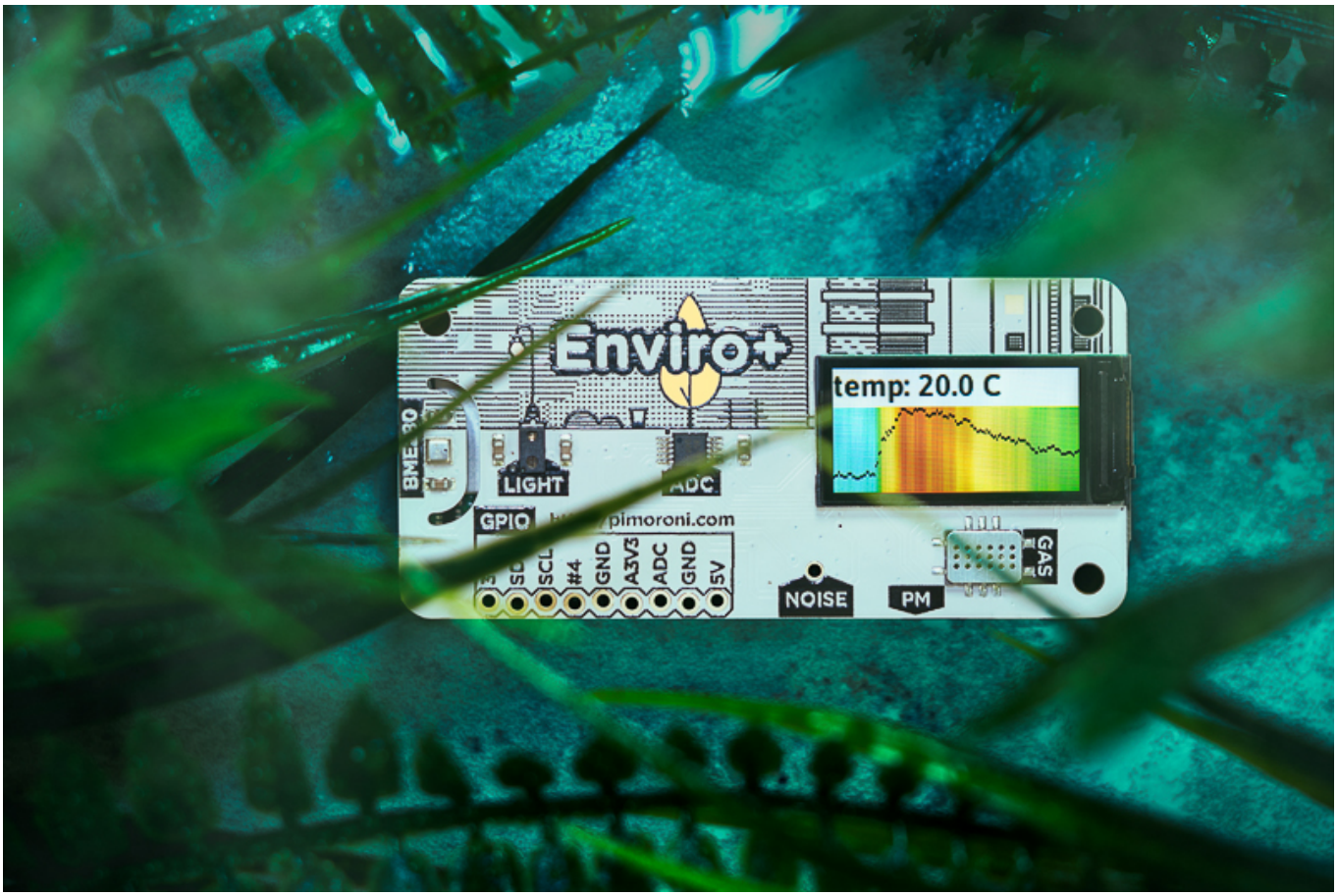
Getting Started with Enviro+

Designed for environmental monitoring, Enviro+ for Raspberry Pi lets you measure air quality (pollutant gases and particulates), temperature, pressure, humidity, light, and noise level. When combined with a particulate matter sensor, it's great for monitoring air quality just outside your house (more information below), or without the particulate sensor you can use it to monitor indoor conditions.



It's an affordable alternative to environmental monitoring stations that can cost tens of thousands of pounds and, best of all, it's small and hackable and lets you contribute your data to citizen science efforts to monitor air quality via projects like Luftdaten.

In this tutorial, we'll go through what the different parts of the board do, how to attach it to your Raspberry Pi, how to install and use the Python library, and also look at how to run a few of the code examples that we've supplied.

A word of warning... this tutorial is *long* as there's a lot to Enviro+, and this tutorial is intended to be a complete reference to all the parts of it. You might need several cups of coffee or tea and biscuits to get through this!

What's on Enviro+?

BME280 temperature, pressure, and humidity sensor

The BME280 is a neat little weather sensor that measures temperature, pressure, and humidity. Air pollution, and particulates (more on them later) especially, can be affected by changes in the weather, so having a weather sensor onboard Enviro+ is really useful. It's also great for indoor monitoring, if you want to keep track of conditions in your home, for example.

On Enviro+, the BME280 sensor is at the left hand edge of the board, and has been placed there deliberately to be positioned away from the Pi's CPU. There's also a little slot routed out next to the sensor that helps to reduce heat radiated through the Enviro+ board to the sensor.

LTR-559 light and proximity sensor

This sensor can read the ambient light level in Lux (a unit of light intensity) and also has a proximity sensor. The LTR-559 is the same sort of sensor that's next to the camera in your mobile phone that automatically dims and brightens the screen depending on the light level and disables the touchscreen when it's next to your ear.

As well as being great for reading the light level, the LTR-559's proximity sensor is really handy as a proximity-sensitive input. We use it in one of our examples to toggle through displays of the different data from Enviro+ on the LCD.

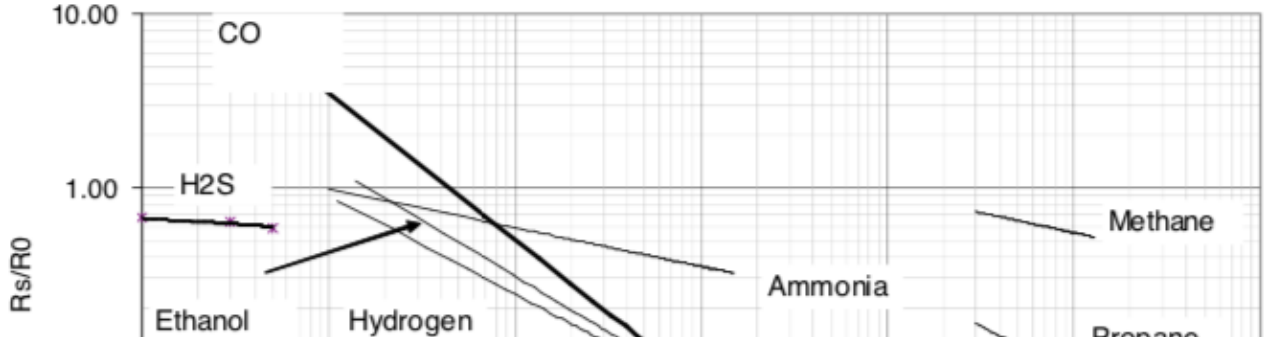
MICS6814 analog gas sensor and ADS1015 analog to digital converter (ADC)

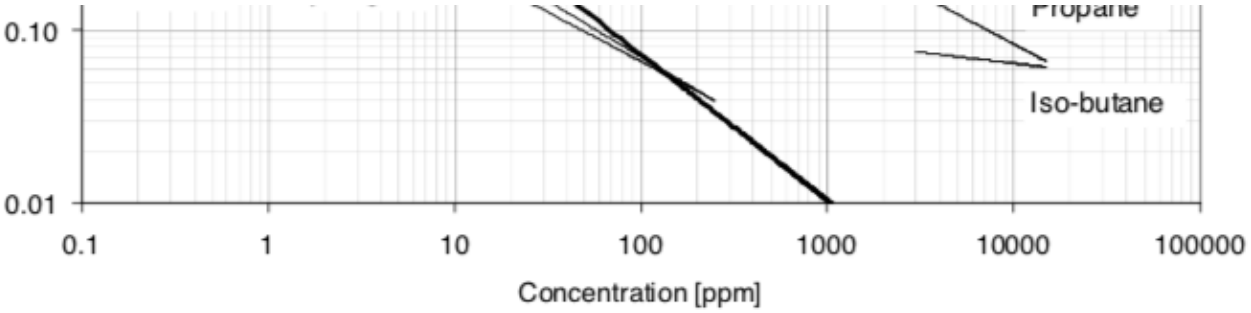
The MICS6814 is an analog gas sensor that can detect three different groups of gases that they refer to in the datasheet as reducing, oxidising, and NH3. The major gases/vapours that the sensor detects are: carbon monoxide (reducing), nitrogen dioxide (oxidising), and ammonia (NH3), but it is also sensitive to others, including hydrogen, ethanol, and hydrocarbons.

Each of the three groups of gases is effectively its own sensor within the MICS68141, and the analog voltage readings that the sensor produces are read by an analog to digital converter (ADC) and then converted into resistances by our code. These resistances will range from low hundreds of Ohms to tens of thousands of Ohms, and vary depending on the levels of each group of gases.

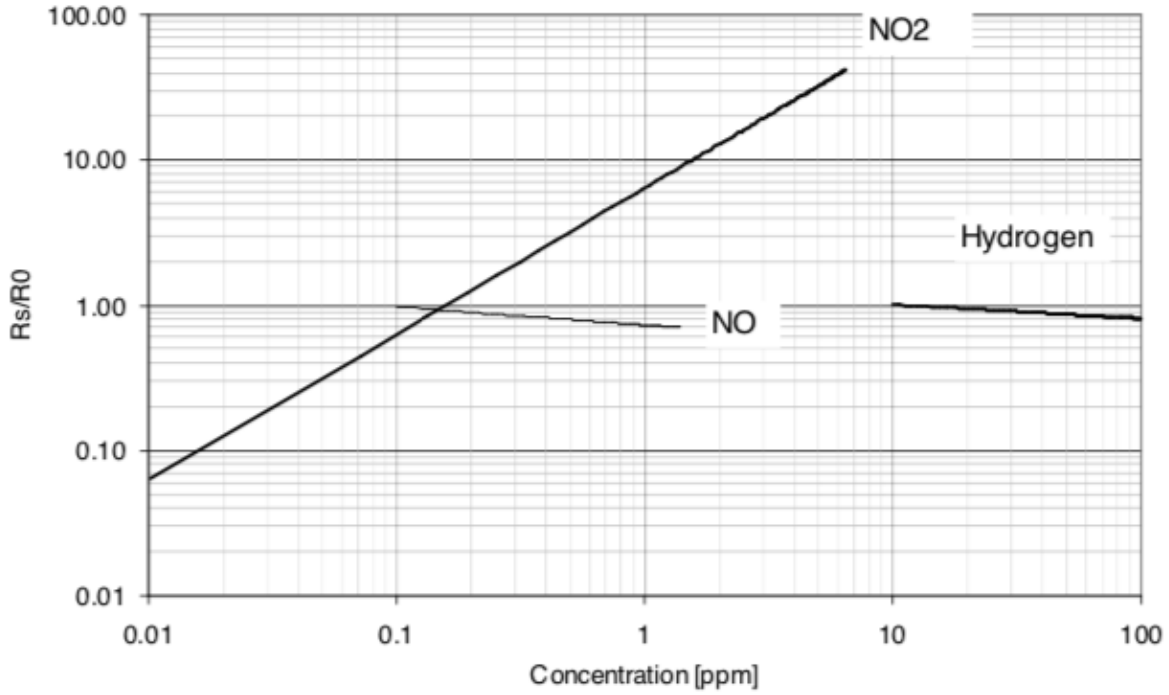
Because each group of gases could be a mix of different gases, it's not possible to single out any one gas specifically or to quantify their levels precisely, so the best way to interpret the data is to take readings until they stabilise, set a baseline, and then look for changes relative to that baseline. This gives you a rough idea of whether the air quality is increasing or decreasing.

The reducing and NH3 resistance readings will drop with increasing concentrations of the gases that they detect, and the oxidising sensor will increase with increasing levels of nitrogen dioxide. You can see how the sensor reacts to the different gases in the charts below.

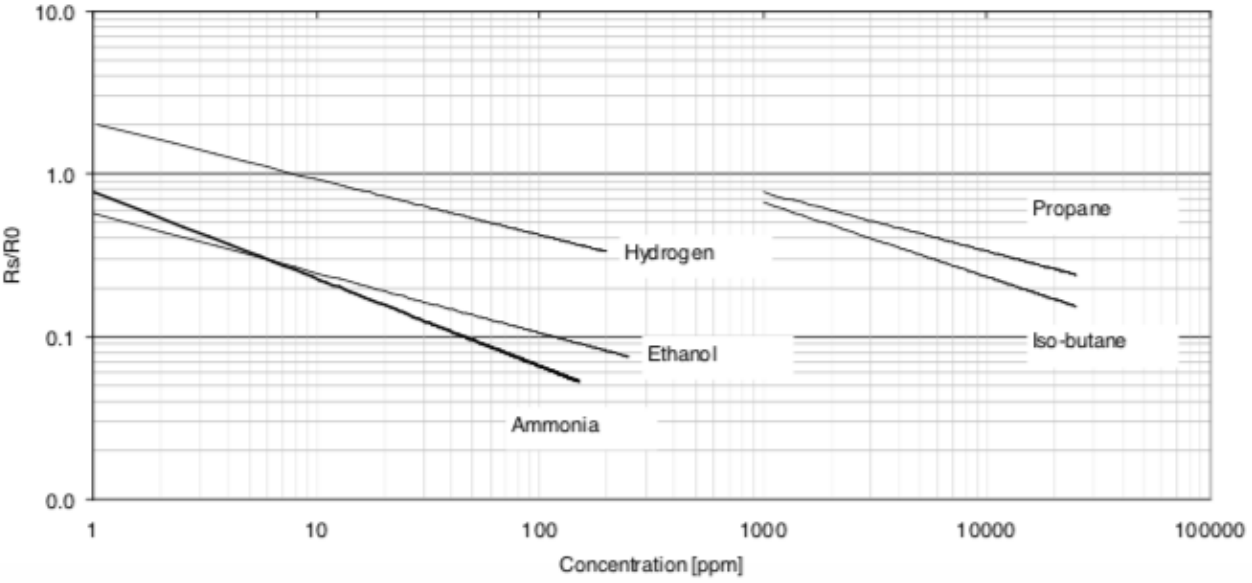




RED sensor, continuous power ON, 25°C, 50% RH



OX sensor, continuous power ON, 25°C, 50% RH



NH3 sensor, continuous power ON, 25°C, 50% RH

PMS5003 particulate matter (PM) sensor

This sensor is an optional add-on to your Enviro+ and connects, using a cable included with the sensor, to the connector on the underside of your Enviro+.

A large part of what we think of as "air pollution" is particulate matter. These particulates range in size and type, from larger ones like dust, pollen, and mould spores, to smaller ones like smoke particles (from combustion), metal ions, and othe organic particles.

The PMS5003 has a little fan that sucks air through the device, and a laser that detects the number and size of particles in the air passing through. It outputs three numbers: PM1.0, PM2.5, and PM10. These numbers refer to the size of the particles in microns, so PM10 is particles of 10 microns and smaller, PM2.5 is particles of 2.5 microns and smaller, and PM1.0 is particles of 1 micron and smaller. The numbers that the sensor spits out are the concentration of each of these sizes of particles in ug/m3.

The particulate matter sensor is most useful outdoors, as the level of particulates won't vary greatly indoors. We've got a separate tutorial (coming soon) that shows you how to build an enclosure for your Enviro+ and particulate matter sensor, and how to send data to the citizen science project Luftdaten.

MEMS microphone

Enviro+ has a tiny MEMS microphone that lets you record audio or detect noise levels. It's really good for monitoring levels of noise pollution, and we'll be adding support for this to our Enviro+ Python library soon.

0.96" colour LCD (160x80)

The tiny full-colour LCD on Enviro+ is a neat way to display live data from the sensors onboard. The display is an IPS display, so it's great quality for a display of its size, and it's driven by SPI so you can update it pretty quickly.

We've got an all-in-one example that displays sensor readings from Enviro+ on the LCD, with the live readings and a graph of the most recent values. We'll show you how to run this example later!

How to attach Enviro+ to your Pi

Enviro+ is the same size as a Raspberry Zero W, so it works great with them, but you can also use it with the larger Raspberry Pis too: the 3 A+ and 3 B+.

If you're using a Raspberry Pi Zero W, then you'll need to have a 40-pin male GPIO header soldered on or, if you're not to confident with soldering, you could use a Raspberry Pi Zero WH with pre-soldered header.

To attach Enviro+ to your Pi, push the female header on Enviro+ down onto the male header on the Raspberry Pi as far as it will go, making sure that you've lined all of the pins up correctly.

To be extra sure that your Enviro+ isn't going to wiggle around, if you're using a Pi Zero W, you can use a couple of 10mm M2.5 standoffs (https://shop.pimoroni.com/products/brass-m2-5-standoffs-for-pi-hats-black-plated-pack-of-2) to secure the board to your Pi.

Do this next part (connecting the particulate sensor) BEFORE powering on your Pi, as hot-plugging the sensor will cause you Pi to reboot!

If you have a particulate matter sensor, then you'll need to use the cable to connect it to your Enviro+. Plug one end of the cable into the connector on the particulate matter sensor. **It'll only fit easily one way round, so if you're having to force it then stop what you're doing and try it the other way round!** Plug the other end of the cable into the connector on the underside of Enviro+ (below the PM label on the top side), again **making sure that the cable is going in the right way round**!

Lastly, carefully peel the protective film off the LCD using the little green tab. You can leave it on, if you wish, to protect the display of course.

Installing the Enviro+ Python library

For this part of the tutorial, you'll need a micro-SD card that's been set up with the Raspbian operating system (https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/3), and to be connected to Wi-Fi (https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/6). It's also really handy to have a display, keyboard, and mouse connected to your Pi for these next bits.

If you're a beginner, it's best to use the full "Raspbian Buster with desktop and recommended software", as it's a little easier to set up and use, but you can also use Raspbian Lite, if you want. This tutorial will only cover how to set things up with full Raspbian Buster with desktop and recommended software, as it has most of the dependencies that we'll need to run the examples.

It's a great idea to start with a fresh install of Raspbian or, if not, then make sure that you run `sudo apt-get update` and `sudo apt-get upgrade` in the terminal to get everything up-to-date.

Open a terminal (press `control-alt-t` or find it in the Raspberry Pi menu) and then type the following:

```
git clone https://github.com/pimoroni/enviroplus-python
cd enviroplus-python
sudo ./install.sh
```

Once that's all done, type `sudo reboot` to reboot your Pi and apply the changes to the Pi's interfaces.

The install script enables I2C, SPI, and serial, disables the serial console, and enables the miniuart interface that the Pi uses to talk to the PMS5003 particulate sensor.

If you ever need to revert these changes to the UART and serial configuration, then you can run the uninstall script by typing `sudo ./uninstall.sh` within the `enviroplus-python` folder in the terminal, and then `sudo reboot` to reboot and apply the changes.

Running the examples

There are several examples in the `examples` folder within the `enviroplus-python` folder. In the terminal, type the following to look at the available examples:

```
cd examples
ls
```

When running the examples that follow, you can type `control-c` at any time to stop the example running.

BME280 weather sensor examples

Let's try the examples for each of the sensors on Enviro+ in turn, beginning with the weather example for the BME280. Type the following to run it:

```
python weather.py
```

You'll see the temperature, pressure, and relative humidity values being printed every second or so. Try touching the BME280 sensor and see how the temperature changes, and try breathing close to it to see how the humidity from your breath affects it.

```
Temperature: 23.35 *C
Pressure: 996.36 hPa
Relative humidity: 39.70 %
```

You'll probably notice that the temperature value is a bit higher than you'd expect (around 5 degrees C higher usually). This is because some heat is radiated from the Pi's CPU through to the Enviro+ board and hence the BME280 sensor. However, we can get the temperature of the Pi's CPU from its internal temperature sensor and use this value and a little bit of maths to compensate the temperature pretty accurately.

Try running the compensated temperature example now, and you should see much more realistic temperature values. Type the following to run it:

```
python compensated-temperature.py
```

There's a variable in that example code that you can change to tweak how much the temperature is compensated. We've calibrated it against an analog alcohol-type thermometer and, ideally, you should too.

Look for the line that says `factor = 2.25` and change this number a little either way until you get to around the right value. Making this factor smaller will shift your compensated temperature further down, and making it larger will shift it back up towards the uncompensated temperaure.

LTR-559 light and proximity example

This example displays the current ambient light level in Lux and a value (from zero to a couple of thousand) for the proximity, with larger numbers being closer proximity and *vice versa*.

Run the example by typing:

```
python light.py
```

You'll see light values in Lux and proximity values (that'll be zero if there's nothing close to the sensor) streaming out.

```
Light: 1497.45 Lux
Proximity: 00.00
```

Try moving your hand close to the sensor (where it says "LIGHT") and see how the values change. We'll use the proximity sensor as an input to trigger changes between the variables displayed on the LCD in a later example.

MICS6814 gas sensor example

The MICS6814 outputs resistance values in Ohms that correspond to the levels of three different types of gas: reducing, oxidising, and NH3 (more info above). Let's look at the numbers that come out of this sensor by running the gas example now. Type the following in the terminal:

```
python gas.py
```

Let it run for a while, and you'll see that the values creep up steadily. The sensor values take quite a while to stabilise, as the sensor warms up gradually.

This means that it's best to keep the sensor running and take readings continuously to get more consistent data. Each time you set it running, it'll take 10 minutes or longer to stabilise, and you should ignore these first values until the sensor has stabilised.

Once the readings have stabilised, try breathing close to the sensor. Which of the values do you see changing, if any? If you have a solvent-based pen like a Sharpie, then try holding it a couple of centimetres from the sensor (marked "GAS") and see how the volatiles affect the readings (**ensure that you don't inhale any solvents and that you do this in a well-ventilated area**).

PMS5003 particulate matter (PM) sensor example

Note that this part requires that you have a PMS5003 particulate sensor connected to your Enviro+. It's available here: https://shop.pimoroni.com/products/pms5003-particulate-matter-sensor-with-cable (https://shop.pimoroni.com/products/pms5003-particulate-matter-sensor-with-cable). And remember to make sure your Pi is powered off before connecting the PM sensor or it will reboot your Pi!

Let's run the `particulates.py` example and see what comes out! Type:

```
python particulates.py
```

You'll see some output similar to this:

```
PM1.0 ug/m3 (ultrafine particles):                             2
PM2.5 ug/m3 (combustion particles, organic compounds, metals): 3
PM10 ug/m3  (dust, pollen, mould spores):                      4
PM1.0 ug/m3 (atmos env):                                       2
PM2.5 ug/m3 (atmos env):                                       3
PM10 ug/m3 (atmos env):                                        4
>0.3um in 0.1L air:                                            663
>0.5um in 0.1L air:                                            168
>1.0um in 0.1L air:                                            24
>2.5um in 0.1L air:                                            2
>5.0um in 0.1L air:                                            0
>10um in 0.1L air:                                             0
```

The first six lines of output: PM1.0, PM2.5, PM10 are the most frequently used units of particulate matter concentration, given in "standard units" (the first three lines) and "environmental units" (the next three lines). We'll be honest... we're not quite sure what the difference between these units is but, in our testing, they've always read the same (let us know if you know the answer!)

You'll see that the PM2.5 reading is always equal to or greater than the PM1.0 reading, and the PM10 reading is greater to or equal to both the PM2.5 and PM1.0 readings because PM10, for example, includes all particles that are smaller than 10 microns.

The last six lines are the numbers of particles per tenth of a litre of air, in various size groupings, and again the >0.3um group should always be a bigger number than the others because it includes all particles of that size and greater, and so on...

0.96" LCD example

We've provided a simple example that shows you how to write text on a coloured background to the little LCD on Enviro+. The all-in-one example below has a much more complex use of the display to show text and a graph of values.

Let's try running the Hello, World! example on the LCD now. Type the following:

```
python lcd.py
```

All-in-one example

This all-in-one example displays all of the data from the sensors on Enviro+ (with the exception of the microphone) and lets you toggle through each of the variables using the light and proximity sensor.

Each variable on Enviro+ is displayed on its own screen, with the live reading at the top, and a graph below with values as a line graph and a coloured heatmap-type background with red-ish colours being higher values and blue-ish colours lower. The graph auto-scales to fit the lowest and highest values currently being displayed.

Type the following to run the example:

```
python all-in-one.py
```

Tap your finger to cycle through the different variables being displayed, and press `control-c` to stop it running again.

Using the Python library

The Enviro+ Python library has several parts to it and, in fact, most of the parts are in separate Python libraries that you have to import on their own. The installer (`install.sh` script) should install all of these other libraries for you.

Let's go through each part of the library (libraries) in turn.

Open a terminal and type `python` to start a Python prompt.

BME280 weather sensor

Let's import some bits we need from the `smbus2` (we'll need that to talk to the BME280) and the `bme280` libraries and create an instance of the BME280 class (it contains all of the functions to read data from the sensor) first. Type:

```
from smbus2 import SMBus
from bme280 import BME280

bus = SMBus(1)
bme280 = BME280(i2c_dev=bus)
```

Now let's read the temperature! Type:

```
print(bme280.get_temperature())
```

And now let's read the pressure and humidity values too. Type:

```
print(bme280.get_pressure(), bme280.get_humidity())
```

That's really all there is to the BME280. If you want to know *everything* it can do, then you can peer deep into the guts of the library here (https://github.com/pimoroni/bme280-python/blob/master/library/bme280/__init__.py).

LTR-559 light and proximity sensor

This library, `ltr559` , is pretty straightforward too, with just two different functions that you'll need to read data from the sensor. Still in the Python prompt, let's import the libraries we need again. Type:

```
import time
import ltr559
```

We're importing the `time` library too this time, as we're going to get a bit more fancy and use a `while` loop to constantly read values from the sensor. Type the following lines:

```
while True:
    print(ltr559.get_lux(), ltr559.get_proximity())
    time.sleep(1)
```

Try moving your hand or finger over the sensor and see how the values change. The code prints the light and proximity values with a one second pause after each print, until you press `control–c` to stop it.

MICS6814 analog gas sensor

Because the MICS6814 is essentially three different analog sensors in one and we have to read the data via an analog-to-digital converter, we wrapped it all up in a simple `gas` module within the Enviro+ Python library to make it nice and easy to read values from it.

Type the following to import the `gas` module from the `enviroplus` library, and again the `time` library:

```
import time
from enviroplus import gas
```

As we did for the last part, we'll read the values from the gas sensor inside a `while` loop so we can see how the values change. Type the following:

```
while True:
    readings = gas.read_all()
    print(readings)
    time.sleep(1.0)
```

If you want a specific value, e.g. just the reducing reading, you can do something like this:

```
while True:
    readings = gas.read_all()
    print(readings.reducing)
    time.sleep(1.0)
```

The same follows for `readings.oxidising` and `readings.nh3`.

Note how the readings are neatly formatted when you print them all together, but appear just as a floating point number when we specifically access each one.

PMS5003 particulate matter sensor

Since the PMS5003 can be used directly with the Pi's UART we've put this in its own library too. Let's import the `PMS5003` class and then create an instance of it to begin with. Type the following:

```
import time
from pms5003 import PMS5003

pms5003 = PMS5003()
```

We'll start by displaying all of the readings, then show you how to access them directly again, as we did for the gas sensor. Type the following:

```
readings = pms5003.read()
print(readings)
```

If this gives a `ReadTimeoutError` then just run it again and it should work.

Note how the readings are formatted nicely again, with descriptions. We'll now retrieve and display just the PM2.5 reading, in a `while` loop. Type the following:

```
while True:
    readings = pms5003.read()
    print(readings.pm_ug_per_m3(2.5))
    time.sleep(1)
```

Watch the readings for a little while, then press `control-c` to stop the code running.

We used the `.pm_ug_per_m3()` method and then passed in `2.5` to get the PM2.5 reading, but you could also have passed in `10` to get the PM10 reading or `1.0` to get the PM1.0 reading.

There's also a `.pm_per_1l_air()` method, to which you can pass the values `0.3`, `0.5`, `1.0`, `2.5`, `5`, or `10`, to get the particles per 0.1l of air (note that it's *per 0.1l air*).

0.96" LCD

The code that controls the little LCD is in the `st7735` library. The main way to create frames that can be shown on the display is using the Python Image Library (PIL), which allows you to do all sorts of things like draw text to the display, draw shapes, draw individual pixels, and even apply effects like blurs.

This part of the Enviro+ library is the most complex to use, in terms of functionality and amount of code. We'll just look at a simple example that fills the LCD with a coloured rectangle and writes some text on top of the rectangle.

Let's import the bits we need first:

```
import ST7735
from PIL import Image, ImageDraw, ImageFont
```

`PIL` is the Python Image Library, and the classes we've imported--- `Image`, `ImageDraw`, and `ImageFont` ---will be used to draw the image.

Next, let's set up the display object. You don't need to worry too much about what the options we pass in to the object do; they specify which pins are used, and you'll note that the rotation is `270` degrees because the default orientation of these displays is portrait.

Type the following:

```
disp = ST7735.ST7735(
    port=0,
    cs=1,
    dc=9,
    backlight=12,
    rotation=270,
    spi_speed_hz=10000000
)
```

Next, we'll initialise the display and create a new image that we can draw onto. Type the following:

```
disp.begin()

img = Image.new('RGB', (WIDTH, HEIGHT), color=(0, 0, 0))
draw = ImageDraw.Draw(img)
```

Notice that we're creating a new image, `img` , then passing that into the `Draw` class that we'll actually draw to. We're setting the background colour to `0, 0, 0` or black, but you could set this to any RGB colour you like.

The first thing we'll draw is our rectangle, a cyan rectangle, that spans the whole 160x80 pixels of the display.

```
rect_colour = (0, 180, 180)
draw.rectangle((0, 0, 160, 80), rect_colour)
```

The first tuple, `(0, 0, 160, 80)` , that we pass to the rectangle function specifies the top left x/y corner of the rectangle (the first two numbers), and the bottom right x/y corner (the second two numbers).

Let's make this next part a bit more interesting by writing the current temperature to the display. To do that, we'll need to import and set up the BME280 (you might have done this already, but we'll do it again just in case you stopped and started again here).

Type the following:

```
from smbus2 import SMBus
from bme280 import BME280

bus = SMBus(1)
bme280 = BME280(i2c_dev=bus)
```

Next, we'll write our temperature text to the display! To do this, we have to create a font instance, loading in a font at a specific size, and then specify where and in what colour the temperature should be drawn.

```
font_size = 18
font = ImageFont.truetype("fonts/Asap/Asap-Bold.ttf", font_size)

colour = (255, 255, 255)
temperature = "Temperature: {:.2f} *C".format(bme280.get_temperature())

x = 0
y = 0
draw.text((x, y), temperature, font=font, fill=colour)
```

We've used a placeholder to format the temperature nicely and to convert it from a float into a string all in one go: `"Temperature: {:.2f} *C"` . The `{:.2f}` specifies that it's a float (`f`) and to display two digits after the decimal point (`:.2`).

All we have to do now is to show the text on the display. If you're doing lots of different drawing steps, then it's best to do them all first and then call this next `.display()` function last of all.

Type the following:

```
disp.display(img)
```

Have a look at the examples in the ST775 Python library (https://github.com/pimoroni/st7735-python/blob/master/examples/) for some more examples of what you can do, like drawing images from files and even animating gifs on the LCD.

Next steps

That covers all of the functionality of Enviro+ and its Python library.

We've also got a Luftdaten tutorial that shows you how to turn your Enviro+ into a complete environmental monitoring station that contributes air quality data to citizen science! Click here to view the tutorial. (https://learn.pimoroni.com/tutorial/sandyj/enviro-plus-and-luftdaten-air-quality-station)

Shopping basket

Need something for this project? You can use the links below to add products to your Pimoroni Shop (http://shop.pimoroni.com/) basket for easy checkout.

	Enviro+ for Raspberry Pi (http://shop.pimoroni.com/products/enviro-plus) **£45.00**	Add
	PMS5003 Particulate Matter Sensor with Cable (http://shop.pimoroni.com/products/pms5003-particulate-matter-sensor-with-cable) **£24.90**	Add
	Raspberry Pi Zero WH (pre-soldered) (http://shop.pimoroni.com/products/raspberry-pi-zero-wh-with-pre-soldered-header) **£13.56**	Add

Want to checkout or change something? Click here to view your cart (http://shop.pimoroni.com/cart).

Tutorial

Intermediate

Raspberry Pi (/?tag=raspberry-pi), Pi Zero (/?tag=pi-zero)

Sandy Macdonald

sandy@pimoroni.com (mailto:sandy@pimoroni.com)
@sandyjmacdonald (https://twitter.com/sandyjmacdonald)
http://sandyjmacdonald.github.io (http://sandyjmacdonald.github.io)

Formerly, Sandy worked at the University of York, in the biology department, analysing data and telling people that they should have used more replicates. Now a fully-fledged crew member at Pimoroni - head of digital content - working on learning materials and digital chunterings. Find him on Twitter and most everywhere else, as sandyjmacdonald.

We are a company of Makers and Educators in Sheffield, UK.
We make the amazing Pibow (https://shop.pimoroni.com/collections/pibow) case for Raspberry Pi®.
Our Picade (https://www.kickstarter.com/projects/pimoroni/picade-the-arcade-cabinet-kit-for-your-raspberry-p) was the UKs 1st Kickstarter.
We're the UKs biggest Adafruit (https://shop.pimoroni.com/collections/adafruit) outlet.

Pimoroni Ltd. 2 Manton Street, Sheffield, S2 4BA, United Kingdom.