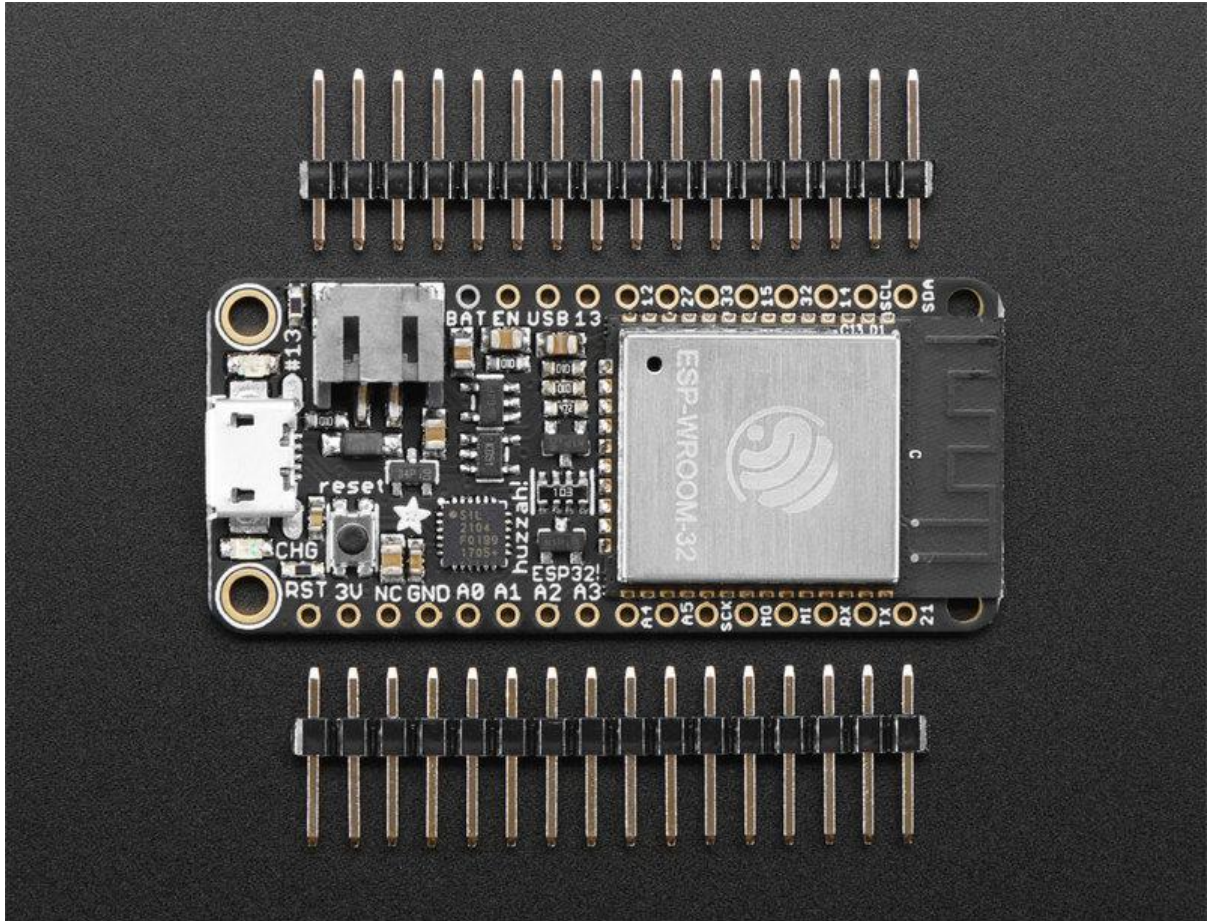




Adafruit HUZZAH32 - ESP32 Feather

Created by lady ada



<https://learn.adafruit.com/adafruit-huzzah32-esp32-feather>

Last updated on 2023-04-28 03:51:52 PM EDT

Table of Contents

Overview	5
Pinouts	8
<ul style="list-style-type: none">• Power Pins• Logic pins• Serial pins• I2C & SPI pins• GPIO & Analog Pins	
Assembly	12
<ul style="list-style-type: none">• Header Options!• Soldering in Plain Headers• Prepare the header strip:• Add the breakout board:• And Solder!• Soldering on Female Header• Tape In Place• Flip & Tack Solder• And Solder!	
Power Management	22
<ul style="list-style-type: none">• Battery + USB Power• Power Supplies• Measuring Battery• ENable pin• Alternative Power Options	
Using with Arduino IDE	26
WipperSnapper Setup	27
<ul style="list-style-type: none">• What is WipperSnapper• Sign up for Adafruit.io• Add a New Device to Adafruit IO• Feedback• Troubleshooting• "Uninstalling" WipperSnapper	
WipperSnapper Essentials	33
LED Blink	35
<ul style="list-style-type: none">• Where is the LED on my board?• Create a LED Component on Adafruit IO• Usage	
Read a Push-button	38
<ul style="list-style-type: none">• Parts• Wiring• Create a Push-button Component on Adafruit IO	
Analog Input	43
<ul style="list-style-type: none">• Analog to Digital Converter (ADC)	

- [Potentiometers](#)
- [Hardware](#)
- [Wire Up the Potentiometer](#)
- [Create a Potentiometer Component on Adafruit IO](#)
- [Read Analog Pin Values](#)
- [Read Analog Pin Voltage Values](#)

[I2C Sensor](#) 49

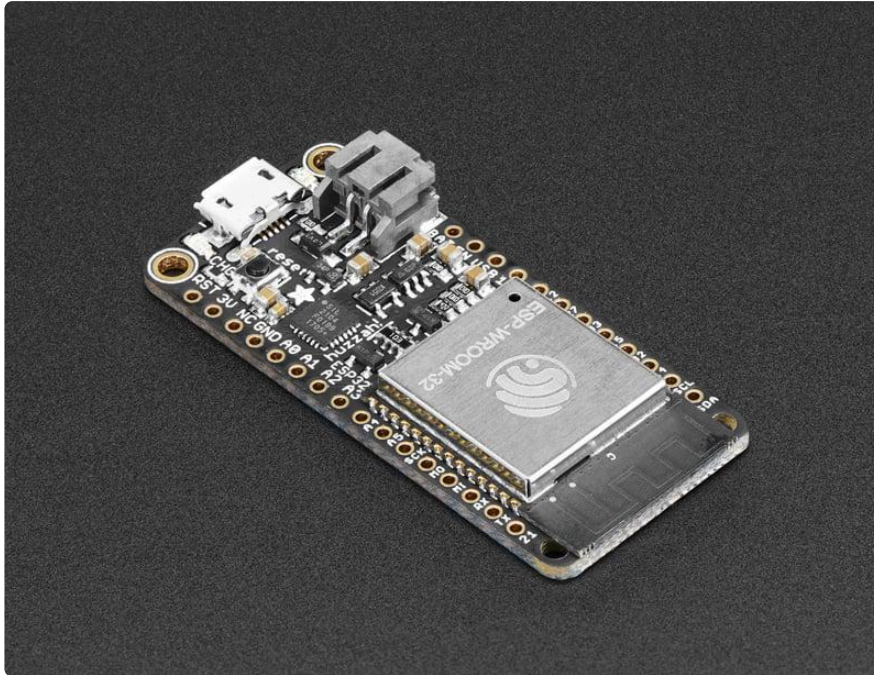
- [Parts](#)
- [Wiring](#)
- [Add an MCP9808 Component](#)
- [Read I2C Sensor Values](#)

[ESP32 F.A.Q](#) 55

[Downloads](#) 56

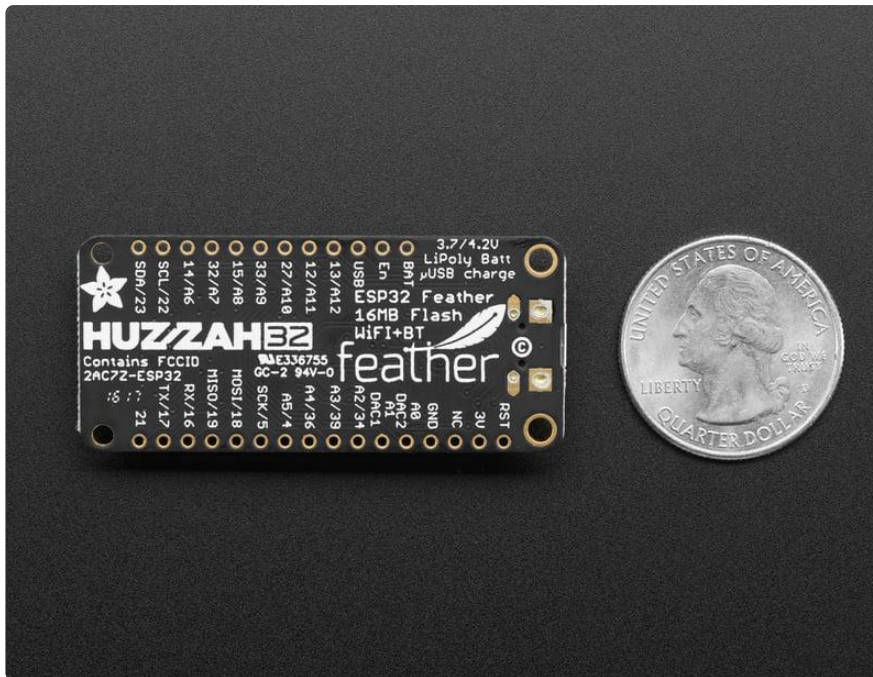
- [Files](#)
- [Schematic and Fabrication print](#)

Overview



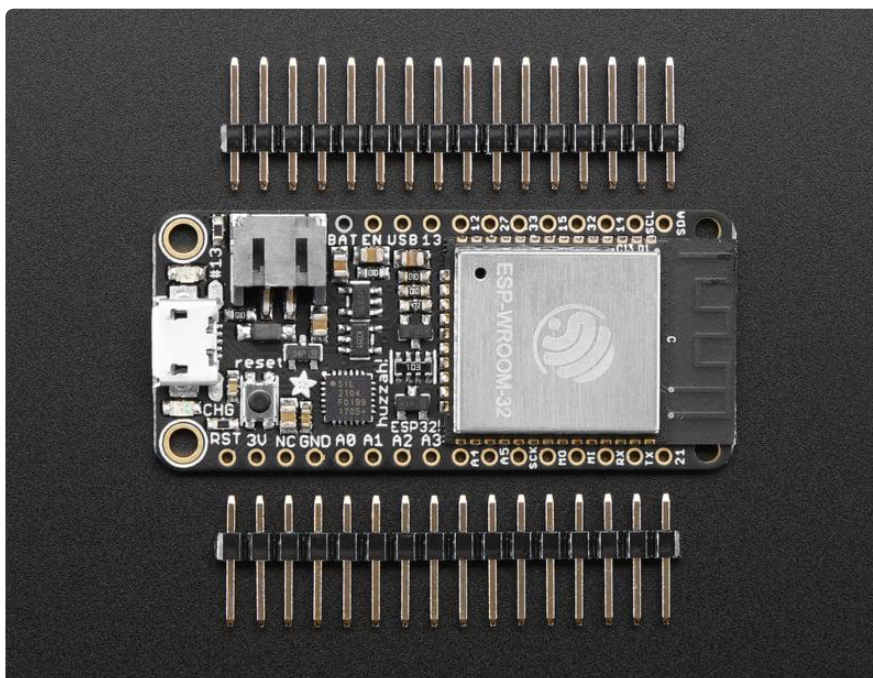
Aww yeah, it's the Feather you have been waiting for! The HUZAZH32 is our ESP32-based Feather, made with the official WROOM32 module. We packed everything you love about Feathers: built in USB-to-Serial converter, automatic bootloader reset, Lithium Ion/Polymer charger, and all the GPIO brought out so you can use it with any of our Feather Wings.

That module nestled in at the end of this Feather contains a dual-core ESP32 chip, 4 MB of SPI Flash, tuned antenna, and all the passives you need to take advantage of this powerful new processor. The ESP32 has both WiFi and Bluetooth Classic/LE support. That means it's perfect for just about any wireless or Internet-connected project.



Because it is part of our [Feather eco-system you can take advantage of the 50+ Wings \(\)](#) that we've designed, to add all sorts of cool accessories

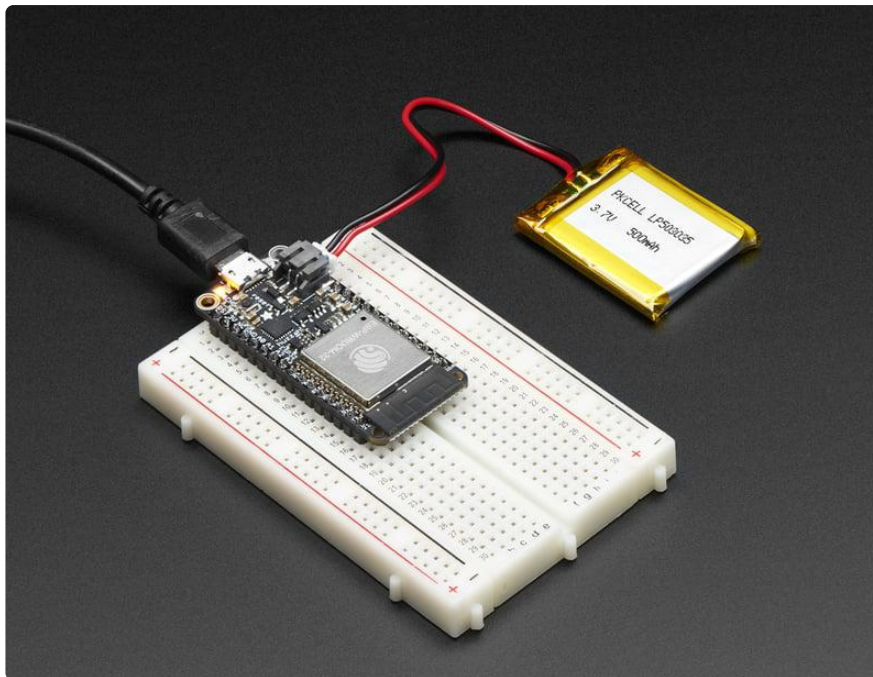
The ESP32 is a perfect upgrade from the ESP8266 that has been so popular. In comparison, the ESP32 has way more GPIO, plenty of analog inputs, two analog outputs, multiple extra peripherals (like a spare UART), two cores so you don't have to yield to the WiFi manager, much higher-speed processor, etc. etc! We think that as the ESP32 gets traction, we'll see more people move to this chip exclusively, as it is so full-featured.



Please note: The ESP32 is still targeted to developers. Not all of the peripherals are fully documented with example code, and there are some bugs still being found and fixed. We got all of our Featherwings working under Arduino IDE, so you can expect things like I2C and SPI and analog reads to work. But other elements are still under development. For that reason, we recommend this Feather for makers who have some experience with microcontroller programming, and not as a first dev board.

Here are [specifications from Espressif about the ESP32 \(\)](#)

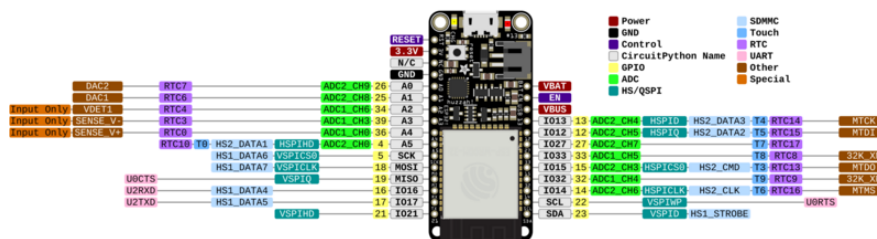
- 240 MHz dual core Tensilica LX6 microcontroller with 600 DMIPS
- Integrated 520 KB SRAM
- Integrated 802.11b/g/n HT40 Wi-Fi transceiver, baseband, stack and LWIP
- Integrated dual mode Bluetooth (classic and BLE)
- 4 MByte flash
- On-board PCB antenna
- Ultra-low noise analog amplifier
- Hall sensor
- 10x capacitive touch interface
- 32 kHz crystal oscillator
- 3 x UARTs (only two are configured by default in the Feather Arduino IDE support, one UART is used for bootloading/debug)
- 3 x SPI (only one is configured by default in the Feather Arduino IDE support)
- 2 x I2C (only one is configured by default in the Feather Arduino IDE support)
- 12 x ADC input channels
- 2 x I2S Audio
- 2 x DAC
- PWM/timer input/output available on every GPIO pin
- OpenOCD debug interface with 32 kB TRAX buffer
- SDIO main/secondary 50 MHz
- SD-card interface support



Comes fully assembled and tested, with a USB interface that lets you quickly use it with the Arduino IDE or the low-level ESP32 IDF. We also toss in some header so you can solder it in and plug into a solderless breadboard. Lipoly battery and USB cable not included (but we do have lots of options in the shop if you'd like!)

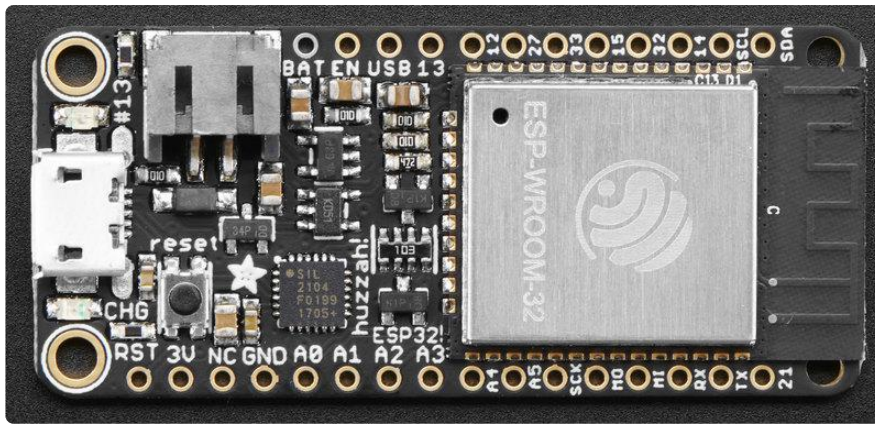
Pinouts

Adafruit HUZAH32 ESP32 Feather
<http://www.adafruit.com/products/3405>

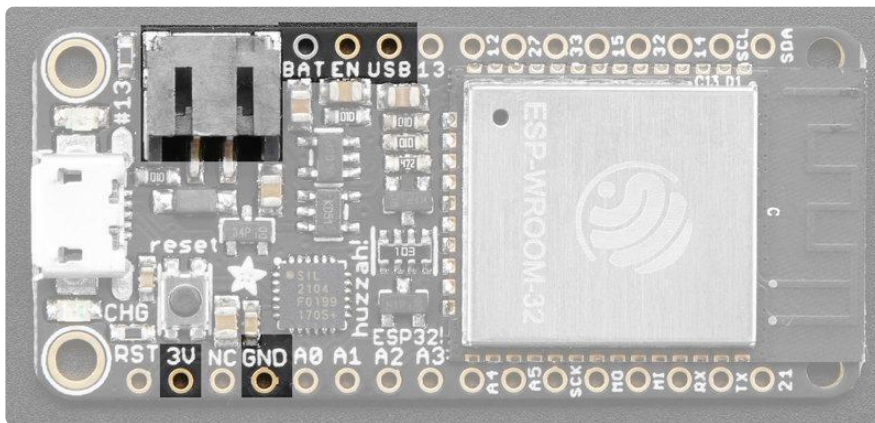


[Click here to view a PDF version of the pinout diagram \(\)](#)

One of the great things about the ESP32 is that it has tons more GPIO than the ESP8266. You won't have to juggle or multiplex your IO pins! There's a few things to watch out for so please read through the pinouts carefully



Power Pins



- GND - this is the common ground for all power and logic
- BAT - this is the positive voltage to/from the JST jack for the optional Lipoly battery
- USB - this is the positive voltage to/from the micro USB jack if connected
- EN - this is the 3.3V regulator's enable pin. It's pulled up, so connect to ground to disable the 3.3V regulator
- 3V - this is the output from the 3.3V regulator. The regulator can supply 500mA peak but half of that is drawn by the ESP32, and it's a fairly power-hungry chip.

So if you need a ton of power for stuff like LEDs, motors, etc. Use the USB or BATT pins, and an additional regulator

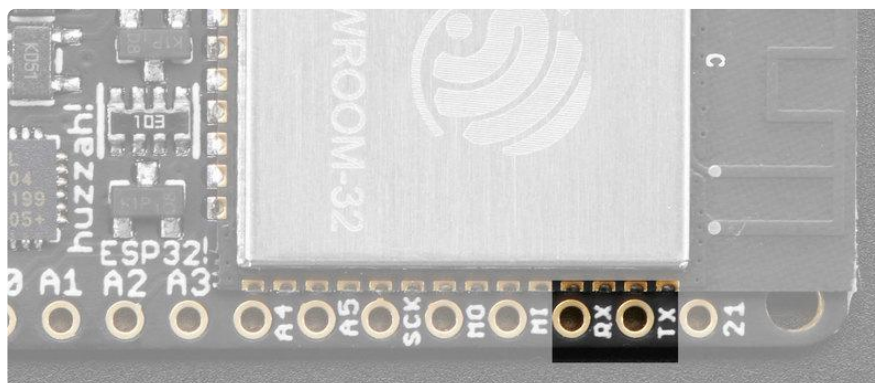
Logic pins

This is the general purpose I/O pin set for the microcontroller. All logic is 3.3V

The ESP32 runs on 3.3V power and logic, and unless otherwise specified, GPIO pins are not 5V safe!

Serial pins

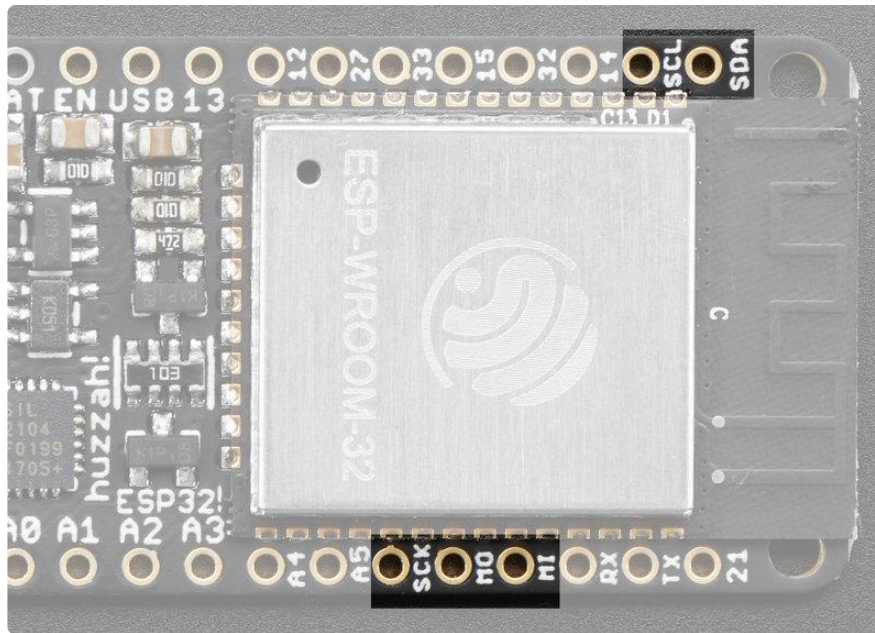
RX and TX are the additional Serial1 pins, and are not connected to the USB/Serial converter. That means you can use them to connect to UART-devices like GPS's, fingerprint sensors, etc.



The TX pin is the output from the module. The RX pin is the input into the module. Both are 3.3V logic

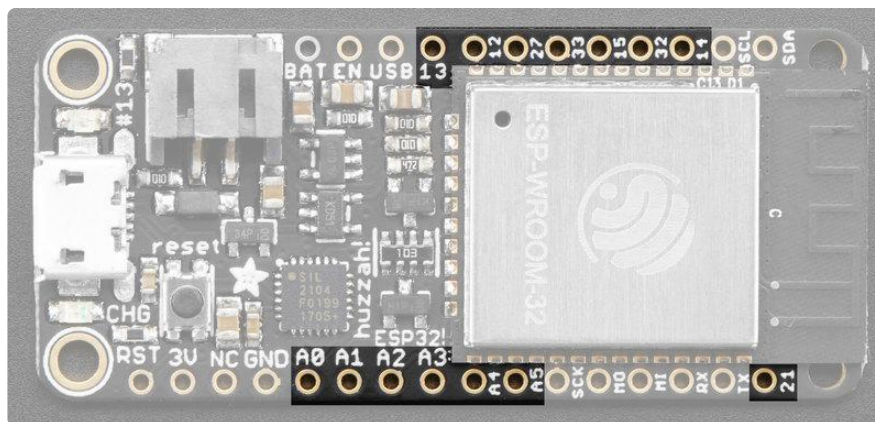
I2C & SPI pins

You can use the ESP32 to control I2C and SPI devices, sensors, outputs, etc. If using with Arduino, the standard Wire and SPI devices work as you'd expect!



Note that the I2C pins do not have pullup resistors already! You must add them if you want to communicate with an I2C device

GPIO & Analog Pins



There are tons of GPIO and analog inputs available to you for connecting LEDs, buttons, switches, sensors, etc. Here's the remaining pins available.

Bottom row:

- A0 - this is an analog input A0 and also an analog output DAC2. It can also be used as a GPIO #26. It uses ADC #2
- A1 - this is an analog input A1 and also an analog output DAC1. It can also be used as a GPIO #25. It uses ADC #2
- A2 - this is an analog input A2 and also GPIO #34. Note it is not an output-capable pin! It uses ADC #1

- A3 - this is an analog input A3 and also GPIO #39. Note it is not an output-capable pin! It uses ADC #1
- A4 - this is an analog input A4 and also GPIO #36. Note it is not an output-capable pin! It uses ADC #1
- A5 - this is an analog input A5 and also GPIO #4. It uses ADC #2
- 21 - General purpose IO pin #21

Top row:

- 13 - This is GPIO #13 and also an analog input A12 on ADC #2. It's also connected to the red LED next to the USB port
- 12 - This is GPIO #12 and also an analog input A11 on ADC #2. This pin has a pull-down resistor built into it, we recommend using it as an output only, or making sure that the pull-down is not affected during boot.
- 27 - This is GPIO #27 and also an analog input A10 on ADC #2
- 33 - This is GPIO #33 and also an analog input A9 on ADC #1. It can also be used to connect a 32 KHz crystal.
- 15 - This is GPIO #15 and also an analog input A8 on ADC #2
- 32 - This is GPIO #32 and also an analog input A7 on ADC #1. It can also be used to connect a 32 KHz crystal.
- 14 - This is GPIO #14 and also an analog input A6 on ADC #2

There's also an external analog input

- A13 - This is general purpose input #35 and also an analog input A13, which is a resistor divider connected to the VBAT line

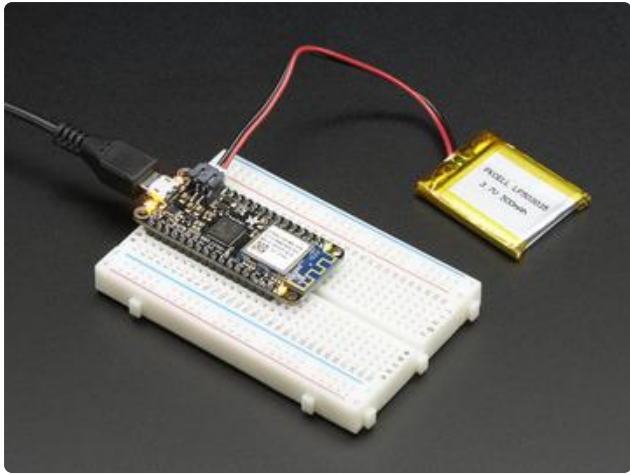
Note you can only read analog inputs on ADC #2 once WiFi has started as it is shared with the WiFi module.

Assembly

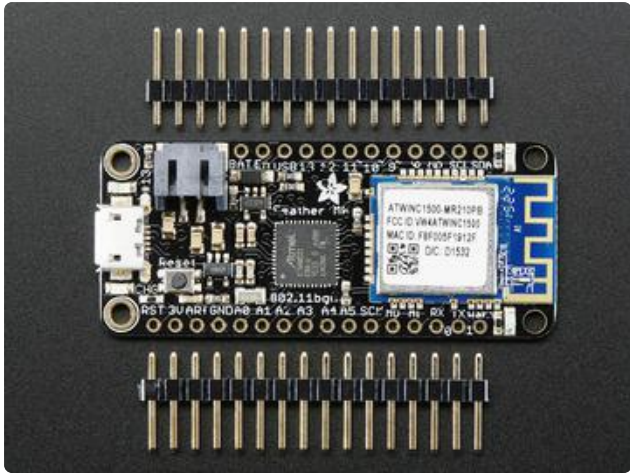
We ship Feathers fully tested but without headers attached - this gives you the most flexibility on choosing how to use and configure your Feather

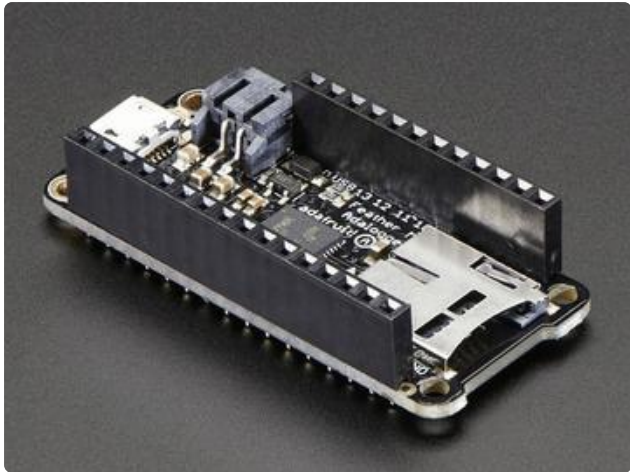
Header Options!

Before you go gung-ho on soldering, there's a few options to consider!

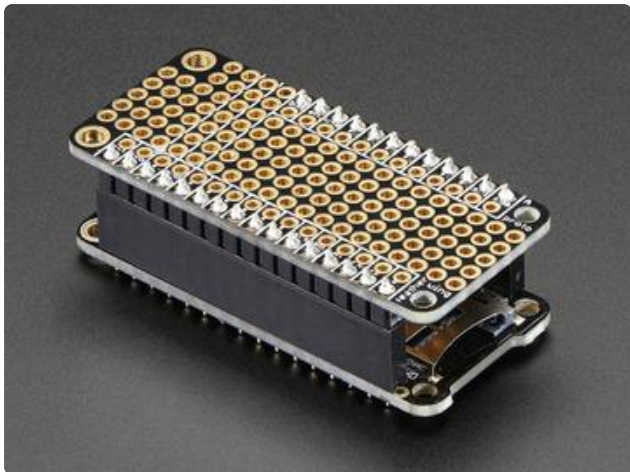


The first option is soldering in plain male headers, this lets you plug in the Feather into a solderless breadboard

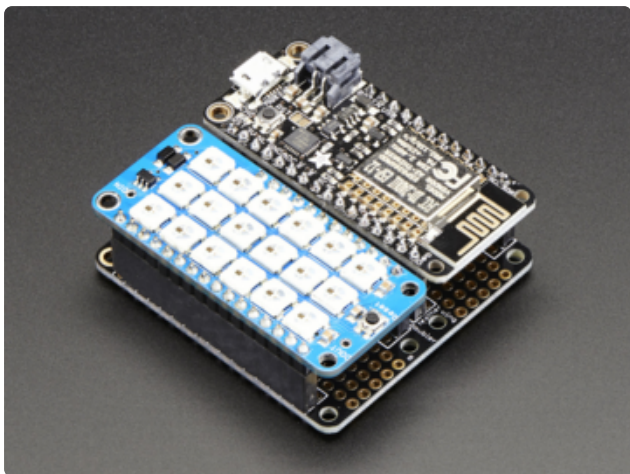


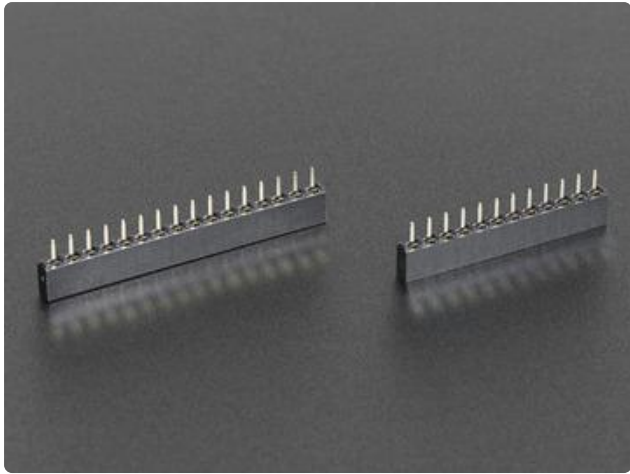


Another option is to go with socket female headers. This won't let you plug the Feather into a breadboard but it will let you attach featherwings very easily

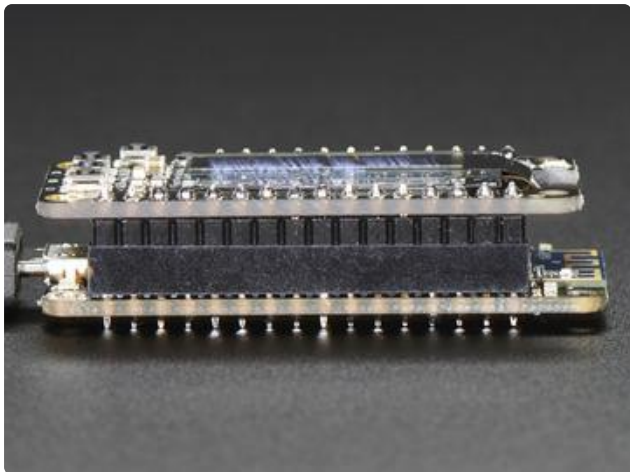


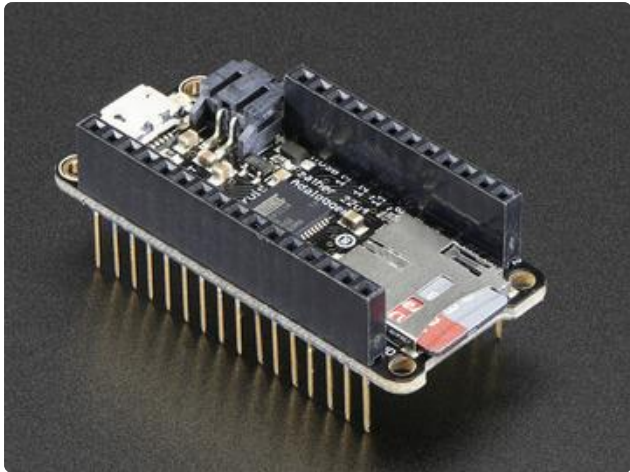
A few Feather boards require access to top-side components like buttons or connectors, making stacking impractical. Sometimes you can stack in the opposite order—FeatherWing underneath—or, if both Feather and Wing require top-side access, place the boards side-by-side with a [FeatherWing Doubler](#) () or [Tripler](#) ().



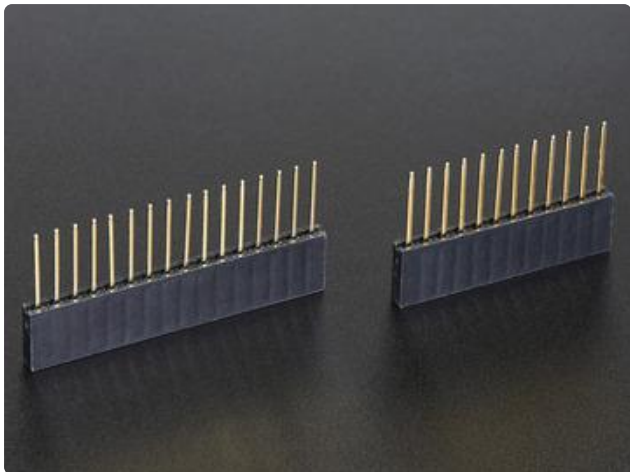


We also have 'slim' versions of the female headers, that are a little shorter and give a more compact shape

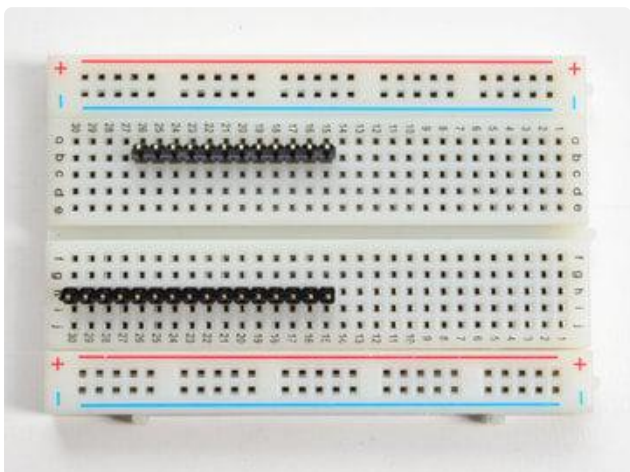




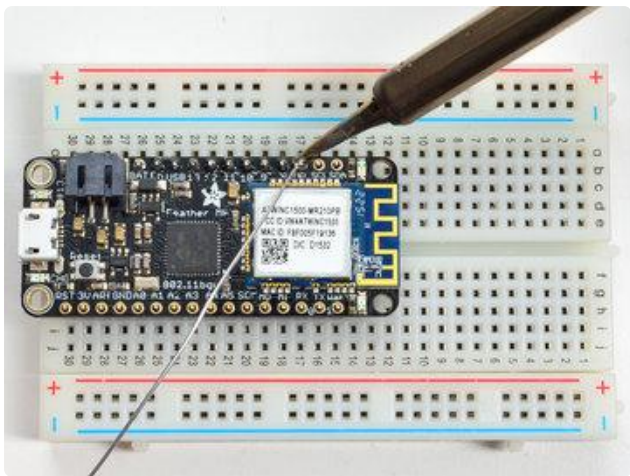
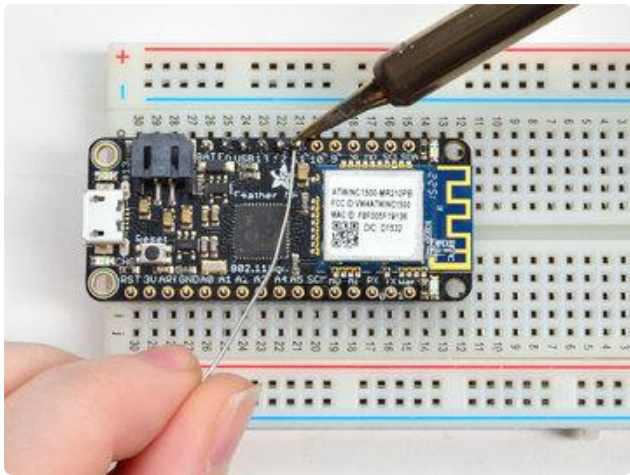
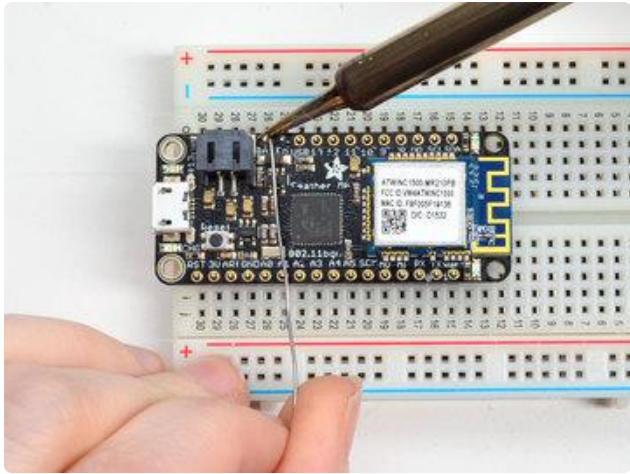
Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a solderless breadboard and plug a featherwing on top. But its a little bulky



Soldering in Plain Headers



Prepare the header strip:
Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - long pins down



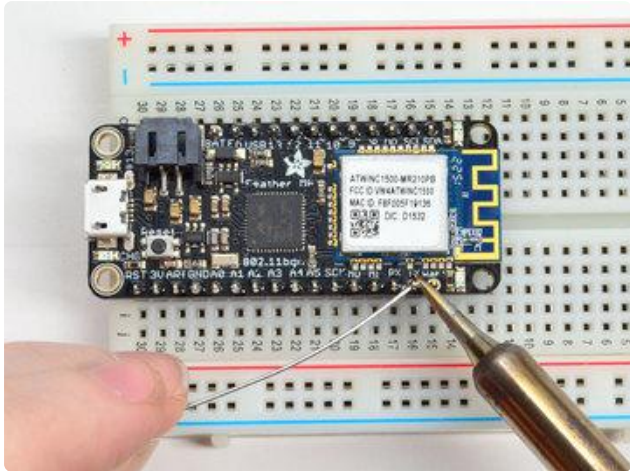
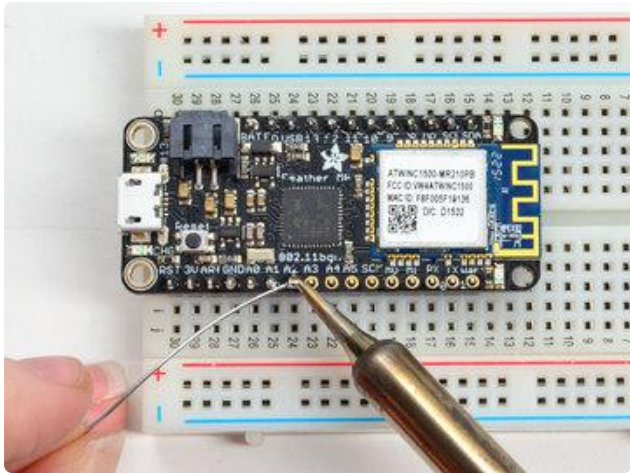
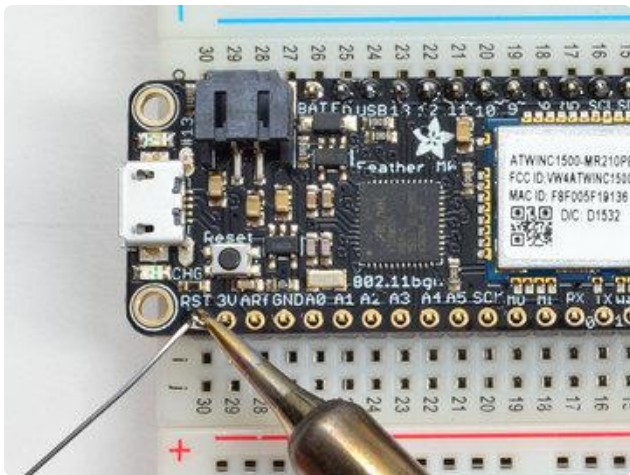
Add the breakout board:

Place the breakout board over the pins so that the short pins poke through the breakout pads

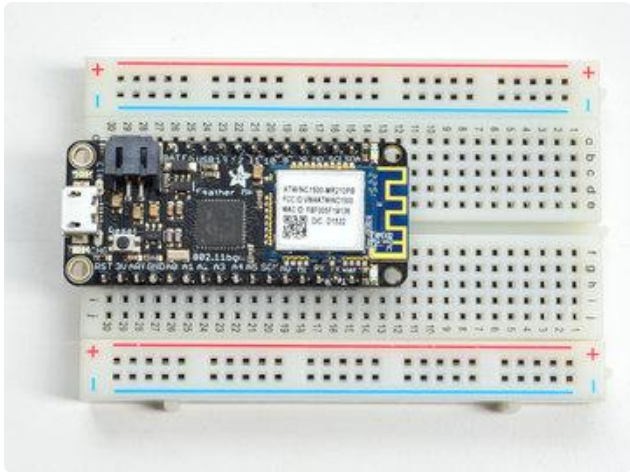
And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](#) ()).



Solder the other strip as well.



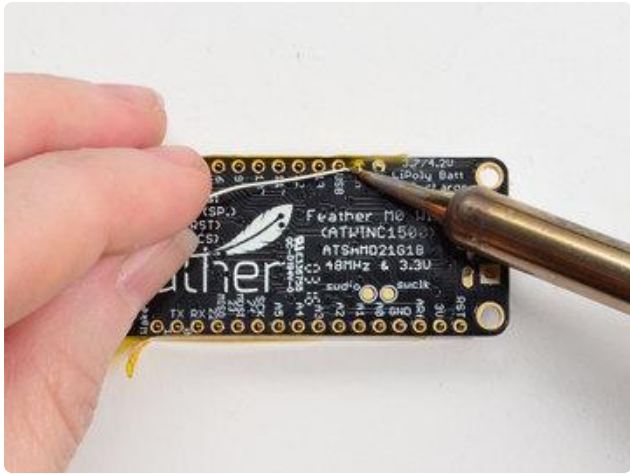
You're done! Check your solder joints visually and continue onto the next steps

Soldering on Female Header



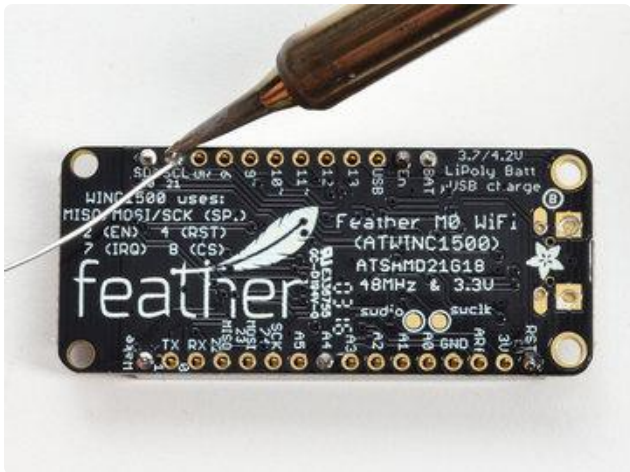
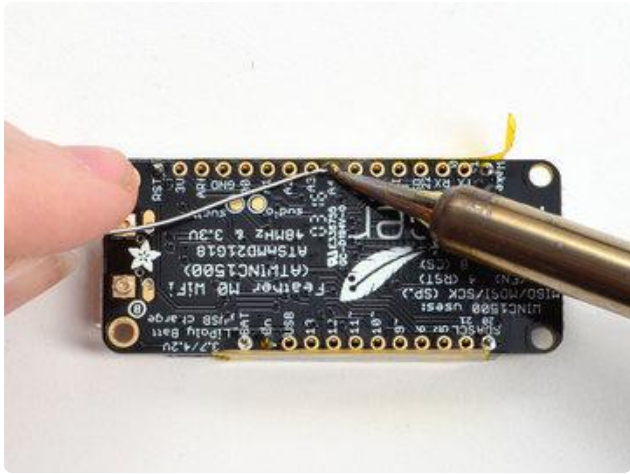
Tape In Place

For sockets you'll want to tape them in place so when you flip over the board they don't fall out



Flip & Tack Solder

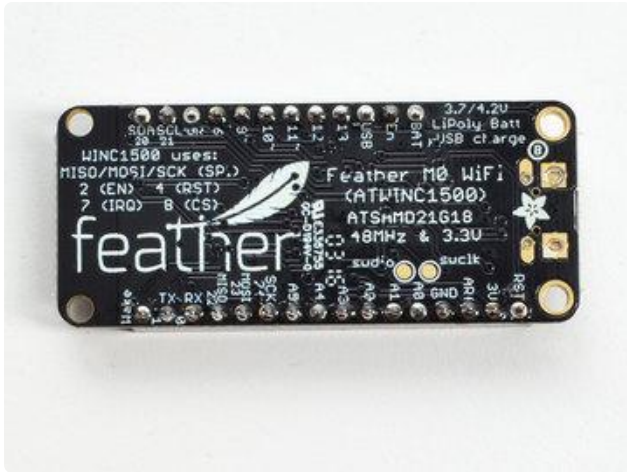
After flipping over, solder one or two points on each strip, to 'tack' the header in place



And Solder!

Be sure to solder all pins for reliable electrical contact.

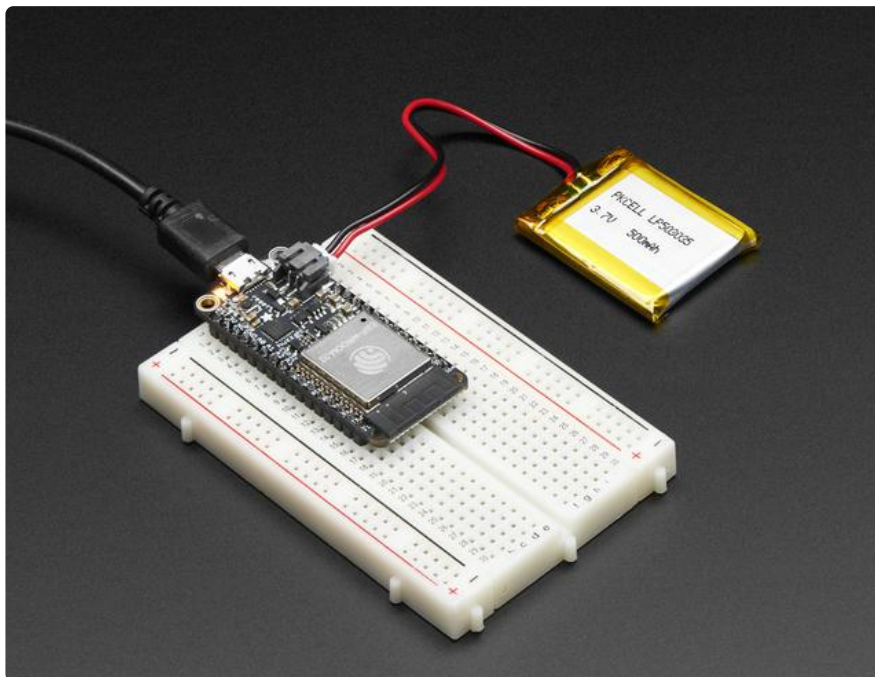
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering \(\)](#)).



You're done! Check your solder joints visually and continue onto the next steps



Power Management



Battery + USB Power

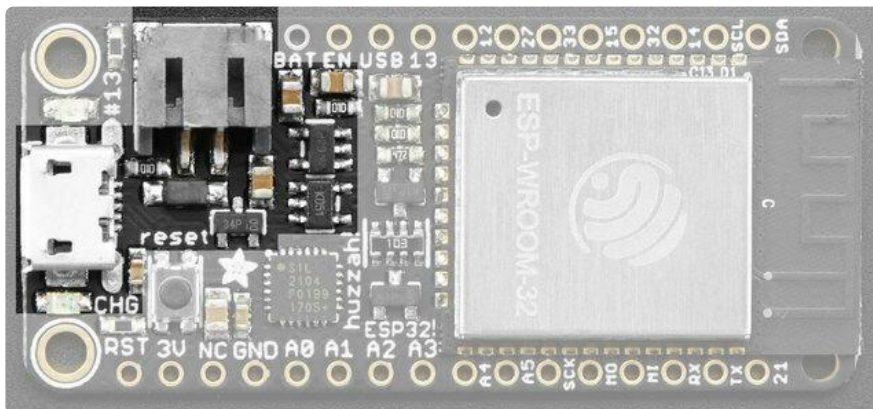
We wanted to make our Feather boards easy to power both when connected to a computer as well as via battery.

There's two ways to power a Feather:

1. You can connect with a USB cable (just plug into the jack) and the Feather will regulate the 5V USB down to 3.3V.
2. You can also connect a 4.2/3.7V Lithium Polymer (LiPo/LiPoly) or Lithium Ion (Lilon) battery to the JST jack. This will let the Feather run on a rechargeable battery.

When the USB power is powered, it will automatically switch over to USB for power, as well as start charging the battery (if attached). This happens 'hot-swap' style so you can always keep the LiPoly connected as a 'backup' power that will only get used when USB power is lost.

The JST connector polarity is matched to Adafruit LiPoly batteries. Using wrong polarity batteries can destroy your Feather. Many customers try to save money by purchasing Lipoly batteries from Amazon only to find that they plug them in and the Feather is destroyed!



The above shows the Micro USB jack (left), LiPoly JST jack (top left), as well as the 3.3V regulator (to the right of the JST jack), changeover diode+transistor (below the JST jack) and the LiPoly charging circuitry (right below the regulator).

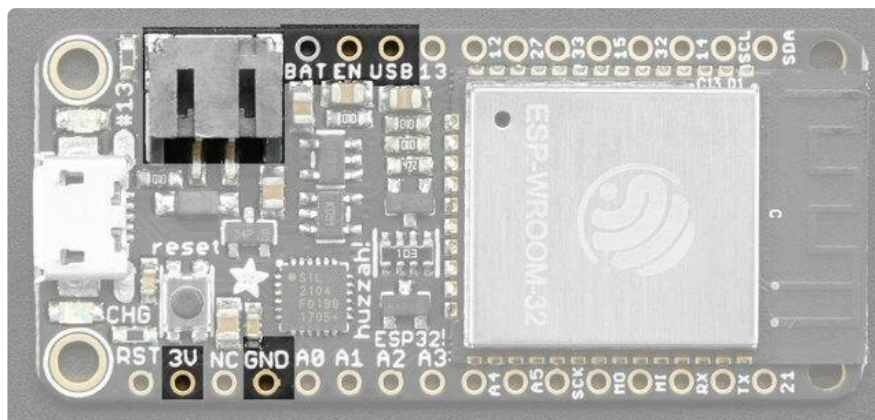
There's also a CHG LED next to the USB jack, which will light up while the battery is charging. This LED might also flicker if the battery is not connected, it's normal.

The charge LED is automatically driven by the LiPoly charger circuit. It will try to detect a battery and is expecting one to be attached. If there isn't one it may flicker once in a while when you use power because it's trying to charge a (non-existent) battery. It's not harmful, and its totally normal!

Power Supplies

You have a lot of power supply options here! We bring out the BAT pin, which is tied to the LiPoly JST connector, as well as USB which is the +5V from USB if connected. We also have the 3V pin which has the output from the 3.3V regulator. We use a 500mA peak regulator. While you can get 500mA from it, you can't do it continuously from 5V as it will overheat the regulator.

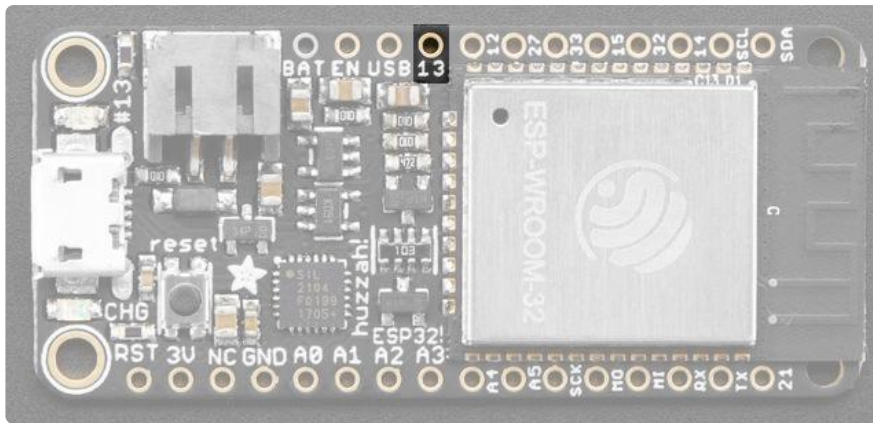
Please budget 250mA for the WROOM32 module. We use this to power the ESP32 which draws about 200mA continuous. The good news is you can put the ESP32 into sleep and low-power modes much easier.



Measuring Battery

If you're running off of a battery, chances are you wanna know what the voltage is at! That way you can tell when the battery needs recharging. LiPoly batteries are 'maxed out' at 4.2V and stick around 3.7V for much of the battery life, then slowly sink down to 3.2V or so before the protection circuitry cuts it off. By measuring the voltage you can quickly tell when you're heading below 3.7V.

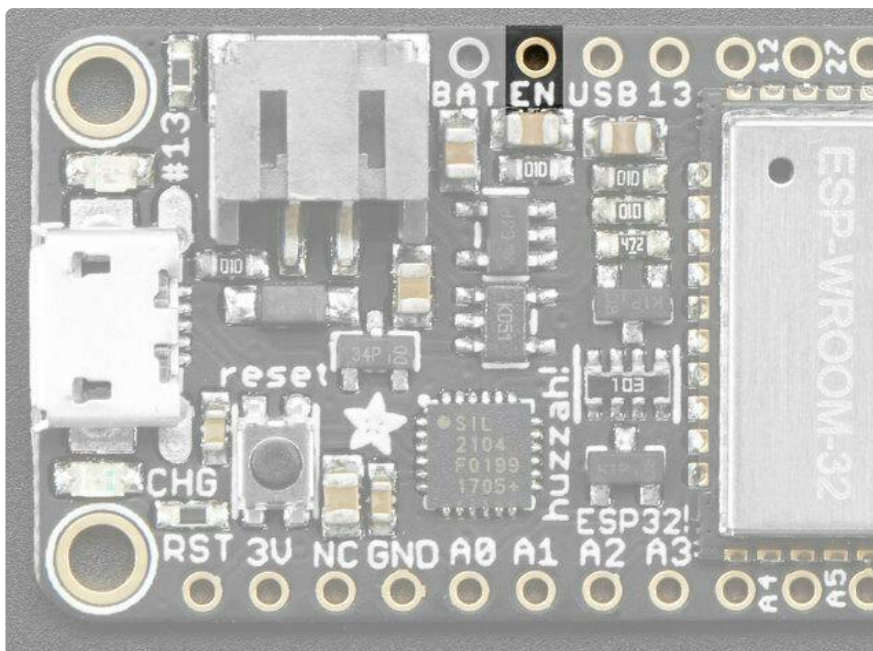
Since the ESP32 has tons of ADC pins, we 'sacrifice' one for Lipoly battery monitoring. You can read half of the battery voltage off of A13. Don't forget to double the voltage you read, since there is a divider.



ENable pin

If you'd like to turn off the 3.3V regulator, you can do that with the EN(able) pin. Simply tie this pin to Ground and it will disable the 3V regulator. The BAT and USB pins will still be powered.

This will turn off the ESP32 processor as well as all onboard circuitry except the USB-Serial converter.



Alternative Power Options

The two primary ways for powering a feather are a 3.7/4.2V LiPo battery plugged into the JST port or a USB power cable.

If you need other ways to power the Feather, here's what we recommend:

- For permanent installations, a [5V 1A USB wall adapter \(\)](#) will let you plug in a USB cable for reliable power
- For mobile use, where you don't want a LiPoly, [use a USB battery pack! \(\)](#)
- If you have a higher voltage power supply, [use a 5V buck converter \(\)](#) and wire it to a [USB cable's 5V and GND input \(\)](#)

Here's what you cannot do:

- Do not use alkaline or NiMH batteries and connect to the battery port - this will destroy the LiPoly charger and there's no way to disable the charger
- Do not use 7.4V RC batteries on the battery port - this will destroy the board

The Feather is not designed for external power supplies - this is a design decision to make the board compact and low cost. It is not recommended, but technically possible:

- Connect an external 3.3V power supply to the 3V and GND pins. Not recommended, this may cause unexpected behavior and the EN pin will no longer work. Also this doesn't provide power on BAT or USB and some Feathers/Wings use those pins for high current usages. You may end up damaging your Feather.
- Connect an external 5V power supply to the USB and GND pins. Not recommended, this may cause unexpected behavior when plugging in the USB port because you will be back-powering the USB port, which could confuse or damage your computer.

Using with Arduino IDE

We primarily recommend using the ESP32 Feather with Arduino.

[Check out the Espressif Arduino repository for details on how to install it \(\)](#)

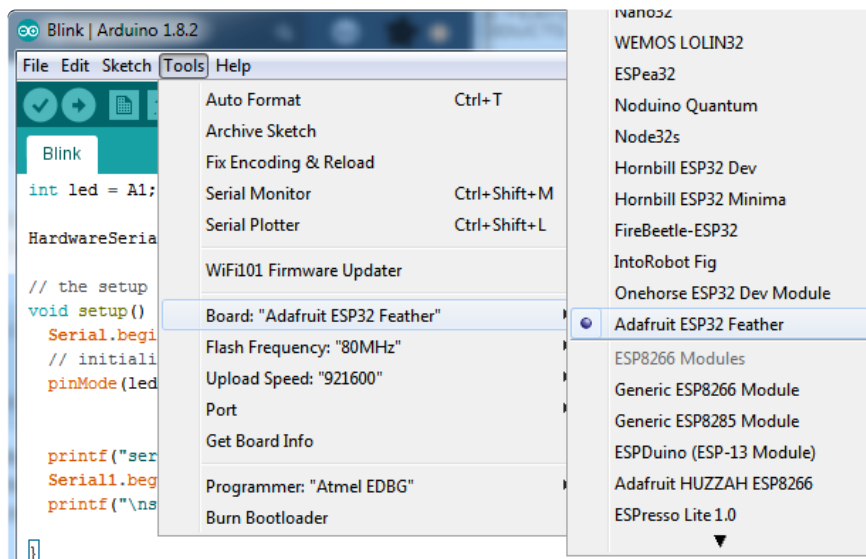
Don't forget you will also need to install the SiLabs CP2104 Driver

Click here to download the CP2104
USB Driver

Once installed, use the Adafruit ESP32 Feather board in the dropdown

For Upload speed we've found 921600 baud works great.

The ESP32 uses a fair amount of current, so if you're getting flakey behavior make sure you are plugging your console cable into either a motherboard USB port or a powered USB hub. Don't use the 'extra' USB port on your monitor or keyboard.



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 \(\)](#), a web platform designed [\(by Adafruit! \(\)\)](#) 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.

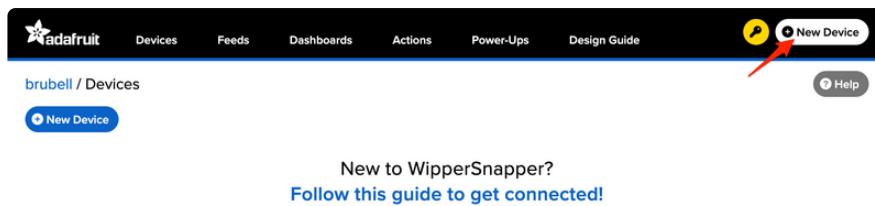
From there, you can add components to your board such as buttons, switches, potentiometers, sensors, and more! Components are dynamically added to hardware, so you can immediately start interacting, logging, and streaming the data your projects produce without writing code.

Sign up for Adafruit.io

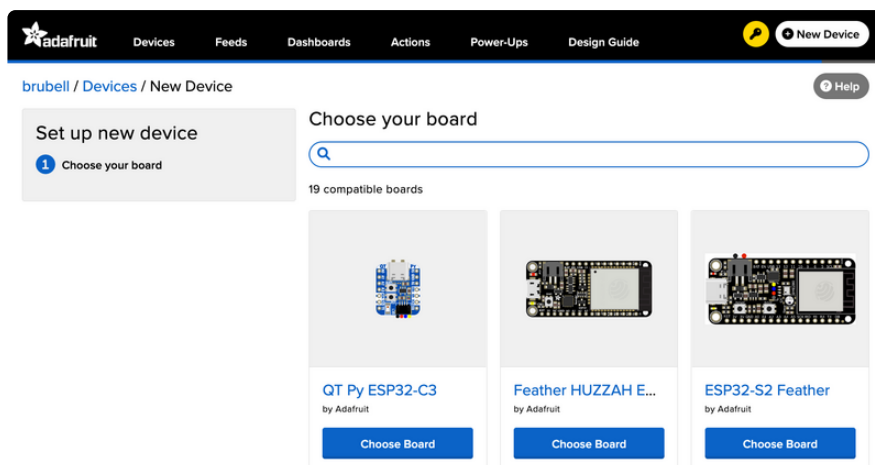
You will need an Adafruit IO account to use WipperSnapper on your board. If you do not already have one, head over to io.adafruit.com () to create a free account.

Add a New Device to Adafruit IO

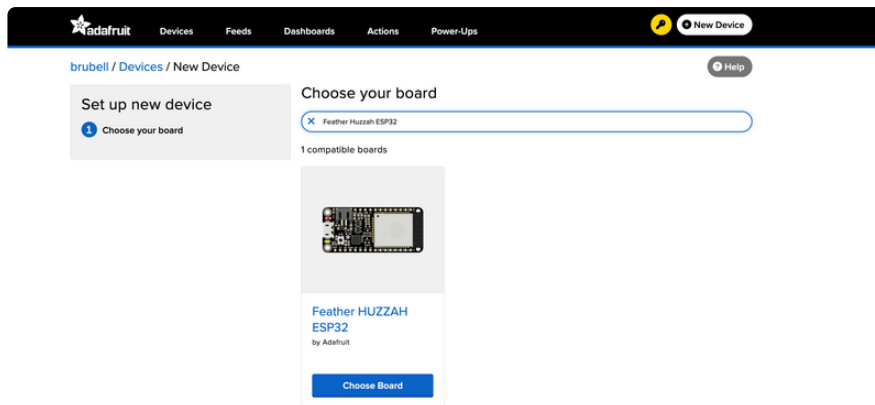
Log into your [Adafruit IO](#) () account. Click the New Device button at the top of the page.



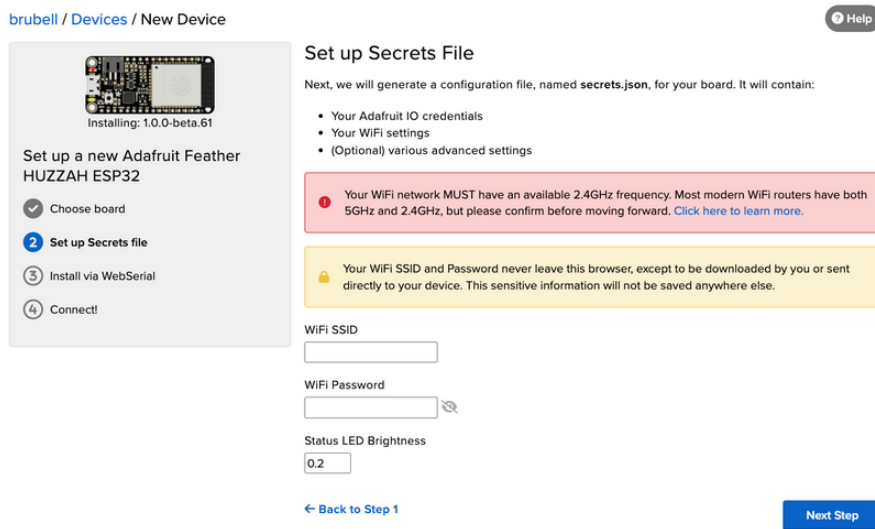
After clicking New Device, you should be on the board selector page. This page displays every board that is compatible with the WipperSnapper firmware.



In the board selector page's search bar, search for the ESP32-S2 Feather. 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.



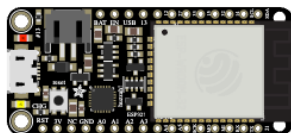
If the installation was successful, a popover should appear displaying that your board has successfully been detected by Adafruit IO.

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

New Device Detected!



You have successfully connected a new **feather-esp32** device to Adafruit IO. It is already set up and submitting data. You can name the device here, and set up components on the device page.



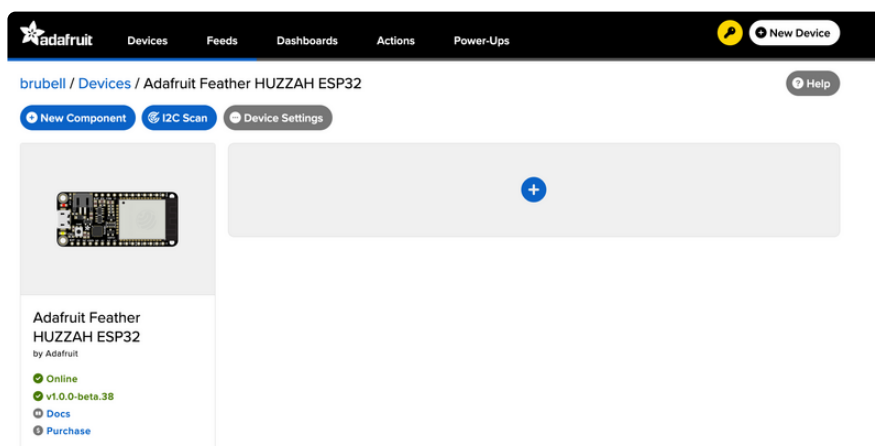
Device Name

Firmware Version:  v1.0.0-beta.38

[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> (), 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> (). Make sure to click "Contact Adafruit IO Support" and select "There is an issue with WipperSnapper. Something is broken!"

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.](#) ()

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.](#) ()

"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 \(\)](#) 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](#)

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

[Upload Arduino "Blink" Sketch](#)

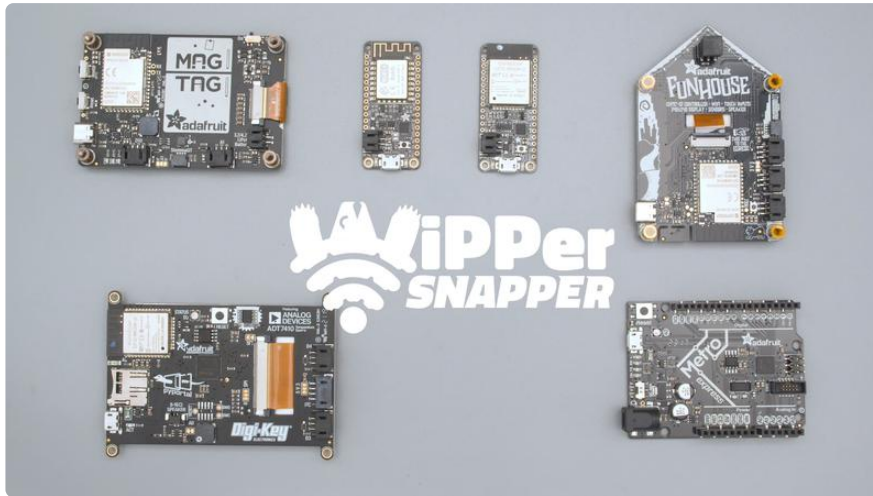
Uploading this sketch will overwrite WipperSnapper. If you want to re-install WipperSnapper, follow the instructions at the top of this page.

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.

Note: There is no "Factory Reset" binary file available for this board. You will need to upload the "Blink" sketch to your board per the instructions above.

WipperSnapper Essentials



You've installed WipperSnapper firmware on your board and connected it to Adafruit IO. Next, let's learn how to use Adafruit IO!

The Adafruit IO supports a large number of components. Components are physical parts such as buttons, switches, sensors, servos, LEDs, RGB LEDs, and more.

The following pages will get you up and running with WipperSnapper as you interact with your board's LED, read the value of a push button, send the value of an I2C sensor to the internet, and wirelessly control colorful LEDs.

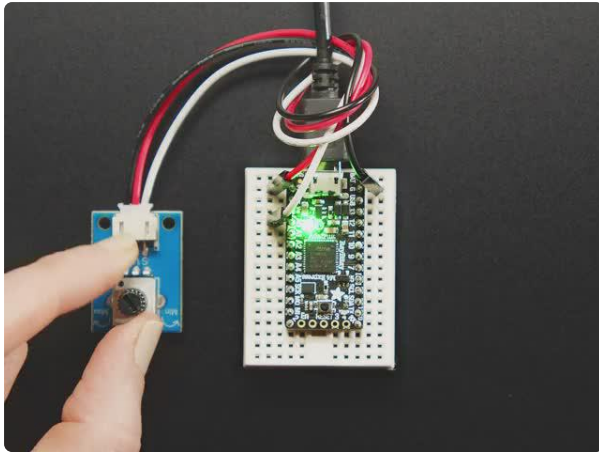
Parts

The following parts are required to complete the WipperSnapper essentials pages for this board:



[Tactile Switch Buttons \(12mm square, 6mm tall\) x 10 pack](#)

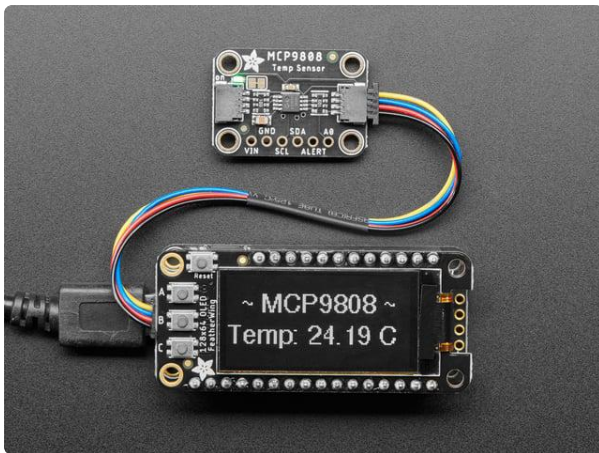
Medium-sized clicky momentary switches are standard input "buttons" on electronic projects. These work best in a PCB but <https://www.adafruit.com/product/1119>



STEMMA Wired Potentiometer Breakout Board - 10K ohm Linear

For the easiest way possible to measure twists, turn to this STEMMA potentiometer breakout (ha!). This plug-n-play pot comes with a JST-PH 2mm connector and a matching

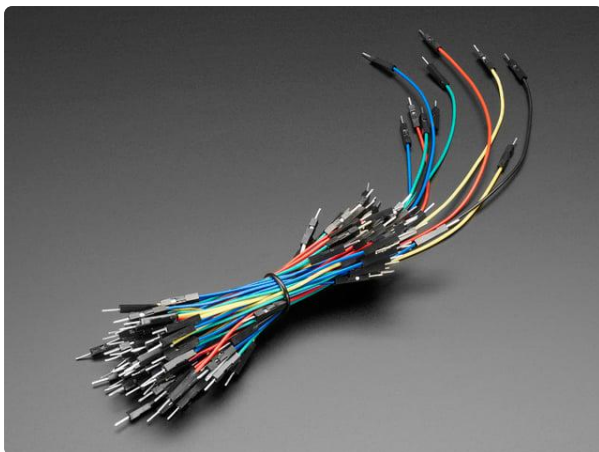
<https://www.adafruit.com/product/4493>



Adafruit MCP9808 High Accuracy I2C Temperature Sensor Breakout

The MCP9808 digital temperature sensor is one of the more accurate/precise we've ever seen, with a typical accuracy of $\pm 0.25^{\circ}\text{C}$ over the sensor's -40°C to...

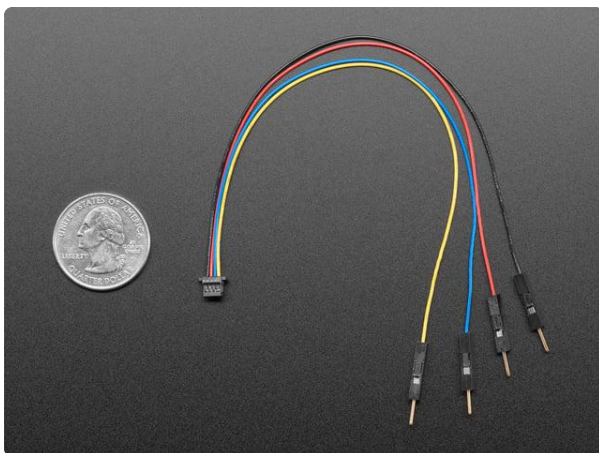
<https://www.adafruit.com/product/5027>



Breadboarding wire bundle

75 flexible stranded core wires with stiff ends molded on in red, orange, yellow, green, blue, brown, black and white. These are a major improvement over the "box of bent..."

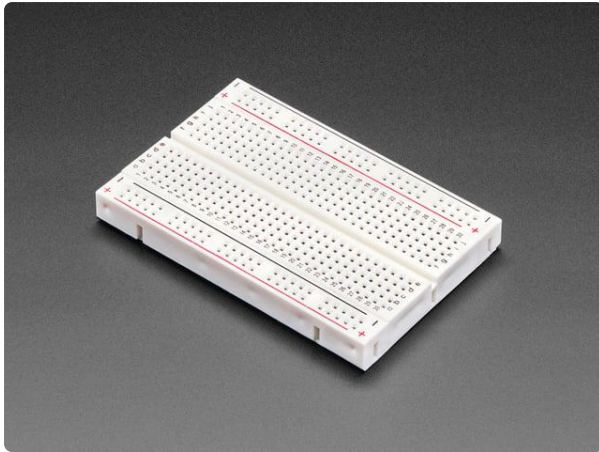
<https://www.adafruit.com/product/153>



STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable

This 4-wire cable is a little over 150mm / 6" long and fitted with JST-SH female 4-pin connectors on one end and premium Dupont male headers on the other. Compared with the...

<https://www.adafruit.com/product/4209>



Half Sized Premium Breadboard - 400 Tie Points

This is a cute, half-size breadboard with 400 tie points, good for small projects. It's 3.25" x 2.2" / 8.3cm x 5.5cm with a standard double-strip in the...

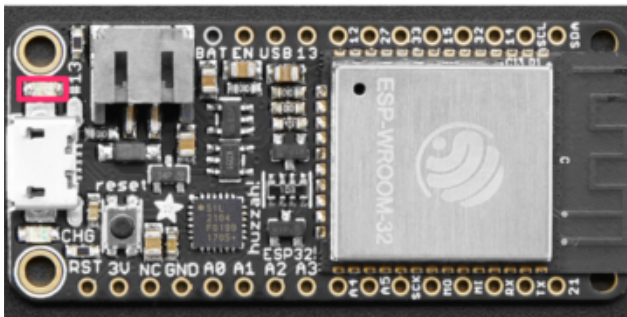
<https://www.adafruit.com/product/64>

LED Blink

In this demo, we show controlling an LED from Adafruit IO. But the same kind of control can be used for relays, lights, motors, or solenoids.

One of the first programs you typically write to get used to embedded programming is a sketch that repeatably blinks an LED. IoT projects are wireless, so after completing this section, you'll be able to turn on (or off) the LED built into your board from anywhere in the world.

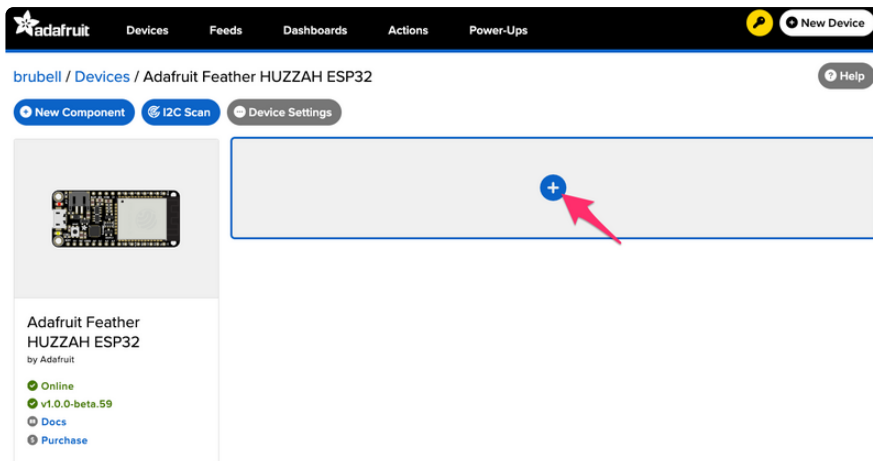
Where is the LED on my board?



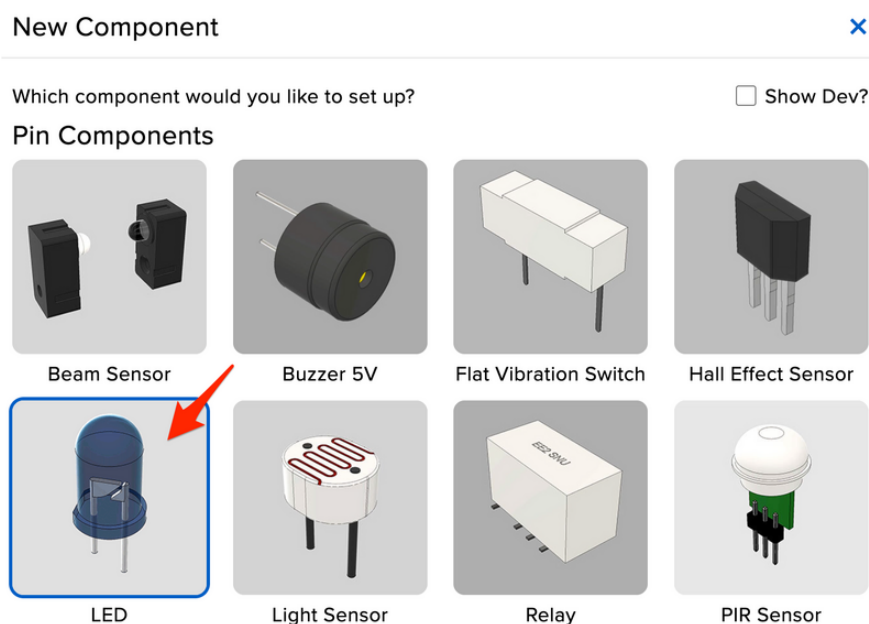
The Adafruit Huzzah 32's LED (highlighted in red) is located next to the micro-USB connector.

Create a LED Component on Adafruit IO

On the device page, click the New Component (or "+") button to open the component picker.

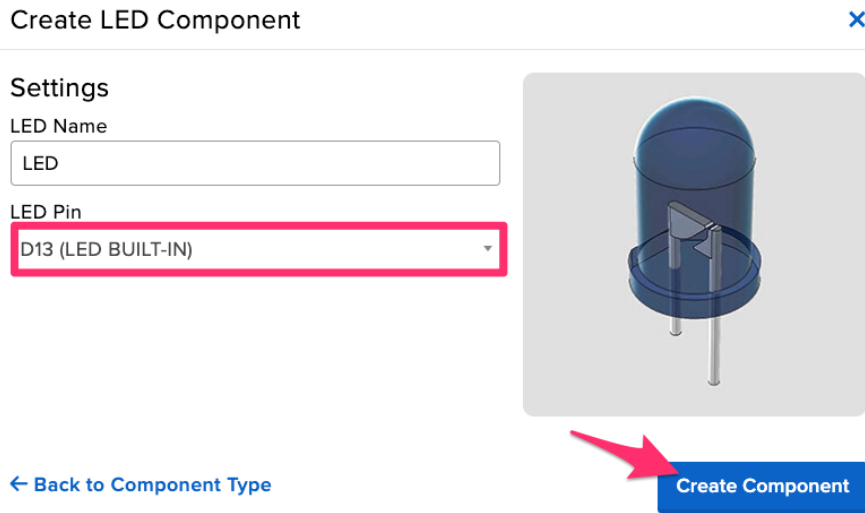


From the component picker, select the LED.



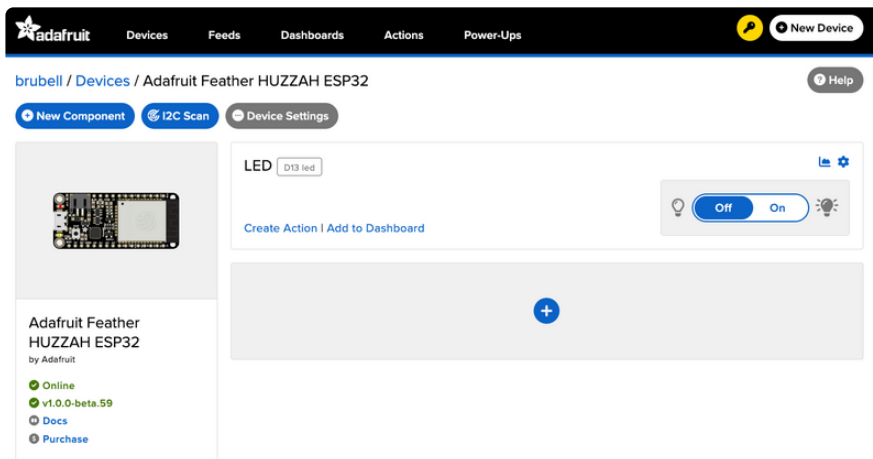
On the Create LED Component form, the board's LED pin is pre-selected.

Click Create Component.



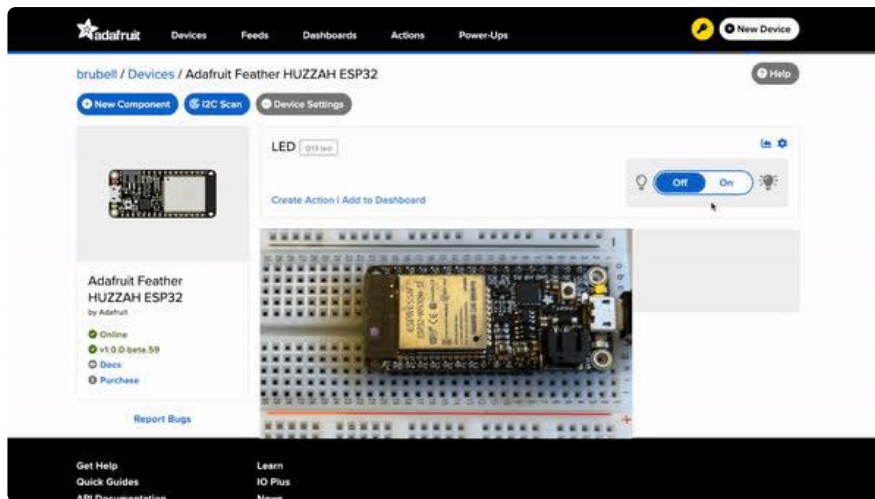
Behind the scenes, Adafruit IO sends a command to your board running WipperSnapper telling it to configure "LED Pin" as a digital output.

Your board's page on Adafruit IO shows a new LED component.



Usage

On the board page, toggle the LED component by clicking the toggle switch. This should turn your board's built-in LED on or off.



Read a Push-button

In this demo, we show reading the state of a push-button from WipperSnapper. But the same kind of control can be used for reading switches, break beam sensors, and other digital sensors.

You can configure a board running WipperSnapper to read data from standard input buttons, switches, or digital sensors, and send the value to Adafruit IO.

From Adafruit IO, you will configure one of the pushbuttons on your board as a push button component. Then, when the button is pressed (or released), a value will be published to Adafruit IO.

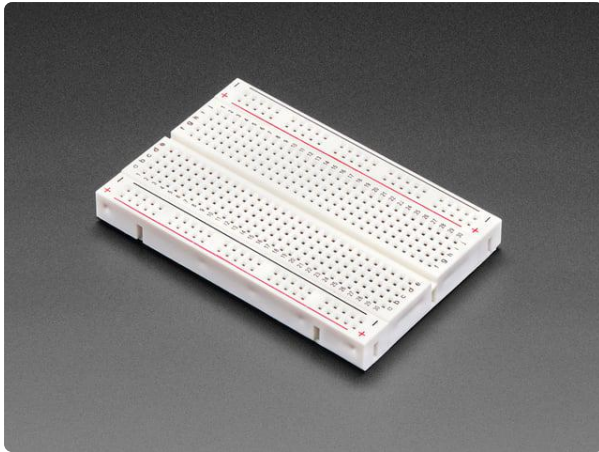
Parts

The following parts are required to complete this page.



[Tactile Switch Buttons \(12mm square, 6mm tall\) x 10 pack](#)

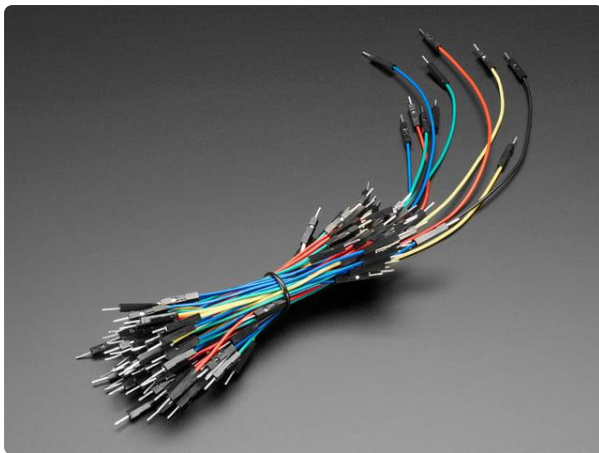
Medium-sized clicky momentary switches are standard input "buttons" on electronic projects. These work best in a PCB but <https://www.adafruit.com/product/1119>



Half Sized Premium Breadboard - 400 Tie Points

This is a cute, half-size breadboard with 400 tie points, good for small projects. It's 3.25" x 2.2" / 8.3cm x 5.5cm with a standard double-strip in the...

<https://www.adafruit.com/product/64>

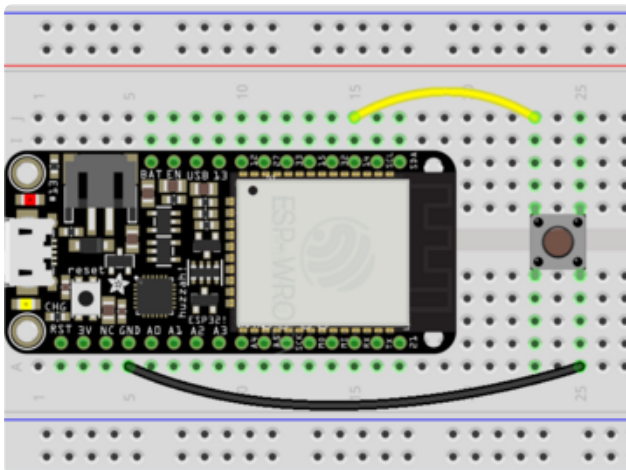


Breadboarding wire bundle

75 flexible stranded core wires with stiff ends molded on in red, orange, yellow, green, blue, brown, black and white. These are a major improvement over the "box of bent..."

<https://www.adafruit.com/product/153>

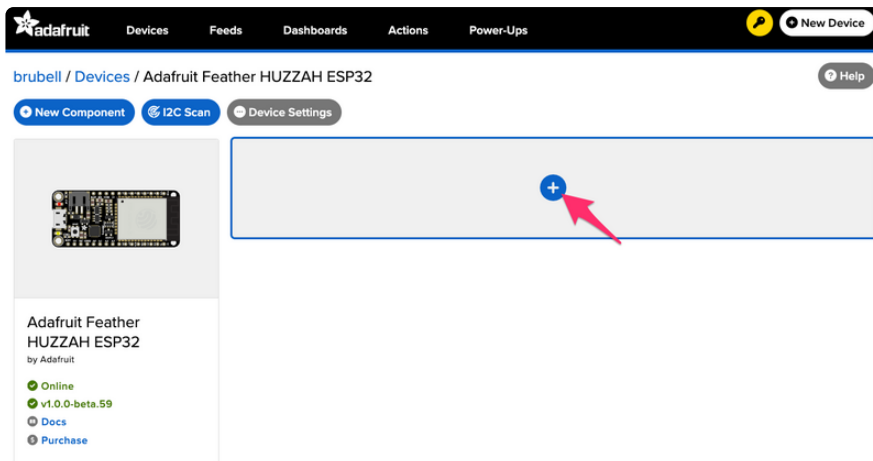
Wiring



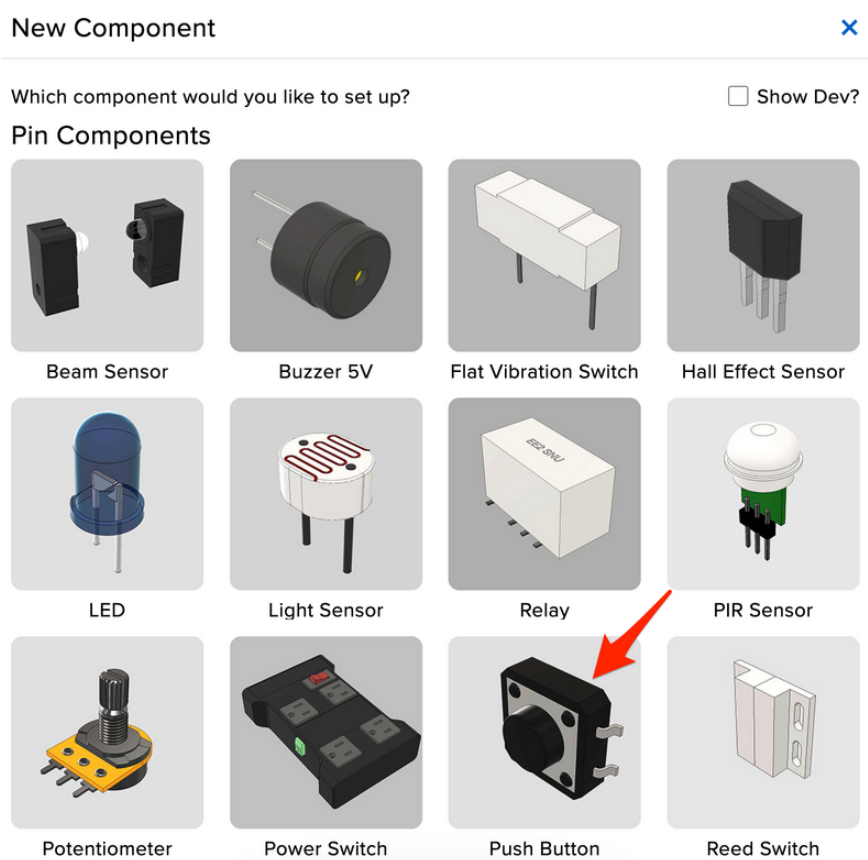
Feather GND to Push Button
Feather GPIO D14 to Push Button
Note - we are not using a resistor part! We will be configuring the Feather ESP32 to use its internal resistor instead.

Create a Push-button Component on Adafruit IO

On the device page, click the New Component (or "+") button to open the component picker.



From the component picker, select the Push Button.



The "Create Push Button Component" form presents you with options for configuring the push button.

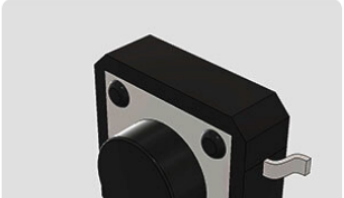
Start by selecting the board's pin connected to the push button.

Create Push Button Component ✕

Settings

Push Button Name

Push Button Pin



The Return Interval dictates how frequently the value of the push-button will be sent from the board to Adafruit IO.

For this example, you will configure the push button value to be only sent when the value changes (i.e.: when it's either pressed or depressed).

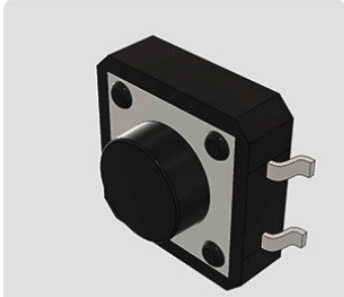
Create Push Button Component ✕

Settings

Push Button Name

Push Button Pin

Return Interval
 On Change
 Periodically



Check the Specify Pin Pull Direction checkbox.

Select Pull Up to turn on the internal pull-up resistor.

Create Push Button Component ✕

Settings

Push Button Name


Push Button Pin

Return Interval
 On Change
 Periodically

Specify Pin Pull Direction? ↖

Pull Up ↖
 Pull Down

[← Back to Component Type](#) Create Component



Make sure the form settings look like the following screenshot. Then, click Create Component.

Create Push Button Component



Settings

Push Button Name

Push Button Pin

Return Interval

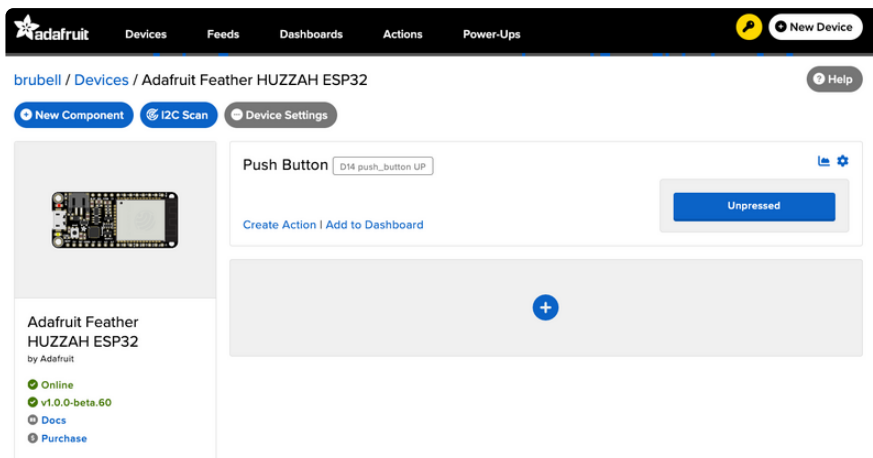
 On Change Periodically Specify Pin Pull Direction? Pull Up Pull Down

[← Back to Component Type](#)

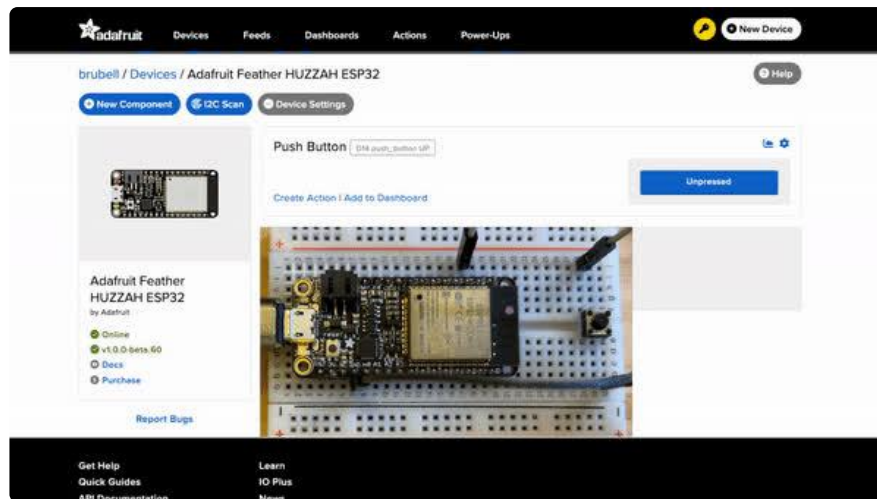
[Create Component](#)

Adafruit IO sends a command to your WipperSnapper board, telling it to configure the GPIO pin you selected to behave as a digital input pin and to enable it to pull up the internal resistor.

Your board page should also show the new push-button component.



Push the button on your board to change the value of the push-button component on Adafruit IO.



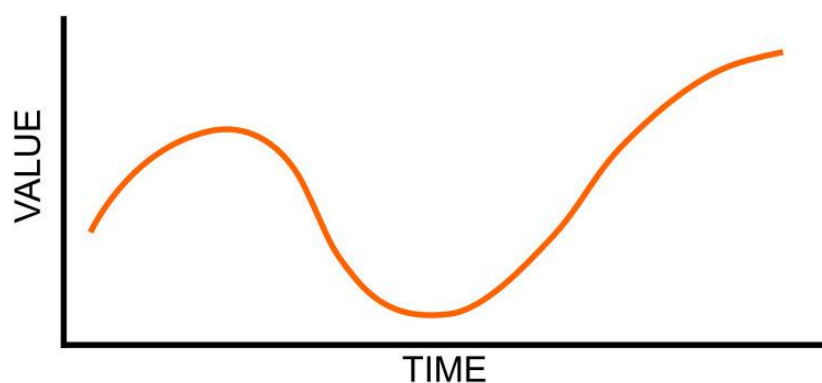
Analog Input

Your microcontroller board has both digital and analog signal capabilities. Some pins are analog, some are digital, and some are capable of both. Check the Pinouts page in this guide for details about your board.

Analog signals are different from digital signals in that they can be any voltage and can vary continuously and smoothly between voltages. An analog signal is like a dimmer switch on a light, whereas a digital signal is like a simple on/off switch.

Digital signals only can ever have two states, they are either are on (high logic level voltage like 3.3V) or off (low logic level voltage like 0V / ground).

By contrast, analog signals can be any voltage in-between on and off, such as 1.8V or 0.001V or 2.98V and so on.



Analog signals are continuous values which means they can be an infinite number of different voltages. Think of analog signals like a floating point or fractional number,

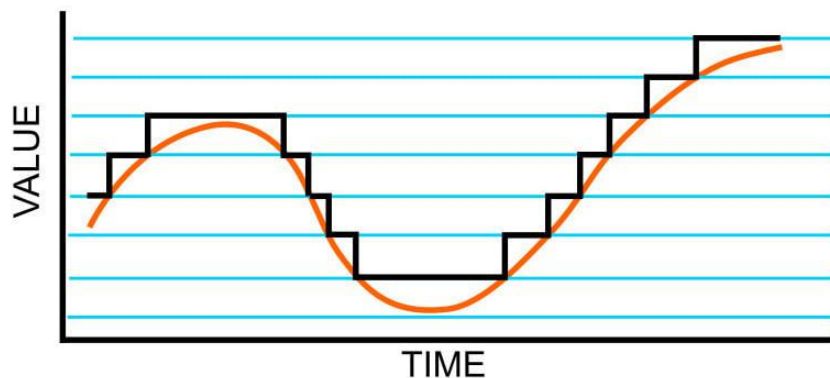
they can smoothly transiting to any in-between value like 1.8V, 1.81V, 1.801V, 1.8001V, 1.80001V and so forth to infinity.

Many devices use analog signals, in particular sensors typically output an analog signal or voltage that varies based on something being sensed like light, heat, humidity, etc.

Analog to Digital Converter (ADC)

An analog-to-digital-converter, or ADC, is the key to reading analog signals and voltages with a microcontroller. An ADC is a device that reads the voltage of an analog signal and converts it into a digital, or numeric, value. The microcontroller can't read analog signals directly, so the analog signal is first converted into a numeric value by the ADC.

The black line below shows a digital signal over time, and the red line shows the converted analog signal over the same amount of time.

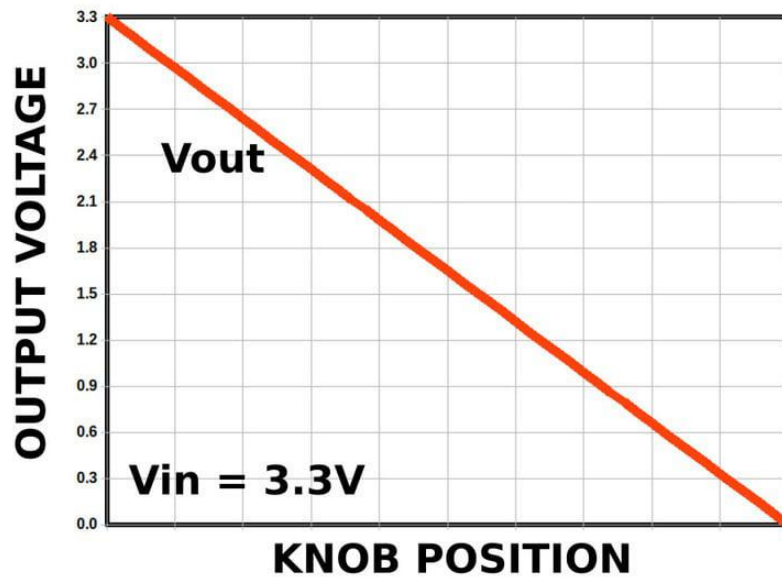


Once that analog signal has been converted by the ADC, the microcontroller can use those digital values any way you like!

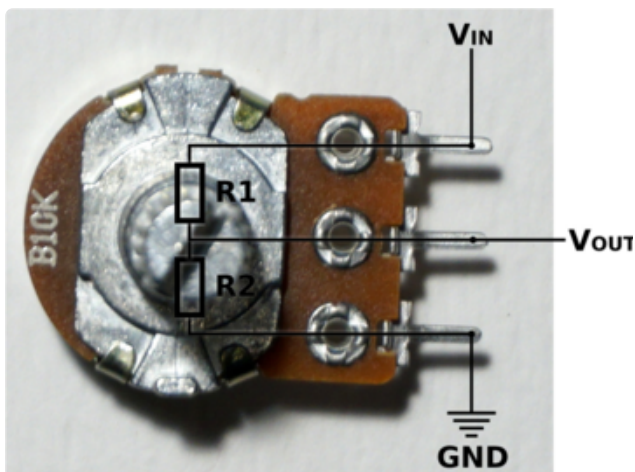
Potentiometers

A potentiometer is a small variable resistor that you can twist a knob or shaft to change its resistance. It has three pins. By twisting the knob on the potentiometer you can change the resistance of the middle pin (called the wiper) to be anywhere within the range of resistance of the potentiometer.

By wiring the potentiometer to your board in a special way (called a voltage divider) you can turn the change in resistance into a change in voltage that your board's analog to digital converter can read.



To wire up a potentiometer as a voltage divider:

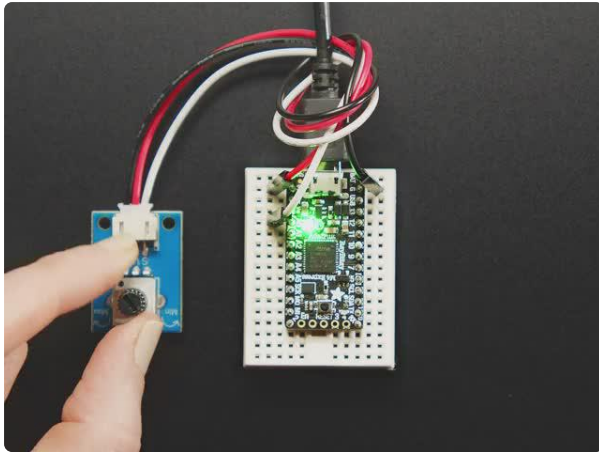


Connect one outside pin to ground
 Connect the other outside pin to voltage in
 (e.g. 3.3V)
 Connect the middle pin to an analog pin
 (e.g. A0)

Hardware

In addition to your microcontroller board, you will need the following hardware to follow along with this example.

Potentiometer



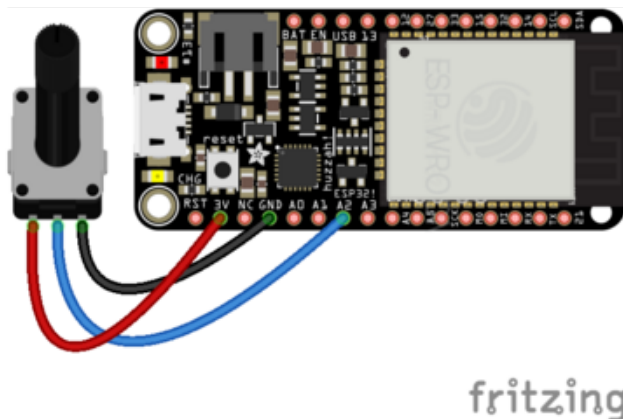
STEMMA Wired Potentiometer Breakout Board - 10K ohm Linear

For the easiest way possible to measure twists, turn to this STEMMA potentiometer breakout (ha!). This plug-n-play pot comes with a JST-PH 2mm connector and a matching

<https://www.adafruit.com/product/4493>

Wire Up the Potentiometer

Connect the potentiometer to your board as follows.



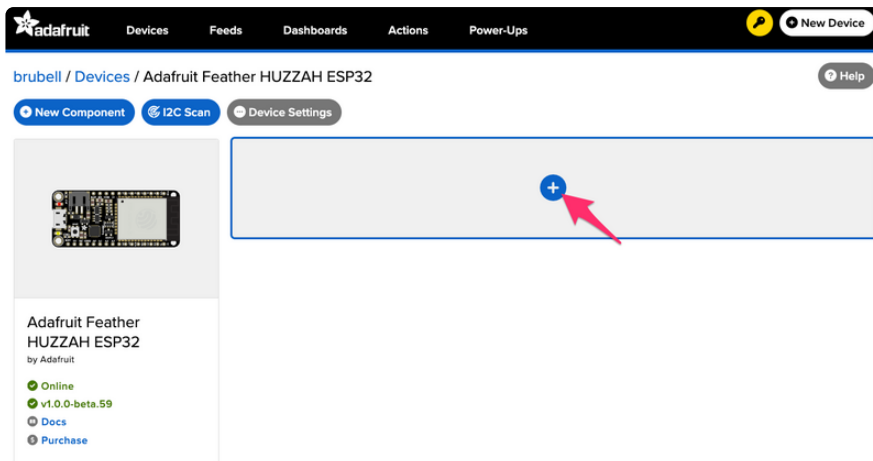
Make the following connections between your ESP32 Feather and the Potentiometer:

Feather 3.3V to potentiometer left pin
Feather A2 to potentiometer middle pin
Feather GND to potentiometer right pin
Note: On WipperSnapper, the ESP32's ADC1 pins are unusable for analog input since WiFi is constantly running.

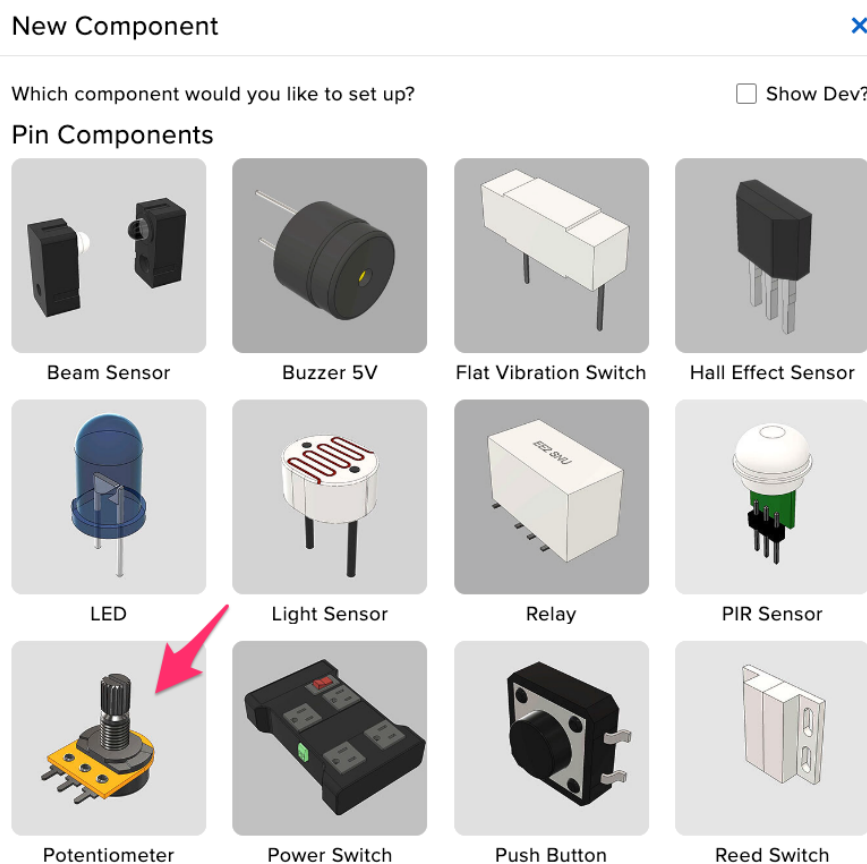
For information on what Analog Input pins can (and can't) be used on this board, click [here >>> \(\)](#)

Create a Potentiometer Component on Adafruit IO

On the device page, click the New Component (or "+") button to open the component picker.



On the component picker, click the potentiometer.



On the Create Potentiometer Component form:

- Set Potentiometer Pin to A2
- Select "On Change" as the Return Interval
- Select Raw Analog Value as the Return Type

Then, click Create Component

Create Potentiometer Component ✕

Settings


Potentiometer Name

Potentiometer Pin

Return Interval
 On Change
 Periodically

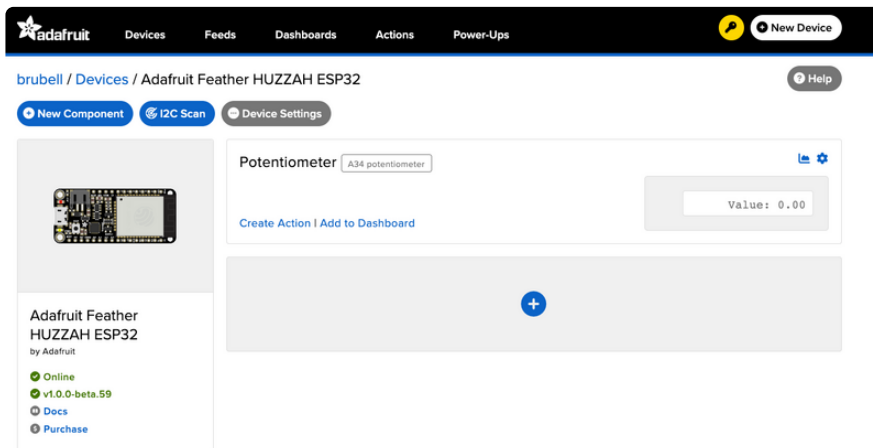
Return Type
 Raw Analog Value
 Voltage

[← Back to Component Type](#)



Create Component

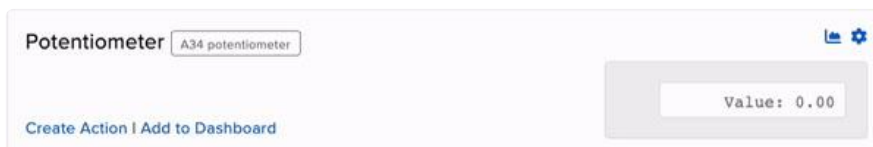
The potentiometer component appears on your board page! Next, learning to read values from it.



The screenshot shows the Adafruit web interface for a device page. At the top, there's a navigation bar with 'adafruit' logo and links for 'Devices', 'Feeds', 'Dashboards', 'Actions', and 'Power-Ups'. Below that, the device name 'brubell / Devices / Adafruit Feather HUZZAH ESP32' is displayed. A 'New Device' button is visible. The main content area shows a 'Potentiometer' component added to the board. The component is labeled 'A34 potentiometer' and has a 'Value: 0.00' display. There are buttons for 'New Component', 'I2C Scan', and 'Device Settings'. A sidebar on the left shows the device details for 'Adafruit Feather HUZZAH ESP32' by Adafruit, including status (Online), version (v1.0.0-beta.59), and links for 'Devs' and 'Purchase'.

Read Analog Pin Values

Rotate the potentiometer to see the value change.



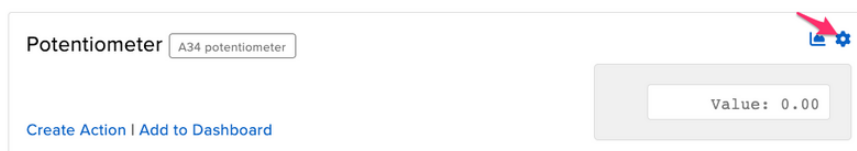
This is a close-up of the potentiometer component from the previous screenshot. It shows the label 'Potentiometer' and 'A34 potentiometer'. Below the label is a button that says 'Create Action | Add to Dashboard'. To the right of the component is a display box showing 'Value: 0.00'.

What do these values mean?

WipperSnapper reports ADC "raw values" as 16-bit unsigned integer values. Your potentiometer will read between 0 (twisting the pot to the leftmost position) and 65535 (twisting the pot to the rightmost position).

Read Analog Pin Voltage Values

You can update the potentiometer component (or any analog pin component in WipperSnapper) to report values in Volts. To do this, on the right-hand side of the potentiometer component, click the cog button.

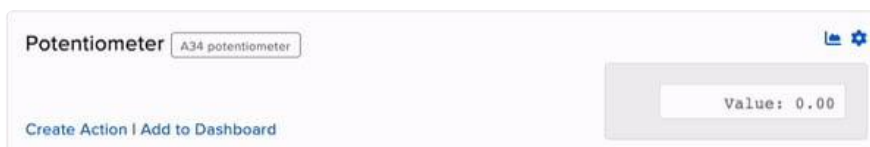


Under Return Type, click Voltage.

Click Update Component to send the updated component settings to your board running WipperSnapper.



Now, twist the potentiometer to see the value reported to Adafruit IO in Volts!



I2C Sensor

While this page uses the "MCP9808 High Accuracy I2C Temperature Sensor Breakout", the process for adding an I2C sensor to your board running WipperSnapper is similar for all I2C sensors.

Inter-Integrated Circuit, aka I2C, is a two-wire protocol for connecting sensors and "devices" to a microcontroller. A large number of sensors, including the ones sold by Adafruit, use I2C to communicate.

Typically, using I2C with a microcontroller involves programming. Adafruit IO and WipperSnapper let you configure a microcontroller to read data from an I2C sensor and publish that data to the internet without writing code.

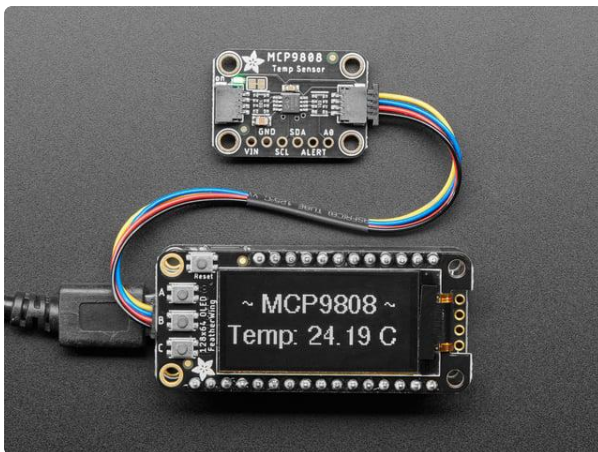
The WipperSnapper firmware supports a number of I2C sensors, [viewable in list format here \(\)](#).

- If you do not see the I2C sensor you're attempting to use with WipperSnapper, [Adafruit has a guide on adding a component to Adafruit IO WipperSnapper here \(\)](#).

On this page, you'll learn how to wire up an I2C sensor to your board. Then, you'll create a new component on Adafruit IO for your I2C sensor and send the sensor values to Adafruit IO. Finally, you'll learn how to locate, interpret, and download the data produced by your sensors.

Parts

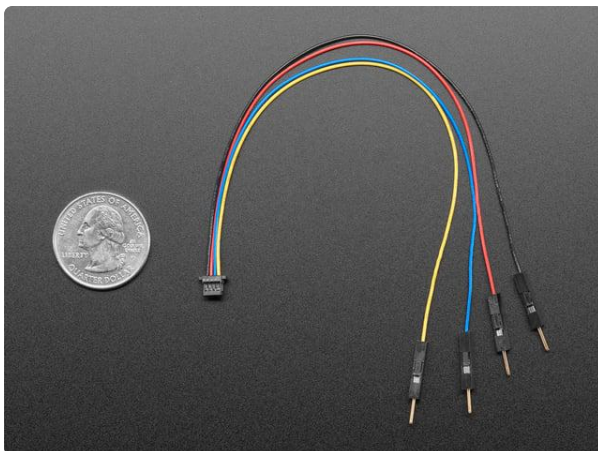
You will need the following parts to complete this page:



[Adafruit MCP9808 High Accuracy I2C Temperature Sensor Breakout](#)

The MCP9808 digital temperature sensor is one of the more accurate/precise we've ever seen, with a typical accuracy of $\pm 0.25^{\circ}\text{C}$ over the sensor's -40°C to...

<https://www.adafruit.com/product/5027>

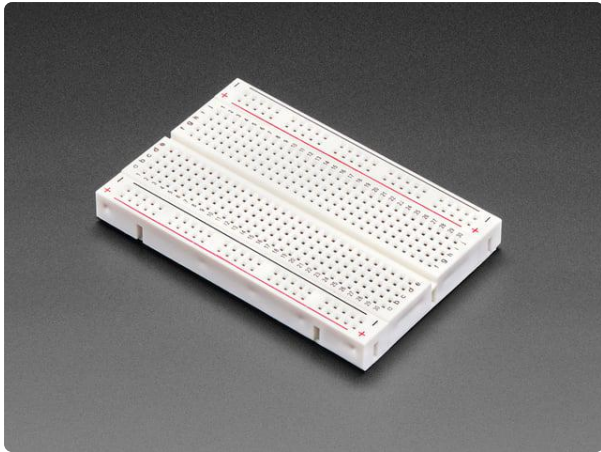


[STEMMA QT / Qwiic JST SH 4-pin to Premium Male Headers Cable](#)

This 4-wire cable is a little over 150mm / 6" long and fitted with JST-SH female 4-pin connectors on one end and premium Dupont male headers on the other.

Compared with the...

<https://www.adafruit.com/product/4209>

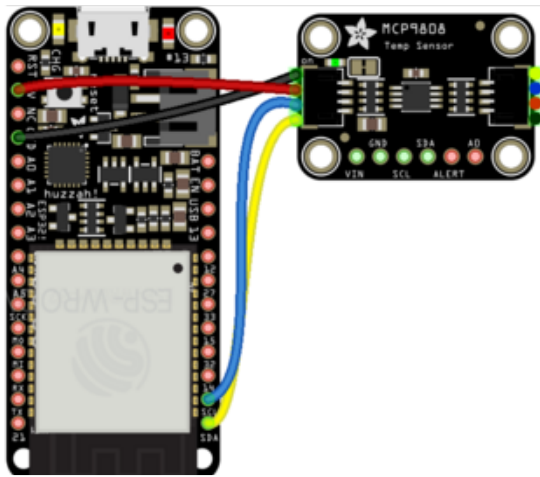


Half Sized Premium Breadboard - 400 Tie Points

This is a cute, half-size breadboard with 400 tie points, good for small projects. It's 3.25" x 2.2" / 8.3cm x 5.5cm with a standard double-strip in the...

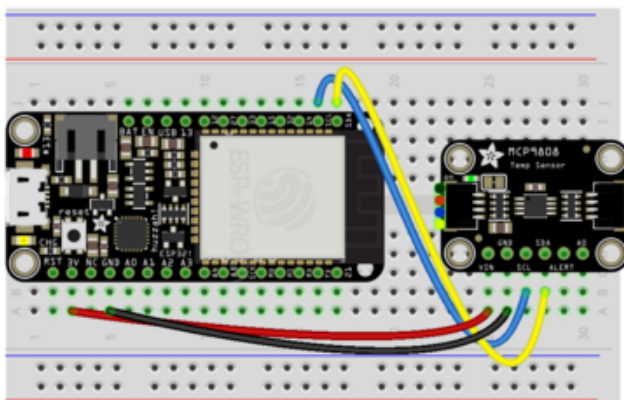
<https://www.adafruit.com/product/64>

Wiring



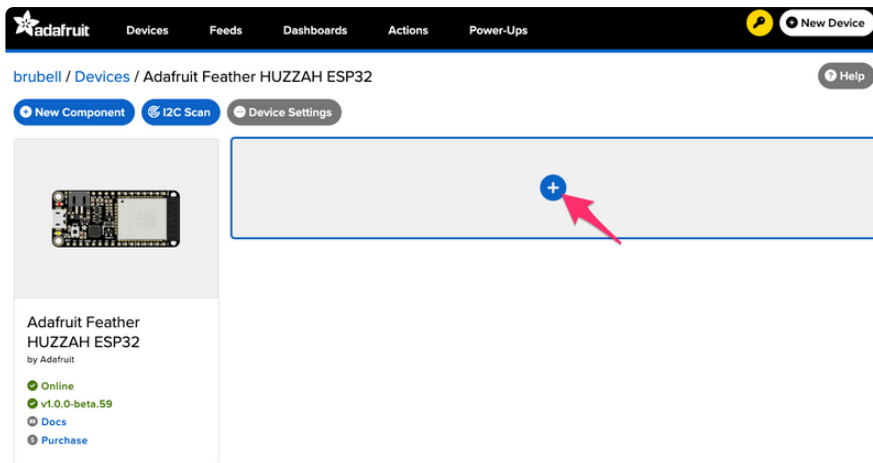
Make the following connections between your Adafruit HUZZAH32 and the MCP9808:

- Board power to MCP9808 VIN
- Board ground to MCP9808 GND
- Board SCL to MCP9808 SCL
- Board SDA to MCP9808 SDA



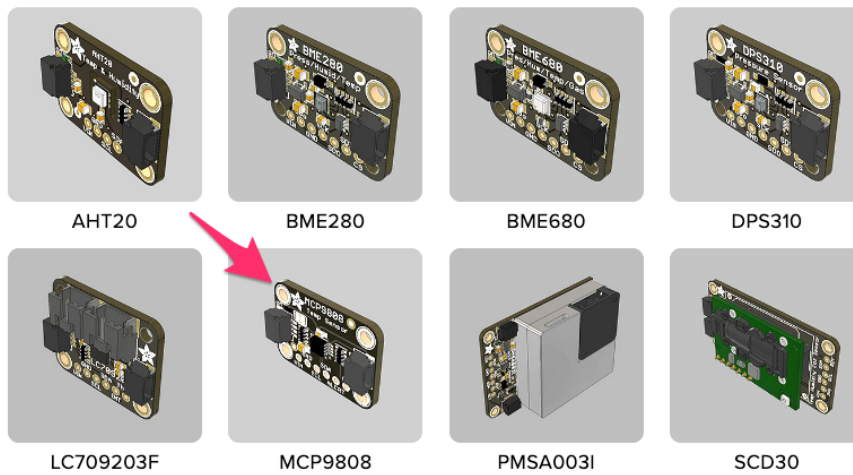
Add an MCP9808 Component

On the device page, click the New Component (or "+") button to open the component picker.



Under the I2C Components header, click MCP9808.

I2C Components



On the component configuration page, the MCP9808's I2C sensor address should be listed along with the sensor's settings.

Create MCP9808 Component ✕

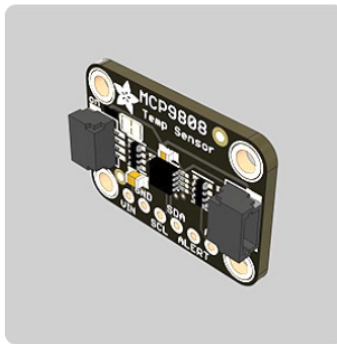
Select I2C Address:

Enable MCP9808: Temperature Sensor (°C)?
Name:

Send Every:

Enable MCP9808: Temperature Sensor (°F)?
Name:

Send Every:



The MCP9808 sensor can measure ambient temperature. This page has individual options for reading the ambient temperature, in either Celsius or Fahrenheit. You may select the readings which are appropriate to your application and region.

The Send Every option is specific to each sensor measurement. This option will tell the board how often it should read from the sensor and send the data to Adafruit IO. Measurements can range from every 30 seconds to every 24 hours.

For this example, set the Send Every interval for both seconds to Every 30 seconds. Click Create Component.

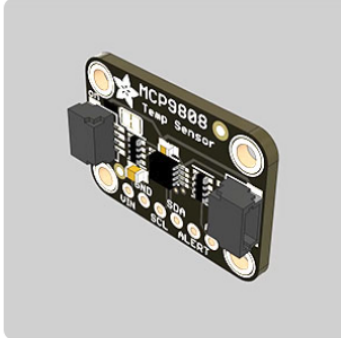
Create MCP9808 Component ✕

Select I2C Address:

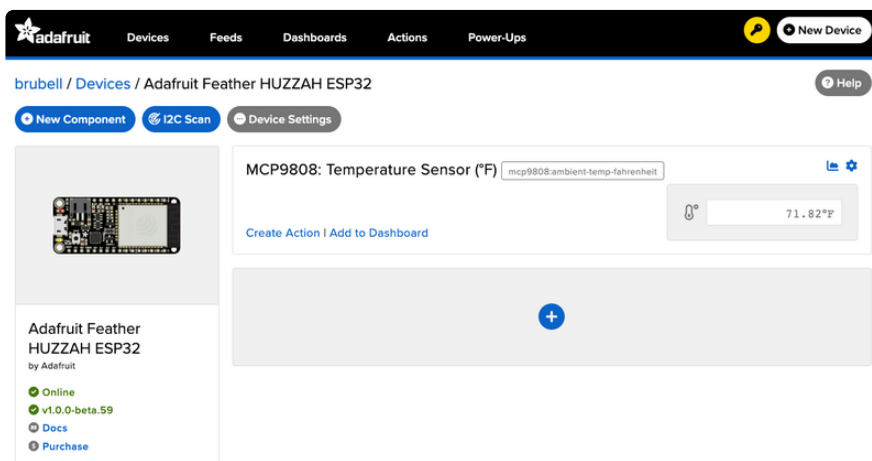
Enable MCP9808: Temperature Sensor (°C)?
 Enable MCP9808: Temperature Sensor (°F)?
Name:

Send Every:

[← Back to Component Type](#) [Create Component](#)



The board page should now show the MCP9808 component you created. After the interval you configured elapses, the WipperSnapper firmware running on your board automatically reads values from the sensor and sends them to Adafruit IO.



Read I2C Sensor Values

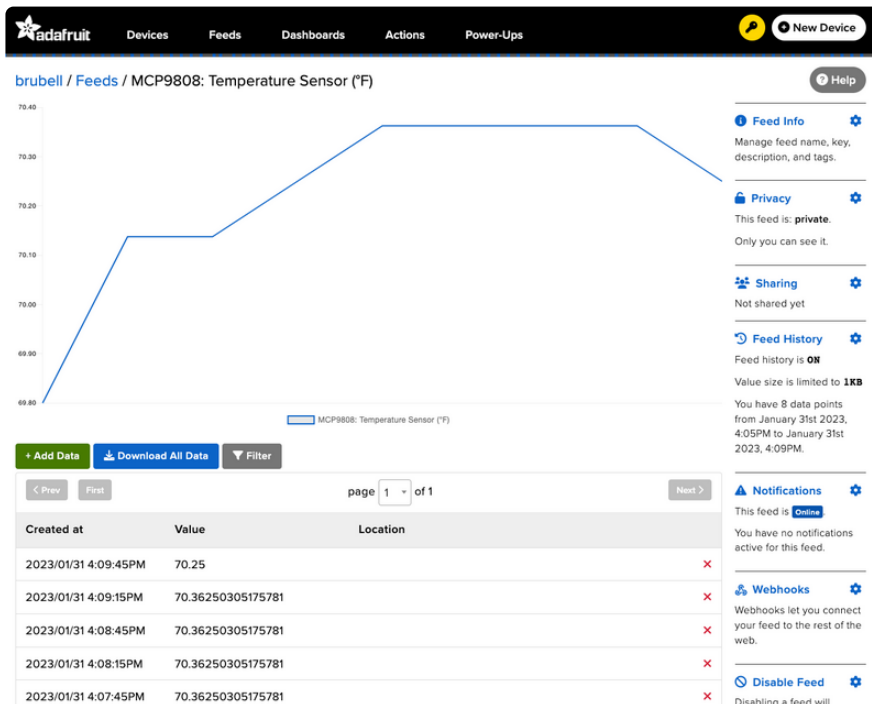
Now to look behind the scenes at a powerful element of using Adafruit IO and WipperSnapper. When a new component is created on Adafruit IO, an [Adafruit IO Feed \(\)](#) is also created. This Feed holds your sensor component values for long-term storage (30 days of storage for Adafruit IO Free and 60 days for Adafruit IO Plus plans).

Aside from holding the values read by a sensor, the component's feed also holds metadata about the data pushed to Adafruit IO. This includes settings for whether the data is public or private, what license the stored sensor data falls under, and a general description of the data.

Next, to look at the sensor temperature feed. To navigate to a component's feed, click on the chart icon in the upper-right-hand corner of the component.



On the component's feed page, you'll each data point read by your sensor and when they were reported to Adafruit IO.



Doing more with your sensor's Adafruit IO Feed

This only scratches the surface of what Adafruit IO Feeds can accomplish for your IoT projects. For a complete overview of Adafruit IO Feeds, including tasks like downloading feed data, sharing a feed, removing erroneous data points from a feed, and more, [head over to the "Adafruit IO Basics: Feed" learning guide](#) ().

ESP32 F.A.Q

Some pins are special about the ESP32 - here's a list of 'notorious' pins to watch for!

- A2 / I34 - this pin is an input only! You can use it as an analog input so we suggest keeping it for that purpose
- A3 / I39 - this pin is an input only! You can use it as an analog input so we suggest keeping it for that purpose
- IO12 - this pin has an internal pulldown, and is used for booting up. We recommend not using it or if you do use it, as an output only so that nothing interferes with the pulldown when the board resets
- A13 / I35 - this pin is not exposed, it is used only for measuring the voltage on the battery. The voltage is divided by 2 so be sure to double it once you've done the analog reading

Why does the yellow CHARGE LED blink while USB powered?

The charging circuit will flash when there is no LiPoly battery plugged in. It's harmless and doesn't mean anything. When a LiPoly battery is connect it will stabilize the charger and will stop flashing

Why can I not read analog inputs once WiFi is initialized?

Due to the design of the ESP32, you can only read analog inputs on ADC #1 once WiFi has started. That means pins on ADC 2 (check the pinouts page) can't be used as analog inputs

Why is Serial.read() not working as expected on ESP32 Breakout?

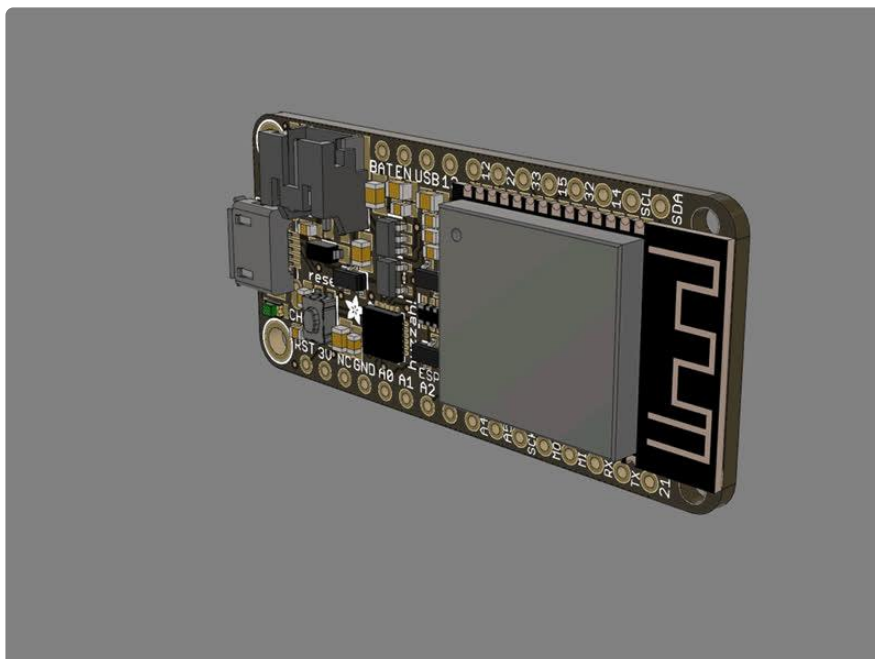
This is a minor design issue with the initial version of the Breakout (does not apply to Feather version). If you are having issues similar to the [discussion here \(\)](#), try the trick of enabling the internal pull-up as [described here \(\)](#).

Downloads

Files

- [ESP32 WROOM32 Datasheet \(\)](#)
- [ESP32 Technical Manual \(\)](#)
- [Don't forget to visit esp32.com for the latest and greatest in ESP32 news, software and gossip! \(\)](#)
- [EagleCAD PCB files on github \(\)](#)
- [Fritzing object in the Adafruit Fritzing library \(\)](#)
- [3D Models on GitHub \(\)](#)
- [PDF for Huzzah32 ESP32 Feather Board Diagram on GitHub \(\)](#)

SVG for Huzzah32 ESP32 Feather Board Diagram



Schematic and Fabrication print

