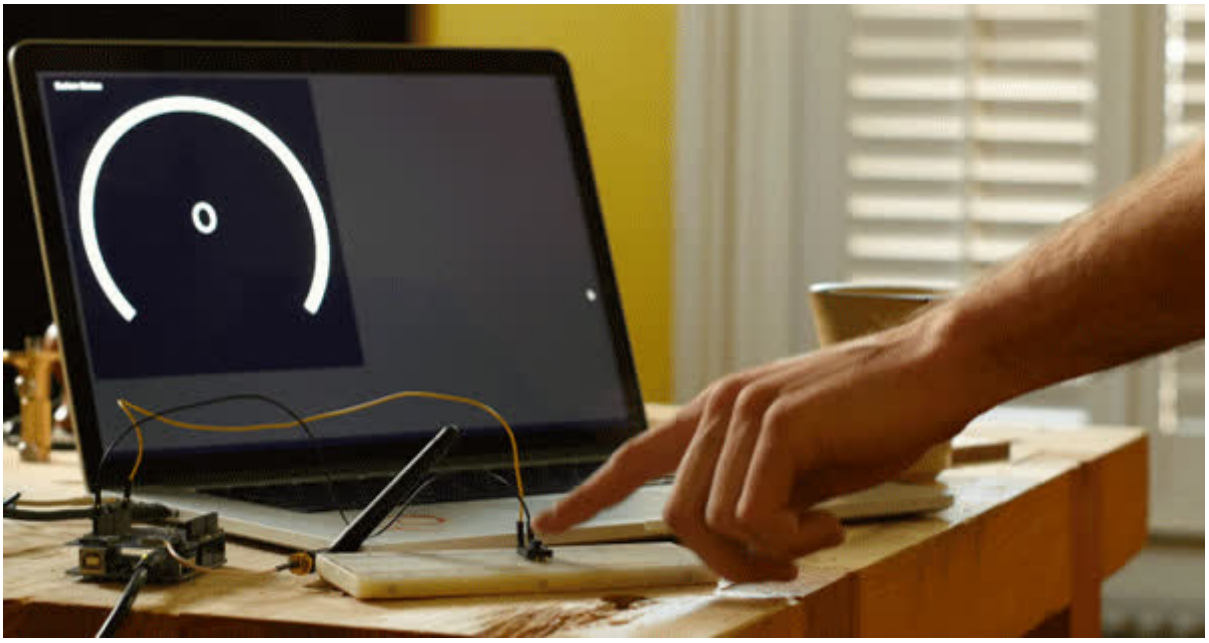
Adafruit IO Basics: Digital Input

Created by Justin Cooper

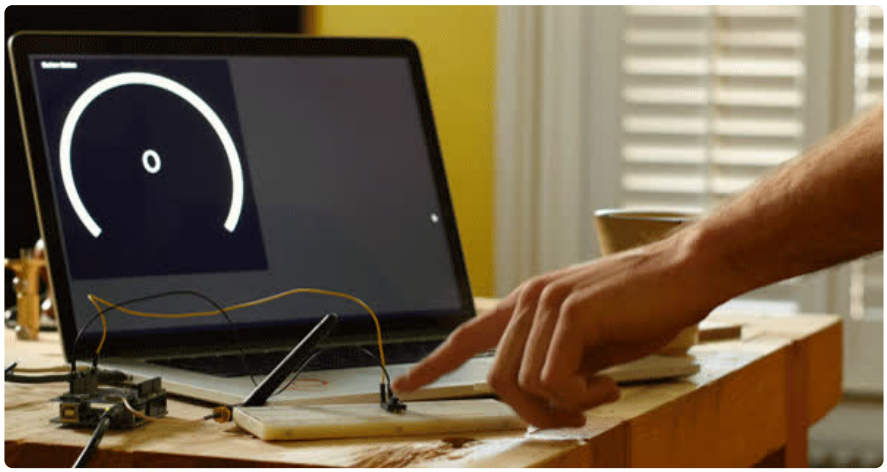


https://learn.adafruit.com/adafruit-io-basics-digital-input

Last updated on 2024-06-03 01:42:33 PM EDT

Table of Contents

Overview



Want to send data from your electronics project to the Internet and display it on a webpage?

This guide will teach you to send data from a digital input, such as a momentary push-button, to Adafruit IO. Using Adafruit IO, you will build a visual Dashboard with a Gauge block to display (in real-time!) when the push button has been pressed or depressed.

Adafruit IO allows you to connect and interact with this project either by:

1. Programming a microcontroller, microcomputer, or desktop computer (configurable, and powerful, but requires coding skills)

   or

2. Using our no-code "WipperSnapper" firmware (super fast! no coding! but only for some microcontroller boards)

Option 1. Build and Program this project with Arduino or CPython

This guide contains instructions and code for building this project using Arduino, the Adafruit IO Arduino client library, and an Arduino-compatible board. We also include instructions for building this project using a Raspberry Pi and code for CPython (a.k.a desktop-computer Python).

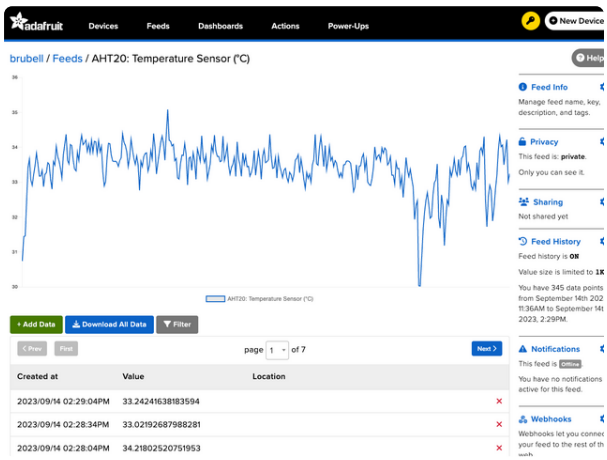
Option 2. No Code? No Problem! Build this guide using Adafruit IO and WipperSnapper!

With Adafruit IO, you can connect your devices without writing a single line of code using our custom WipperSnapper firmware. Load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

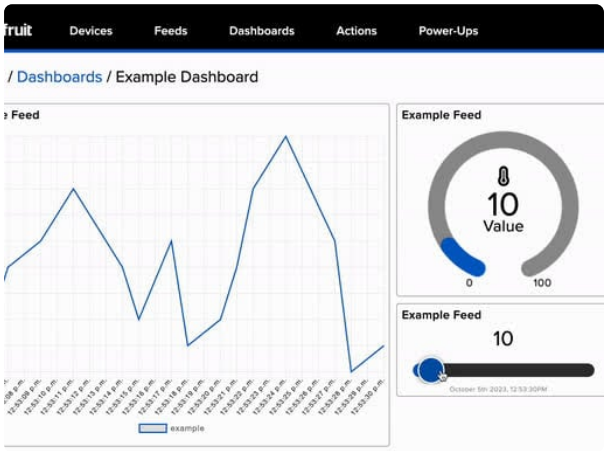
Pre-Requisite Reading: Adafruit IO Basics

This guide is part of a series of guides that cover the basics of using Adafruit IO. If you haven't read 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 the core functionality of Adafruit IO.



[Adafruit IO Basics: Feeds](https://learn.adafruit.com/adafruit-io-basics-feeds)
By Todd Treece

https://learn.adafruit.com/adafruit-io-basics-feeds



[Adafruit IO Basics: Dashboards](https://learn.adafruit.com/adafruit-io-basics-dashboards)
By Todd Treece

https://learn.adafruit.com/adafruit-io-basics-dashboards

Get Started with Adafruit IO

Adafruit IO is integrated with your adafruit.com account (https://adafru.it/dyy) so you don't need to create yet another online account! You need an Adafruit account to use Adafruit IO because we want to make sure the data you upload is available to only you (unless you decide to publish your data).

I have an Adafruit.com Account already

If you already have an Adafruit account, then you already have access to Adafruit IO. It doesn't matter how you signed up, your account will make all three available.

To access Adafruit IO, simply visit https://io.adafruit.com (https://adafru.it/eZ8) to start streaming, logging, and interacting with your data.

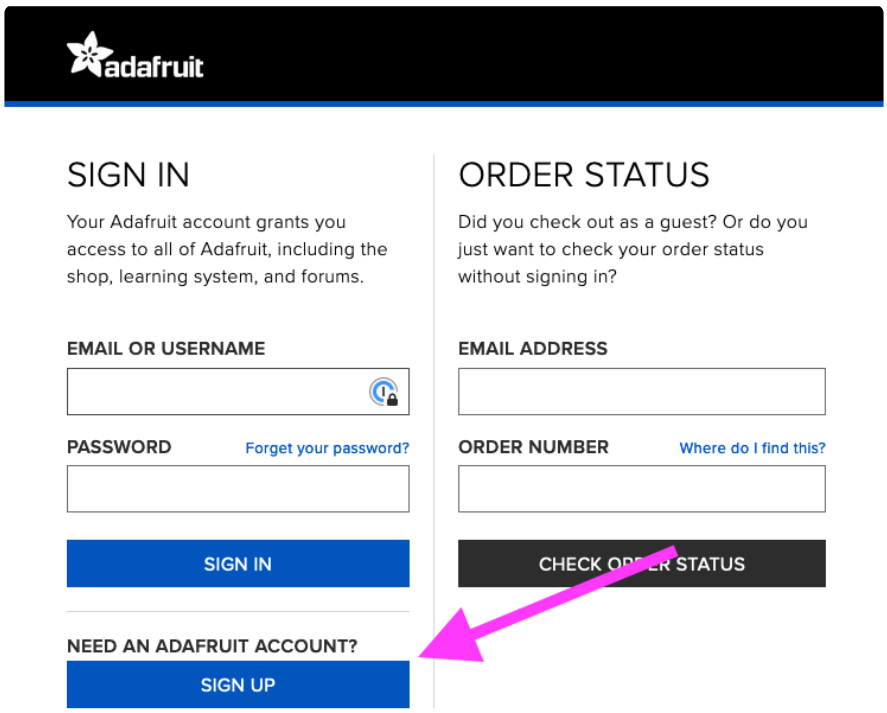
Create an Adafruit Account (for Adafruit IO)

An Adafruit account makes Adafruit content and services available to you in one place. Your account provides access to the Adafruit shop (https://adafru.it/dAR), the Adafruit Learning System (https://adafru.it/dlu), and Adafruit IO (https://adafru.it/fsU). This means only one account, one username, and one password are necessary to engage with the content and services that Adafruit offers.

If you do not have an Adafruit account, signing up for a new Adafruit account only takes a couple of steps.

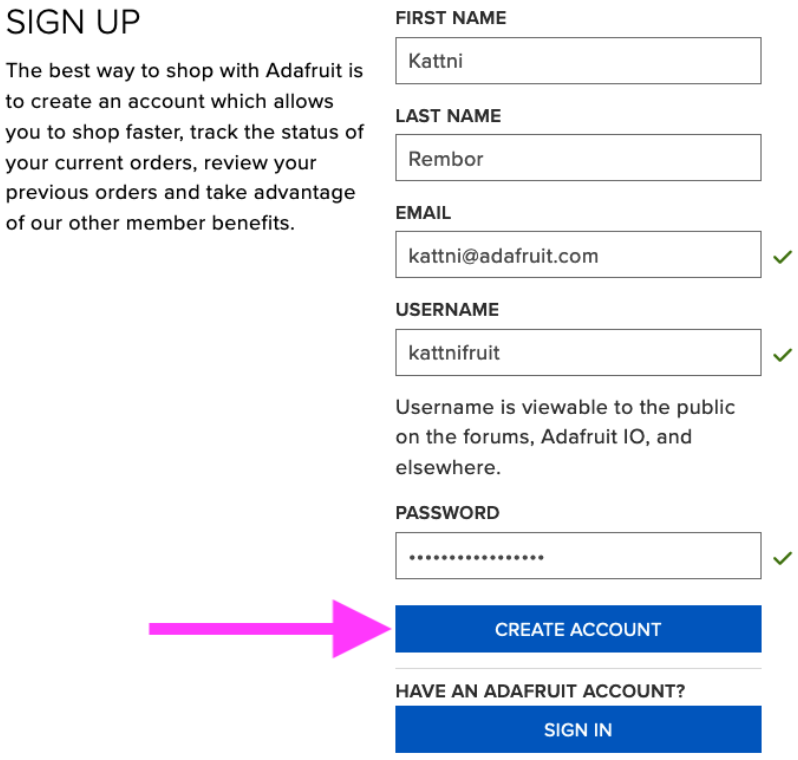
Begin by visiting https://accounts.adafruit.com (https://adafru.it/18ta).

Click the Sign Up button under the "Need An Adafruit Account?" title, below the Sign In section.



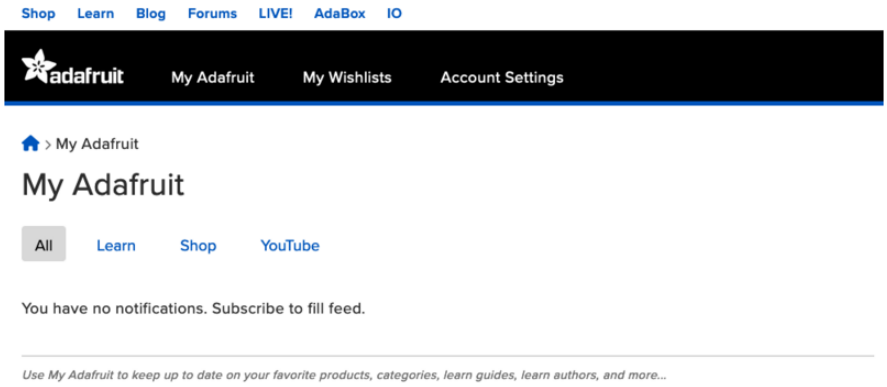
This will take you to the **Sign Up** page.

Fill in the requested information, and **click the Create Account button.**



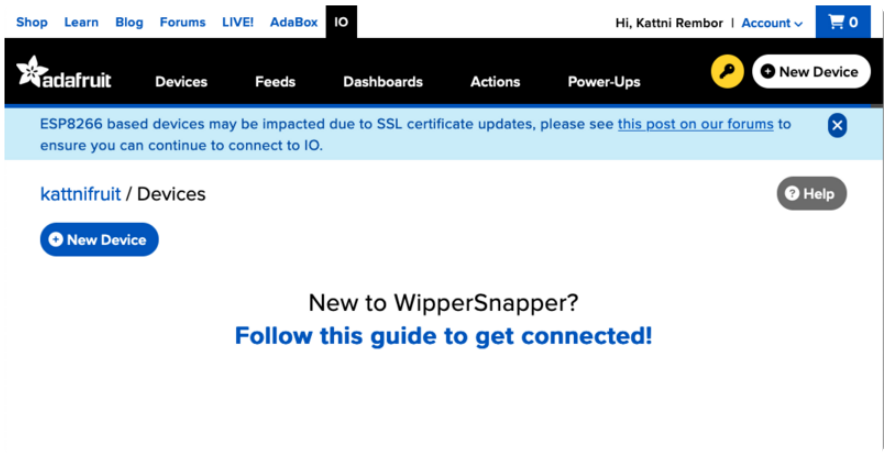
This takes you to your Adafruit Account home page. From here, you can access all the features of your account.

You can also access the Adafruit content and services right from this page. Along the top of the page, you'll see a series of links beginning with "Shop". To access any of these, simply click the link.



For example, **to begin working with Adafruit IO, click the IO link** to the right of the rest of the links. This is the same for the other links as well.

That's all there is to creating a new Adafruit account, and navigating to Adafruit IO.



Arduino Wiring

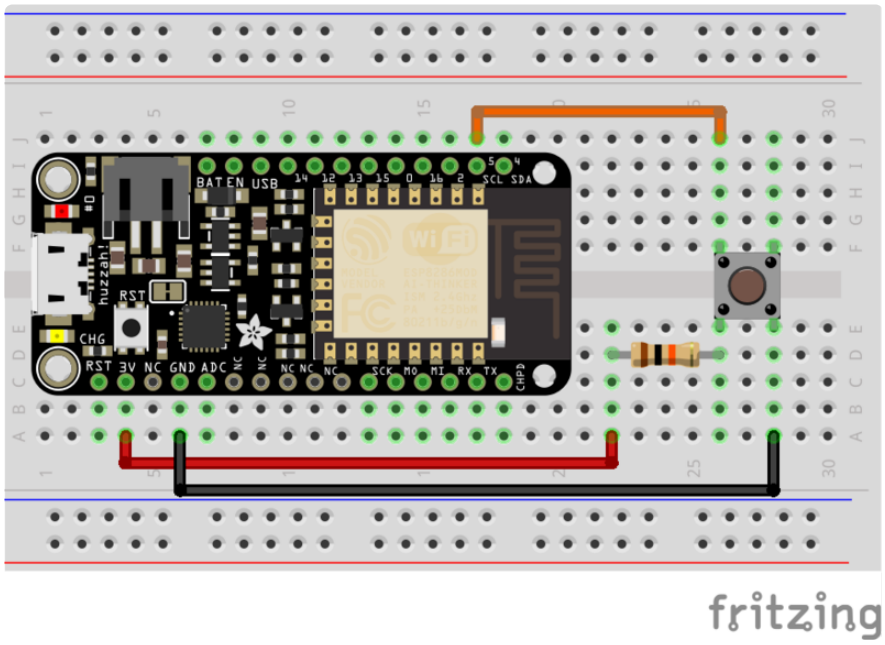
You will need the following parts for this tutorial:

- **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.

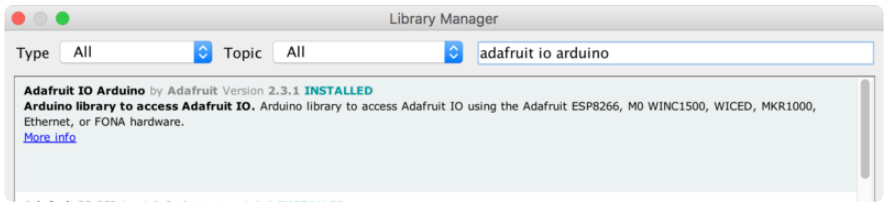


Arduino Setup

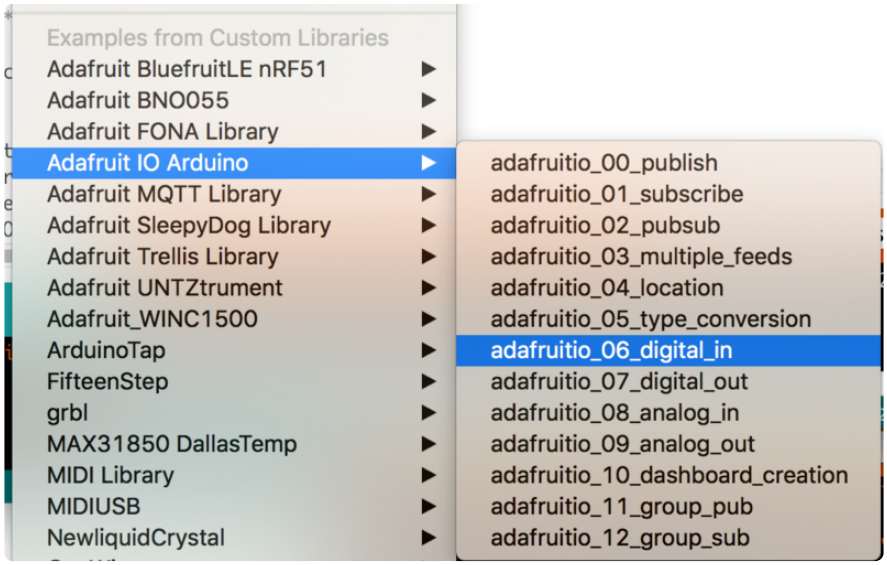
You should go through the setup guides associated with your selected set of hardware, and make sure you have internet connectivity with the device before continuing. The following links will take you to the guides for your selected platform.

- 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!



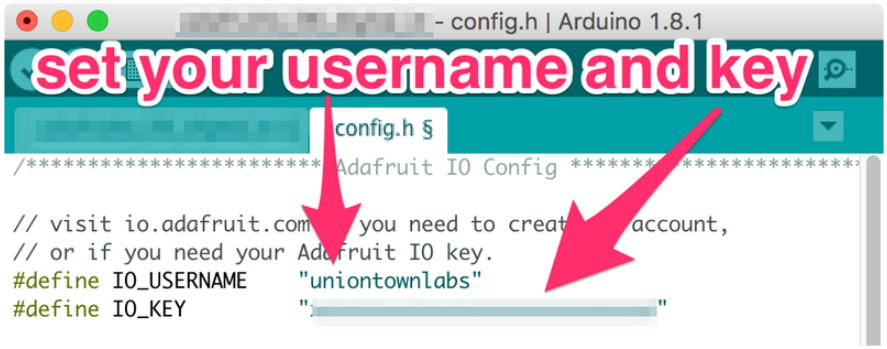
For this example, you will need to open the **adafruitio_06_digital_in** example in the **Adafruit IO Arduino** library.



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

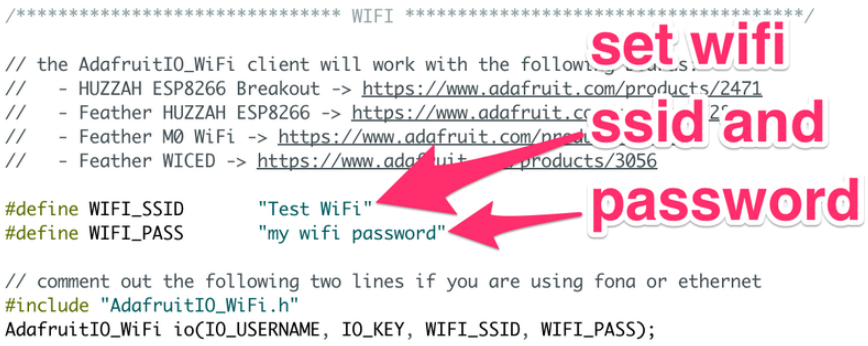
Arduino Network Config

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.



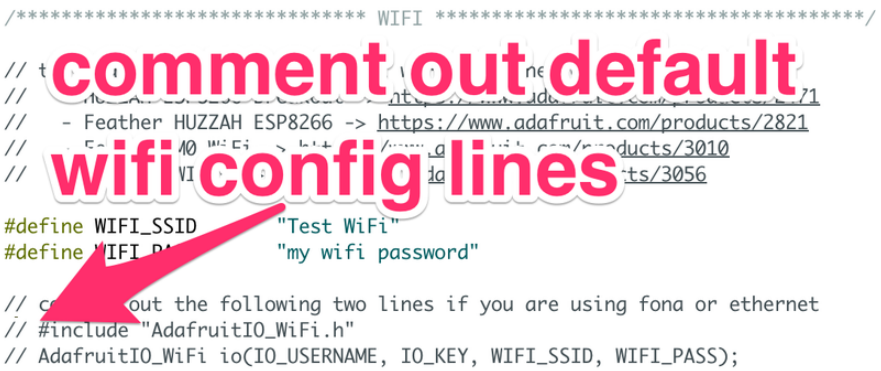
WiFi Config

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.



FONA Config

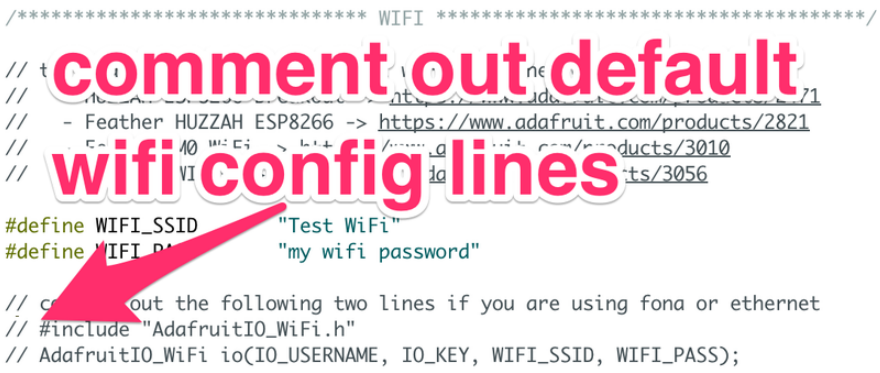
If you wish to use the FONA 32u4 Feather to connect to Adafruit IO, you will need to first comment out the WiFi support in **config.h**



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.

Arduino 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() &lt; 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 -&gt; ");
  Serial.println(current);
  digital-&gt;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.

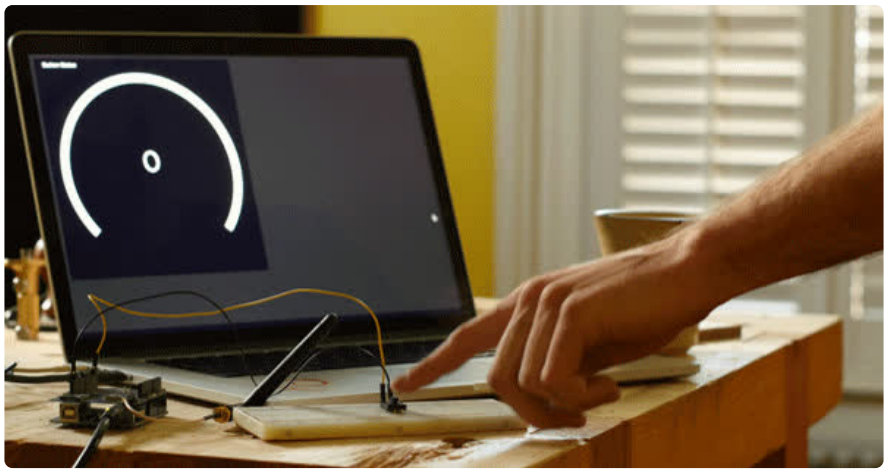
```
Connecting to Adafruit IO....

Adafruit IO connected.
```

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

```
sending button -&gt; 1
sending button -&gt; 0
sending button -&gt; 1
sending button -&gt; 0
```

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



WipperSnapper Setup

The WipperSnapper firmware and ecosystem are in BETA and are actively being developed to add functionality, more boards, more sensors, and fix bugs. We encourage you to try out WipperSnapper with the understanding that it is not final release software and is still in development.

If you encounter any bugs, glitches, or difficulties during the beta period, or with this guide, please contact us via http://io.adafruit.com/support

What is WipperSnapper

WipperSnapper is a firmware designed to turn any WiFi-capable board into an Internet-of-Things device without programming a single line of code. WipperSnapper connects to Adafruit IO (https://adafru.it/fsU), a web platform designed (by Adafruit! (https://adafru.it/Bo5)) to display, respond, and interact with your project's data.

Simply load the WipperSnapper firmware onto your board, add credentials, and plug it into power. Your board will automatically register itself with your Adafruit IO account.

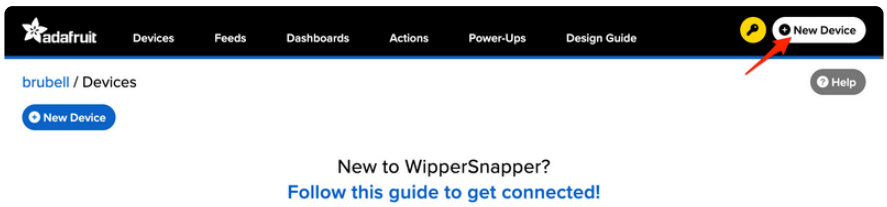
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

Sign up for Adafruit.io

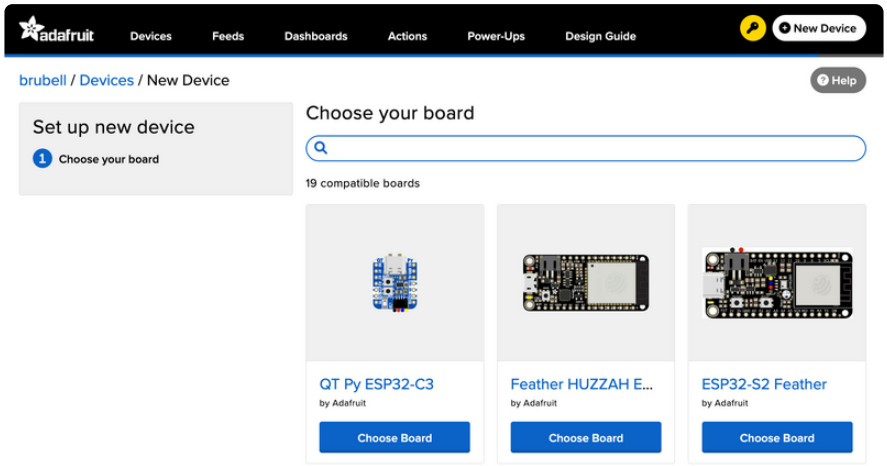
You will need an Adafruit IO account to use WipperSnapper on your board. If you do not already have one, head over to io.adafruit.com (https://adafru.it/fsU) to create a free account.

Install WipperSnapper

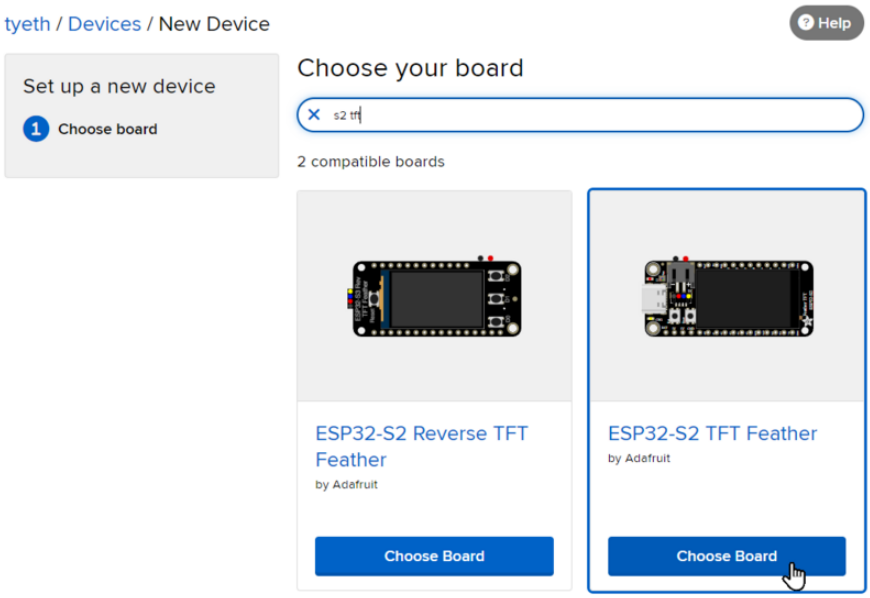
Log into your Adafruit IO (https://adafru.it/fsU) account. Click the New Device button at the top of the page.



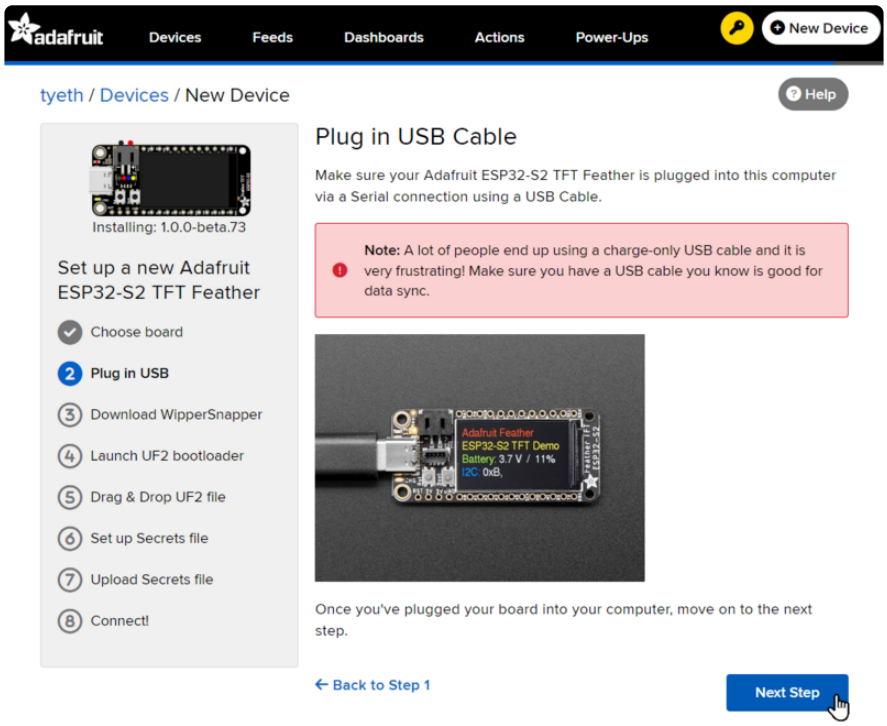
After clicking New Device, you should be on the board selector page. This page displays every board that is compatible with the WipperSnapper firmware.



In the board selector page's search bar, search for the ESP32-S2 TFT Feather (or your board of choice). Once you've located the board you'd like to install WipperSnapper on, click the Choose Board button to bring you to the self-guided installation wizard.



Follow the step-by-step instructions on the page to install Wippersnapper on your device and connect it to Adafruit IO. You may need to revisit this New Device page if you're Wireless Network details change in the future.

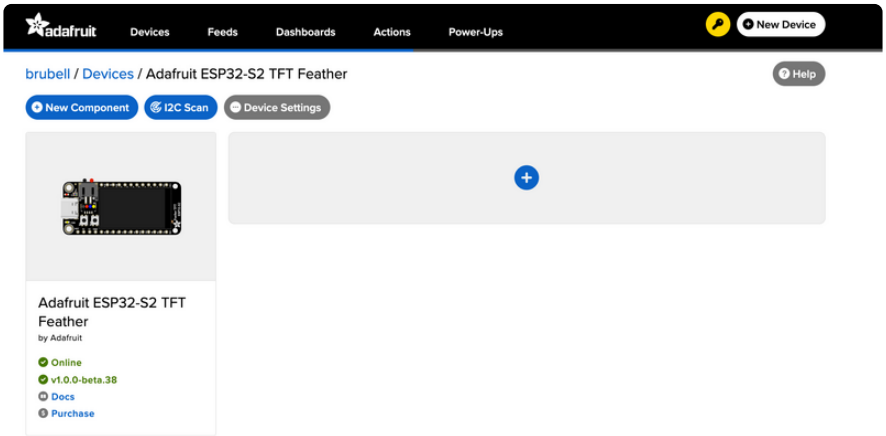


If the installation was successful, a popover should appear displaying that your board has successfully been detected by Adafruit IO.

Give your board a name and click "Continue to Device Page".

You should be brought to your board's device page.

Next, Visit this guide's **WipperSnapper Essentials** pages to learn how to interact with your board using Adafruit IO.



Feedback

Adafruit.io WipperSnapper is in **beta** and you can help improve it!

If you have suggestions or general feedback about the installation process - visit https://io.adafruit.com/support (https://adafru.it/Sgb), click "Contact Adafruit IO Support" and select "I have feedback or suggestions for the WipperSnapper Beta".

Troubleshooting

If you encountered an issue during installation, please try the steps below first.

If you're still unable to resolve the issue, or if your issue is not listed below, get in touch with us directly at https://io.adafruit.com/support (https://adafru.it/Sgb). Make sure to click "Contact Adafruit IO Support" and select "There is an issue with WipperSnapper. Something is broken!"

My computer doesn't see my device so I can't install WipperSnapper

A lot of people end up using a charge-only USB cable and it is very frustrating! Make sure you have a USB cable you know is good for data sync. USB Hubs can also interfere so make sure the device is directly connected to your computer.

If you're computer isn't up to date with the latest driver updates, then you may not have the software driver necessary for the USB Serial controller that speaks to the Microcontroller. Find the latest drivers on your boards learn guide page, or here (https://adafru.it/10aP) for the ESP32-S2 TFT Feather.

If you've already installed the firmware, but can't set your secrets then try resetting the board and it should show up (if it supports USB Mass Storage), otherwise it might be easiest to go back to the new device page and set up the board's software again by following the steps for your board.

I don't see my board on Adafruit IO, it is stuck connecting to WiFi

First, make sure that you selected the correct board on the board selector.

Next, please make sure that you entered your WiFi credentials properly, there are no spaces/special characters in either your network name (SSID) or password, and that you are connected to a 2.4GHz wireless network.

If you're still unable to connect your board to WiFi, please make a new post on the WipperSnapper technical support forum with the error you're experiencing, the LED colors which are blinking, and the board you're using. (https://adafru.it/V6a)

I don't see my board on Adafruit IO, it is stuck "Registering with Adafruit IO"

Try hard-resetting your board by unplugging it from USB power and plugging it back in.

If the error is still occurring, please make a new post on the WipperSnapper technical support forum with information about what you're experiencing, the LED colors which are blinking (if applicable), and the board you're using. (https://adafru.it/V6a)

"Uninstalling" WipperSnapper

WipperSnapper firmware is an application that is loaded onto your board. There is nothing to "uninstall". However, you may want to "move" your board from running WipperSnapper to running Arduino or CircuitPython. You also may need to restore your board to the state it was shipped to you from the Adafruit factory.

Moving from WipperSnapper to CircuitPython

Follow the steps on the Installing CircuitPython page (https://adafru.it/Amd) to install CircuitPython on your board running WipperSnapper.

- If you are unable to double-tap the RST button to enter the UF2 bootloader, follow the "Factory Resetting a WipperSnapper Board" instructions below.

Uploading this sketch will overwrite WipperSnapper. If you want to re-install WipperSnapper, follow the instructions at the top of this page.

Moving from WipperSnapper to Arduino

If you want to use your board with Arduino, you will use the Arduino IDE to load any sketch onto your board.

First, follow the page below to set up your Arduino IDE environment for use with your board.

Setup Arduino IDE

https://adafru.it/10aP

Then, follow the page below to upload the "Arduino Blink" sketch to your board.

Upload Arduino Blink Sketch

https://adafru.it/19a7

Uploading this sketch will overwrite WipperSnapper. If you want to re-install WipperSnapper, follow the instructions at the top of this page.

Factory Resetting a WipperSnapper Board

Sometimes, hardware gets into a state that requires it to be "restored" to the original state it shipped in. If you'd like to get your board back to its original factory state, follow the guide below.

Factory Reset Guide

https://adafru.it/XTE

Wiring

Before you get this wired up, you'll need a few components!

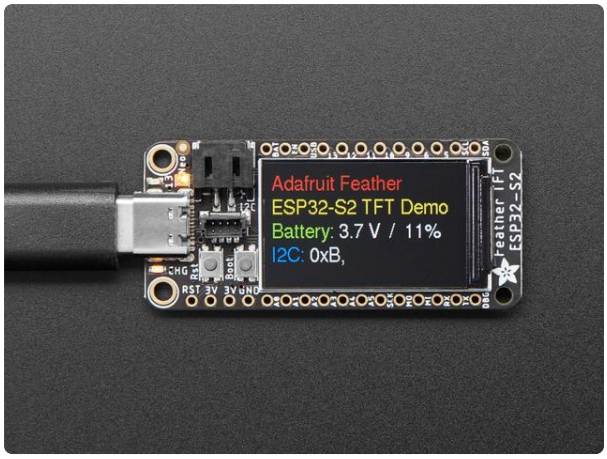
Mainly you'll want a Push Button (almost any size button should be fine), a resistor (to pull the signal wire's voltage up to a high level - 3volts), and a few wires (or Jumper Cables).

Have a look at the shopping list below, then check your existing components for any matching resistors / buttons, or pickup a few from the store.

Parts

For this example, we are using an Adafruit ESP32-S2 TFT Feather.

This board is **perfect** for IoT projects as it includes all the features of a Feather main board: the comforting warmth of an ESP32-S2 WiFi microcontroller, and the crispness of a 240x135 pixel color TFT display. This feather comes with native USB and **4 MB flash** + **2 MB of PSRAM**, so it is perfect for use with CircuitPython, WipperSnapper, or Arduino with low-cost WiFi. Native USB means it can act like a keyboard or a disk drive. WiFi means it's awesome for IoT projects. And Feather means it works with the large community of Feather Wings for expandability.



Adafruit ESP32-S2 TFT Feather - 4MB Flash, 2MB PSRAM, STEMMA QT
We've got a new machine here at Adafruit, it can uncover your deepest desires. Don't believe me? I'll turn it on right now to prove it to you! What, you want unlimited...
https://www.adafruit.com/product/5300

However, you can use any WipperSnapper-compatible hardware with this tutorial. You will need to modify the pin location as a result since it won't match the Adafruit ESP32-S2 TFT Feather.

You will also need the following additional parts:

1 x 10K Ohm Resistor — https://www.adafruit.com/product/2784
25 Pack of 10K Ω Resistors (You only need one for this project!)

---

1 x Breadboard — https://www.adafruit.com/product/4539
Half-Size Breadboard with Mounting Holes

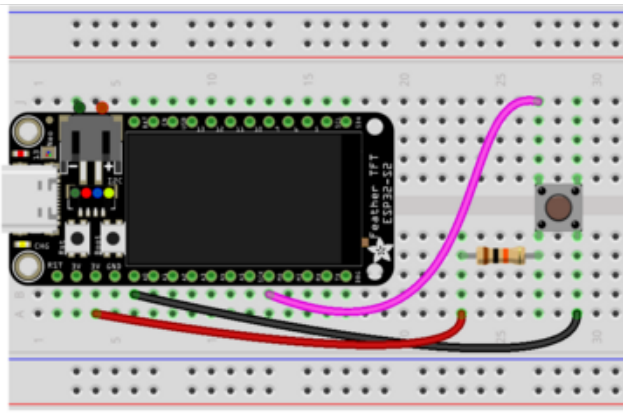
---

1 x Jumper Wires — https://www.adafruit.com/product/153
Breadboarding wire bundle

---

1 x Momentary Pushbutton — https://www.adafruit.com/product/1119
Tactile Switch Buttons (12mm square, 6mm tall) x 10 pack

Wiring



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 GPIO5 (D5)** 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 **D5** (left side of button in this image)
Note: Resistors are not polarized, so the 10k resistor can be connected to the circuit in either direction.

Configuring WipperSnapper

This page assumes you've setup your device with the WipperSnapper firmware and your Wireless Network details, so you should already be able to get your board online! If not, then head over to the first WipperSnapper page in this guide.

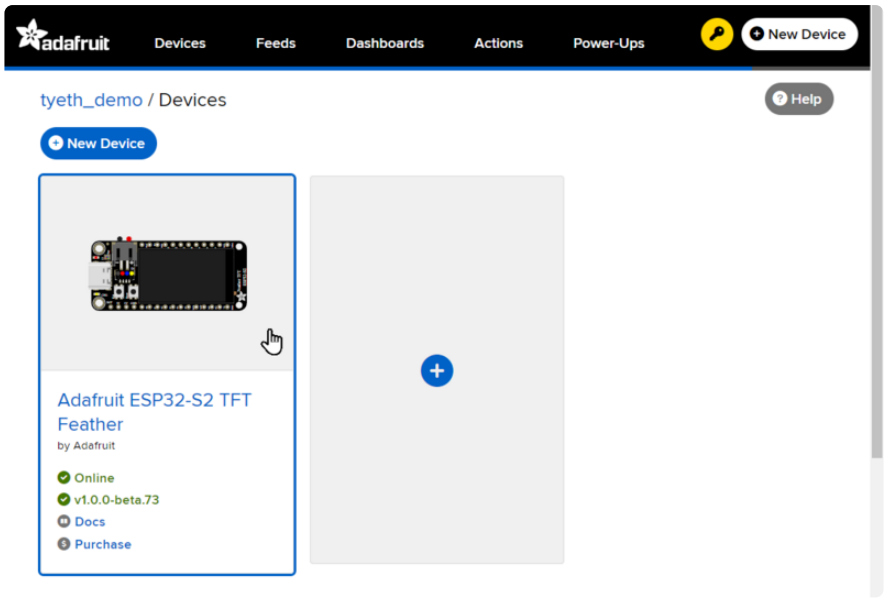
In the real (physical) world we have electrical components, so to represent that on our WipperSnapper device page in Adafruit IO we also have components! They are like digital twins with the real world electrical components, but before you can use them you have to tell your device which ones to use and how!

So now that you have the physical hardware setup, including the electrical components and wiring, it's time to setup your Button component on the WipperSnapper Device page...

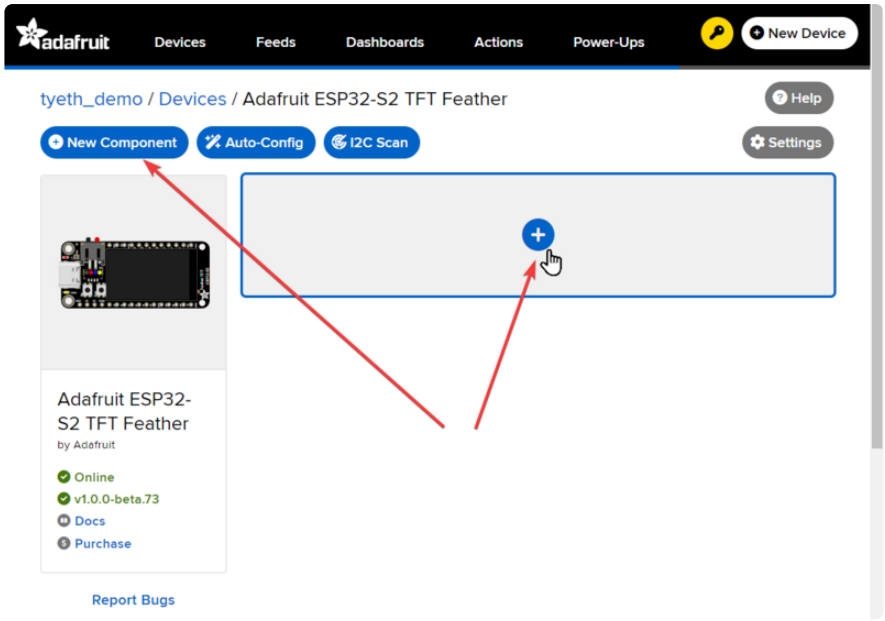
Add a Component to Your Device

Connect your board to Adafruit IO Wippersnapper and navigate to the WipperSnapper board list **(https://adafru.it/TAu)**.

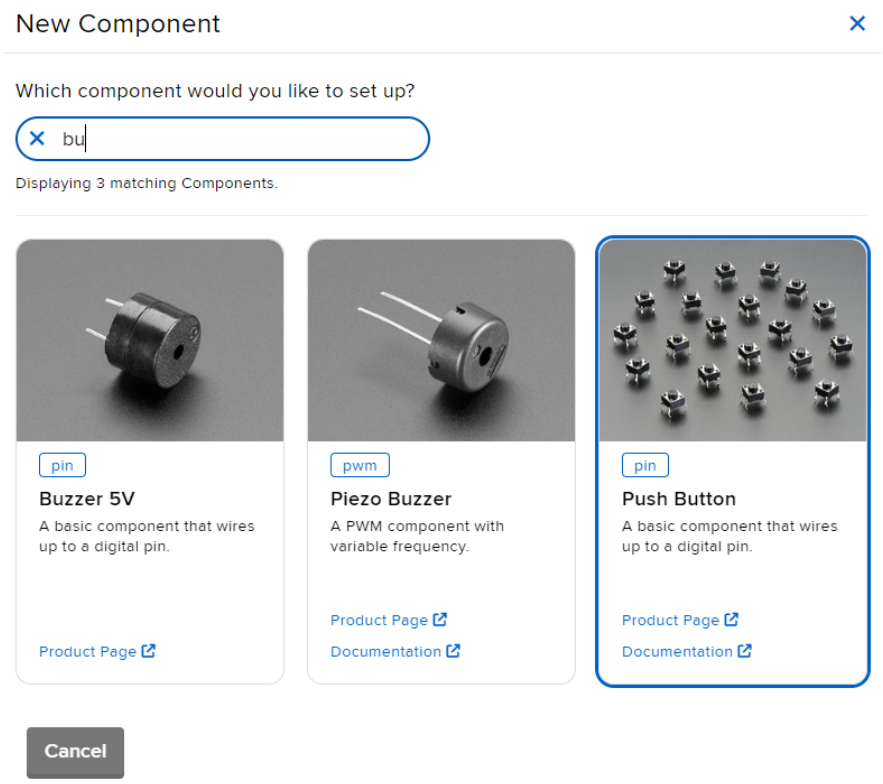
On this page, **select the WipperSnapper board you're using** to be brought to the board's interface page.



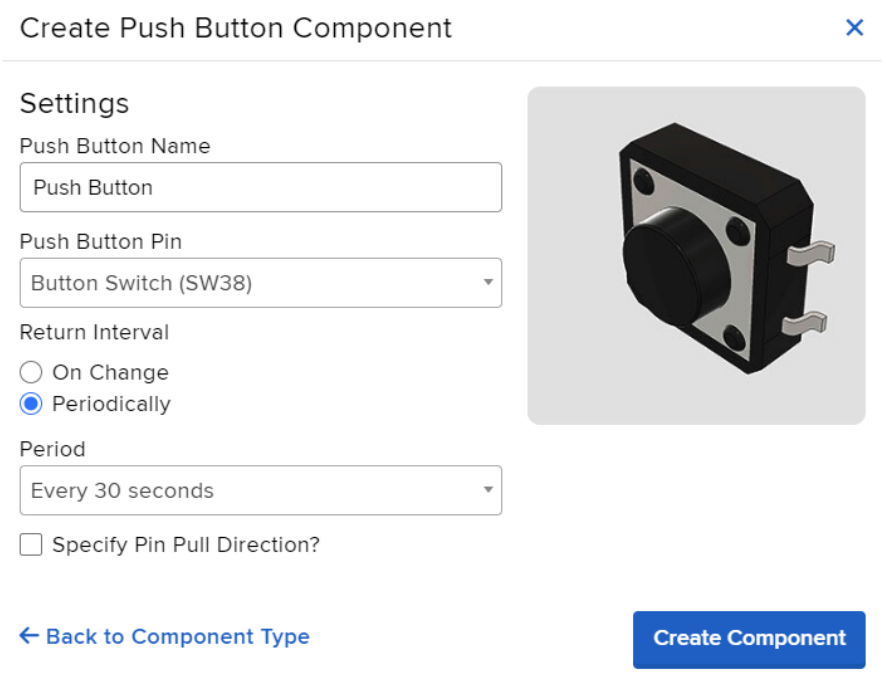
Use the New Component button to bring up the Component Picker window.



Select the **Push Button** component from the New Component window.



The **Create Push Button Component** window will be shown.



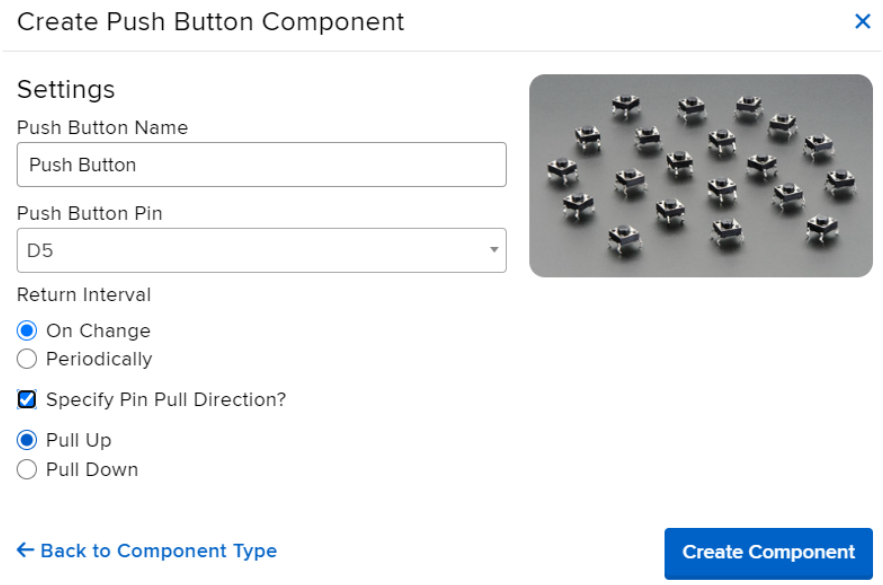
Change the **Push Button Pin** to match your wiring, if you followed the Wiring guidance above then the pin is **D5 (SCK)**.

It's always worth double-checking the datasheet / pinout-diagram for pin numbering, in this example the GPIO pins are prefixed with D so GPIO5 is D5

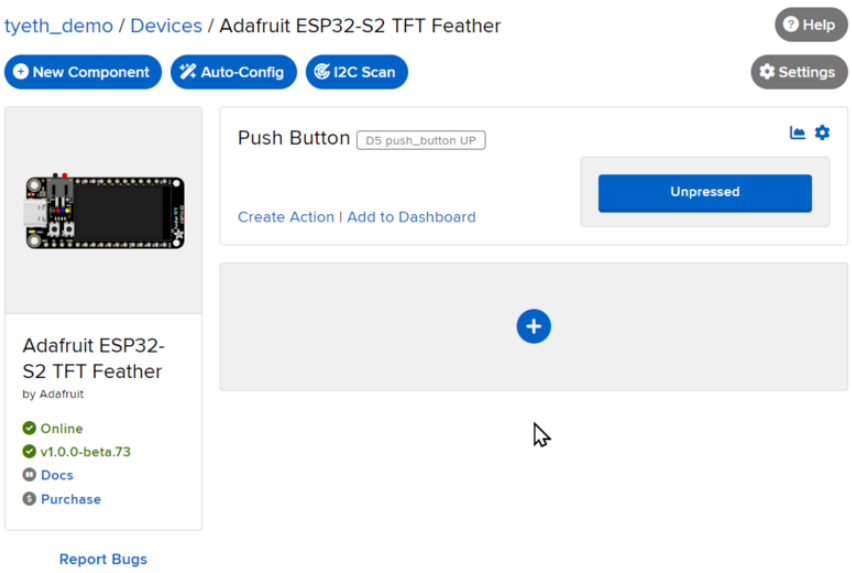
Adjust the **Return Interval** to be **On Change** instead of Periodically (periodically measures at regular intervals, whereas on-change records a measurement every time the pin value changes)

Enable the option to **Specify Pin Pull Direction**, and select the **Pull Up** radio button.

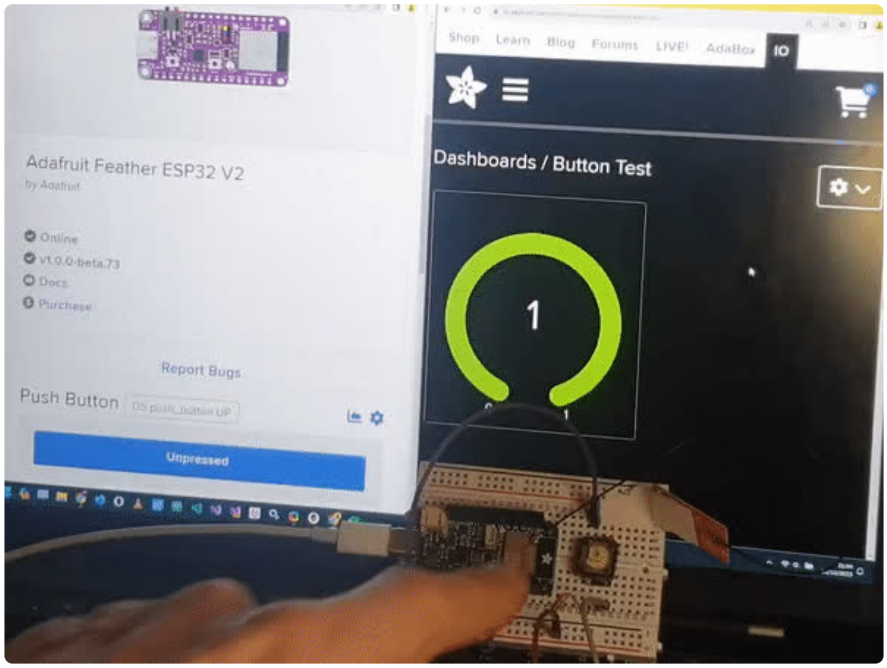
This is to determine if a logic high (1) or logic low (0) should indicate the pressing of the button. This is determined in our circuit by the resistor pulling the button signal line up to 3 volts when not pushed (logic high). When the button is pushed, the internal switch contacts connect, which joins the other side of the switch to the signal wire (ground becomes connected which takes the signal to zero volts a.k.a. low logic level).



You should now be presented with the Push Button component on your device page.



Test the circuit by holding the button down until you see the value on the device page change, then release and watch the state return to Unpressed.



It's worth noting that behind the scenes WipperSnapper + Adafruit IO creates a new data feed (https://adafru.it/Cf-) for each component. A feed (https://adafru.it/Cf-) is where new data values are published or subscribed to, you can see all your feeds on your Feeds page (https://adafru.it/mxC).

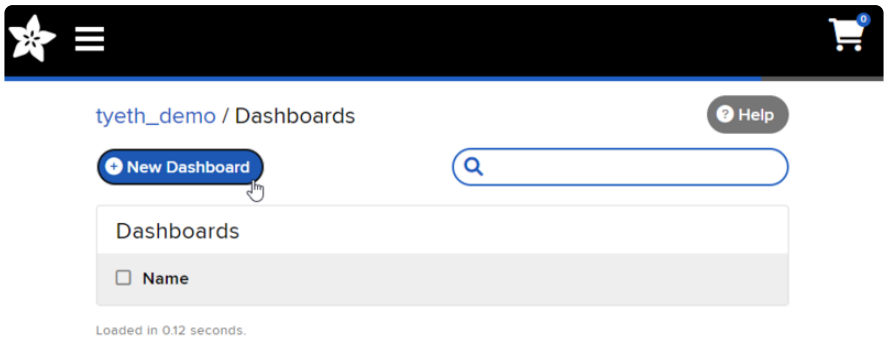
Great, now that's working as expected, it's on to creating the Adafruit IO Dashboard. Head on over to the Dashboards (https://adafru.it/eIS) page.

Usage

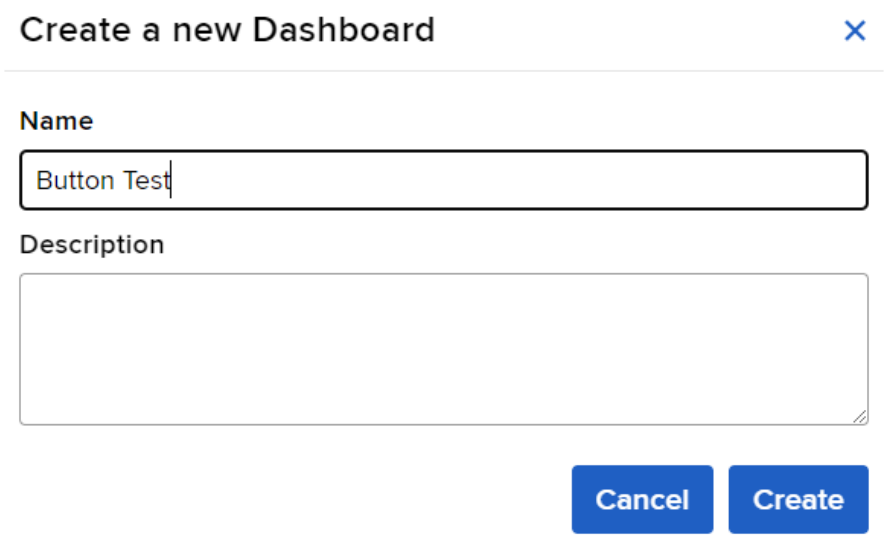
Here you'll create then test an Adafruit IO Dashboard, to allow interactive visualisation and control of your WipperSnapper components (or any other Adafruit IO feed). To get started head on over to the Dashboards (https://adafru.it/eIS) page.

Build Adafruit IO Dashboard

Click the **New Dashboard** button on the Dashboards page to create a new dashboard.

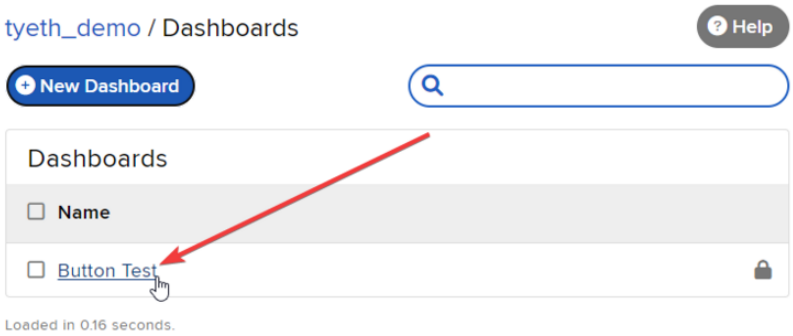


In the **Create a new Dashboard** dialog give the new dashboard an appropriate name, plus a description if you wish, then click the **Create** button.

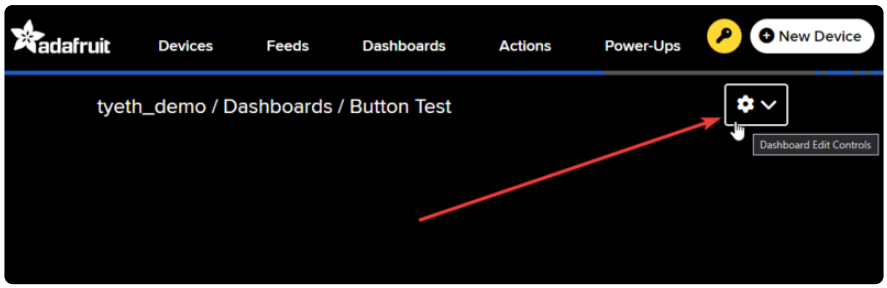


You will be returned to the list of dashboards, with a new entry for the one you created.

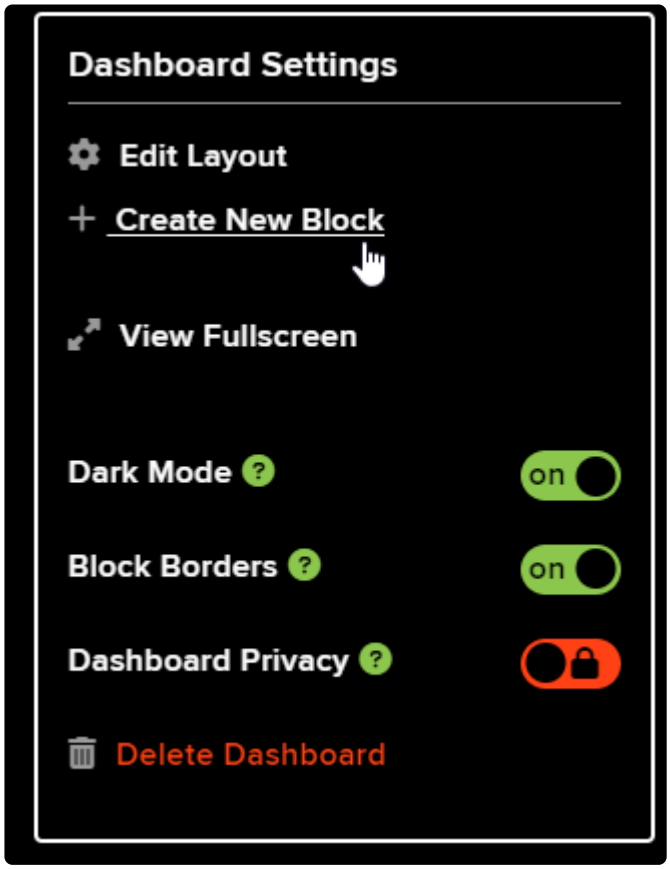
Click the dashboard name in the list to be taken to the new dashboard.



You'll be presented with an empty dashboard, with a drop-down menu to the right-hand side for Dashboard Edit Controls, which is where the **Create New Block** option is located.

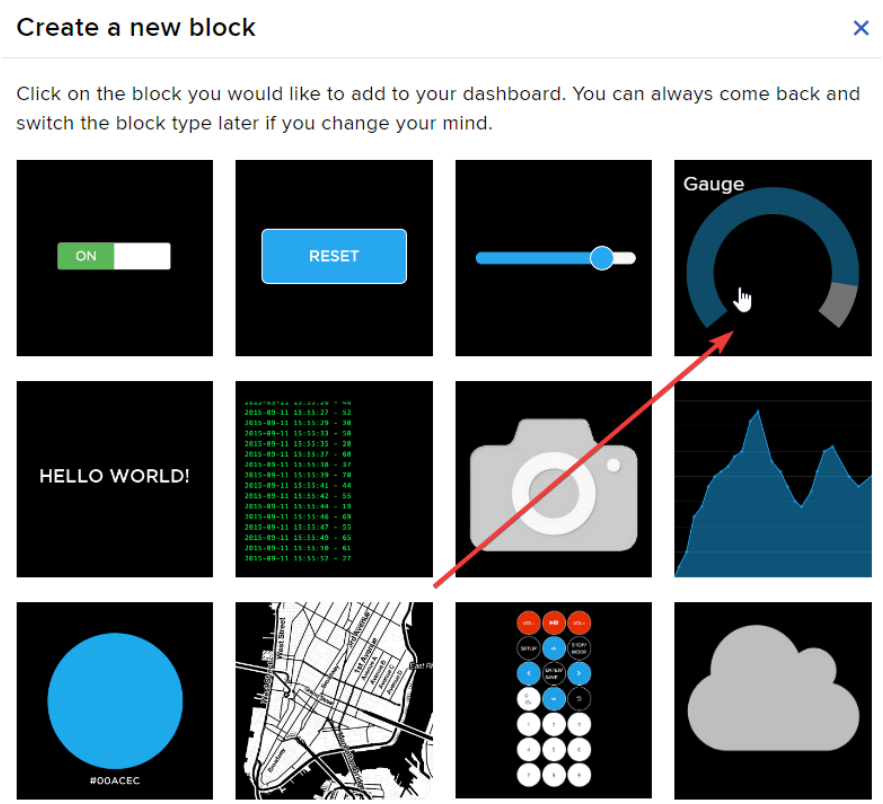


There are also options for editing layout, dark/light mode, block borders, and dashboard privacy (as well as deleting the dashboard). You can get a more in-depth learn guide for dashboards here (https://adafru.it/f5m). For now, click the **Create New Block** option.



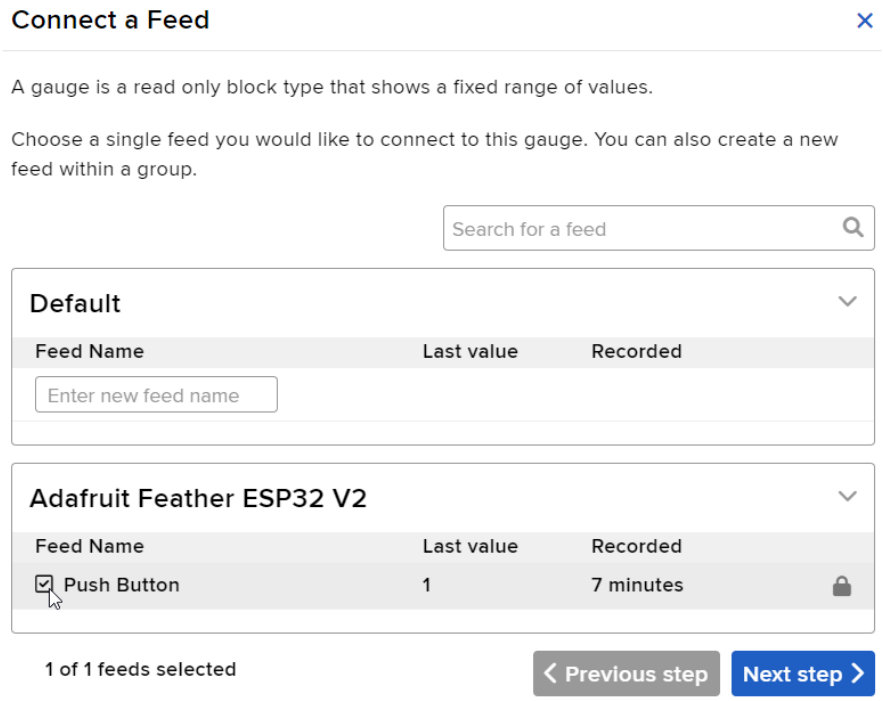
On the **Create a new block** dialog you'll see a list of available block types.

Click on the **Gauge** block type to start the process of adding the block.

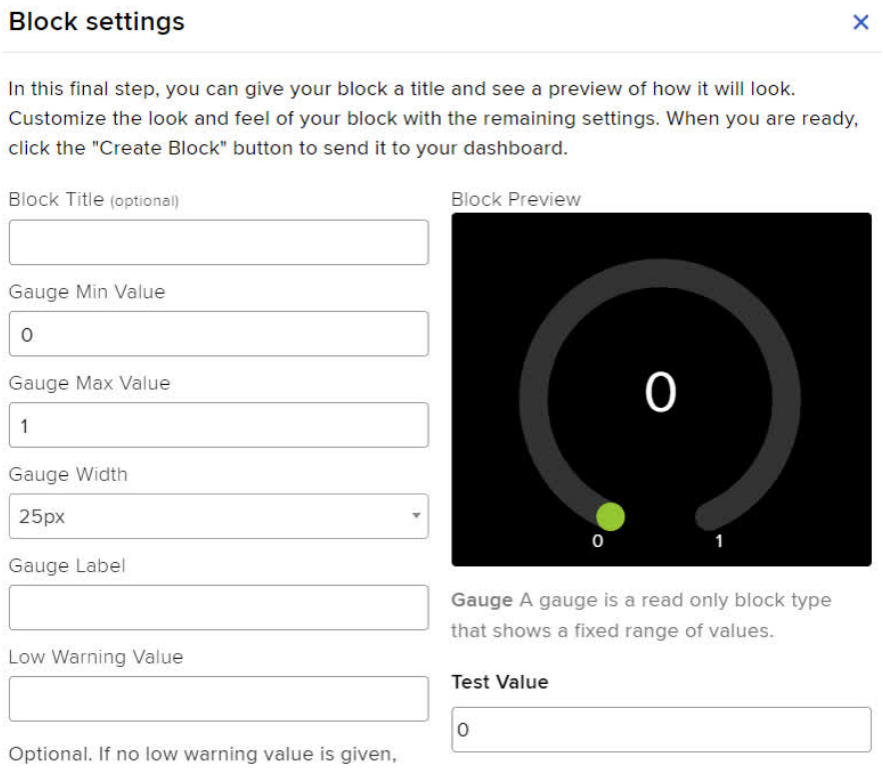


Next, you'll be taken to the **Connect a Feed** page, where you need to select which feed to connect to the Gauge block. Select the new button feed we created then use the **Next Step** button.

Note that the groups/devices in the list can be collapsed/expanded to hide or show their feeds. You may need to expand a group to find your button feed.



On the **Block Settings** page change the **Gauge Max Value** to **1**, the **Decimal Places** to **0**, and the **Gauge Label** to anything you like, or blank (an empty value in the box), then press the **Create Block** button.

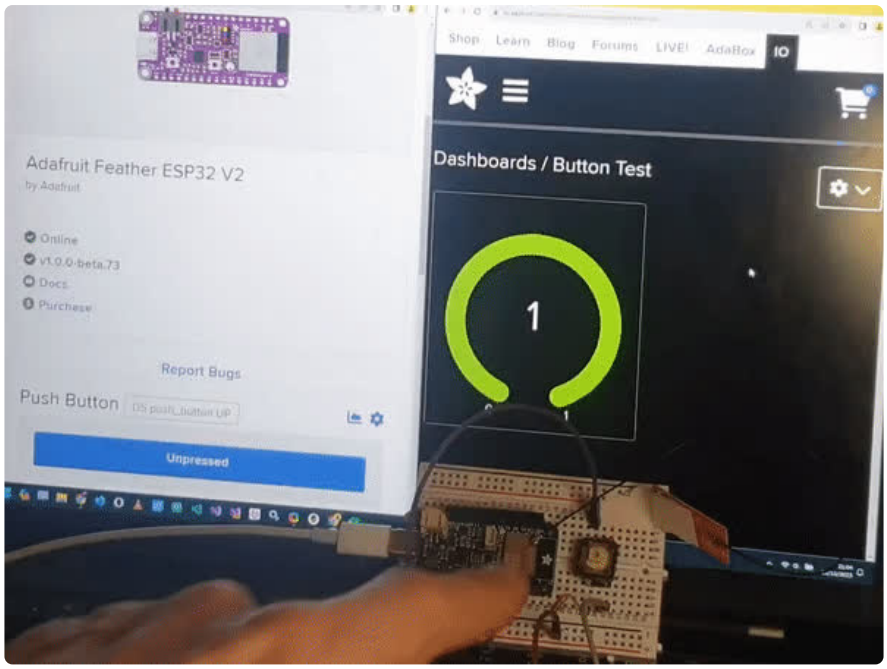


You may wish to Edit the layout (using the Dashboard Settings drop-down), or just continue onto usage so we can test this thing!

Usage

While looking at the WipperSnapper Device page, or the Dashboard, **press and hold the button** for a second or two then release.

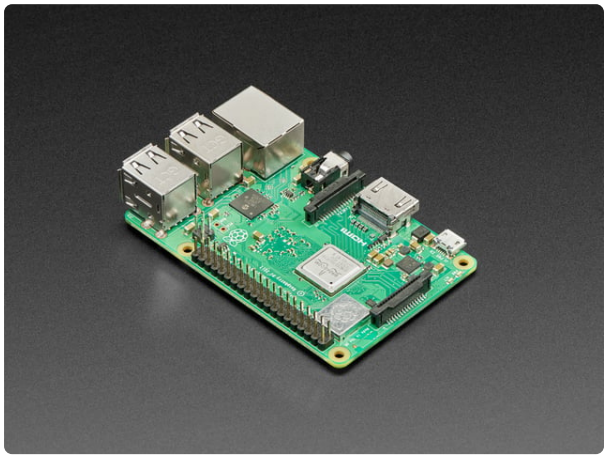
On the dashboard you'll see the Gauge change from 1 to 0 when the button is held down, and from 0 to 1 when released. The WipperSnapper Device page will show the Push Button component state change between Pressed and Unpressed.



Congratulations, you've successfully completed the guide!

Python Wiring

Parts



Raspberry Pi 3 - Model B+ - 1.4GHz Cortex-A53 with 1GB RAM

The Raspberry Pi 3 Model B is the most popular Raspberry Pi computer made, and the Pi Foundation knows you can always make a good thing better! And what could make the Pi 3...

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

If you're following along with a Raspberry Pi (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

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

1 x Jumper Wires https://www.adafruit.com/product/153

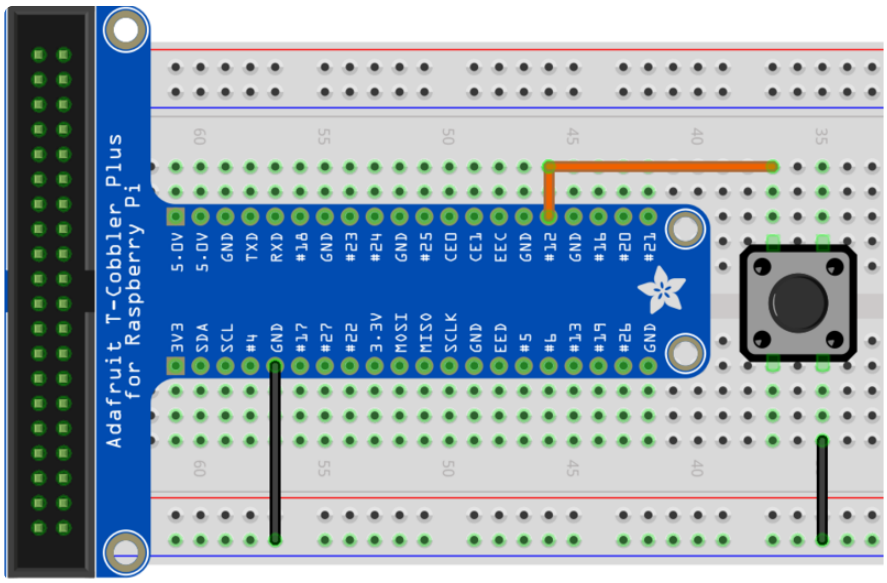
Breadboarding wire bundle.

---

1 x Button https://www.adafruit.com/product/1119

Tactile Switch Buttons (12mm square, 6mm tall) x 10 pack

Wiring



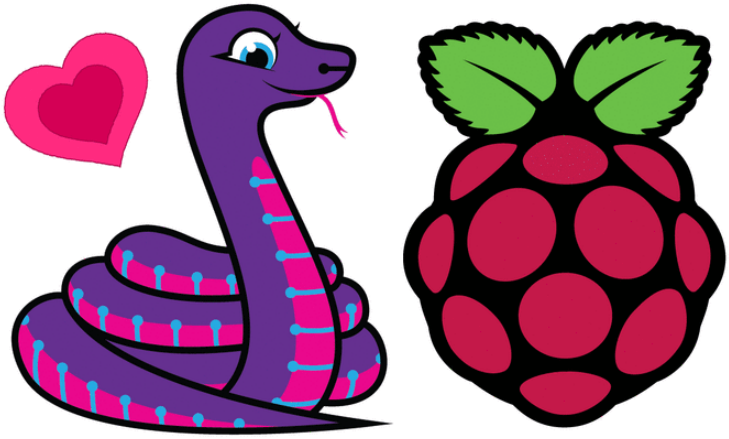
You'll need to make the following connections:

- Pi **GND** to a common **ground** rail.
- Pi **GND** to one side of the momentary button.
- Pi **Pin 12** to the other side of the momentary button.

Next, proceed to the Python Setup Page (https://adafru.it/BMB) of this guide.

Python Setup

If you're following along with a Raspberry Pi, Beaglebone or any other supported small linux computer, we'll use a special library called **adafruit_blinka** (https://adafru.it/BJS) (named after Blinka, the CircuitPython mascot (https://adafru.it/BJT)) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides. It's CircuitPython, on Pi!



Update your Pi and Python

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

In this page 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

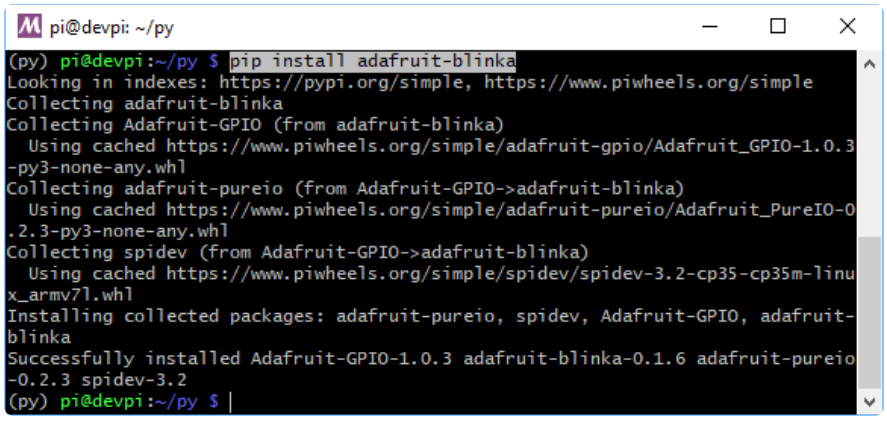
Install Python Libraries

Installing Adafruit Blinka Library

Now you're ready to install all the python support

Run the following command to install the Raspberry PI GPIO library:

```
pip3 install RPI.GPIO
```



Run the following command to install **adafruit_blinka**

```
pip3 install adafruit-blinka
```

The computer will install a few different libraries such as `adafruit-pureio` (our ioctl-only i2c library), `spidev` (for SPI interfacing), `Adafruit-GPIO` (for detecting your board) and of course `adafruit-blinka`

Installing Adafruit IO Python Library

We'll also need to install the Adafruit IO Python Client Library (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 Python IO Client's examples/ basics subdirectory.

Navigate to the root directory of your Pi:

```
cd ~
```

Then, download the latest version of the adafruit/io-client-python repository by running:

```
git clone https://github.com/adafruit/io-client-python.git
```

Navigate to that folder's example folder for the examples:

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

That's it! We're all set up.

Next, let's upload some code and learn how it works.

Python Code

Code

Before we run the script at the bottom of this page, we'll need to change `ADAFRUIT_IO_USERNAME` and `ADAFRUIT_IO_KEY` to the username and key for your Adafruit IO account.

- If you need the AIO Key, navigate to your Adafruit IO Profile (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'
```

We're going to set up an instance of the feed we created earlier:

`digital = aio.create_feed(Feed(name="digital"))`

In the `while True` loop, we're going to check the value of the button and send it to Adafruit IO. A delay (time.sleep()) has been added to avoid timing out by sending too many requests to Adafruit IO.

```
while True:
    if not button.value:
        button_current = 1
    else:
        button_current = 0

    print('Button -&gt; ', button_current)
    aio.send(digital.key, button_current)

    # avoid timeout from adafruit io
    time.sleep(1)
```

Running the Code

Make sure you're within the /io-client-python/examples/basics directory.

If you're not sure which directory you're in, you can check this by running `pwd` and you should see the following output from your terminal:

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

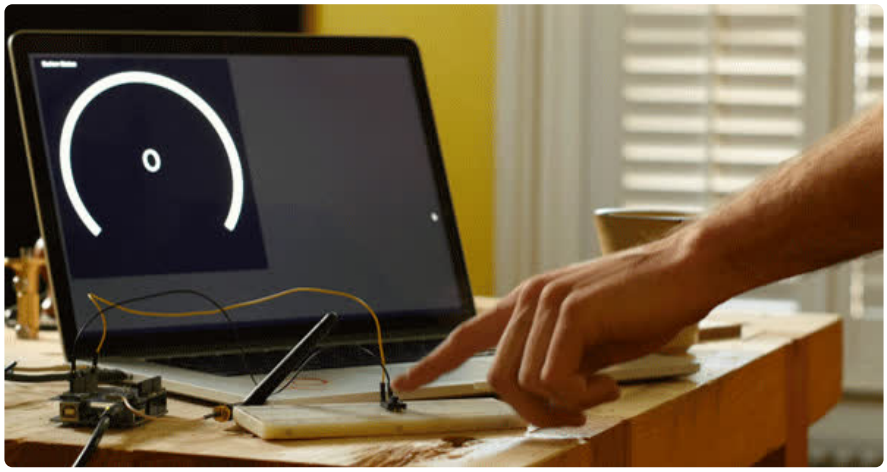
Let's run the script. In your terminal, run:

```
python3 digital-in.py
```

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

```
Button -&gt; 1
Button -&gt; 0
Button -&gt; 1
Button -&gt; 0
```

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



Code

```
"""
'digital_in.py'
==================================
Example of sending button values
to an Adafruit IO feed.

Author(s): Brent Rubell, Todd Treece
"""
# Import standard python modules
import time

# import Adafruit Blinka
import board
import digitalio

# import Adafruit IO REST client.
```

```
from Adafruit_IO import Client, Feed, RequestError

# 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)

try: # if we have a 'digital' feed
    digital = aio.feeds('digital')
except RequestError: # create a digital feed
    feed = Feed(name="digital")
    digital = aio.create_feed(feed)

# button set up
button = digitalio.DigitalInOut(board.D12)
button.direction = digitalio.Direction.INPUT
button.pull = digitalio.Pull.UP
button_current = 0


while True:
    if not button.value:
        button_current = 1
    else:
        button_current = 0

    print('Button -> ', button_current)
    aio.send(digital.key, button_current)

    # avoid timeout from adafruit io
    time.sleep(1)
```

Adafruit IO FAQ

Encountering an issue with your Adafruit IO Arduino Project?

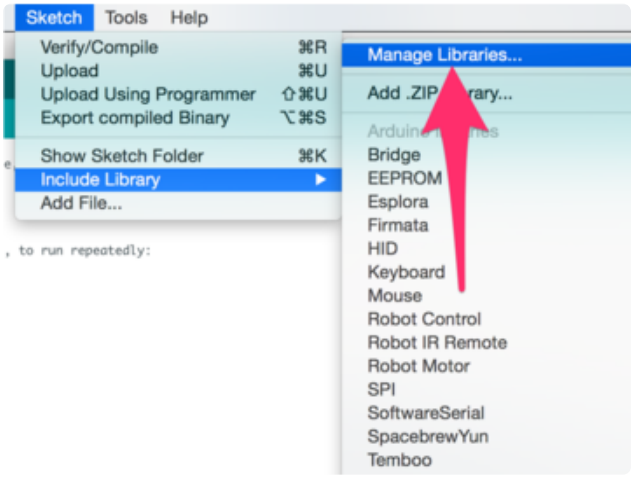
If you're having an issue compiling, connecting, or troubleshooting your project, check this page first.

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

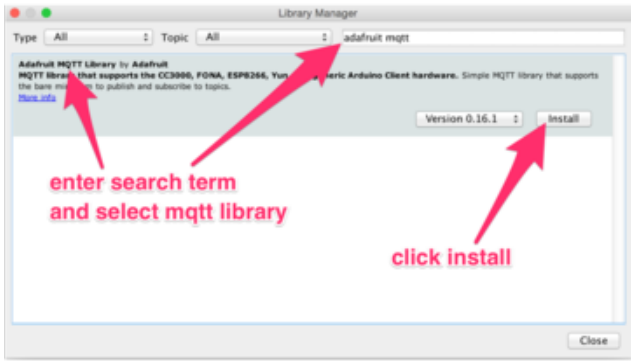
I encounter the following error when compiling my sketch:

`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) 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).

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) 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.