Grill Thermometer Dashboard

Created by John Park

https://learn.adafruit.com/grill-thermometer-dashboard

Last updated on 2025-02-09 11:55:56 PM EST

Table of Contents

Overview

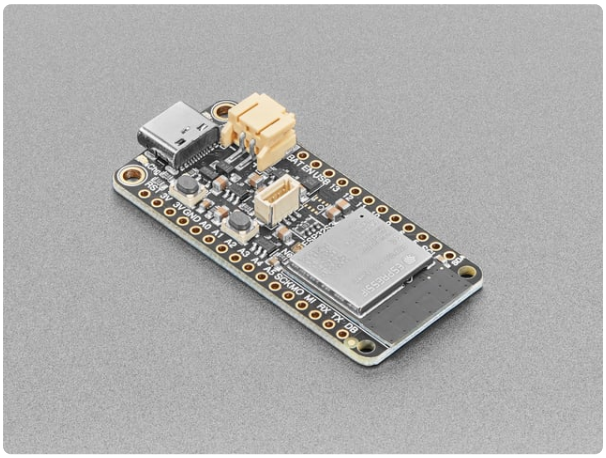
Cook your meats and non-meats with ultimate instrumentation! The Grill Thermometer Dashboard lets you monitor common Bluetooth LE temperature probes over the internet -- thanks to the dual BLE/WiFi capabilities of the Feather ESP32-S3 board!

Keep an eye on the grill temperatures from the comfort of your air conditioned home on your computer -- or on your phone while you soak in the pool! Any internet capable device can be used to monitor your Adafruit IO web-based dashboard.

Simple CircuitPython code allows you to grab the temperatures reported over Bluetooth and pass them along to an Adafruit IO dashboard via MQTT on your WiFi network.

What time is it? Grilled foods time.

Parts

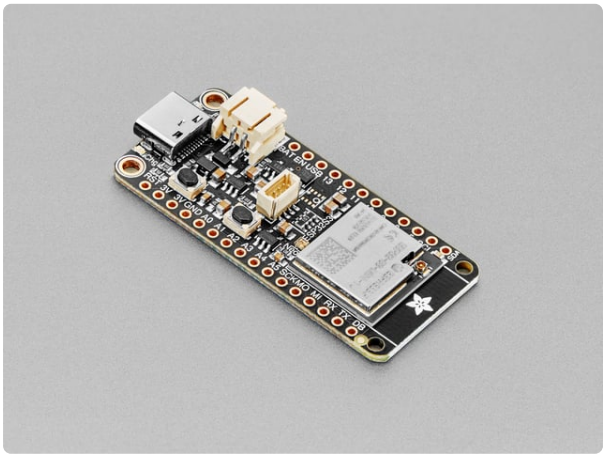


Adafruit ESP32-S3 Feather with STEMMA QT / Qwiic

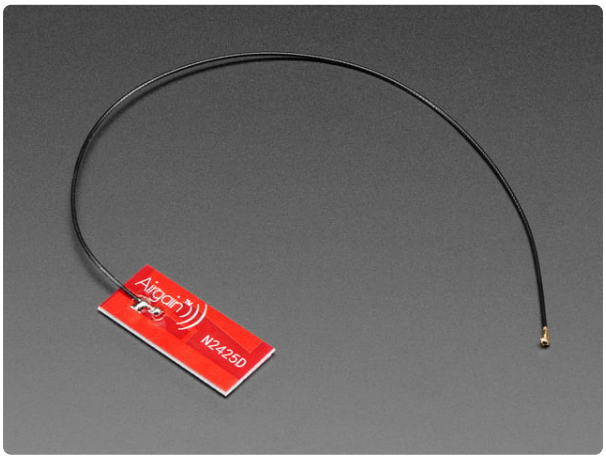
The ESP32-S3 has arrived in Feather format - and what a great way to get started with this powerful new chip from Espressif! With dual 240 MHz cores, WiFi and BLE support, and native...

https://www.adafruit.com/product/5323

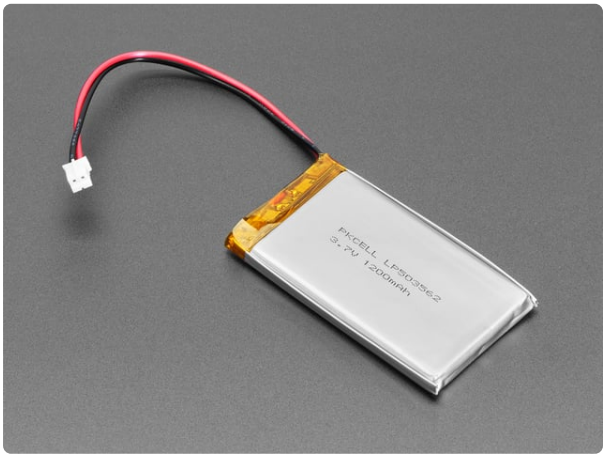
or



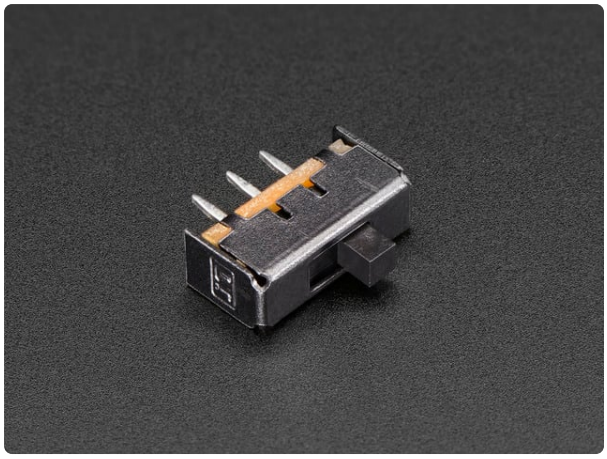
[Adafruit ESP32-S3 Feather 8MB with w.FL Antenna]
The ESP32-S3 has arrived in Feather format - and what a great way to get started with this powerful new chip from Espressif! With dual 240 MHz cores, WiFi and BLE support, native...
https://www.adafruit.com/product/5885



[WiFi Antenna with w.FL / MHF3 / IPEX3 Connector]
That's one slim cellular antenna! At about 220mm long from tip to tip and with a cable thickness of just 0.8mm, this 2.4GHz WiFi or BLE antenna is slim, compact, and...
https://www.adafruit.com/product/5445



[Lithium Ion Polymer Battery - 3.7v 1200mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/258



Breadboard-friendly SPDT Slide Switch
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805

Bluetooth BBQ Thermometer

You'll need a Bluetooth BBQ thermometer such as the one seen here (https://adafru.it/1a54) that uses the iBBQ BLE service -- most manufacturers don't advertise which service they use, but this is a very common one.

InkBird and EasyBBQ from PyleUSA are brands to look for.

CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (https://adafru.it/BeZ) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

> Verify which version of the Feather ESP32-S3 you have before downloading CircuitPython below!

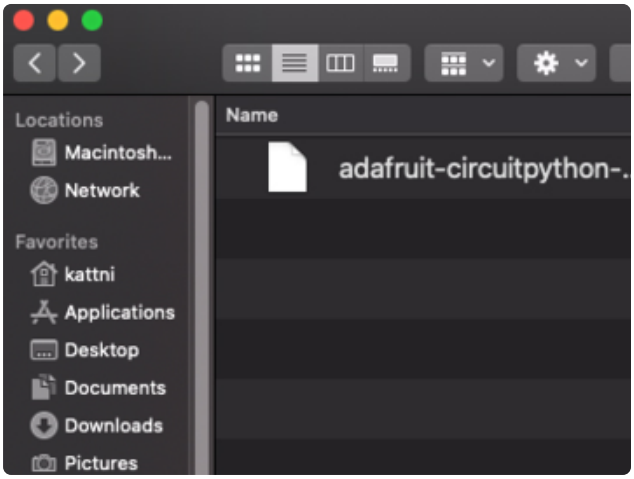
Below are links to CircuitPython for the **Feather ESP32-S3 8MB No PSRAM** and the **Feather ESP32-S3 4MB Flash 2MB PSRAM.** Be sure to choose the one that matches your board.

Download the latest version of CircuitPython for the Feather ESP32-S3 8MB Flash No PSRAM via circuitpython.org

https://adafru.it/ZCa

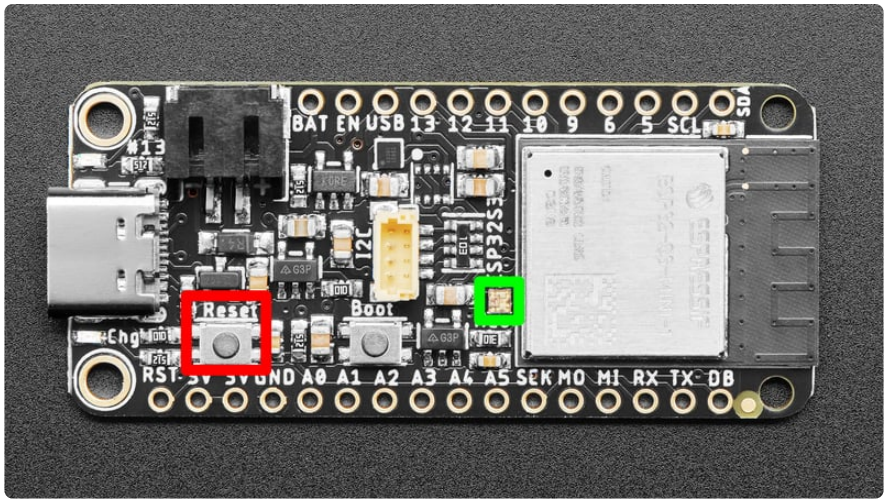
Download the latest version of CircuitPython for the Feather ESP32-S3 4MB Flash 2MB PSRAM via circuitpython.org

https://adafru.it/11BQ



Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.



Plug your board into your computer, using a known-good data-sync cable, directly, or via an adapter if needed.

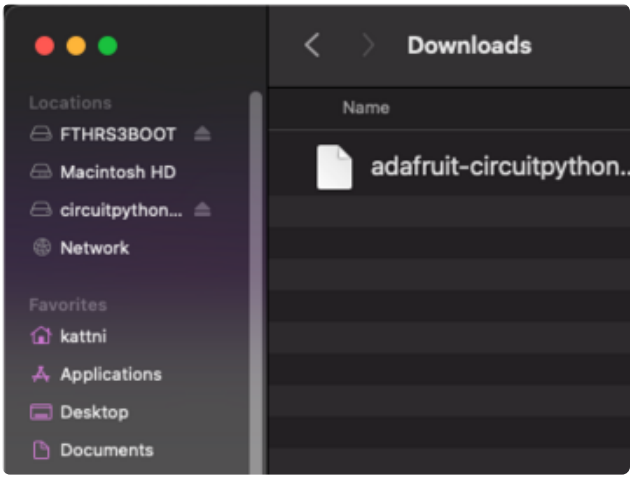
Double-click the **reset** button (highlighted in red above), and you will see the **RGB status LED(s)** turn green (highlighted in green above). If you see red, try another port, or if you're using an adapter or hub, try without the hub, or different adapter or hub.

For this board, tap reset and wait for the LED to turn purple, and as soon as it turns purple, tap reset again. The second tap needs to happen while the LED is still purple.

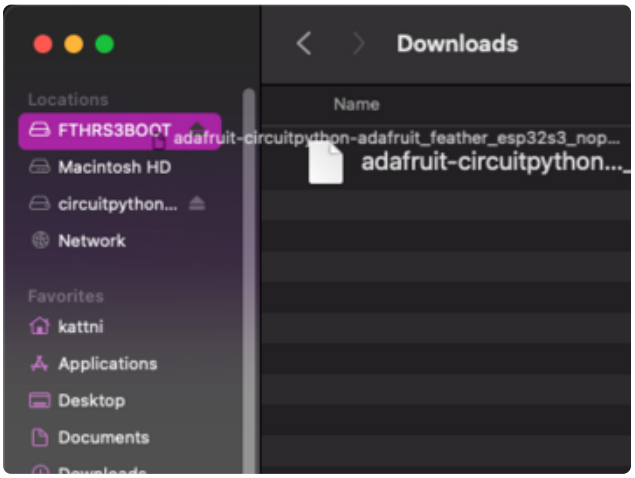
If you do not see the LED turning purple, you will need to reinstall the UF2 bootloader. See the **Factory Reset** page in this guide for details.

If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!

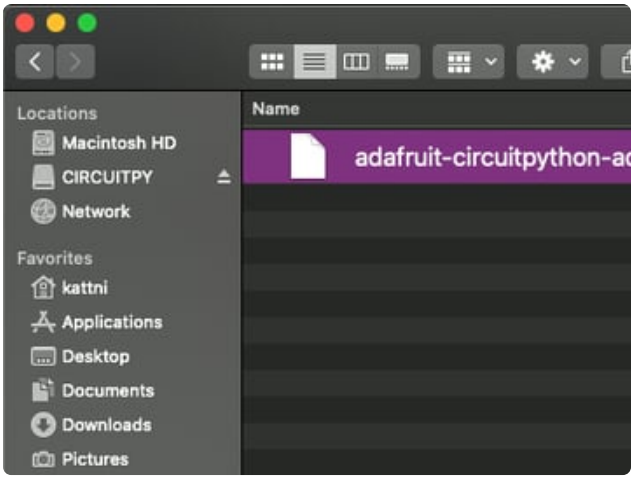
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



You will see a new disk drive appear called **FTHRS3BOOT**.



Drag the **adafruit_circuitpython_etc.uf2** file to **FTHRS3BOOT**.



The **BOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it!

Code the Grill Dashboard

Text Editor

Adafruit recommends using the **Mu** editor for editing your CircuitPython code. You can get more info in this guide (https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file. To get everything you need, click on the **Download Project Bundle** link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your board's **CIRCUITPY** drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.


```
# SPDX-FileCopyrightText: 2024 johnpark for Adafruit Industries
#
```

```
# SPDX-License-Identifier: MIT
'''
BLE BBQ Thermometer to WiFi to Adafruit IO Dashboard
Feather ESP32-S3 8MB No PSRAM
'''
import os
import time
import adafruit_connection_manager
import wifi
import adafruit_minimqtt.adafruit_minimqtt as MQTT
import adafruit_ble
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble_ibbq import IBBQService


aio_username = os.getenv("aio_username")
aio_key = os.getenv("aio_key")

print(f"Connecting to {os.getenv('CIRCUITPY_WIFI_SSID')}")
wifi.radio.connect(
    os.getenv("CIRCUITPY_WIFI_SSID"), os.getenv("CIRCUITPY_WIFI_PASSWORD")
)
print(f"Connected to {os.getenv('CIRCUITPY_WIFI_SSID')}")

### Feeds ###
feeds = [aio_username + f"/feeds/bbq{i}" for i in range(1, 7)]
battery_feed = aio_username + "/feeds/bbq_battery"

# Define callback methods which are called when events occur
# pylint: disable=unused-argument, redefined-outer-name
def connected(client, userdata, flags, rc):
    print("Connected to Adafruit IO")

def disconnected(client, userdata, rc):
    print("Disconnected from Adafruit IO")

# Create a socket pool
pool = adafruit_connection_manager.get_radio_socketpool(wifi.radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(wifi.radio)
connection_manager = adafruit_connection_manager.get_connection_manager(pool)

# Set up a MiniMQTT Client
mqtt_client = MQTT.MQTT(
    broker="io.adafruit.com",
    port=1883,
    username=aio_username,
    password=aio_key,
    socket_pool=pool,
    ssl_context=ssl_context,
)

# Setup the callback methods above
mqtt_client.on_connect = connected
mqtt_client.on_disconnect = disconnected

# Connect the client to the MQTT broker.
print("Connecting to Adafruit IO...")
mqtt_client.connect()

# PyLint can't find BLERadio for some reason so special case it here.
ble = adafruit_ble.BLERadio()  # pylint: disable=no-member

ibbq_connection = None
battery_percentage = 100

def c_to_f(temp_c):
    return (temp_c * 9/5) + 32

def volt_to_percent(voltage, max_voltage):
```

```
    return (voltage / max_voltage) * 100

def probe_check(temp):  # if value is wildly high no probe is connected
    return temp if temp <= 11000 else 0

battery_val = 3.3


while True:
    print("Scanning...")
    for adv in ble.start_scan(ProvideServicesAdvertisement, timeout=5):
        if IBBQService in adv.services:
            print("found an IBBq advertisement")
            ibbq_connection = ble.connect(adv)
            print("Connected")
            break

    # Stop scanning whether or not we are connected.
    ble.stop_scan()

    if ibbq_connection and ibbq_connection.connected:
        ibbq_service = ibbq_connection[IBBQService]
        ibbq_service.init()
        while ibbq_connection.connected:
            print(
                "Temperatures:",
                ibbq_service.temperatures,
                "; Battery:",
                ibbq_service.battery_level,
            )

            grill_vals = [probe_check(c_to_f(temp)) for temp in
ibbq_service.temperatures]
            battery_val, battery_max = ibbq_service.battery_level
            battery_percentage = (volt_to_percent(battery_val, 3.3))

            mqtt_client.loop(timeout=1)

            for feed, val in zip(feeds, grill_vals):
                print(f"Sending grill value: {val} to {feed}...")
                mqtt_client.publish(feed, val)

            mqtt_client.publish(battery_feed, battery_percentage)
            print("Sent")
            time.sleep(5)
```

How It Works

The code does two key things -- connect to a Bluetooth iBBQ device to receive data and connect to the Internet via WiFi to send data via MQTT to AIO feeds.

Libraries

Import necessary libraries for WiFi connection, BLE communication, and MQTT for data transfer.

```
import os
import time
import adafruit_connection_manager
import wifi
import adafruit_minimqtt.adafruit_minimqtt as MQTT
import adafruit_ble
```

```
from adafruit_ble.advertising.standard import ProvideServicesAdvertisement
from adafruit_ble_ibbq import IBBQService
```

Environment Variables for Adafruit IO

Retrieve Adafruit IO username and key from environment variables from the **settings.toml** file.

```
aio_username = os.getenv("aio_username")
aio_key = os.getenv("aio_key")
```

Connect to WiFi

Connect to WiFi using the SSID and password stored in **settings.toml** file.

```
print(f"Connecting to {os.getenv('CIRCUITPY_WIFI_SSID')}")
wifi.radio.connect(os.getenv("CIRCUITPY_WIFI_SSID"),
os.getenv("CIRCUITPY_WIFI_PASSWORD"))
print(f"Connected to {os.getenv('CIRCUITPY_WIFI_SSID')}")
```

Setup MQTT Feeds

Define functions to handle connection and disconnection events with the MQTT broker.

```
feeds = [aio_username + f"/feeds/bbq{i}" for i in range(1, 7)]
battery_feed = aio_username + "/feeds/bbq_battery"
```

MQTT Callback Functions

Define functions to handle connection and disconnection events with the MQTT broker.

```
def connected(client, userdata, flags, rc):
    print("Connected to Adafruit IO")

def disconnected(client, userdata, rc):
    print("Disconnected from Adafruit IO")
```

Create Socket Pool and SSL Context

Set up the socket pool and SSL context for secure MQTT communication.

```
pool = adafruit_connection_manager.get_radio_socketpool(wifi.radio)
ssl_context = adafruit_connection_manager.get_radio_ssl_context(wifi.radio)
connection_manager = adafruit_connection_manager.get_connection_manager(pool)
```

Setup MQTT Client

Configure the MQTT client and connect it to the Adafruit IO broker.

```
mqtt_client = MQTT.MQTT(
    broker="io.adafruit.com",
    port=1883,
    username=aio_username,
    password=aio_key,
    socket_pool=pool,
    ssl_context=ssl_context,
)
mqtt_client.on_connect = connected
mqtt_client.on_disconnect = disconnected
print("Connecting to Adafruit IO...")
mqtt_client.connect()
```

Initialize BLE Radio

Initialize the BLE radio for scanning and connecting to the BBQ thermometer.

```
ble = adafruit_ble.BLERadio()
```

Helper Functions

There are three functions created here. One to convert Celsius to Fahrenheit, another to convert voltage to percentage, and one to check if a temperature probe is connected.

```
def c_to_f(temp_c):
    return (temp_c * 9/5) + 32

def volt_to_percent(voltage, max_voltage):
    return (voltage / max_voltage) * 100

def probe_check(temp):
    return temp if temp <= 11000 else 0
```

Main Loop

The main loop first scans for the BBQ thermometer, and then connects to it.

Next, it reads temperature and battery data, converts them to Farenheit and Voltage, respectively.

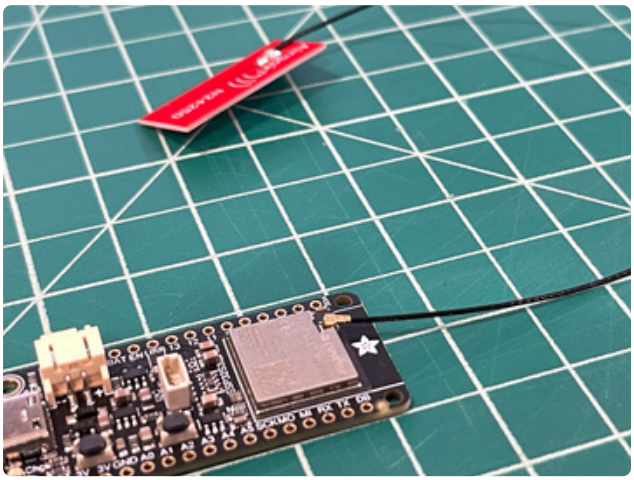
Then, it sends this data to Adafruit IO using MQTT, and repeats the process in a loop every five seconds.

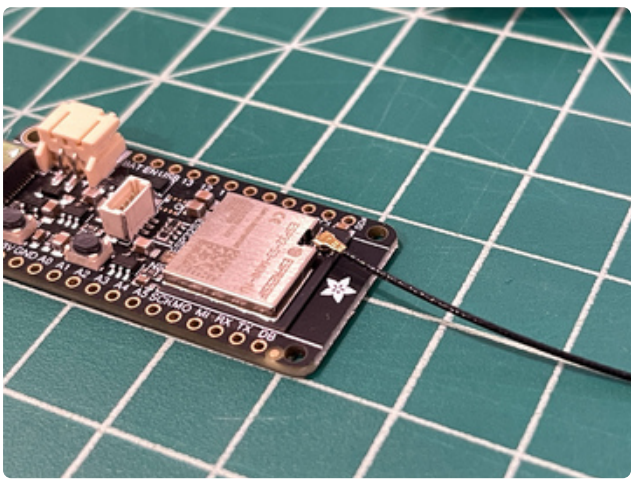
```
while True:
    print("Scanning...")
    for adv in ble.start_scan(ProvideServicesAdvertisement, timeout=5):
        if IBBQService in adv.services:
            print("found an IBBq advertisement")
            ibbq_connection = ble.connect(adv)
            print("Connected")
            break
    ble.stop_scan()
    if ibbq_connection and ibbq_connection.connected:
        ibbq_service = ibbq_connection[IBBQService]
        ibbq_service.init()
```

```
        while ibbq_connection.connected:
            print("Temperatures:", ibbq_service.temperatures, "; Battery:",
ibbq_service.battery_level)
            grill_vals = [probe_check(c_to_f(temp)) for temp in
ibbq_service.temperatures]
            battery_val, battery_max = ibbq_service.battery_level
            battery_percentage = volt_to_percent(battery_val, 3.3)
            mqtt_client.loop(timeout=1)
            for feed, val in zip(feeds, grill_vals):
                print(f"Sending grill value: {val} to {feed}...")
                mqtt_client.publish(feed, val)
            mqtt_client.publish(battery_feed, battery_percentage)
            print("Sent")
            time.sleep(5)
```

On the next page follow the MQTT in CircuitPython guide to get your Adafruit IO account and feed set up.

Build the Grill Dashboard Feather Case





Feather Prep

If you chose to use the external antenna version of the Feather ESP32-S3 carefully plug in the antenna connector now.

You can make a neat enclosure for your Grill Dashboard Feather, including optional battery and on/off switch.

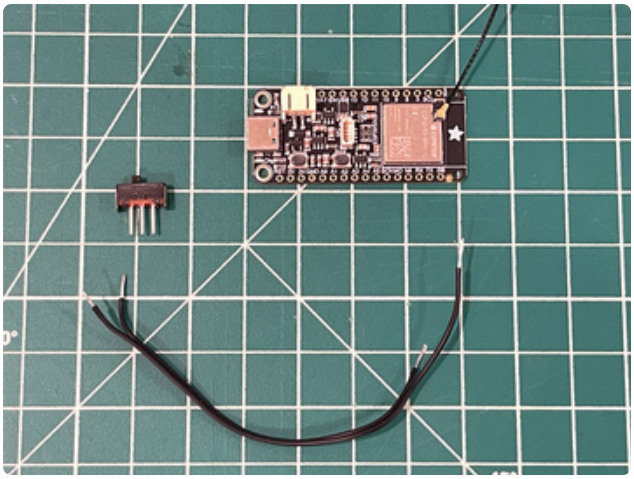
This also gives you a neat mounting surface for the external antenna if you chose to use that version of the Feather board.

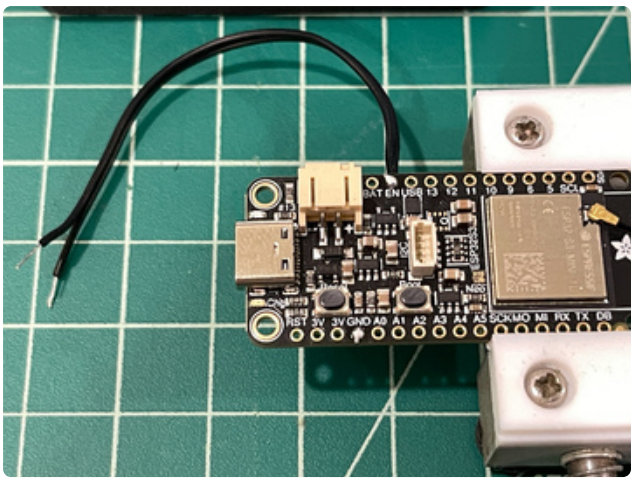
First, head to the **3D Printed Case for Adafruit Feather** Learn Guide (https://adafru.it/XeH).

Download the .zip file of the case models here (https://adafru.it/vek).

Print these three models:

feather-case.stl
feather-top.stl
feather-bat-tab-switch.stl



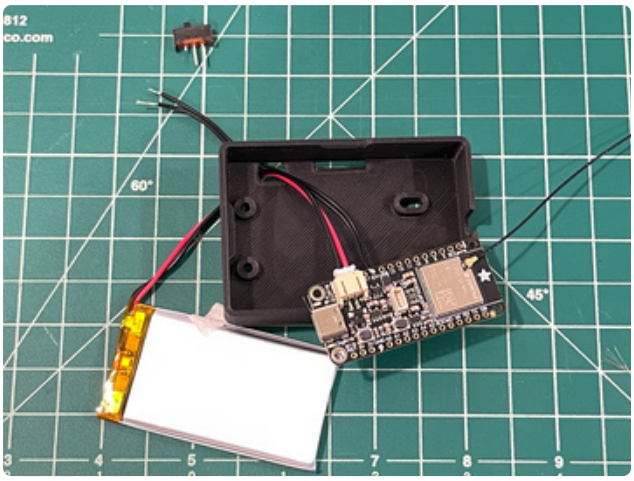


Enable Switch

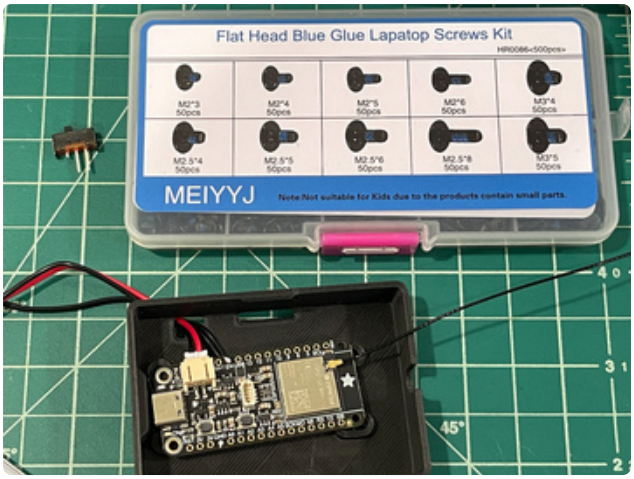
Prep a short length of two conductor wire -- about 4" will do. You can shorten one wire about 1/2" for neater routing, although this step isn't strictly necessary.

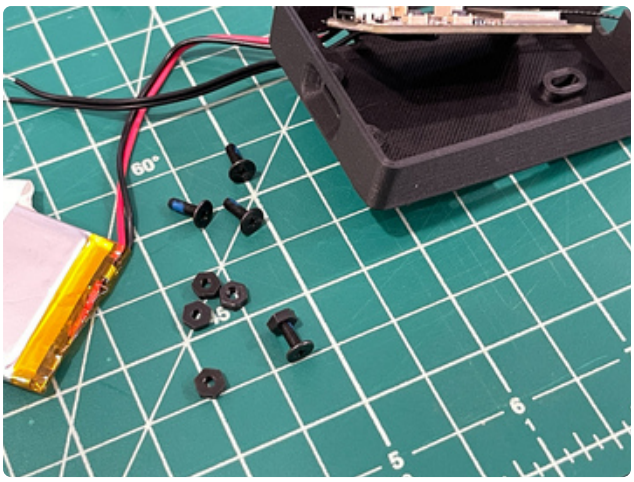
Strip the insulation from the ends.

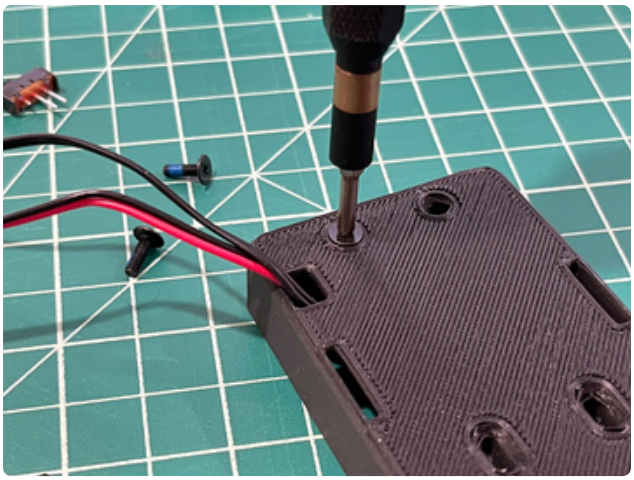
Solder the wires to the Feather **GND** and **En**(able) pins.



Feed the switch wires through the case top as shown. You can also plug in the battery to the Feather at this time.





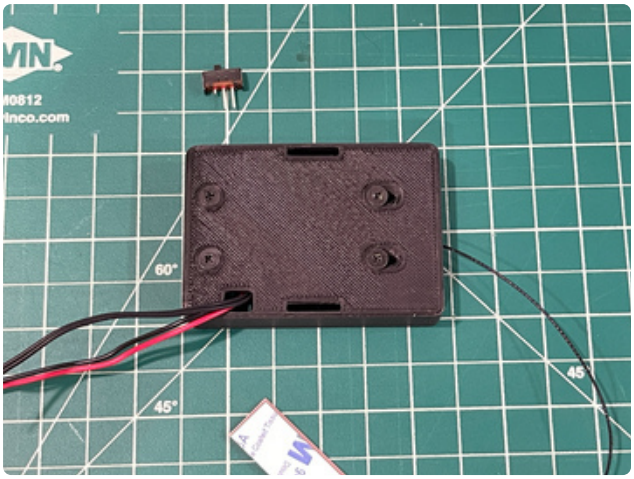


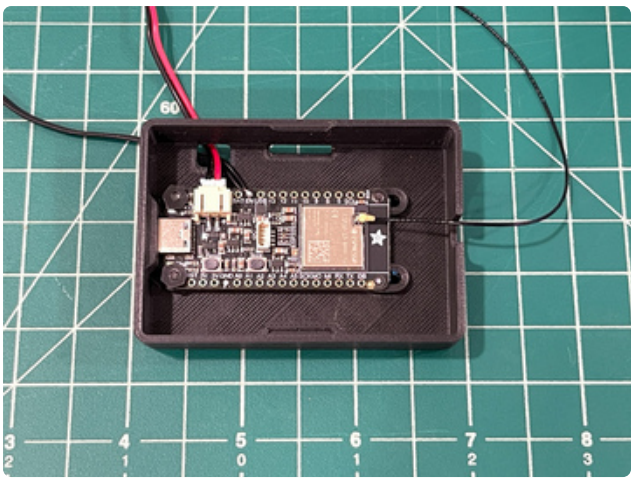
Feather Fasteners

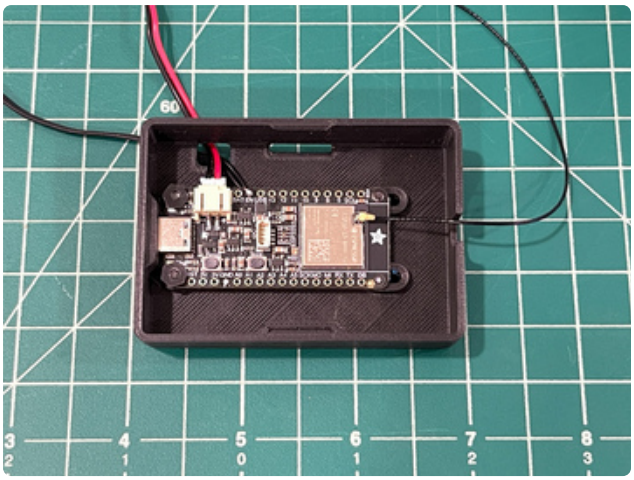
Use M2.5 screws and nuts to secure the Feather to the case as shown.

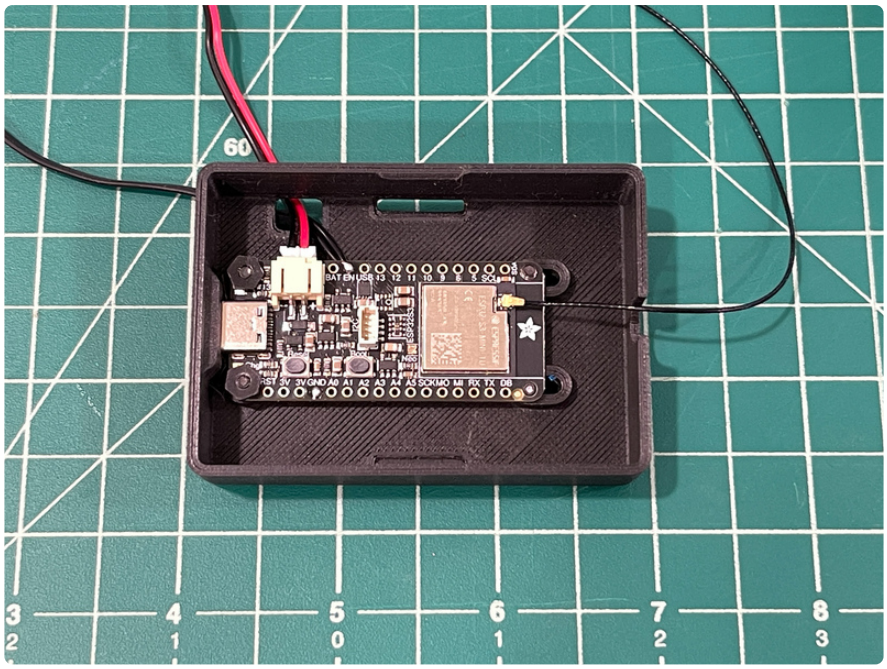
The non-plated holes at the right end of the Feather are technically M2, but you can screw metal screws directly into them without needing nuts if you don't want to scrounge up some M2 screws and nuts.

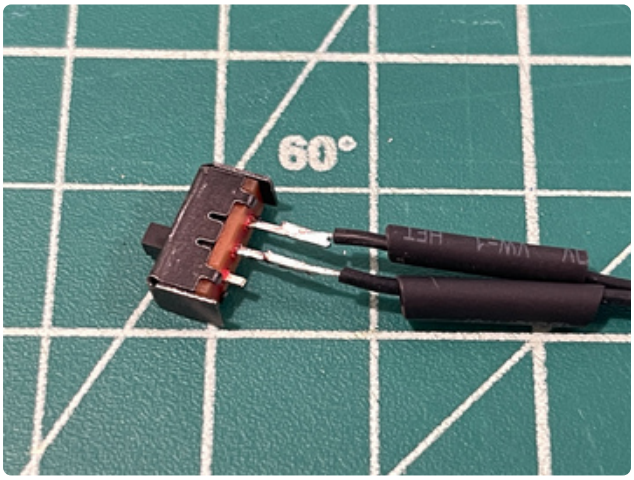
I mean, I literally had some right there in that assortment box and still didn't feel like bothering -- just don't snort up any fiberglass dust created by tapping the PCB with the oversized screws please.


60°
45°
45°









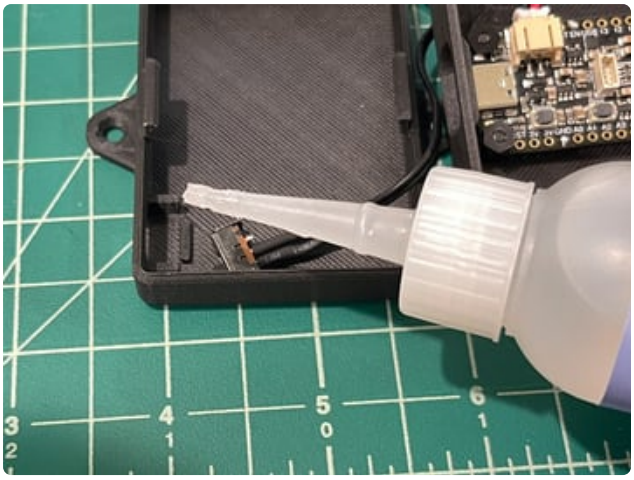
Switch Soldering

Tin the common and one side leg on the switch, do the same for the wires.

Add a bit of heat shrink tubing to the wires.

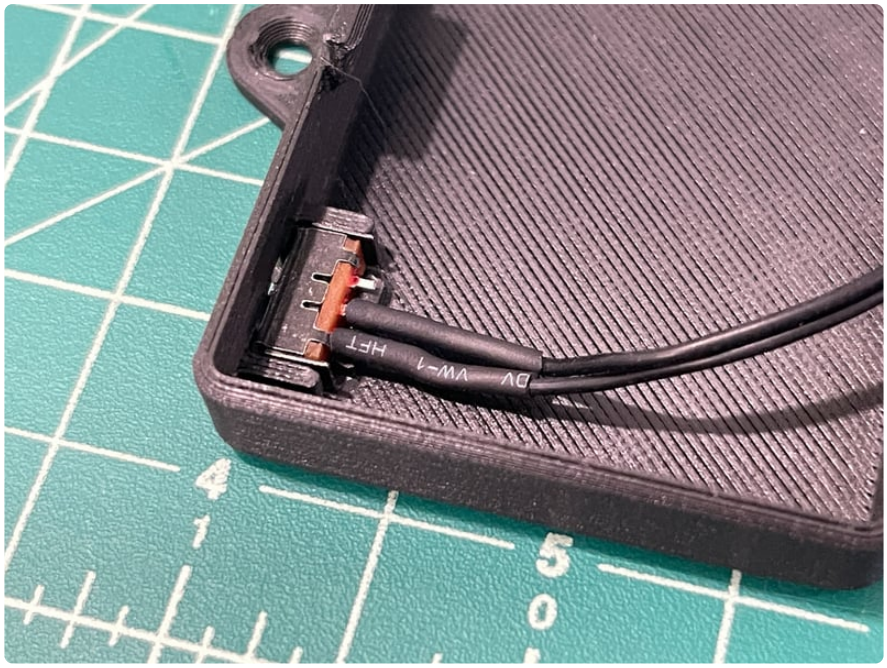
Solder the wires to the switch legs as shown -- polarity doesn't matter, since we're using the switch to ground the Enable pin.

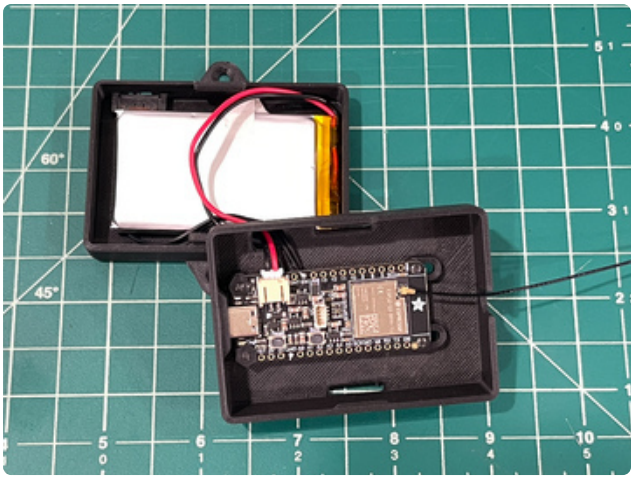
Shrink the tubing with a heat gun, lighter, soldering iron barrel, scathing remark, or minor incantation.



Switch Attachment

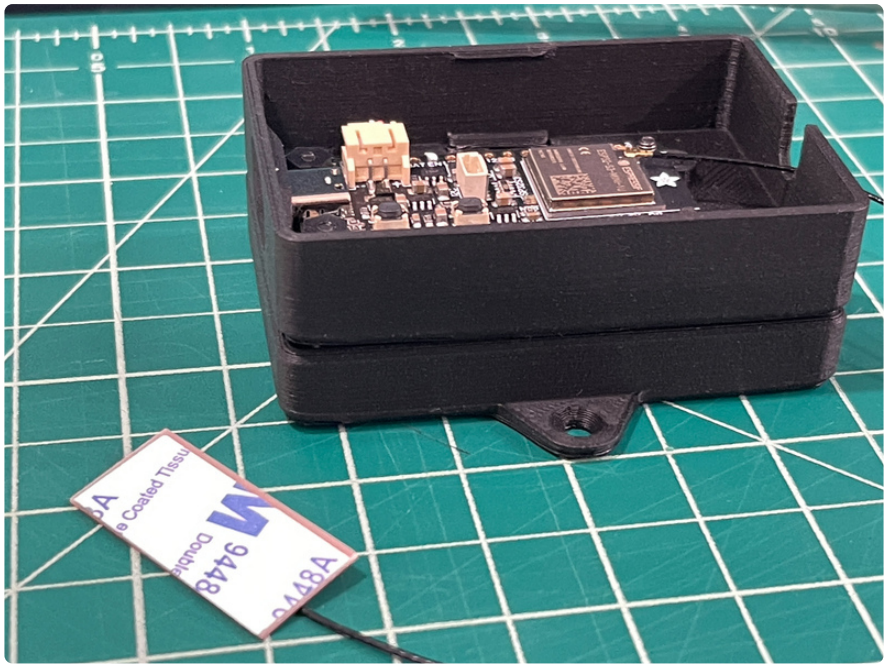
Use a small dab of CA glue to adhere the switch to the case.

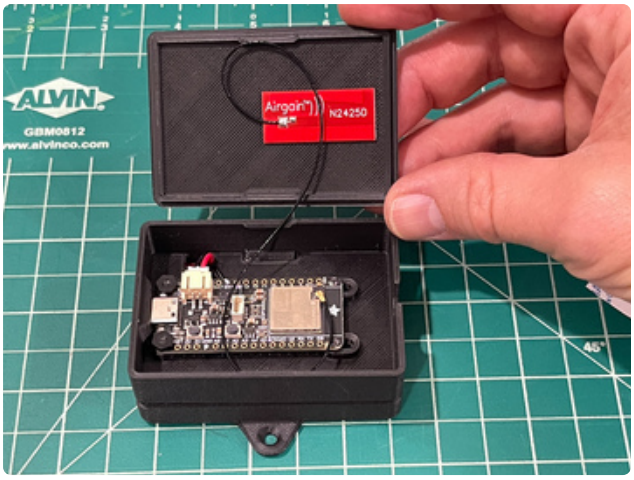




Battery Placement

Insert the battery in the bottom of the case, dressing the wires away from the edges to the middle case section can snap fit in place.

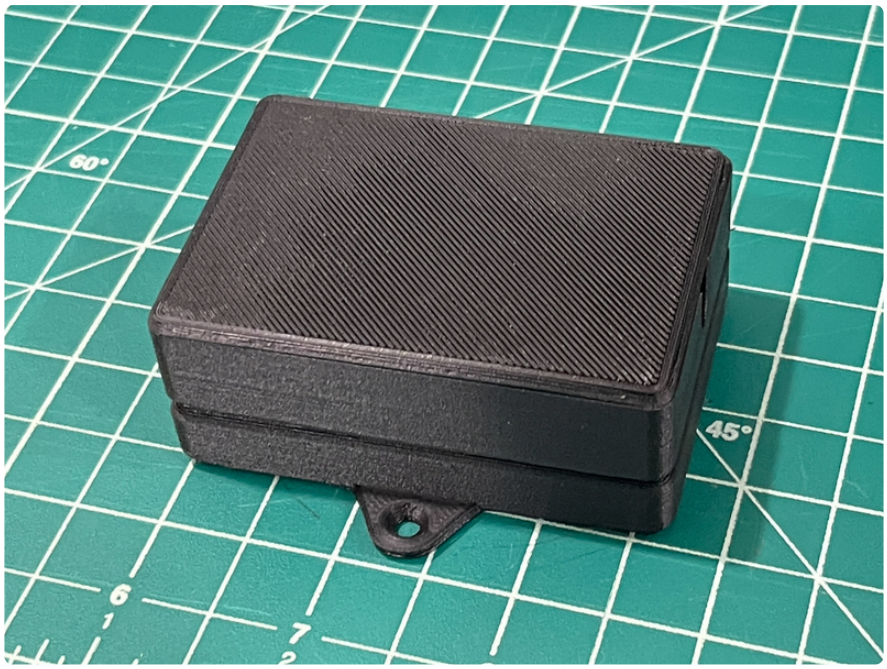




Antenna Mount

If you're using the external antenna version of the Feather ESP32-S3, remove the adhesive protective backing and stick it inside of the case lid as shown.

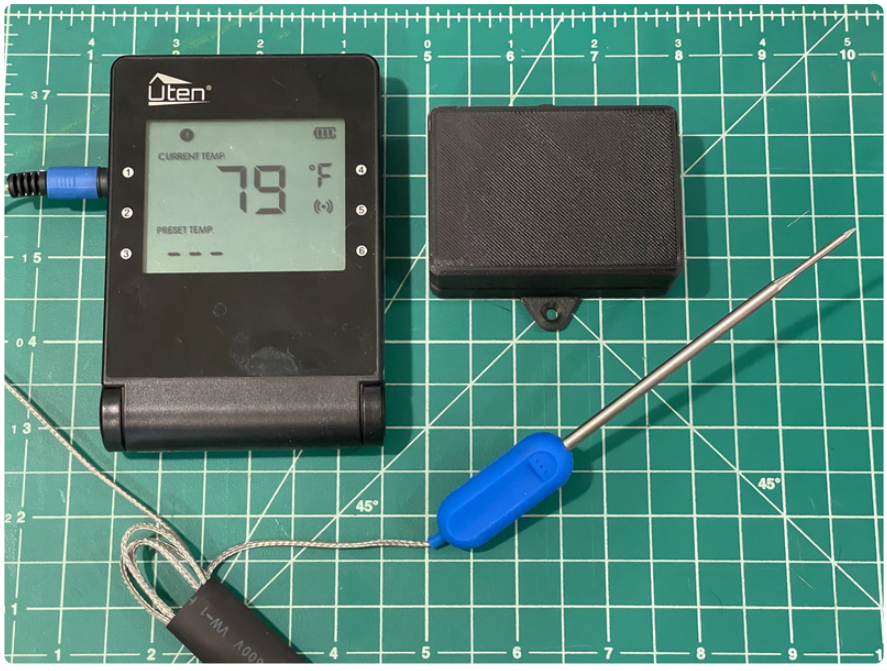
You can now close up the case.



Feather Usage

Plug in USB C cable to code the Feather and charge the LiPoly battery.

Flip the switch to turn it on or off.



CircuitPython Setup

MQTT devices, like your CircuitPython board, connect to a broker with a client library.

We've written an awesome CircuitPython MQTT client library called Adafruit MiniMQTT **(https://adafru.it/Fm2)**.

This library is based off previous work by pfalcon on uMQTT (https://adafru.it/Fm3) (and the umqtt port to ESP32SPI by beachbc (https://adafru.it/Fm4)). MiniMQTT's primary difference from MicroPython's uMQTT library is its use of calling conventions and method names similar to The Eclipse Foundation's Paho.Mqtt.Python (https://adafru.it/Fm5).

Install CircuitPython

Some CircuitPython compatible boards come with CircuitPython installed. Others are CircuitPython-ready, but need to have it installed. As well, you may want to update the version of CircuitPython already installed on your board. The steps are the same for installing and updating.

- To **install (or update) your CircuitPython board**, follow this page and come back here when you've successfully installed (or updated) CircuitPython. (https://adafru.it/Amd)

CircuitPython Library Installation

To interface your AirLift breakout/board with and the internet - you'll need to install a few CircuitPython libraries on your board.

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) matching your version of CircuitPython.

CircuitPython hardware shows up on your computer operating system as a flash drive when connected via usb. The flash drive is called **CIRCUITPY** and contains a number of files. You will need to add additional files to enable the features of this project.

First, create a folder on the drive named lib if it is not already there.

Ensure your board's **lib** folder has the following files and folders copied over. The version of the files must be the same major version as your version of CircuitPython (i.e. 4.x for 4.x, 5.x for 5.x, etc.)

- **adafruit_minimqtt**

Connecting to the Adafruit IO MQTT Broker

If you do not want to host your own MQTT broker, using Adafruit IO (https://adafru.it/eZ8)'s MQTT broker is a great way to get started connecting your CircuitPython project to the internet. Best of all - it's free to use!

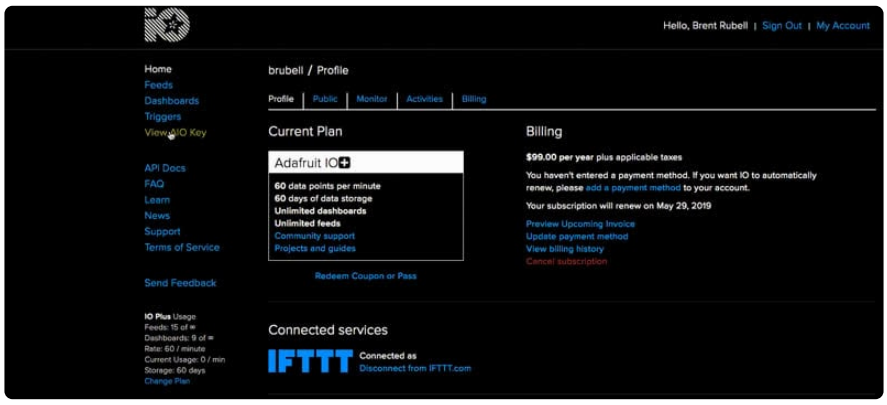
You're going to build an Adafruit IO Dashboard which can visualize incoming data from your CircuitPython board, and send data to it.

Obtain Adafruit IO Username and Key

If you have not already, sign up for an Adafruit IO account by clicking this link (https://adafru.it/Fm6).

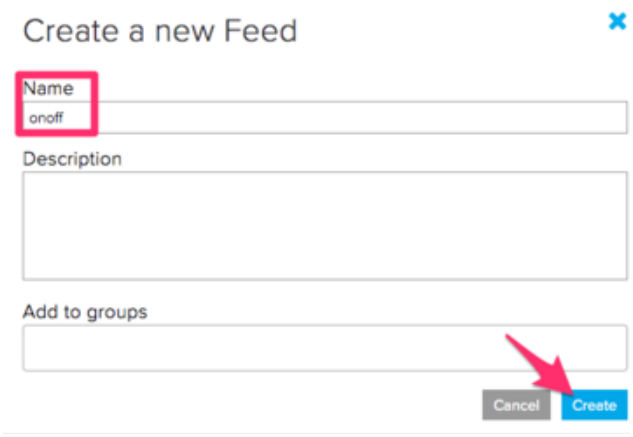
Next, you're going to need your Adafruit IO username and secret API key.

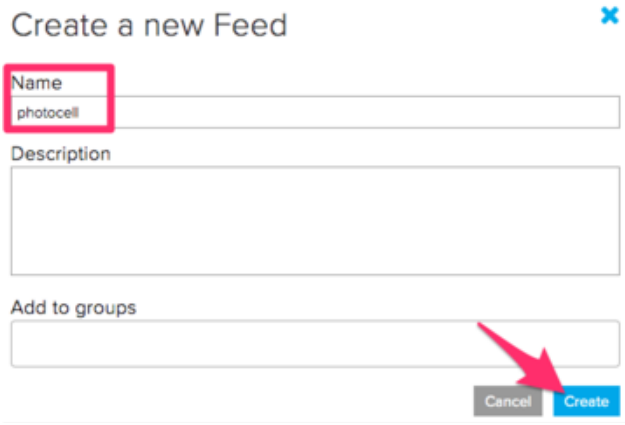
Navigate to your profile **(https://adafru.it/fsU) and click the View AIO Key button** to retrieve them. Write them down in a safe place, you'll need them later.



Create Adafruit IO Feeds

Adafruit IO uses a special type of MQTT Topic named a Feed to store data along with metadata (information about the data). You'll be publishing data to one feed, and subscribing to another.





Create two new Adafruit IO Feeds named onoff and photocell.

photocell - This feed will store light data published from your device to Adafruit IO
onoff - This feed will act as an on/off switch, publishing data to your device from Adafruit IO

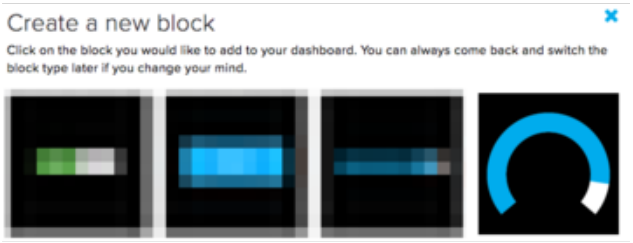
If you have not created an Adafruit IO Feed before, follow this page and come back once you've the created two feeds above (https://adafru.it/f5k).

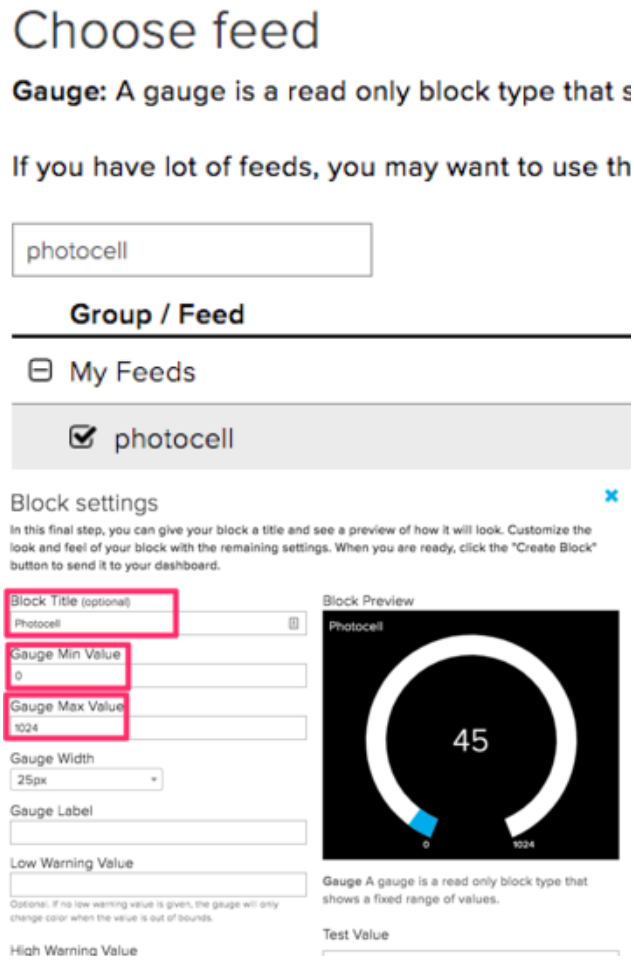
Create an Adafruit IO Dashboard

Adafruit IO Dashboards (https://adafru.it/f5m) are a way to interact with feeds. You can link blocks on dashboards to your feeds. The blocks can either display information about the feed (such as the current temperature) or allow you to interact with a feed by setting it to different values.

Start by creating a new dashboard. Name it whatever you'd like!

- **If you do not know how to create a dashboard**, head over to this page and come back here when you've successfully created a dashboard. (https://adafru.it/Fm7)



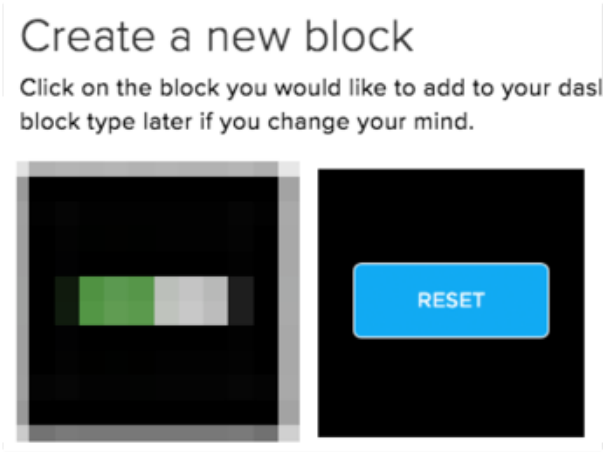


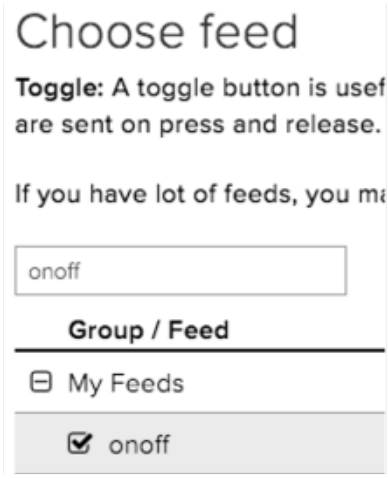
Create a Gauge Block

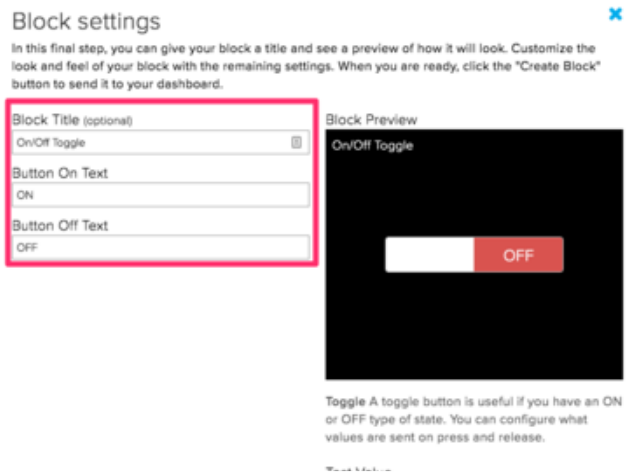
After creating a dashboard, **create a Gauge Block** to display the value of the Photocell feed.

Choose the photocell feed
Change the block title to Photocell
Set the Gauge Minimum Value to 0
Set the Gauge Maximum Value to 1024

If you do not know how to add blocks to a dashboard, head to over this page and come back when you've added a gauge block to your dashboard. (https://adafru.it/DZe)





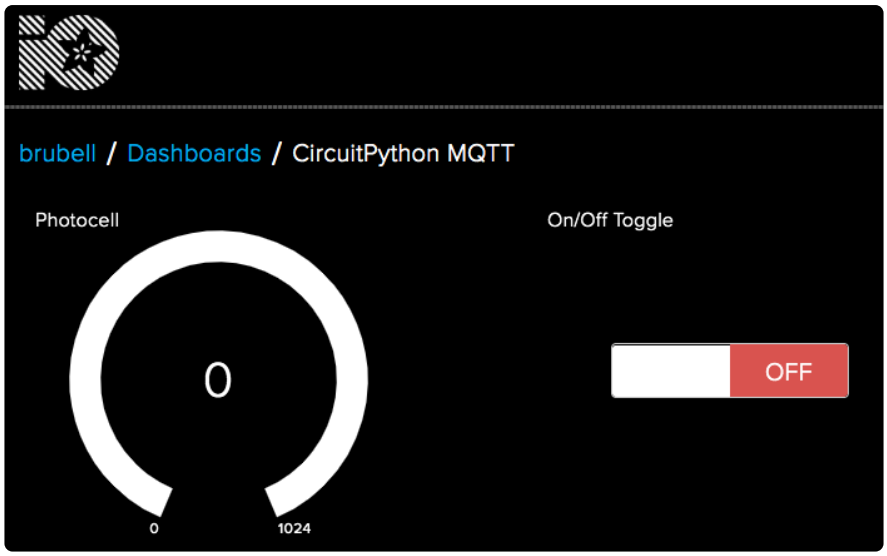


Create a Toggle Switch Block

To send values to the onoff feed you created - create a toggle switch block.

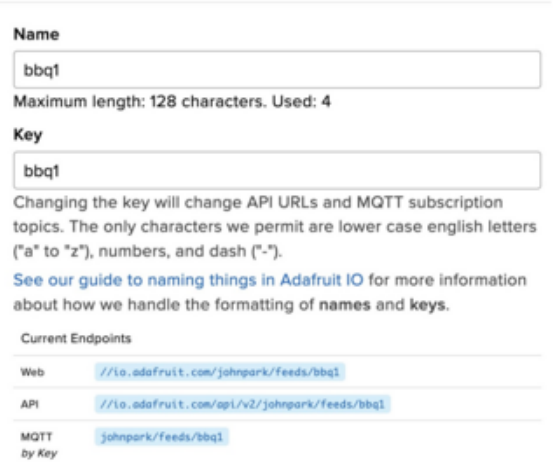
Choose the onoff **feed**
Set the block title to On/Off
Set the Button On Text to ON
Set the Button Off Text to OFF

Your dashboard should look like the following:



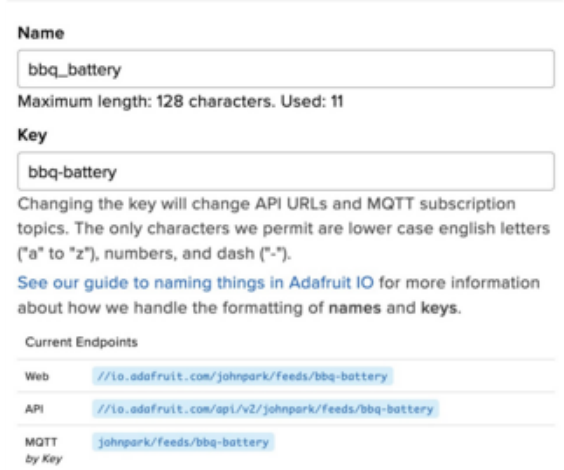
Make the Grill Dashboard

Following the detailed instructions here (https://adafru.it/ioA), create one feed per temperature probe with the following settings (increment the name numbering to avoid naming conflicts).



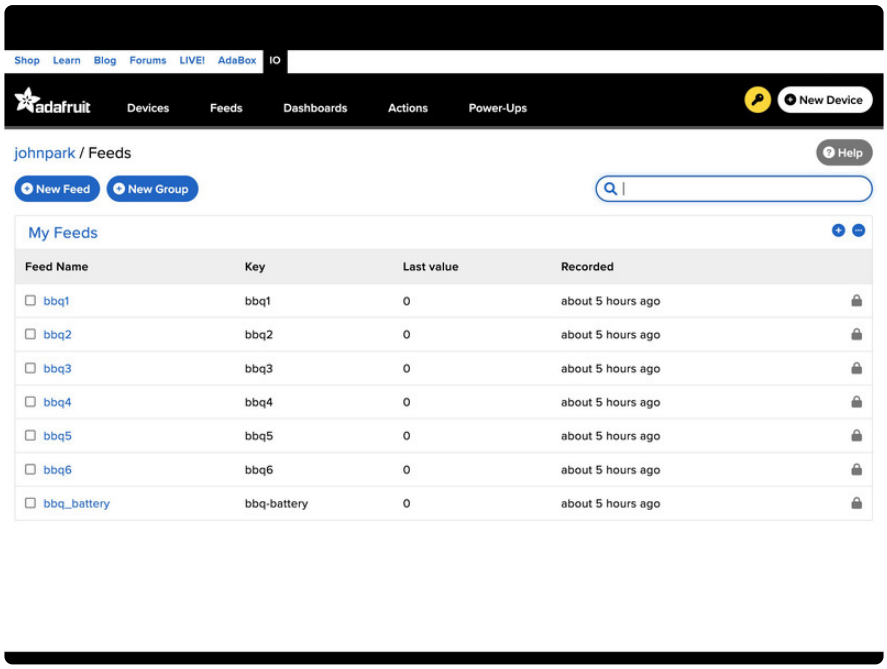
Create Temperature Feeds

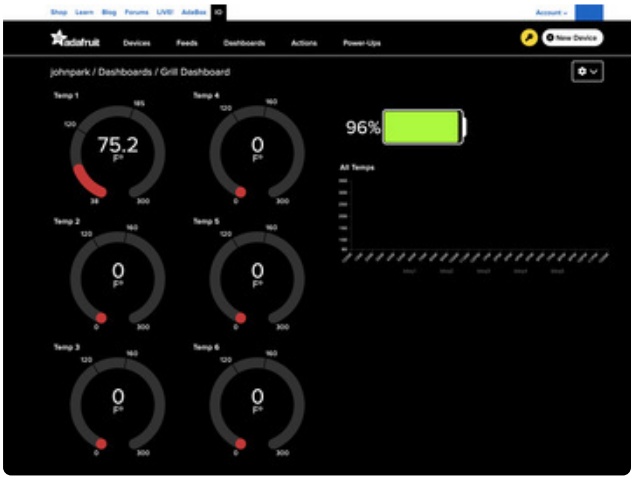
Following the detailed instructions here, create one feed per temperature probe with the following settings (increment the name numbering to avoid naming conflicts).



Create Battery Feed

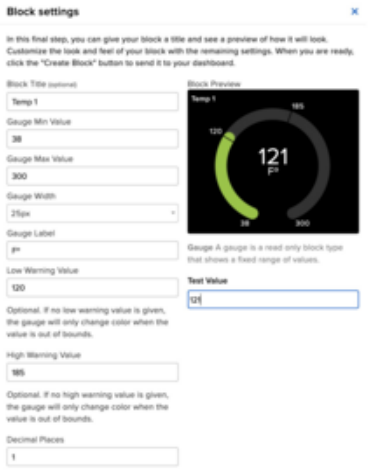
The iBBQ service can also report the battery level for the transmitter unit. Create a feed for this with the settings shown.





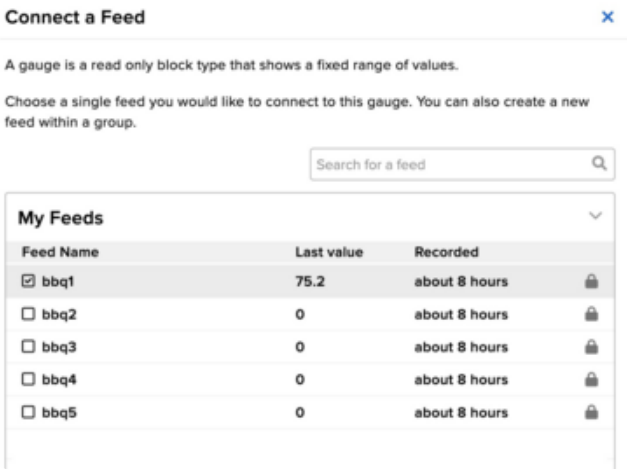
Dashboard

Following the detailed instructions in this guide, create a new dashboard named **Grill Dashboard.**



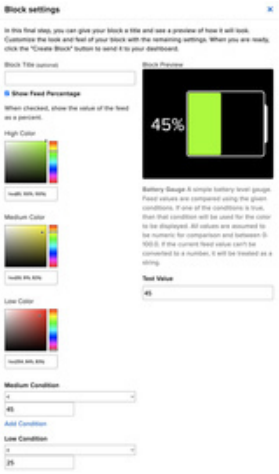
Create Gauge Blocks

Create one gauge block per temperature feed as shown.



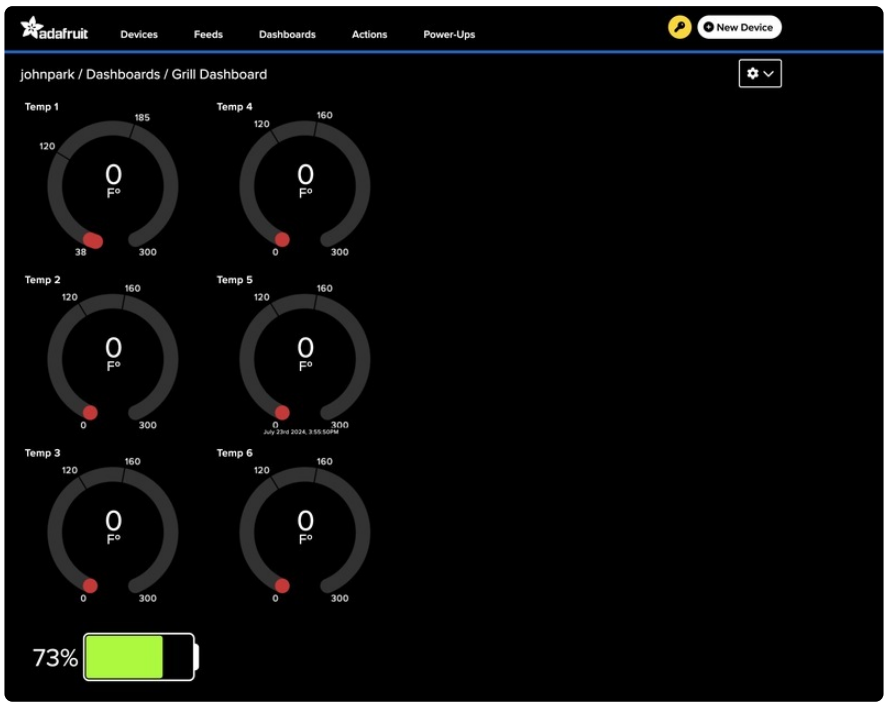
Connect Feeds

Connect the gauge blocks to their relative feeds.



Battery Gauge

You can also create a battery gauge as shown.



Use the Grill Thermometer Dashboard

To use the Grill Thermometer Dashboard:

- Turn on the Feather enable switch
- Turn on the iBBQ capable BLE transmitter
- Insert probes in your food
- Fire up the grill
- Keep an eye on the grill temperature from the comfort of your air conditioned home on your computer -- or maybe on your phone while you soak in the pool! Any internet capable device can be used to monitor your Adafruit IO web-based dashboard.