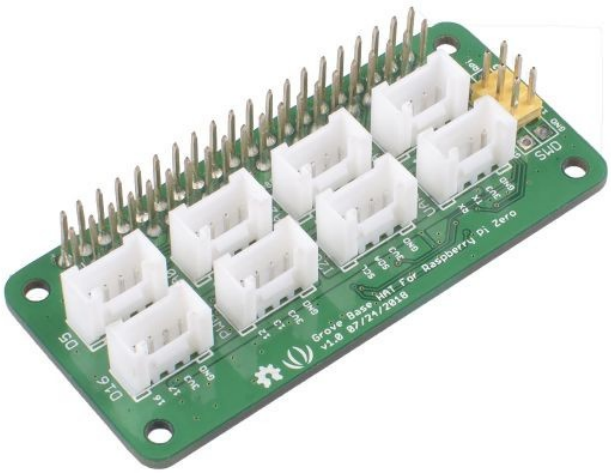
7. Use your Pi LCD as a Soil Moisture Sensor Display

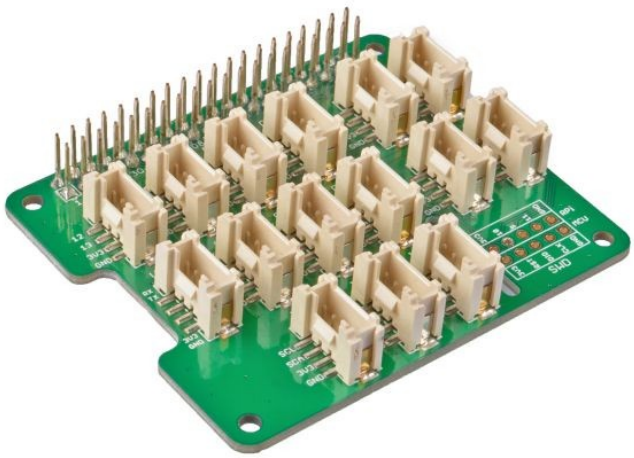
This is best achieved by purchasing a Grove **Capacitive** Moisture sensor, as this type does not corrode in the way that a standard, two-pronged, non-capacitive moisture sensor does. The sensor can be used for monitoring the soil moisture content of a potted plant, although care should be taken not to let moisture come into contact with the part of the sensor above the line or with the Pi. There are examples online of how the information can be sent to a mobile phone as an alert, or used to trigger a watering system.



More details from here: https://wiki.seeedstudio.com/Grove-Capacitive_Moisture_Sensor-Corrosion-Resistant/

In order to use this particular type of sensor, you will also need a Grove HAT for your Pi. There are four types available, depending on the model of your Pi. You can find them for sale on Amazon, at The Pi Hut, and at many other online electronic stores.





Grove Base HAT for a Raspberry Pi Zero …….. and for a Raspberry Pi.

The manufacturers, SeeedStudio, provide detailed instructions on how to fit them and use them on the Wiki page of each model. Their advantage is that they remove the need to fit some small components for managing the power to and signals from a variety of sensors, whilst sitting neatly over the top of a Pi, yet still allowing the Pi's GPIO pins to be used. The DHT sensor could therefore

still be connected to the GPIO pins, while the moisture sensor is connected to the HAT. The HAT keeps both the Pi and the sensor(s) safe from being damaged by excessive power draw /supply, although the same precautions should be taken when using the HAT's GPIO pins as when the Pi's own GPIO pins are used.

Notice that the I2C Port of the Raspberry Pi must be enabled for the HAT and moisture sensor to work.

Having fitted the HAT, we connect the LCD display to exactly the same GPIO pins as we did on the Pi.

Install the Grove software using the terminal screen. It may take a while to download and install.

For Raspberry Pi Zero and Raspberry Pi 2/3/4B:

```
curl -sL https://github.com/Seeed-Studio/grove.py/raw/master/install.sh |
sudo bash -s -
```

If this does not work for any reason, try this command:

`sudo pip3 install grove.py` **(Enter)**

Connect the sensor to the HAT. It is an analog sensor, so it has to be connected to a socket marked 'A_'. On the Raspberry Pi Zero HAT we can use any of the sockets marked A0, A2 or A4. There is probably a wider choice of 'A_' sockets on the Raspberry Pi Base HAT for the other models of Pi. The HATs convert the analog signal from the sensor into a digital signal which can be understood by the Pi, and potentially used to trigger a valve to water an area of soil until a defined level of moisture is reached. For our purposes, we are just going to display the readings

Note that, in the software, we must state the socket to which the sensor is connected. Socket A0 = 0; socket A2 = 2, and socket A4 = 4 on Raspberry Pi Zero HAT.

To find all the Grove software, issue this command:

`cd /usr/local/lib/python3.7/dist-packages` **(Enter)**

If we then issue the command `ls` in this directory, we will see the associated software. By typing the following, we can look inside the `grove` directory:

`cd grove` **(Enter)**
`ls` **(Enter)**

Here we find the software for a wide range of Grove sensors. The software for our sensor is based on that for a similar sensor listed as `grove_moisture_sensor.py` only that one is a different, two-pronged resistive sensor. The readings from our capacitive sensor simply indicate the approximate opposite to those in the original software, so that has been accounted for in the software below.

Next we need to `cd` **(Enter)** to go back to the 'home' terminal.

The next step is to install the software found on the next pages. To do that, we proceed as for the DHT software – by opening a new file in the terminal using a name we have chosen for it, with the `.py` extension:

`nano moist.py`

….. and then copying and pasting the code given below:

```
# Working Moisture Sensor on GrovePi HAT Port A4
#!/usr/bin/env python -*- coding: utf-8 -*-
#
# The MIT License (MIT)
#
# Grove Base Hat for the Raspberry Pi, used to connect grove sensors.
# Copyright (C) 2018  Seeed Technology Co.,Ltd.
# Use this command to start: python3 moist.py 4  where 4 = Port A4 on HAT
'''
This is the code for
    - Grove - Moisture Sensor <https://www.seeedstudio.com/Grove-Moisture-Sensor-p-955.html>`_

Examples:

    .. code-block:: python

        import time
        from grove.grove_moisture_sensor import GroveMoistureSensor

        # connect to analog pin 4(slot A4) - Can be any of A0, A2, A4
        PIN = 4

        sensor = GroveMoistureSensor(PIN)

        print('Detecting moisture...')
        while True:
            m = sensor.moisture
            if 0 <= m and m < 1400:
                result = 'Wet'
            elif 1400 <= m and m < 1800:
                result = 'Moist'
            else:
                result = 'Dry'
            print('Moisture value: {0}, {1}'.format(m, result))
            time.sleep(5)
'''
# Code for GrovePi Zero HAT with Grove Capacitive Moisture Sensor on Port A2 with non-Grove
1602LCD

import math
import sys
import time
import signal # Needed for the Cleaning Up messages at press of CTRL+C
import os # To help tidy up
from grove.grove_moisture_sensor import GroveMoistureSensor
from grove.adc import ADC
import adafruit_character_lcd.character_lcd as characterlcd # Import LCD library

import board # LCD Hardware detection
import digitalio # LCD Software


PIN = 4
sensor_name = GroveMoistureSensor(PIN)

lcd_rs = digitalio.DigitalInOut(board.D25) #RS of LCD is connected to GPIO 25 on Pi
lcd_en = digitalio.DigitalInOut(board.D24) #EN of LCD is connected to GPIO 24 on Pi
lcd_d7 = digitalio.DigitalInOut(board.D22) #D7 of LCD is connected to GPIO 22 on Pi
lcd_d6 = digitalio.DigitalInOut(board.D18) #D6 of LCD is connected to GPIO 18 on Pi
lcd_d5 = digitalio.DigitalInOut(board.D17) #D5 of LCD is connected to GPIO 17 on Pi
lcd_d4 = digitalio.DigitalInOut(board.D23) #D4 of LCD is connected to GPIO 23 on Pi
#lcd_backlight = 2 #If LCD is mono, i.e. colours cannot be changed, do not assign a GPIO pin
to backlight pin
```

```
lcd_columns = 16  # for 16*2 LCD
lcd_rows    = 2  # for 16*2 LCD

# Define and activate lcd and its pins
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_rows)


__all__ = ["GroveMoistureSensor"]

class GroveMoistureSensor:
    '''
    Grove Moisture Sensor class

    Args:
        pin(int): number of analog pin/channel the sensor connected.
    '''

    def __init__(self, channel):
        self.channel = channel
        self.adc = ADC()

    @property
    def moisture(self):
        '''
        Get the moisture strength value/voltage

        Returns:
             (int): voltage, in mV
        '''
        value = self.adc.read_voltage(self.channel)
        return value

Grove = GroveMoistureSensor

def signal_handler(signal, frame):
    print("Cleaning up!")
    lcd.clear()
    lcd.message = ("Cleaning Up!")
    time.sleep(3.0)
    lcd.clear()
    exit()


def main():
    from grove.helper import SlotHelper
    sh = SlotHelper(SlotHelper.ADC)
    pin = sh.argv2pin()

    sensor = GroveMoistureSensor(pin)
    print('Detecting moisture...')
    lcd.message = ('Detecting...')
    time.sleep(5.0)
    lcd.clear()

    signal.signal(signal.SIGINT, signal_handler)

    while True:
        m = sensor.moisture
        if 0 <= m and m < 1400:
            result = 'Wet'
        elif 1400 <= m and m < 1800:
            result = 'Moist'
        else:
            result = 'Dry'
            print('Moisture value: {0}, {1}'.format(m, result))
            lcd.clear() # Clear the LCD screen
            time.sleep(2.0) # Gives a break with a clear screen between readings
            lcd.message = ('Value:{0}={1}'.format(m, result))  # Display the findings of the
sensor
            time.sleep(5.0) # Wait for X sec then update the values (at least 4 seconds is
recommended)

if __name__ == '__main__':
    main()
```

We start the program with the command:

`python3 moist.py 4` **(Enter)** …. where the number 4 stands for Port A4 on the GrovePi HAT

Again, to stop the program, hold down the CTRL key and press **(Enter)**.

We can try out the response of the sensor by putting the lower half of it in a glass of water, or by sticking it in some soil.

To change the timing, parameters or text of the readings, use the nano text editor to open the software:

`sudo nano moist.py` **(Enter)**

…. and make any changes required.

8. Use your Pi LCD as a streaming news display

This service makes use of the RSS feed from BBC News. We must acknowledge this source when using it, although any other RSS feed may be used to replace it – e.g. feeds from other news sources, from weather services or from websites. RSS feeds may be found simply by using any search engine. The web address of any feed needs to be processed by a Python module named 'feedparser', so that should be installed from the terminal of the Raspberry Pi using the command below:

`sudo pip3 install feedparser` **(Enter)**

If an alternative newsfeed is chosen, then simply replace the address of the example in line 41 of the code with that of the alternative.

Also notice that the same software as for the DHT and moisture sensors is used for detection and control of the LCD and GPIO pins.

We can then create a file for the software in the terminal named news.py, and copy and paste the software into it before saving it (see page 11):

`sudo nano news.py` **(Enter)**

Here is the code. Please be aware that it is here to be modified as desired. The limitation of the LCD is that it cannot display more than 40 scrolling characters per line, so the line break may not occur in an ideal position if the headline contains more than 40 characters (including spaces). Some characters may not be displayed if the headline is more than 80 characters, but this does not often happen.

```
# Program to read BBC RSS newsfeed
# The LCD backlight is not controlled in this program
# Ensure you have installed the adafruit_character_lcd library
# Ensure that you have installed digitalio library
# Ensure that the pin connections on your RPi are correct
# Ensure that you have installed the feedparser module
# Start with command: python3 news.py

#!/usr/bin/python3

import os
from time import sleep
import re
import sys
import feedparser
import time # For 'sleep' commands
```

```
import adafruit_character_lcd.character_lcd as characterlcd
import board # LCD Hardware detection
import digitalio # LCD Software
lcd_rs = digitalio.DigitalInOut(board.D25) #RS of LCD is connected to GPIO 25 on Pi
lcd_en = digitalio.DigitalInOut(board.D24) #EN of LCD is connected to GPIO 24 on Pi
lcd_d7 = digitalio.DigitalInOut(board.D22) #D7 of LCD is connected to GPIO 22 on Pi
lcd_d6 = digitalio.DigitalInOut(board.D18) #D6 of LCD is connected to GPIO 18 on Pi
lcd_d5 = digitalio.DigitalInOut(board.D17) #D5 of LCD is connected to GPIO 17 on Pi
lcd_d4 = digitalio.DigitalInOut(board.D23) #D4 of LCD is connected to GPIO 23 on Pi
#lcd_backlight = 2 #If LCD is mono, i.e. colours cannot be changed, do not assign a GPIO pin
to backlight pin

lcd_columns = 16  # for 16*2 LCD

lcd_rows    = 2  # for 16*2 LCD

# Define and activate lcd and its pins
lcd = characterlcd.Character_LCD_Mono(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
lcd_columns, lcd_rows)

lcd.message = 'BBC News on RPi \n- Streaming now ' # Give an intro message
print ('BBC News Headlines') # Intro on monitor
time.sleep(6.0) # wait for X secs
lcd.clear()

# Create object bbc to store the bbc's rss feed data

d = feedparser.parse("http://newsrss.bbc.co.uk/rss/newsonline_uk_edition/front_page/rss.xml")

# Get x number of headlines from feed data and print to monitor.
for i in range(2):
    s = (d['entries'][i]['title']) # Define string as 's'
    print(s)

# Print "BBC News" to the first line of the LCD.
lcd.message = 'BBC News'
sleep(4.0)
lcd.clear()

# Create a loop to print the contents of the split object to the LCD screen.

try:
    while True:

        for i in range(1): # Works best if you scroll only the first headline.
            w = 80 # Create an object called 'w' and use that to save character chunks of the
RSS feed. Best with number of characters in headline.
            s = (d['entries'][i]['title']) # Define string of incoming news.
            s = f'{s: <{w}}' # Format string, with left justification.
            s = ''.join(s[i:i+80] for i in range(0, len(s), 80)) # Ensure string chunks are x
characters long, joined with a space.
#            s = ((re.sub('([A-Z])', r' \1', s,))) # String 's', putting a space before every
capital letter. Optional.

            lcd.move_left() # Direction of scroll, one character at a time.

            if (len(s)>16): # If the length of the string is greater than 16 characters ...
                a,b = s[:40],s[40:] # Divide it into 2 parts, a & b, each of half the width of
the feed.
                headline = "{:s}\n{:s}".format(a,b)
                lcd.message = headline
            else:
                lcd.message = (s) # If headline has fewer than 16 characters, its string will
not be split in half.

            sleep(0.3) # Controls speed of headline scroll. Increase sleep to slow it down.

except KeyboardInterrupt: # If there is a KeyboardInterrupt by pressing Ctrl+C, exit the
program and cleanup.
    print("Cleaning up!")
    lcd.clear()
    lcd.message = "Goodbye!"
    sleep(3.0)
    lcd.clear()
    quit()
```

9. Use Your Pi LCD with other sensors

If you wish to use your Raspberry Pi with other sensors, you will need to decide whether you are going to connect them to the Pi using the Grove HAT or the GPIO pins.

If you use the Grove HAT's ports, you will need to consult the GrovePi software to determine which port to use, then adapt the software to your own needs. The software is already on your Pi at:

`cd /usr/local/lib/python3.7/dist-packages` **(Enter)**

Then use the `nano` editor to modify it, if necessary.

If you use the GPIO pins, you should pay careful attention to how the sensor should be connected – it is usually best to buy a breadboard to wire all the components correctly to the Pi, as we did with the DHT monitor. You should also ensure that the sensor cannot send data using more than 3V to the Pi in order to avoid damaging the Pi's circuitry. A resistor of the correct value will usually be needed. If you have an analog sensor and you are using the Pi's GPIO pins, you will also need an ADC device to convert Analog to Digital signals that the Pi can process. You can then find and download the Adafruit software for your sensor, as we did for the DHT monitor, and adapt it as required. To use the LCD with other sensors, pay attention to the Python modules from Adafruit which we imported in all the software described earlier.

We hope you have enjoyed using this guide, and that you have found it useful.

Look out for more Use Your Pi guides in the future!