adafruit learning system

Adafruit IO Basics: Analog Input

Created by Todd Treece

Last updated on 2017-09-12 03:10:58 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 analog values to Adafruit IO from a photocell.

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.

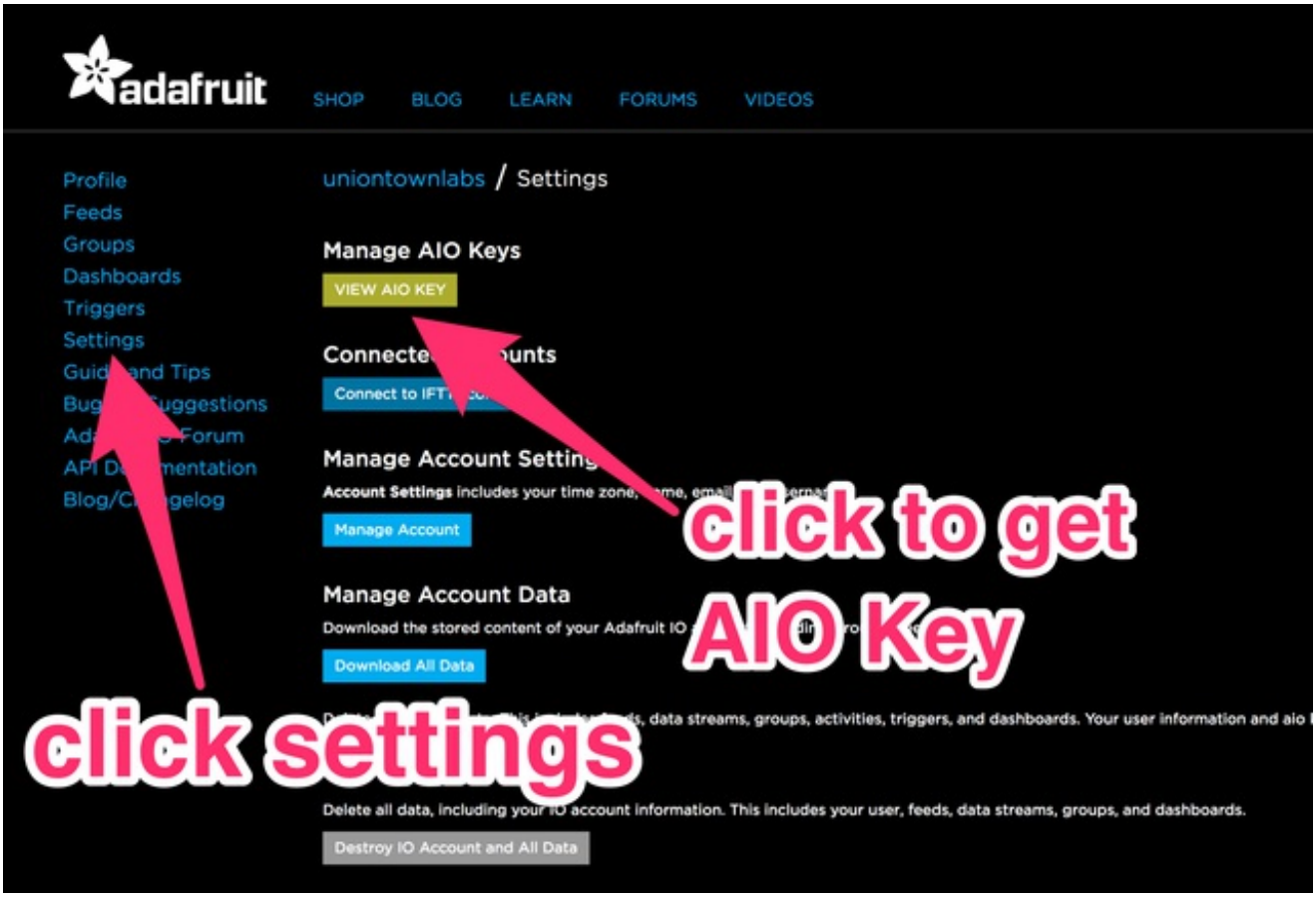
- Adafruit Feather HUZZAH ESP8266 Setup Guide

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!

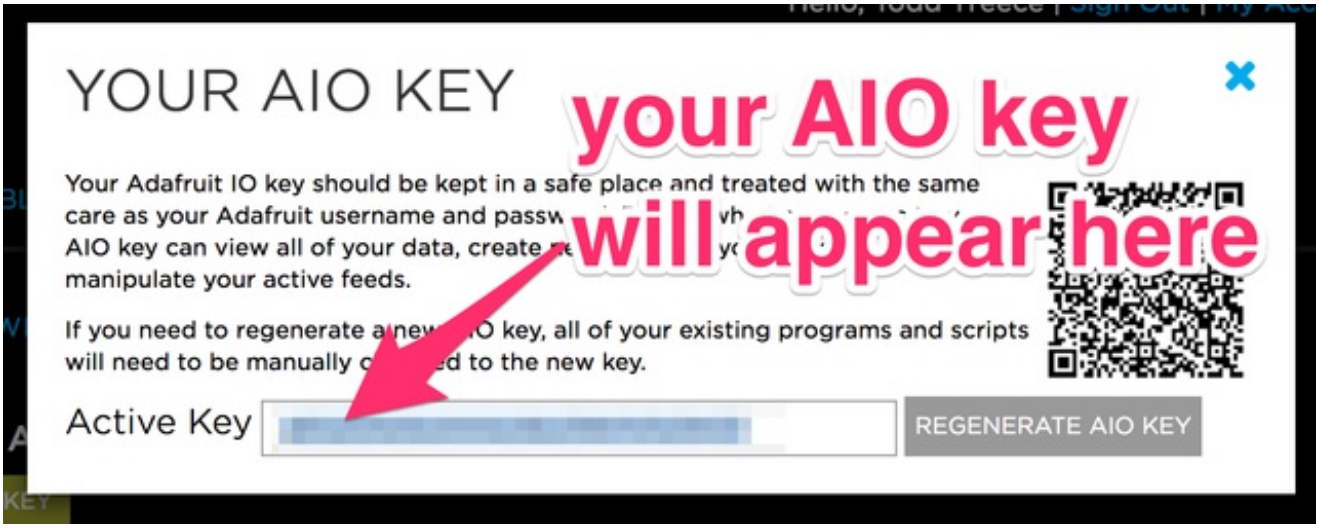
Adafruit IO Setup

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 Analog Feed

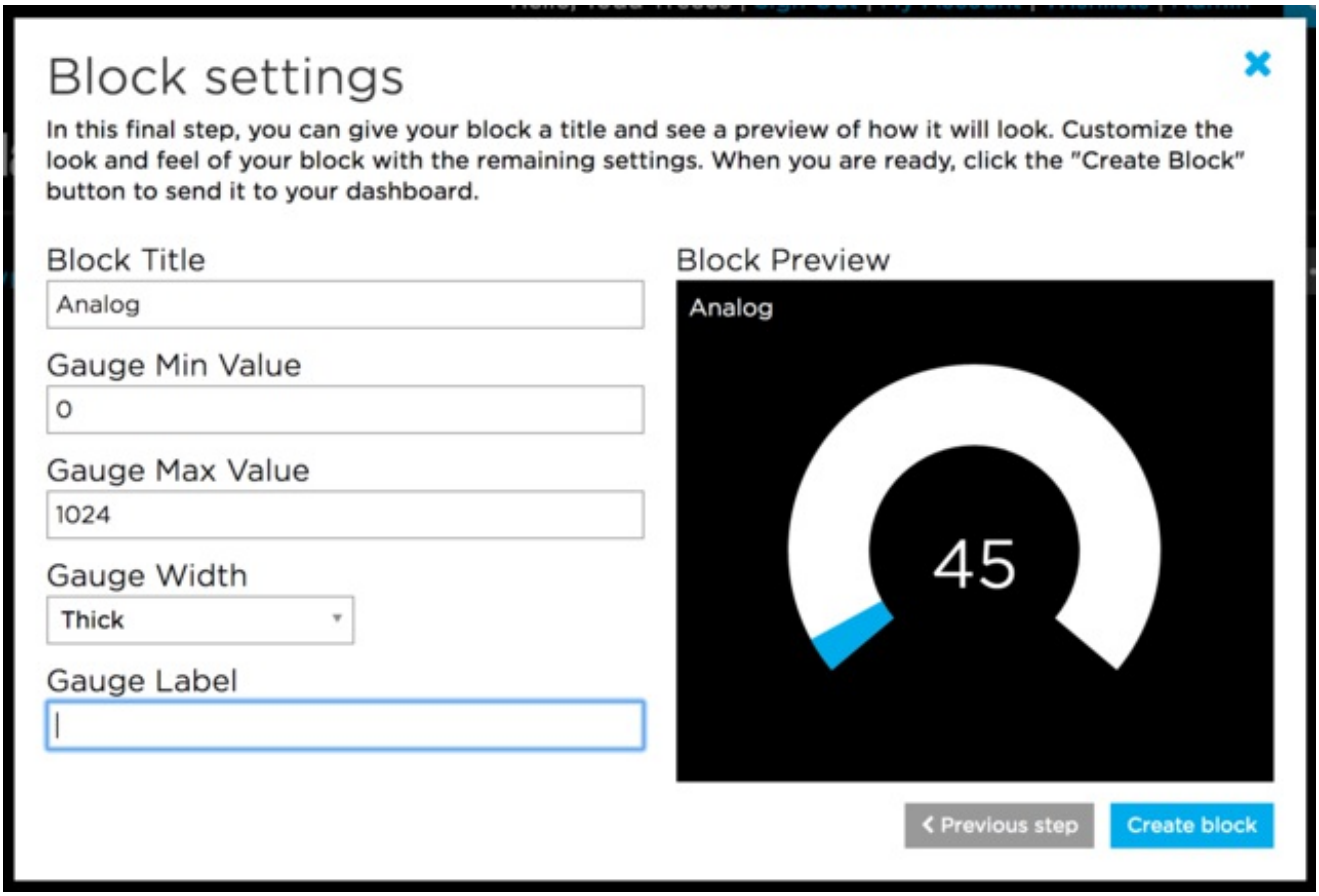
Next, you will need to create a feed called**Analog**. If you need help getting started with creating feeds on Adafruit IO, check out the [Adafruit IO Feed Basics guide](http://adafru.it/ioA) (http://adafru.it/ioA).

Adding the Gauge Block

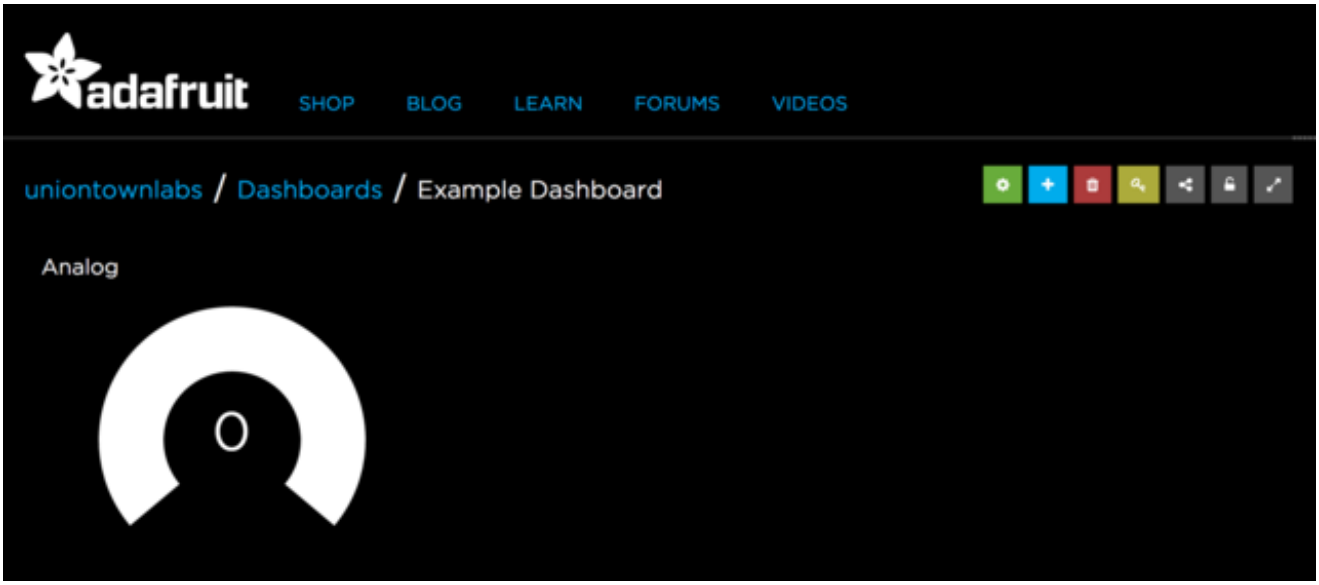
Next, add a new Gauge block to a new or existing dashboard. Name the block whatever

you would like, and *give it a **max value** of 1024 and a **min value** of 0*. Make sure you have selected the **Analog** 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).



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

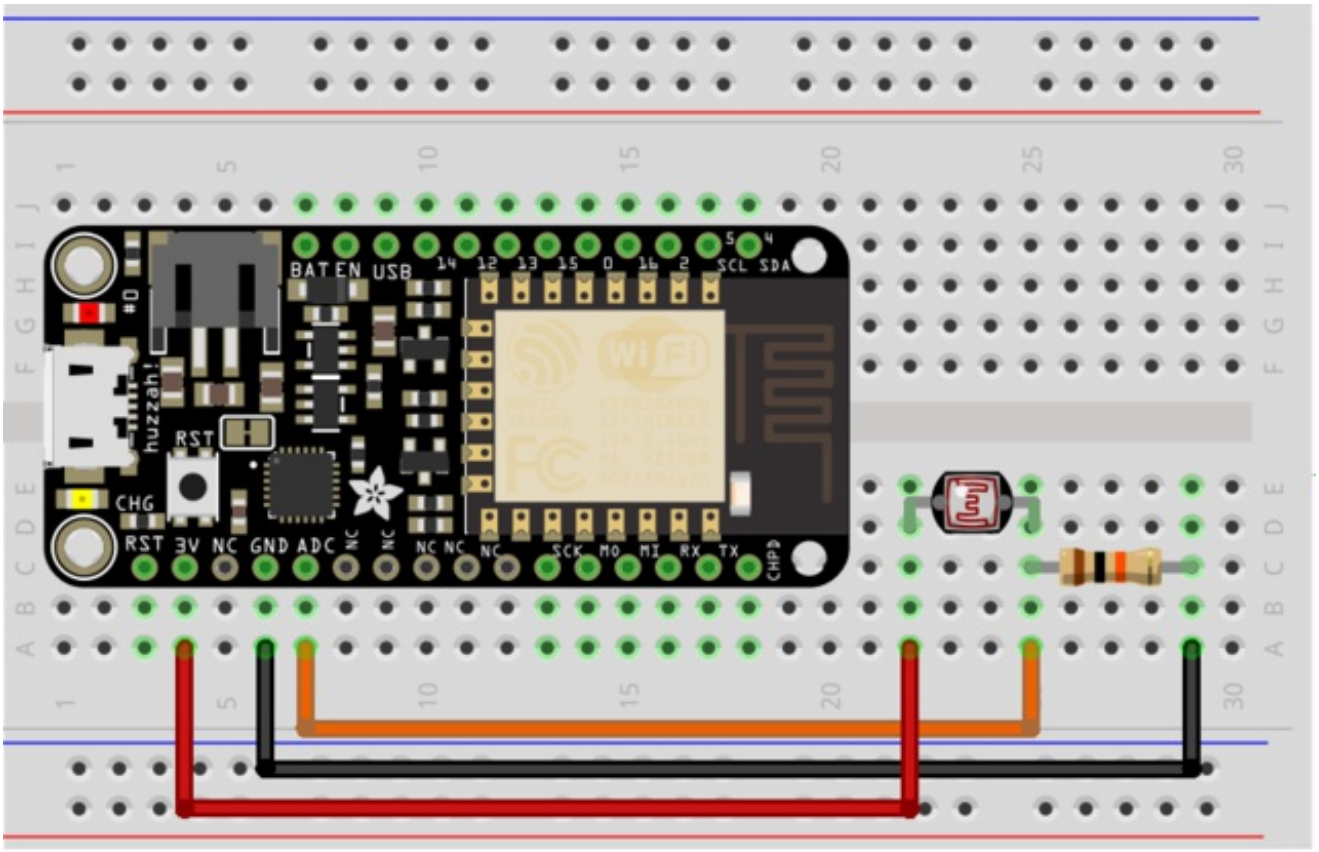
You will need the following parts for this tutorial:

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

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

- Feather **GND** to one leg of the **10k** resistor
- The second leg of the **10k** resistor to one leg of the photocell
- Feather **A0** (or ADC) to the same leg of the photocell as the **10k** resistor.
- Feather **3V** to the second leg of the photocell

Note: Resistors and photocells are *not* polarized, so they can be connected to the circuit in either direction.



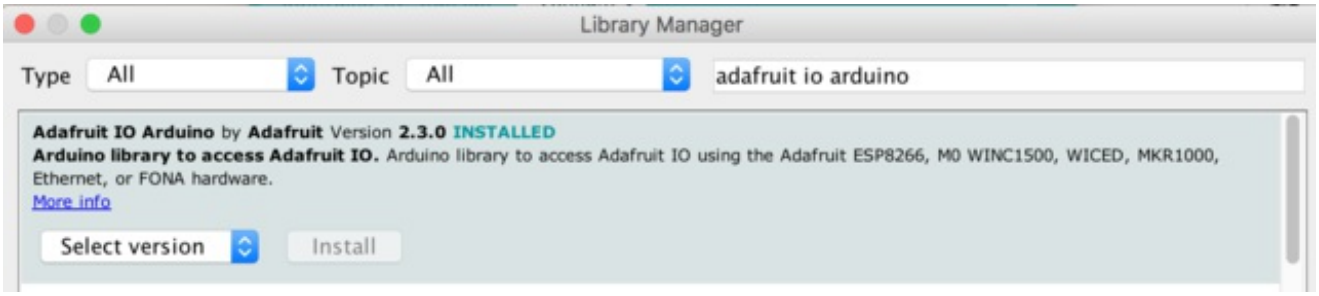
Next, let's look at the example sketch we will be using.

Arduino Setup

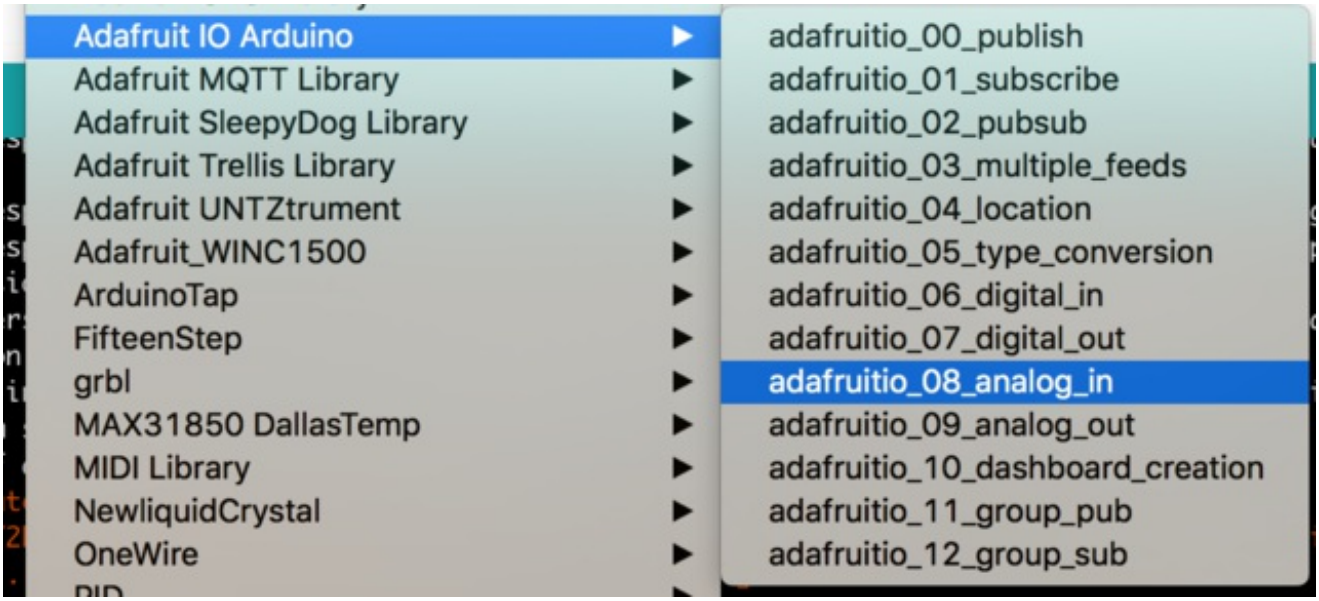
You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

- Adafruit Feather HUZZAH ESP8266 Setup Guide

You will need to make sure you have at least**version 2.3.0** of the Adafruit IO Arduino library installed before continuing.



For this example, you will need to open the**adafruitio_08_analog_in** example in the **Adafruit IO Arduino** library.



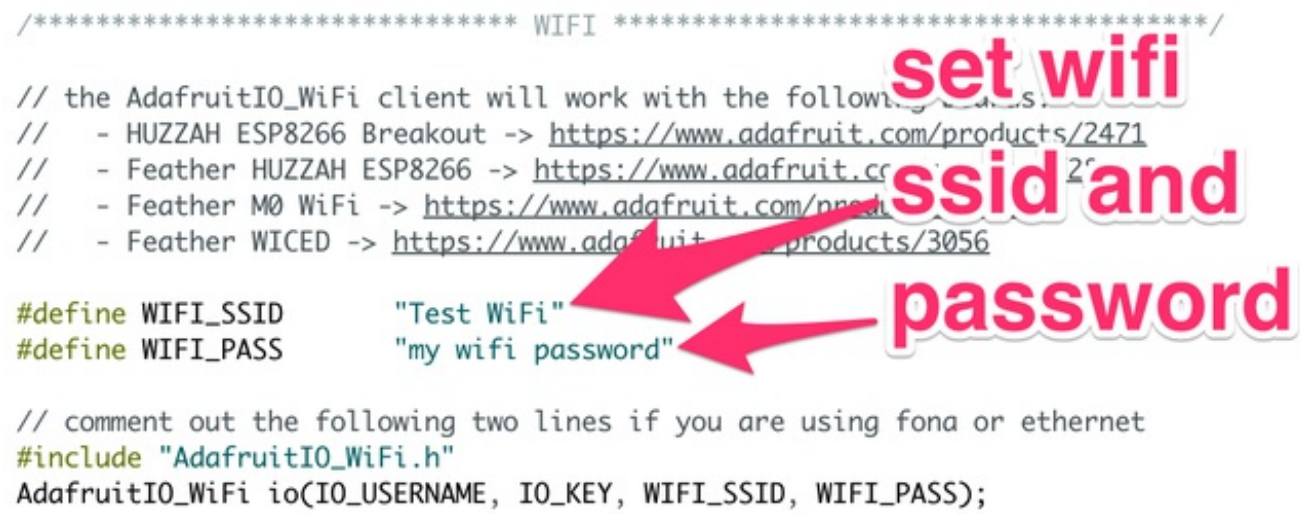
Next, we will look at the network configuration options in the sketch.

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.

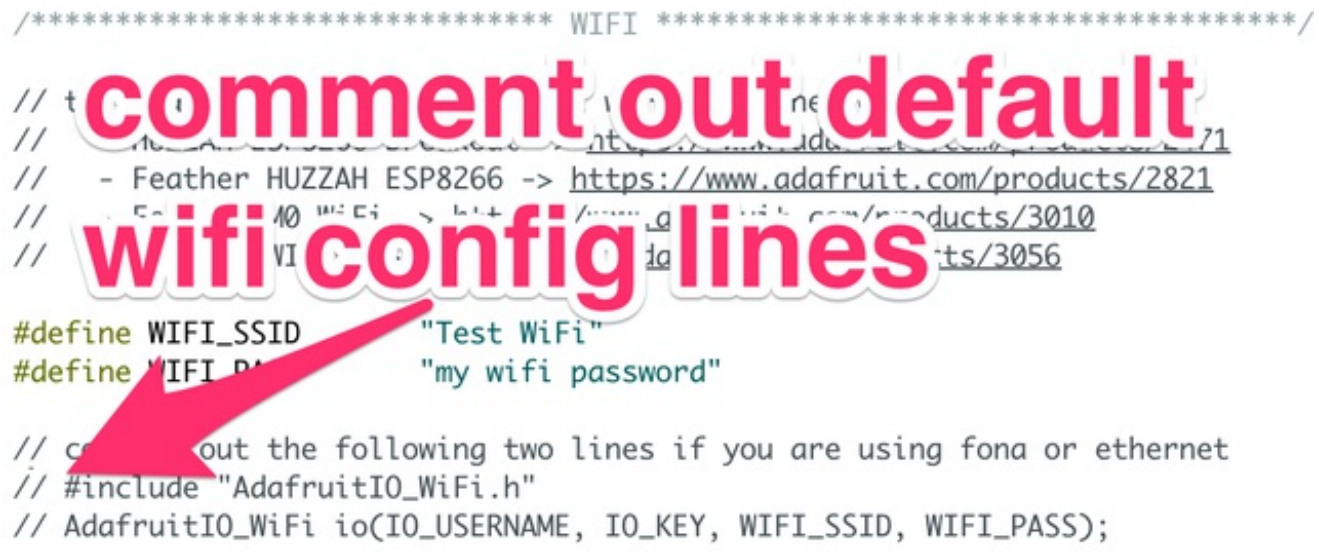
WiFi Config

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.



FONA Config

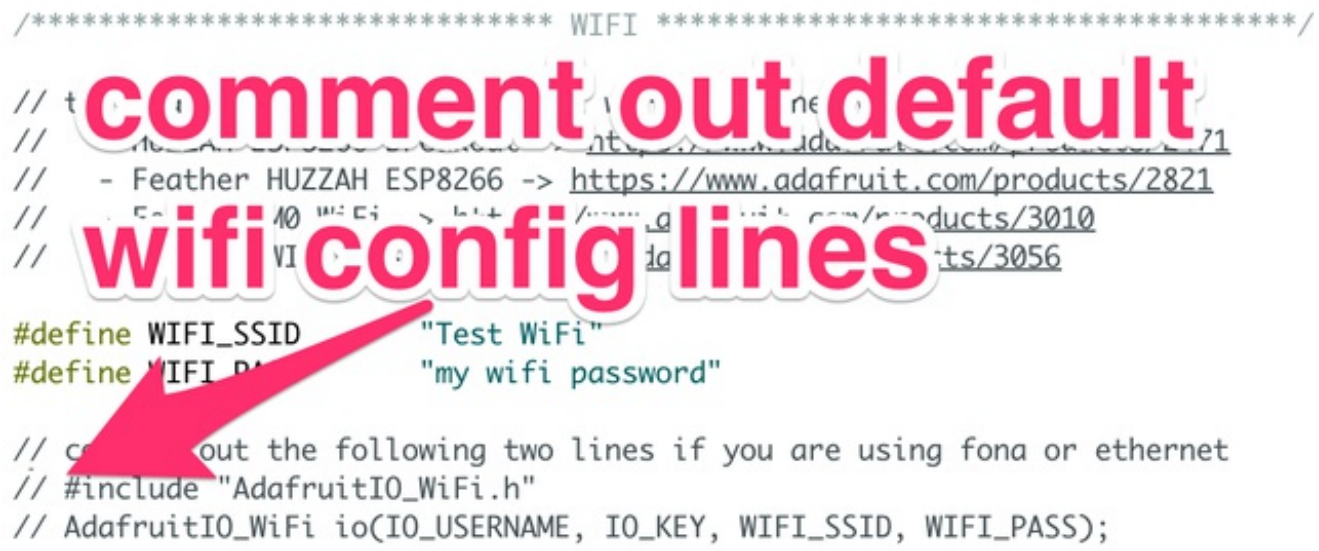
If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**



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

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**



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

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

Code

The **adafruitio_08_analog_in** example uses pin **A0** by default on all boards, and that can be modified if needed by changing the **PHOTOCELL_PIN** define.

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

// analog pin 0
#define PHOTOCELL_PIN A0
```

The next chunk of code sets up two integer variables to track the state of the photocell, and an Adafruit IO Feed instance for a feed called **analog**.

```
// photocell state
int current = 0;
int last = -1;

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

The setup function 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);

  // 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 value of the photocell, and saves the value of the photocell in the **current** variable.

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

```
  // grab the current state of the photocell
  current = analogRead(PHOTOCELL_PIN);

  // 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 **analog** feed on Adafruit IO. We also set `last = current;` so we can tell if the value of the photocell has changed in the next run of the loop.

```
  // save the current state to the analog feed
  Serial.print("sending -> ");
  Serial.println(current);
  analog->save(current);

  // store last photocell state
  last = current;

  // wait one second (1000 milliseconds == 1 second)
  delay(1000);

}
```

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 cover the photocell with your hand, and you should see the changing values being sent to Adafruit IO.

```
sending -> 1024
sending -> 953
sending -> 259
sending -> 476
sending -> 248
sending -> 173
sending -> 224
sending -> 291
sending -> 1024
```

Check your dashboard on Adafruit IO, and you should see the gauge respond to the changes in photocell values.