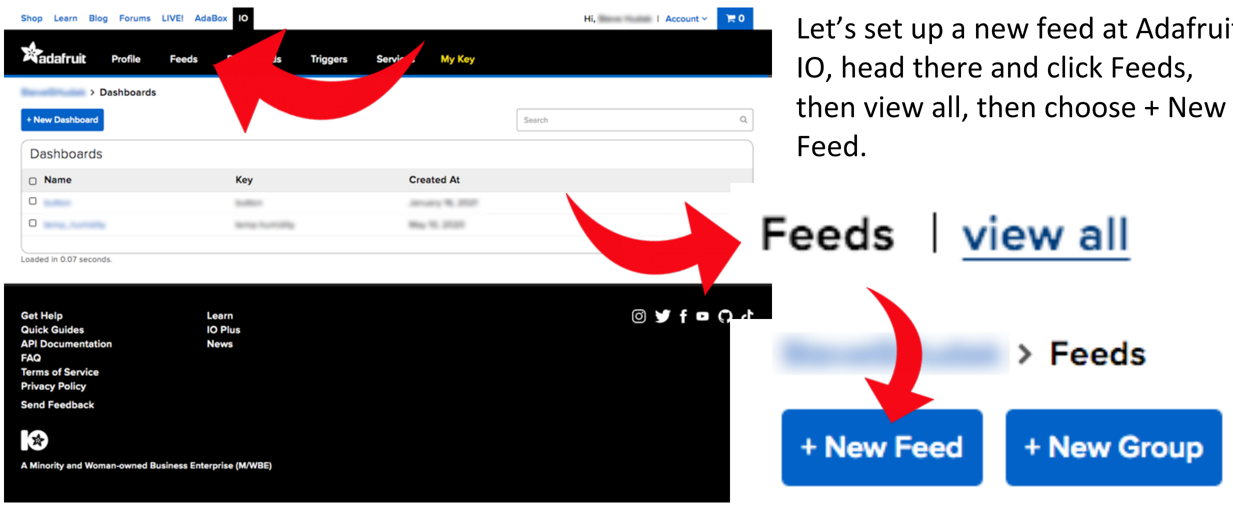
Adafruit 8266 Feather Huzzah Guide 3: web2feather

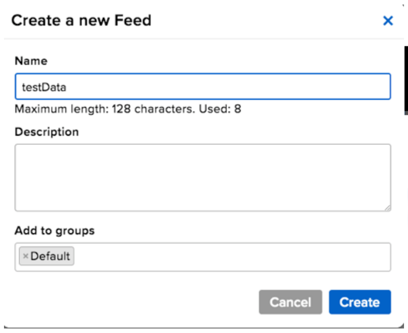
In the last guide we finished with your Feather sending data to Adafruit IO and then sending that live data to a basic P5.js sketch uploaded to your Firebird server. In this guide we will be taking the next logical step and setting up a new server based p5 sketch that can send data to Adafruit IO when a button is pressed and upload a new program to the Feather that will allow it to connect to this data and respond by turning on an LED. While this will look like a simple enough interaction it will open the door to developing an interface and explore more meaningful network-based connections.

Step 1:



Let's set up a new feed at Adafruit IO, head there and click Feeds, then view all, then choose + New Feed.

Create a new feed and name it testData, head back to your Feeds and click on the testData feed, there's nothing there yet so let's get it set up. Click on Privacy and set it to Public.

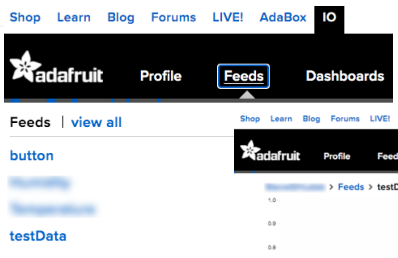

Create a new Feed

Name: testData

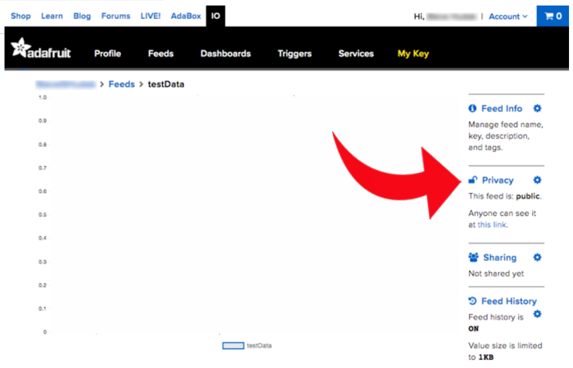
Maximum length: 128 characters. Used: 8

Description:

Add to groups: × Default



Shop Learn Blog Forums LIVE! AdaBox IO
adafruit Profile Feeds Dashboards
Feeds | view all
button
[redacted]
[redacted]
testData



Shop Learn Blog Forums LIVE! AdaBox IO
Hi, [redacted] | Account
0
adafruit Profile Feeds Dashboards Triggers Services My Key
[redacted] > Feeds > testData
testData

Feed Info — Manage feed name, key, description, and tags.

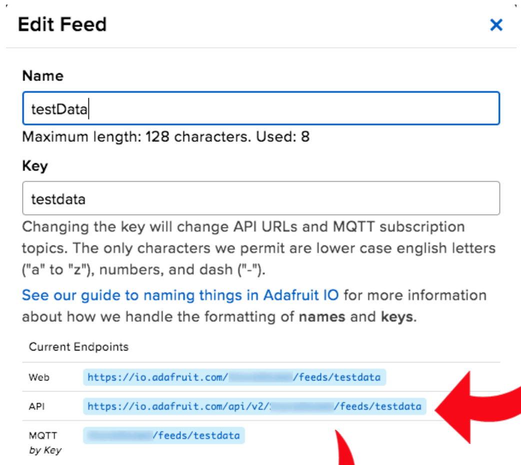
Privacy — This feed is: **public**. Anyone can see it at this link.

Sharing — Not shared yet

Feed History — Feed history is **ON**

Value size is limited to **1KB**

Click on Feed info, you will need the API info so keep this handy.


Edit Feed

Name: testData

Maximum length: 128 characters. Used: 8

Key: testdata

Changing the key will change API URLs and MQTT subscription topics. The only characters we permit are lower case english letters ("a" to "z"), numbers, and dash ("-").

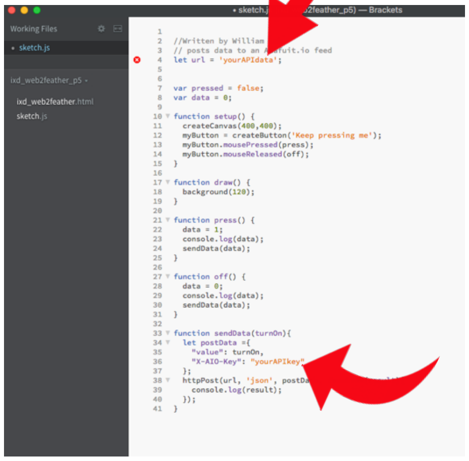
See our guide to naming things in Adafruit IO for more information about how we handle the formatting of **names** and **keys**.

Current Endpoints

Web	https://io.adafruit.com/[redacted]/feeds/testdata
API	https://io.adafruit.com/api/v2/[redacted]/feeds/testdata
MQTT *by Key*	[redacted]/feeds/testdata

Now with our feed ready and with the info we need to send data we can move on to p5.js

Step 2:

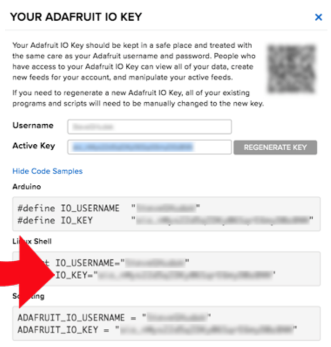


```
1
2   //Written by William
3   // posts data to an Adafruit.io feed
4   let url = 'yourAPIdata';
5
6
7   var pressed = false;
8   var data = 0;
9
10  function setup() {
11    createCanvas(400,400);
12    myButton = createButton('Keep pressing me');
13    myButton.mousePressed(press);
14    myButton.mouseReleased(off);
15  }
16
17  function draw() {
18    background(120);
19  }
20
21  function press() {
22    data = 1;
23    console.log(data);
24    sendData(data);
25  }
26
27  function off() {
28    data = 0;
29    console.log(data);
30    sendData(data);
31  }
32
33  function sendData(turnOn){
34    let postData ={
35      "value": turnOn,
36      "X-AIO-Key": "yourAPIkey"
37    };
38    httpPost(url, 'json', postDa
39      console.log(result);
40    });
41  }
```

Drag the folder 'ixd_web2feather_p5' into Brackets. Replace 'yourAPIdata' on line 4 with the API info from your testData feed – add '/data' at the end of the API so it reads:

```
let url = 'https://io.adafruit.com/api/v2/[redacted]/feeds/testdata/data';
```

Then on line 36 you will need to add the info from your AIO key. Save and upload this to Firebird.


YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username: [redacted]

Active Key: [redacted]

Arduino

```
#define IO_USERNAME  "[redacted]"
#define IO_KEY       "[redacted]"
```

Linux Shell

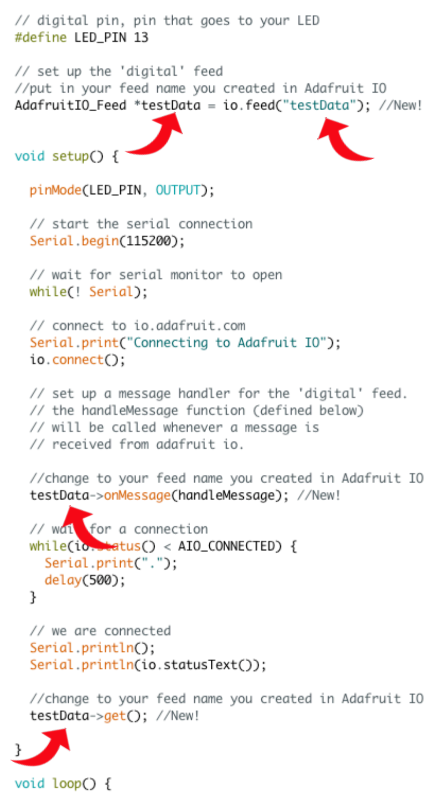
```
t IO_USERNAME="[redacted]"
IO_KEY="[redacted]"
```

Scripting

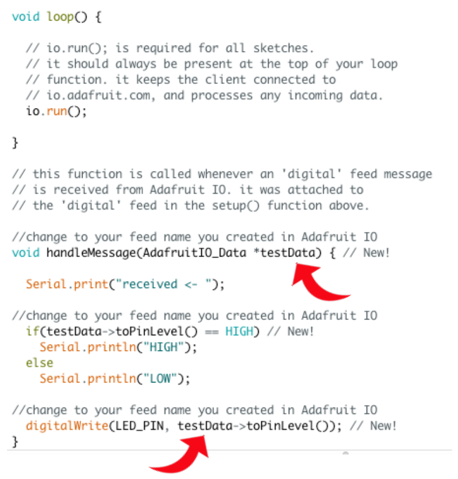
```
ADAFRUIT_IO_USERNAME = "[redacted]"
ADAFRUIT_IO_KEY = "[redacted]"
```

Step 3:

In the ‘ixd_web2feather_arduino’ folder open the ‘ixd_web2feather.ide’ file. The LED pin is set to Pin 13, ‘testData’ is the name assigned to the variable of the data that will be sent to Adafruit IO, it occurs at several points in the code so be aware that if you were to rename one you would need to rename them all for this code to function.



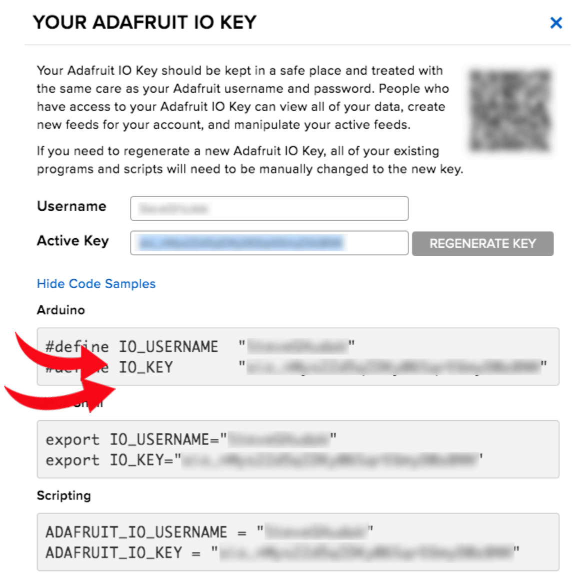
```
// digital pin, pin that goes to your LED
#define LED_PIN 13

// set up the 'digital' feed
//put in your feed name you created in Adafruit IO
AdafruitIO_Feed *testData = io.feed("testData"); //New!


void setup() {

  pinMode(LED_PIN, OUTPUT);

  // 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();

  // set up a message handler for the 'digital' feed.
  // the handleMessage function (defined below)
  // will be called whenever a message is
  // received from adafruit io.

  //change to your feed name you created in Adafruit IO
  testData->onMessage(handleMessage); //New!

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

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

  //change to your feed name you created in Adafruit IO
  testData->get(); //New!

}

void loop() {
```



```
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();

}

// this function is called whenever an 'digital' feed message
// is received from Adafruit IO. it was attached to
// the 'digital' feed in the setup() function above.

//change to your feed name you created in Adafruit IO
void handleMessage(AdafruitIO_Data *testData) { // New!

  Serial.print("received <- ");

//change to your feed name you created in Adafruit IO
  if(testData->toPinLevel() == HIGH) // New!
    Serial.println("HIGH");
  else
    Serial.println("LOW");

//change to your feed name you created in Adafruit IO
  digitalWrite(LED_PIN, testData->toPinLevel()); // New!
}
```

Step 4:

In the config.h file we will make the same changes we made to the config.h file from the previous guide.This file is where the information concerning your Wi-Fi password and Adafruit Key will be kept, do not share this information and always make sure you have blurred out or removed it when sharing files or documenting your process. This file will be loaded onto your Arduino but cannot be accessed by anyone unless you share it.



YOUR ADAFRUIT IO KEY

Your Adafruit IO Key should be kept in a safe place and treated with the same care as your Adafruit username and password. People who have access to your Adafruit IO Key can view all of your data, create new feeds for your account, and manipulate your active feeds.

If you need to regenerate a new Adafruit IO Key, all of your existing programs and scripts will need to be manually changed to the new key.

Username [redacted]

Active Key [redacted]

Arduino

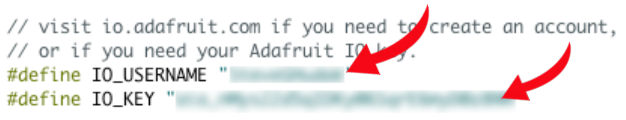
```
#define IO_USERNAME   "[redacted]"
#define IO_KEY        "[redacted]"
```

```
export IO_USERNAME="[redacted]"
export IO_KEY="[redacted]"
```

Scripting

```
ADAFRUIT_IO_USERNAME = "[redacted]"
ADAFRUIT_IO_KEY = "[redacted]"
```

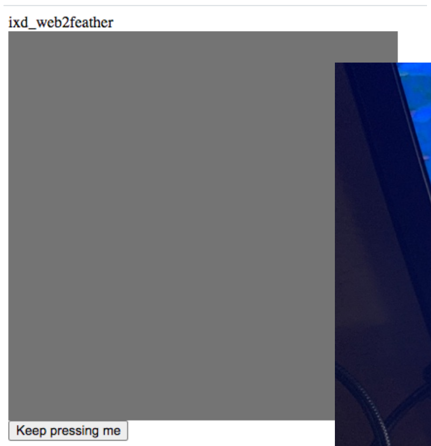
First right at the top of the code there is a location to add your Adafruit IO Key

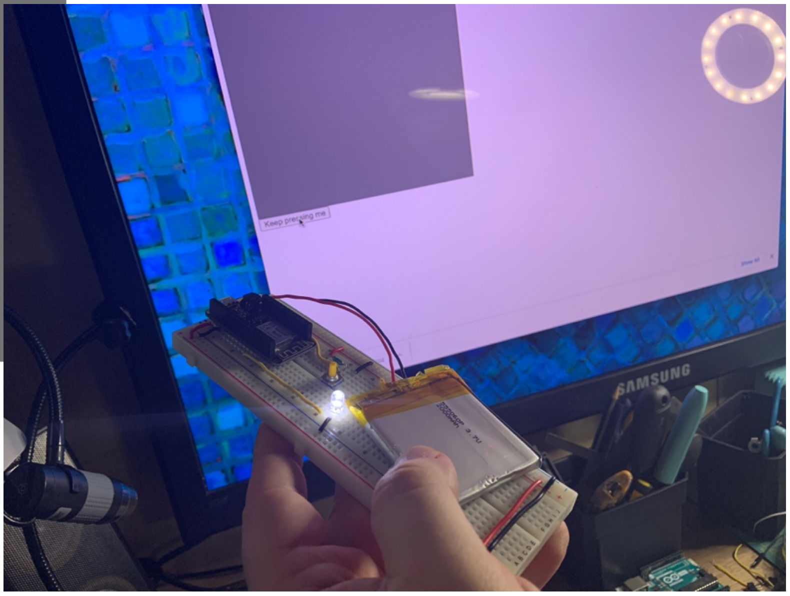


```
// visit io.adafruit.com if you need to create an account,
// or if you need your Adafruit IO key.
#define IO_USERNAME "[redacted]"
#define IO_KEY "[redacted]"
```

Using the specific information from your Key fill in the areas named “yourUsername” and “yourKey”.

Step 5: Upload these files to your Feather and give it a minute for the board to connect to your Wi-Fi. Head over to your web2feather Firebird page and click the button element.





Running on a 3.7v LiPo cellphone battery the Feather now connected to the local network, the button in the p5 sketch is sending data to Adafruit IO feed and the Feather is receiving data from and the LED is switched on.

This concludes guide 3, if you were successful at each step by this point you are well equipped to begin experimenting and following where your curiosity leads you.

The purpose of making this guide has been to create an easy to follow step by step process so IO&E students can begin developing networked Feather projects, this guide was only made possible using code developed by William Luk and Michael Brzuchalski. Their tutorial along with official Adafruit guides and images, individual blog posts, and Youtube videos have been compiled and parsed in making this guide.

Links to these sources are included below.

https://www.youtube.com/watch?v=GXECML4S4XI

https://www.youtube.com/watch?v=9mp-moZnAlA&t=0s

https://learn.adafruit.com/adafruit-io-basics-esp8266-arduino/overview

https://learn.adafruit.com/adafruit-io-basics-digital-input/overview