Instructions:

Air Quality Kit

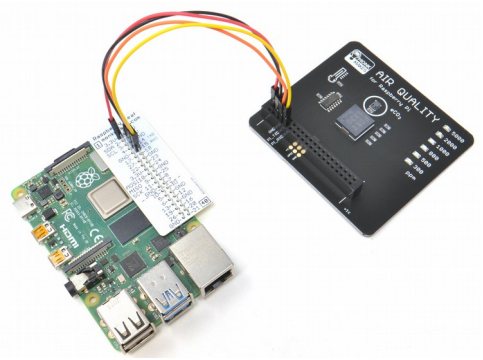
For Raspberry Pi

Designed for Raspberry Pi 400. Compatible with Raspberry Pi 2, 3 and 4.

V1c

Introduction

The MonkMakes Air Quality Kit for Raspberry Pi is based around the MonkMakes Air Quality Sensor board. This add-on for the Raspberry Pi measures the quality of the air in a room (how *stale* the air is) as well as the temperature. The board has a display of six LEDs (green, orange and red) that display the air quality and a buzzer. Temperature and air quality readings can be read by your Raspberry Pi, and the buzzer and LED display can also be controlled from your Raspberry Pi.



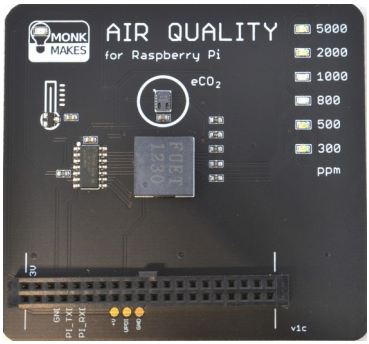
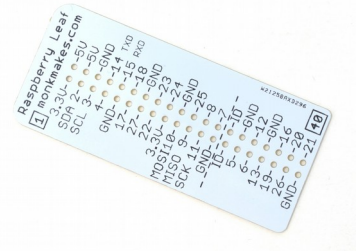
The Air Quality Sensor board, plugs directly into the back of a Raspberry Pi 400, but, can also be used with other models of Raspberry Pi, using the jumper wires and GPIO template included in the kit.

Table of Contents

Parts

Please note that a Raspberry Pi is NOT included in this kit.

Before you do anything else, check that your kit includes the items below.

Air Quality Sensor Board for Raspberry Pi.	
Female to male jumper wires.	
Raspberry Leaf GPIO template	

Air Quality and eCO2

The Air Quality Sensor board uses a sensor with a part number of CCS811. This small chip does not actually measure the level of CO2 (carbon dioxide) but instead the level of a group of gasses called volatile organic compounds (VOCs). When indoors, the level of these gasses rises at a fairly similar rate to that of CO2, and can therefore be used to estimate the level of CO2 (called the equivalent CO2 or eCO2).

The level of CO2 in the air we breathe has a direct influence on our well-being. CO2 levels are of particular interest from a public health point of view as, to put it simply, they are a measure of how much we are breathing other people's air. We humans breathe out CO2 and so, if several people are in a poorly ventilated room, the level of CO2 will gradually increase. This is much the same as the viral aerosols that spread colds, flus and Coronavirus as people breathe both out together.

Another important impact of CO2 levels is in cognitive function – how well you can think. This study (amongst many more) have some interesting findings. The following quote is from the National Centre for Biotechnology Information in the USA:

"at 1,000 ppm CO2, moderate and statistically significant decrements occurred in six of nine scales of decision-making performance. At 2,500 ppm, large and statistically significant reductions occurred in seven scales of decision-making performance" *Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3548274/*

The table below is based on information from https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms and shows the levels at which CO2 can become unhealthy. The CO2 readings are in ppm (parts per million).

Level of CO2 (ppm)	**Notes**
250-400	Normal concentration in ambient air.
400-1000	Concentrations typical of occupied indoor spaces with good air exchange.
1000-2000	Complaints of drowsiness and poor air.
2000-5000	Headaches, sleepiness and stagnant, stale, stuffy air. Poor concentration, loss of attention, increased heart rate and slight nausea may also be present.
5000	Workplace exposure limit in most countries.
>40000	Exposure may lead to serious oxygen deprivation resulting in permanent brain damage, coma, even death.

Setting Up

Whether you are using a Raspberry Pi 400 or a Raspberry Pi 2, 3 or 4, make sure that the Raspberry Pi is shutdown and **powered off** before you connect the Air Quality Sensor.

The Air Quality Sensor will display the eCO2 readings as soon as it gets power from your Raspberry Pi. So, once you have connected it, the display should indicate the eCO2 level. You will then learn how to interact with the board, receiving readings and controlling the LEDs and buzzer from a Python program.

Connecting the Air Quality Sensor (Raspberry Pi 400)

It is very important that you do not push the connector in at an angle, or push it too hard, as you may bend the pins on the GPIO connector. When the pins are lined up correctly, it should push into place easily.

The connector fits as shown above. Notice that the bottom edge of the board lines up with the bottom of the Pi 400's case, and the side of the board leaves just enough room for easy access to the micro SD card.

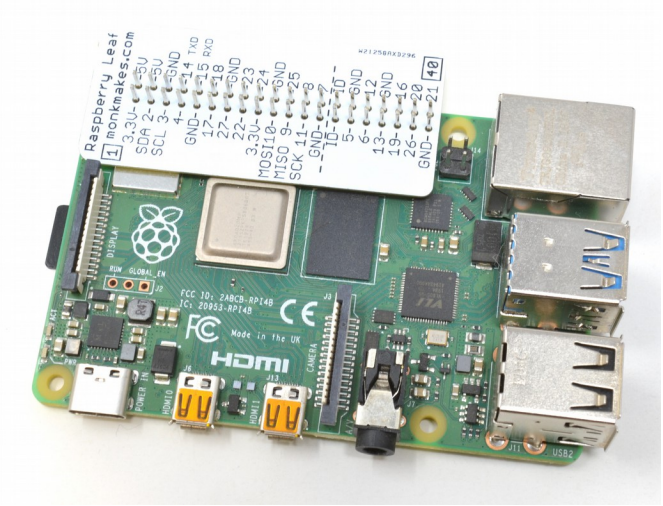
Once you have connected the board, power up your Raspberry Pi -- both the power LED (in the MonkMakes logo) and one of the eCO2 LEDs should also light.

Connecting the Air Quality Sensor (Raspberry Pi 2/3/4)

If you have a Raspberry Pi 2, 3, 4, then you will need the Raspberry Leaf and some female to male jumper wires to connect the Air Quality Sensor board to your Raspberry Pi.

WARNING: Reversing the power leads or connecting the Air Quality Sensor to 5V rather than the 3V pin of the Raspberry Pi is likely to break the sensor and may damage your Raspberry Pi. So, please check the wiring carefully before powering on your Raspberry Pi.

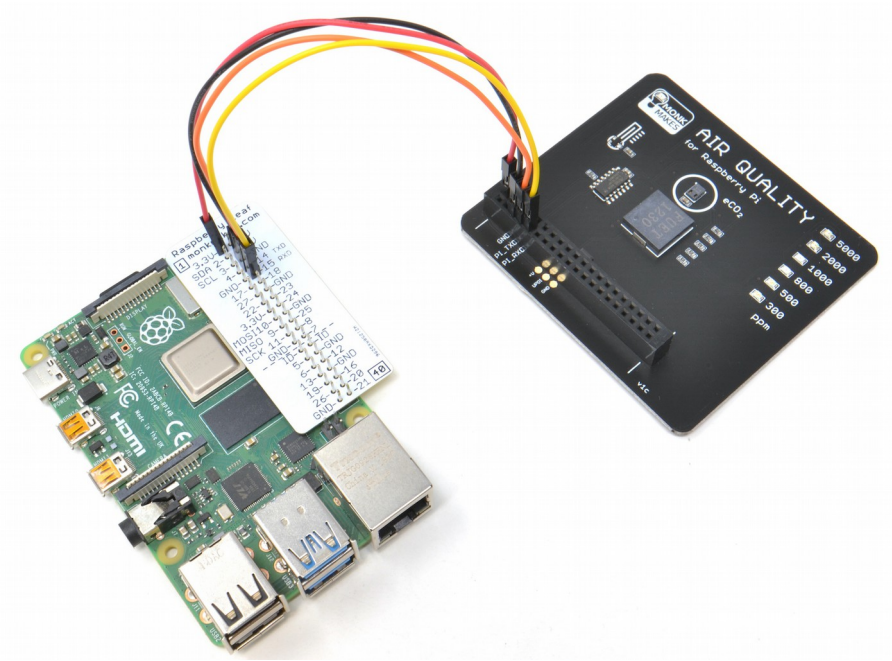
Start by fitting the Raspberry Leaf over your Raspberry Pi's GPIO pins so that you can tell which pin is which. The template can fit either way around, so make sure you follow the diagram below.



Next you are going to connect four leads between the Raspberry Pi's GPIO pins and the Air Quality board like this:

Raspberry Pi Pin (as labelled on the Leaf)	Air Quality Board (as labelled on connector)	Suggested wire color.
GND (any pin marked GND will do)	GND	Black
3.3V	3V	Red
14 TXD	PI_TXD	Orange
15 RXD	PI_RXD	Yellow

Once it's all connected, it should look like this:



Check your wiring carefully and then power up your Raspberry Pi -- both the power LED (in the MonkMakes logo) and one of the LEDs should also light.

Unplugging the Air Quality Board

Before removing the board from a Raspberry Pi 400.

1. Shutdown the Raspberry Pi.

2. Gently ease the board off the back of the Pi 400, edging it a little from each side in turn, so as not to bend the pins.

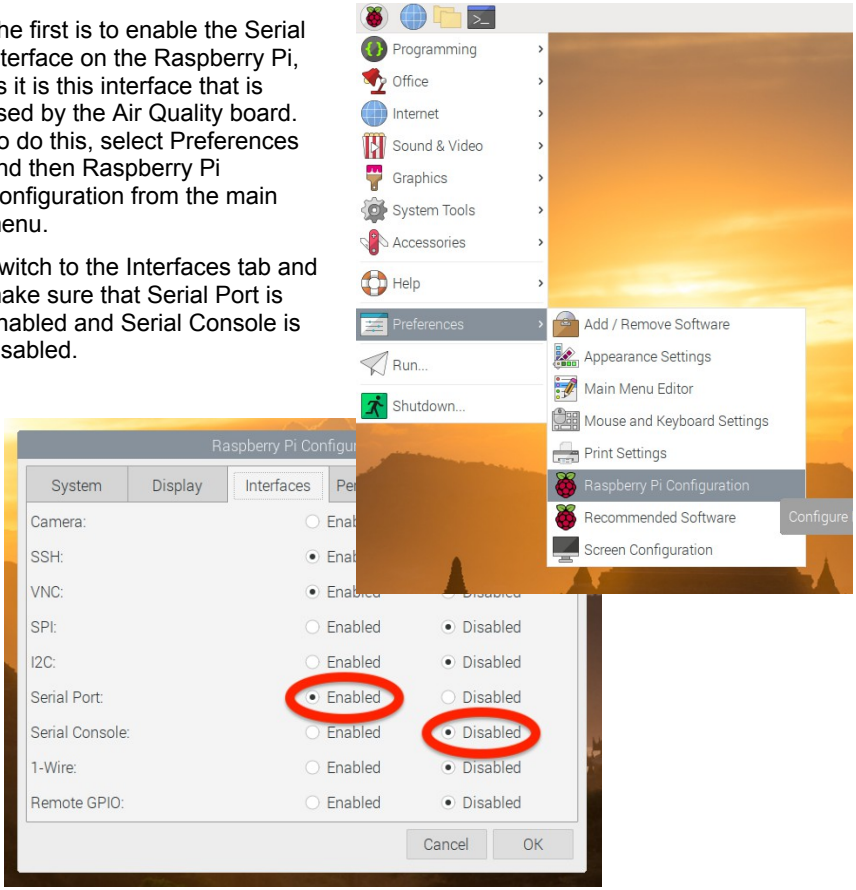
If you have a Pi 2/3/4 just remove the jumper wires from the Raspberry Pi.

Enabling the Serial Interface

Even though the board will show the eCO2 level without any programming, that means we are just using the Raspberry Pi as a power source. To be able to interact with the board from a Python program, on our Raspberry Pi, there are a few more steps that we need to take.

The first is to enable the Serial interface on the Raspberry Pi, as it is this interface that is used by the Air Quality board. To do this, select Preferences and then Raspberry Pi Configuration from the main menu.

Switch to the Interfaces tab and make sure that Serial Port is enabled and Serial Console is disabled.

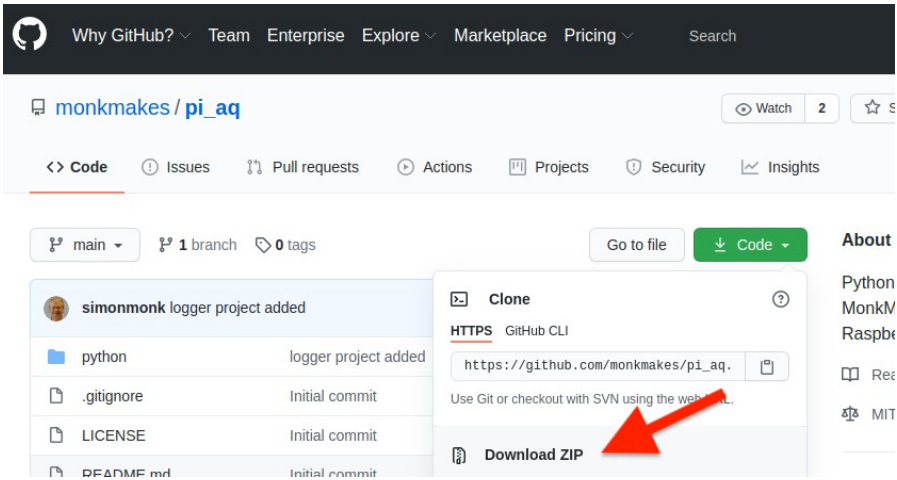


Downloading the Example Programs

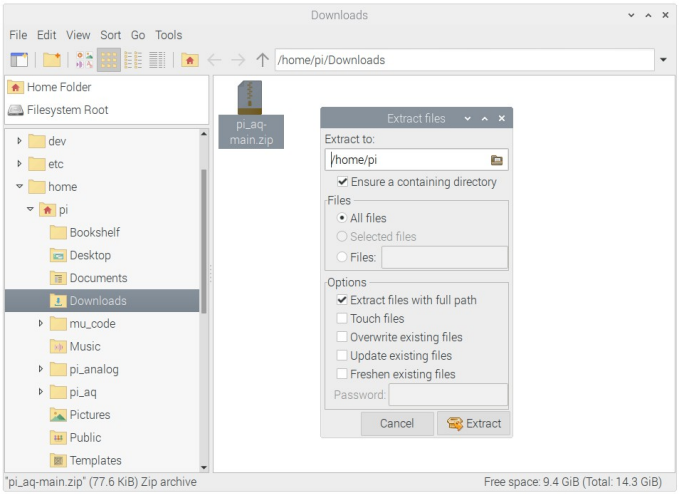
The example programs for this kit are available for download from GitHub. To fetch them, start a browser window on your Raspberry Pi and go to this address:

https://github.com/monkmakes/pi_aq

Download a zip archive of the project by clicking on the Code button and then the Download ZIP option.



Once the download is finished, extract the files from the ZIP archive by finding the ZIP file in your Downloads folder and then then right-clicking on it and selecting the option Extract To..

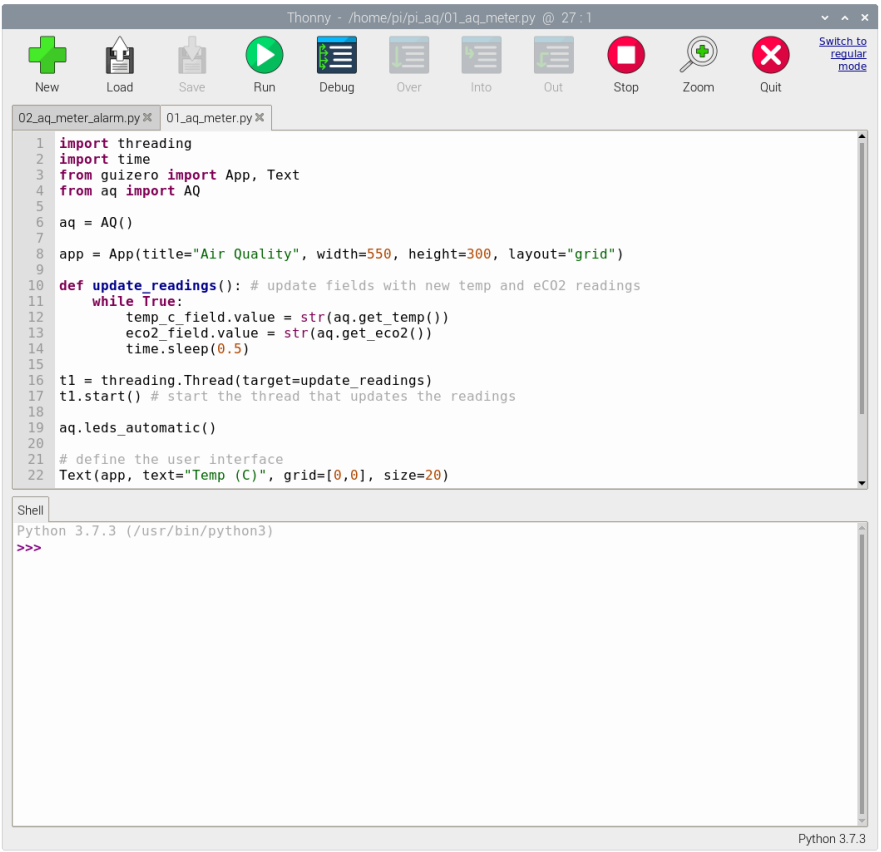


Select a suitable directory (I would recommend your home directory – /home/pi) and extract the files. This will create a folder called pi_aq-main. Rename this to just pi_aq.

Thonny

Having downloaded the programs, you could just run them from the command line. However, it's good to take a look at the files, and the Thonny editor will allow us to edit the files and to run them.

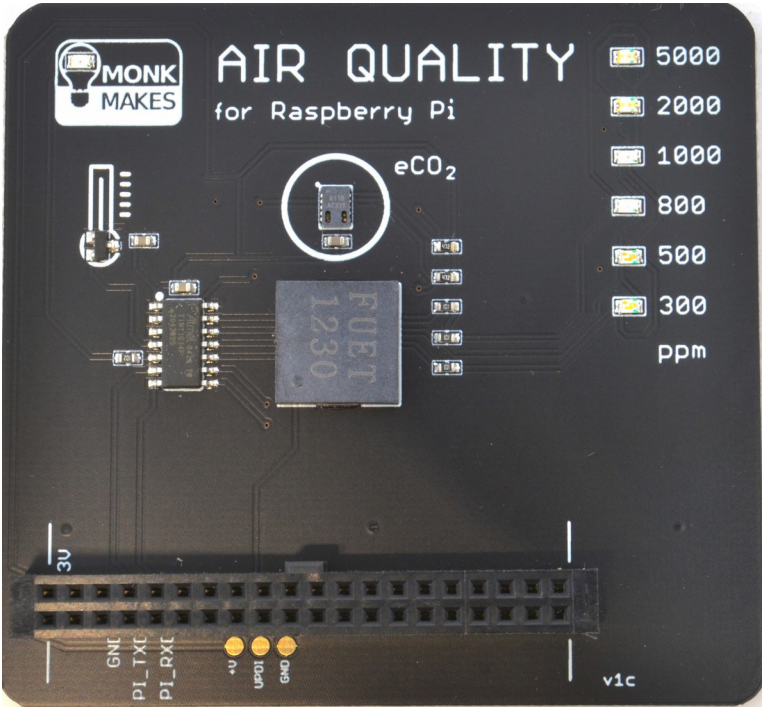
The Thonny Python editor is pre-installed in Raspberry Pi OS. You will find it in the Programming section of the main menu. If for any reason it's not installed on your Raspberry Pi, then you can install it using the Add / Remove Software menu option on the Preferences Menu item.



The next section explains a bit more about what this sensor is measuring, before we get on to interacting with the Air Quality board using Python and Thonny.

Getting Started

Before we start the Python programming, let's take a look at the Air Quality Board.

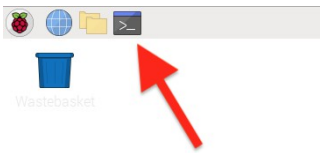


The power indicator LED in the top left, provides a quick check that the board is receiving power. Below this is a temperature sensor chip, and next to this is the eCO2 sensor chip itself. If you look closely at it, you will see that it has tiny holes for the air to get in and out.

Directly beneath the eCO2 sensor is a buzzer, that you can turn on and off from your programs. This is useful for providing alarms.

The column of six LEDs is made up (from bottom to top) of two green LEDs, two orange LEDs and two red LEDs. These will light when the level of eCO2 marked next to each LED is exceeded. They will show the level as soon as the Raspberry Pi powers up, but you can also control them using Python.

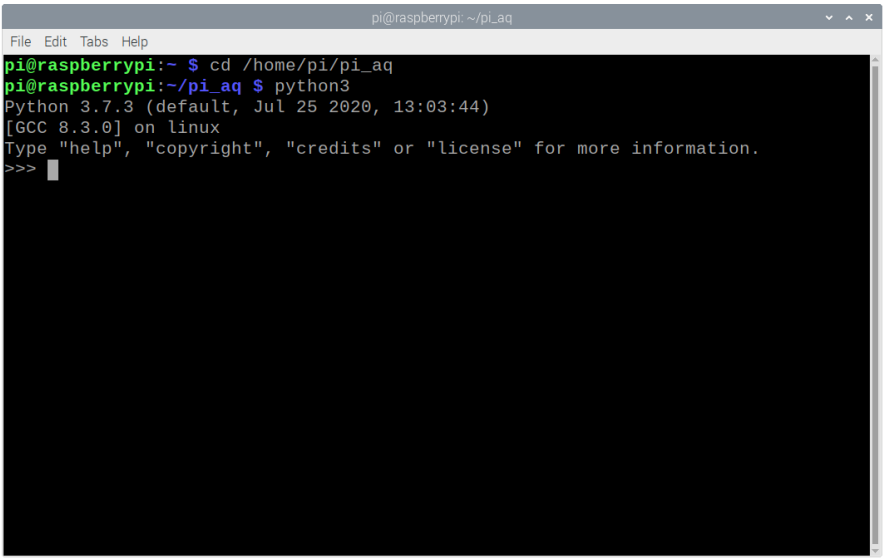
Let's start by trying out a few experiments from the command line. Open a Terminal session by clicking on the Terminal icon at the top of your



screen, or the Accessories section on the Main menu.

When the terminal opens, type the following commands after the $ prompt, to change directories (cd) and to open a Python console, by typing the following commands:

```
$ cd /home/pi/pi_aq
$ python3
```



Open the local aq module by typing the command:

```
>>> from aq import AQ
>>>
```

Then create an instance of the AQ class by typing:

```
>>> aq = AQ()
>>>
```

We can now read the CO2 level by typing the command:

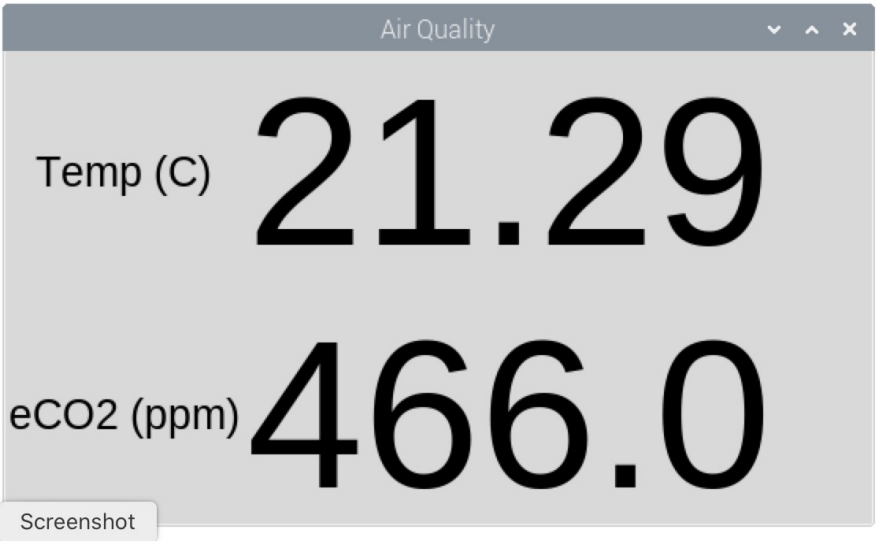
```
>>> aq.get_eco2()
434.0
>>>
```

So in this case, the eCO2 level is a nice fresh 434 ppm. Lets get the temperature now (in degrees Celcius).

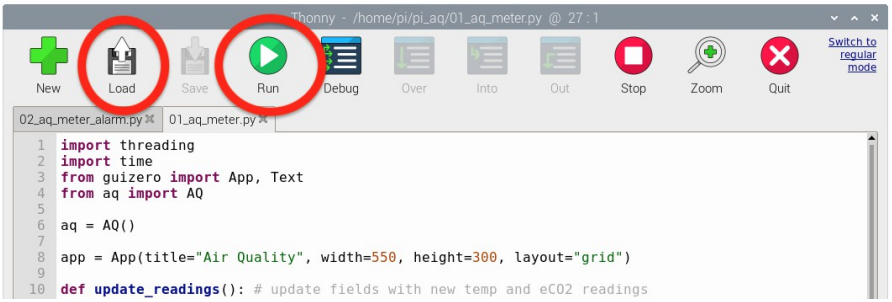
```
>>> aq.get_temp()
20.32
```

Program 1. eCO2 Meter

When you run this program the window similar to the one shown below will open, showing you the temperature and eCO2 level. Try putting your finger on the temperature sensor and the temperature readings should rise. You can also breathe gently on the eCO2 sensor and the readings should increase.



To run the program, Load the file 01_aq_meter.py in Thonny and then click on the Run button.



Here's the code for the project. The code makes use of the GUI Zero library which you can read more about in Appendix B.

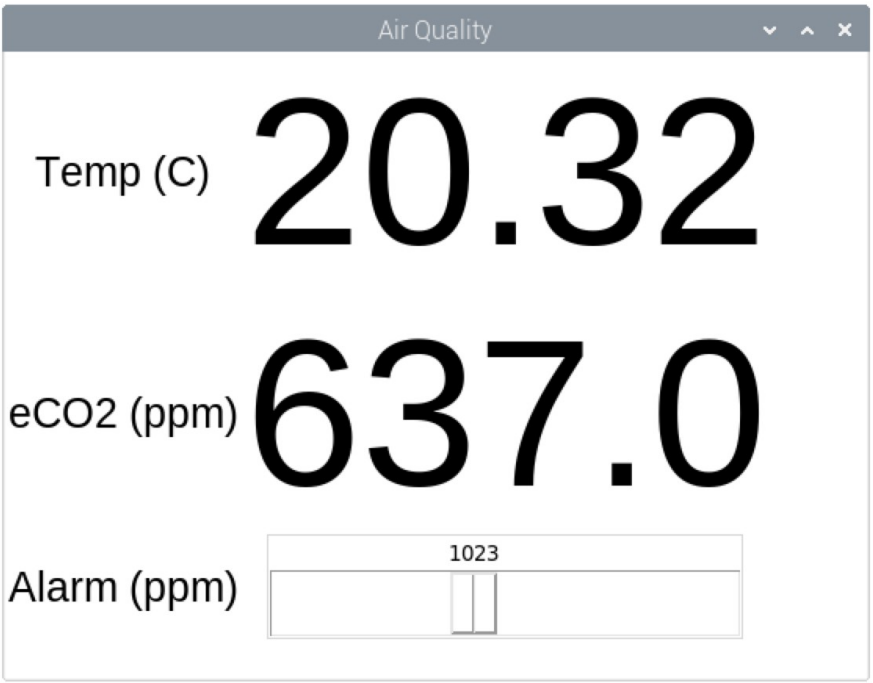
```
import threading
import time
from guizero import App, Text
from aq import AQ

aq = AQ()

app = App(title="Air Quality", width=550, height=300,
layout="grid")

def update_readings(): # update fields with new
                       # temp and eCO2 readings
    while True:
        temp_c_field.value = str(aq.get_temp())
        eco2_field.value = str(aq.get_eco2())
        time.sleep(0.5)

t1 = threading.Thread(target=update_readings)
t1.start() # start the thread that updates the readings

aq.leds_automatic()

# define the user interface
Text(app, text="Temp (C)", grid=[0,0], size=20)
temp_c_field = Text(app, text="-", grid=[1,0], size=100)
Text(app, text="eCO2 (ppm)", grid=[0,1], size=20)
eco2_field = Text(app, text="-", grid=[1,1], size=100)
app.display()
```

To allow readings of temperature and light to take place without interrupting the workings of the user interface, the *threading* library is imported. The function *update_readings* will loop forever, taking readings every half second and updating the fields in the window.

The rest of the code provides the user interface fields needed to display the temperature and eCO2 level. These are laid out as a grid, so that the fields line up. So, each field is defined with a *grid* attribute that represent the column and row positions. So, the field that displays the text *Temp (C)* is at column 0, row 0 and the corresponding temperature value (*temp_c_field*) is at column 1, row 0.

Program 2. eCO2 Meter with alarm

This program extends Program one, by making use of the buzzer and some fancy user interface features, to make an alarm sound and the window turn red if a set level of eCO2 is exceeded.



The slider at the bottom of the window sets the eCO2 level at which the buzzer should sound and the window turn red. Try setting the Alarm level a little higher than the current eCO2 level and then breathe on the sensor.

Here is the code for Program 2, much of it is very similar to Program 1. Areas of interest have been highlighted in bold.

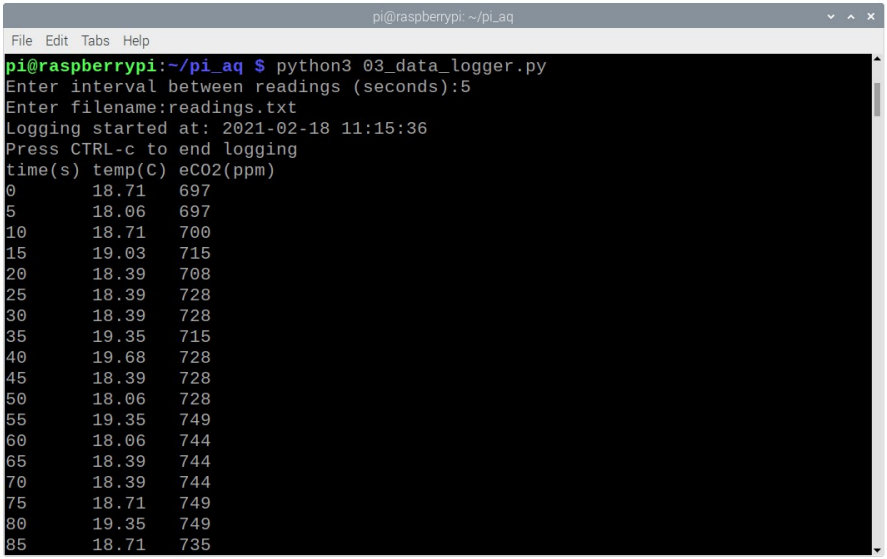
```
import threading
import time
from guizero import App, Text, Slider
from aq import AQ

aq = AQ()

app = App(title="Air Quality", width=550, height=400,
layout="grid")

def update_readings():
    while True:
        temp_c_field.value = str(aq.get_temp())
        eco2 = aq.get_eco2()
        eco2_field.value = str(eco2)
        if eco2 > slider.value:
            app.bg = "red"
            app.text_color = "white"
            aq.buzzer_on()
        else:
            app.bg = "white"
            app.text_color = "black"
            aq.buzzer_off()
        time.sleep(0.5)

t1 = threading.Thread(target=update_readings)
t1.start() # start the thread that updates the readings

aq.leds_automatic()

# define the user interface
Text(app, text="Temp (C)", grid=[0,0], size=20)
temp_c_field = Text(app, text="-", grid=[1,0], size=100)
Text(app, text="eCO2 (ppm)", grid=[0,1], size=20)
eco2_field = Text(app, text="-", grid=[1,1], size=100)
Text(app, text="Alarm (ppm)", grid=[0,2], size=20)
slider = Slider(app, start=300, end=2000, width=300,
                    height=40, grid=[1,2])
app.display()
```

Firstly, we need to add *Slider* to the list of things we import from guizero.

We also need to extend the *update_readings* function, so that, as well as displaying the temperature and eCO2 level, it also checks to see if the level is above the threshold. If it is, it sets the window background to red, the text to white and turns the buzzer on. If the eCO2 level is below the threshold set by the slider, it reverses this, and turns the buzzer off.

Program 3. Data logger

This program (03_data_logger.py) doesn't have a graphical interface. It just prompts you to enter an interval in seconds between readings, followed by the name of a file in which to save the readings.



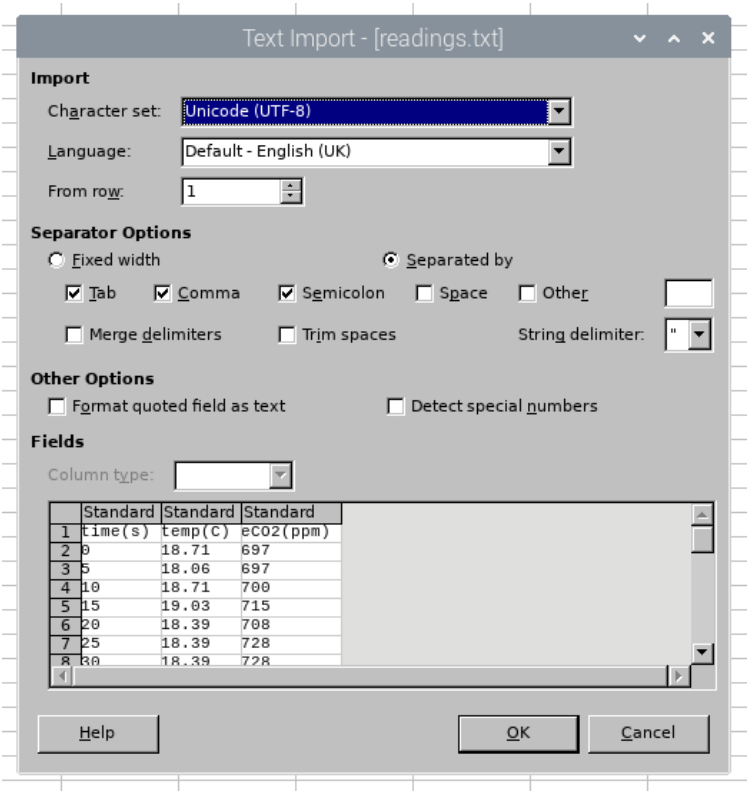
In the example above, sampling is set to 5 seconds and the file is called *readings.txt*. When you have finished logging data, CTRL-c will end logging and close the file.

The data are saved in the same format as they are shown in the screen capture above. That is, the first line specifies the headings, with each value delimited by a TAB character. The file is saved in the same directory as the program.

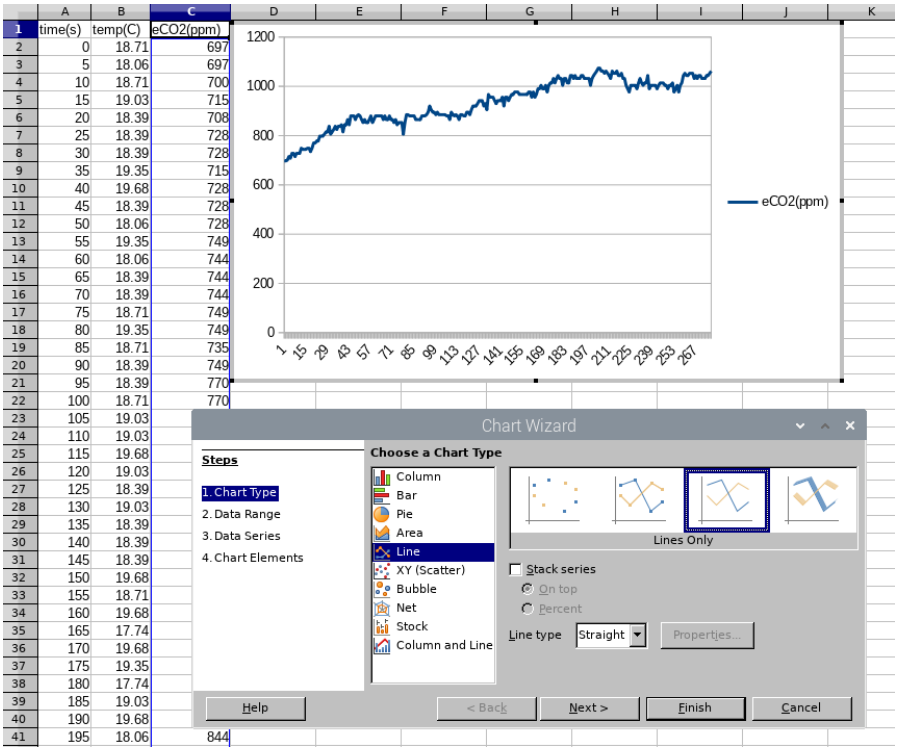
Having captured the data, you can then import it into a spreadsheet (like LibreOffice) on your Raspberry Pi and then plot a chart from the data.

If LibreOffice is not installed on your Raspberry Pi, you can install it using the *Add/Remove Software* option on the *Preferences* Menu.

Open a new spreadsheet, chose Open from the file menu, and navigate to the data file you want to look at. This will open an import dialog (see the next page) showing that the spreadsheet has automatically detected the columns of the data.



Click OK to import the data, and then select the column for the eCO2 readings. You can then plot a graph of these readings by selecting *Chart* from the *Insert* menu, and then choosing a Chart type of *Line*, followed by *Line Only*. This gives you the graph shown on the next page.



As an experiment, try leaving the logger program running for a 24 hour period to see how the eCO2 level changes throughout the day.

Appendix A. API Documentation

For the serious programmers – here is the technical documentation. The file monkmakes_aq.py is not installed as a full Python library, but should just be copied into the same folder as any other code that needs to use it.

aq.py

The monkmakes_aq.py module is a class that wraps the serial communication between your Raspberry Pi and the Air Quality board.

Creating an instance of AQ:

```
aq = AQ()
```

Reading the eCO2 reading

```
aq.get_eco2() # returns the eCO2 reading in ppm
```

Reading the temperature in degrees C

```
aq.get_temp() # returns the temperature in degrees C
```

The LED display

```
aq.leds_manual()        # set LED mode to manual
aq.leds_automatic()     # set LED mode to automatic
                        #   so that LEDs display eCO2
aq.set_led_level(level) # level 0-LEDs off,
                        # level 1-6 LED 1 to 6 lit
```

Buzzer

```
aq.buzzer_on()
aq_buzzer_off()
```

The class communicates with the sensor board using the Pi's serial interface. If you want to see details of the serial interface, then please take a look at the datasheet for this product. You will find a link to this from the product's web page (http://monkmakes.com/pi_aq)

Appendix B. GUI Zero

Laura Sach and Martin O'Hanlon at The Raspberry Pi Foundation have created a Python library (GUI Zero) that makes it super easy to design GUIs. This kit uses that library.

Before you can use the library, you need to import the bits of it that you want to use in your program.

For example, if we just wanted a window containing a message, here's the import command:

```
from guizero import App, Text
```

The class *App* represents the application itself, and every program you write that uses guizero needs to import this. The only other class needed here is *Text,* that is used to display the message.

The following command creates the application window, specifying a title and the window's starting dimensions.

```
app = App(title = "My Window", width="400", height="300")
```

To add some text to the window, we can use the line:

```
Text(app, text="Hello World", size=32)
```

The window is now prepared for display, but won't actually appear until the program runs the line:

```
app.display()
```

You can find out more about guizero here: https://lawsie.github.io/guizero/start/

Troubleshooting

Problem: The board is plugged into my Pi 400 but the power LED is not lit.

Solution: Check that the GPIO pins are lined up correctly with the socket. See page 4.

Problem: The board is plugged into my Pi 400 but the power LED is flashing rapidly.

Solution: This indicates a problem with the sensor. Sometimes, all it needs is for the power to be reset by turning your Raspberry Pi off and on again. If you do this and the flashing continues, you probably have a faulty board, so please contact support@monkmakes.com

Problem: I've just connected everything up, but the eCO2 readings seem wrong.

Solution: The type of sensor used in the MonkMakes Air Quality Sensor, will start producing readings from the first time you connect it. However, the readings will become more accurate with time. The datasheet for the sensor IC suggests the readings will only start to become accurate after 20 minutes of running time.

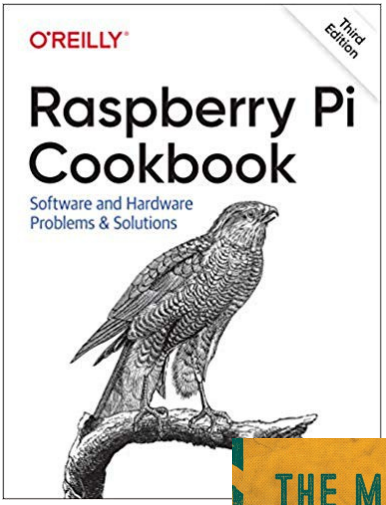
Problem: I'm comparing the readings from this sensor with a true CO2 meter and the readings are different.

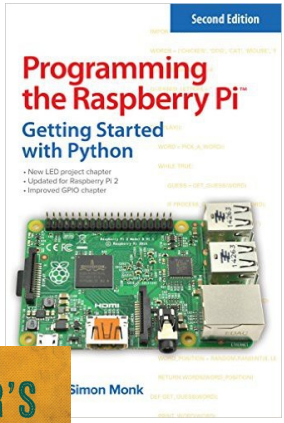
Solution: That's to be expected. The Air Quality Sensor estimates the CO2 concentration (that's what the 'e' is for in eCO2) by measuring the level of volatile organic compounds (VOCs). True CO2 sensors are much more expensive.

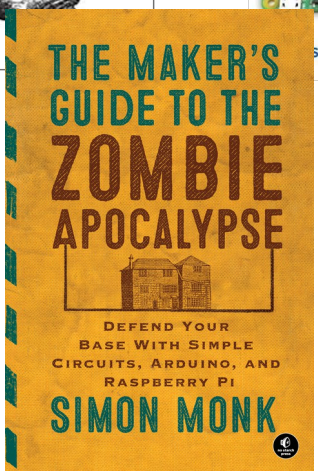
Programming & Electronics

If you want to learn more about programming the Raspberry Pi and Electronics, then the designer of this kit (Simon Monk) has written a number of books that you might enjoy.

You can find out more about books by Simon Monk at: http://simonmonk.org or follow him on Twitter where he is @simonmonk2







MonkMakes

For more information on this kit, the product's home page is here:
https://monkmakes.com/pi_aq

As well as this kit, MonkMakes makes all sorts of kits and gadgets to help with your maker projects. Find out more, as well as where to buy at:
https://www.monkmakes.com/products

You can also follow MonkMakes on Twitter @monkmakes.

Project Box 1 for Raspberry Pi

Amplified Speaker Kit for Raspberry Pi

Air Quality Kit for micro:bit