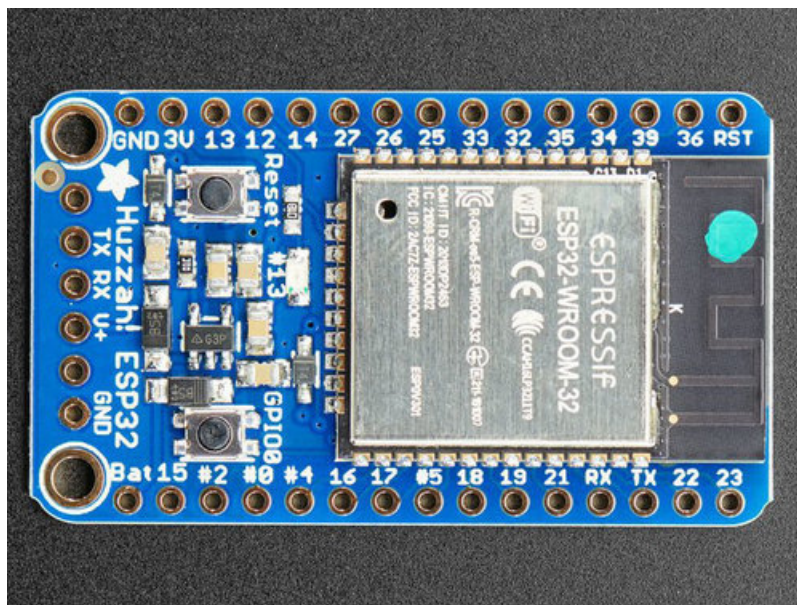


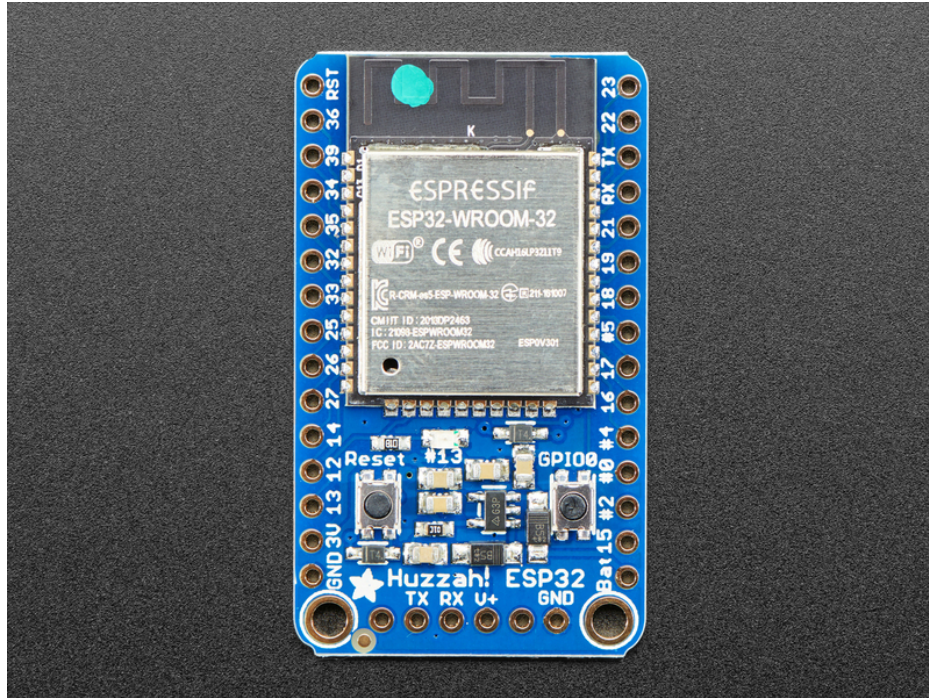
Adafruit HUZZAH32 - ESP32 Breakout Board

Created by Kattni Rembor



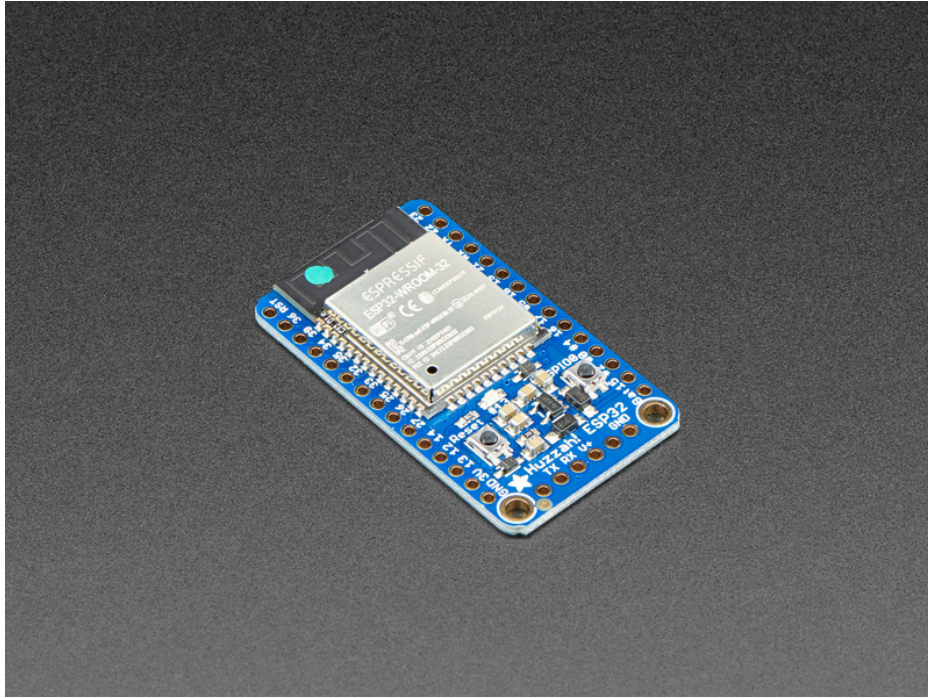
Last updated on 2020-12-09 03:50:02 PM EST

Overview



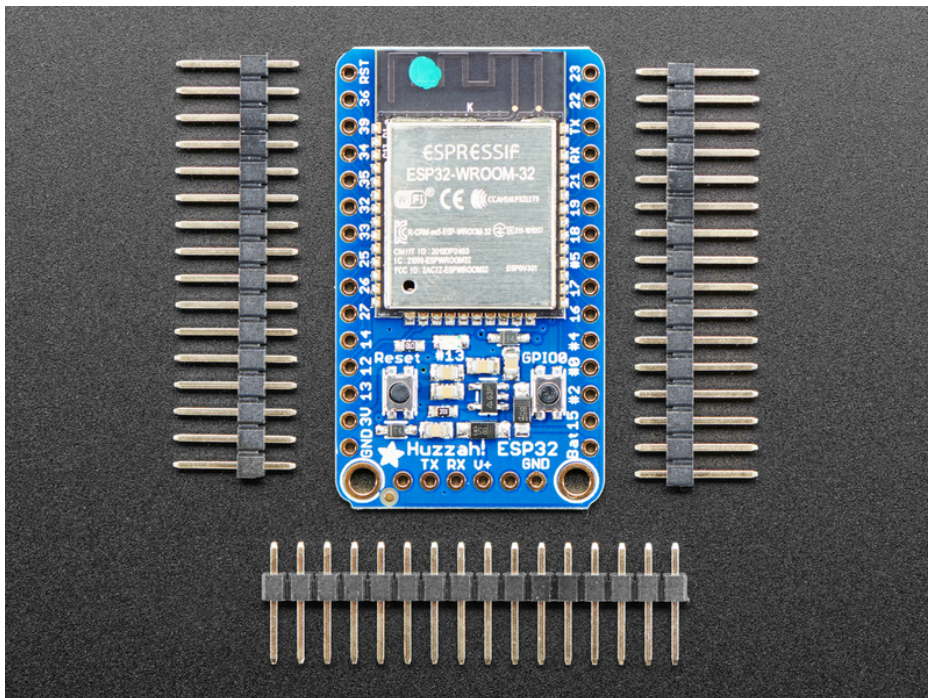
Squeeeeeeze down your next ESP32 project to its bare-bones essential with the **Adafruit HUZZAH32 Breakout**. This breakout is basically the 'big sister' of our HUZZAH 8266, but instead of an ESP8266 it has the '32! We've pared down our popular [Feather ESP32](https://adafru.it/wcN) (<https://adafru.it/wcN>), removing the battery charger and USB-serial converter. You just get a regulator, some protection diodes, two buttons and an LED. For some projects, where price and size are at a premium, you can program this board over the 'FTDI cable' breakout when needed, and leave it alone otherwise.

Note that this board *doesn't* come with a USB to serial converter chip and auto-reset circuit. Instead, you will need to plug in a [CP2104 Friend](https://adafru.it/C7B) (<https://adafru.it/C7B>) or [FTDI cable](https://adafru.it/dNN) (<https://adafru.it/dNN>). Then, before uploading code, put it into bootloader mode by holding down the GPIO #0 button and clicking Reset button, then releasing the #0 button.



That module in the middle of the breakout 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.

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!

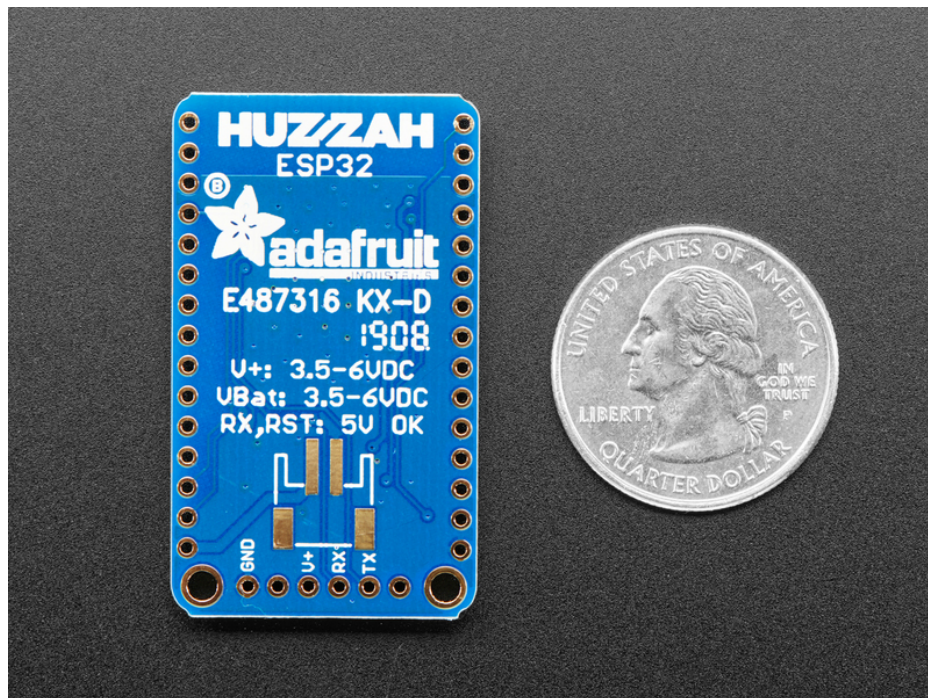


Comes fully assembled and tested, pre-programmed with ESP32 SPI WiFi co-processor firmware that [you can use in CircuitPython to use this into a WiFi co-processor over SPI + 2 pins \(https://adafruit.it/EvI\)](https://adafruit.it/EvI). We

also toss in some header so you can solder it in and plug into a solderless breadboard.

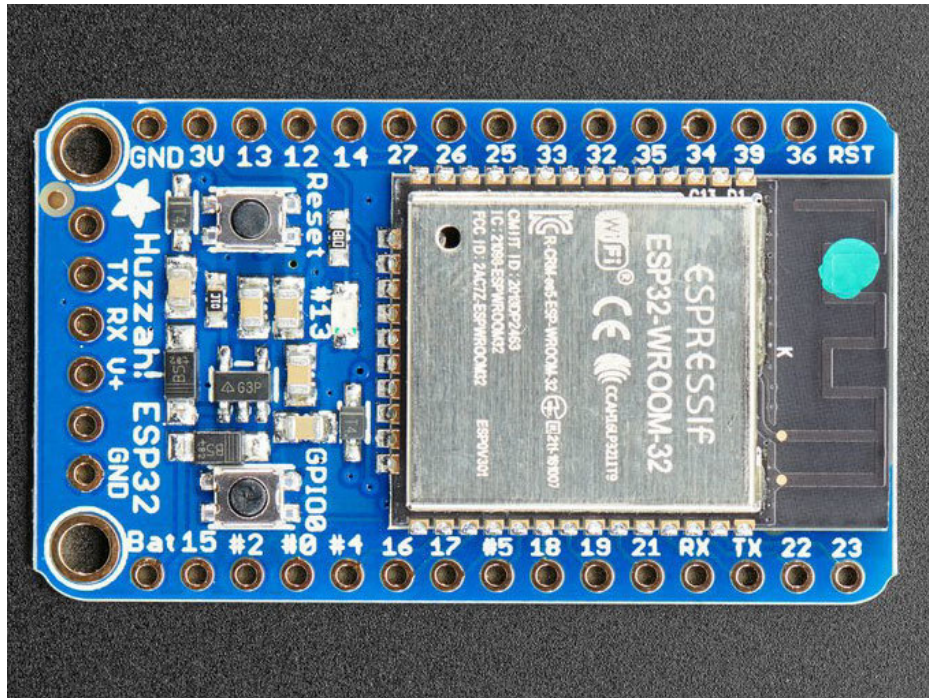
Here are [specifications from Espressif about the ESP32 \(https://adafru.it/wew\)](https://adafru.it/wew):

- 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 include in the WROOM32 module
- 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

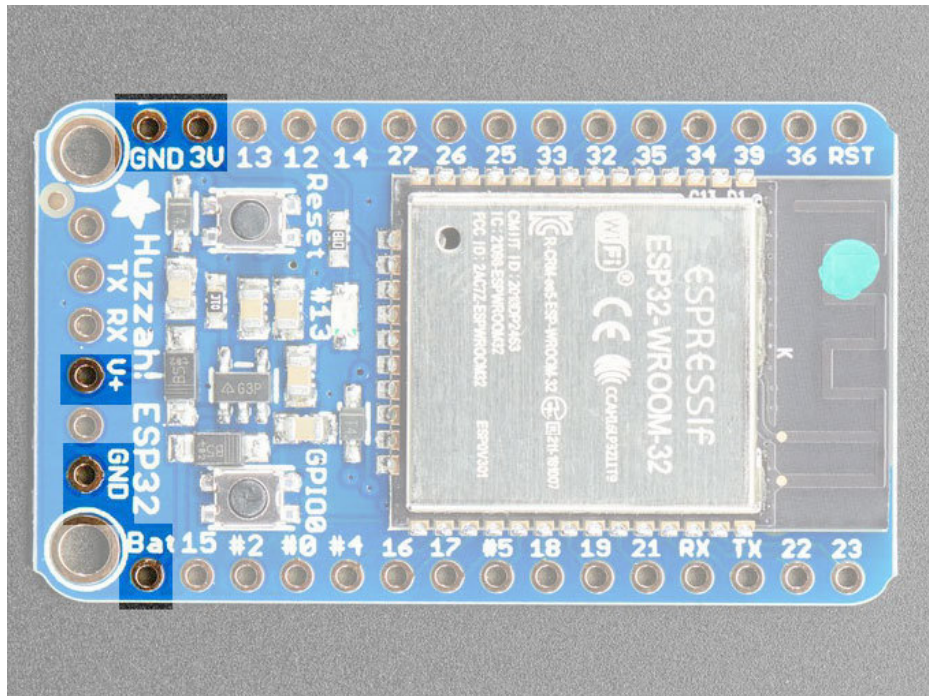


Pinouts

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

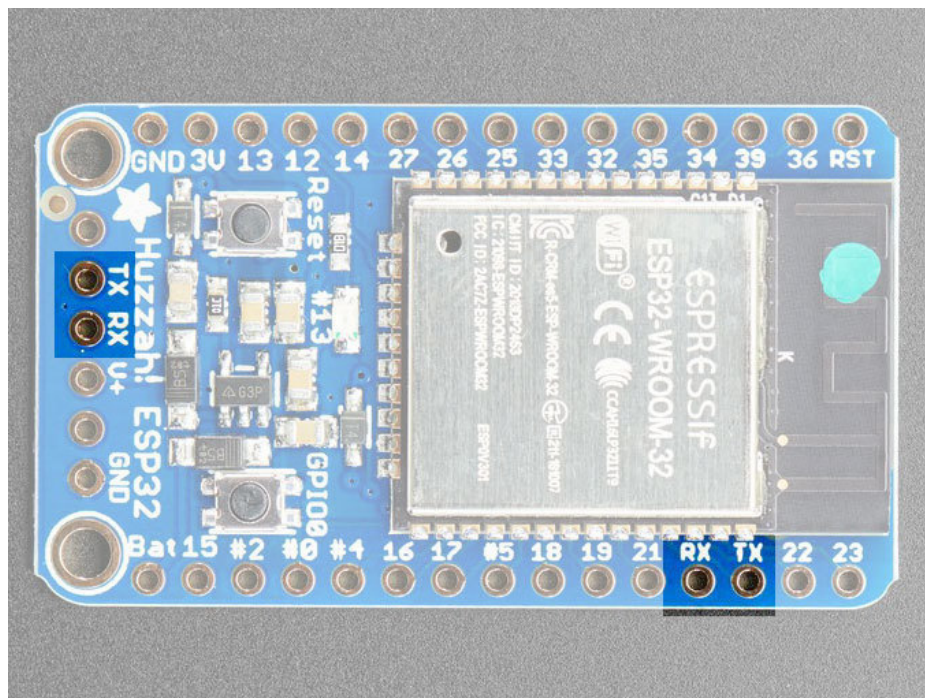


- **GND** - this is the common ground for all power and logic
- **BAT** - this is the positive voltage to/from the JST jack on the back as well as input into the 3.3V regulator

- **V+** - this is the positive voltage from the USB-to-Serial converter if one is plugged into the 6-pin header
- **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

Serial pins

RX and **TX** are the main Serial pins, and are connected to the USB/Serial converter. These are for use to program/bootload or debug the ESP32. They shouldn't be connected to any other hardware since the bootloader uses these pins only!



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!

There are tons of GPIO and analog inputs available to you for connecting LEDs, buttons, switches, sensors, etc. Here's the remaining pins available. Note that SPI/I2C/UART can be on any pins but we mark out the ones we've set up in our Arduino definition which is probably what you want. Likewise for the analog pin names.

- **IO0** - Used primarily for bootloader detect. Hold low while resetting to enter the bootloader. This is also connected to a small tactile button on the board.
- **IO2** - This is GPIO #2
- **IO4** - This is GPIO #4 and also an analog input **A5** on ADC #2.
- **IO5** - This is GPIO #5 and also SPI **SCK**

- **IO12** - 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.
- **IO13** - 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
- **IO14** - This is GPIO #14 and also an analog input **A6** on ADC #2
- **IO15** - This is GPIO #15 and also an analog input **A8** on ADC #2
- **IO16** - This is GPIO #16 and also Serial1 **RX**
- **IO17** - This is GPIO #17 and also Serial1 **TX**
- **IO18** - This is GPIO #18 and also SPI **MOSI**
- **IO19** - This is GPIO #19 and also SPI **MISO**
- **IO21** - This is GPIO #21
- **IO22** - This is GPIO #22 and also I2C **SCL**
- **IO23** - This is GPIO #23 and also I2C **SDA**
- **IO25** - This is GPIO #25 and an analog input **A1** on ADC #2 and also an analog output DAC1
- **IO26** - This is GPIO #26 and an analog input **A0** on ADC #2 and also an analog output DAC2
- **IO27** - This is GPIO #27 and also an analog input **A10** on ADC #2
- **IO32** - This is GPIO #32 and also an analog input **A7** on ADC #1. It can also be used to connect a 32 KHz crystal.
- **IO33** - This is GPIO #33 and also an analog input **A9** on ADC #1. It can also be used to connect a 32 KHz crystal.
- **IO34** - This is GPI #34 and also an analog input **A2**. Note it is *not* an output-capable pin! It uses ADC #1
- **IO35** - This is GPI #35 and also an analog input **A13**. Note it is *not* an output-capable pin! It uses ADC #1
- **IO36** - This is GPI #36 and also an analog input **A4**. Note it is *not* an output-capable pin! It uses ADC #1
- **IO39** - This is GPI #39 and also an analog input **A3**. Note it is *not* an output-capable pin! It uses ADC #1

Note you can only read analog inputs on **ADC #1** once WiFi has started

Using with Arduino IDE

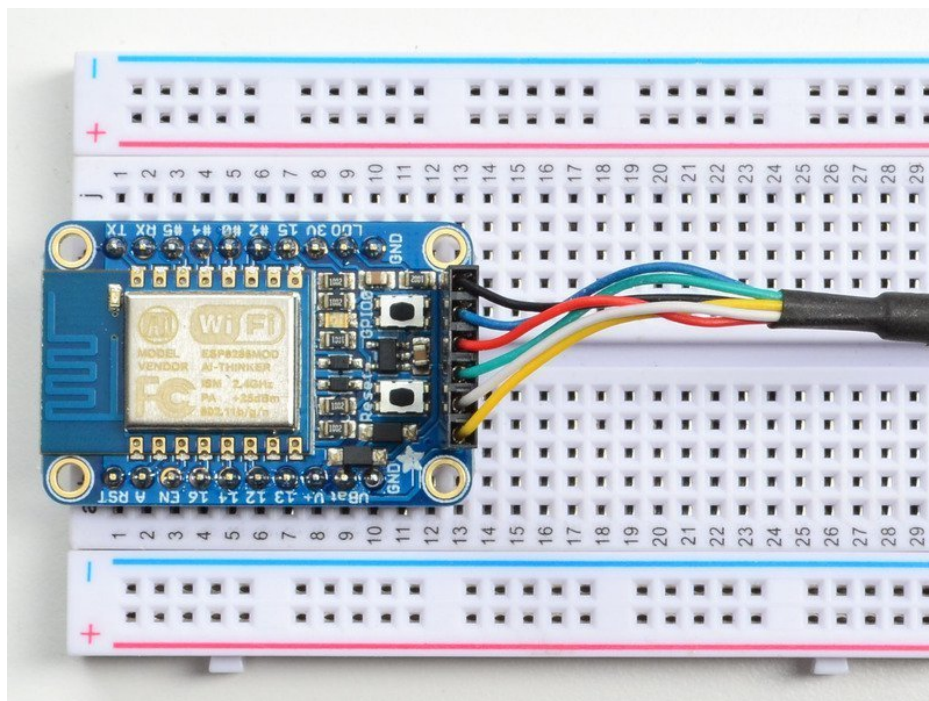
In order to upload code to the ESP32 and use the serial console, you will need a USB to serial converter! Use either an FTDI cable (<https://adafru.it/dNN>) or any console cable (<http://adafru.it/954>), you can use either 3V or 5V logic and power as there is level shifting on the RX pin.

□ The ESP32 uses a lot 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.

Connect USB-Serial cable

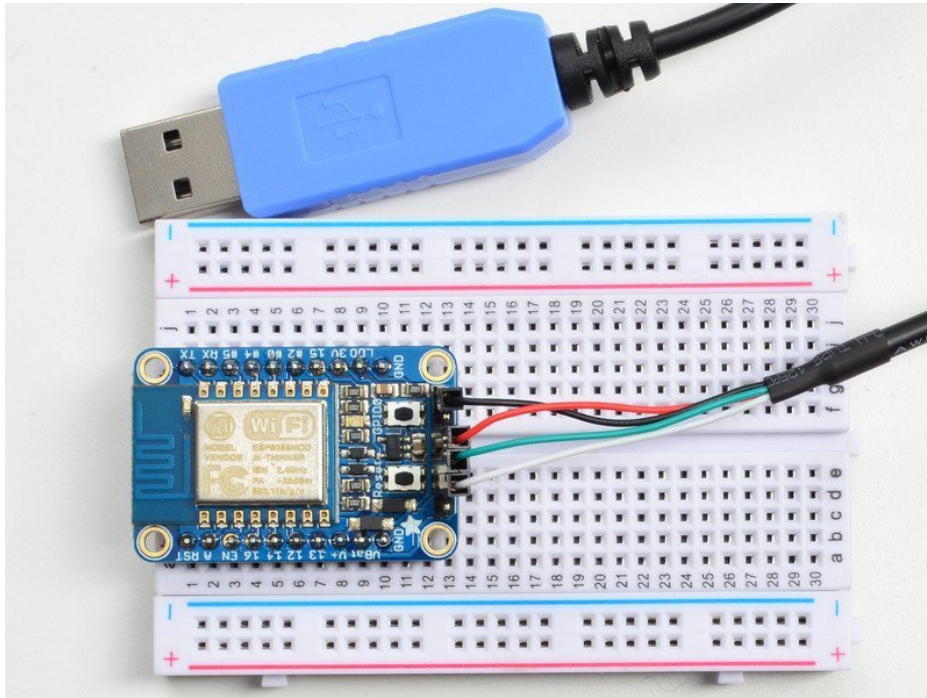
Connect either your console cable or FTDI cable. If using FTDI, make sure the black wire goes to the GND (ground) pin

We show a Huzzah ESP8266 in this photo but the connection setup is the same



If using a console cable, connect the black wire to ground, red wire to **V+**, white wire to **TX** and green wire to **RX**

We show a Huzzah ESP8266 in this photo but the connection setup is the same



Don't forget you may also need to install the SiLabs CP2104 Driver or FTDI driver for your cable!

<https://adafruit.it/vrf>

<https://adafruit.it/vrf>

<https://adafruit.it/ENP>

<https://adafruit.it/ENP>

Install the Arduino IDE & ESP32 package

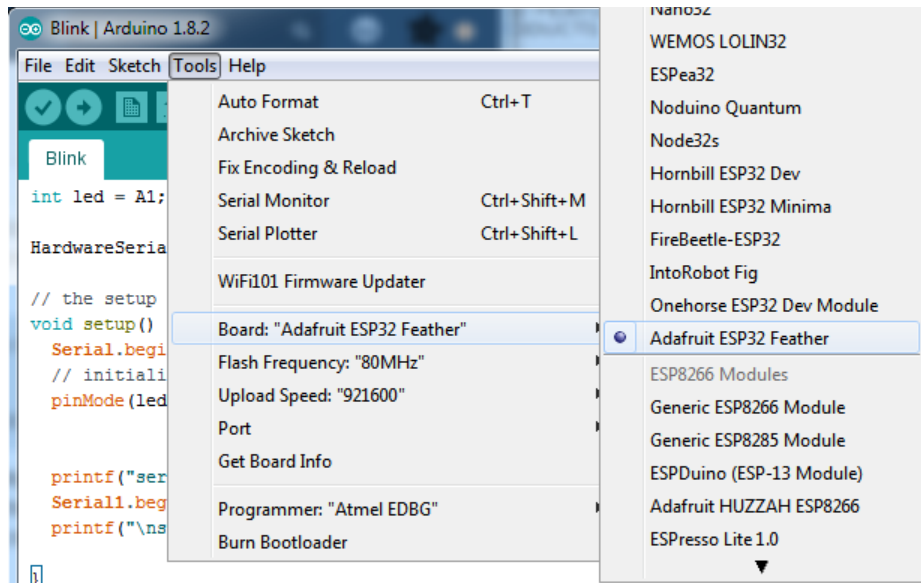
Download the latest Arduino IDE from [Arduino.cc](https://www.arduino.cc/)! You can use your existing IDE if you have already installed it (<https://adafruit.it/f1P>)

We primarily recommend using the ESP32 Huzzah with Arduino.

Check out the [Espressif Arduino repository](https://adafruit.it/weF) for details on how to install it (<https://adafruit.it/weF>)

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

For **Upload speed** we've found **921600** baud works great, **but use 115200** if you're having upload issues.



Blink Test

We'll begin with the simple blink test

Enter this into the sketch window (and save since you'll have to)

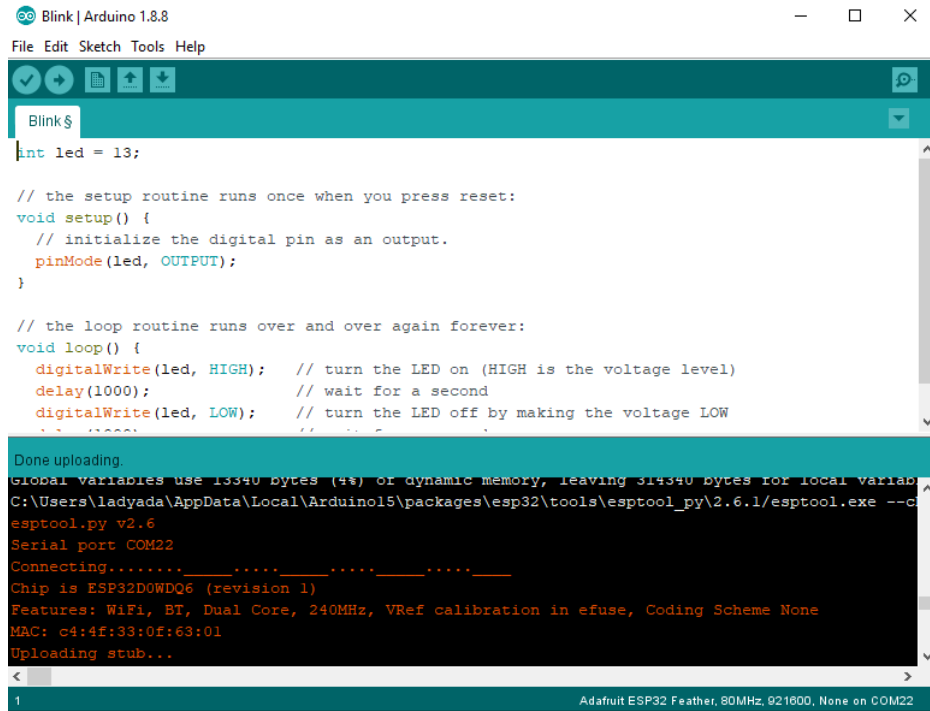
```
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH);
  delay(500);
  digitalWrite(13, LOW);
  delay(500);
}
```

Now you'll need to put the board into bootload mode. You'll have to do this before each upload. There is no timeout for bootload mode, so you don't have to rush!

1. Hold down the **GPIO0** button
2. While holding down **GPIO0**, click the **RESET** button
3. Release **RESET**, then release **GPIO0**

Once the ESP board is in bootload mode, upload the sketch via the IDE



The screenshot shows the Arduino IDE interface. The top menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for opening, saving, and running the sketch. The main text area displays the 'Blink' sketch code, which defines a pin, a setup routine, and a loop routine. The serial monitor at the bottom shows the upload progress and the final status message.

```
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Done uploading.
Global variables use 13340 bytes (44%) of dynamic memory, leaving 314340 bytes for local variables.
C:\Users\ladyada\AppData\Local\Arduino15\packages\esp32\tools\esptool_py\2.6.1/esptool.exe --chip esp32 --port COM22 --baud 115200 --flash-mode dio_qout -- esptool.py v2.6
Serial port COM22
Connecting.....
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
MAC: c4:4f:33:0f:63:01
Uploading stub...

1 Adafruit ESP32 Feather, 80MHz, 921600, None on COM22

During upload if you see

```
esptool.py v2.6
Serial port COM22
Connecting.....
```

Try resetting the board into bootloader mode again, by holding down the **GPIO0** button and pressing **Reset** then releasing **GPIO0**

If the upload is successful, it should end with this message:

```
Hard resetting via RTS pin...
```

Once you see that, press the **RESET** button and the sketch will then run.

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



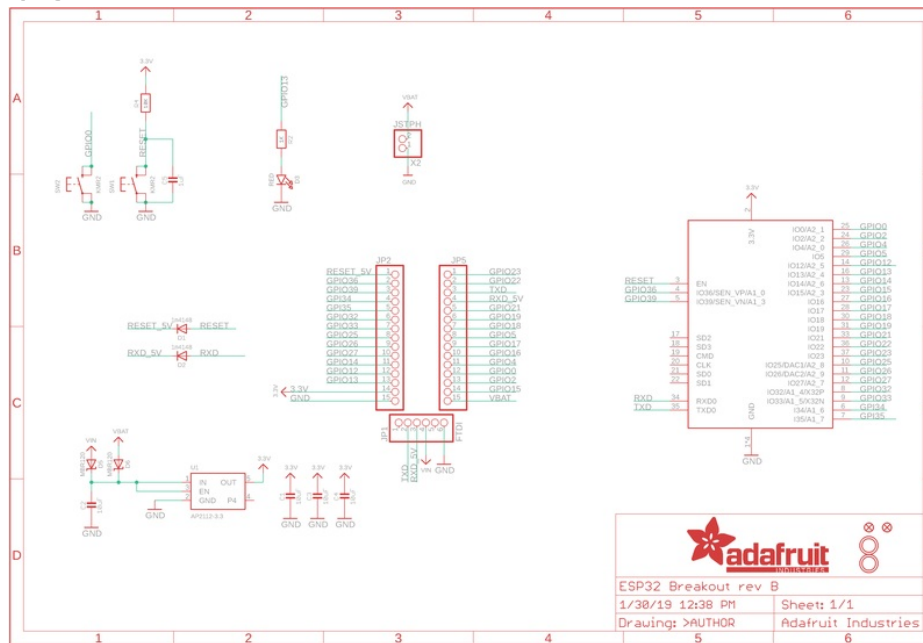
Why is Serial.read() not working as expected on ESP32 Breakout?

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 \(https://adafru.it/ven\)](https://adafru.it/ven), try the trick of enabling the internal pull-up as [described here \(https://adafru.it/ven\)](https://adafru.it/ven).

Downloads Files

- EagleCAD PCB files on GitHub (<https://adafru.it/Ewa>)
- Fritzing object on Adafruit Fritzing Library (<https://adafru.it/Ewb>)

Schematic



Fab Print



