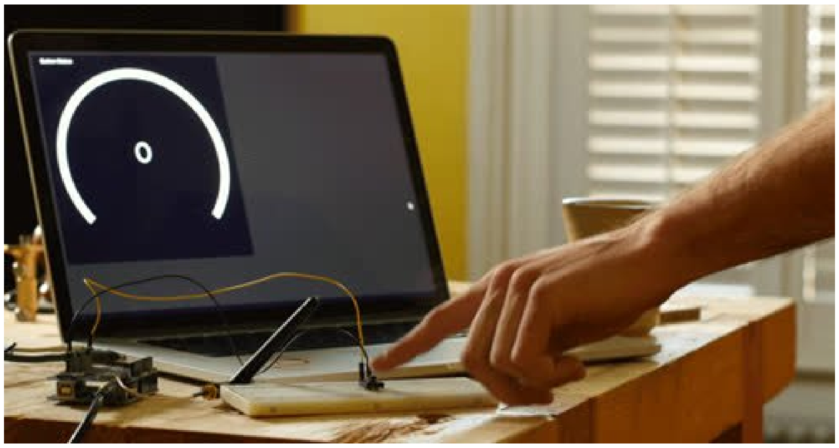
adafruit learning system

Adafruit IO Basics: Digital Input

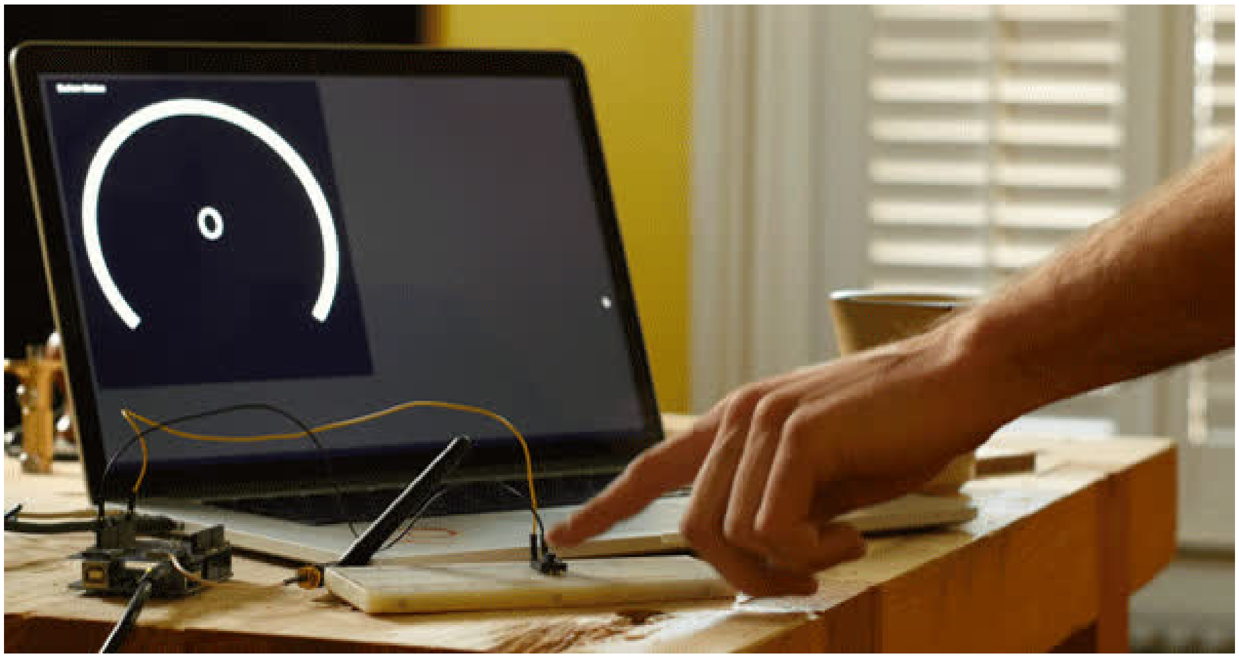
Created by Todd Treece



Last updated on 2017-07-14 11:49:29 PM UTC

Guide Contents

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 momentary button press data to Adafruit IO.

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.

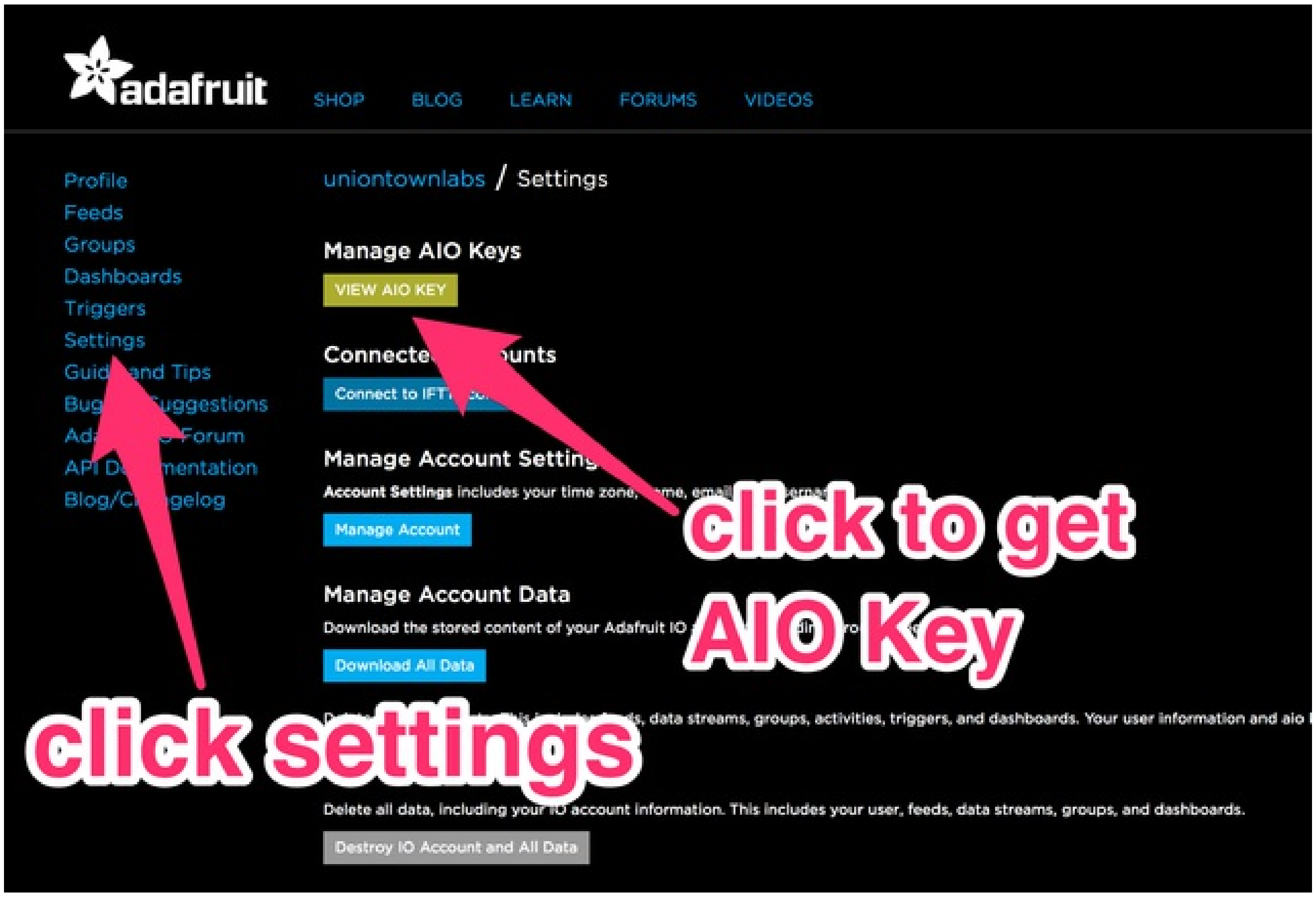
- Adafruit Feather HUZZAH ESP8266 Setup Guide

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!

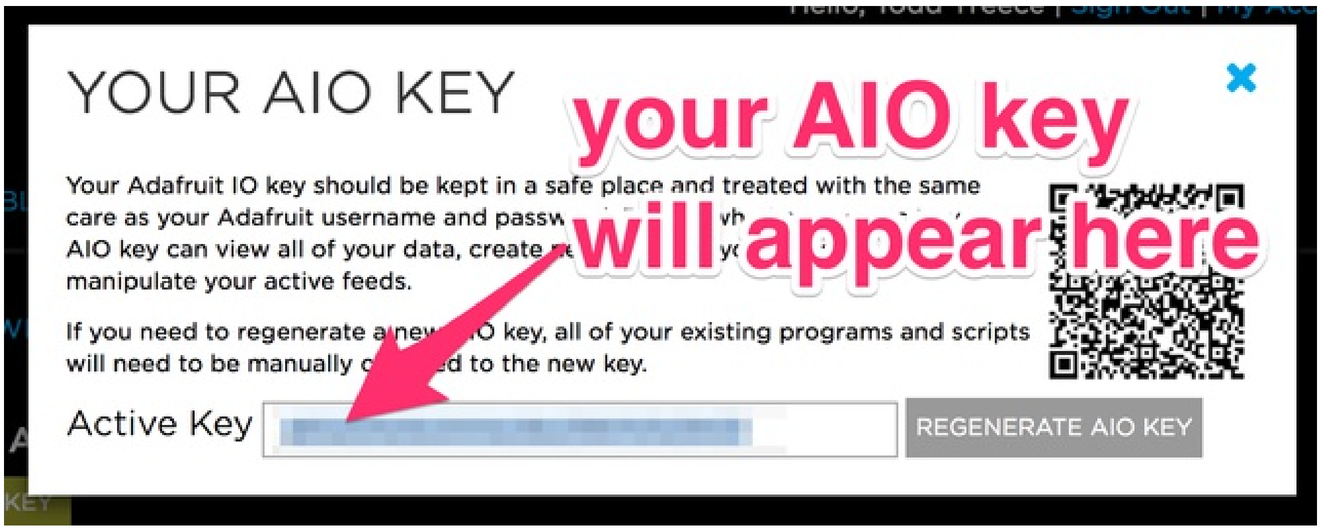
Adafruit IO Setup

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.


YOUR AIO KEY

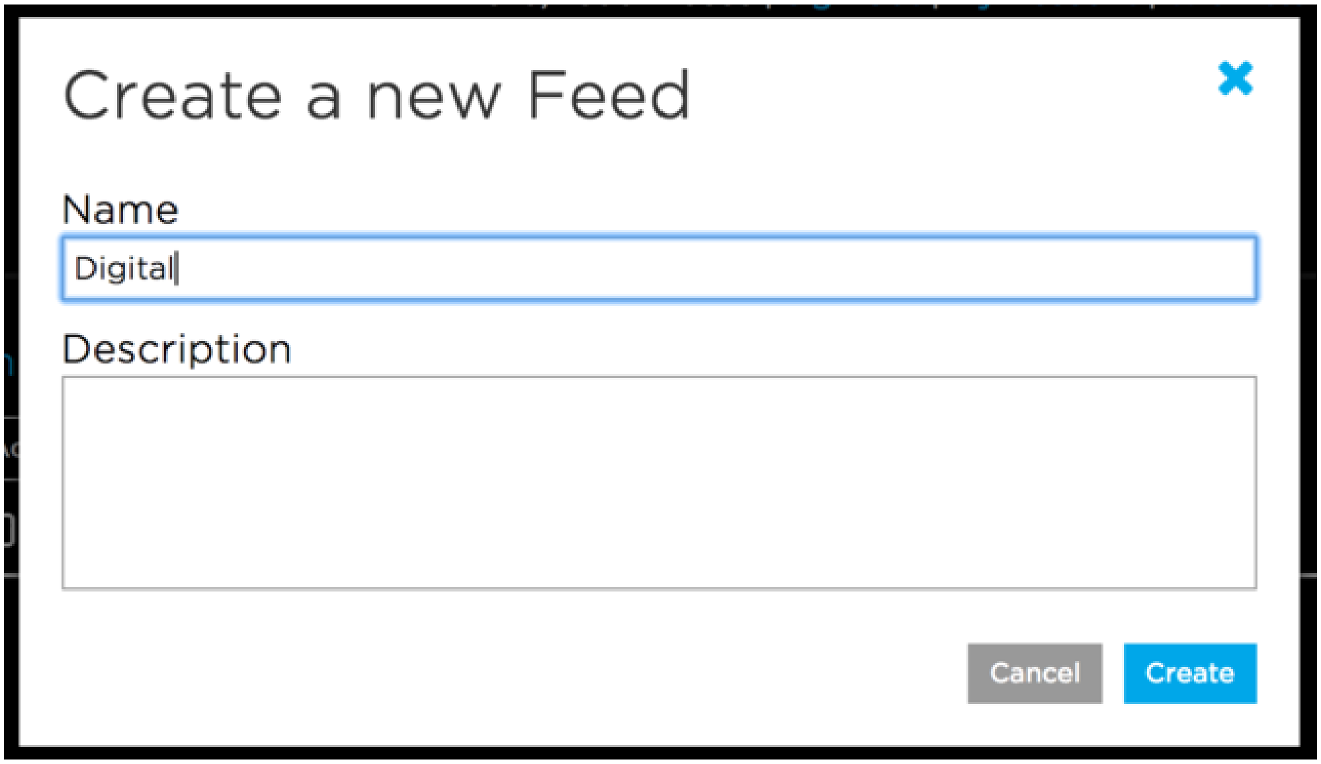
Your Adafruit IO key should be kept in a safe place and treated with the same care as your Adafruit username and passw... AIO key can view all of your data, create ... manipulate your active feeds.

If you need to regenerate a new ... key, all of your existing programs and scripts will need to be manually c...d to the new key.

Active Key [redacted]

Creating the Digital Feed

Next, you will need to create a feed called**Digital**. If you need help getting started with creating feeds on Adafruit IO, check out the Adafruit IO Feed Basics guide (http://adafru.it/ioA).


Create a new Feed

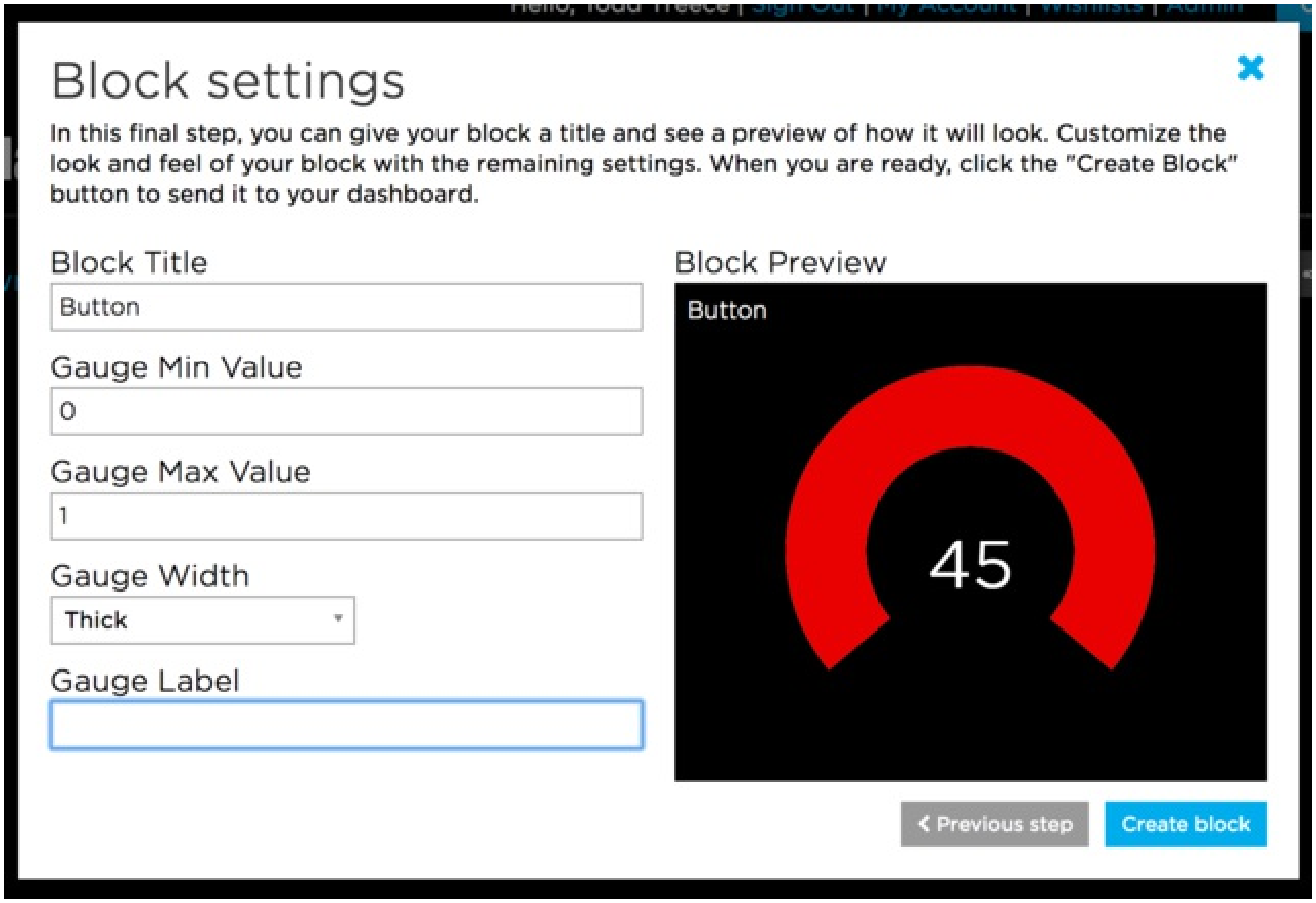
Name: Digital

Description:

Adding the Gauge Block

Next, add a new Gauge block to a new or existing dashboard. Name the block whatever you would like, and *give it a* ***max value*** *of 1 and a* ***min value*** *of 0*. Make sure you have selected the ***Digital*** feed as the data source for the gauge.

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



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: Button

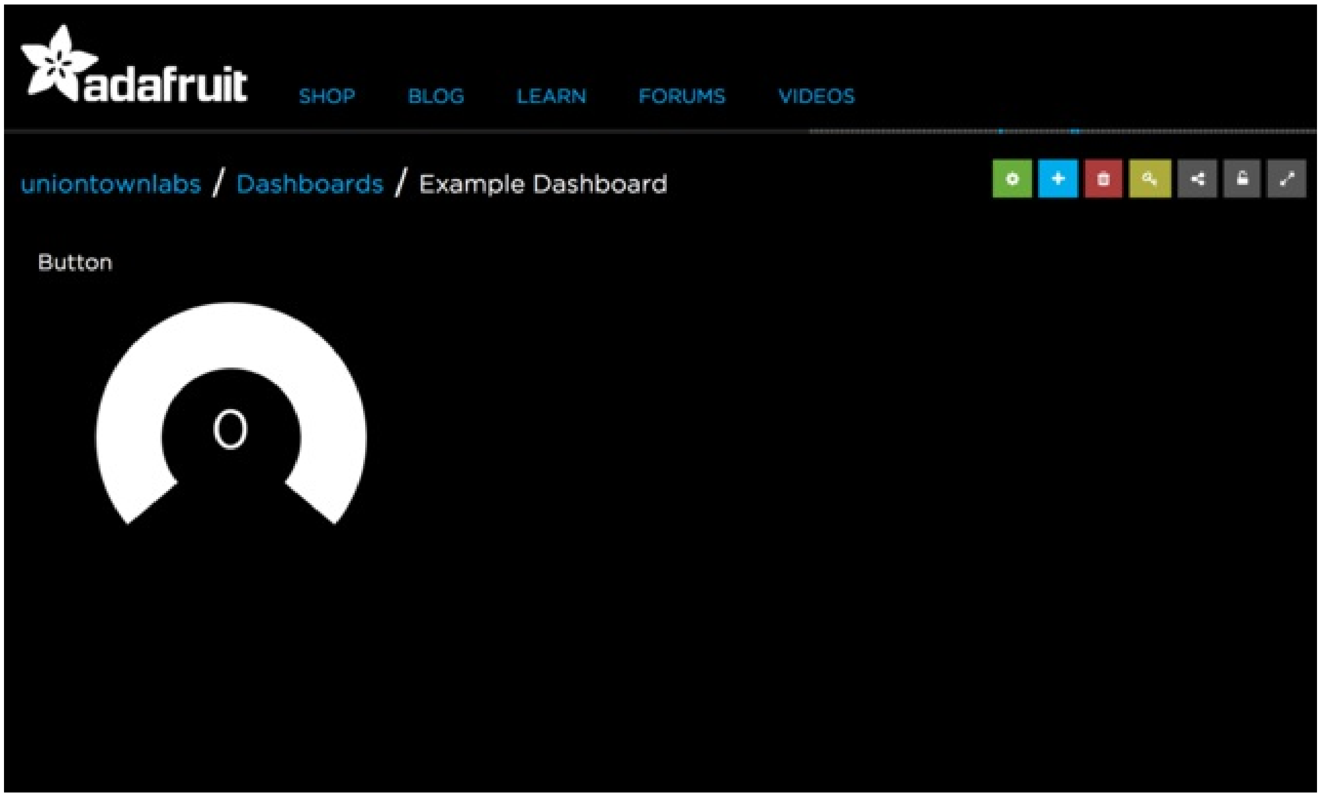
Gauge Min Value: 0

Gauge Max Value: 1

Gauge Width: Thick

Gauge Label:

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



Next, we will look at wiring the circuit.

Wiring

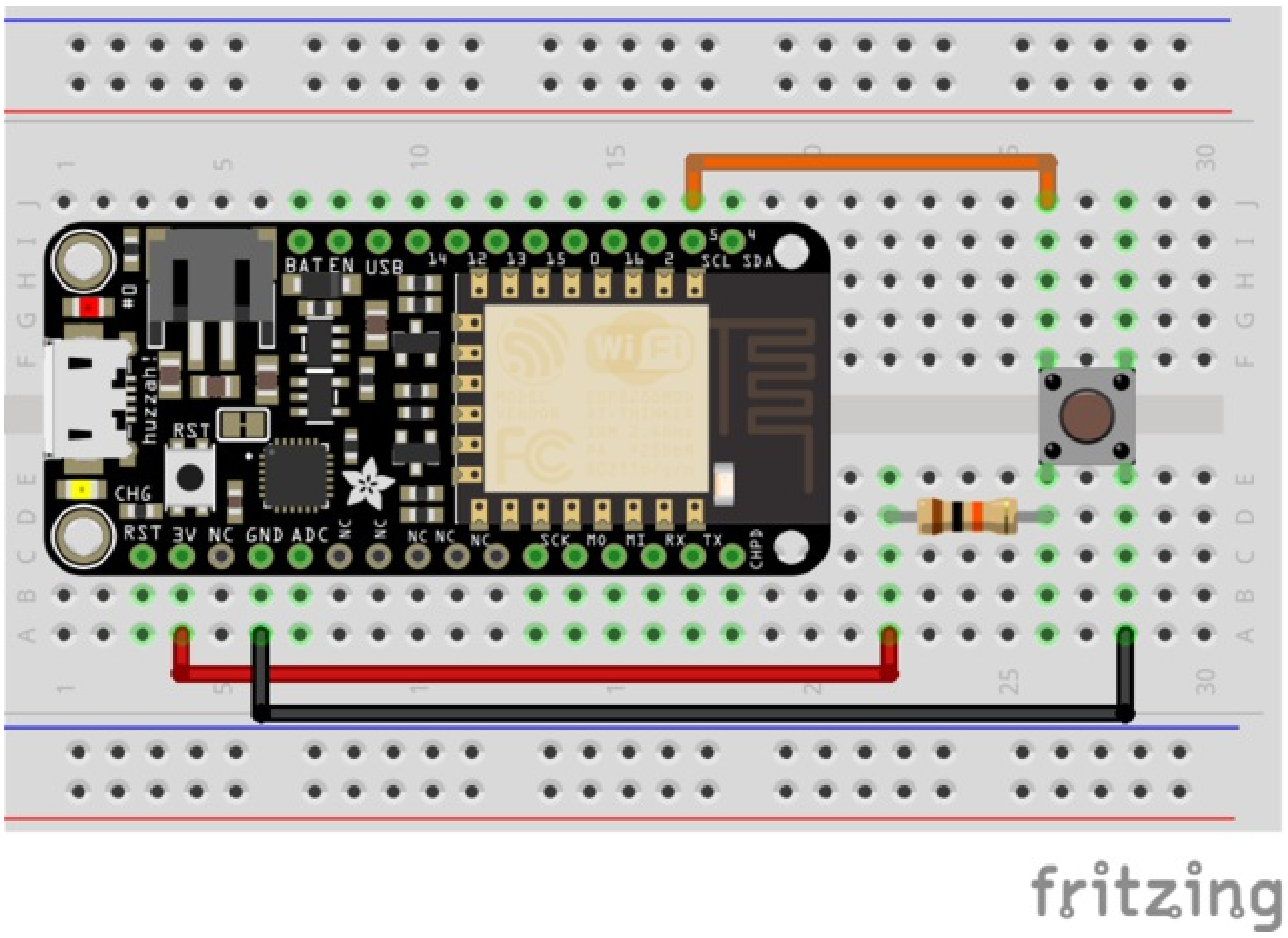
You will need the following parts for this tutorial:

- **1x** Adafruit IO compatible Feather
- **3x** jumper wires
- **1x** 10k resistor
- **1x** momentary button

You will need to connect the following pins to the button and 10k resistor:

- Feather **GND** to one side of the momentary button
- Feather **Pin 5** to the other side of the momentary button
- Feather **3V** to one leg of a **10k** resistor
- The second leg of the **10k** resistor to the same side of the momentary button as **Pin 5**

Note: Resistors are *not* polarized, so the 10k resistor can be connected to the circuit in either direction.



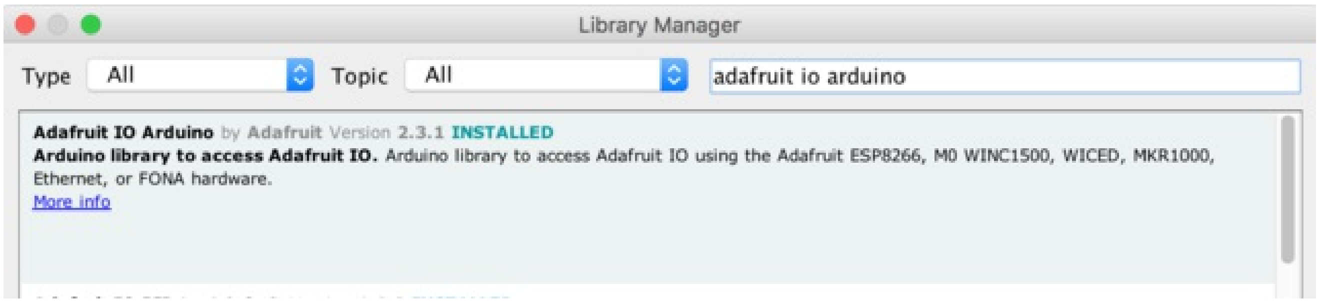
Next, let's look at the example sketch we will be using.

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

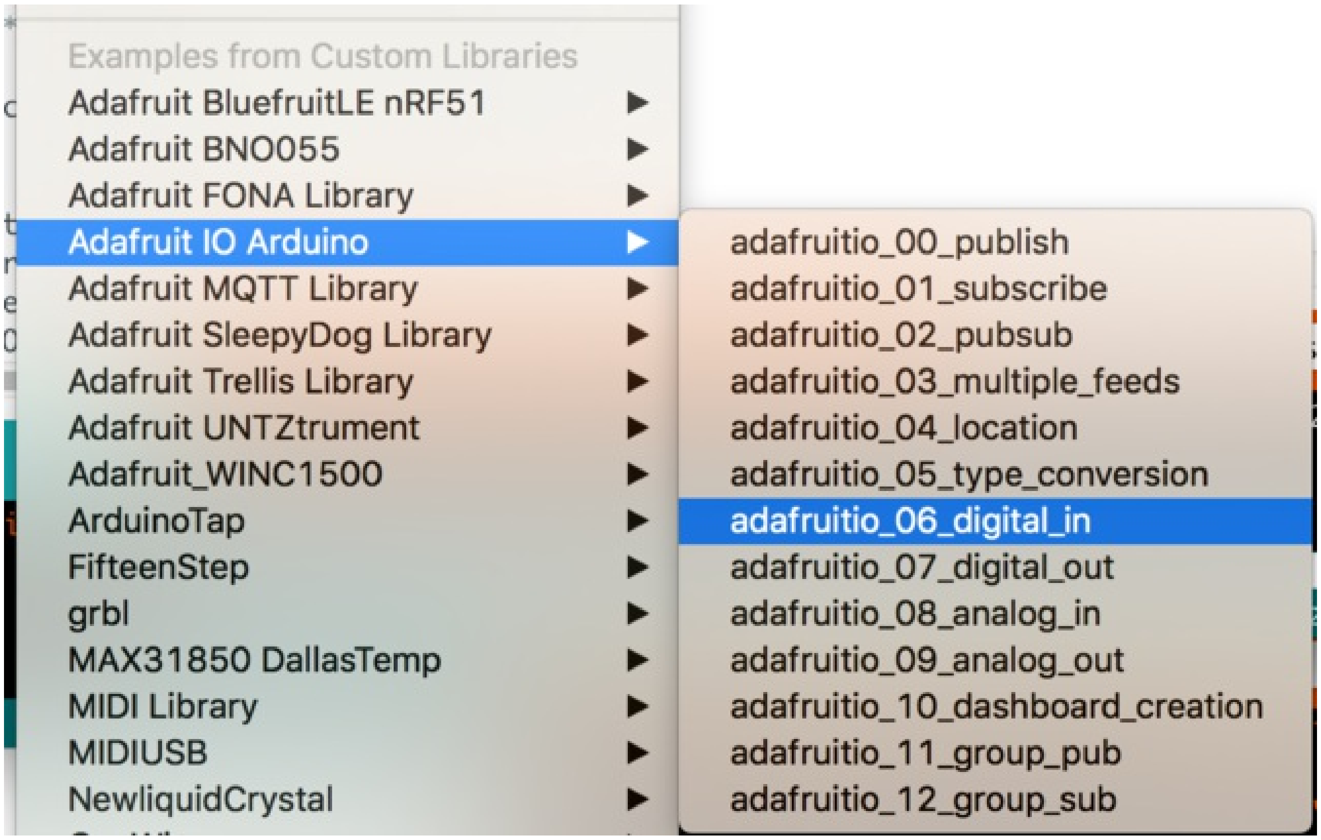
You will need to make sure you have at least**version 2.3.1** of the Adafruit IO Arduino library installed before continuing. You will also need the Arduino HTTP library and Adafruit MQTT library so check the setup guide above to get all set up!



Type: All
Topic: All
adafruit io arduino

Adafruit IO Arduino by Adafruit Version 2.3.1 **INSTALLED**
Arduino library to access Adafruit IO. Arduino library to access Adafruit IO using the Adafruit ESP8266, M0 WINC1500, WICED, MKR1000, Ethernet, or FONA hardware.

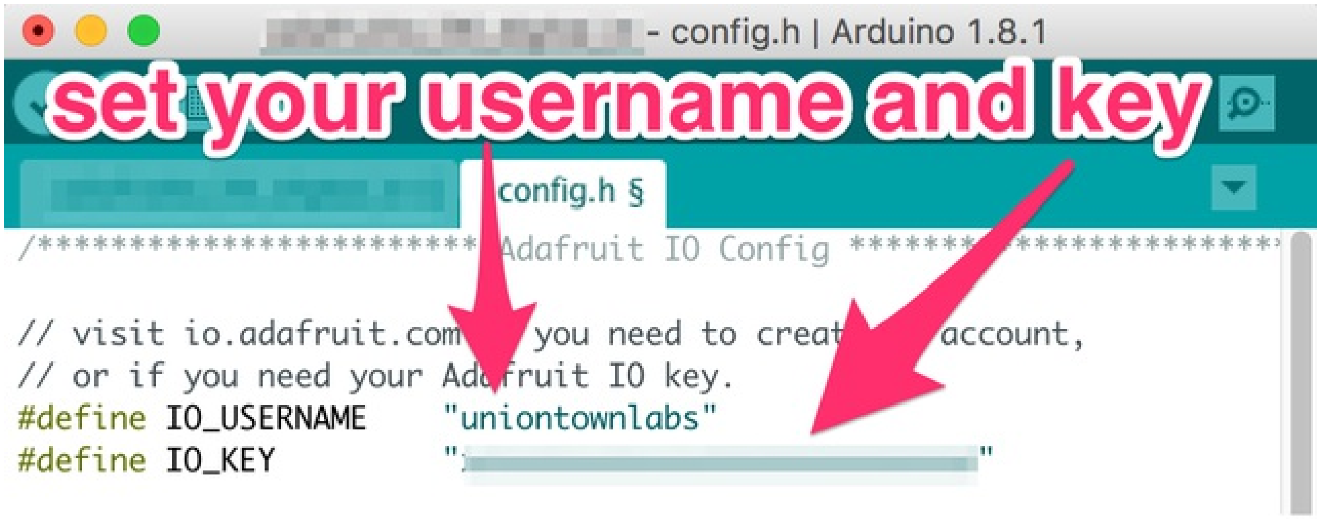
For this example, you will need to open the**adafruitio_06_digital_in** example in the **Adafruit IO Arduino** library.



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

Network Config

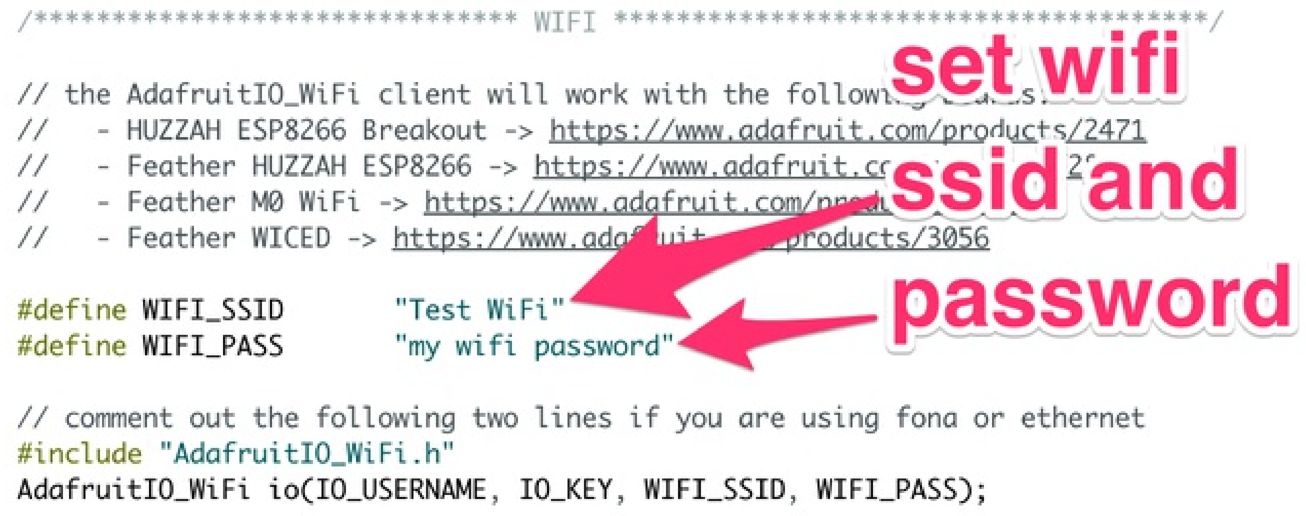
To configure the network settings, click on the **config.h** tab in the sketch. You will need to set your Adafruit IO username in the **IO_USERNAME** define, and your Adafruit IO key in the **IO_KEY** define.


[redacted] - config.h | Arduino 1.8.1
set your username and key
config.h §

```
/************************ Adafruit IO Config ******************************/

// visit io.adafruit.com    you need to creat     account,
// or if you need your Adafruit IO key.
#define IO_USERNAME    "uniontownlabs"
#define IO_KEY         "[redacted]"
```

WiFi Config

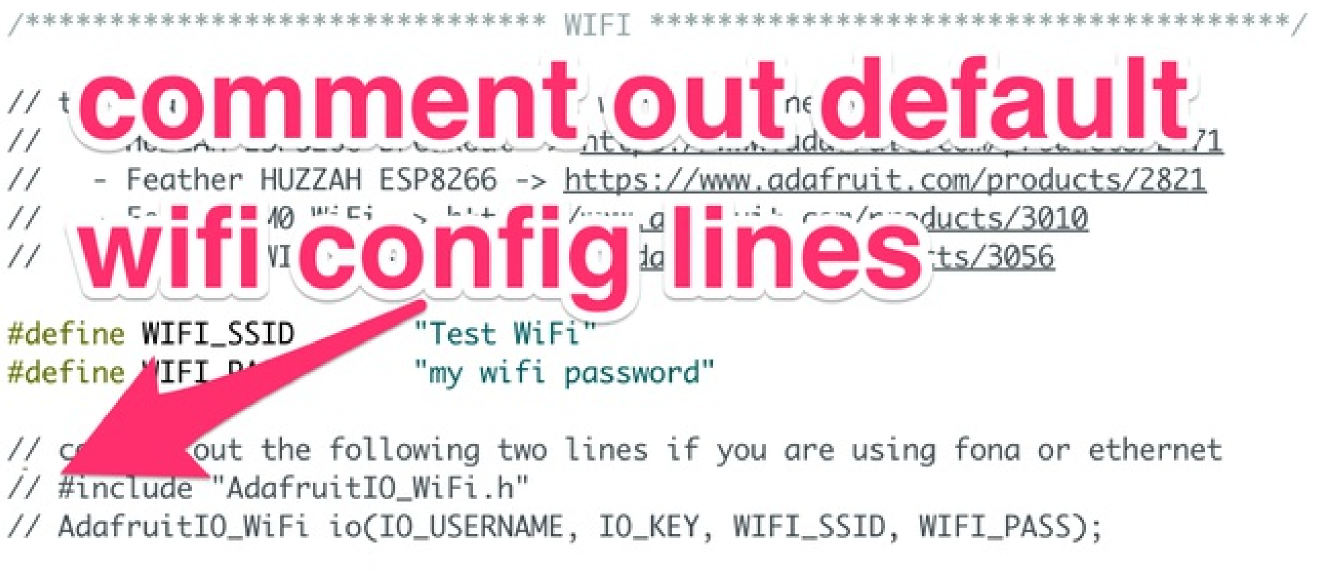
WiFi is enabled by default in **config.h** so if you are using one of the supported WiFi boards, you will only need to modify the **WIFI_SSID** and **WIFI_PASS** options in the **config.h** tab.



```
/******************************* WIFI **************************************/

// the AdafruitIO_WiFi client will work with the followi
//   - HUZZAH ESP8266 Breakout -> https://www.adafruit.com/products/2471
//   - Feather HUZZAH ESP8266 -> https://www.adafruit.cc
//   - Feather M0 WiFi -> https://www.adafruit.com/
//   - Feather WICED -> https://www.ada        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);
```

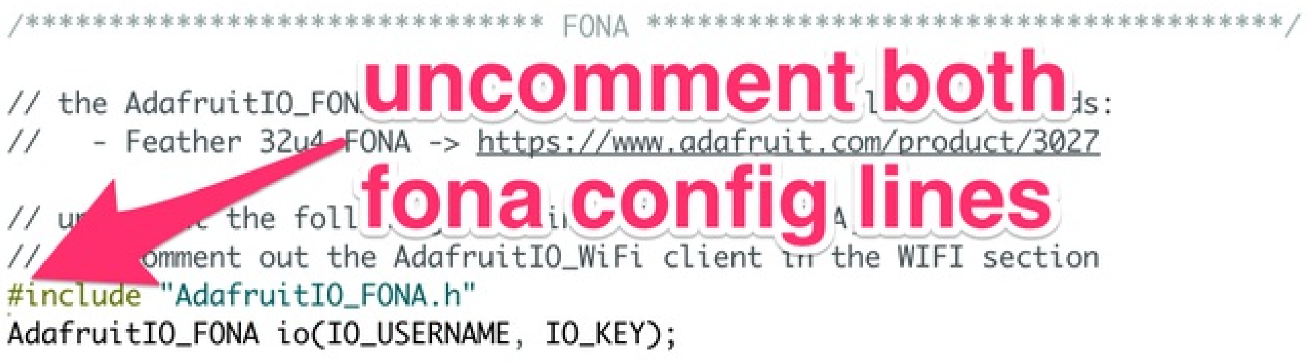
FONA Config

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 **************************************/

// t                    ne
//
//   - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
//                               ducts/3010
//                               ts/3056

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

// c      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);
```

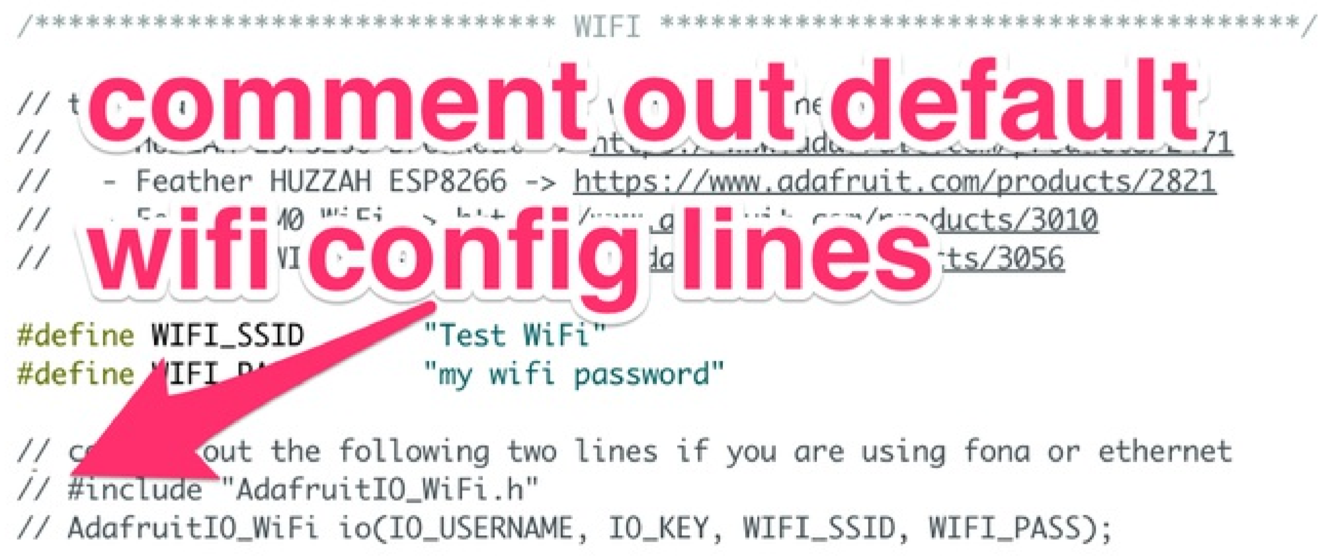
Next, remove the comments from both of the FONA config lines in the FONA section of **config.h** to enable FONA support.



```
/******************************* FONA **************************************/

// the AdafruitIO_FON                          ds:
//   - Feather 32u4 FONA -> https://www.adafruit.com/product/3027

// u        the foll
//        omment out the AdafruitIO_WiFi client in the WIFI section
#include "AdafruitIO_FONA.h"
AdafruitIO_FONA io(IO_USERNAME, IO_KEY);
```

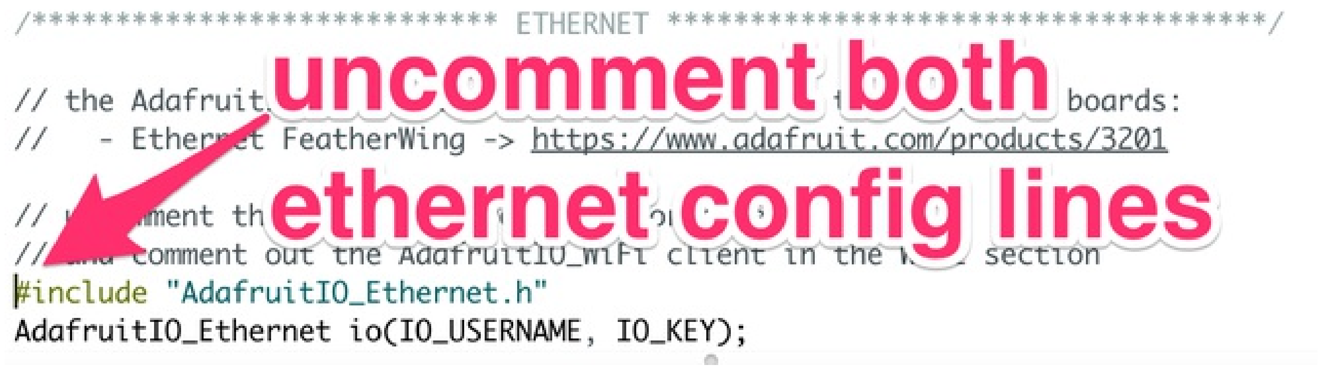
Ethernet Config

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 **************************************/

// t[illegible]
//   [illegible]
//   - Feather HUZZAH ESP8266 -> https://www.adafruit.com/products/2821
//   [illegible]ducts/3010
//   [illegible]ts/3056

#define WIFI_SSID       "Test WiFi"
#define [illegible]     "my wifi password"

// c[illegible] 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[illegible] boards:
//   - Ether[illegible] FeatherWing -> https://www.adafruit.com/products/3201

// [illegible]ment th[illegible]
// [illegible] comment out the AdafruitIO_WiFi client in the [illegible] section
#include "AdafruitIO_Ethernet.h"
AdafruitIO_Ethernet io(IO_USERNAME, IO_KEY);
```

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

Code

The **adafruitio_06_digital_in** example uses digital pin 5 by default on all boards, and that can be modified if needed by changing the **BUTTON_PIN** define.

Note: If you are using the WICED Feather, you will need to change the **BUTTON_PIN** define to *PC5* instead of the default setting of *5*.

```
/************************ Example Starts Here *******************************/

// digital pin 5
#define BUTTON_PIN 5
```

The next chunk of code sets up two boolean variables to track button state, and an Adafruit IO Feed instance for a feed called **digital**.

```
// button state
bool current = false;
bool last = false;

// set up the 'digital' feed
AdafruitIO_Feed *digital = io.feed("digital");
```

In the setup function, we set the **BUTTON_PIN** as a digital input, and connect 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() {

  // set button pin as an input
  pinMode(BUTTON_PIN, INPUT);

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

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

  // 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 state of the button, and saves the state of the button in the **current** variable. Because we are using a pullup resistor, we will need to flip the button state.

If the button state is **LOW** it means the button is pressed, so we set `current = true;`. If the button state is **HIGH** it means the button is released, so we set `current = false;`.

We then check if the **current** button state is equal to the **last** button state. If it is equal, we will return early and not continue with the rest of the loop.

```
  // grab the current state of the button.
  // we have to flip the logic because we are
  // using a pullup resistor.
  if(digitalRead(BUTTON_PIN) == LOW)
    current = true;
  else
    current = false;

  // return if the value hasn't changed
  if(current == last)
    return;
```

The final chunk of the `loop()` function prints the current value to the Arduino Serial Monitor, and sends the current value to the **digital** feed on Adafruit IO. We also set `last = current;` so we can tell if the state of the button has changed in the next run of the loop.

```
  // save the current state to the 'digital' feed on adafruit io
  Serial.print("sending button -> ");
  Serial.println(current);
  digital->save(current);

  // store last button state
  last = current;

}
```

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 can now press the button, and you should see button presses being sent to Adafruit IO.

```
sending button -> 1
sending button -> 0
sending button -> 1
sending button -> 0
```

Check your dashboard on Adafruit IO, and you should see the gauge respond to button presses.