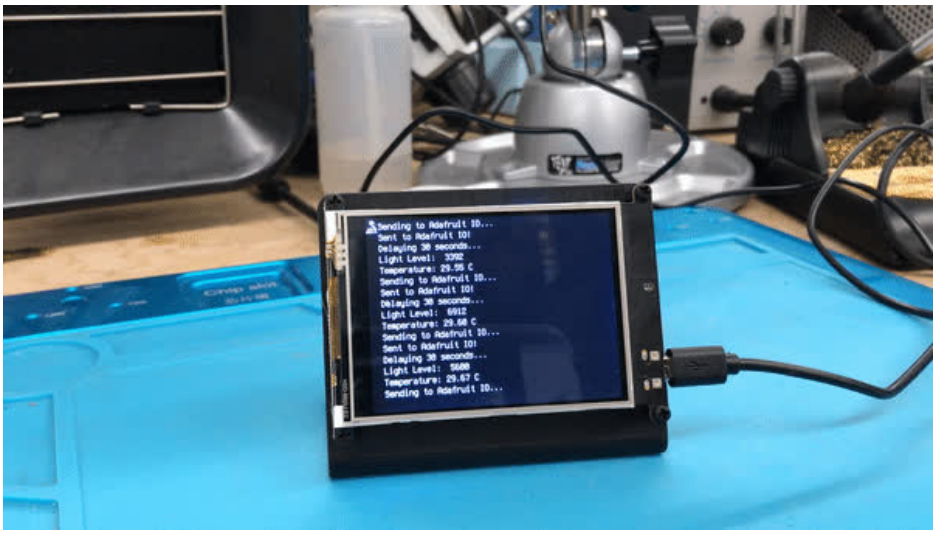
adafruit learning system

PyPortal IoT Data Logger with Analog Devices ADT7410, Adafruit IO and CircuitPython

Created by Brent Rubell

Last updated on 2019-04-04 07:02:37 PM UTC

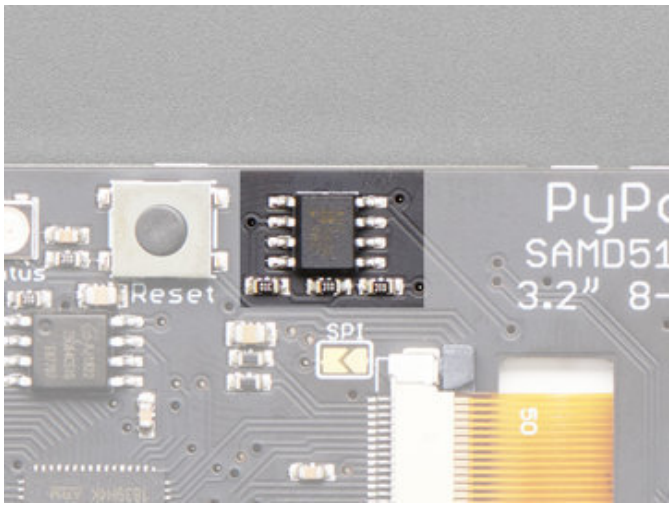
Overview



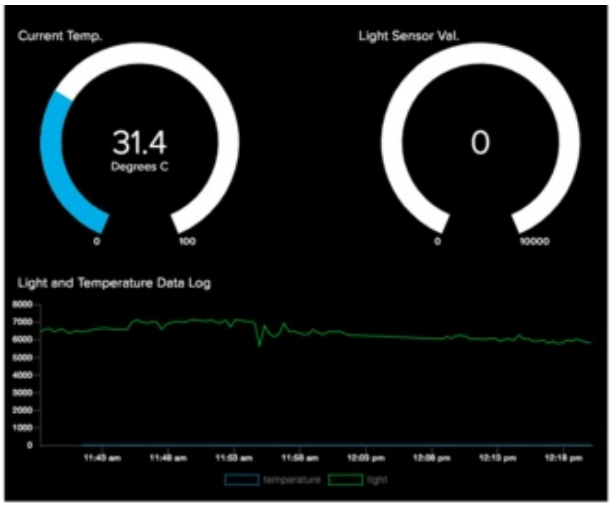
Connected your PyPortal to the internet, but do you want to do more with your data?

This guide will get your PyPortal communicating with our Internet of Things service - Adafruit IO using the easy-to-use Adafruit IO CircuitPython library (https://adafru.it/Ean).

Aside from being a fantastic internet-connected display, the PyPortal includes a light sensor and a temperature sensor. Using these two sensors, we can capture and send data to *the best data service in the world* - Adafruit IO (https://adafru.it/eIC) - for real-time data visualization and logging.



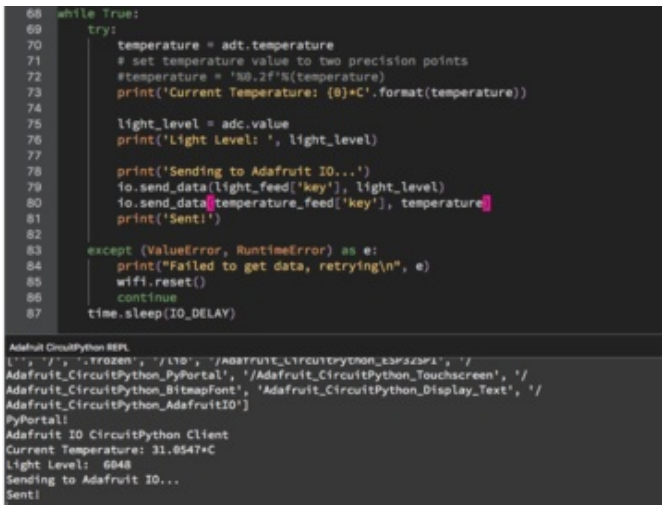
We'll be using the Analog Devices ADT7410 (https://adafru.it/EaC) built into the PyPortal to measure the ambient temperature over I2C



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 temperature and light levels can be hard to visualize and quantify - Adafruit IO makes it simple. Send IO your data and it can store and display it using 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.

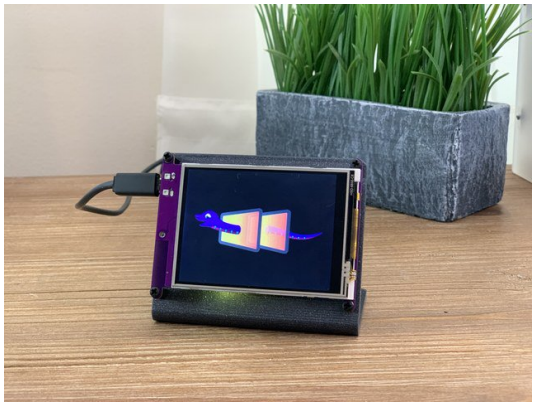
Prerequisite Guides

If you're new to 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 only need a PyPortal for this guide - you'll be using the temperature and light sensors included with the PyPortal. No other sensors or external circuitry required!



Adafruit PyPortal - CircuitPython Powered Internet Display

$54.95
OUT OF STOCK

OUT OF STOCK

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

ADD TO CART

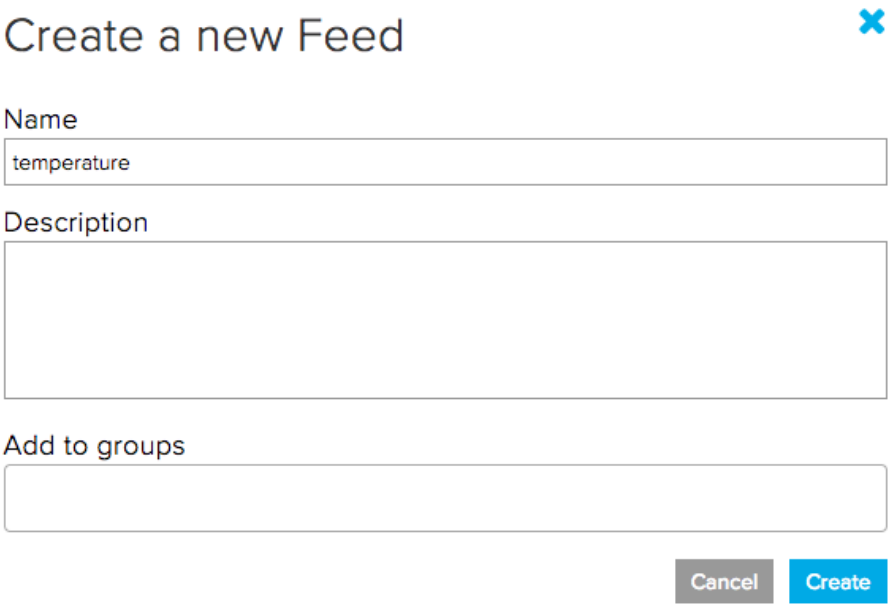
Adafruit IO Setup

Feed 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.

The first step is to create a new Adafruit IO feed to hold the data from the PyPortal's temperature sensor. Navigate to the feeds page (https://adafru.it/mxC) on Adafruit IO. Then click **Actions -> Create New Feed**, and name this feed **temperature**.

- If you do not already know how to create a feed, head over to Adafruit IO Basics: Feeds (https://adafru.it/ioA).



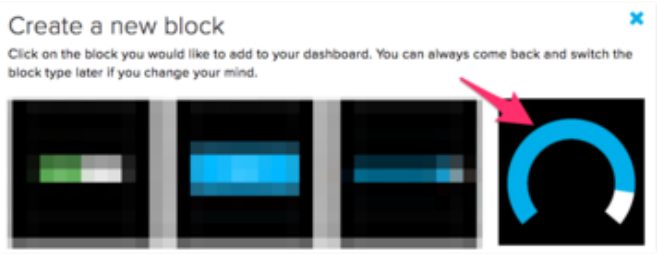
You'll also want a second feed to store the value of the light sensor - **create another feed named *light*.**

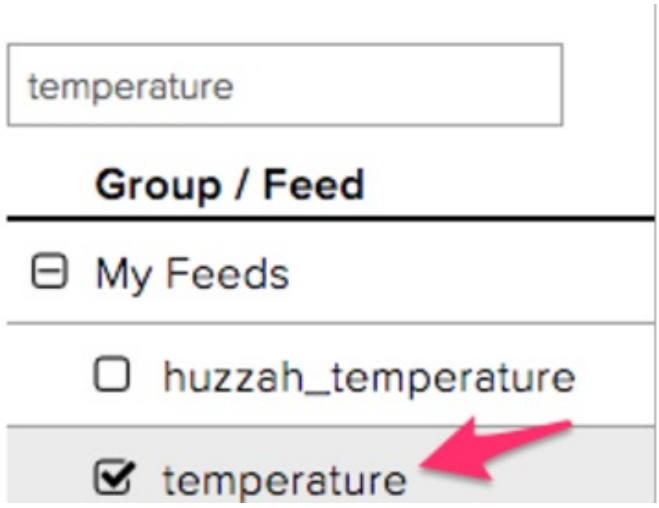
Dashboard Setup

Next, you'll create a dashboard to display the values from the feeds you created.

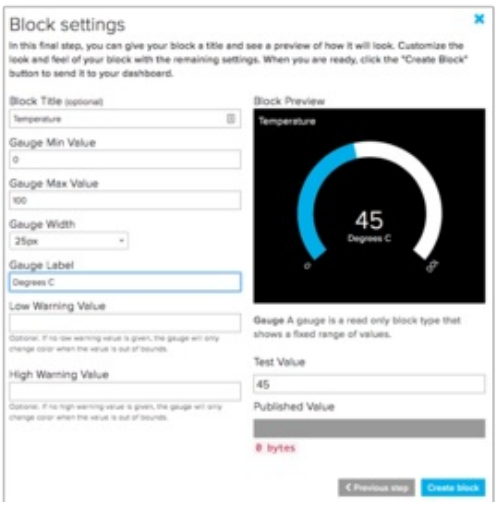
- If you do not know how to create or use Dashboards in Adafruit IO, head over to the Adafruit IO Basics: Dashboards (https://adafru.it/f5m) guide.

From your dashboard, **select the Gauge block.**





Select the *temperature* feed

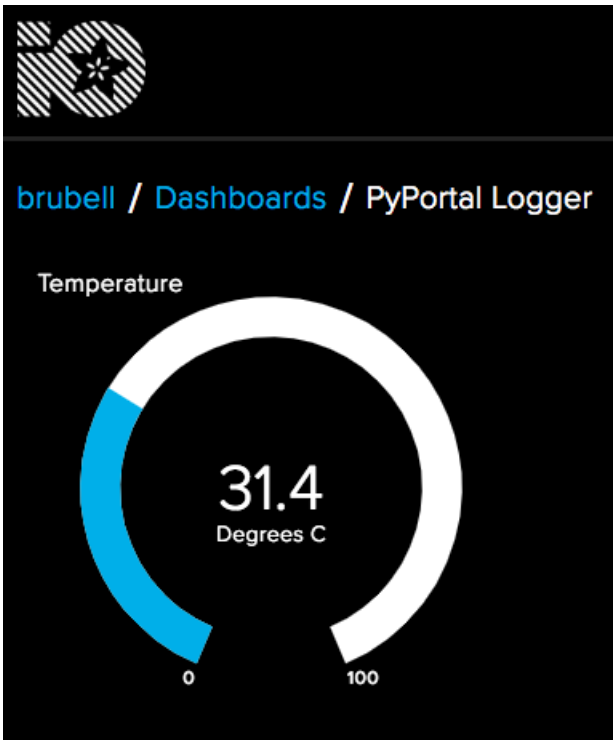


In the Block Settings step, set the **Block Title** to **Temperature**, set the **Gauge Min/Max Values** to the upper and lower temperature thresholds you want to measure.

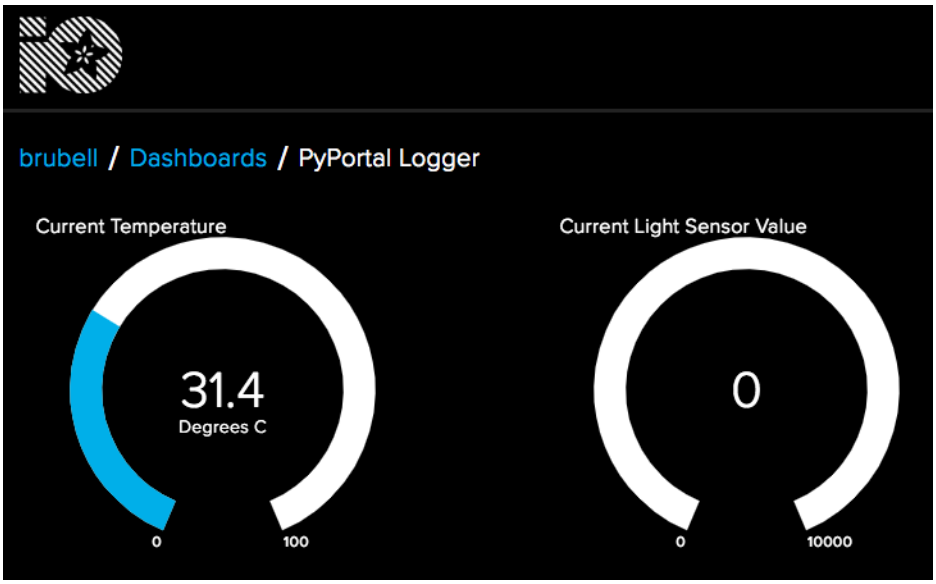
You can label the gauge by setting the **Gauge Label** - this example assumes temperature is to be measured in Degrees C.

Uncomfortably hot or cold? You can optionally set the gauge change color to warn you if the temperature goes above (or below) a certain value.

After adding the gauge element, your dashboard will look like the following:

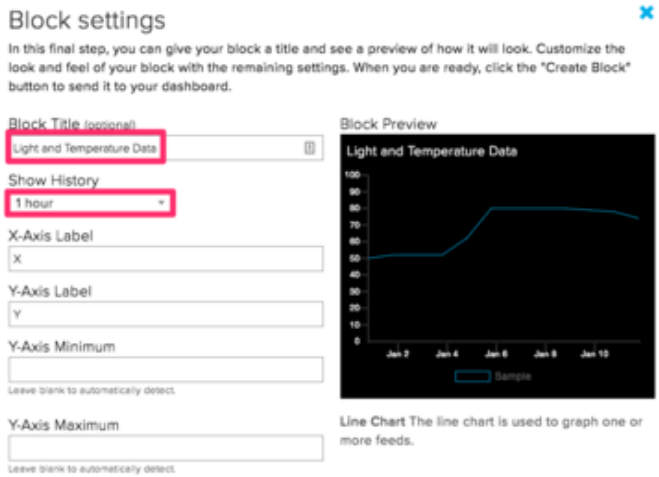


Next, **create another gauge block for the *light feed***.



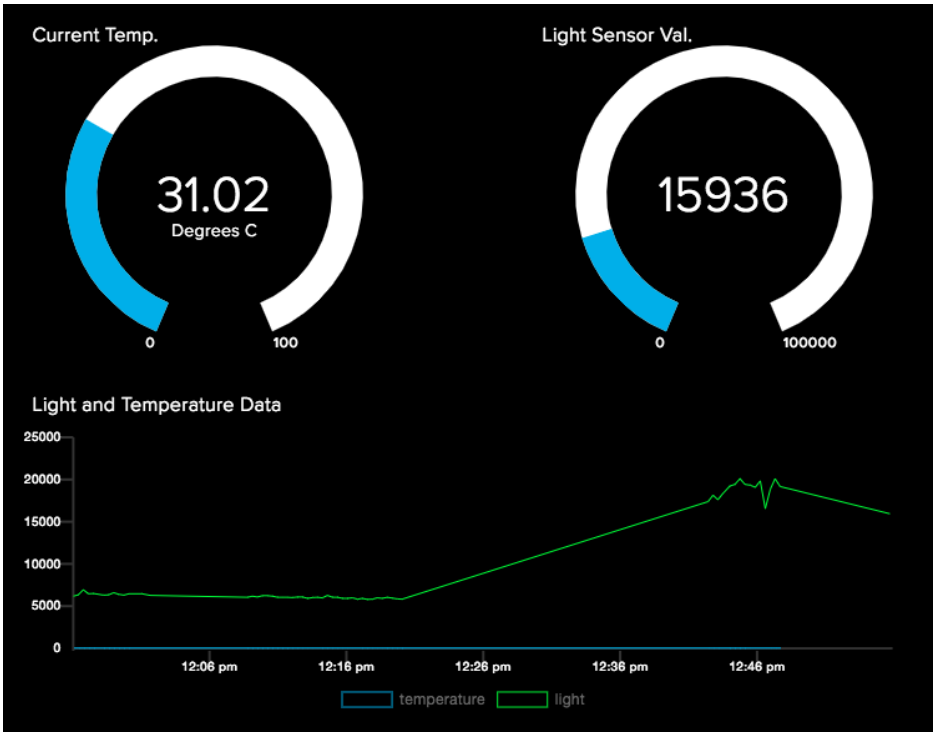
While displaying the current values of the temperature is useful, Adafruit IO stores data so you can monitor how it changes a long period of time.

To do this, we'll use the **Line Chart** block - a block used to graph one or more feeds - and set it up to show both the light value and the temperature value over a long period of time.



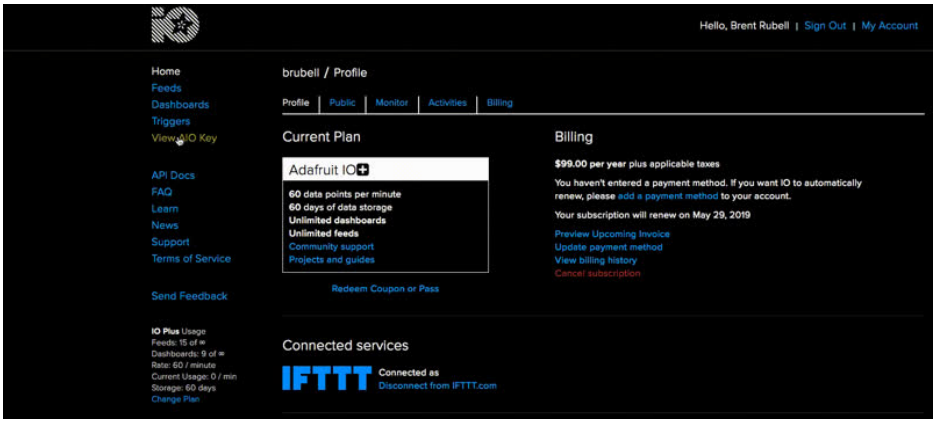
Create a new block and check the light feed, and then the temperature feed. You can select up to five different feeds to display using this block.

Your finished dashboard should look like the following:



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.



CircuitPython Code

CircuitPython Library Installation

The Adafruit_CircuitPython_AdafruitO module (https://adafru.it/Ean) allows you to easily write CircuitPython code which can interact with Adafruit IO.

You'll need to install the Adafruit CircuitPython Adafruit IO (https://adafru.it/Ean) and the Adafruit CircuitPython ADT7410 (https://adafru.it/DPz) libraries on your PyPortal.

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

Next you'll need to install the necessary libraries to use the hardware--carefully follow the steps to find and install these libraries from Adafruit's CircuitPython library bundle (https://adafru.it/zdx). For example the Circuit Playground Express guide has a great page on how to install the library bundle (https://adafru.it/C9M) for both express and non-express boards.

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

- **Adafruit_CircuitPython_AdafruitIO**
- **Adafruit_CircuitPython_ESP32SPI**
- **adafruit_adt7410.mpy**
- **adafruit_bus_device**

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

Once you have your CircuitPython libraries installed, let's get your PyPortal connected to Adafruit IO and the internet. To this, you'll create a *secrets* file.

Secrets File Setup

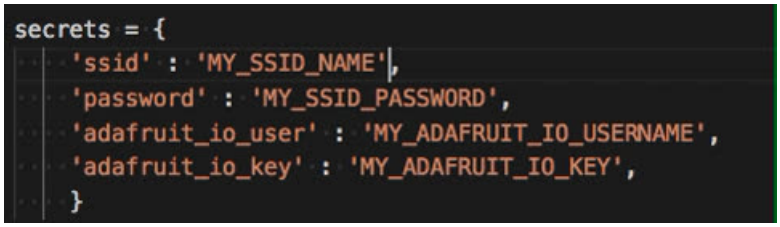
If you have not yet set up a secrets.py file in your `CIRCUITPY` drive and connected to the internet using it, follow this guide and come back when you've successfully connected to the internet (https://adafru.it/Eao).

Next, you should add your Adafruit IO Username and Adafruit IO Key to the **secrets.py** file.

Your **secrets.py** file should look like this:

```
secrets = {
    'ssid' : 'home ssid',
    'password' : 'my password',
    'timezone' : "America/New_York", # http://worldtimeapi.org/timezones
    'aio_username' : 'MY_ADAFRUIT_IO_USERNAME',
    'aio_key' : 'MY_ADAFRUIT_IO_KEY',
    }
```

Change `MY_ADAFRUIT_USERNAME` and `MY_ADAFRUIT_IO_KEY` **to your Adafruit IO username and the secret key.**



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

Code

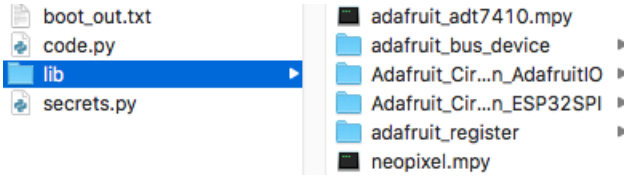
Using a text-editor (we like Mu (https://adafru.it/Be6) since it has a built-in serial REPL), copy the code below to your CircuitPython board,and save it as **code.py.**

```
"""
PyPortal IOT Data Logger for Adafruit IO

Dependencies:
    * CircuitPython_ADT7410
        https://github.com/adafruit/Adafruit_CircuitPython_ADT7410

    * CircuitPython_AdafruitIO
        https://github.com/adafruit/Adafruit_CircuitPython_AdafruitIO
"""
import time
import board
import busio
from digitalio import DigitalInOut
from analogio import AnalogIn

# ESP32 SPI
from adafruit_esp32spi import adafruit_esp32spi, adafruit_esp32spi_wifimanager

# Import NeoPixel Library
import neopixel

# Import Adafruit IO REST Client
from adafruit_io.adafruit_io import RESTClient, AdafruitIO_RequestError

# Import ADT7410 Library
import adafruit_adt7410

# Timeout between sending data to Adafruit IO, in seconds
IO_DELAY = 30

# Get wifi details and more from a secrets.py file
try:
    from secrets import secrets
except ImportError:
    print("WiFi secrets are kept in secrets.py, please add them there!")
    raise

# 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_light = neopixel.NeoPixel(board.NEOPIXEL, 1, brightness=0.2)
wifi = adafruit_esp32spi_wifimanager.ESPSPI_WiFiManager(esp, secrets, status_light)

# Set your Adafruit IO Username and Key in secrets.py
# (visit io.adafruit.com if you need to create an account,
# or if you need your Adafruit IO key.)
ADAFRUIT_IO_USER = secrets['aio_username']
ADAFRUIT_IO_KEY = secrets['aio_key']

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

try:
    # Get the 'temperature' feed from Adafruit IO
    temperature_feed = io.get_feed('temperature')
    light_feed = io.get_feed('light')
except AdafruitIO_RequestError:
    # If no 'temperature' feed exists, create one
    temperature_feed = io.create_new_feed('temperature')
    light_feed = io.create_new_feed('light')

# Set up ADT7410 sensor
i2c_bus = busio.I2C(board.SCL, board.SDA)
adt = adafruit_adt7410.ADT7410(i2c_bus, address=0x48)
adt.high_resolution = True

# Set up an analog light sensor on the PyPortal
adc = AnalogIn(board.LIGHT)

while True:
    try:
        light_value = adc.value
        print('Light Level: ', light_value)

        temperature = adt.temperature
        print('Temperature: %0.2f C'%(temperature))

        print('Sending to Adafruit IO...')

        io.send_data(light_feed['key'], light_value)
        io.send_data(temperature_feed['key'], temperature, precision=2)
        print('Sent to Adafruit IO!')
    except (ValueError, RuntimeError) as e:
        print("Failed to get data, retrying\n", e)
        wifi.reset()
        continue
    print('Delaying {0} seconds...'.format(IO_DELAY))
    time.sleep(IO_DELAY)
```

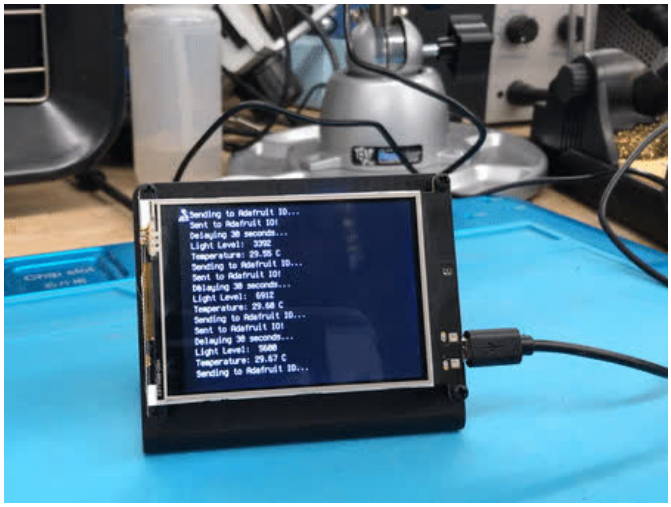
If you run into any errors, such as "ImportError: no module named `adafruit_display_text.label`" be sure to update your libraries to the latest release bundle!

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



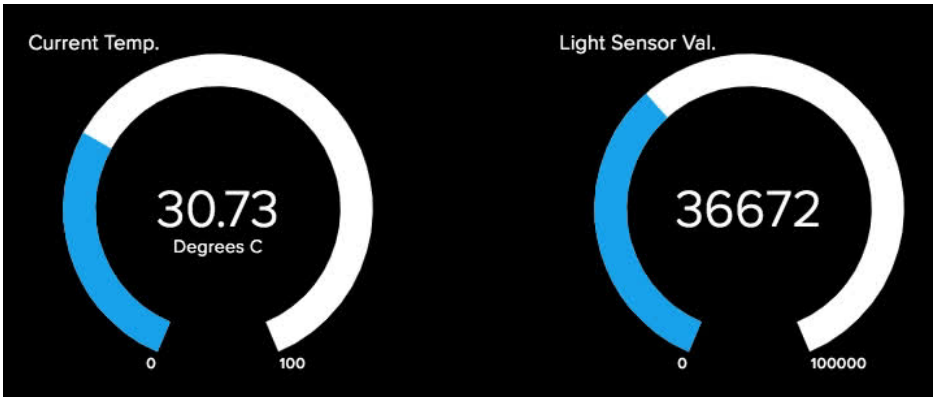
From the Mu Editor, **click the Serial button** to open the REPL. You should see the REPL displaying the temperature and light values from the PyPortal's onboard sensors, and sending the data to Adafruit IO:

```
Light Level:  37376
Temperature: 31.60 C
Sending to Adafruit IO...
Sent to Adafruit IO!
Delaying 30 seconds...
```

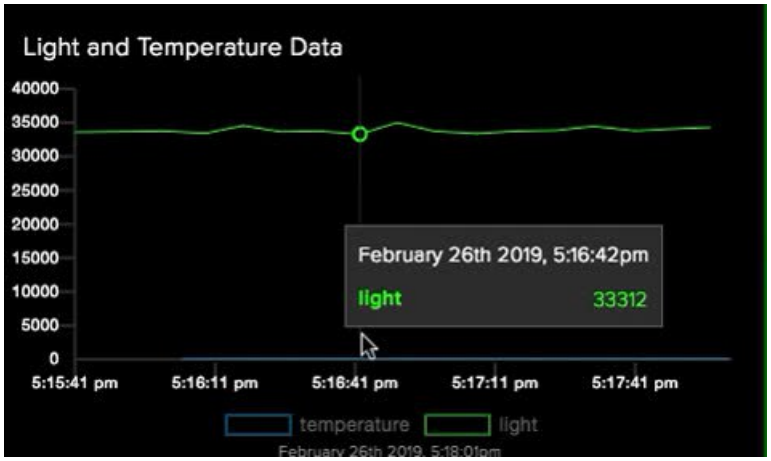


Your PyPortal should also display what is currently printed to the REPL.

Open the Adafruit IO Dashboard you created earlier. Notice that the fill and values of the gauges change as values are sent from your PyPortal to Adafruit IO.



Then, leave the PyPortal running for a while and come back later to see new data appear on the line graph.



Want to send data less frequently to Adafruit IO?

- Too much precision? Too little?
  -

Taking it Further

For more examples and ideas - check out the Adafruit IO category on the Adafruit Learning System (https://adafru.it/iRB).

On the CircuitPython Adafruit IO library repository, you'll find we've included lots of examples for sending data (https://adafru.it/Eap), interacting with your CircuitPython board's digital inputs (https://adafru.it/Eaq) and outputs (https://adafru.it/Ear), feed (https://adafru.it/Eas)/group (https://adafru.it/Eat)/data (https://adafru.it/Eap) interaction, and *more* (https://adafru.it/Eau).