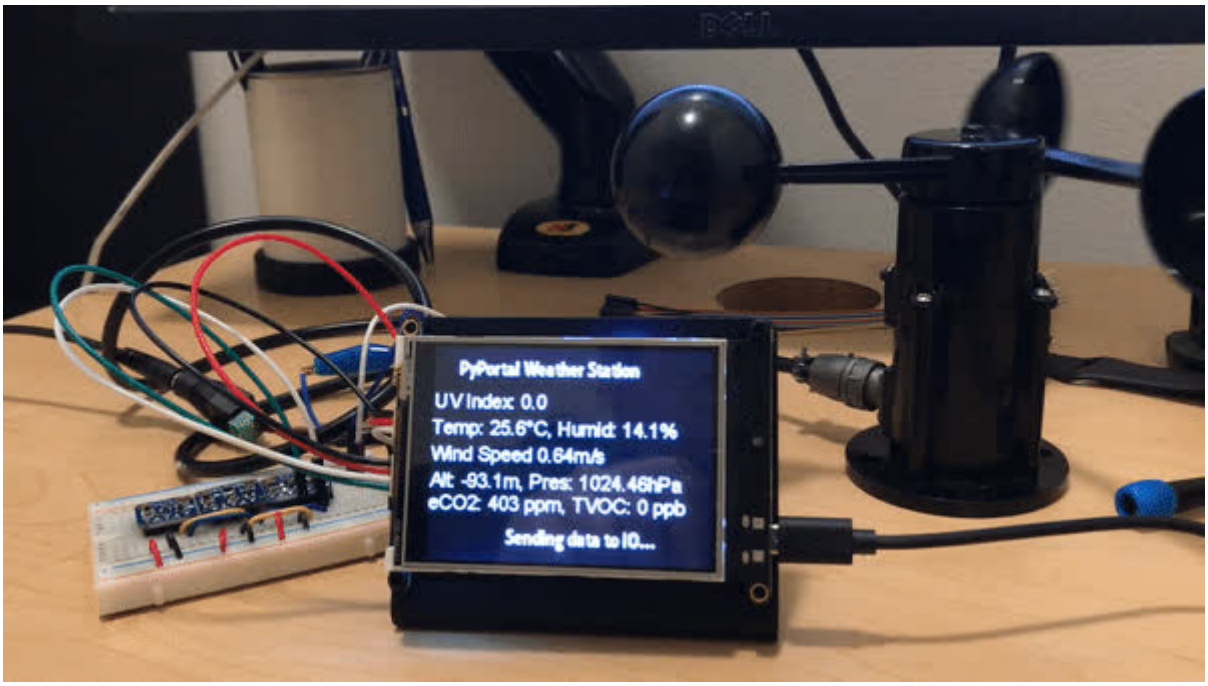
PyPortal IoT Weather Station

Created by Brent Rubell



https://learn.adafruit.com/pyportal-iot-weather-station

Last updated on 2025-04-15 11:24:51 AM EDT

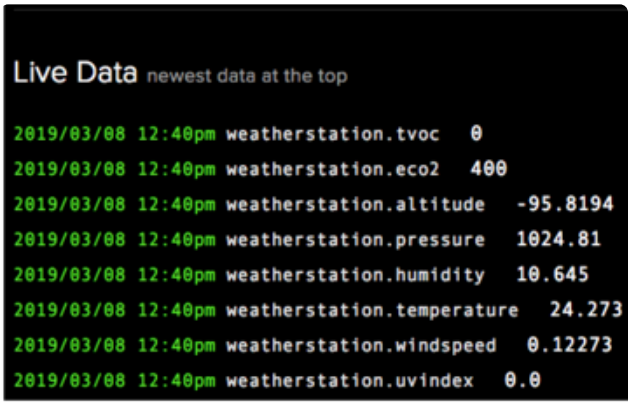
Table of Contents

Overview

Harness the power of nature with your PyPortal by building an internet-connected Weather Station!

This guide will guide you through building an internet-connected weather station, complete with sensors to measure everything from the wind speed to the amount of volatile organic compounds which are present in the air.

This weather-station doesn't just display temperature on the PyPortal's 3.2″ 320 x 240 color TFT screen - it also wirelessly sends data to Adafruit IO for real-time visualization and logging using the [easy-to-use Adafruit IO CircuitPython library (https://adafru.it/Ean)](https://adafru.it/Ean).



Adafruit IO

Adafruit IO is the easiest way to stream, log, and interact with your data. It's built from the ground up to be easy to use - we do the hard stuff so you can focus on the fun stuff.

Data such as relative humidity and light levels can be hard to visualize and quantify - Adafruit IO makes it simple. Send IO your data and it can store the data for up to thirty days. Want to visualize it? Display your data using Blocks such as charts, graphs, gauges, and more!

CircuitPython Code

Adafruit's CircuitPython is great for building Internet-of-Things projects. Using the Adafruit IO CircuitPython module (https://adafru.it/Ean), you can easily send data to Adafruit IO, receive data from Adafruit IO, and easily manipulate data with the powerful Adafruit IO API.

You can rapidly update your code without having to compile and store WiFi and API secret keys on the device. This means that there's no editing code and re-uploading whenever you move the PyPortal to another network - just update a file and you're set.

Prerequisite Guides

Learn all about the Adafruit PyPortal by checking out the guide. It has documentation and tour walk through of the pin outs and components. Libraries for Circuit Python and demo code is available.

Adafruit PyPortal Guide (https://adafru.it/Ecp)

If you're new to either Adafruit IO or CircuitPython, take a moment to walk through the following guides to get you started and up-to-speed:

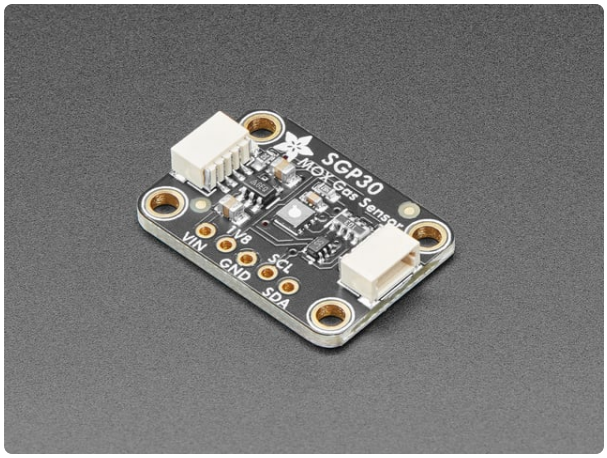
- Welcome to Adafruit IO (https://adafru.it/DZd)
- Welcome to CircuitPython (https://adafru.it/cpy-welcome)

Parts

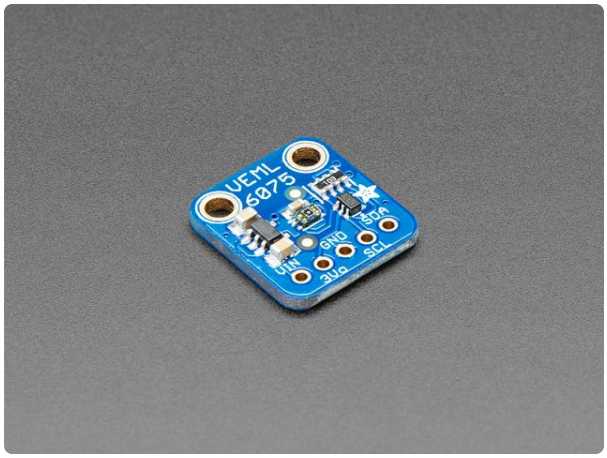
You will need the following parts for this guide:

[Adafruit PyPortal - CircuitPython Powered Internet Display]
PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...
https://www.adafruit.com/product/4116

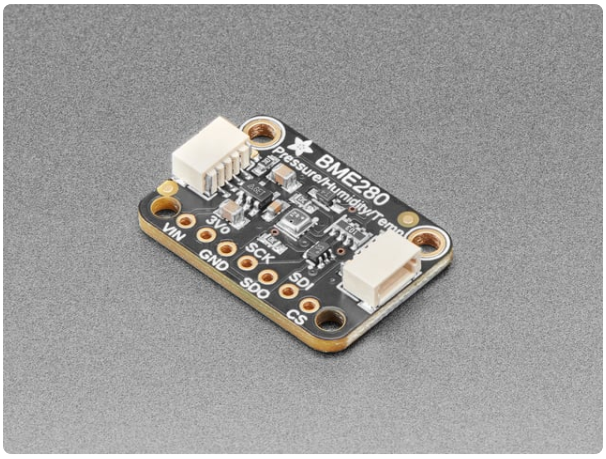
[Anemometer Wind Speed Sensor w/ Analog Voltage Output]
An anemometer is a device used for measuring wind speed, and is a common weather station instrument. This well made anemometer is designed to sit outside and measure wind speed with...
https://www.adafruit.com/product/1733



[Adafruit SGP30 Air Quality Sensor Breakout - VOC and eCO2]
Breathe easy with the SGP30 Multi-Pixel Gas Sensor, a fully integrated MOX gas sensor. This is a very fine air quality sensor from the sensor experts...
https://www.adafruit.com/product/3709



[Adafruit VEML6075 UVA UVB and UV Index Sensor Breakout]
This little sensor is a great way to add UVA and UVB light sensing to any microcontroller project. The VEML6075 from Vishay has both true UVA and UVB band light sensors and an...
https://www.adafruit.com/product/3964



Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor

Bosch has stepped up their game with their new BME280 sensor, an environmental sensor with temperature, barometric pressure and humidity! This sensor is great for all sorts...

https://www.adafruit.com/product/2652

1 x JST PH 4-Pin to Male Header — https://www.adafruit.com/product/3955

JST PH 4-Pin to Male Header Cable - I2C STEMMA Cable - 200mm

---

1 x JST PH 3-Pin to Male Header — https://www.adafruit.com/product/3893

JST PH 3-Pin to Male Header Cable - 200mm

---

1 x 9V Power Adapter — https://www.adafruit.com/product/63

9 VDC 1000mA regulated switching power adapter

---

1 x Female DC Power Adapter — https://www.adafruit.com/product/368

Female DC Power adapter - 2.1mm jack to screw terminal block

Materials

You'll need some extra supplies to finish this project. If you do not have them already, pick some up from Adafruit:

1 x USB Cable — https://www.adafruit.com/product/592

USB cable - USB A to Micro-B - 3 foot long

---

1 x Breadboard — https://www.adafruit.com/product/64

Half-size breadboard

---

1 x Hook-up Wire Spool Set — https://www.adafruit.com/product/1311

Hook-up Wire Spool Set - 22AWG Solid Core - 6 x 25 ft

---

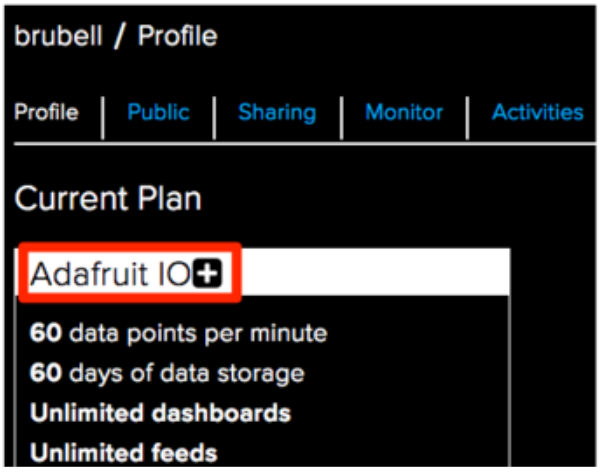
1 x Aligator Clip to Male Jumper Wire — https://www.adafruit.com/product/3255

Small Alligator Clip to Male Jumper Wire Bundle - 6 Pieces

Adafruit IO Setup

If you do not already have an Adafruit IO account set up, head over to io.adafruit.com (https://adafru.it/fH9) to link your Adafruit.com account to Adafruit IO.

This guide requires an active Adafruit IO Plus (IO+) account.



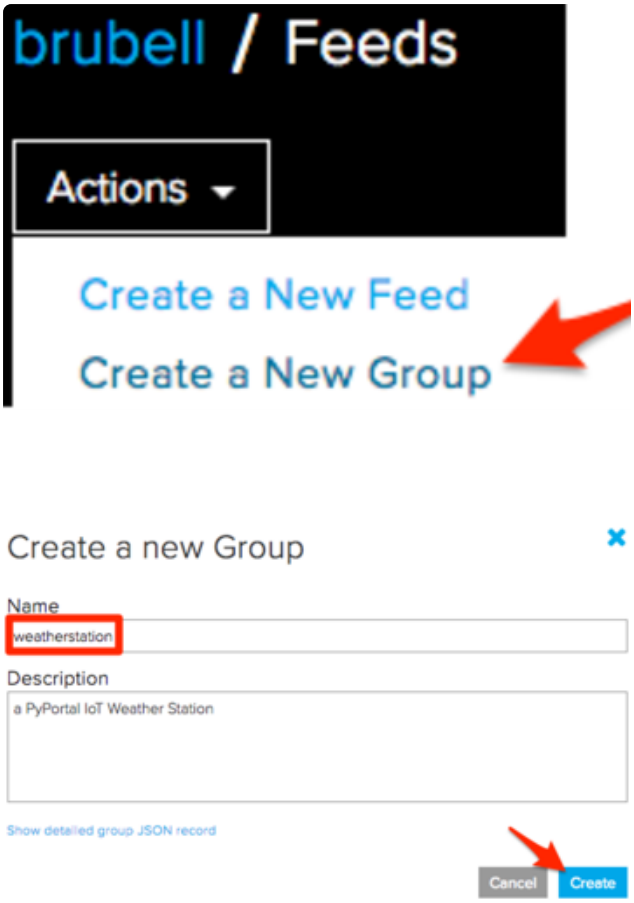
The large amount of feeds and higher data rate per minute (due to the amount of feeds) in this guide requires an Adafruit IO Account to have an active Adafruit IO Plus (IO+) subscription.

To check if you have an Adafruit IO Plus subscription: Navigate to your Adafruit IO Profile page (https://adafru.it/BmD) and **check your Current Plan**.

Interested in upgrading to IO Plus? Learn more about the upgraded, all systems go version of the Adafruit IO service here... (https://adafru.it/Eg3)

Set up a Group

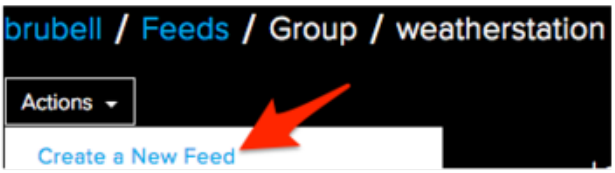
The first step is to create a new Adafruit IO Group to hold all the feeds associated with the PyPortal Weather Station (for more information about feeds, visit this guide (https://adafru.it/ioA)). We are using a group to reduce the amount of requests the PyPortal needs to make to Adafruit IO and to group all the feeds for this project in one place.



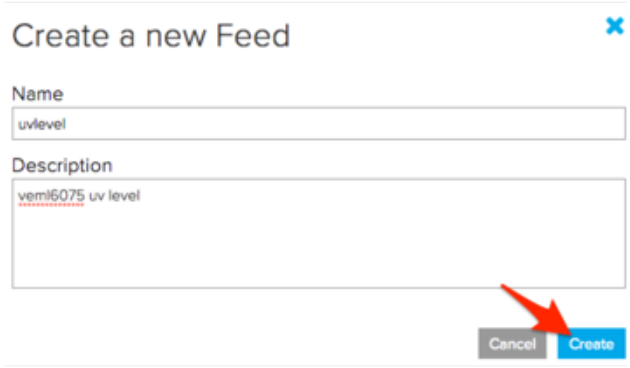
To create a group, **navigate to the** [feeds page (https://adafru.it/mxC)](https://adafru.it/mxC) **on Adafruit IO**. Then **click Actions -> Create New Group**, and **name the group weatherstation**.

Set up Feeds

The first step is to create a new Adafruit IO feed to hold the UV Level reported by the VEML6075 sensor. From the weatherstation group, **click Actions->Create New Feed**. **Name the new feed uvlevel**



To create a new feed, **navigate to the weatherstation group**.



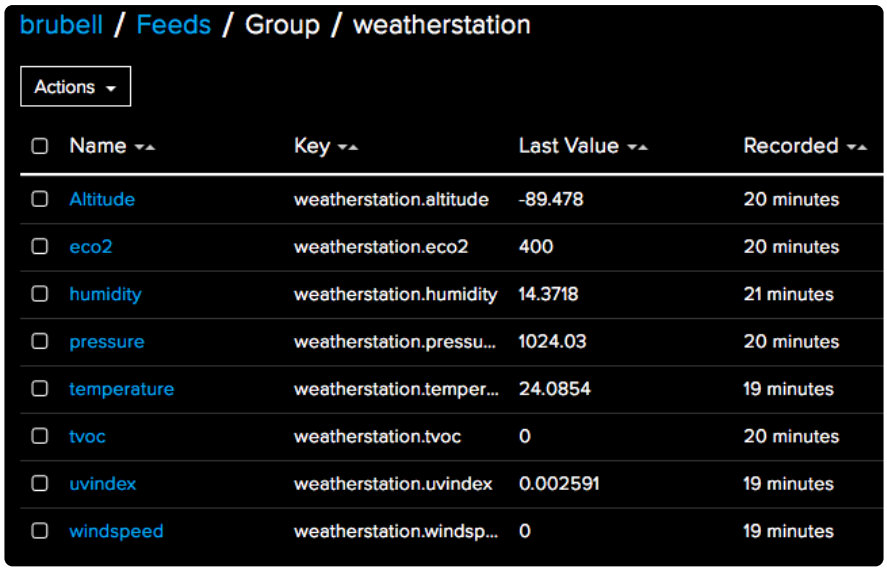
Click Actions -> Create New Feed.

Then, **name the new feed uvlevel**

Each Adafruit IO feed corresponds to a specific value sent by a sensor connected to your PyPortal. This project has a lot of incoming data. You'll need to **repeat the step above to create the following feeds within the weatherstatation group**:

1. **altitude**
2. **eco2**
3. **humidity**
4. **pressure**
5. **temperature**
6. **tvoc**
7. **uvindex**
8. **windspeed**

Once completed, the weatherstation group should look like the following:



Optional - Build a Dashboard

Dashboards allow you to visualize data and control Adafruit IO connected projects from any modern web browser. Blocks such as charts, sliders, and buttons are available to help you quickly get your IoT project up and running without the need for any custom code.

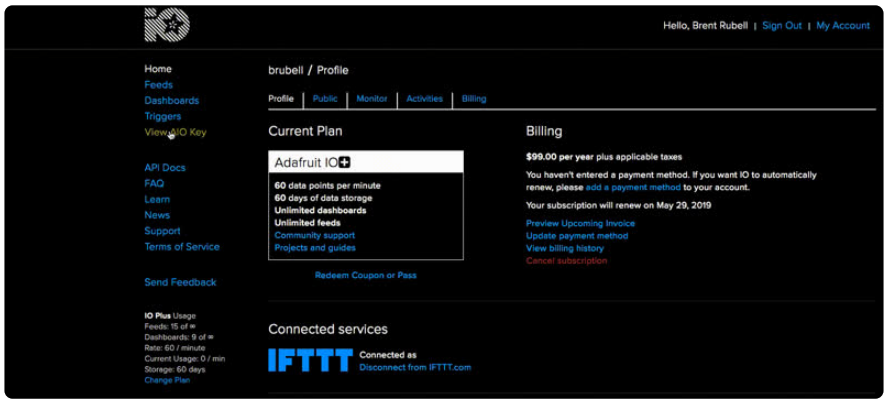
Since you'll be monitoring data using the PyPortal's built-in display, building a dashboard is optional for this project.

- Learn more building and using Adafruit IO Dashboards here... (https://adafru.it/f5m)

Obtain Adafruit IO Secret Key

You are also going to need your Adafruit IO username and secret API key.

Navigate to your profile and click the View AIO Key button to retrieve them. Write them down in a safe place, you'll need them for the next step.



Assembly and Wiring

Assembling the Anemometer

You wouldn't be building a proper weather station without an anemometer (https://adafru.it/Erj) - a weather station instrument used to measure wind speed. Place it outside and when the wind speed picks up, the anemometer's cups rotate making a rod spin internally. The faster the rod spins, the higher the voltage output from the anemometer's signal wire.

Connect the 4-Pin connector at the base of the anemometer to the 4-Pin weatherproof cable. Note that the notch (circled in red) should be oriented the same way for both the cable and the connector.

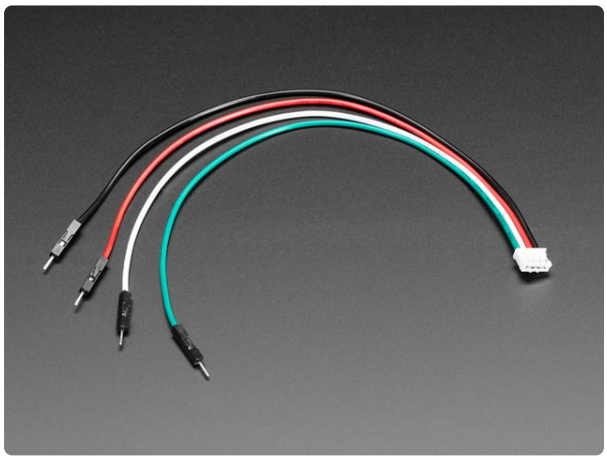
After connecting the cable, twist it clockwise a few times to secure it to the connector.

Next, you'll need to externally power the Anemometer. **You can not power it with your PyPortal - the PyPortal's output voltage is too low.**

Using an **external** power supply (7-24VDC) and a Female DC Power Adapter (http://adafru.it/368), **connect the anemometer's red wire to power**. Then, **connect the black wire to ground**.

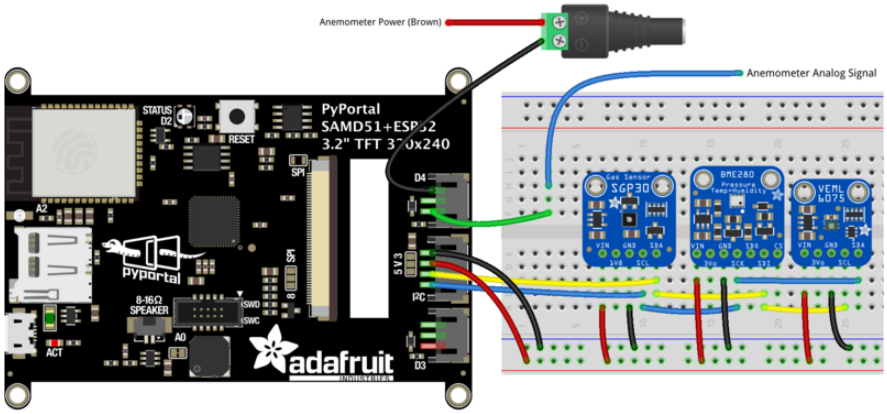
Wiring the PyPortal

These cables make wiring easy!



JST PH 2mm 4-Pin to Male Header Cable - I2C STEMMA Cable - 200mm
This cable will let you turn a JST PH 4-pin cable port into 4 individual wires with high-quality 0.1" male header plugs on the end. We're carrying these to match up with any...
https://www.adafruit.com/product/3955

STEMMA JST PH 2mm 3-Pin to Male Header Cable - 200mm
This cable will let you turn a JST PH 3-pin cable port into 3 individual wires with high-quality 0.1" male header plugs on the end. We're carrying these to match up with our...
https://www.adafruit.com/product/3893



Connecting the Anemometer

On the right side of the PyPortal, there is a connector labeled D4. This is a 3-Pin JST connector which can be used for analog sensors (like the anemometer!).

Using a JST PH 3-Pin to Male Header Cable (http://adafru.it/3893), **connect the blue (signal) wire from the anemometer to the signal wire (white) from the PyPortal**. If you are not comfortable with soldering - use a small alligator clip to male jumper wire (http://adafru.it/3255) to connect the anemometer's signal wire to the breadboard. Then, **connect the ground pin to the GND terminal of the female DC power adapter**.

Connecting Sensor Breakouts

There is a 4-Pin JST I2C connector in the center of the PyPortal which is also STEMMA compatible. We suggest using a JST PH 4-Pin to Male Header Cable (http://adafru.it/3955) to connect the I2C connector to a breadboard.

The sensors used in this guide connect to the PyPortal using I2C. You'll be connecting multiple I2C devices to the PyPortal using only two wires (for more information about I2C, check out our guide on the topic here (https://adafru.it/Erk)).

Make the following connections **between the PyPortal's I2C connector and the VEML6075**:

- **PyPortal SCL** to **VEML6075 SCL**
- **PyPortal SDA** to **VEML6075 SDA**
- **PyPortal VCC** to **VEML6075 VCC**
- **PyPortal GND** to **VEML6075 GND**

Make the following connections **between the SGP30 and the BME280:**

- **SGP30 VIN** to **BME280 VIN**
- **SGP30 GND** to **BME280 GND**
- **SGP30 SCL** to **BME280 SCK**
- **SGP30 SDA** to **BME280 SDI**

Make the following connection **between the BME280 and the VEML6075**:

- **BME280 VIN** to **VEML6075 VIN**
- **BME280 GND** to **VEML6075 GND**
- **BME280 SCK** to **VEML6075 SDA**
- **BME280 SDI** to **VEML6075 SCL**

CircuitPython Code

CircuitPython Library Installation

First make sure you are running the latest version of Adafruit CircuitPython (https://adafru.it/tBa) for your board.

COnnect the PyPortal to your computer via a known good data+power USB cable.

In Mu, you can connect to the board's serial REPL (https://adafru.it/Bec)so you are at the CircuitPython >>> prompt.

Now to get the PyPortal connected to Adafruit IO and the internet. To do this, you'll create a **settings.toml** text file.

Settings File Setup

If you have not yet set up a **settings.toml** file in your **CIRCUITPY** drive and connected to the internet using it, follow this guide and come back when you've successfully connected to the internet (https://adafru.it/Eao).

With a **settings.toml** file on your **CIRCUITPY** drive - add your Adafruit IO Username and Adafruit IO Key to the **settings.toml** file.

Your **settings.toml** file should look like this:

```
CIRCUITPY_WIFI_SSID="your-wifi-ssid"
CIRCUITPY_WIFI_PASSWORD="your-wifi-password"
ADAFRUIT_AIO_USERNAME="my_username"
ADAFRUIT_AIO_KEY="my_key"
```

After you finish editing **settings.toml**, make sure to **save the file** (**cmd/ctrl+s**).

Add CircuitPython Code and Project Assets

In the embedded code element below, click on the **Download Project Bundle** button, and save the .zip archive file to your computer.

Then, **uncompress the .zip file**, it will unpack to a folder named **pyportal_weatherstation**.

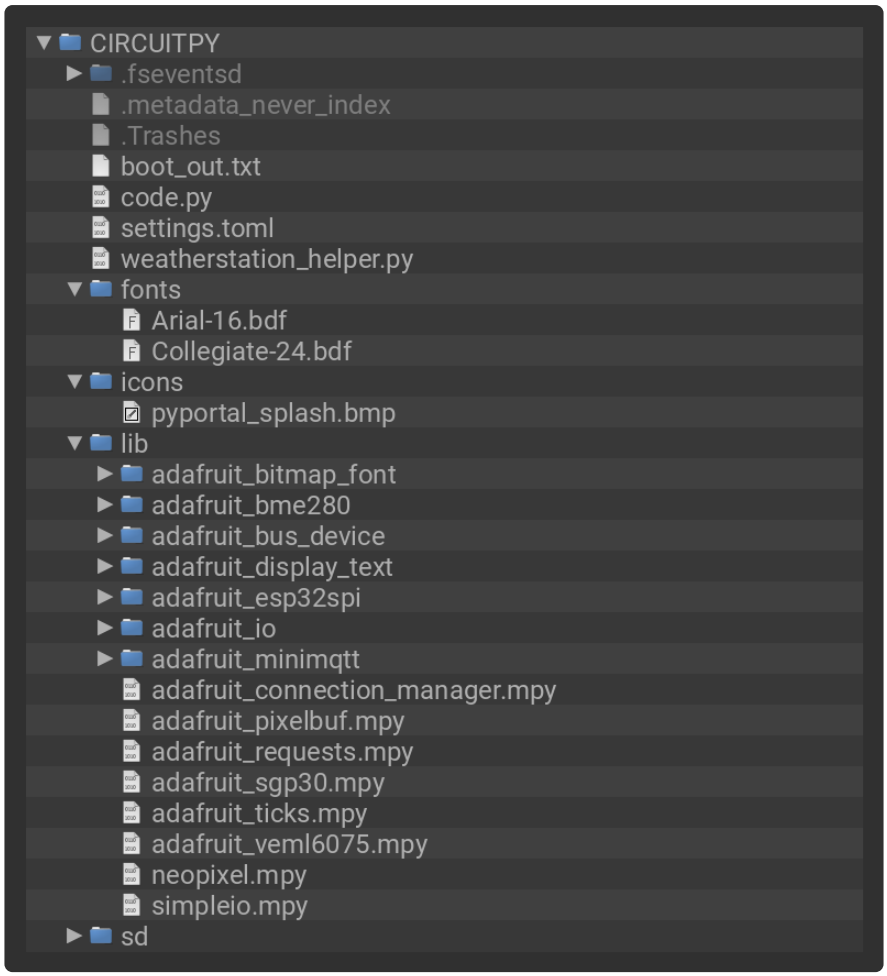
Copy the entire contents of the **pyportal_weatherstation** directory to your PyPortal **CIRCUITPY** drive including subdirectories.

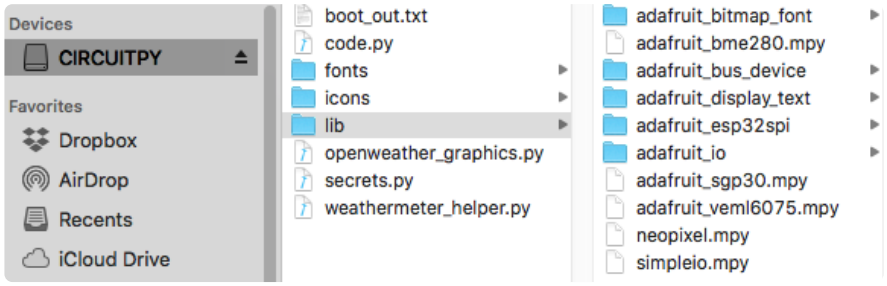
```
# SPDX-FileCopyrightText: 2019 Brent Rubell for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
PyPortal Weather Station
=============================================
Turn your PyPortal into a weatherstation with
Adafruit IO

Author: Brent Rubell for Adafruit Industries, 2019
"""

from os import getenv
import time
import board
import neopixel
import busio
import analogio
from simpleio import map_range
from digitalio import DigitalInOut

from adafruit_esp32spi import adafruit_esp32spi, adafruit_esp32spi_wifimanager
from adafruit_io.adafruit_io import IO_HTTP, AdafruitIO_RequestError

# sensor libs
import adafruit_veml6075
import adafruit_sgp30
from adafruit_bme280 import basic as adafruit_bme280

# weatherstation graphics helper
import weatherstation_helper

# rate at which to refresh the pyportal and sensors, in seconds
PYPORTAL_REFRESH = 30
```

```
# anemometer defaults
anemometer_min_volts = 0.4
anemometer_max_volts = 2.0
min_wind_speed = 0.0
max_wind_speed = 32.4

# Get WiFi details and Adafruit IO keys, ensure these are setup in settings.toml
# (visit io.adafruit.com if you need to create an account, or if you need your
Adafruit IO key.)
ssid = getenv("CIRCUITPY_WIFI_SSID")
password = getenv("CIRCUITPY_WIFI_PASSWORD")
aio_username = getenv("ADAFRUIT_AIO_USERNAME")
aio_key = getenv("ADAFRUIT_AIO_KEY")

if None in [ssid, password, aio_username, aio_key]:
    raise RuntimeError(
        "WiFi and Adafruit IO settings are kept in settings.toml, "
        "please add them there. The settings file must contain "
        "'CIRCUITPY_WIFI_SSID', 'CIRCUITPY_WIFI_PASSWORD', "
        "'ADAFRUIT_AIO_USERNAME' and 'ADAFRUIT_AIO_KEY' at a minimum."
    )

# PyPortal ESP32 Setup
esp32_cs = DigitalInOut(board.ESP_CS)
esp32_ready = DigitalInOut(board.ESP_BUSY)
esp32_reset = DigitalInOut(board.ESP_RESET)
spi = busio.SPI(board.SCK, board.MOSI, board.MISO)
esp = adafruit_esp32spi.ESP_SPIcontrol(spi, esp32_cs, esp32_ready, esp32_reset)
status_pixel = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.2)
wifi = adafruit_esp32spi_wifimanager.WiFiManager(esp, ssid, password,
status_pixel=status_pixel)

# Create an instance of the Adafruit IO HTTP client
io = IO_HTTP(ADAFRUIT_IO_USER, ADAFRUIT_IO_KEY, wifi)

# create an i2c object
i2c = busio.I2C(board.SCL, board.SDA)

# instantiate the sensor objects
veml = adafruit_veml6075.VEML6075(i2c, integration_time=100)
sgp30 = adafruit_sgp30.Adafruit_SGP30(i2c)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)
# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25

# init. the graphics helper
gfx = weatherstation_helper.WeatherStation_GFX()

# init. the ADC
adc = analogio.AnalogIn(board.D4)

# Set up Adafruit IO Feeds
print('Getting Group data from Adafruit IO...')
station_group = io.get_group('weatherstation')
feed_list = station_group['feeds']
altitude_feed = feed_list[0]
eco2_feed = feed_list[1]
humidity_feed = feed_list[2]
pressure_feed = feed_list[3]
temperature_feed = feed_list[4]
tvoc_feed = feed_list[5]
uv_index_feed = feed_list[6]
wind_speed_feed = feed_list[7]

def adc_to_wind_speed(val):
    """Returns anemometer wind speed, in m/s.
    :param int val: ADC value
    """
    voltage_val = val / 65535 * 3.3
```

```
    return map_range(voltage_val, 0.4, 2, 0, 32.4)

def send_to_io():
    # handle sending sensor data to Adafruit IO
    io.send_data(uv_index_feed['key'], uv_index)
    io.send_data(wind_speed_feed['key'], wind_speed)
    io.send_data(temperature_feed['key'], bme280_data[0])
    io.send_data(humidity_feed['key'], bme280_data[1])
    io.send_data(pressure_feed['key'], bme280_data[2])
    io.send_data(altitude_feed['key'], bme280_data[3])
    io.send_data(eco2_feed['key'], sgp_data[0])
    io.send_data(tvoc_feed['key'], sgp_data[1])

while True:
    print('obtaining sensor data...')
    # Get uv index from veml6075
    uv_index = veml.uv_index
    # Get eco2, tvoc from sgp30
    eCO2, TVOC = sgp30.iaq_measure()
    sgp_data = [eCO2, TVOC]
    # Store bme280 data as a list
    bme280_data = [bme280.temperature, bme280.humidity,
                   bme280.pressure, bme280.altitude]
    # Get wind speed
    wind_speed = adc_to_wind_speed(adc.value)
    # Display sensor data on PyPortal using the gfx helper
    print('displaying sensor data...')
    gfx.display_data(uv_index, bme280_data,
                     sgp_data, wind_speed)
    print('sensor data displayed!')
    try:
        try:
            print('Sending data to Adafruit IO...')
            gfx.display_io_status('Sending data to IO...')
            send_to_io()
            gfx.display_io_status('Data Sent!')
            print('Data sent!')
        except AdafruitIO_RequestError as e:
            raise AdafruitIO_RequestError('IO Error: ', e)
    except (ValueError, RuntimeError, ConnectionError, OSError) as e:
        print("Failed to get data, retrying\n", e)
        wifi.reset()
        continue
    time.sleep(PYPORTAL_REFRESH)
```

Before running the code, verify **CIRCUITPY** looks like the following:





Before continuing make sure your board's **lib** folder has the following files and folders copied over.

- **adafruit_bitmap_font**
- **adafruit_bme_280**
- **adafruit_bus_device**
- **adafruit_display_text**
- **adafruit_esp32spi**
- **adafruit_io**
- **adafruit_sgp30**
- **adafruit_veml6075**
- **neopixel**

- **simpleio**

From the Mu Editor, **click the Serial button** to open the REPL. The output should look like the following:

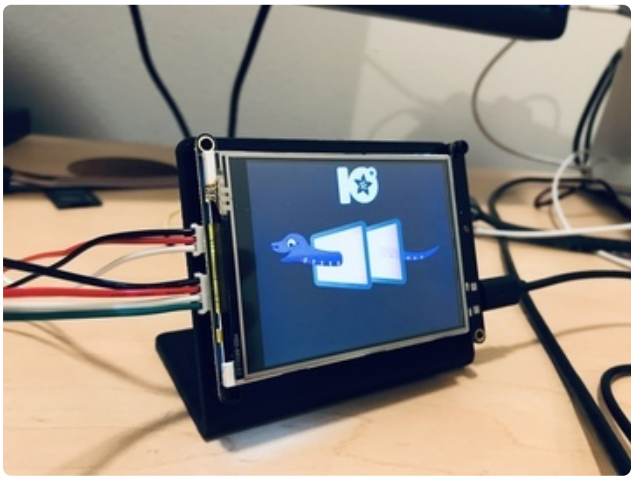
```
code.py output:
Set icon to  /icons/pyportal_splash.bmp
loading fonts...
setting up textareas...
Getting Group data from Adafruit IO...
obtaining sensor data...
displaying sensor data...
UV Index:  0.0110945
Temperature: 24.7 C
Humidity: 10.2%
Wind Speed: 0.090100 m/s
Altitude: -98.900 meters, Pressure: 1025.18 hPa
eCO2 = 400 ppm   TVOC = 0 ppb
```

Code Usage

The code will first attempt to initialize all of the sensors, libraries, and get the Adafruit IO weatherstation group.

If everything worked properly, you should see the PyPortal and Adafruit IO+ Logo appear on your PyPortal.

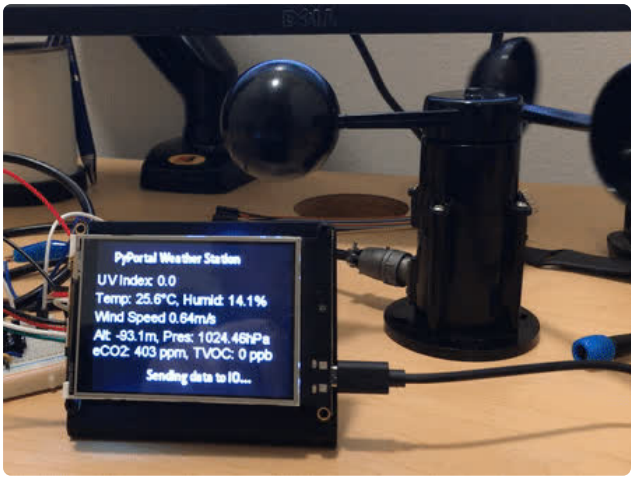
Shortly after the splash screen disappears, you should see the sensor text fields appear.



The code will first attempt to initialize all of the sensors, libraries, and get the Adafruit IO weatherstation group.

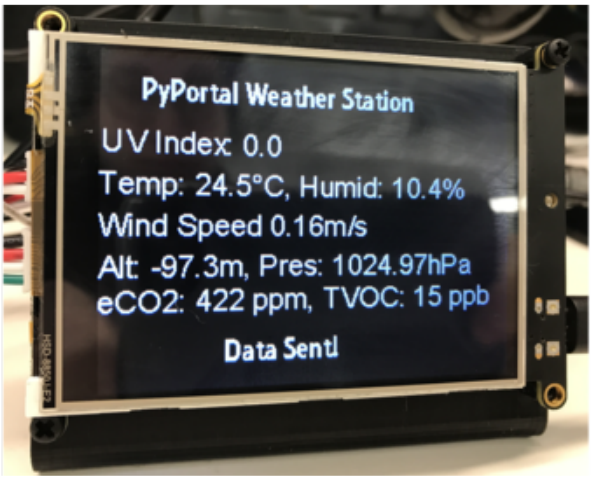
If everything worked properly, you should see the PyPortal and Adafruit IO+ Logo appear on your PyPortal.

Shortly after the splash screen disappears, you should see the sensor text fields appear.



Every 30 seconds, the PyPortal will obtain new data from the sensors and send it to Adafruit IO.

To change the rate at which the PyPortal samples and sends data (in seconds), modify the `PYPORTAL_REFRESH` variable.

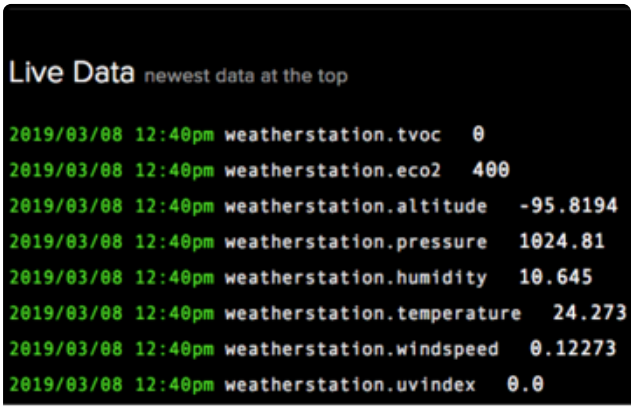


When the data is sent from the weather station to Adafruit IO, the status text area will change to **Data Sent!**

But how do we know the data was actually sent to Adafruit IO?

Adafruit IO Usage

While your PyPortal displays Data Sent!, how do you know the data reached the Adafruit IO service?



The monitor page on Adafruit IO displays a live view off incoming data (and errors!).

Navigate to [the Adafruit IO Monitor page (https://adafru.it/DOK)](). Only data received while the tab is open will be displayed on the page.

Wait for the PyPortal to send data to Adafruit IO. You should see incoming data appear underneath the Live Data heading.