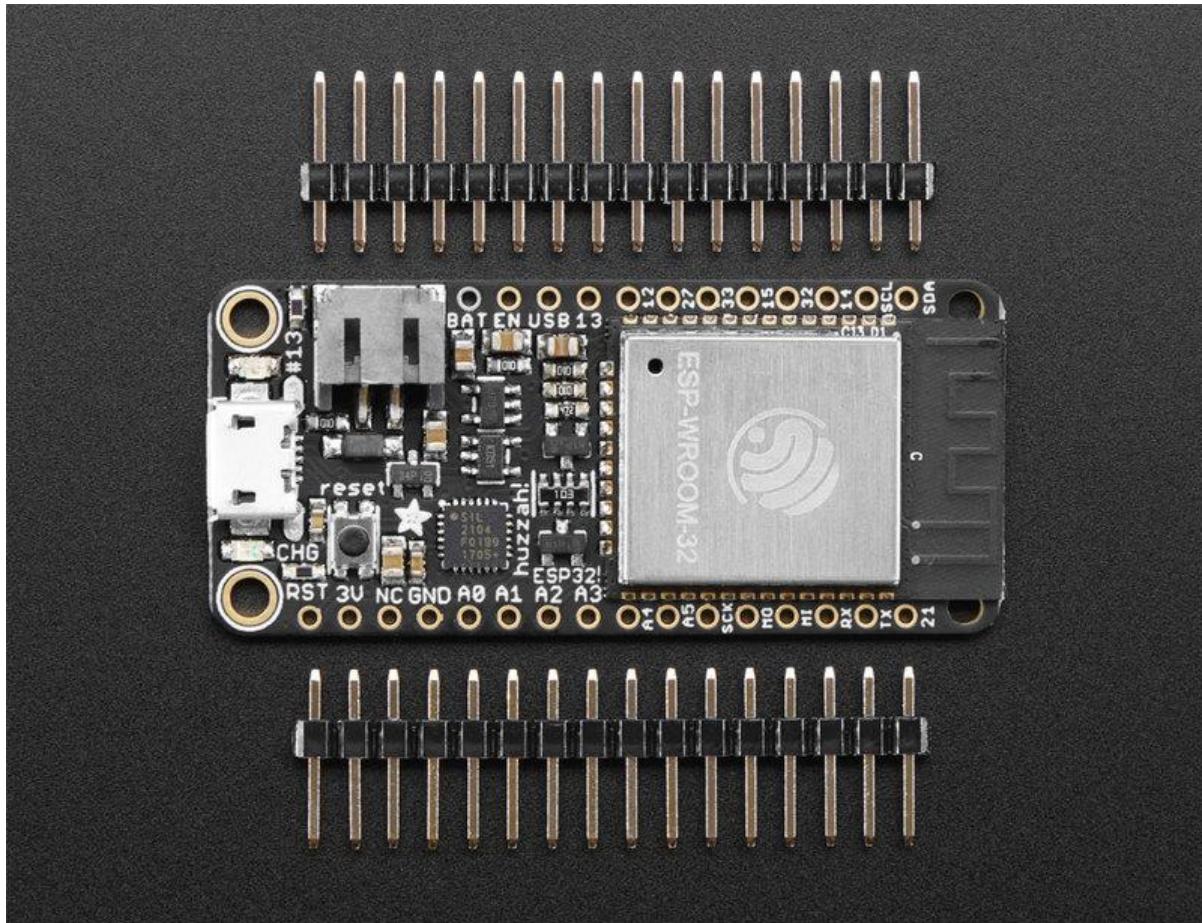




Adafruit HUZZAH32 - ESP32 Feather

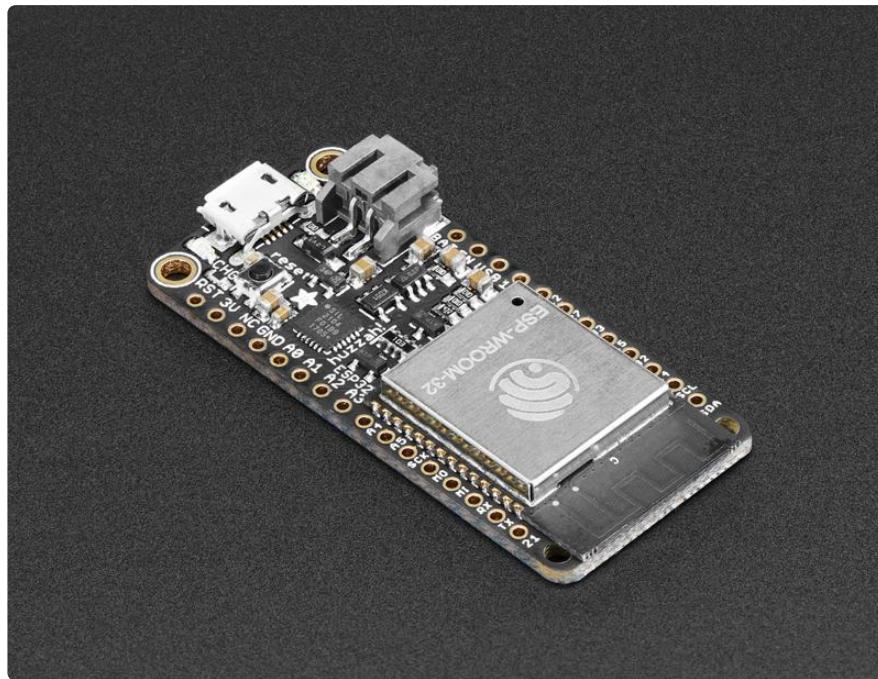
Created by lady ada



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

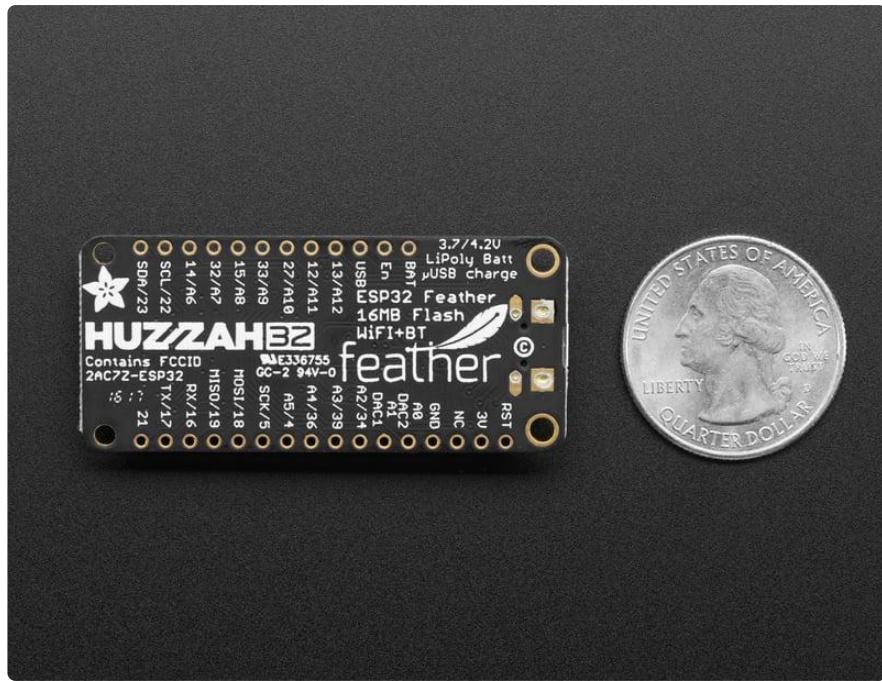
Last updated on 2023-02-23 05:39:18 PM EST

Overview



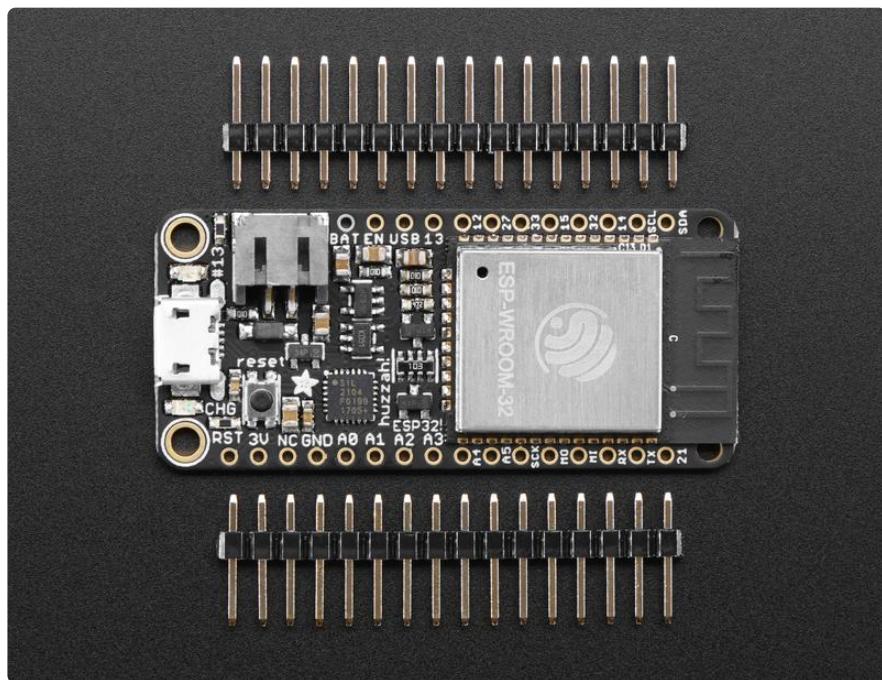
Aww yeah, it's the Feather you have been waiting for! The HUZZAH32 is our ESP32-based Feather, made with the official WROOM32 module. We packed everything you love about Feathers: built in USB-to-Serial converter, automatic bootloader reset, Lithium Ion/Polymer charger, and all the GPIO brought out so 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

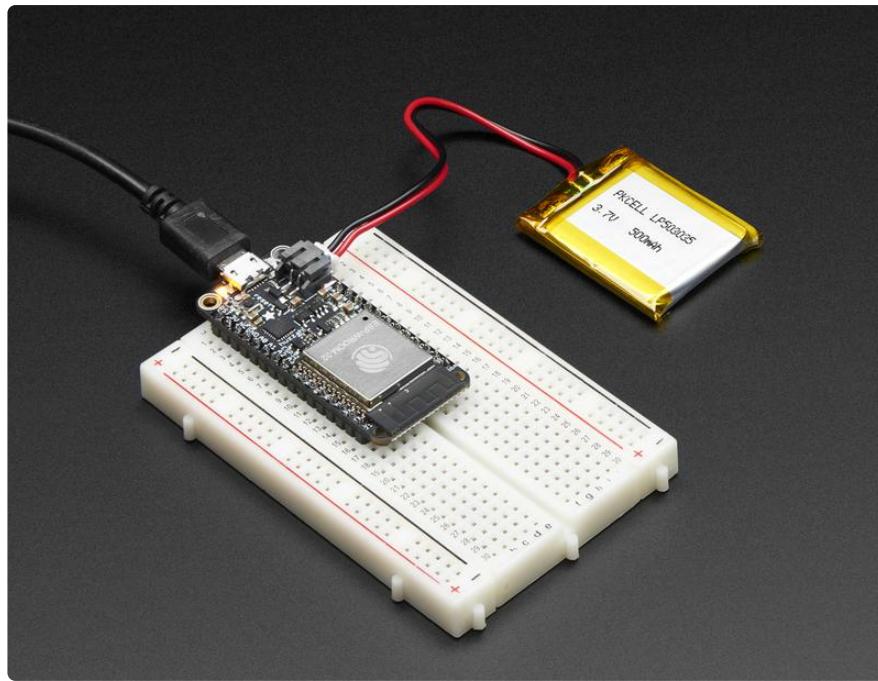
The ESP32 is a perfect upgrade from the ESP8266 that has been so popular. In comparison, the ESP32 has way more GPIO, plenty of analog inputs, two analog outputs, multiple extra peripherals (like a spare UART), two cores so you don't have to yield to the WiFi manager, much higher-speed processor, etc. 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

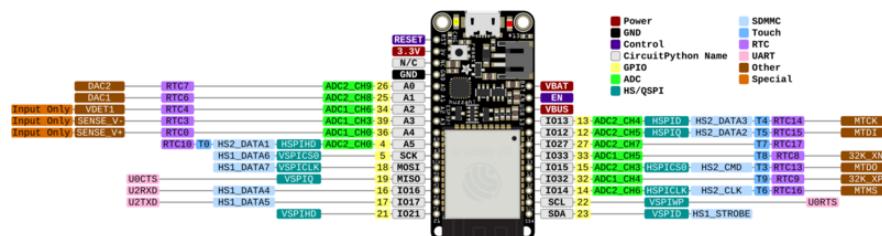


Comes fully assembled and tested, with a USB interface that lets you quickly use it with the Arduino IDE or the low-level ESP32 IDF. We also toss in some header so you can solder it in and plug into a solderless breadboard.

Lipoly battery and USB cable not included (but we do have lots of options in the shop if you'd like!)

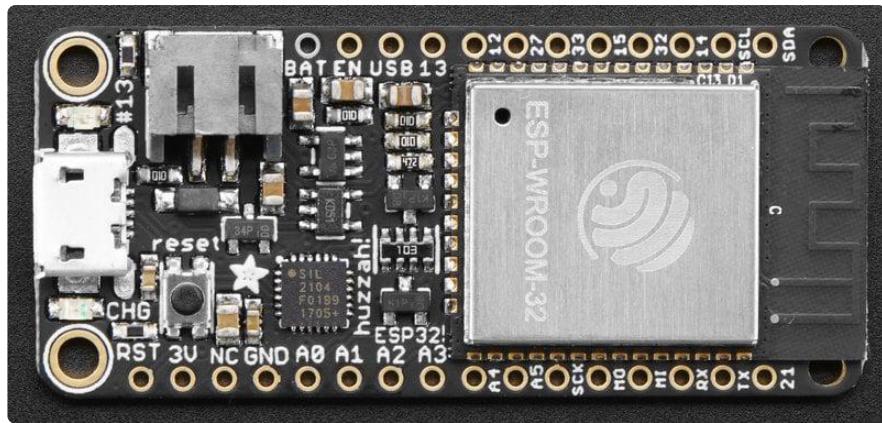
Pinouts

Adafruit HUZZAH32 ESP32 Feather
<http://www.adafruit.com/products/3405>

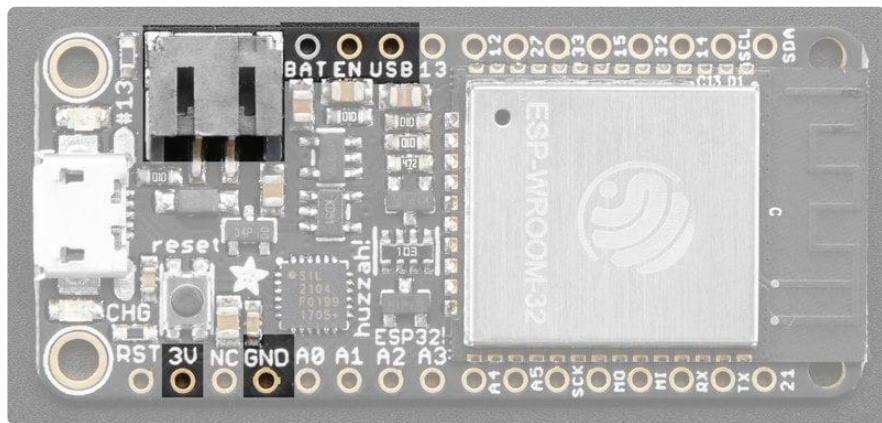


[Click here to view a PDF version of the pinout diagram \(\)](#)

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 BAT 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

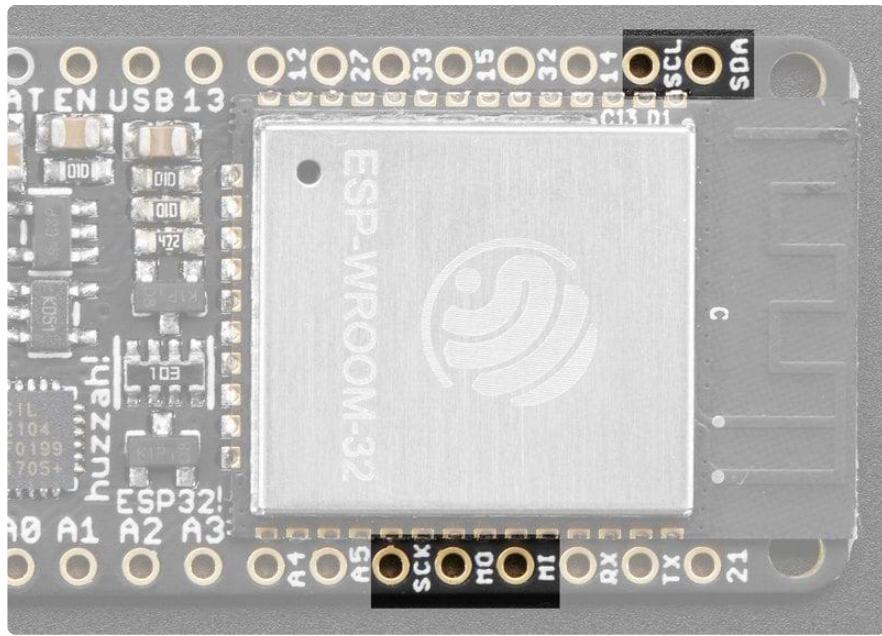
RX and TX are the additional Serial1 pins, and are not connected to the USB/Serial converter. That means you can use them to connect to UART-devices like GPS's, fingerprint sensors, etc.



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

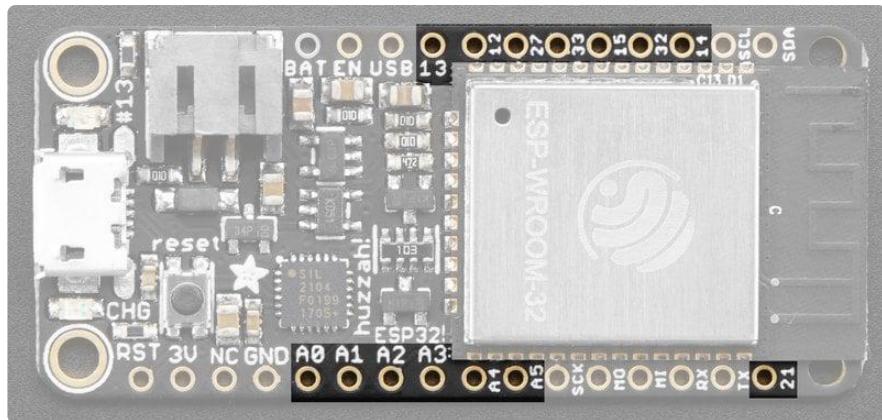
I2C & SPI pins

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!

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.



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.

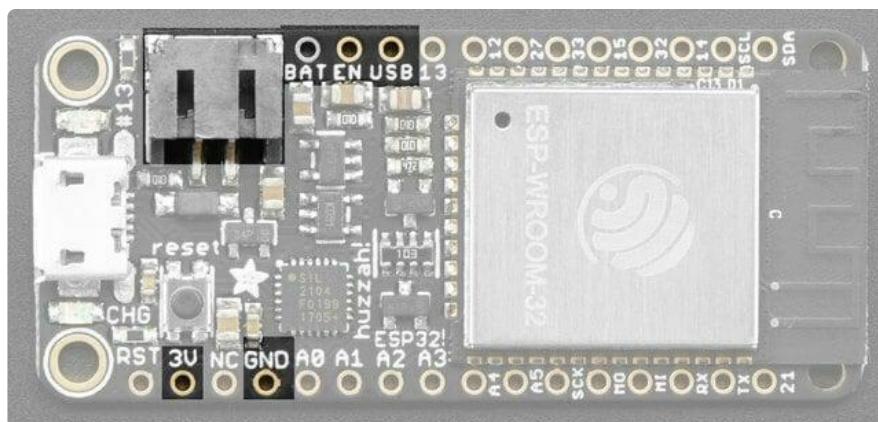
The charge LED is automatically driven by the LiPoly charger circuit. It will try to detect a battery and is expecting one to be attached. If there isn't one it may

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.

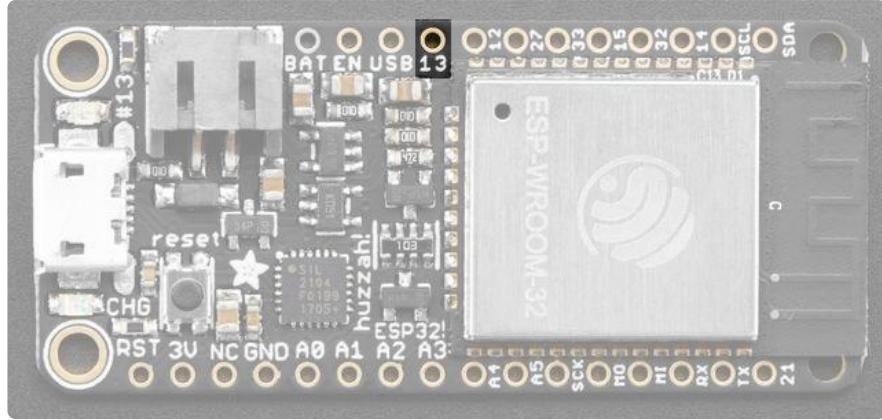
Please budget 250mA for the WROOM32 module. We use this to power the ESP32 which draws about 200mA continuous. The good news is you can put the ESP32 into sleep and low-power modes much easier.



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.

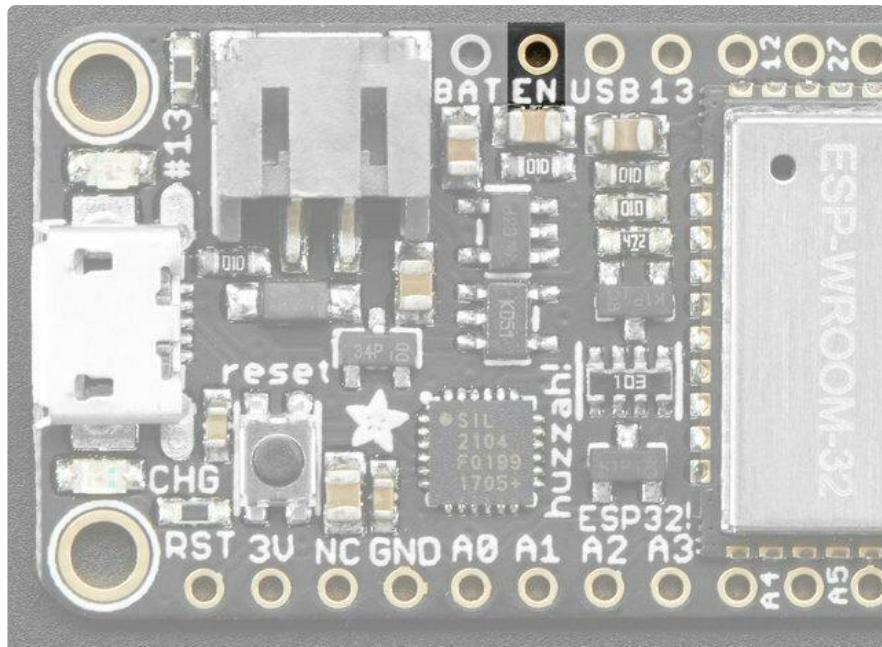
Since the ESP32 has tons of ADC pins, we 'sacrifice' one for Lipoly battery monitoring. You can read half of the battery voltage off of A13. Don't forget to double the voltage you read, since there is a divider.



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

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

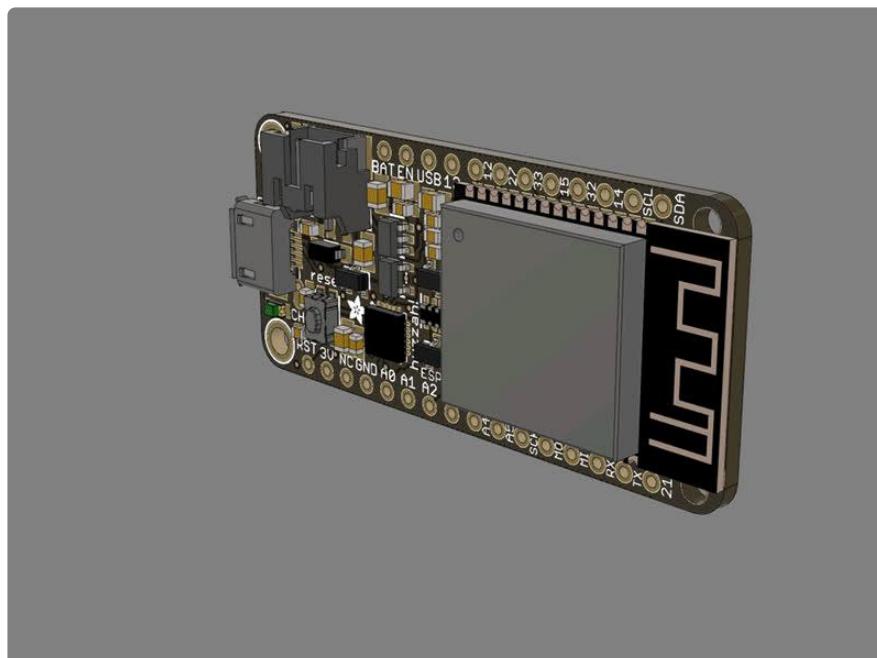
This is a minor design issue with the initial version of the Breakout (does not apply to Feather version). If you are having issues similar to the [discussion here](#) (), try the trick of enabling the internal pull-up as [described here](#) ().

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

