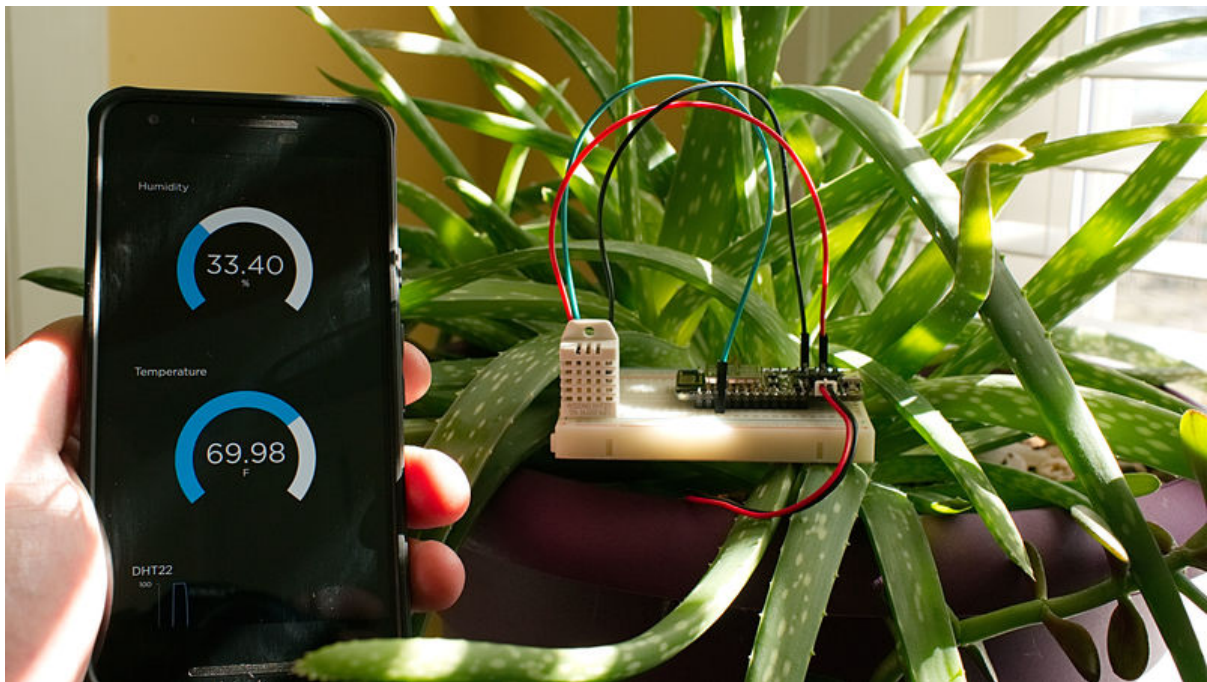




Adafruit IO Basics: Temperature & Humidity

Created by Todd Treece



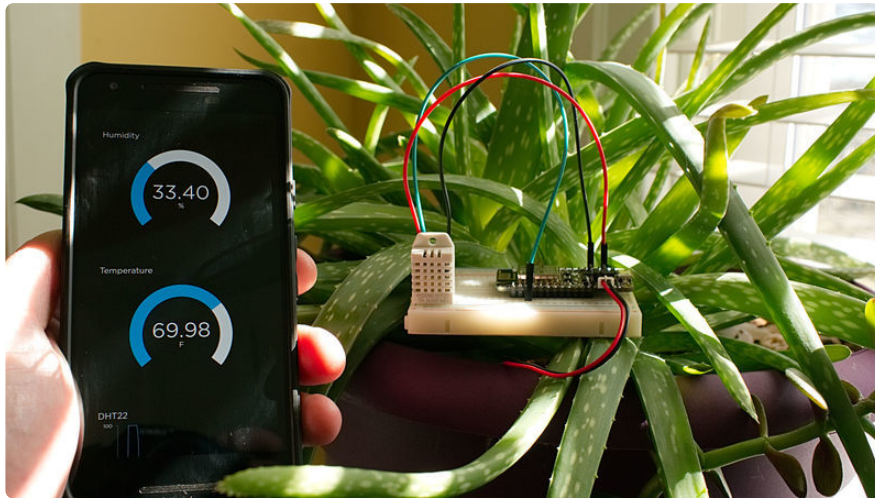
<https://learn.adafruit.com/adafruit-io-basics-temperature-and-humidity>

Last updated on 2024-06-03 02:04:55 PM EDT

Table of Contents

Overview	3
Adafruit IO Setup	3
<ul style="list-style-type: none">• Creating the Feeds• Adding the Line Chart Block	
Arduino Wiring	6
Arduino Setup	7
Arduino Network Config	8
<ul style="list-style-type: none">• WiFi Config• FONA Config• Ethernet Config	
Arduino Code	10
Add an OLED	13
Python Wiring	16
<ul style="list-style-type: none">• Parts• Wiring	
Python Setup	17
<ul style="list-style-type: none">• Update your Pi and Python• Make sure you're using Python 3!• Installing Adafruit Python DHT Library• Installing Adafruit IO Python Library• Downloading Example Code	
Python Code	20
<ul style="list-style-type: none">• Code	
Adafruit IO FAQ	23
<ul style="list-style-type: none">• Encountering an issue with your Adafruit IO Arduino Project?	

Overview



This guide is part of a series of guides that cover the basics of using Adafruit IO. It will show you how to send temperature and humidity values wirelessly to Adafruit IO from a DHT22 sensor.

If you haven't worked your way through the Adafruit IO feed and dashboard basics guides, you should do that before continuing with this guide so you have a basic understanding of Adafruit IO.

- [Adafruit IO Basics: Feeds](#)
- [Adafruit IO Basics: Dashboards](#)

You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

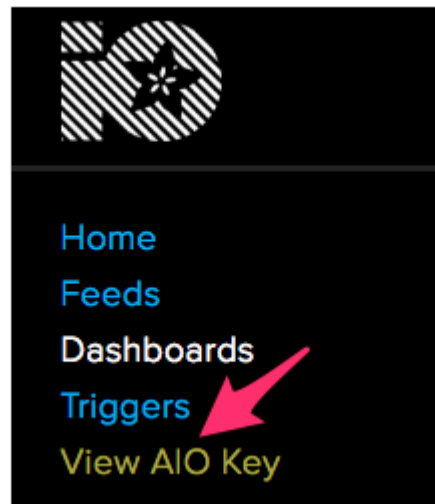
- [Adafruit Feather HUZZAH ESP8266 Setup Guide](#)

If you have went through all of the prerequisites for your selected hardware, you are now ready to move on to the Adafruit IO setup steps that are common between all of the hardware choices for this project. Let's get started!

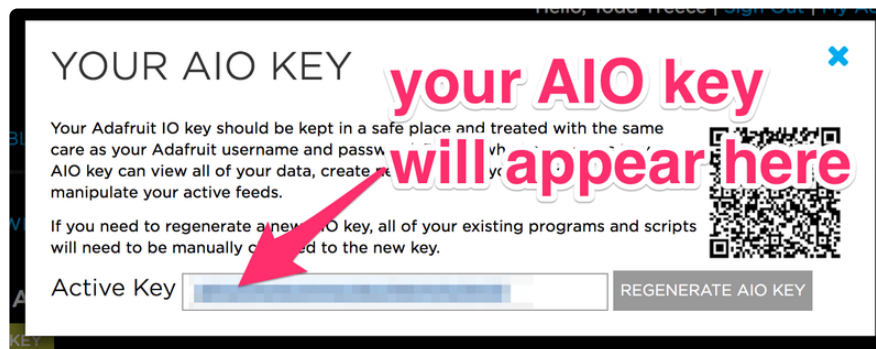
Adafruit IO Setup

The first thing you will need to do is to login to [Adafruit IO](#) and visit the **Settings** page.

Click the **VIEW AIO KEY** button to retrieve your key.



A window will pop up with your Adafruit IO. Keep a copy of this in a safe place. We'll need it later.



Creating the Feeds

First, you will need to create a feed called **Temperature**.

A screenshot of a form titled 'Create a new Feed'. It has a 'Name' field with 'Temperature' entered and a 'Description' field which is empty. At the bottom right are 'Cancel' and 'Create' buttons.

You will also need to create a feed called **Humidity**.

Create a new Feed

Name
Humidity

Description

Cancel Create

If you need help getting started with creating feeds on Adafruit IO, check out the [Adafruit IO Feed Basics guide \(https://adafru.it/ioA\)](https://adafru.it/ioA).

Adding the Line Chart Block

Add a new Line Chart block to a new or existing dashboard. Make sure you have selected both the **Temperature** and **Humidity** feeds as the data sources for the block.

Choose up to 5 feeds

The line chart is used to chart one or more feeds. If you have lot of feeds, you may want to use the search field. You can also create a feed quickly below.

Search: Enter new feed name: Create

Group / Feed	Last value	Recorded
default		
<input checked="" type="checkbox"/> Humidity	3 minutes ago	1 of 5
<input checked="" type="checkbox"/> Temperature	3 minutes ago	2 of 5

< Previous step Next step >

When you reach the block settings, set the Hours of History setting to 24 hours, and name the block whatever you would like.

When you are finished editing the form, click Create Block to add the new block to the dashboard.

Block settings

In this final step, you can give your block a title and see a preview of how it will look. Customize the look and feel of your block with the remaining settings. When you are ready, click the "Create Block" button to send it to your dashboard.

Block Title

Hours of History (0 for realtime)

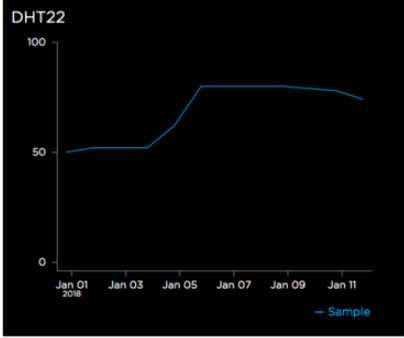
X-Axis Label

Y-Axis Label

Y-Axis Minimum

Y-Axis Maximum

Block Preview



[< Previous step](#)[Create block](#)

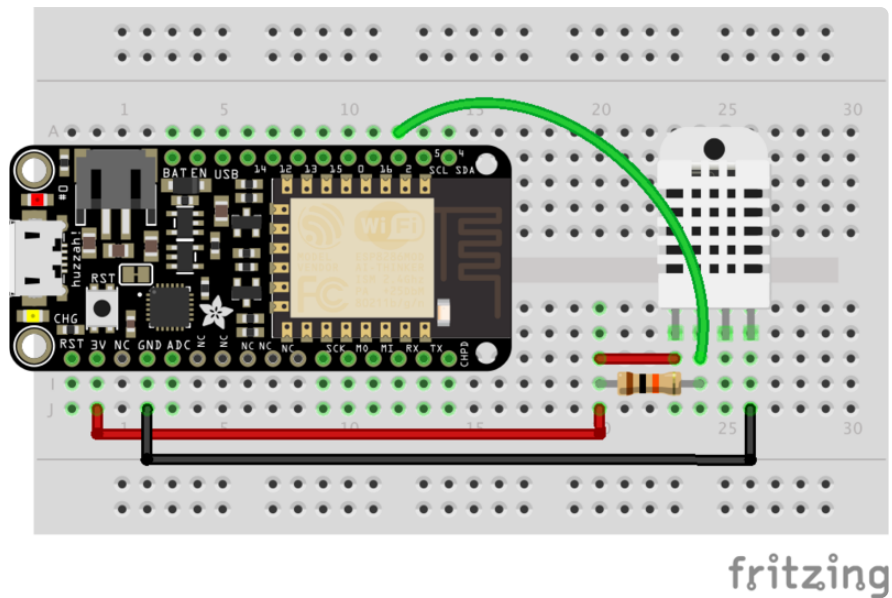
If you need help getting started with Dashboards on Adafruit IO, check out the [Adafruit IO Dashboard Basics guide \(https://adafru.it/f5m\)](https://adafru.it/f5m).

Next, we will look at wiring the circuit.

Arduino Wiring

You will need the following parts for this tutorial:

- 1x Adafruit IO compatible Feather
- 1x DH22 temperature & humidity sensor
- 1x 10k ohm resistor
- 4x jumper wires



We will need to connect the following pins from the Feather to the resistor and DHT22:

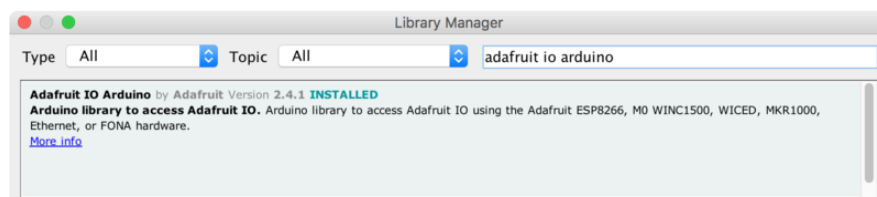
- * Feather 3V to Pin 1 of the DHT22
- * Feather 3V to one leg of a 10k ohm resistor, and the other leg of the resistor to Pin 2 of the DHT22
- * Feather Pin 2 to Pin 2 of the DHT22
- * Feather GND to Pin 4 of the DHT22

Arduino Setup

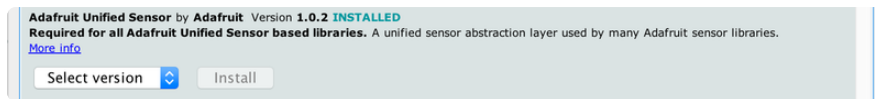
You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

- [Adafruit Feather HUZZAH ESP8266 Setup Guide](#)

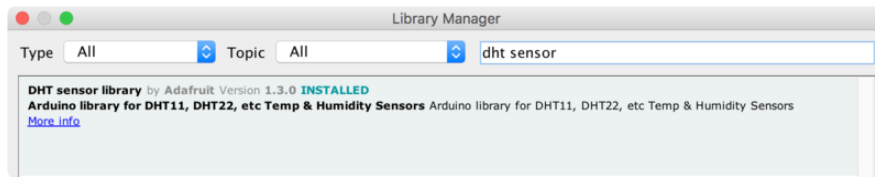
You will need to make sure you have at least **version 2.4.1** of the Adafruit IO Arduino library installed before continuing.



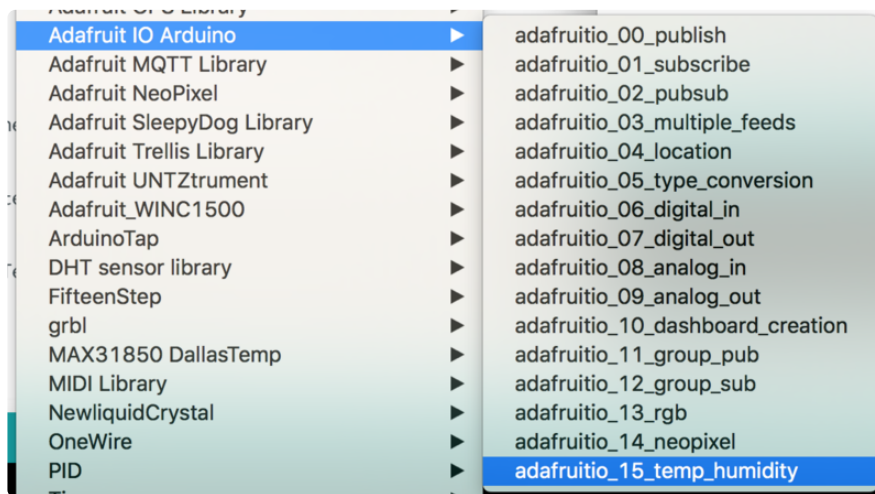
You will also need to install the **Adafruit Unified Sensor** library.



As well as the **DHT Sensor Library**.



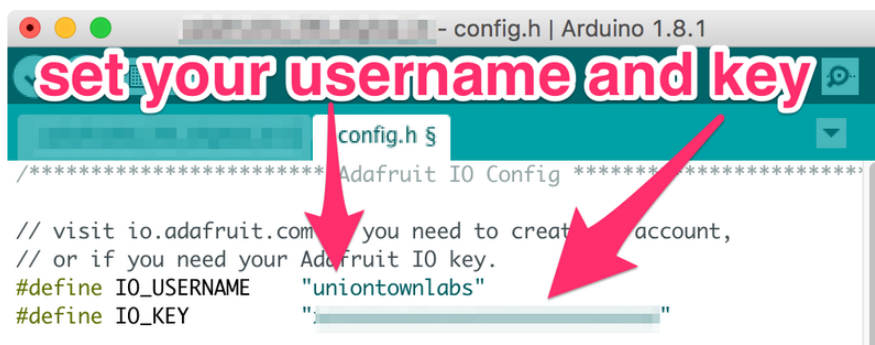
For this example, you will need to open the **adafruitio_15_temp_humidity** example in the **Adafruit IO Arduino** library.



Next, we will look at the network configuration options in the sketch.

Arduino Network Config

To configure the network settings, click on the **config.h** tab in the sketch. You will need to set your Adafruit IO username in the **IO_USERNAME** define, and your Adafruit IO key in the **IO_KEY** define.



WiFi Config

WiFi is enabled by default in **config.h** so if you are using one of the supported WiFi boards, you will only need to modify the **WIFI_SSID** and **WIFI_PASS** options in the **config.h** tab.

```

/***** WIFI *****/

// the AdafruitIO_WiFi client will work with the following boards:
// - HUZZAH ESP8266 Breakout -> https://www.adafruit.com/products/2471
// - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
// - Feather M0 WiFi -> https://www.adafruit.com/products/3010
// - Feather WICED -> https://www.adafruit.com/products/3056

#define WIFI_SSID      "Test WiFi"
#define WIFI_PASS      "my wifi password"

// comment out the following two lines if you are using fona or ethernet
#include "AdafruitIO_WiFi.h"
AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

```

set wifi ssid and password

FONA Config

If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**

```

/***** WIFI *****/

// the AdafruitIO_WiFi client will work with the following boards:
// - HUZZAH ESP8266 Breakout -> https://www.adafruit.com/products/2471
// - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
// - Feather M0 WiFi -> https://www.adafruit.com/products/3010
// - Feather WICED -> https://www.adafruit.com/products/3056

#define WIFI_SSID      "Test WiFi"
#define WIFI_PASS      "my wifi password"

// comment out the following two lines if you are using fona or ethernet
// #include "AdafruitIO_WiFi.h"
// AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);

```

comment out default wifi config lines

Next, remove the comments from both of the FONA config lines in the FONA section of **config.h** to enable FONA support.

```

/***** FONA *****/

// the AdafruitIO_FONA client will work with the following boards:
// - Feather 32u4 FONA -> https://www.adafruit.com/product/3027

// uncomment the following two lines if you are using fona or ethernet
// #include "AdafruitIO_FONA.h"
// AdafruitIO_FONA io(IO_USERNAME, IO_KEY);

```

uncomment both fona config lines

Ethernet Config

If you wish to use the Ethernet Wing to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**

```
/****** WIFI *****/  
// comment out default  
// wifi config lines  
// - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821  
// - Feather WiFi -> https://www.adafruit.com/products/3010  
// - Adafruit Fona -> https://www.adafruit.com/products/3056  
  
#define WIFI_SSID "Test WiFi"  
#define WIFI_PASSWORD "my wifi password"  
  
// comment out the following two lines if you are using fona or ethernet  
// #include "AdafruitIO_WiFi.h"  
// AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Next, remove the comments from both of the Ethernet config lines in the Ethernet section of **config.h** to enable Ethernet Wing support.

```
/****** ETHERNET *****/  
// the Adafruit Ethernet FeatherWing boards:  
// - Ethernet FeatherWing -> https://www.adafruit.com/products/3201  
// uncomment both  
// ethernet config lines  
// #include "AdafruitIO_Ethernet.h"  
// AdafruitIO_Ethernet io(IO_USERNAME, IO_KEY);
```

Next, we will look at how the example sketch works.

Arduino Code

The **adafruitio_15_temp_humidity** example uses digital pin **2** by default on all boards, and that can be modified if needed by changing the **DATA_PIN** define.

```
// pin connected to DH22 data line  
#define DATA_PIN 2
```

The next chunk of code creates an instance of the DHT class, and also sets up feed instances for the **temperature** and **humidity** feeds.

```
// create DHT22 instance  
DHT_Unified dht(DATA_PIN, DHT22);  
  
// set up the 'temperature' and 'humidity' feeds  
AdafruitIO_Feed *temperature = io.feed("temperature");  
AdafruitIO_Feed *humidity = io.feed("humidity");
```

The setup function initializes the **DHT22** sensor, and also connects your feather to Adafruit IO. The code will wait until you have a valid connection to Adafruit IO before continuing with the sketch. If you have any issues connecting, check **config.h** for any typos in your username or key.

```
void setup() {  
  
  // start the serial connection  
  Serial.begin(115200);  
  
  // wait for serial monitor to open  
  while(! Serial);  
  
  // initialize dht22  
  dht.begin();  
  
  // connect to io.adafruit.com  
  Serial.print("Connecting to Adafruit IO");  
  io.connect();  
  
  // wait for a connection  
  while(io.status() < AIO_CONNECTED) {  
    Serial.print(".");  
    delay(500);  
  }  
  
  // we are connected  
  Serial.println();  
  Serial.println(io.statusText());  
  
}
```

Next, we have the main **loop()** function. The first line of the loop function calls **io.run()**; this line will need to be present at the top of your loop in every sketch. It helps keep your device connected to Adafruit IO, and processes any incoming data.

```
void loop() {  
  
  // io.run(); is required for all sketches.  
  // it should always be present at the top of your loop  
  // function. it keeps the client connected to  
  // io.adafruit.com, and processes any incoming data.  
  io.run();  
  
}
```

The next chunk of code inside the **loop()** checks the current DHT22 temperature value, and saves the value in the **celsius** and **fahrenheit** variables.

We then print both celsius and fahrenheit to the Arduino Serial Monitor, and save the fahrenheit value to the **temperature** feed on Adafruit IO.

```
sensors_event_t event;  
dht.temperature().getEvent(&event);  
  
float celsius = event.temperature;  
float fahrenheit = (celsius * 1.8) + 32;  
  
Serial.print("celsius: ");
```

```

Serial.print(celsius);
Serial.println("C");

Serial.print("fahrenheit: ");
Serial.print(fahrenheit);
Serial.println("F");

// save fahrenheit (or celsius) to Adafruit IO
temperature->save(fahrenheit);

```

If you prefer to log **celsius** values, you can modify the call to the **save()** function.

```

temperature->save(celsius);

```

The final chunk of the **loop()** function requests a humidity reading from the DHT22, and prints the value to the Arduino Serial Monitor. We also save the humidity value to the humidity feed on Adafruit IO.

```

dht.humidity().getEvent(&event);

Serial.print("humidity: ");
Serial.print(event.relative_humidity);
Serial.println("%");

// save humidity to Adafruit IO
humidity->save(event.relative_humidity);

// wait 5 seconds (5000 milliseconds == 5 seconds)
delay(5000);
}

```

Upload the sketch to your board, and open the Arduino Serial Monitor. Your board should now connect to Adafruit IO.

```

Connecting to Adafruit IO....
Adafruit IO connected.

```

You should now see the temperature and humidity values being sent to Adafruit IO.

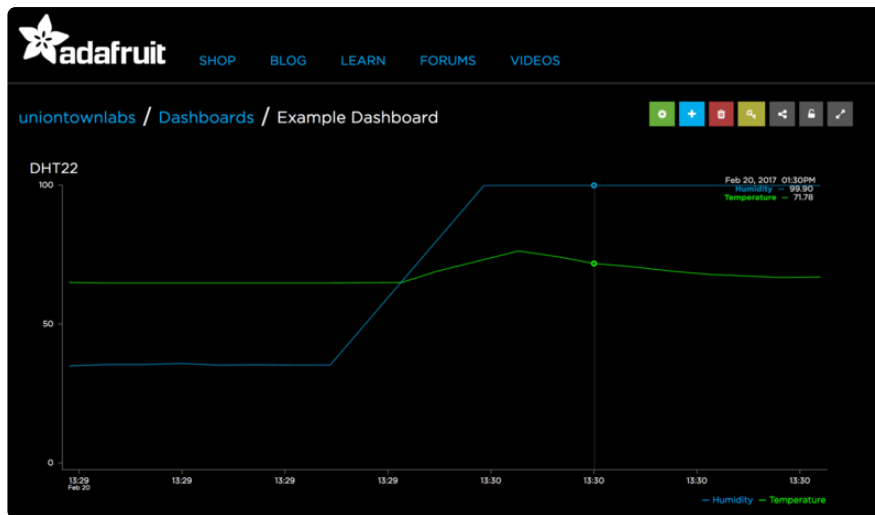
```

celsius: 18.30C
fahrenheit: 64.94F
humidity: 34.90%

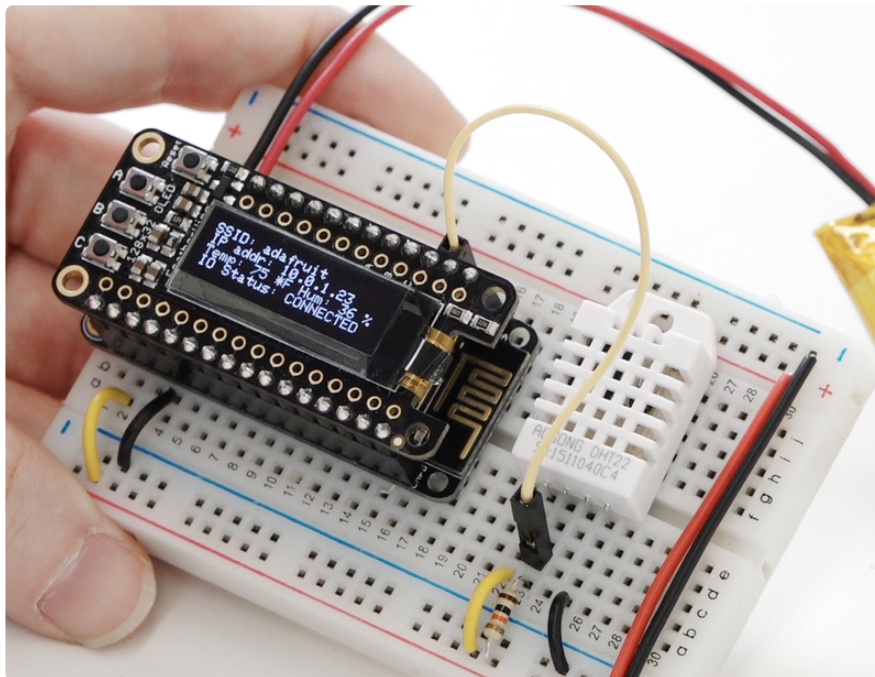
celsius: 18.20C
fahrenheit: 64.76F
humidity: 35.40%

```

Check your dashboard on Adafruit IO, and you should see the line chart update with the changes in temperature and humidity.



Add an OLED



Now that you've got a graphing weather device using IO, you can add an OLED feather so you can see network status, IP address, and the latest measurements!

Plug the OLED FeatherWing on top of your Feather, and [check out our guide to get set up and test it \(https://adafru.it/nek\)](https://adafru.it/nek)! Once you've verified that the OLED works, you can use this new code. Use the same `config.h` file from the previous section, just replace the 'main' tab of code:

This is just the main code, and does not include the config.h - use the same config.h you had from the previous working demo!

```

// Adafruit IO Temperature & Humidity Example
// Tutorial Link: https://learn.adafruit.com/adafruit-io-basics-temperature-and-humidity
//
// Adafruit invests time and resources providing this open source code.
// Please support Adafruit and open source hardware by purchasing
// products from Adafruit!
//
// Written by Todd Treece for Adafruit Industries
// Copyright (c) 2016-2017 Adafruit Industries
// Licensed under the MIT license.
//
// All text above must be included in any redistribution.

/***** Configuration *****/

// edit the config.h tab and enter your Adafruit IO credentials
// and any additional configuration needed for WiFi, cellular,
// or ethernet clients.
#include "config.h"

/***** Example Starts Here *****/
#include <Adafruit_Sensor.h>;
#include <DHT.h>;
#include <DHT_U.h>;
#include <Adafruit_SSD1306.h>;

// oled display
Adafruit_SSD1306 oled = Adafruit_SSD1306(128, 32, &Wire);

// pin connected to DH22 data line
#define DATA_PIN 2

// create DHT22 instance
DHT_Unified dht(DATA_PIN, DHT22);

// set up the 'temperature' and 'humidity' feeds
AdafruitIO_Feed *temperature = io.feed("temperature");
AdafruitIO_Feed *humidity = io.feed("humidity");

void setup() {
  oled.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C
  (for the 128x32)
  oled.display();

  // start the serial connection
  Serial.begin(115200);

  // wait for serial monitor to open
  while(! Serial);

  // initialize dht22
  dht.begin();

  // connect to io.adafruit.com
  Serial.print("Connecting to Adafruit IO");
  io.connect();

  // wait for a connection
  while(io.status() < AIO_CONNECTED) {
    Serial.print(".");
    delay(500);
  }

  // we are connected
  Serial.println();
  Serial.println(io.statusText());
}

```

```

// text display tests
oled.setTextSize(1);
oled.setTextColor(WHITE);
}

void loop() {

  // io.run(); is required for all sketches.
  // it should always be present at the top of your loop
  // function. it keeps the client connected to
  // io.adafruit.com, and processes any incoming data.
  io.run();

  sensors_event_t event;
  dht.temperature().getEvent(&event);

  float celsius = event.temperature;
  float fahrenheit = (celsius * 1.8) + 32;

  Serial.print("celsius: ");
  Serial.print(celsius);
  Serial.println("C");

  Serial.print("fahrenheit: ");
  Serial.print(fahrenheit);
  Serial.println("F");

  // save fahrenheit (or celsius) to Adafruit IO
  temperature->save(fahrenheit);

  dht.humidity().getEvent(&event);

  Serial.print("humidity: ");
  Serial.print(event.relative_humidity);
  Serial.println("%");

  // save humidity to Adafruit IO
  humidity->save(event.relative_humidity);

  // print it to the OLED
  oled.clearDisplay();
  oled.setCursor(0,0);
  oled.print("SSID: "); oled.println(WIFI_SSID);
  oled.print("IP: "); oled.println(WiFi.localIP());
  oled.print("Temp: "); oled.print(fahrenheit,0); oled.print(" *F ");
  oled.print("Hum: "); oled.print(event.relative_humidity,0); oled.println(" %");
  oled.print("IO Status: ");
  aio_status_t aio_status = io.status();
  Serial.print("Status: "); Serial.println(aio_status);
  switch (aio_status) {
    case AIO_IDLE: oled.println("IDLE"); break;
    case AIO_DISCONNECTED:
    case AIO_NET_DISCONNECTED: oled.println("DISCONNECT"); break;
    case AIO_NET_CONNECTED:
    case AIO_CONNECTED_INSECURE:
    case AIO_CONNECTED: oled.println("CONNECTED"); break;
  }
  oled.display();

  // wait 5 seconds (5000 milliseconds == 5 seconds)
  delay(2000);
}

```

Python Wiring

We're going to use a combination of the Adafruit IO Client Library and Adafruit's CircuitPython to control a Raspberry Pi over Adafruit IO.

Parts

1 x [Raspberry Pi 3 - Model B+](https://www.adafruit.com/product/3775)

<https://www.adafruit.com/product/3775>

The Raspberry Pi is a small linux board compatible with Adafruit IO projects.

If you're following along with a [Raspberry Pi \(https://adafru.it/ejq\)](https://adafru.it/ejq), we're going to use a T-Cobbler Plus for the IO Basics Projects. This add-on prototyping board lets you easily connect a Raspberry Pi (Raspberry Pi Model Zero, A+, B+, Pi 2, Pi 3) to a solderless breadboard:

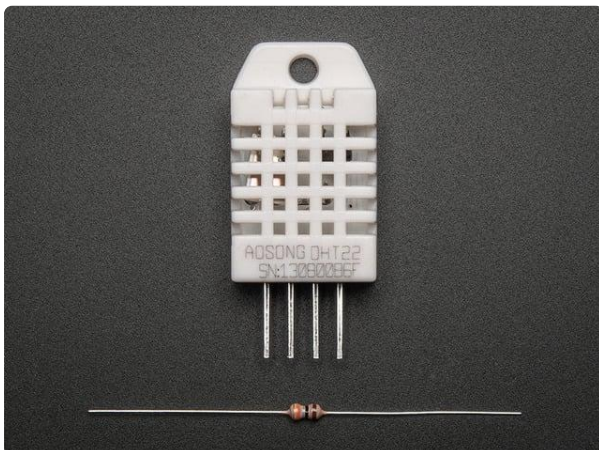


[Assembled Pi T-Cobbler Plus - GPIO Breakout](https://www.adafruit.com/product/2028)

This is the assembled version of the Pi T-Cobbler Plus. It only works with the Raspberry Pi Model Zero, A+, B+, Pi 2, Pi 3 & Pi 4! (Any Pi with 2x20...

<https://www.adafruit.com/product/2028>

We'll also need a temperature and humidity sensor. The DHT22 is a low-cost temperature and humidity sensor which is easy to wire up. You'll also need a 10k ohm resistor.

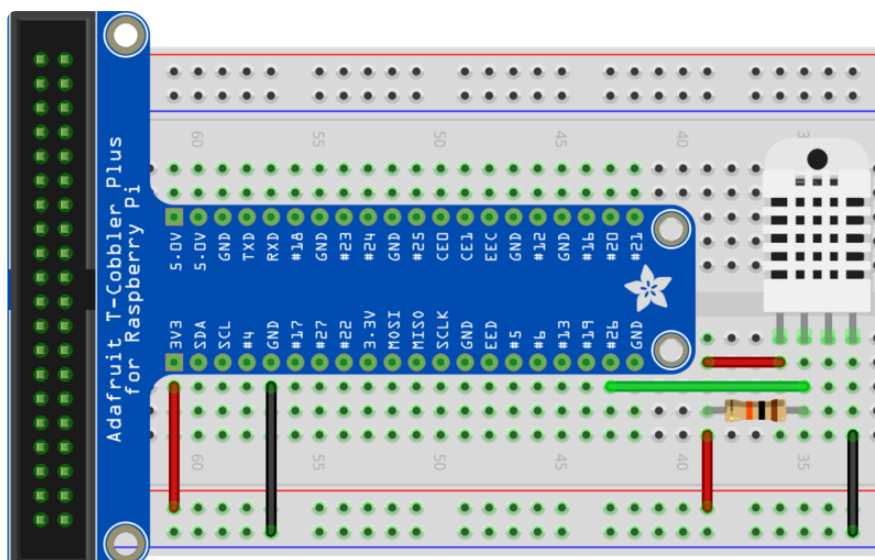


[DHT22 temperature-humidity sensor + extras](https://www.adafruit.com/product/385)

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital...

<https://www.adafruit.com/product/385>

Wiring



Make the following connections between the Pi and the DHT22:

- **Pi 3.3V to DHT Pin 1**
- **Pi Digital Pin 26 to DHT Pin 2**
- **Pi GND to DHT Pin 4**
- **10k ohm resistor (left pin) to DHT Pin 2**
- **10k ohm resistor (right pin) to GND**

Python Setup

If you've been following along with the **IO Basics: Python** guides, we are not using Adafruit Blinka for this specific tutorial. Blinka doesn't have a special timing module called `pulseio` (**yet!**).

Instead, we are going to use the Adafruit Python DHT Sensor Library and the Adafruit IO Python Client Library.

The latest Raspbian (currently this is `Stretch`) is required for the installation of Adafruit IO + Blinky.

Update your Pi and Python

We'll assume you've already gotten your Raspberry Pi up and running and can log into the command line.

Go ahead and **ssh** into your Raspberry Pi via terminal or a ssh client:

```
ssh pi@raspberrypi.local
```

Run the standard updates:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

and

```
sudo pip3 install --upgrade setuptools
```

Make sure you're using Python 3!

The default python on your computer may not be python 3. Python 2 is officially discontinued and all our libraries are Python 3 only.

We'll be using **python3** and **pip3** in our commands, use those versions of python and pip to make sure you're using 3 and not 2

Installing Adafruit Python DHT Library

We're ready to install the [Adafruit Python DHT Library \(https://adafru.it/dCJ\)](https://adafru.it/dCJ). This library will allow our Raspberry Pi to read the temperature and humidity values from the DHT22. We'll use **pip** to install the library from PyPi.

Enter the following in your terminal to install the Adafruit Python DHT Library:

```
sudo pip3 install Adafruit_DHT
```

Installing Adafruit IO Python Library



We'll also need to install the [Adafruit IO Python Client Library \(https://adafru.it/eli\)](https://adafru.it/eli) to communicate with Adafruit IO.

Run the following command to install the **Adafruit IO Client for Python**:

```
pip3 install adafruit-io
```

If the installation gives you 'insufficient permissions' errors, add 'sudo' before the call to pip3

Downloading Example Code

The example code is contained within the Adafruit IO Python Client's examples/basics subdirectory.

Navigate to the root directory of the Pi by entering the following in the terminal:

```
cd ~
```

Download the latest version of the Adafruit IO Python Client's GitHub repository by running

```
git clone https://github.com/adafruit/io-client-python.git (https://adafru.it/osB)
```

Navigate to the folder's example code for the examples

```
cd io-client-python/examples/basics/
```

Python Code

The `temp_humidity.py` code uses Raspberry Pi GPIO Pin 26 by default. If you'd like to use a different pin, change the **DHT_DATA_PIN** variable.

```
DHT_DATA_PIN = 26
```

Before running, we'll need to set our Adafruit IO Key and Adafruit IO Username. You can find both of these on your [Adafruit IO profile page \(https://adafru.it/BmD\)](https://adafru.it/BmD)

```
# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username).
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'
```

The next chunk of code creates an instance of the Adafruit IO REST client, sets up the **temperature** and **humidity** feeds, and sets up the DHT22 sensor.

```
# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

# Set up Adafruit IO Feeds.
temperature_feed = aio.feeds('temperature')
humidity_feed = aio.feeds('humidity')

# Set up DHT22 Sensor.
dht22_sensor = Adafruit_DHT.DHT22
```

in the **while True** loop, we first try to grab the humidity and sensor readings using **Adafruit_DHT.read_retry** which retries (up to 15 times) to get a sensor reading.

```
humidity, temperature = Adafruit_DHT.read_retry(sensor, DHT_DATA_PIN)
```

If the DHT sensor receives a reading, it'll print out both of the values and send them to the Adafruit IO temperature and humidity feeds.

```
print('Temp={0:0.1f}*C Humidity={1:0.1f}%'.format(temperature, humidity))
aio.send(temperature.key, temperature)
aio.send(humidity.key, humidity)
```

Sometimes you won't get a sensor reading and the results will be null (because Linux can't guarantee the timing of calls to read to the sensor). If that occurs, we'll print to the terminal. Then, we sleep for **DHT_READ_TIMEOUT** until the next read.

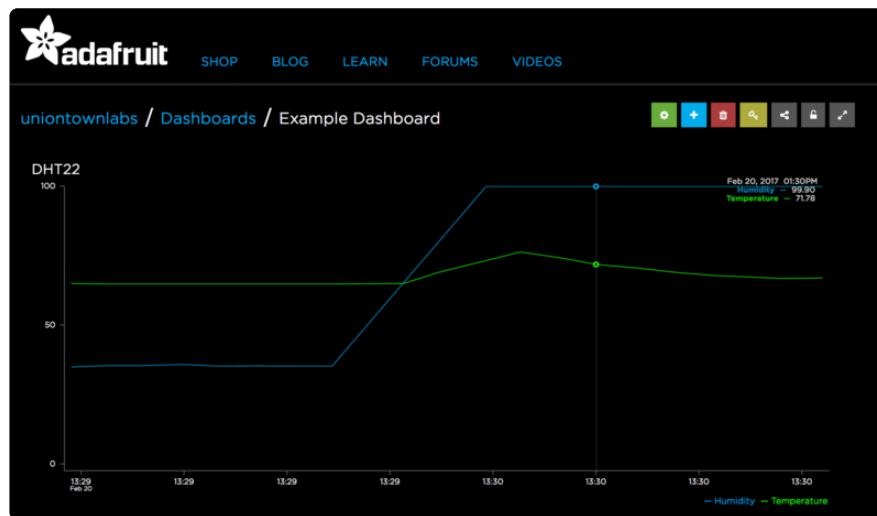
In your terminal, enter the following command to run the code:

```
python3 temp_humidity.py
```

You should now see the temperature and humidity values being sent to Adafruit IO.

```
Temp=25.5*C Humidity=59.1%
Temp=25.5*C Humidity=59.1%
Temp=25.5*C Humidity=59.1%
Temp=25.4*C Humidity=59.0%
```

Check your dashboard on Adafruit IO, and you should see the line chart update with the changes in temperature and humidity.



Code

```
"""
'temp_humidity.py'
=====
Example of sending temperature and humidity data to Adafruit IO

Author(s): Brent Rubell

Tutorial Link: Tutorial Link: https://learn.adafruit.com/adafruit-io-basics-temperature-and-humidity

Dependencies:
- Adafruit IO Python Client
  (https://github.com/adafruit/io-client-python)
- Adafruit_CircuitPython_AHTx0
  (https://github.com/adafruit/Adafruit_CircuitPython_AHTx0)
"""

# import standard python modules.
import time

# import adafruit-blinka modules
import board

# import Adafruit IO REST client.
from Adafruit_IO import Client, Feed, RequestError

# Import AHTx0 library
```

```

import adafruit_ahtx0

# Set true to send tempertaure data in degrees fahrenheit ('f')?
USE_DEGREES_F = False

# Time between sensor reads, in seconds
READ_TIMEOUT = 60

# Set to your Adafruit IO key.
# Remember, your key is a secret,
# so make sure not to publish it when you publish this code!
ADAFRUIT_IO_KEY = 'YOUR_AIO_KEY'

# Set to your Adafruit IO username.
# (go to https://accounts.adafruit.com to find your username).
ADAFRUIT_IO_USERNAME = 'YOUR_AIO_USERNAME'

# Create an instance of the REST client.
aio = Client(ADAFRUIT_IO_USERNAME, ADAFRUIT_IO_KEY)

# Assign a temperature feed, if one exists already
try:
    temperature_feed = aio.feeds('temperature')
except RequestError: # Doesn't exist, create a new feed
    feed_temp = Feed(name="temperature")
    temperature_feed = aio.create_feed(feed_temp)

# Assign a humidity feed, if one exists already
try:
    humidity_feed = aio.feeds('humidity')
except RequestError: # Doesn't exist, create a new feed
    feed_humid = Feed(name="humidity")
    humidity_feed = aio.create_feed(feed_humid)

# Initialize the board's default I2C bus
i2c = board.I2C() # uses board.SCL and board.SDA
# Initialize AHT20 using the default address (0x38) and the board's default i2c bus
sensor = adafruit_ahtx0.AHTx0(i2c)

while True:
    temperature = sensor.temperature
    humidity = sensor.relative_humidity
    if USE_DEGREES_F:
        temperature = temperature * 9.0 / 5.0 + 32.0
        print('Temp={0:0.1f}*F'.format(temperature))
    else:
        print('Temp={0:0.1f}*C'.format(temperature))
    print('Humidity={1:0.1f}%'.format(humidity))
    # Format sensor data as string for sending to Adafruit IO
    temperature = '%.2f'%(temperature)
    humidity = '%.2f'%(humidity)
    # Send humidity and temperature data to Adafruit IO
    aio.send(temperature_feed.key, str(temperature))
    aio.send(humidity_feed.key, str(humidity))

    # Timeout to avoid flooding Adafruit IO
    time.sleep(READ_TIMEOUT)

```

Adafruit IO FAQ

Encountering an issue with your Adafruit IO Arduino Project?

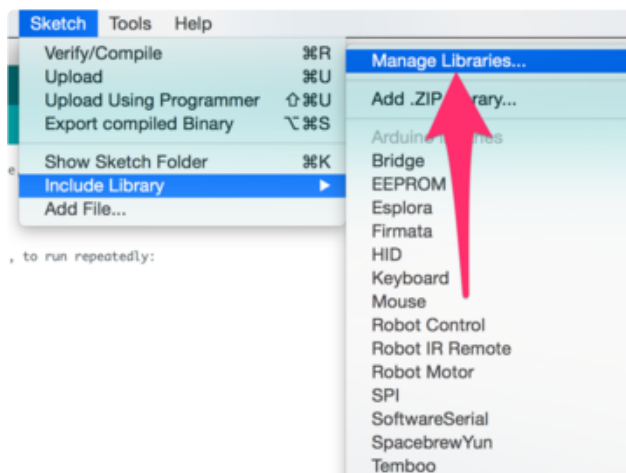
If you're having an issue compiling, connecting, or troubleshooting your project, check this page first.

Don't see your issue? [Post up on the Adafruit IO Forum with your issue \(https://adafru.it/plC\)](https://adafru.it/plC).

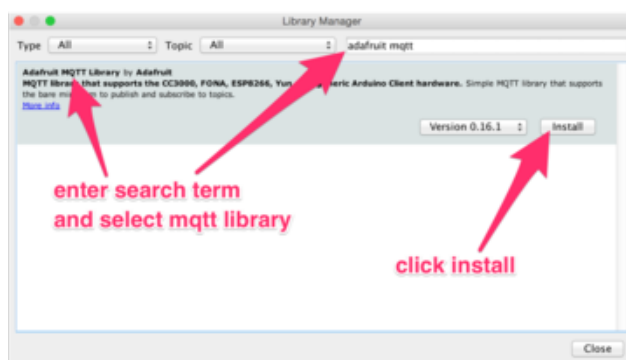
I encounter the following error when compiling my sketch:

```
fatal error: Adafruit_MQTT.h: No such file or directory, #include "Adafruit_MQTT.h"
```

The Adafruit IO Arduino library is dependent on our Adafruit IO MQTT Library.



To resolve this error, from the Arduino IDE, navigate to the **Manage Libraries...** option in the **Sketch -> Include Library** menu.



To resolve this error, from the Arduino IDE, navigate to the **Manage Libraries...** option in the **Sketch -> Include Library** menu.

My Serial Monitor prints "...\" endlessly after the "Connecting to Adafruit IO" message

Your board is not connecting to Adafruit IO, but why? Let's find out:

First, check in `config.h` that you have the correct `IO_USERNAME`, `IO_KEY`, `WIFI_SSID`, and `WIFI_PASS` are set correctly.

Next, we're going to modify the while loop which waits for an IO connection in your sketch. Change the line in the status check loop

from `Serial.println(.);` to `Serial.println(io.statusText());`

```
// wait for a connection
while(io.status() < AIO_CONNECTED) {
  Serial.println(io.statusText());
  delay(500);
}
```

Verify and re-upload the sketch. If you're receiving a **Network disconnected** error message, the board is not able to talk to the internet. Re-check your hardware, connections, and router settings.

If it's still not showing **Adafruit IO connected**, check the [IO status on the Adafruit Status page \(https://adafru.it/Oc0\)](https://adafru.it/Oc0) to make sure the service is online.

My data isn't displaying, is Adafruit IO's {service/MQTT/API} down?

Possibly - you can check [IO status on the Adafruit Status page \(https://adafru.it/Oc0\)](https://adafru.it/Oc0).

Is my data being sent properly? Am I sending too much data?

There's a [monitor page built-into Adafruit IO \(https://adafru.it/DOK\)](https://adafru.it/DOK) which provides a live view of incoming data and error messages. Keep this page open while you send data to your Adafruit IO devices to monitor data and errors.