

# BUILD GUIDE

Kingswood School, Bath

Pi-Oxide

An easy to use, round-the-clock carbon dioxide monitoring device for schools and businesses

2025

## BILL of MATERIALS:

ITEM	COST
Adafruit SCD-40 - True CO2, Temperature and Humidity Sensor - <a href="https://www.adafruit.com/product/5187">https://www.adafruit.com/product/5187</a>	£34.60
STEMMA QT / Qwiic JST SH 4-pin Cable with Premium Female Sockets - 150mm Long <a href="https://www.adafruit.com/product/4397">https://www.adafruit.com/product/4397</a>	£0.73
4 off 11mm M2.5 Standoffs & screws <a href="https://tinyurl.com/kdtwx5km">https://tinyurl.com/kdtwx5km</a>	£3.00
4 off M2x10mm screws <a href="https://www.componentshop.co.uk/m2-stainless-steel-pan-head-pozidrive-screw-10mm-pack-of-10.html">https://www.componentshop.co.uk/m2-stainless-steel-pan-head-pozidrive-screw-10mm-pack-of-10.html</a>	£0.50 for pack of 10
4 off M3x5mm screws <a href="https://www.westfieldfasteners.co.uk/Bolts-Screws-Metric/304-Allen-Key-Screw-M3x5mm.html">https://www.westfieldfasteners.co.uk/Bolts-Screws-Metric/304-Allen-Key-Screw-M3x5mm.html</a>	£1.35 for pack of 15
	Total: £ 65.18

## STEP 01 – SETTING UP THE HARDWARE

To begin, you need to image and successfully boot up a Raspberry Pi. There are many different tutorials online on how to do this, including the very good official documentation: <https://www.raspberrypi.com/documentation/computers/getting-started.html>

Once booted up, you can begin to attach the peripherals. Start by shutting off the raspberry pi, as it is generally unsafe to plug in peripherals with the pi still running. Then plug in the STEMMA QT cable. To do this, take the end with the 4 separate female sockets (as seen in Figure 1), and plug them in to the pins on the raspberry pi (as seen in Figure 2). Then plug the QT side into one of the sockets on either side of the SCD 40 sensor.

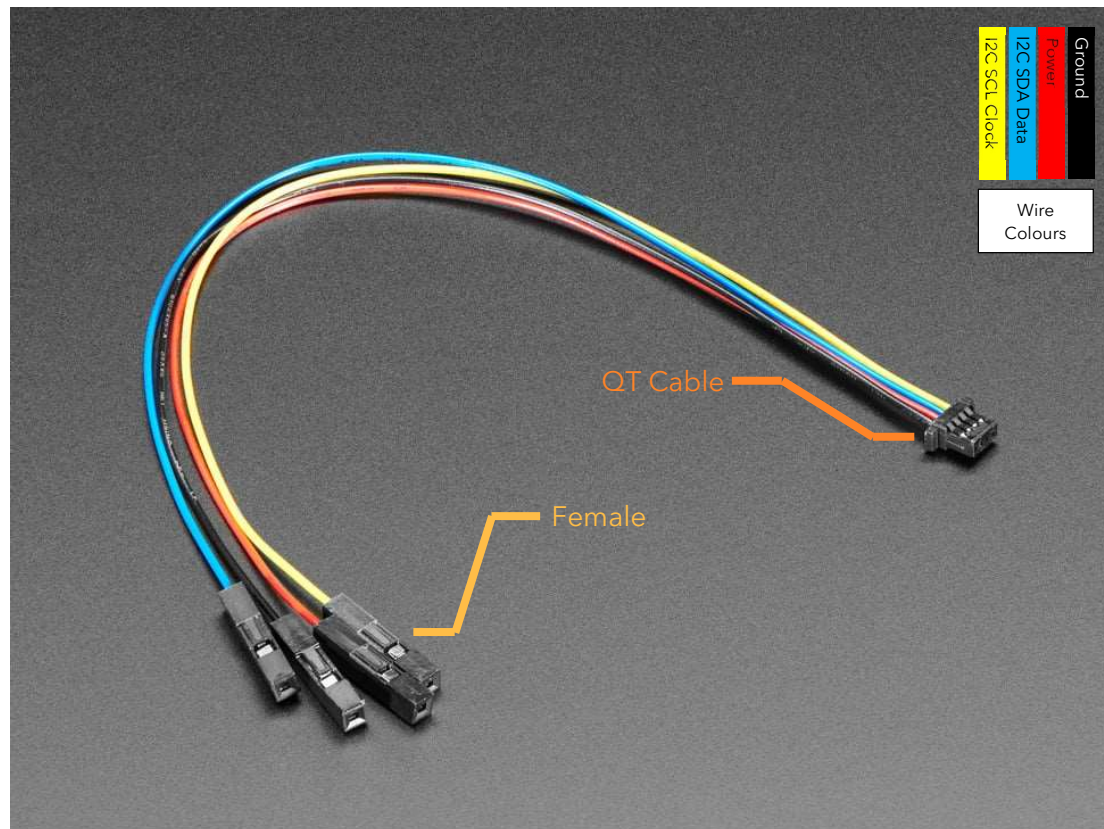
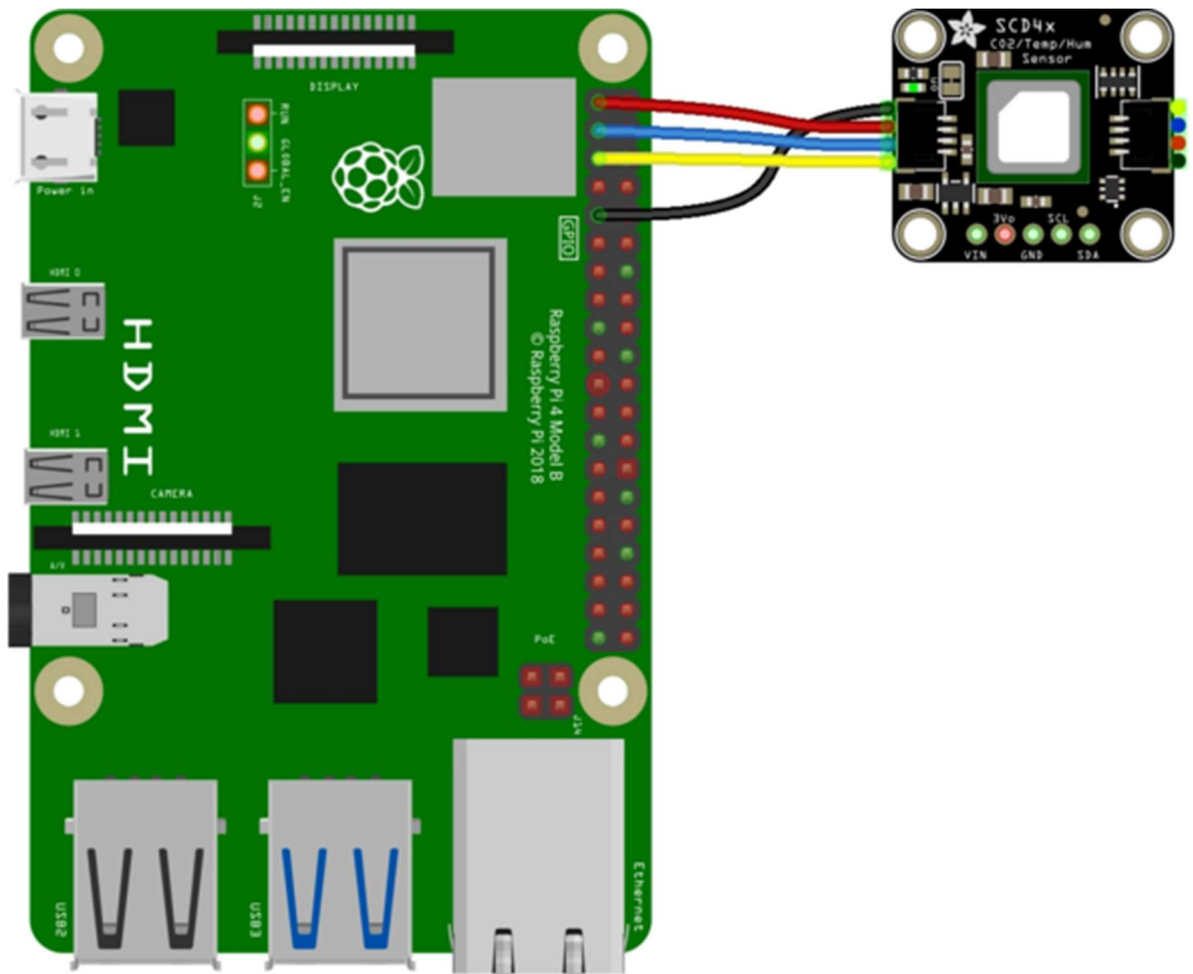


Figure 1 STEMMA QT CABLE



**Figure 2 CONNECTING THE CABLE TO THE PI**

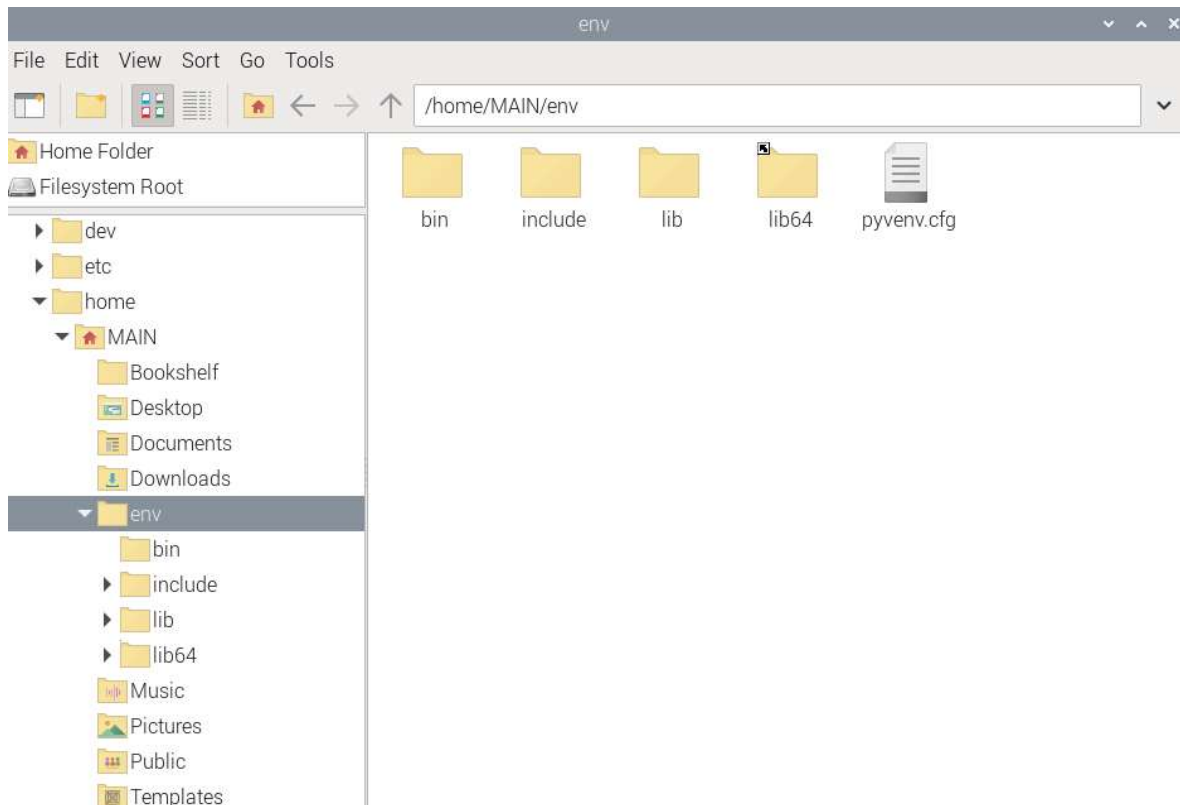
## STEP 02 – SETTING UP THE VIRTUAL ENVIRONMENT

Then you are safe to start up your raspberry pi once again, to setup the virtual environment. Start by opening the terminal and entering the following commands to create the virtual environment:

```
python3 -m venv env
```

“env” is the name of the virtual environment, it may be good to name it something else as env is used elsewhere in the pi, but for our purposes env is fine, so the tutorial will continue using env. If you named it anything else, remember to replace env with the name of your virtual environment. After it has finished (you know it has finished when the ~ \$ appears again, prompting you to enter a new input), you can check it has worked by going to the file manager. There should be a folder called the name of your virtual environment. Inside that

folder should be the a "bin", "include", "lib" and "lib64" folders, and a pyvenv.cfg file (as seen in Figure 3)



**Figure 3 The Default Virtual Environment File**

## STEP 03 – IMPORTING THE MODULES

The next step is importing the required modules. The modules we will be importing are as follows:

- Matplotlib <https://github.com/matplotlib/matplotlib>
- The Adafruit SCD4x library [https://github.com/adafruit/Adafruit\\_CircuitPython\\_SCD4X](https://github.com/adafruit/Adafruit_CircuitPython_SCD4X)
- Blinka (a dependency for the SCD4x library) [https://github.com/adafruit/Adafruit\\_Blinka](https://github.com/adafruit/Adafruit_Blinka)

First, let's install matplotlib. Open the command prompt and type in the following:

```
sudo apt install python3-matplotlib
```

Any missing dependencies are automatically installed by APT

In order to install the second 2 modules properly on the virtual environment, open a new terminal page and type

```
source env/bin/activate
```

then a (env) should appear at the beginning of the next command prompt. Then install the two modules:

```
pip3 install adafruit-circuitpython-scd4x
```

```
pip3 install Adafruit-Blinka
```

the circuitpython library does have some depreciation warnings, but these do not affect the project, so can be ignored.

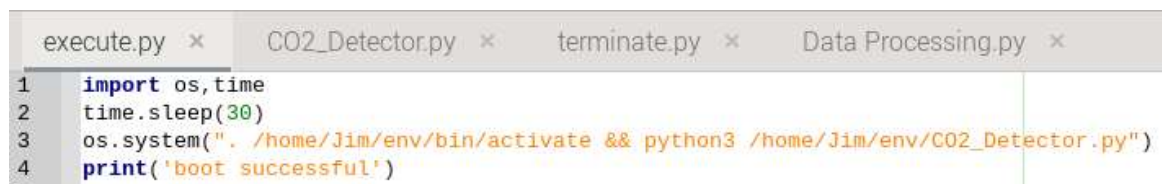
## STEP 04 – WRITING THE CODE

---

All files are stored on the following GitHub repo for reference:

<https://github.com/JimPilk/PiOxide/tree/main>

First, let's create a file to execute our main program (all these coming files will stay in the "env" folder).



```
execute.py × CO2_Detector.py × terminate.py × Data Processing.py ×
1 import os, time
2 time.sleep(30)
3 os.system("../home/Jim/env/bin/activate && python3 /home/Jim/env/CO2_Detector.py")
4 print('boot successful')
```

This script will run as soon as you boot the raspberry pi (this will be set up with a cronjob later), so it waits for a short period to allow everything to load properly and for Wi-Fi to connect before continuing. It then activates the virtual environment, and if this is successful, it starts the data collection code. Our raspberry pi was set up by Jim, hence the name in the file paths. This should be replaced with an appropriate username for your case.

Now for the data collection code. This program collects data from the sensor and stores it in a text file to be processed later. To prevent overwriting previously recorded data, it creates a new file every time it runs, creating a unique file name out of the start date and time. You can adjust the time between samples, however by default it is set to 1 minute to get optimum

```

execute.py × CO2_Detector.py × terminate.py × Data Processing.py ×
1 import board
2 import adafruit_scd4x
3 from adafruit_extended_bus import ExtendedI2C as I2C
4 import time, os
5 import datetime
6
7 #create file name in the form: Data From DD-MM-YY, Start Time HH-MM-SS.txt
8 file_name=f'Data From {datetime.date.today().strftime('%d-%m-%y')}, Start Time {datetime.datetime.now().strftime('%H-%M-%S')}.txt'
9 file_location = os.path.join("/home/jim/env", file_name)
10
11 #create file if not exists
12 try:
13     records = open(file_location, "r")
14 except:
15     records = open(file_location, "w")
16     records.close()
17     print('file created successfully...')
18
19 #set up board
20 i2c = I2C(1)
21 scd4x = adafruit_scd4x.SCD4X(i2c)
22 print('board setup complete...')
23
24 #start measurements
25 scd4x.start_periodic_measurement()
26 print("waiting for first measurement...")
27
28 minutes_between_samples=1
29 while True:
30     if scd4x.data_ready:
31         data = open(file_location, "a")
32         #read sensor data
33         ppm=scd4x.CO2
34         temp=scd4x.temperature
35         humidity=scd4x.relative_humidity
36         #create timestamp in the form HH:MM:SS/DD-MM-YYYY
37         timestamp=f'{datetime.datetime.now().strftime('%H:%M:%S')}/{datetime.date.today().strftime('%d-%m-%y')}'
38         #print data to terminal
39         print(f'CO2: {ppm} ppm')
40         print(f'Temperature: {temp:.2f}*C')
41         print(f'Humidity: {humidity:.2f} %')
42         print(f'Time: {timestamp}\n')
43         #write data to file
44         data.write(f'{timestamp},{ppm},{temp:.2f},{humidity:.2f}\n')
45         data.close()
46         #wait until next sample
47         time.sleep(60 * minutes_between_samples)
48

```

resolution. The comments in this file do a good job at showing what is going on, so there is no need to explain further.

We also need a program to terminate data collection if we don't want to unplug the sensor. This program executes some commands to terminate all python programs running on the device. It prints all running python programs before and after this so you can check it has worked properly.

```

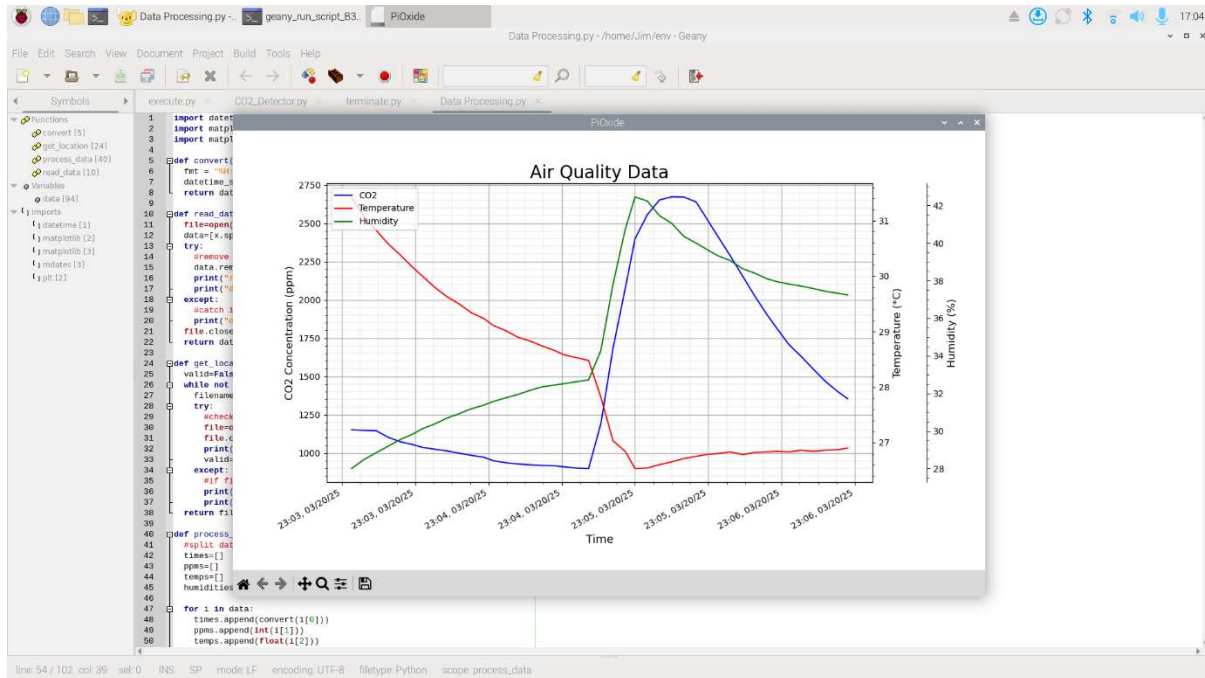
execute.py × CO2_Detector.py × terminate.py ×
1 import os
2 print('running programs:')
3 os.system("ps -ef | grep python")
4 print('terminating...')
5 os.system("sudo killall python3")
6 print('remaining programs:')
7 os.system("ps -ef | grep python")
8

```

Now that we have a system to collect and store the data, we want to be able to view it locally in a pretty format. To do this, we will be using matplotlib. This program (**see figure 4**) accepts a target file path input from the user, and extracts and plots the data from this file. I included several try except statements to try and catch user input errors and give guidance as to how to diagnose and fix problems, aiding ease of use. We want to be able to view all data (CO<sub>2</sub> Conc., Temp., humidity) in one graph so that we can spot trends between multiple different factors, so to do this I made use of matplotlib's twin functionality to plot all the data over a common time axis. All axes are labelled, the lines for the different bits of data are in different



colours, and there is a legend to distinguish them, making the graph very easy to read. Matplotlib also comes with some nice options for formatting a time axis, so I used this to show the date and time of each sample in a convenient way. With all of the software written, it is time for a bit of a test. To check everything is working, set the minutes between samples in the CO2 detection file to something small ( $\sim 0.01$ ) and run `execute.py` in your IDE. Try blowing on the sensor to show some obvious variation in the data, then leave it for about a minute. Stop the program, then plot the data using the algorithm you just wrote. You should end up with something that looks a bit like this:



Here you can clearly see where I blew on it, the CO<sub>2</sub> concentration and humidity went up and the temperature decreased, which is what we would expect, so everything is working! (make sure you reset the time between samples to something sensible before collecting more data)

```

execute.py × CO2_Detector.py × terminate.py × Data Processing.py ×
1 import datetime
2 import matplotlib.pyplot as plt
3 import matplotlib.dates as mdates
4
5 def convert(timestamp): #convert timestamp to datetime object
6     fmt = "%H:%M:%S/%a-%b-%Y"
7     datetime_str = datetime.datetime.strptime(timestamp, fmt)
8     return datetime_str
9
10 def read_data(address): #read data from file and split into a list
11     file=open(address,"r")
12     data=[x.split(",") for x in file.read().split("\n")] #split by new line, then split each line by comma
13     try:
14         #remove null item created by newline charcter at end of file
15         data.remove([""])
16         print("removed null item...")
17         print("data loaded...")
18     except:
19         #catch in case this character is removed
20         print("data loaded...")
21     file.close()
22     return data
23
24 def get_location(): #get file location input from user
25     valid=False
26     while not valid:
27         filename=input("file name:\n>")
28         try:
29             #check if file exists
30             file=open(filename,"r")
31             file.close()
32             print("file found")
33             valid=True
34         except:
35             #if file not found
36             print("no such file/invalid file name")
37             print("common errors:\n-include .txt file extension\n-file not in program directory\n-type in file name\n")
38     return filename
39
40 def process_data(data): #output data in a graph
41     #split data array into components
42     times=[]
43     ppms=[]
44     temps=[]
45     humidities=[]
46
47     for i in data:
48         times.append(convert(i[0]))
49         ppms.append(int(i[1]))
50         temps.append(float(i[2]))
51         humidities.append(float(i[3]))
52
53     #set up axes
54     fig,ax1=plt.subplots(figsize=(12,7),num="PiOxide")
55     ax2=ax1.twinx()
56     ax3=ax1.twinx()
57     ax3.spines.right.set_position(("axes", 1.1))
58
59     #plot data
60     p1, = ax1.plot(times,ppms,label="CO2",color="blue")
61     p2, = ax2.plot(times,temps,label="Temperature",color="red")
62     p3, = ax3.plot(times,humidities,label="Humidity",color="green")
63
64     #label axes and add minor grid lines
65     label_size=12
66     ax1.set_xlabel("Time",fontsize=label_size+1)
67     ax1.set_ylabel("CO2 Concentration (ppm)",fontsize=label_size)
68     ax1.minorticks_on()
69     ax2.set_ylabel("Temperature (°C)",fontsize=label_size)
70     ax2.minorticks_on()
71     ax3.set_ylabel("Humidity (%)",fontsize=label_size)
72     ax3.minorticks_on()
73
74     #add legend
75     plt.legend(handles=[p1, p2, p3], loc="upper left")
76
77     #draw grid
78     ax1.grid(linewidth=1)
79     ax1.grid(which="minor",linewidth=0.2)
80
81     #change title
82     plt.title("Air Quality Data",fontsize=20)
83
84     #format time labels on x axis
85     myFmt = mdates.DateFormatter("%H:%M, %D")
86     plt.gca().xaxis.set_major_formatter(myFmt)
87     plt.gcf().autofmt_xdate()
88
89     #decrease plot width so all y scales are seen on startup
90     plt.subplots_adjust(right=0.85)
91     plt.show()
92
93     #body code
94     data=read_data(get_location())
95     print("processing data...")
96     try:
97         process_data(data)
98         print("done")
99     except:
100         print("error in data handling")
101         print("make sure:\n-file name is correct\n-data is in correct format, e.g: 23:03:04/20-03-2025,1152,31.43,28.00")
102

```

Figure 4- Data Plotting Code



## STEP 05 – SCHEDULING A CRONJOB

Now we have the scripts all set up! But we cannot run them without having a mouse and keyboard and monitor. To avoid having to connect the pi to a monitor to start data collection, we can start it with a cronjob! Cronjobs are used to schedule certain events. We could setup a cronjob every 10 minutes to record data, but we decided that would be harder to terminate if you didn't want it to run. So instead, we will just schedule a cronjob to run the "execute.py" script on every reboot. To do this, open the crontab file by opening a new terminal panel and entering:

```
crontab -e
```

It may ask for you to select an editor, I would go with nano (it might be called /bin/nano and is usually option 1). Once opened, you should see a file filled with comments describing cronjobs. This may be worth a read if you're interested, if not scroll down to the bottom and on a new line add on the following line:

```
@reboot /usr/bin/python3 /home/(Your  
username name)/env/execute.py
```

Then remember to add a new line to the end of the crontab (it must end in a newline otherwise it will not work) and then save the file by pressing Ctrl-X and then Y to confirm the save. Now you can test it out! Shut off the RPI and unplug all the peripherals other than the sensor and then turn it on! Wait however long you need, depending on what you set the interval to, and then shut off power to the RPI and plug everything back in and check the logs folder for yourself! You should see lots of timestamped environmental data start to roll in like in Figure 5 →

**Figure 5 Some Environmental Data gathered over 20 minutes**

1	18:34:40/21-03-2025,782,23.52,44.28
2	18:35:40/21-03-2025,744,20.97,49.67
3	18:36:40/21-03-2025,732,20.02,52.37
4	18:37:40/21-03-2025,724,19.63,53.30
5	18:38:40/21-03-2025,714,19.46,53.59
6	18:39:40/21-03-2025,706,19.36,53.66
7	18:40:40/21-03-2025,682,19.35,53.58
8	18:41:40/21-03-2025,660,19.32,53.39
9	18:42:40/21-03-2025,641,19.31,53.34
10	18:43:40/21-03-2025,636,19.29,53.30
11	18:44:40/21-03-2025,612,19.30,53.16
12	18:45:40/21-03-2025,612,19.29,53.14
13	18:46:40/21-03-2025,615,19.25,53.12
14	18:47:40/21-03-2025,614,19.25,53.00
15	18:48:40/21-03-2025,616,19.22,52.97
16	18:49:40/21-03-2025,618,19.22,52.88
17	18:50:40/21-03-2025,616,19.23,52.91
18	18:51:40/21-03-2025,609,19.22,52.78
19	18:52:40/21-03-2025,602,19.20,52.81
20	18:53:40/21-03-2025,599,19.22,52.77
21	18:54:40/21-03-2025,599,19.22,52.73

## STEP 06 – ASSEMBLING THE CASE

Now all of the setup is complete and the sensor is working, it is time to put the pi in its case.

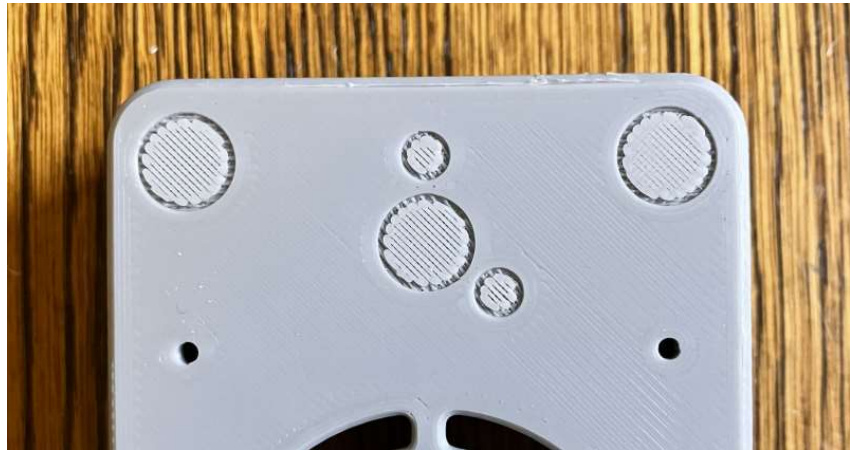
(Note: assembly is shown using a raspberry pi 3, however the case fits all models excluding the raspberry pi 1)

Here are all of the printed parts you need straight off the print bed:



From left to right you have the lid, the case itself, 4 feet, and the sensor mounting plate. First, let's get the parts ready for assembly.

On the bottom of the case there is a small amount of support material for the foot recesses (fig.6). This can easily be removed using the end of a knife or just your fingernail. If there are any remaining artefacts left in the recesses, remove them again with the end of a knife or a bit of sandpaper.



**Figure 6- Support Material on Bottom of Case**

Now, get the 4 feet; these should be ready to use instantly, so need no processing. Apply a small quantity of glue (superglue or equivalent is ideal) to each recess (fig.7), then add a foot on top, making sure it is centred, and press down (fig.8). repeat this for each foot.



**Figure 7- Superglue in Recess**

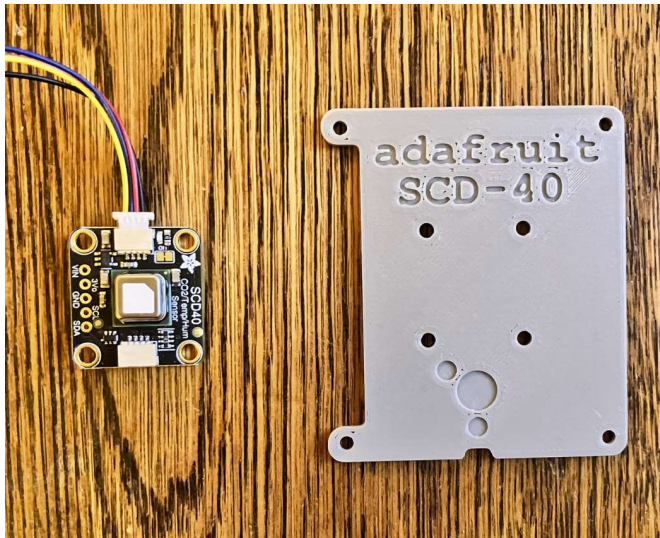


**Figure 8- Foot in place**



Once you have finished, it should look like this →

Next, we'll put the sensor on its mounting plate. For this you will need the Adafruit SCD-40 sensor, and the 3d printed plate.



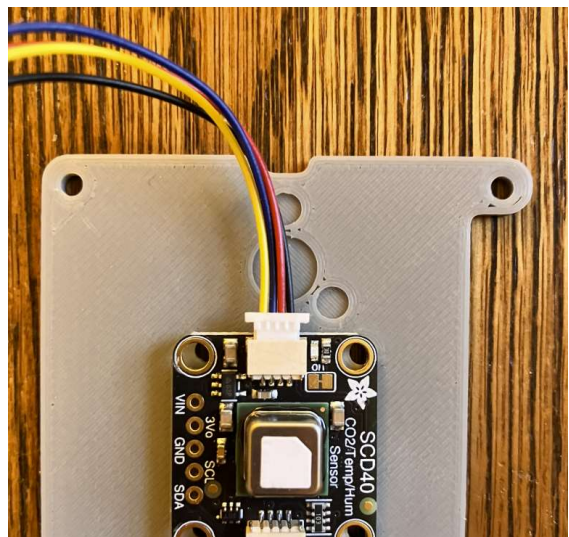
First, line the holes up on the sensor board with the holes on the mounting plate, making sure that the cable is facing towards the small cutout on the bottom edge of the plate (see fig.10). Now screw it on using 4 M3x5mm machine screws (fig.11). Once this is done, we can move on to fitting everything into the case.



**Figure 9- Screws in Place**



**Figure 9- All Feet Added**



**Figure 10- Correct Board Alignment**

Get all the sub-assemblies in one place and double-check everything is correct- poking around trying to adjust things in-situ can potentially cause damage to the components, so avoid it if possible.





Now, line up the USB ports with the large cutout, and place the pi in the case.

**IMPORTANT NOTE: make sure the SD card is removed before carrying out the next step otherwise the pi will not fit, and you may damage the SD card slot.**

once the pi is in place, apply gentle pressure to the raised side to push it into place. If resistance increases too much, make sure that everything is aligned correctly and there is no debris in the case.

If everything is correct, the pi should drop into place as shown in figure 13. You can now replace the SD.



**Figure 13- Pi in Correct Position**

First, screw the male end of the standoffs into the corresponding holes in the pi as shown in figure 14. The easiest way to do this is by inserting one of the M2.5x6 screws into the top of the standoff and using this to tighten it down.

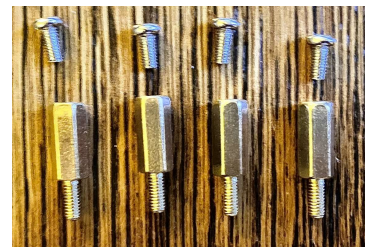
Then, align the sensor mounting plate above these standoffs, making sure the cable is tucked neatly through the cutout on the plate. Then, you can fix it in place using the machine screws as shown in figure 15→



**Figure 10- Pi Placed in Case**

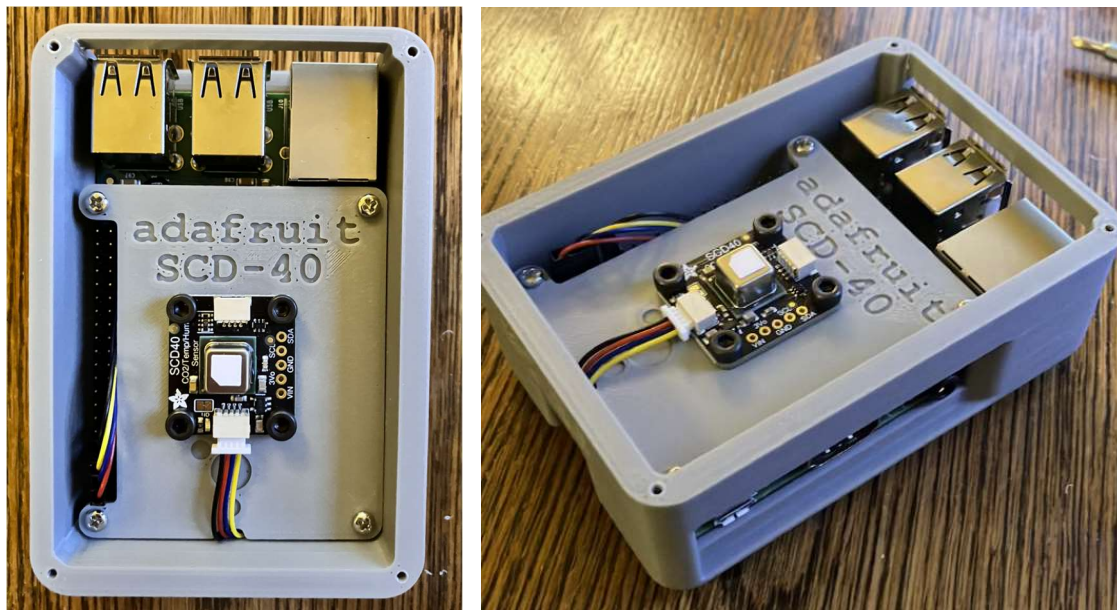
Once this is done, you are ready to add the sensor mounting plate. For this you need four

M2.5x11mm male to female standoffs, and four M2.5x6mm machine screws.



**Figure 11- Standoffs in Place**





**Figure 15- Sensor Mount in Position**

Now all of this is complete, you can finally screw the lid on. Make sure that the largest circular cutout is directly above the sensor, and screw it into place using four M2 screws- any thread length from 5mm to 10mm will work. You should be left with something that looks like this:



Now you are all done! plug it in to make sure everything works properly, then enjoy your new tool to ensure healthier, more productive learning.