**Good Places to Look for information/sensors**

* Adafruit.com
* Atlas Scientific
* Arrow.com
* Apogee Instruments
* Campbell Scientific
* Raspberrypi.org

**Good Sensors**

For any sensor, a good place to start is Adafruit or the manufacturer’s website if purchased elsewhere. If you search for sensor datasheets, you can usually find information about their margin of error, drift, and other technical details. Usually datasheets are located on the manufacturer’s site or by searching on Google for “sensor name” and “datasheet”.

* DS18B20
  + Easy digital temperature sensor
  + Fairly accurate and I’ve noticed minimal drift when using
  + Easy to program
  + Comes in waterproof version and because it uses one-wire protocol a virtually unlimited number can be connected with unique ID’s
    - [Waterproof Version](https://www.adafruit.com/product/381)
    - [Regular Version](https://www.adafruit.com/product/374)
  + Easy tutorials for using the sensor are located on Adafruit.com (same as the above links)
  + If using multiple of these sensors I would recommend finding the unique ID for the sensor (easily done with the tutorials) and then label that ID directly on the sensor with a piece of tape to keep track of where each measurement is coming from
  + The sensor will require a 4.7K resistor like this one. If you need more information on how resistors are color coded [this](https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code) and [this](https://www.electronics-tutorials.ws/resistor/res_2.html) are good references. There should be plenty out at 620 in one of the drawers.
  + [Tutorial](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing)
* BME280
  + Temperature, humidity, and air pressure sensor
  + Best used for atmospheric measurements
  + For accurate air temperature, the sensor must be shielded from the sun
  + [Tutorial](https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/overview)
  + There are upgraded versions of this as well such as the BME680 which can be found on Adafruit.
* DS3231
  + This is an Real Time Clock (RTC) clock module
  + Most microprocessors and microcontrollers do not have built in clocks and thus rely on the internet to keep an accurate time. An RTC is necessary if you build your functions based on measuring something at a regular interval and it isn’t always connected to the internet.
  + [Tutorial](https://learn.adafruit.com/adding-a-real-time-clock-to-raspberry-pi)
* INA260
  + Voltage, current, and power sensor
  + Used to measure power consumed by a DC RABR motor
  + [Tutorial](https://learn.adafruit.com/adafruit-ina260-current-voltage-power-sensor-breakout)
* OLED Displays
  + There’s an enormous variety of screens depending on the amount of information you need to display. A screen isn’t necessary but it can help if you’re goal is real time information without having to connect to the Raspberry Pi directly. I’ve never used a screen, but it might be something you look at.
  + [Overview](https://learn.adafruit.com/monochrome-oled-breakouts)
* Atlas Scientific
  + Has a variety of sensors including pH and DO
  + I haven’t used them before so I don’t know how reliable they would be or how to set one up but I know the site has a variety of tutorials on using them.
  + The main challenge on using any probe like sensor such as pH or DO will be fouling of the tip in the wastewater and you’d need to find a solution to that.
  + [Link](https://atlas-scientific.com/#)
* Apogee Instruments
  + Has a variety of light sensors depending on the accuracy and connection type you need and can be visited [here](http://apogeeinstruments.com).
  + The PAR sensor I’ve used from them in the past is the SQ-520 USB version which is pretty straightforward to use. More information is [here](https://www.apogeeinstruments.com/sq-520-full-spectrum-smart-quantum-sensor-usb/).
  + I heard they made a couple versions of sensors that can connect digitally now so that may be an option but I’m not sure how to use them or which would be best.
* Campbell Scientific
  + There’s really too much to write about all of the sensors and how to use them here but I’ll write a few key points. The rest will be learned easiest by visiting their [site](https://www.campbellsci.com/).
  + The Campbell CR1000 are the standard dataloggers they sell and and more information and shortcuts can be found [here](https://www.campbellsci.com/cr1000).
  + The simplest way to use it is the PC200W software which is free to download and connect to the datalogger via USB.
  + There are a bunch of different sensors available which can be found on there site.
* Licor
  + Most of the photosynthetically active radiation (PAR) sensors we have in the lab are LI-190R and can be connected to the Campbell Datalogger. These are highly accurate but need to be calibrated. Apogee Instruments offers this service [here](https://www.apogeeinstruments.com/recalibration-and-repairs/).
  + [Licor Site](https://www.licor.com/env/)
  + I also saw a way someone connected one of these sensors to an Arduino but don’t know how difficult it is. The paper is linked [here](https://academic.oup.com/treephys/article/34/6/640/2338115).
  + I also saw a different PAR sensor advertised as connecting to the Raspberry Pi [here](https://www.ufire.co/par/).

**Raspberry Pi Basics**

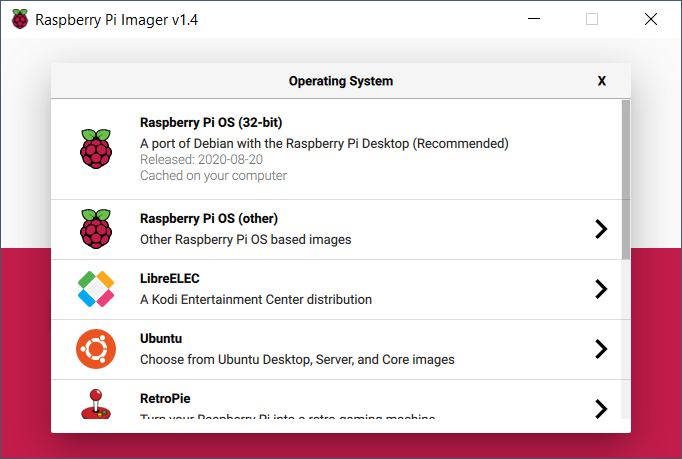
* Easy to use microprocessor. There are a few versions out there but for the project I’d recommend using the Raspberry Pi 3, 3B+, or Zero W. The Raspberry Pi 4 can do more powerful tasks but is also unnecessary for simple recording of sensors.
* Depending on setup and power supply can have some reliability issues
* Runs on Linux (if using the main image)
* Basic setup guide
  + [Link](https://projects.raspberrypi.org/en/pathways/getting-started-with-raspberry-pi)
  + [Link](https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up)
* Some Linux commands that you’ll use frequently
  + [Link](https://www.raspberrypi.org/documentation/linux/usage/commands.md)
* General usage guide
  + [Link](https://projects.raspberrypi.org/en/projects/raspberry-pi-using/11)
* Here’s another series of links with videos as well. They go through the entire process of setup to programming with a sensor.
  + https://pythonprogramming.net/introduction-raspberry-pi-tutorials/
  + https://pythonprogramming.net/remote-access-raspberry-pi-tutorials/?completed=/introduction-raspberry-pi-tutorials/
  + https://pythonprogramming.net/terminal-navigation-raspberry-pi-tutorials/?completed=/remote-access-raspberry-pi-tutorials/
  + <https://pythonprogramming.net/camera-module-raspberry-pi-tutorials/?completed=/terminal-navigation-raspberry-pi-tutorials/>
  + <https://pythonprogramming.net/gpio-raspberry-pi-tutorials/?completed=/camera-module-raspberry-pi-tutorials/>
  + <https://pythonprogramming.net/gpio-input-raspberry-pi-tutorials/?completed=/gpio-raspberry-pi-tutorials/>
* As for storing data there a a variety options. The two best options will be saving values to a USB drive or to the cloud through an interface such as Google Sheets. No matter the method I would recommend saving the files as a CSV or similar spreadsheet format for easy processing later.
  + Google Sheet method tutorial on [Medium](https://medium.com/@greg.business.automated/sending-raspberry-pi-data-to-google-sheets-for-low-code-people-the-easy-way-34f9bbe2ce47)
  + Google Sheet method tutorial on [SB Components](https://shop.sb-components.co.uk/blogs/posts/log-raspberry-pi-data-to-google-sheets)
  + Google Sheet method tutorial on [What I Made](http://www.whatimade.today/log-sensor-data-straight-to-google-sheets-from-a-raspberry-pi-zero-all-the-python-code/)
* When using a USB for storage there are two routes. You can mount a USB to the Pi which essentially means the USB ID will stay constant making it easier to reference as a filepath in programs. If you’ll be removing the USB a lot then you’ll be better off leaving the drive unmounted and adjusting the code accordingly. I should mention that these methods have been changing a bit with the latest Raspberry Pi releases and thus may need to be adjusted by searching through internet forums.
  + [Mounting a Drive](https://www.instructables.com/Mounting-a-USB-Thumb-Drive-with-the-Raspberry-Pi/)

**Other Miscellaneous Resources**

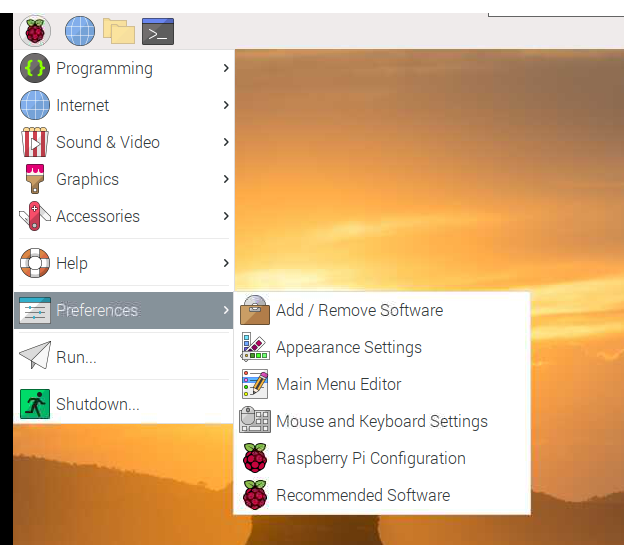
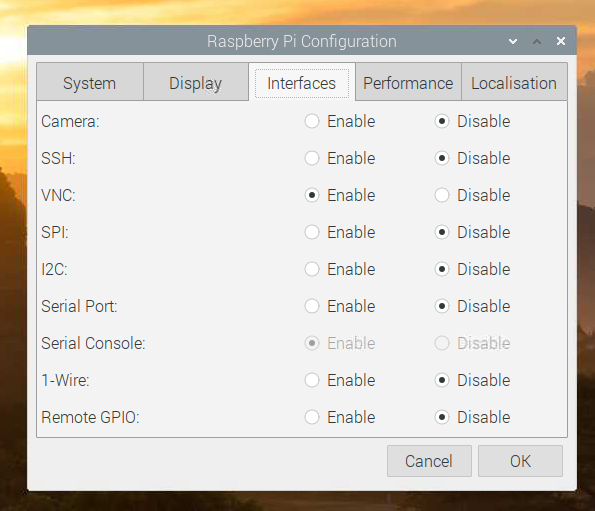
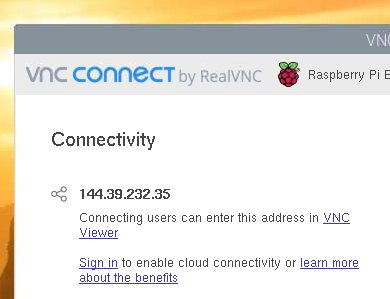
* Google Drive folder with photos of some of the previous assemblies used for monitoring RABRs
  + [Link](https://drive.google.com/drive/folders/1EU68ro3ITDvLr8CerluWeiLOEsewMyye?usp=sharing)
* Tutorial on soldering. You will come across times that soldering is required to connect parts. I would recommend going to the Idea Factory on campus to learn as they will be the most knowledgeable, but here are a couple guides as well.
  + [Idea Factory Tutorial](https://engineering.usu.edu/students/idea-factory/users/training-materials)
  + [Online Tutorial](https://www.makerspaces.com/how-to-solder/)
* 3D Printing
  + You may decide that 3D printing is a good way to make custom sensor mounts and other devices. I would recommend visiting the Idea Factory to learn more as well. Some general tips that might help are below.
    - When picking a material make sure to get 1.75 mm diameter as it works with most printers.
    - The 3 most common materials are PLA, PETG, and ABS. A quick Google search or visiting the Idea Factory can help you decide what type is best. [Here](https://all3dp.com/1/3d-printer-filament-types-3d-printing-3d-filament/) is another resource that can come in handy.
* USU IDEA factory and Metal Factory
  + These will be the best resources for learning about and using equipment such as laser cutters, PCB printing machines, and a variety of equipment that may come in handy for the project.

**First Setup**

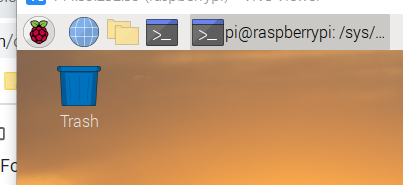
**Setup with Raspberry Pi OS using company tutorial**

* <https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2>
* 
* Select main Raspberry Pi OS 32-bit to add to SD card
* After setting up SD card connect keyboard, mouse, HDMI, and SD card to Raspberry Pi
* Next plug the Pi into power and wait to boot up
  + Walk through menus and setup as needed
  + Both Pi logins are currently
    - Username: Pi
    - Password: Raspberry

**Setup VNC Server on Pi**

* 
* Click in Pi icon in top left corner → Preferences → Raspberry Pi Configuration
* 
* Under the Interfaces tab enable VNC and click OK
* Reboot Raspberry Pi (use command: “sudo reboot”)
* 
* Once PI is rebooted click on VNC icon in top right corner of screen
* 
* Under the connectivity is the IP address. Download VNC server on your computer and use this number to connect to Pi without need for a display, keyboard, and mouse.
  + <https://www.realvnc.com/en/connect/download/vnc/>
* Default username and password:
  + Username: pi
  + Password: raspberry

**Temperature Sensor (DS18B20)**

* The guide I’m referencing to setup the DS18B20 is below
  + <https://learn.adafruit.com/adafruits-raspberry-pi-lesson-11-ds18b20-temperature-sensing>
  + <https://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/>
* I will be making some changes to the setup as well as using my own code so I’ll note the main steps below too as well as a wiring diagram.
* Enable 1-wire on the Pi (this will be under the same Interfaces are used to enable VNC above)
* To find the ID of each sensor attached go to the command prompt
  + 
  + Enter **sudo modprobe w1-gpio**
  + Enter **sudo modprobe w1-therm**
  + Enter **cd /sys/bus/w1/devices**
  + Enter **ls** to view connected devices
  + Connected sensors will show up with a 28- then random numbers and letters
  + To view the raw data from a sensor enter **cd 28-XXXXXXXXXXXX** (change the X’s to your own address)
  + Enter **cat w1\_slave**
  + The temperature is the value of t=20875 (or whatever yours shows) with that meaning a temperature of 20.875 C
* Code examples for the sensors are found below

**Main Temperature File**

| #Temperature.py (put in same folder as other files) #Libraries needed import os import glob import time  #Connects to pins of Pi os.system('modprobe w1-gpio') os.system('modprobe w1-therm')  def read\_temp(device):  #Finds device directory  base\_dir = '/sys/bus/w1/devices/'  device\_folder = glob.glob(base\_dir + device)[0]  device\_file = device\_folder + '/w1\_slave'    #Reads raw data from sensor  f = open(device\_file, 'r')  lines = f.readlines()  f.close()    #Converts raw data to Celcius and Farenheit  while lines[0].strip()[-3:] != 'YES':  time.sleep(0.2)  lines = read\_temp\_raw()  equals\_pos = lines[1].find('t=')  if equals\_pos != -1:  temp\_string = lines[1][equals\_pos+2:]  temp\_c = float(temp\_string) / 1000.0  temp\_f = temp\_c \* 9.0 / 5.0 + 32.0  #Returns a tuple with Celcius and Farenheit temperatures  return temp\_c, temp\_f |
| --- |

**Example for read from another file**

| from Temperature import read\_temp as temp  temp\_c1, temp\_f1 = temp('28-041685b81aff') #Use id of sensor attached as string print('Sensor 1 Temp: '+str(round(temp\_c1,2))+'°C and '+str(round(temp\_f1,2))+'°F.')  temp\_c2, temp\_f2 = temp('28-0416a450abff') #Use id of sensor attached as string print('Sensor 2 Temp: '+str(round(temp\_c2,2))+'°C and '+str(round(temp\_f2,2))+'°F.') |
| --- |

**Temperature + Humidity + Pressure (BME280)**

* The reference for this section is below
  + <https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/pinouts>
  + <https://learn.adafruit.com/adafruit-bme280-humidity-barometric-pressure-temperature-sensor-breakout/python-circuitpython-test>
* Wiring guide
* Install library for the sensor in the command line
  + **sudo pip3 install adafruit-circuitpython-bme280**
* Enable an I2C connection under the interfaces tab like above

**Example Code for BME280 Functions**

| #Temp\_Hum\_Press.py #Functions to read values #Import libraries import board import busio import adafruit\_bme280 #Initializes connection i2c = busio.I2C(board.SCL, board.SDA) bme280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c)  #Reads temperature and returns tuple of values def temp\_BME280():  temp\_c = bme280.temperature  temp\_f = temp\_c \* 9.0 / 5.0 + 32.0  return temp\_c, temp\_f  [**Connecting to Google Sheets**](#_hbm7ltk76rqq) **11**  [**Crontab for Scheduling Tasks**](#_8mk5rksqxwa4) **16**  #Reads humidity def hum\_BME280():  hum = bme280.humidity  return hum  #Reads pressure def press\_BME280():  press = bme280.pressure  return press |
| --- |

**Example code to read from BME280 functions**

| from Temp\_Hum\_Press import temp\_BME280, hum\_BME280, press\_BME280  temp\_c\_BME280, temp\_f\_BME280 = temp\_BME280() print('BME280 Temp: '+str(round(temp\_c\_BME280,2))+'°C and '+str(round(temp\_f\_BME280,2))+'°F.')  hum\_BME280() print('BME280 Hum: '+str(round(hum\_BME280,2))+'%.')  press\_BME280 = press\_BME280() print('BME280 Press: '+str(round(press\_BME280,2))+'hPa.') |
| --- |

**Wiring Diagram**

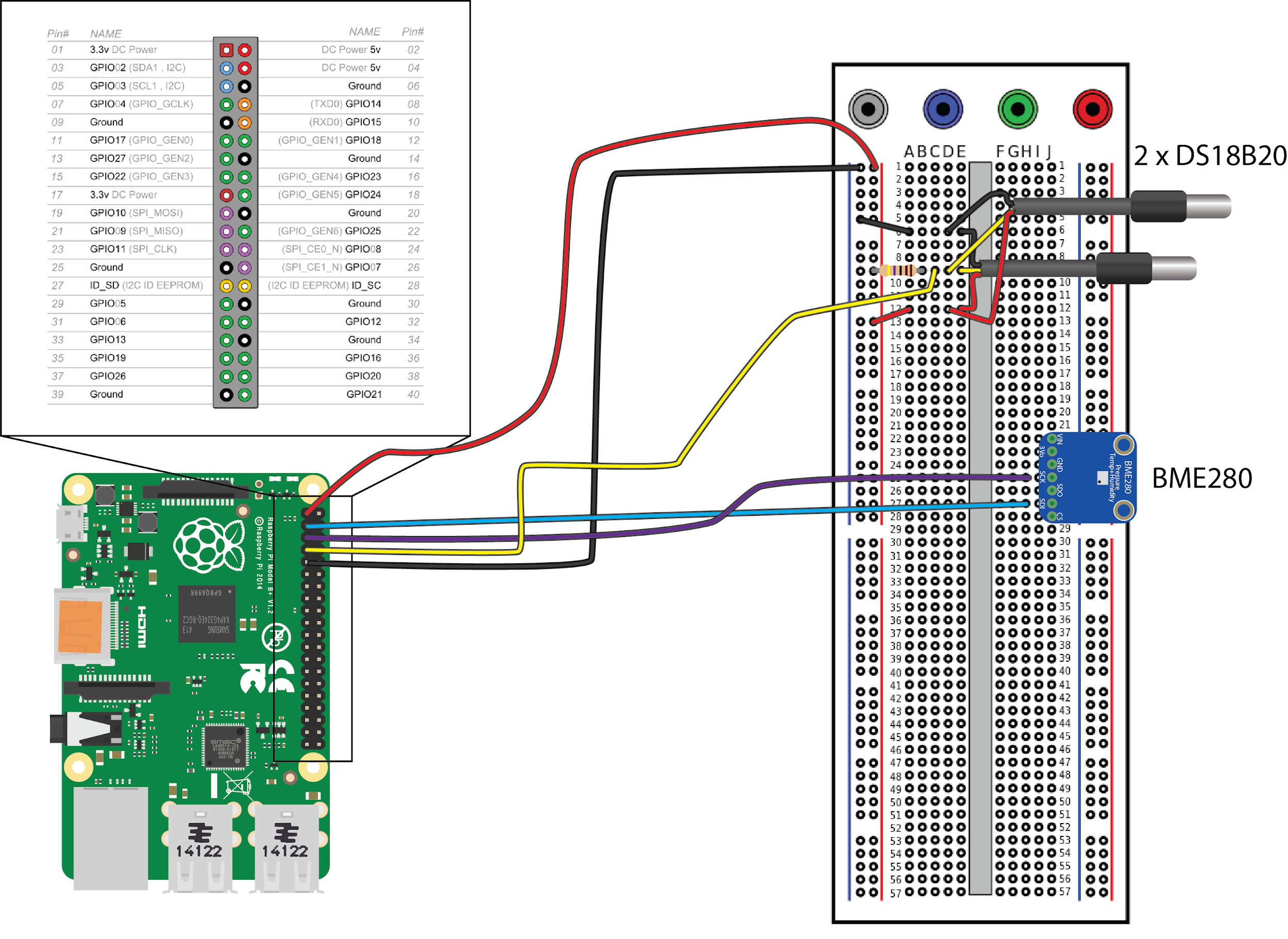
This is included at the bottom of the document ([Link](https://drive.google.com/file/d/1DofkMGtcemqyLohKFlj1EUR4nEt1CfkA/view?usp=sharing))

**Other Sensors**

I’ll leave some other sensors if you want to try others out and learn more.

**TSL2591** (<https://learn.adafruit.com/adafruit-tsl2591>)

**DS3231** (<https://learn.adafruit.com/adafruit-ds3231-precision-rtc-breakout>)



## Connecting to Google Sheets

The Google Sheets script is built on the gspread python API. Most general questions will be in the documentation at:

<https://gspread.readthedocs.io/en/latest/>

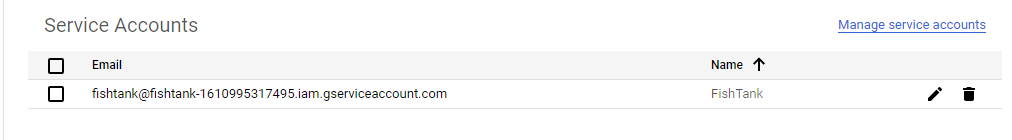
To authenticate the script on the Raspberry Pi there are two main steps.

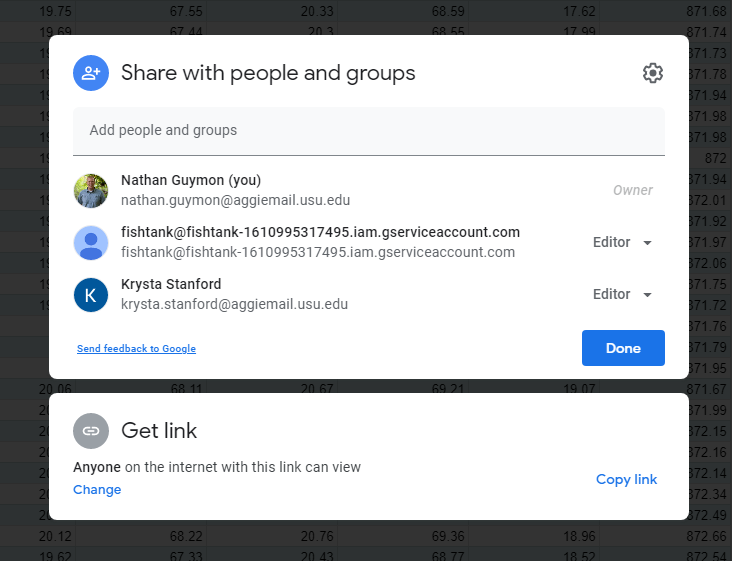
1. Enable API Access to Google Drive
2. Create credentials for the script to access the Google Sheet

Both of these steps are covered and updated in the documentation at:

<https://gspread.readthedocs.io/en/latest/oauth2.html#for-bots-using-service-account>

Also, after following the above steps the Google Sheet will need to be shared with the service account email.



****

Here’s the two scripts I’ve been using for recording data.

**Sensor Script:**

| # Import Libraries import board import busio import adafruit\_bme280 import os import glob import time  #Initializes connection to BME280 i2c = busio.I2C(board.SCL, board.SDA) bme280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c)  #Reads temperature and returns tuple of values def temp\_BME280():  temp\_c = bme280.temperature  temp\_f = temp\_c \* 9.0 / 5.0 + 32.0  return temp\_c, temp\_f  #Reads humidity def hum\_BME280():  hum = bme280.humidity  return hum  #Reads pressure def press\_BME280():  press = bme280.pressure  return press  #Connects to pins of Pi os.system('modprobe w1-gpio') os.system('modprobe w1-therm')  # Function to read the temperature sensor by inputting the sensor ID def read\_temp(device):  #Finds device directory  base\_dir = '/sys/bus/w1/devices/'  device\_folder = glob.glob(base\_dir + device)[0]  device\_file = device\_folder + '/w1\_slave'    #Reads raw data from sensor  f = open(device\_file, 'r')  lines = f.readlines()  f.close()    #Converts raw data to Celcius and Farenheit  while lines[0].strip()[-3:] != 'YES':  time.sleep(0.2)  lines = read\_temp\_raw()  equals\_pos = lines[1].find('t=')  if equals\_pos != -1:  temp\_string = lines[1][equals\_pos+2:]  temp\_c = float(temp\_string) / 1000.0  temp\_f = temp\_c \* 9.0 / 5.0 + 32.0  #Returns a tuple with Celcius and Farenheit temperatures  return temp\_c, temp\_f |
| --- |

**Google Sheet Script:**

| # Import libraries import os import sys import time import datetime   import board import gspread from oauth2client.service\_account import ServiceAccountCredentials  from datetime import datetime  from Sensors import temp\_BME280, hum\_BME280, press\_BME280 from Sensors import read\_temp as temp  # Tells the script to look in the Documents folder for related files sys.path.append('/home/pi/Documents')  # datetime object containing current date and time now = datetime.now()  # Convert the now datetime to a string dt\_string = now.strftime("%Y-%m-%d %H:%M:%S")  # Read in values from sensors temp\_c1, temp\_f1 = temp('28-0516a40d6bff') temp\_c1 = round(temp\_c1,2) temp\_f1 = round(temp\_f1,2)  temp\_c\_BME280, temp\_f\_BME280 = temp\_BME280() temp\_c\_BME280 = round(temp\_c\_BME280,2) temp\_f\_BME280 = round(temp\_f\_BME280,2)  hum\_BME280 = round(hum\_BME280(),2)  press\_BME280 = round(press\_BME280(),2)  # Set the json target (change to yours) GDOCS\_OAUTH\_JSON = 'fishtank-main.json'  # Set the spreadsheet target (change to yours) GDOCS\_SPREADSHEET\_NAME = 'FishTank'  # Set the sheet target (change to yours) GDOCS\_SHEET\_NAME = 'Sheet1'  def login\_open\_sheet(oauth\_key\_file, spreadsheet, sheet):  """Connect to Google Docs spreadsheet and return the first worksheet."""  try:  scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']  credentials = ServiceAccountCredentials.from\_json\_keyfile\_name(oauth\_key\_file, scope) # Sets the credentials  gc = gspread.authorize(credentials) # Authorizes the script with your account  sh = gc.open(spreadsheet) # Open worksheet  worksheet = sh.worksheet(sheet) # Open sheet  return worksheet  except Exception as ex: # If unable to access Google Sheet prints an error statement to help debug  print('Unable to login and get spreadsheet. Check OAuth credentials, spreadsheet name,\n'  'and make sure spreadsheet is shared to the client\_email address in the OAuth .json file!')  sys.exit(1)   worksheet = login\_open\_sheet(GDOCS\_OAUTH\_JSON, GDOCS\_SPREADSHEET\_NAME, GDOCS\_SHEET\_NAME) # Calls on the sheet function  try:  worksheet.append\_row((dt\_string, temp\_c1, temp\_f1, temp\_c\_BME280, temp\_f\_BME280, hum\_BME280, press\_BME280)) except:  print('Append error, logging in again')  worksheet = None  #Formats the first fow to ensure it stays datetime format worksheet.format('A1:A1000', {'numberFormat':{'type': 'DATE\_TIME'}}) worksheet.format('A1:A1000', {'numberFormat':{'type': 'DATE\_TIME', 'pattern': 'yyyy-mm-dd hh:mm:ss'}}) |
| --- |

## Crontab for Scheduling Tasks

Cron is a tool in Linux for scheduling commands or scripts to be run. I’d recommend using it for running the datalogger scripts on the Raspberry Pi. The best method I found is to make a shell script to run the file then create the crontab task.

<https://www.raspberrypi.org/documentation/linux/usage/cron.md>

Here’s the basic steps:

* Go to directory where the other python files are
  + **cd *path/to/directory***
* Create a shell script (format as ***sudo nano nameofscript.sh***)
  + **sudo nano GoogleSheets.sh**
* Make it executable
  + **sudo chmod +x GoogleSheets.sh**
* Here is an example of a shell script to use:

| #!/bin/bash cd /home/pi/Desktop python3 GoogleSheets.py |
| --- |

* Next add it to the crontab
  + **crontab -e**
* Edit the bottom of the crontab with the below (runs every 15 minutes)
  + **\*/15 \* \* \* \* /bin/bash /home/pi/Desktop/GoogleSheets.sh**