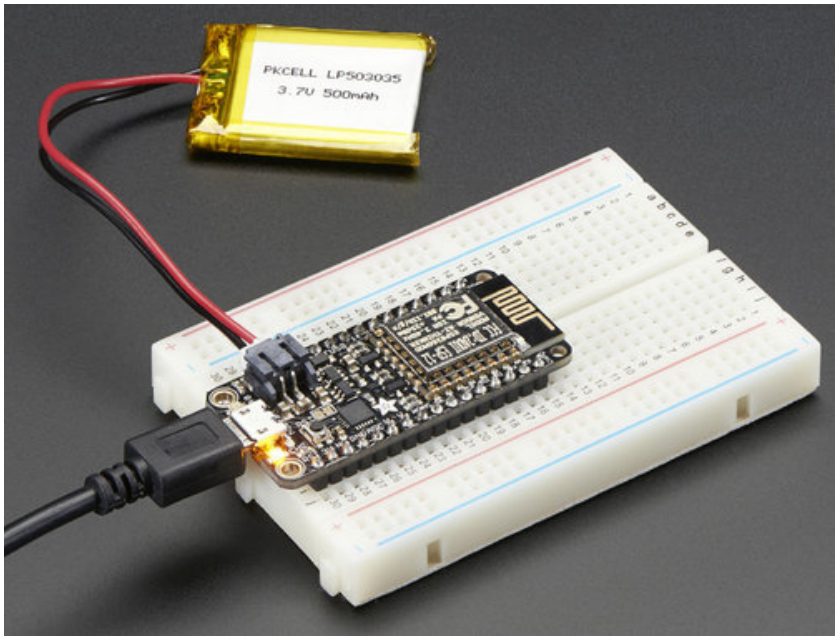
adafruit learning system

Adafruit Feather HUZZAH ESP8266

Created by lady ada



Last updated on 2016-03-19 08:43:43 PM EDT

Guide Contents

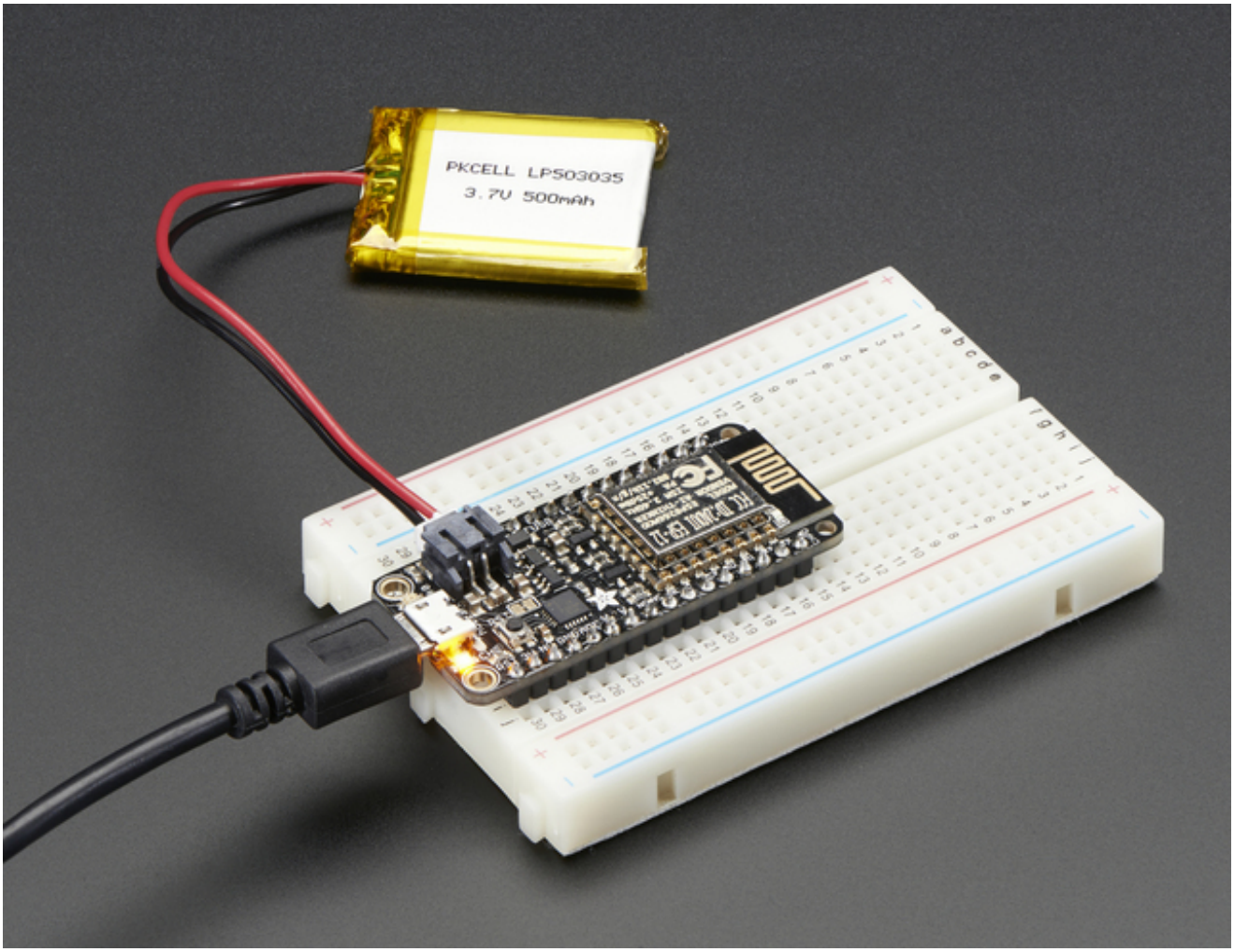
Guide Contents 2
Overview 4
Pinouts 9
Power Pins 9
Logic pins 10
Serial pins 10
I2C & SPI pins 11
GPIO pins 12
Analog Pins 13
Other control pins 13
Assembly 15
Header Options! 15
Soldering in Plain Headers 16
- Prepare the header strip: 16
- Add the breakout board: 17
- And Solder! 17

Soldering on Female Header 18
- Tape In Place 18
- Flip & Tack Solder 19
- And Solder! 19

Power Management 21
Battery + USB Power 21
Power supplies 22
Measuring Battery 23
ENable pin 23
Using NodeMCU Lua 25
Open up serial console 25
Hello world! 27
Scanning & Connecting to WiFi 28
WebClient example 29
Using Arduino IDE 31
Install the Arduino IDE 1.6.4 or greater 32

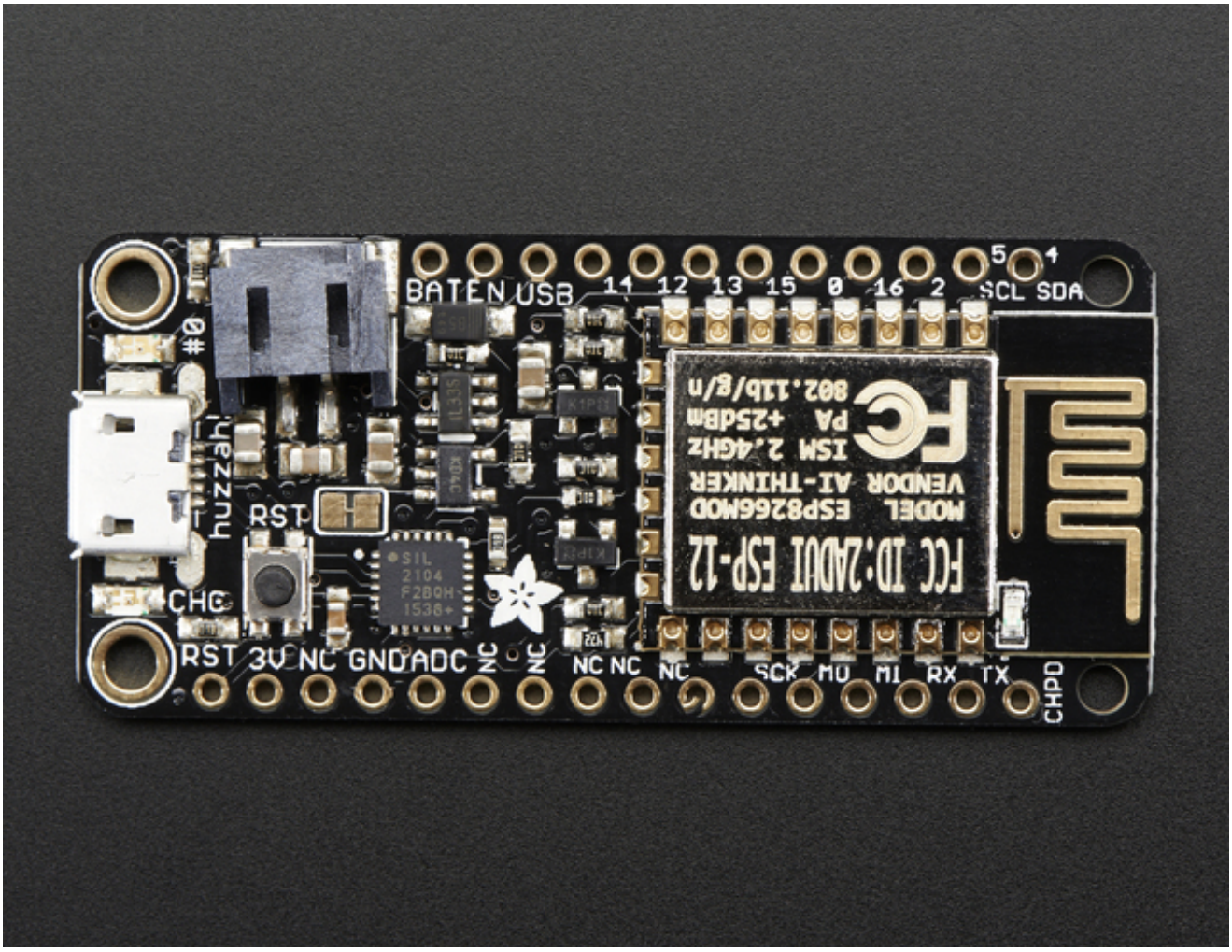
Install the ESP8266 Board Package 32
Setup ESP8266 Support 33
Blink Test 36
Connecting via WiFi 37
F.A.Q. 42
Downloads 43
Datasheets 43
More info about the ESP8266 43
Schematic 43
Fabrication Print 44

Overview

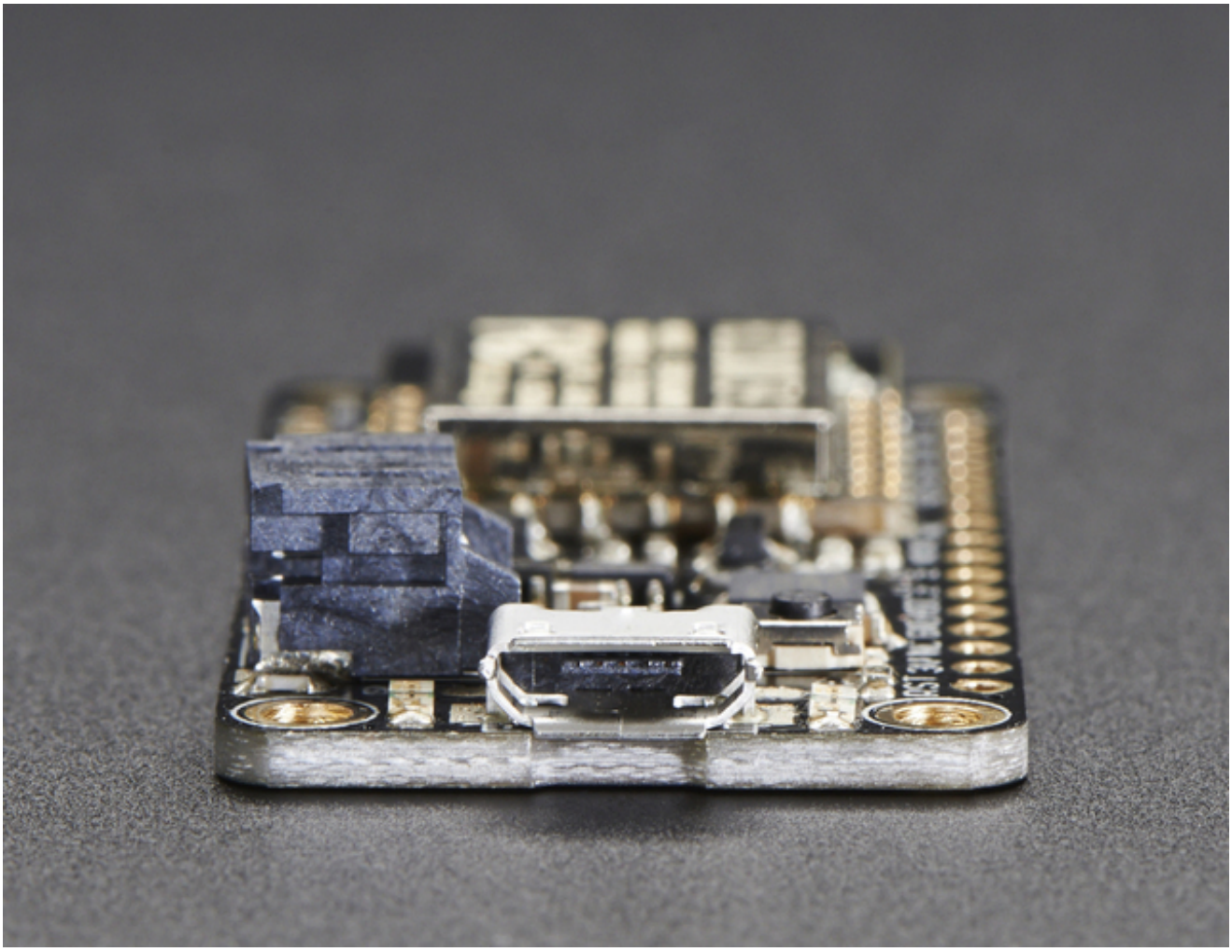


Feather is the new development board from Adafruit, and like it's namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores.

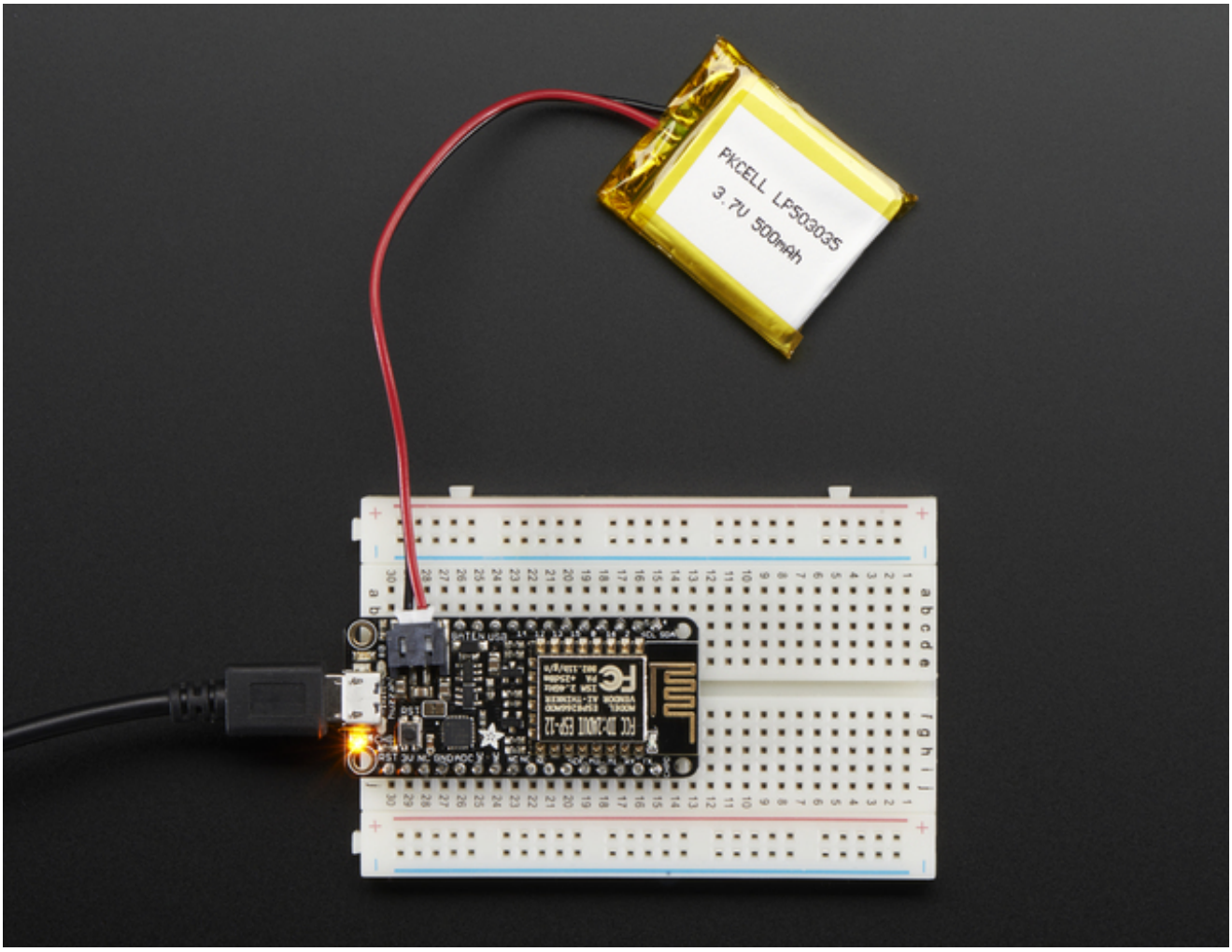
This is the **Adafruit Feather HUZZAH ESP8266** - our take on an 'all-in-one' ESP8226 WiFi development board with built in USB and battery charging. Its an ESP8266 WiFi module with all the extras you need, ready to rock! We have other boards in the Feather family, check'em out here (http://adafru.it/jAQ).



At the Feather HUZZAH's heart is an ESP8266 WiFi microcontroller clocked at 80 MHz and at 3.3V logic. This microcontroller contains a Tensilica chip core as well as a full WiFi stack. You can progam the microcontroller using the Arduino IDE for an easy-to-run Internet of Things core. We wired up a USB-Serial chip that an upload code at a blistering 921600 baud for fast development time. It also has auto-reset so no noodling with pins and reset button pressings.

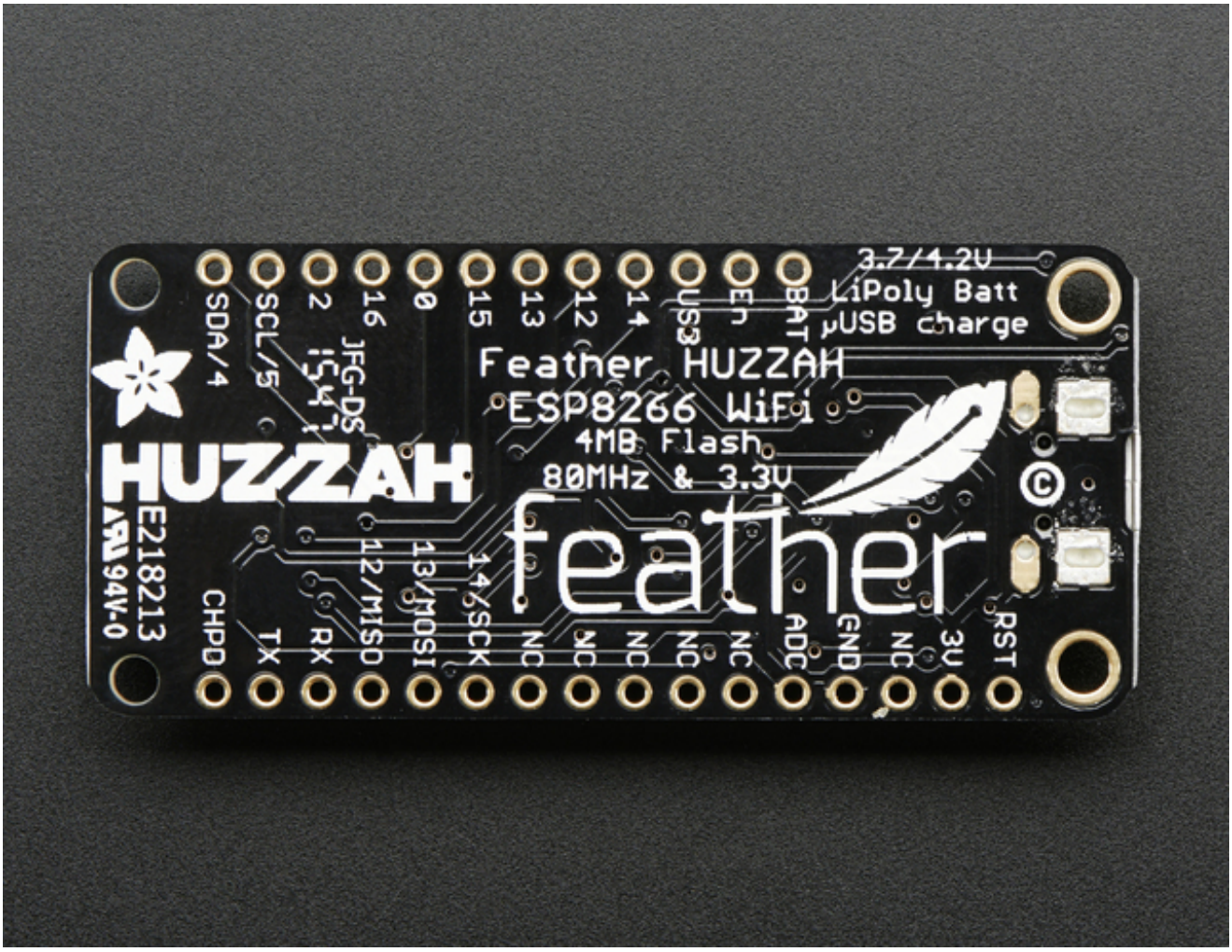


To make it easy to use for portable projects, we added a connector for any of our 3.7V Lithium polymer batteries and built in battery charging. You don't need a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then plug in the USB to recharge. The Feather will automatically switch over to USB power when its available.



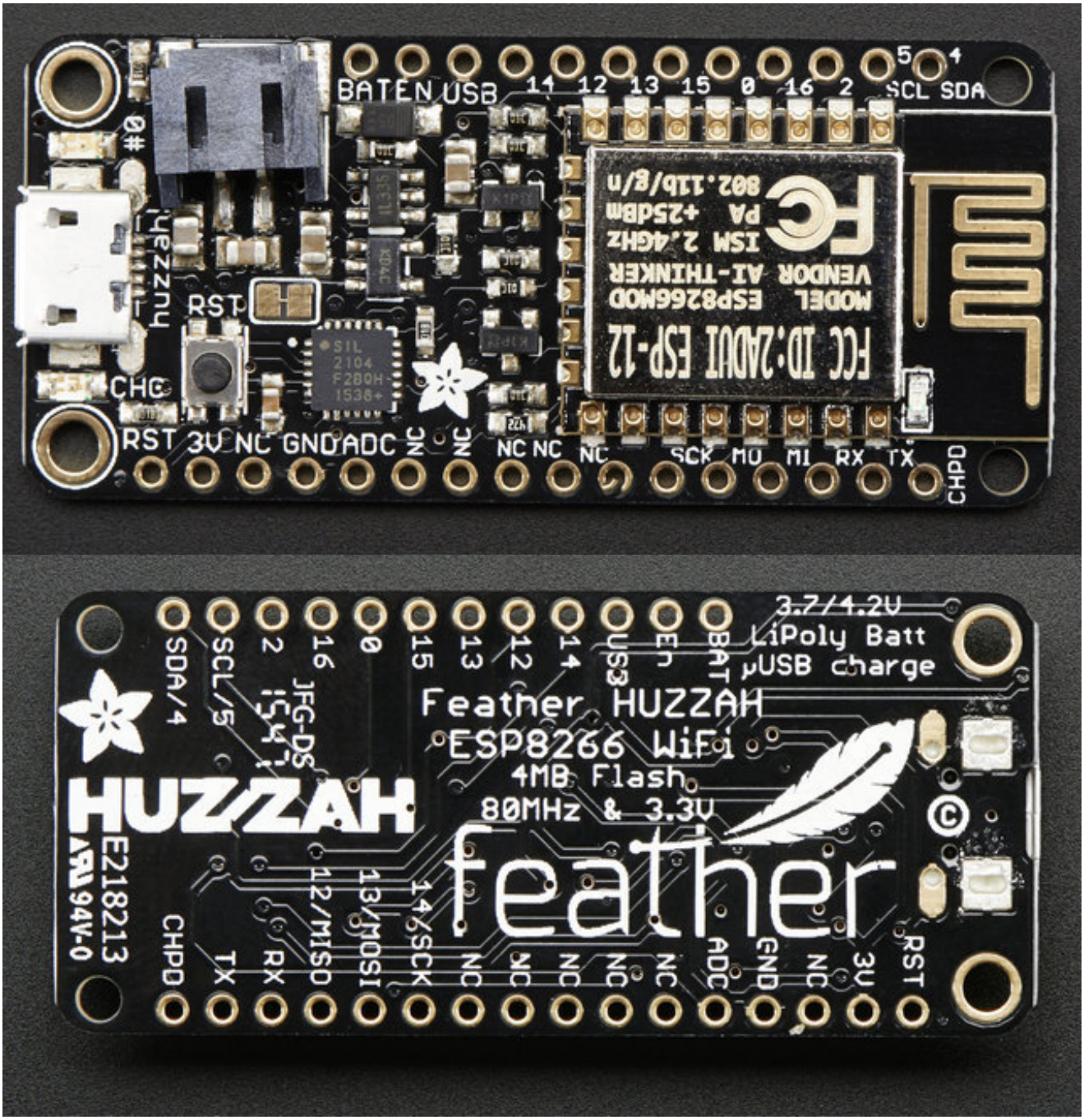
Here's some handy specs! Like all Feather 32u4's you get:

- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather - 6 grams
- ESP8266 @ 40MHz with 3.3V logic/power
- 4MB of FLASH (32 MBit)
- 3.3V regulator with 500mA peak current output
- CP2104 USB-Serial converter onboard with 921600 max baudrate for uploading
- Auto-reset support for getting into bootload mode before firmware upload
- 9 GPIO pins - can also be used as I2C and SPI
- 1 x analog inputs 1.0V max
- Built in 100mA lipoly charger with charging status indicator LED
- Pin #0 red LED for general purpose blinking. Pin #2 blue LED for bootloading debug & general purpose blinking
- Power/enable pin
- 4 mounting holes
- Reset button

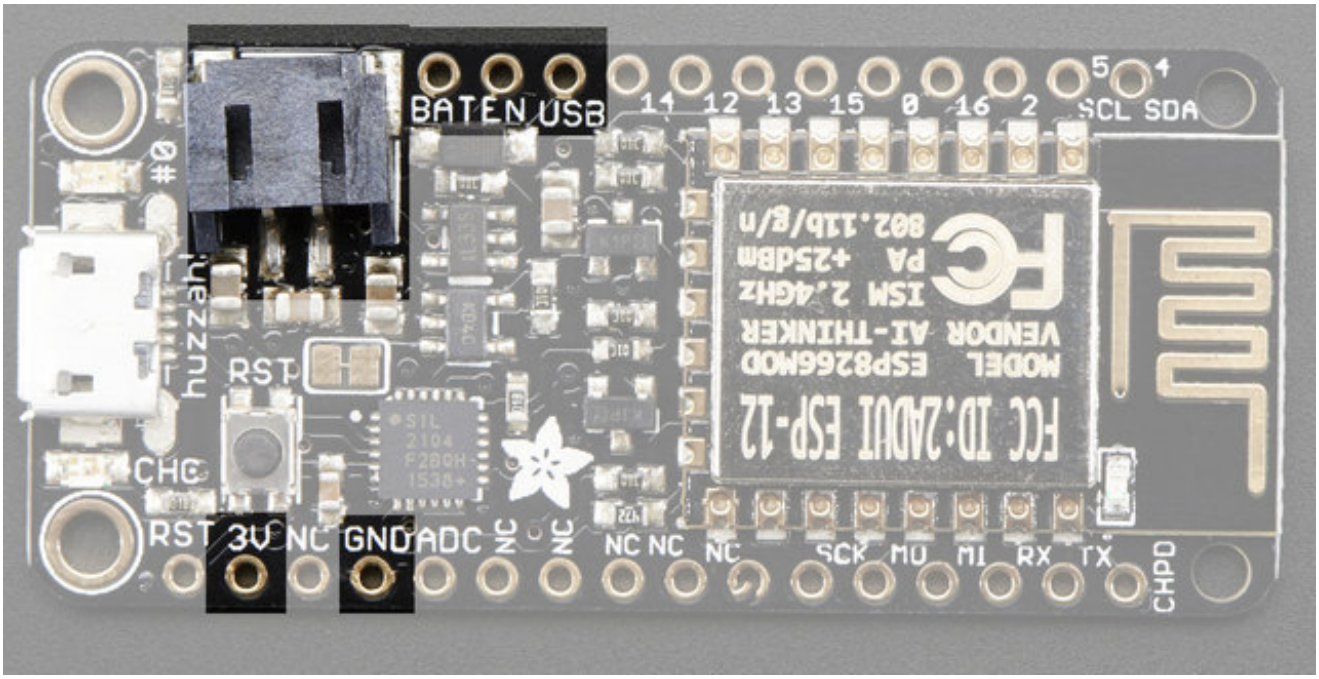


Comes fully assembled and tested, with a USB interface that lets you quickly use it with the Arduino IDE or NodeMCU Lua. (It comes preprogrammed with the Lua interpretter) We also toss in some header so you can solder it in and plug into a solderless breadboard. **Lipoly battery and USB cable not included**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

Pinouts



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, it can supply 500mA peak (try to keep your current draw under 250mA so you have plenty for the ESP8266's power requirements!)

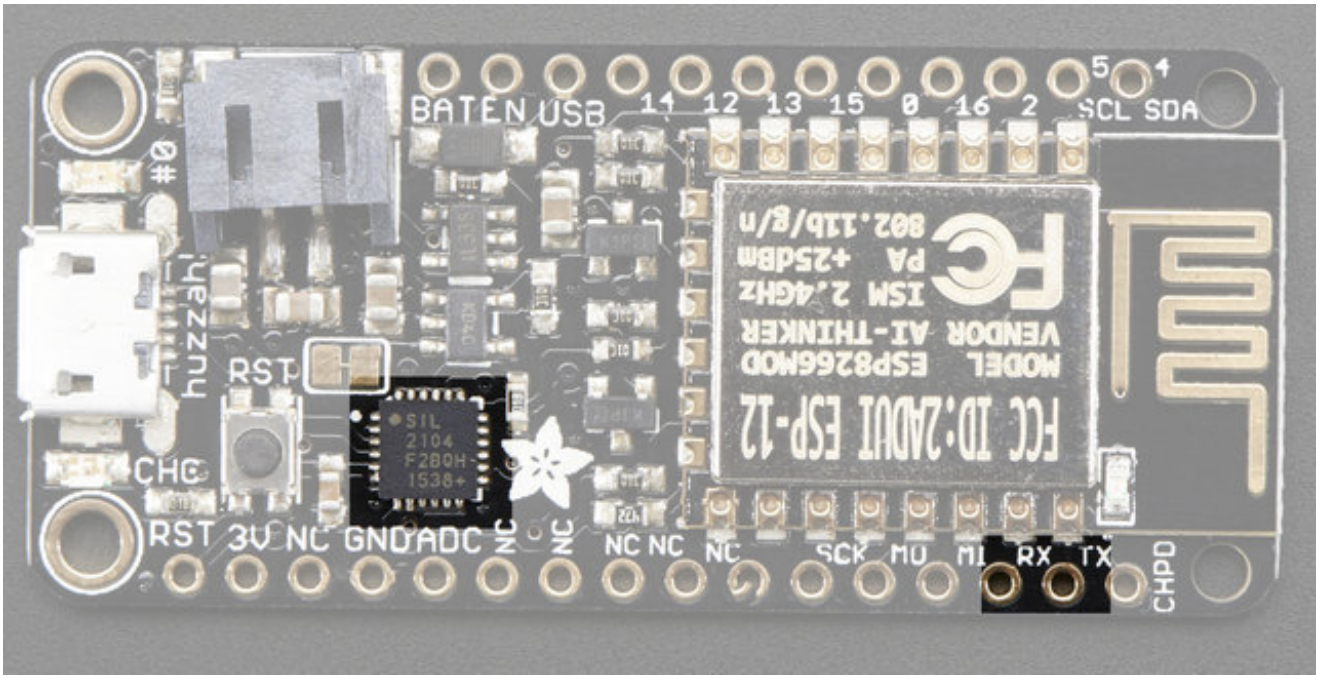
Logic pins

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

> The ESP8266 runs on 3.3V power and logic, and unless otherwise specified, GPIO pins are not 5V safe! The analog pin is also 1.0V max!

Serial pins

RX and **TX** are the serial control and bootloading pins, and are how you will spend most of your time communicating with the ESP module



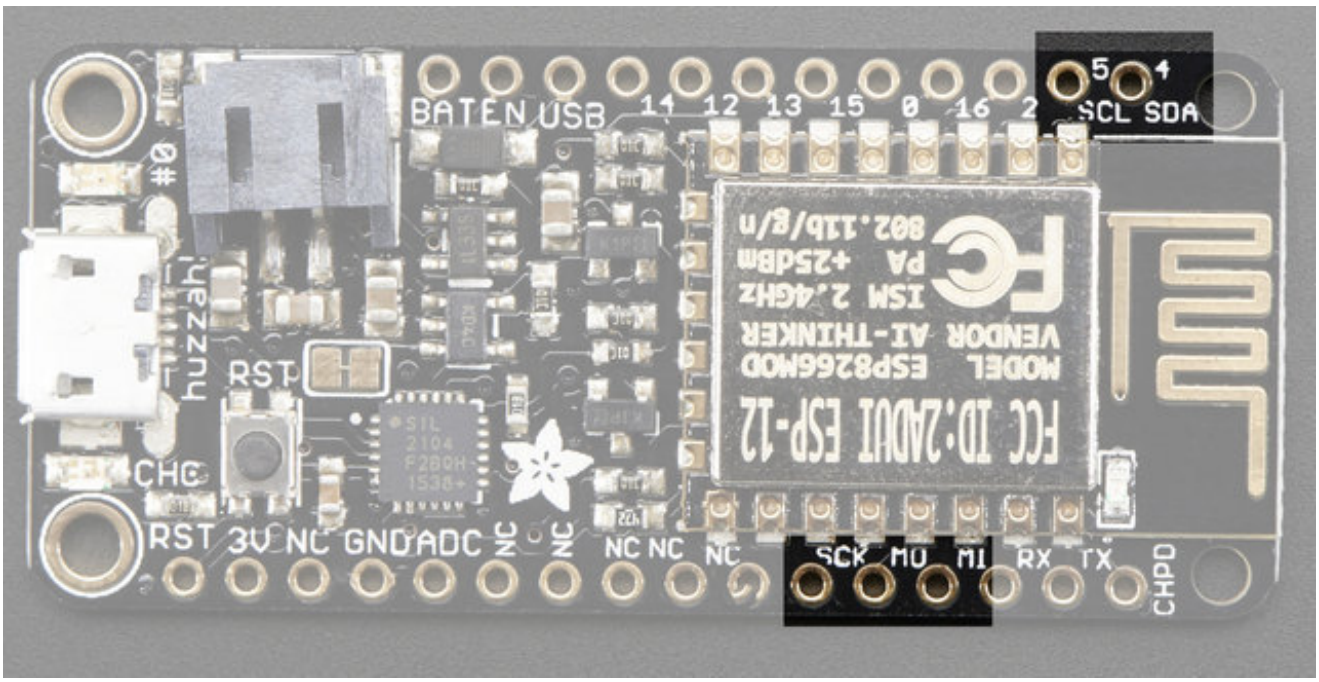
The **TX** pin is the output *from* the module and is 3.3V logic.

The **RX** pin is the input *into* the module and is 5V compliant (there is a level shifter on this pin)

These are connected through to the CP2104 USB-to-Serial converter so they shouldn't be connected to unless you're super sure you want to because you will also be getting the USB traffic on these!

I2C & SPI pins

You can use the ESP8266 to control I2C and SPI devices, sensors, outputs, etc. While this is done by 'bitbanging', it works quite well and the ESP8266 is fast enough to match 'Arduino level' speeds.



In theory you can use *any* pins for I2C and SPI but to make it easier for people using existing Arduino code, libraries, sketches we set up the following:

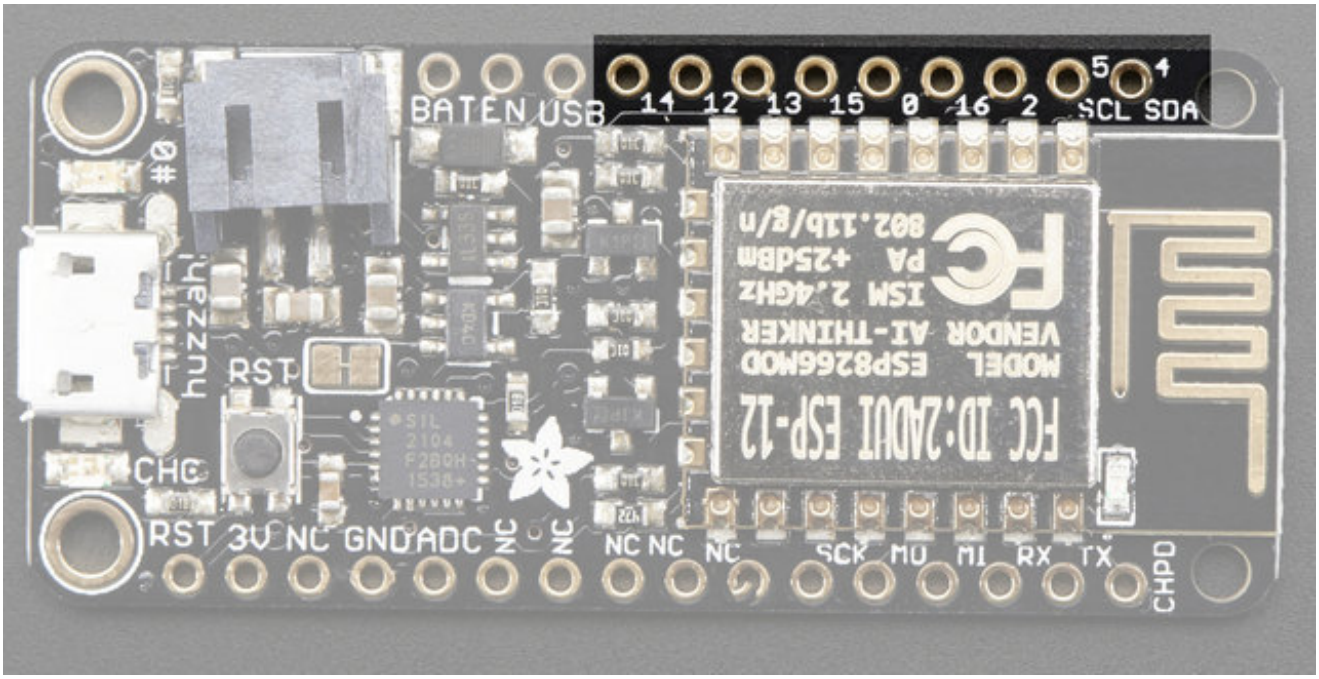
- **I2C SDA** = **GPIO #4** (default)
- **I2C SCL** = **GPIO #5** (default)

If you want, you can connect to I2C devices using other 2 pins in the Arduino IDE, by calling `Wire.pins(sda, scl)` before any other Wire code is called (so, do this at the begining of `setup()` for example

Likewise, you can use SPI on any pins but if you end up using 'hardware SPI' you will want to use the following:

- **SPI SCK = GPIO #14** (default)
- **SPI MOSI = GPIO #13** (default)
- **SPI MISO = GPIO #12** (default)

GPIO pins



This breakout has 9 GPIO: **#0, #2, #4, #5, #12, #13, #14, #15, #16** arranged at the top edge of the Feather PCB

All GPIO are 3.3V logic level in and out, and are **not 5V compatible.** Read the full spec sheet (http://adafru.it/f1E) to learn more about the GPIO pin limits, but be aware the maximum current drawn per pin is **12mA**.

These pins are general purpose and can be used for any sort of input or output. Most also have the ability to turn on an internal pullup. Many have *special* functionality:

GPIO #0, which does not have an internal pullup, and is also connected a red LED. This pin is used by

the ESP8266 to determine when to boot into the bootloader. If the pin is held low during power-up it will start bootloading! That said, you can always use it as an output, and blink the red LED.

GPIO #2, is also used to detect boot-mode. It also is connected to the blue LED that is near the WiFi antenna. It has a pullup resistor connected to it, and you can use it as any output (like #0) and blink the blue LED.

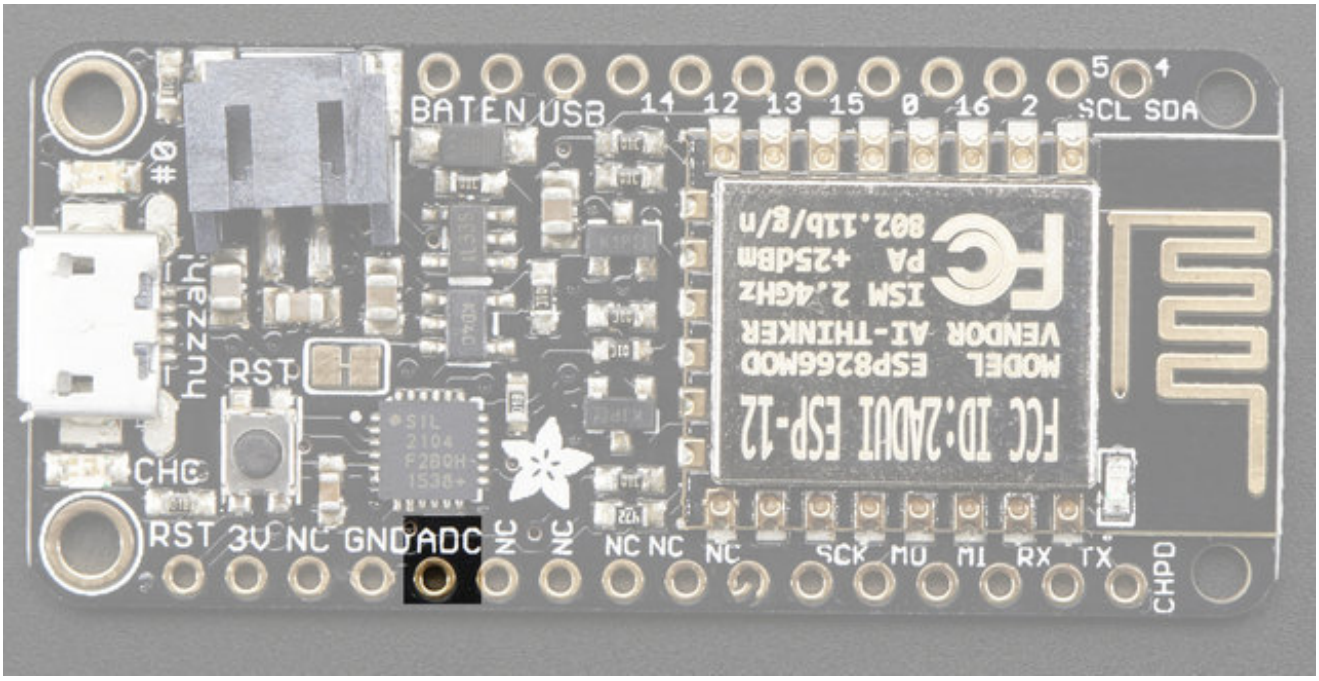
GPIO #15, is also used to detect boot-mode. It has a pulldown resistor connected to it, make sure this pin isn't pulled high on startup. You can always just use it as an output

GPIO #16 can be used to wake up out of deep-sleep mode, you'll need to connect it to the RESET pin

Also note that GPIO #**12**/**13**/**14** are the same as the **SCK**/**MOSI**/**MISO** 'SPI' pins!

Analog Pins

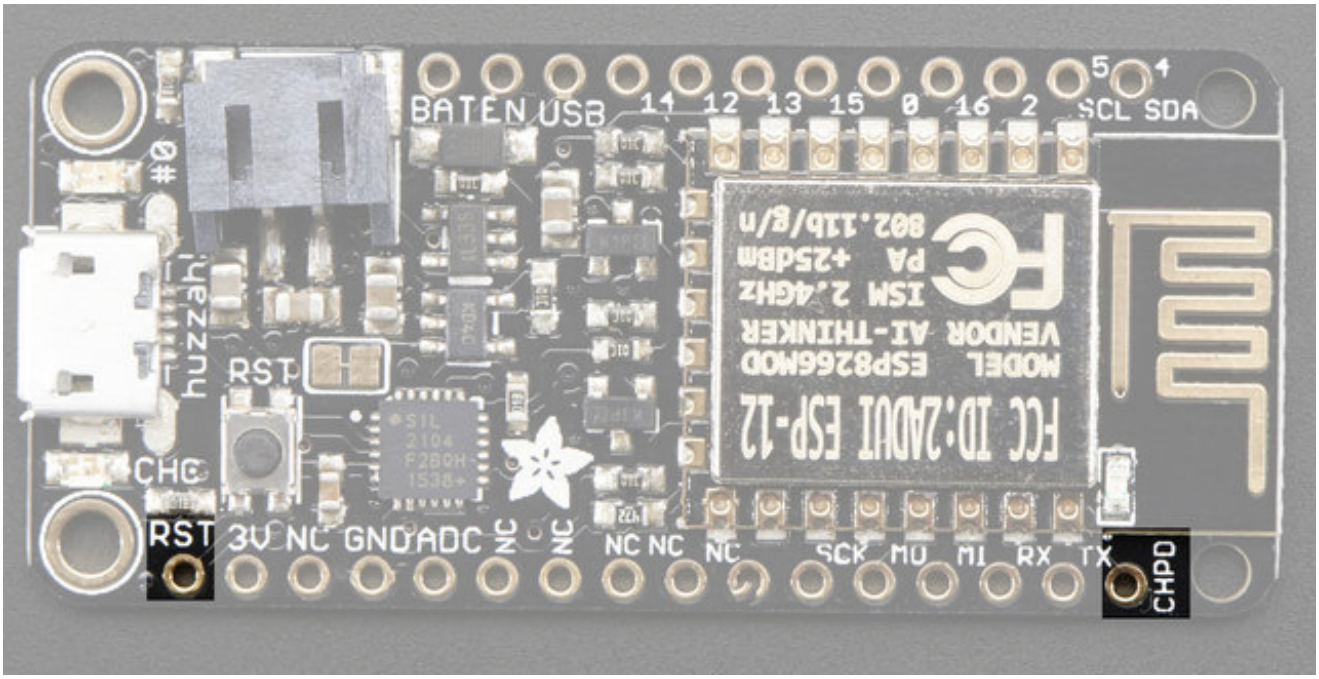
There is also a single analog input pin called **A**. This pin has a ~1.0V maximum voltage, so if you have an analog voltage you want to read that is higher, it will have to be divided down to 0 - 1.0V range



Other control pins

We have a few other pins for controlling the ESP8266

- **RST** - this is the reset pin for the ESP8266, pulled high by default. When pulled down to ground momentarily it will reset the ESP8266 system. This pin is 3.3V logic only
- **EN (CH_PD)** - This is the enable pin for the ESP8266, pulled high by default. When pulled down to ground momentarily it will reset the ESP8266 system. This pin is 3.3V logic only

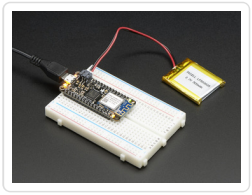
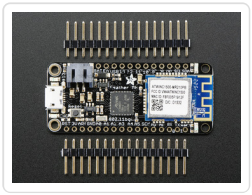
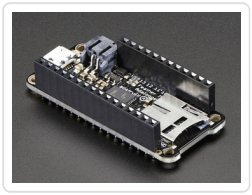
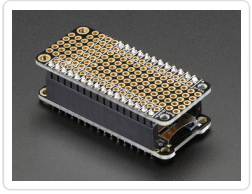

BAT EN USB 11 12 13 15 0 16 2 5 4 SCL SDA
#0
huzzah
RST
CHG
RST 3V NC GND ADC NC NC NC NC NC SCK MO MI RX TX CHPD
FCC ID:2ADUI ESP-12
MODEL ESP8266MOD
VENDOR AI-THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n

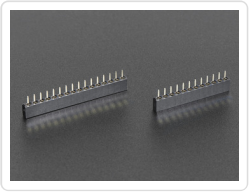
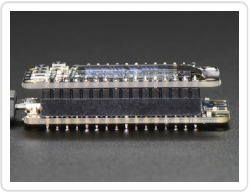
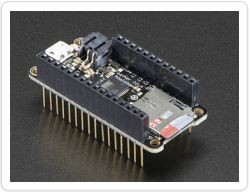
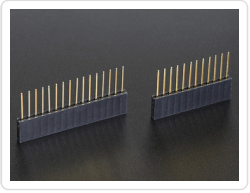
Assembly

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

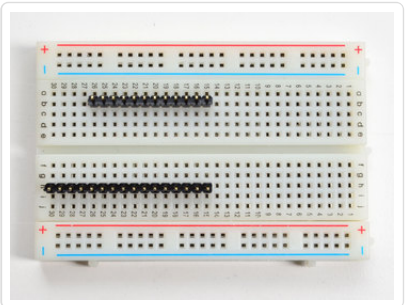
Header Options!

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
- 

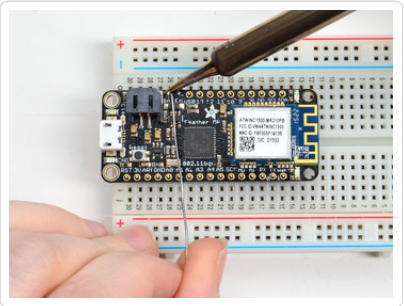
- 
  We also have 'slim' versions of the female headers, that are a little shorter and give a more compact shape
- 
- 
  Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a solderless breadboard *and* plug a featherwing on top. But its a little bulky
- 

Soldering in Plain Headers

- 

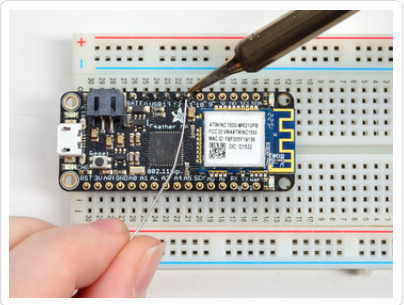
Prepare the header strip:

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:

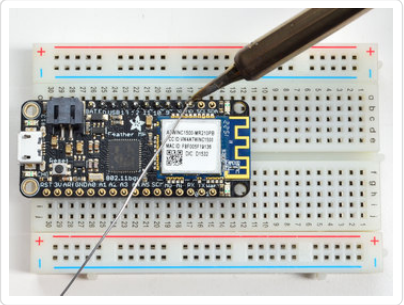
- Place the breakout board over the pins so that the short pins poke through the breakout pads

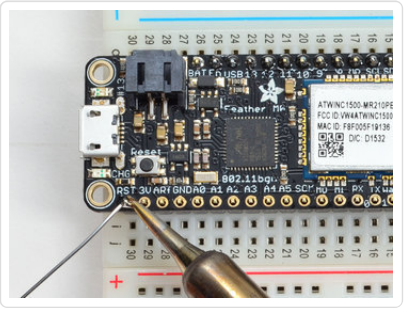
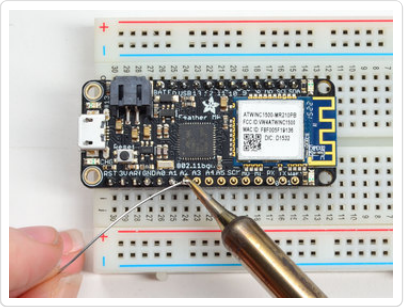
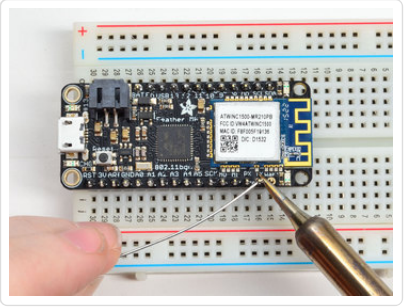
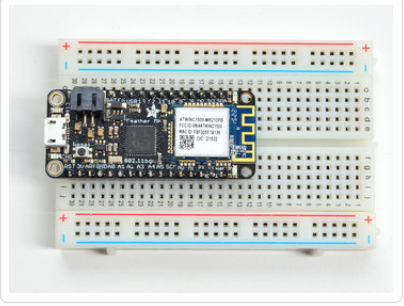


And Solder!

- Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (http://adafru.it/aTk).

- 

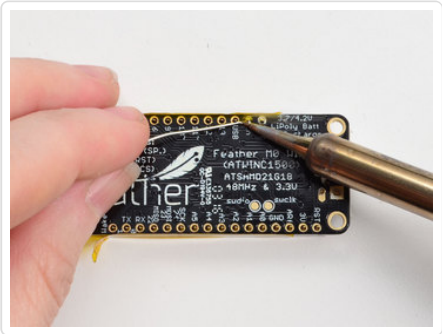
-  Solder the other strip as well.
- 
- 
-  You're done! Check your solder joints visually and continue onto the next steps

Soldering on Female Header

-

Tape In Place

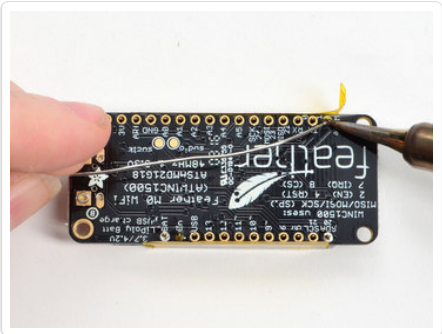
For sockets you'll want to tape them in place so when you flip over the board they don't fall out

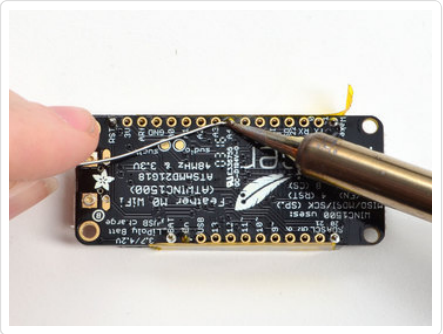
- 

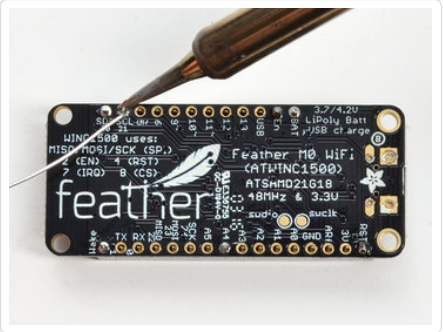
-

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 (http://adafru.it/aTk).

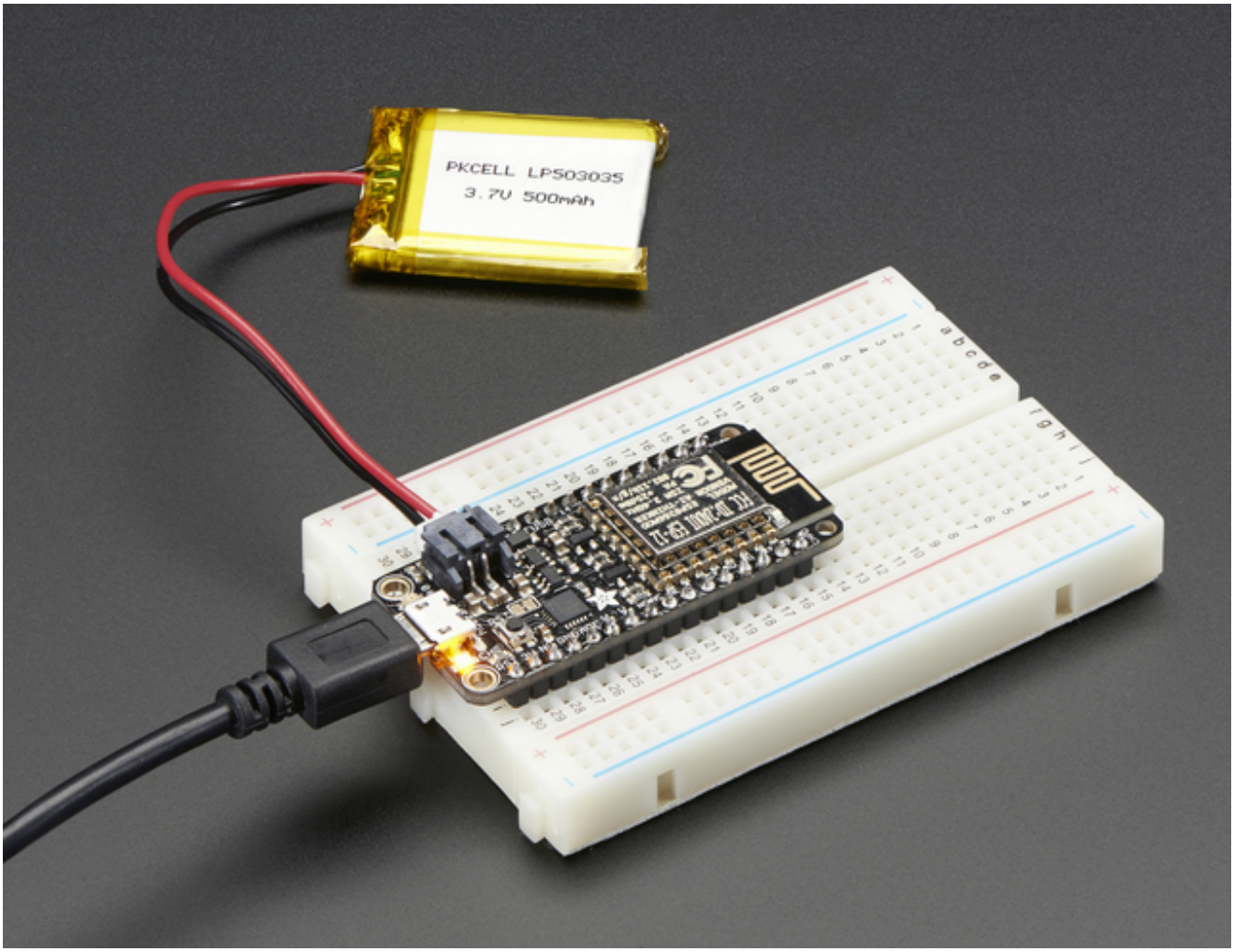
-

-

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

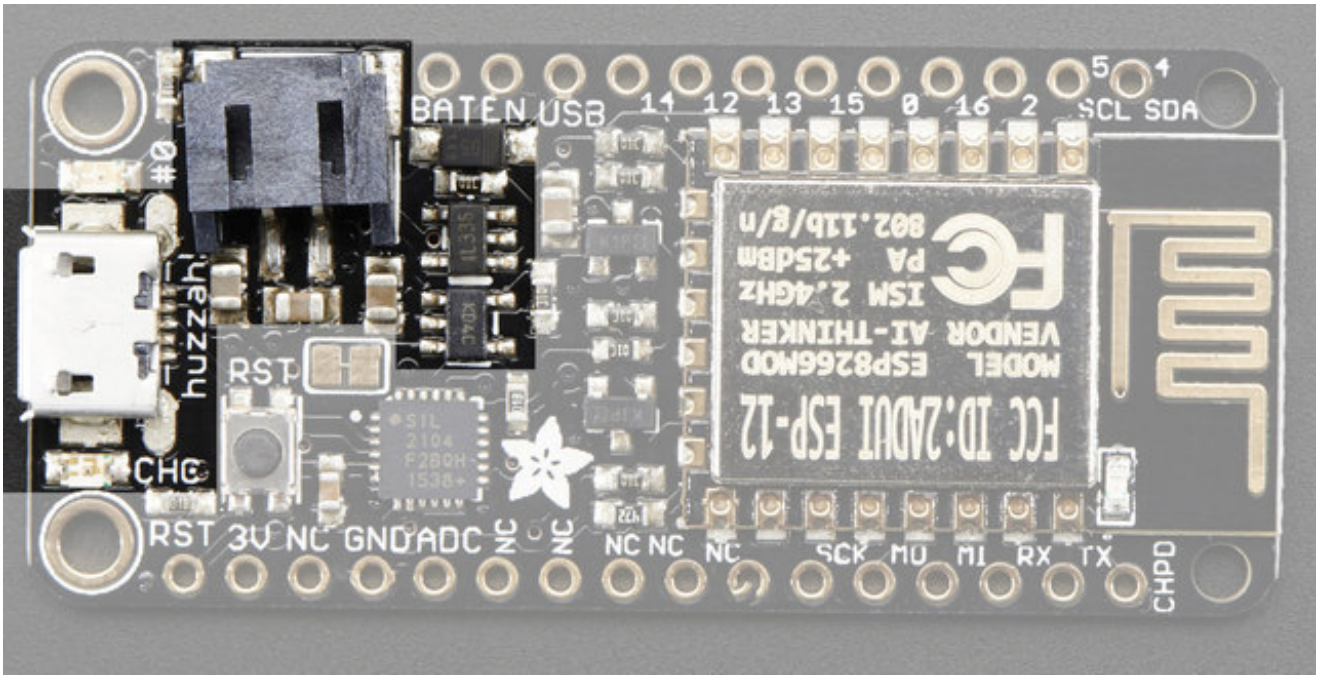
-

Power Management



Battery + USB Power

We wanted to make the Feather HUZZAH easy to power both when connected to a computer as well as via battery. There's **two ways to power** a Feather. You can connect with a MicroUSB cable (just plug into the jack) and the Feather will regulate the 5V USB down to 3.3V. You can also connect a 4.2/3.7V Lithium Polymer (Lipo/Lipoly) or Lithium Ion (LiIon) battery to the JST jack. This will let the Feather run on a rechargable 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) at 100mA.** This happens 'hotswap' style so you can always keep the LiPoly connected as a 'backup' power that will only get used when USB power is lost.



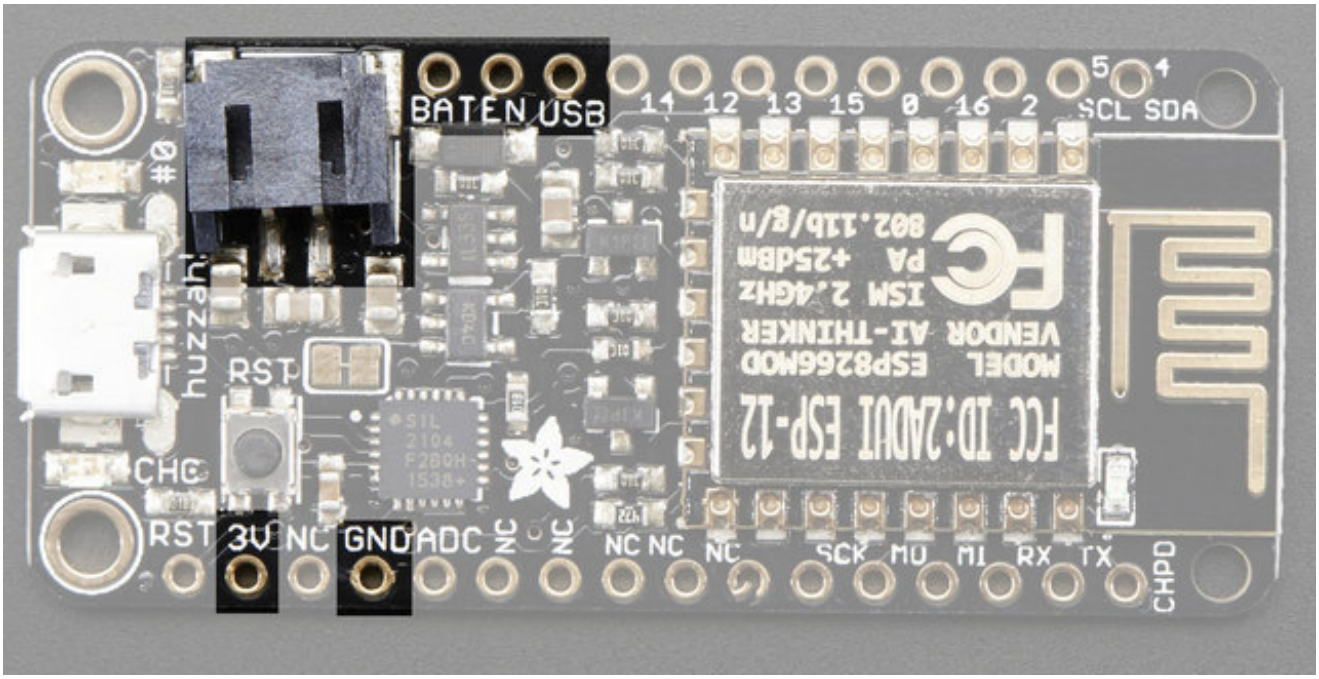
The above shows the Micro USB jack (left), Lipoly JST jack (top left), as well as the 3.3V regulator and changeover diode (just to the right of the JST jack) and the Lipoly charging circuitry (right below the regulator).

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.

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 low-dropout regulator. While you can get 500mA from it, you can't do it continuously from 5V as it will overheat the regulator. We use this to power the ESP8266 which can draw spikes of 250+mA (although its not continuous).

You should be able to budget about 250mA current available from the regulator, which will leave plenty for the WiFi module.



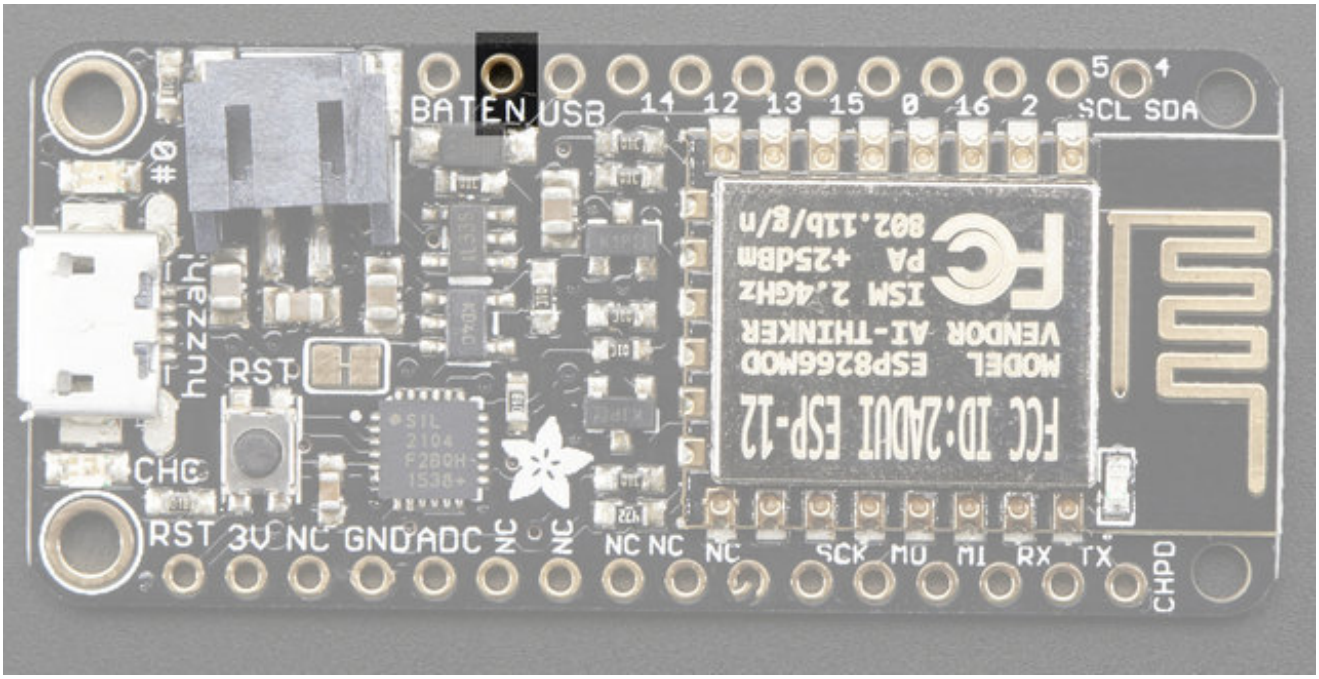
Measuring Battery

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 ESP8266 does not have multiple ADC pins, we didn't want to 'sacrifice' one for Lipoly battery monitoring. **However** we do have a tutorial that mentions how to do it, using two resistors. You can check out the wiring diagram here (use the VBat pin to measure) (http://adafru.it/jCY) and the code here (http://adafru.it/jCZ).

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


BAT EN USB 11 12 13 15 0 16 2 5 4 SCL SDA
#0
huzzah
RST
CHG
RST 3V NC GND ADC NC NC NC NC NC SCK MO MI RX TX CHPD
FCC ID:2ADUI ESP-12
MODEL ESP8266MOD
VENDOR AI-THINKER
ISM 2.4GHz
PA +25dBm
802.11b/g/n

Using NodeMCU Lua

Each Feather HUZZAH ESP8266 breakout comes pre-programmed with NodeMCU's Lua interpretter. As of this writing, we ship with **NodeMCU 0.9.5 build 20150318 powered by Lua 5.1.4** but it may be more recent

The Lua interpretter runs on the ESP8266 and you can type in commands and read out the results over serial. In order to upload code to the ESP8266 and use the serial console, connect any data-capable micro USB cable to the Feather HUZZAH and the other side to your computer's USB port. Install the required CP2104 USB driver to have the COM/Serial port appear properly (http://adafru.it/jCs)

Don't forget to visit esp8266.com for the latest and greatest in ESP8266 news, software and gossip! (http://adafru.it/f1F)

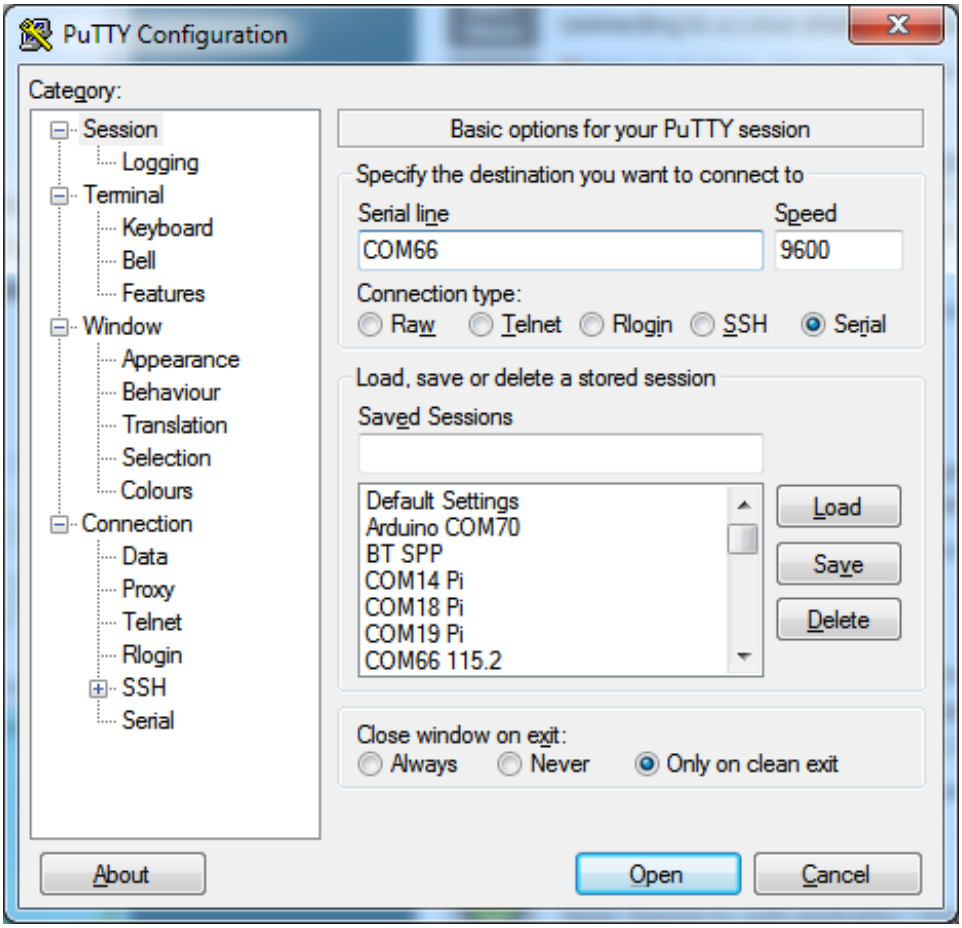
Don't forget to install the USB driver for the CP2104 USB-to-Serial chip!

Open up serial console

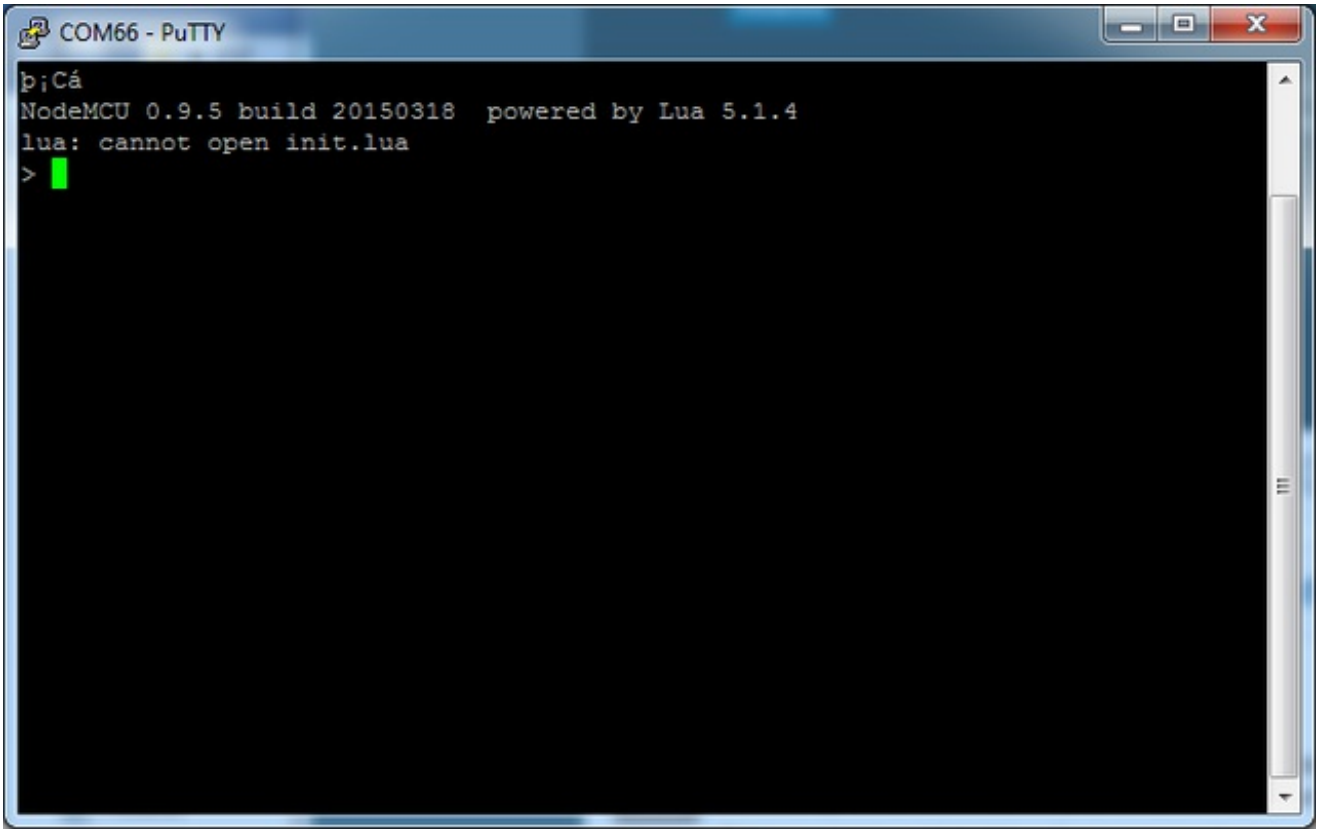
Next up, on your computer, use a serial console program such as **CoolTerm** (Mac) or **Putty** (Windows) or **screen** (linux). Teraterm seems to dislike the initial 74400bps data stream from the ESP8266 so you can try it but you'll possibly need to reset the terminal software.

Connect up to the COM or Serial port used by your cable, at 9600 Baud

Make sure you have turned off any hardware handshake or flow control



Once the terminal software is connected, click the **Reset** button on the Feather HUZZAH ESP8266 board to reset it and have it print out the welcome message:



If you don't get this message, first check that the red/blue leds flickered when you press the reset button. If they didnt, make sure you've got the right baud rate selected in the software (9600)

Hello world!

Ok we can now turn on an LED. There is a red LED on each board, connected to **GPIO #0**

> NodeMCU's pinouts are not the same as the Arduino/gcc pinouts. We print the Arduino pinouts on the board so watch out!

> The Lua documentation for the ESP8266 has GPIO #4 and #5 swapped so if #4/#5 aren't working for you, try swapping!

Pin Notes	**PCB/Arduino**	**NodeMCU/Lua**
No pullups!	0	3
	2	4
	3	9
	4	1

5	2
9	11
10	12
12	6
13	7
14	5
15	8
16	0

So to set the pin #0 LED on and off (which would be pin #3 in Lua) first make it an output:

```
gpio.mode(3, gpio.OUTPUT)
```

Turn the LED on with:

```
gpio.write(3, gpio.LOW)
```

And off with:

```
gpio.write(3, gpio.HIGH)
```

You can make this a little more automated by running:

```
while 1 do
  gpio.write(3, gpio.HIGH)
  tmr.delay(1000000)   -- wait 1,000,000 us = 1 second
  gpio.write(3, gpio.LOW)
  tmr.delay(1000000)   -- wait 1,000,000 us = 1 second
end
```

The LED will now be blinking on and off.

Note that since its in a loop, its not possible to get it to stop via the interpretter. To stop it, click the **Reset** button again!

Scanning & Connecting to WiFi

We'll continue with a quick demo of scanning for WiFi and connecting.

Once you're back at the Lua prompt, set the ESP8266 into WiFi Client mode with

```
wifi.setmode(wifi.STATION)
```

Then you can run the scanner and have it print out the available AP's

```
-- print ap list
function listap(t)
    for k,v in pairs(t) do
      print(k.." : "..v)
    end
end
wifi.sta.getap(listap)
```

or for more detail...

```
-- print ap list
function listap(t)
    for ssid,v in pairs(t) do
      authmode, rssi, bssid, channel = string.match(v, "(%d),(-?%d+),(%x%x:%x%x:%x%x:%x%x:%x%x:%x%x),(%d+
)")
      print(ssid,authmode,rssi,bssid,channel)
    end
end

wifi.sta.getap(listap)
```

We can connect to the access point with **wifi.sta.config** and **wifi.sta.connect** - it will take a second or two to complete the connection, you can query the module to ask the status with **wifi.sta.status()** - when you get a 5 it means the connection is completed and DHCP successful

```
wifi.sta.config("accesspointname","yourpassword")
wifi.sta.connect()
tmr.delay(1000000)   -- wait 1,000,000 us = 1 second
print(wifi.sta.status())
print(wifi.sta.getip())
```

WebClient example

Once you're got the IP address you can connect to adafruit, for example, and read a webpage and print it out:

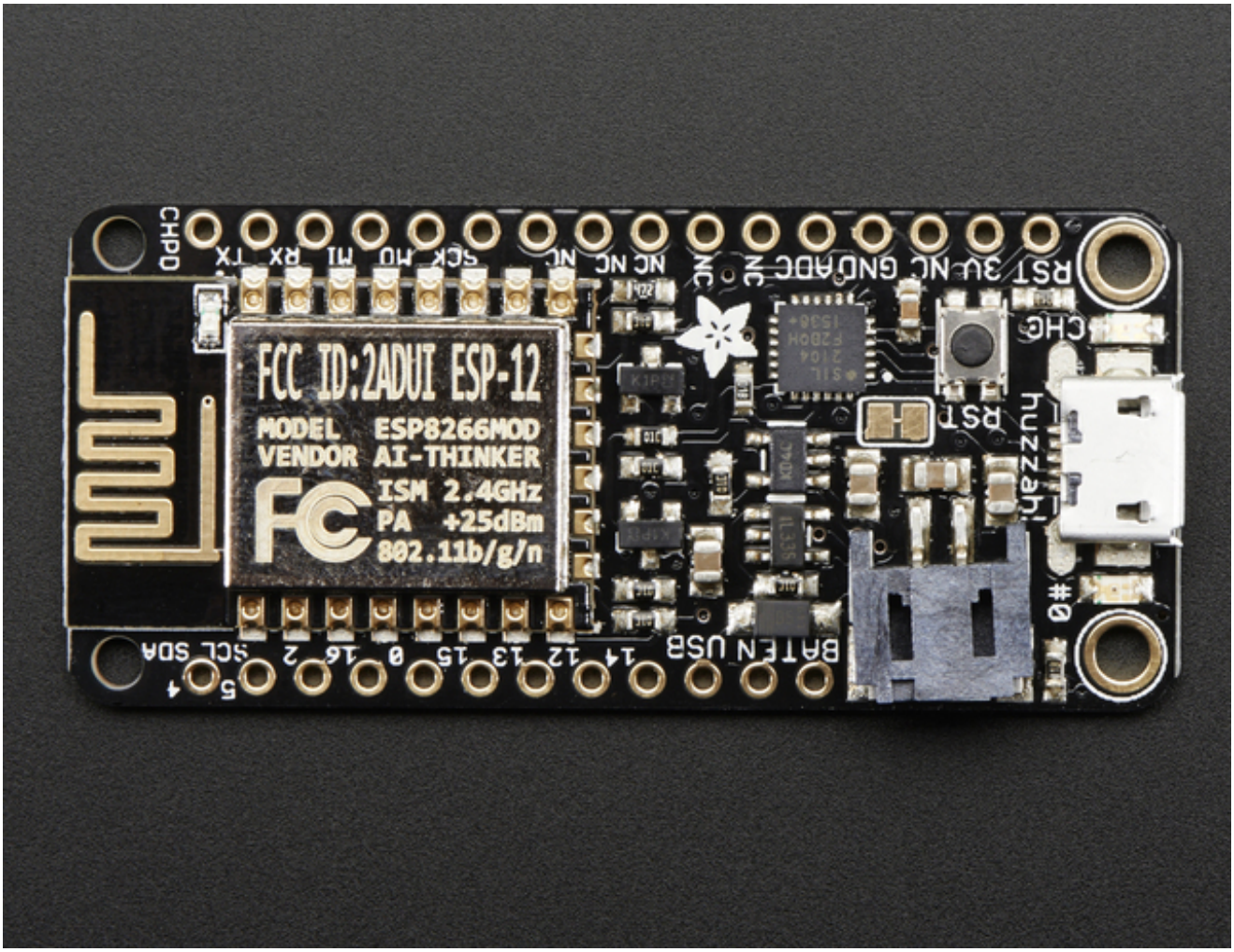
```
sk=net.createConnection(net.TCP, 0)
sk:on("receive", function(sck, c) print(c) end )
sk:connect(80,"207.58.139.247")
sk:send("GET /testwifi/index.html HTTP/1.1\r\nHost: www.adafruit.com\r\nConnection: keep-alive\r\nAccept: */*\r\n\r\n
")
```

You can also have the module do DNS for you, just give it the hostname instead of IP address:

```
sk=net.createConnection(net.TCP, 0)
sk:on("receive", function(sck, c) print(c) end )
sk:connect(80,"www.adafruit.com")
sk:send("GET /testwifi/index.html HTTP/1.1\r\nHost: www.adafruit.com\r\nConnection: keep-alive\r\nAccept: */*\r\n\r\n
")
```

This is just a light overview of testing out your HUZZAH ESP breakout! For much more, check out NodeMCU's tutorial page https://github.com/nodemcu/nodemcu-firmware/wiki/nodemcu_api_en (http://adafru.it/f1M) for the details on what functions are available to you, as well as http://www.lua.org (http://adafru.it/f1N) to learn more about the Lua scripting language

Using Arduino IDE



While the Feather HUZZAH ESP8266 comes pre-programmed with NodeMCU's Lua interpretter, you don't have to use it! Instead, you can use the Arduino IDE which may be more familar. **This will write directly to the firmware, erasing the NodeMCU firmware**, so if you want to go back to Lua, use the flasher to re-install it (http://adafru.it/f1O)

In order to upload code to the ESP8266 and use the serial console, connect any data-capable micro USB cable to the Feather HUZZAH and the other side to your computer's USB port. Install the required CP2104 USB driver to have the COM/Serial port appear properly (http://adafru.it/jCs)

Don't forget to visit esp8266.com for the latest and greatest in ESP8266 news, software and gossip! (http://adafru.it/f1F)

Don't forget to install the USB driver for the CP2104 USB-to-Serial chip!

Install the Arduino IDE 1.6.4 or greater

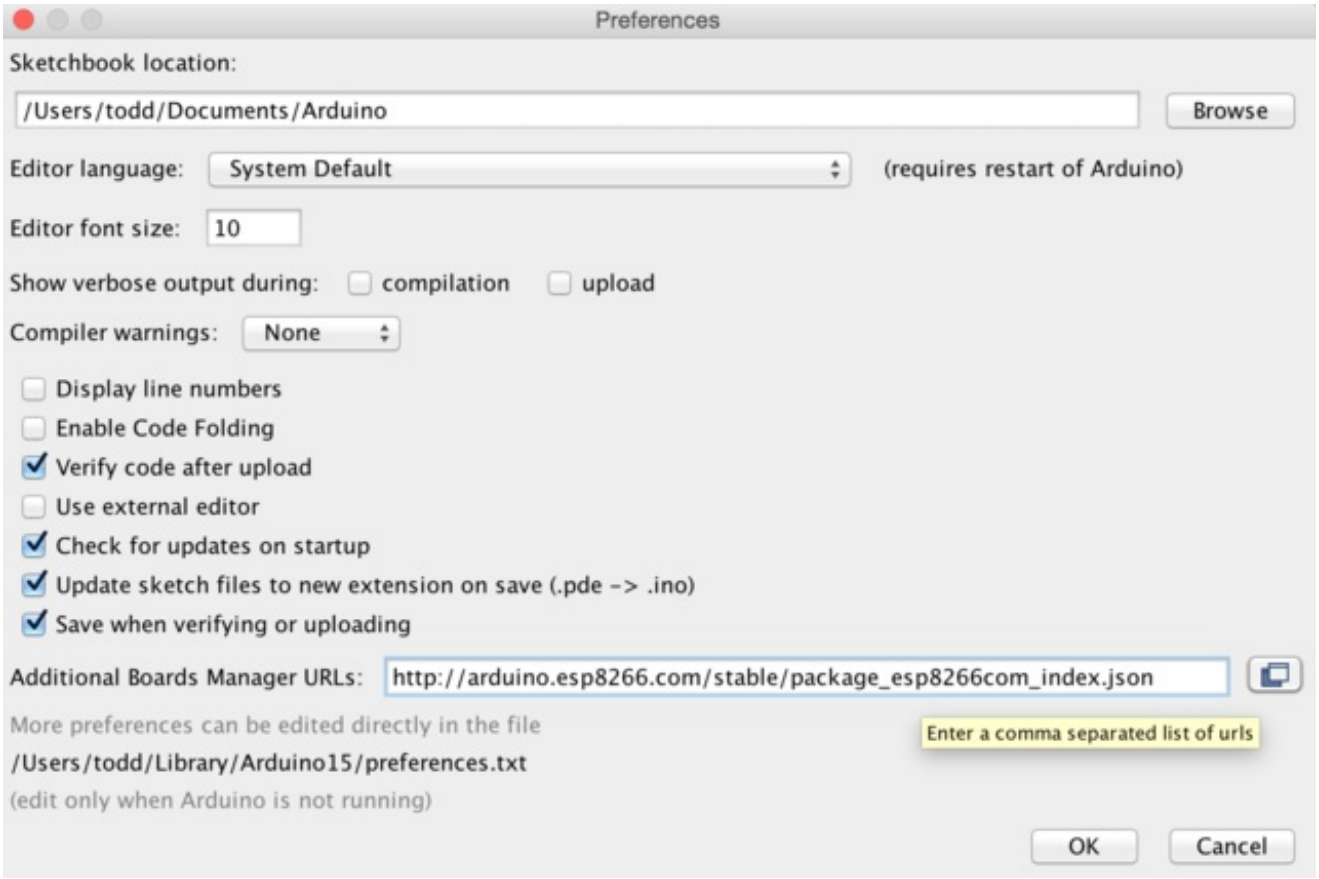
Download Arduino IDE from Arduino.cc (1.6.4 or greater) - don't use 1.6.2! You can use your existing IDE if you have already installed it (http://adafru.it/f1P)

You can also try downloading the ready-to-go package from the ESP8266-Arduino project (http://adafru.it/eSH), if the proxy is giving you problems

We're seeing some difficulties with IDE 1.6.6 so please try 1.6.5 or skip 1.6.6!

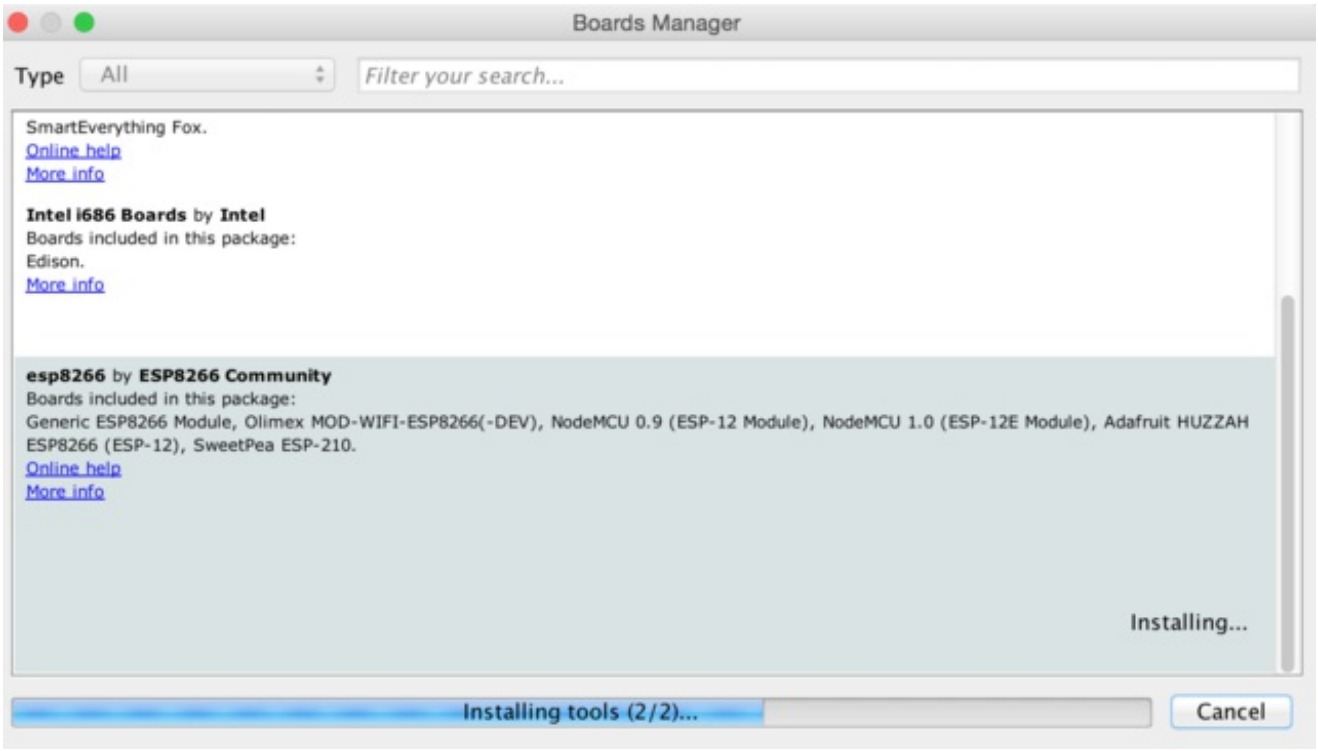
Install the ESP8266 Board Package

Enter `http://arduino.esp8266.com/stable/package_esp8266com_index.json` into *Additional Board Manager URLs* field in the Arduino v1.6.4+ preferences.



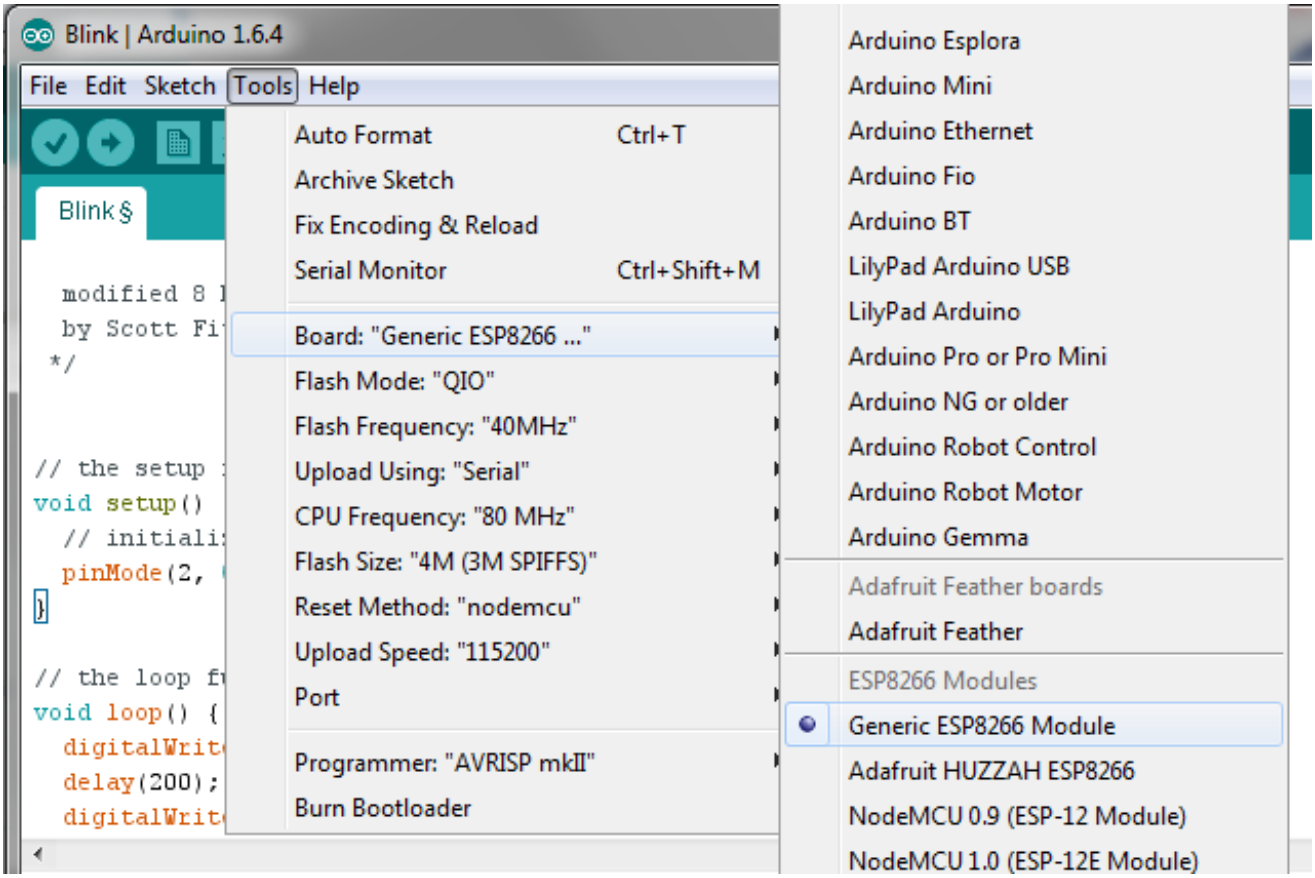
Visit our guide for how to add **new boards to the Arduino 1.6.4+ IDE** for more info about adding third party boards (http://adafru.it/f7X).

Next, use the **Board manager** to install the ESP8266 package.

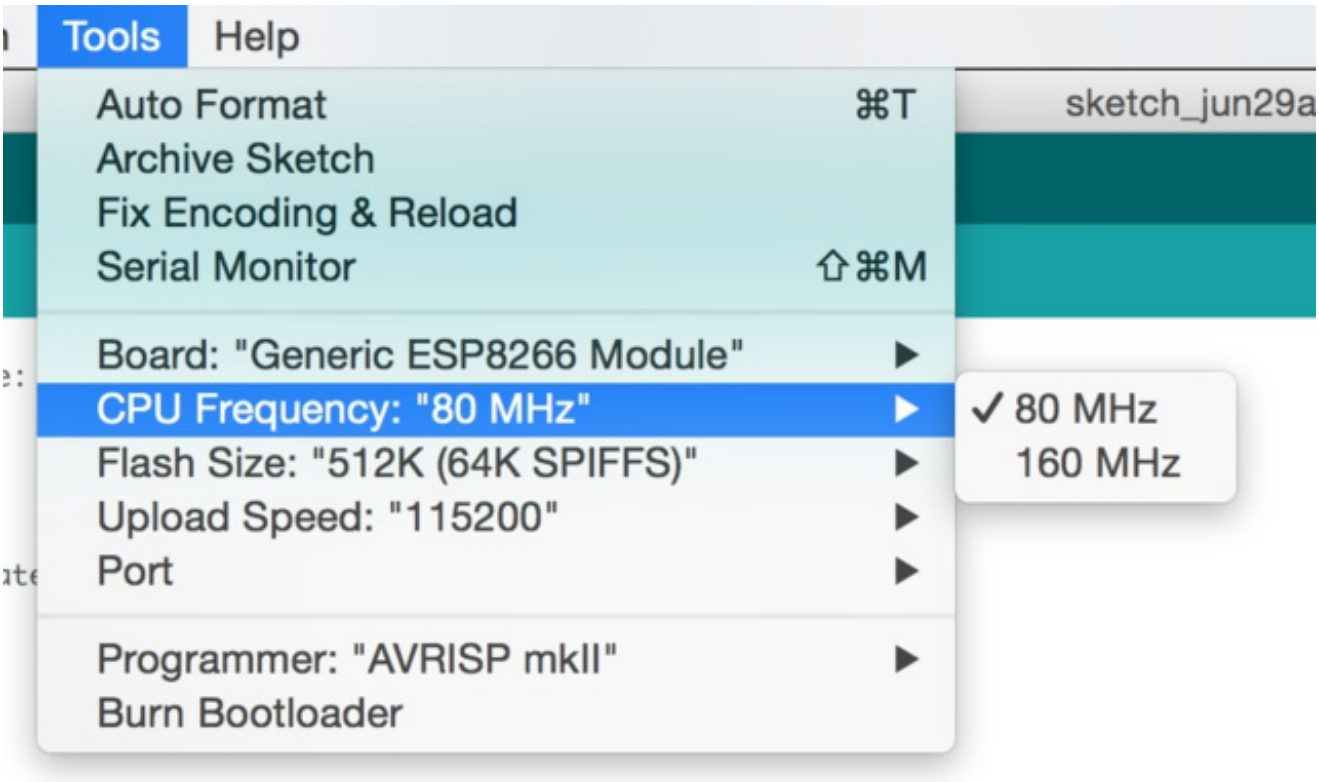


Setup ESP8266 Support

When you've restarted, select **Generic ESP8266 Module** from the Tools->Board dropdown

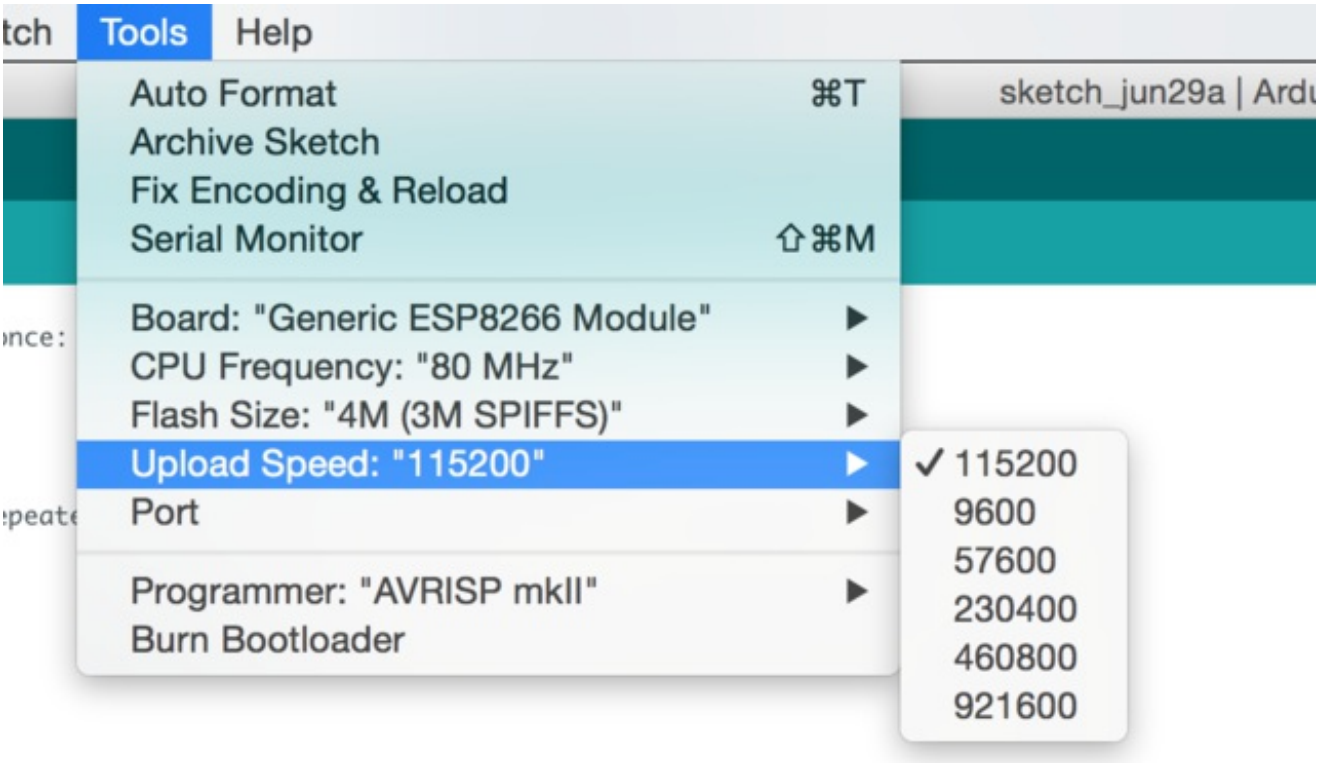


80 MHz as the CPU frequency

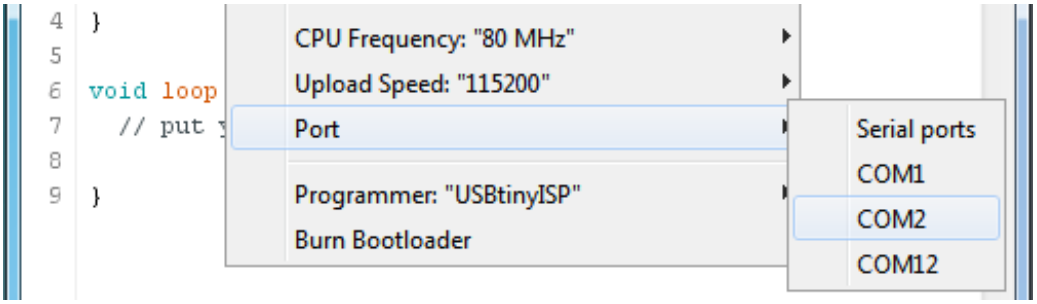


115200 baud upload speed (You can also try faster baud rates, we were able to upload at a blistering

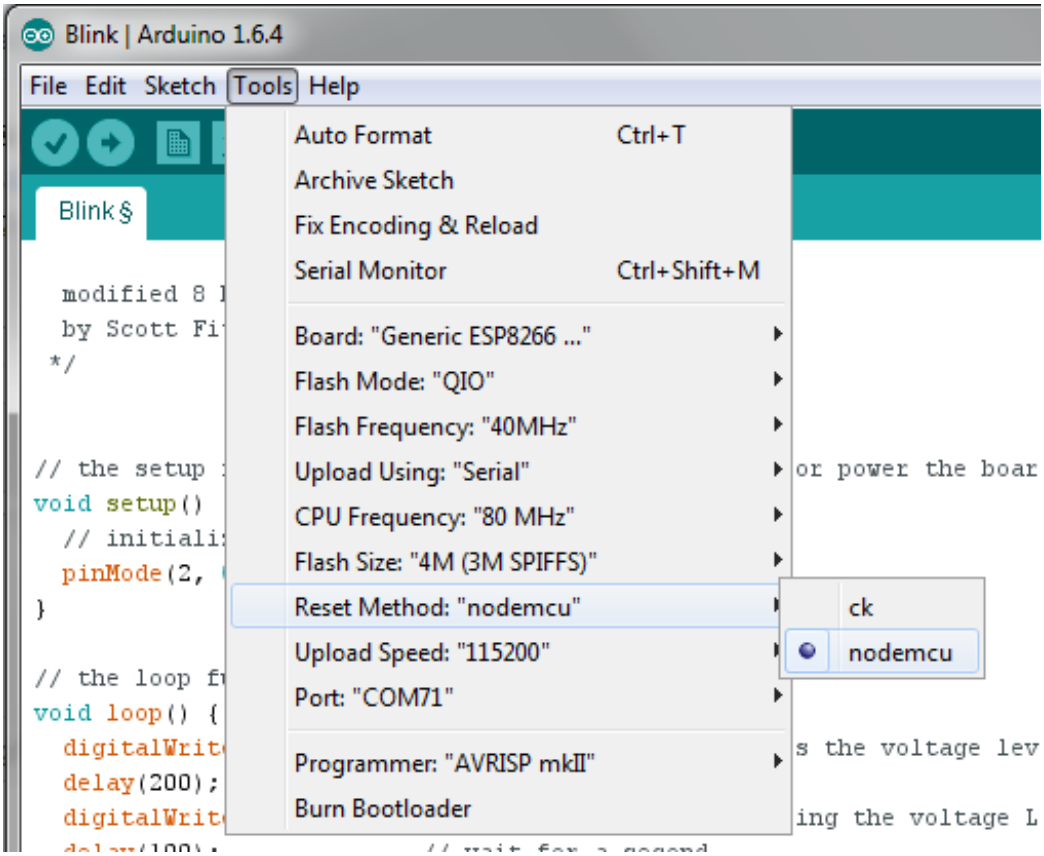
921600 baud but sometimes it fails & you have to retry)



The matching COM port for your FTDI or USB-Serial cable



and **nodemcu** as the reset method



Check also that you have

- **Flash Mode** "QIO"
- **Flash Frequency** "40MHz"
- **Upload Using** "Serial"
- **CPU Frequency** "80 MHz"
- **Flash Size** "4M"
- **Reset Method** "nodemcu"

Blink Test

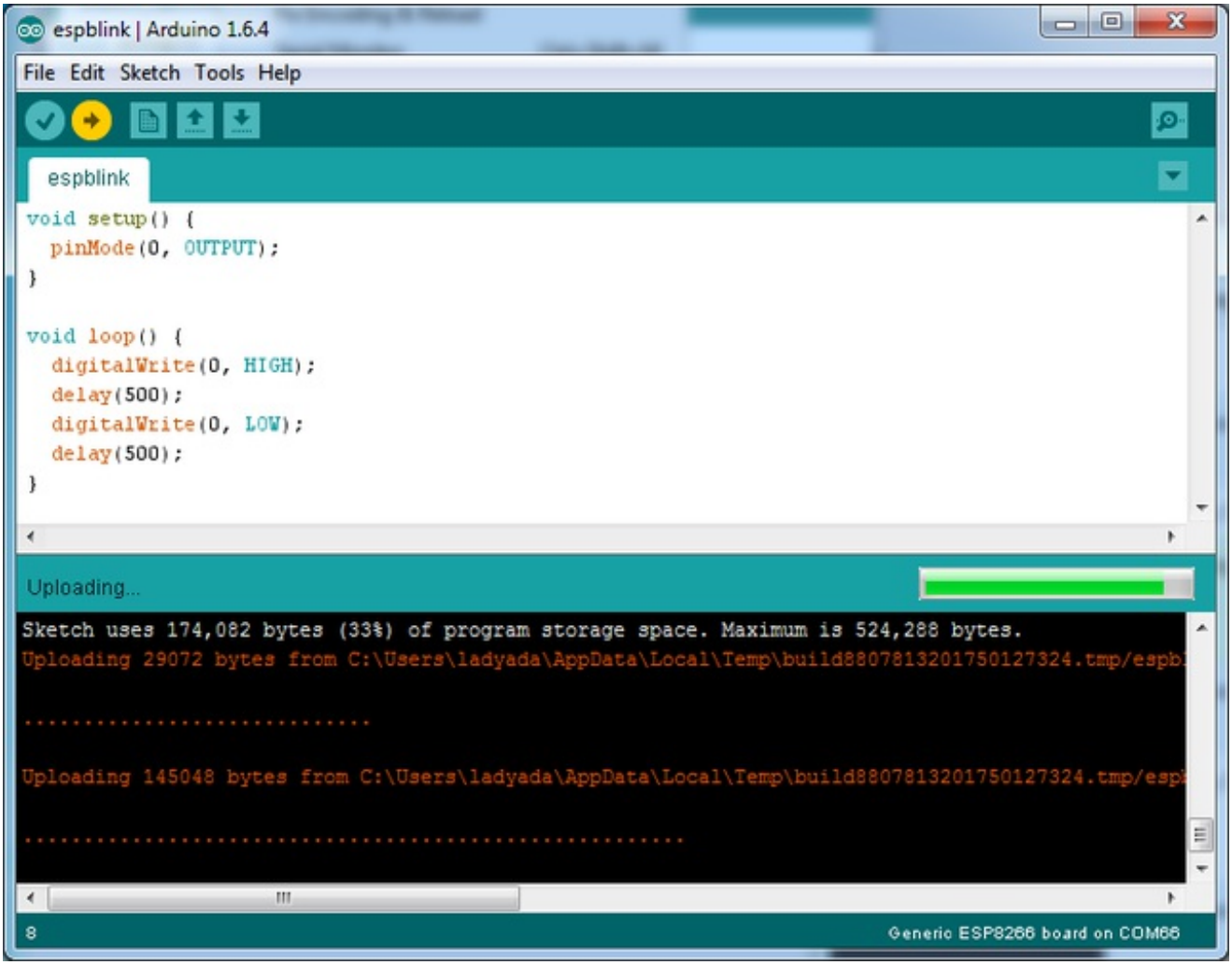
We'll begin with the simple blink test

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

```
void setup() {
  pinMode(0, OUTPUT);
}

void loop() {
  digitalWrite(0, HIGH);
  delay(500);
  digitalWrite(0, LOW);
  delay(500);
```

```
}
```

Now you can simply upload! The **Feather HUZZAH** has built in auto-reset that puts it into bootloading mode automagically



The sketch will start immediately - you'll see the LED blinking. Hooray!

Connecting via WiFi

OK once you've got the LED blinking, lets go straight to the fun part, connecting to a webserver. Create a new sketch with this code:

```
/*
 *  Simple HTTP get webclient test
 */

#include <ESP8266WiFi.h>

const char* ssid     = "yourssid";
const char* password = "yourpassword";
```

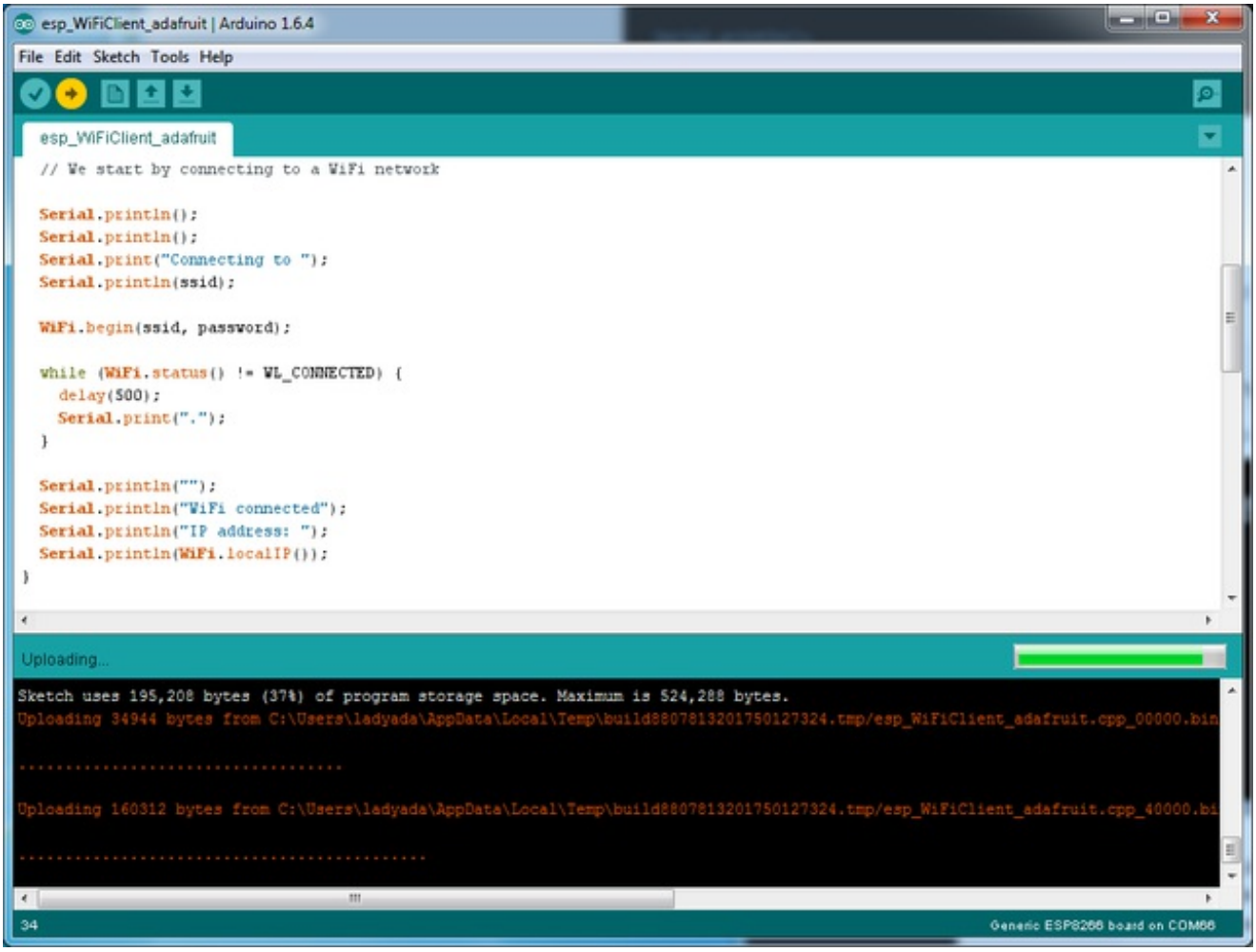
```
const char* host = "www.adafruit.com";

void setup() {
  Serial.begin(115200);
  delay(100);

  // We start by connecting to a WiFi network

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {
  delay(5000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // We now create a URI for the request
  String url = "/testwifi/index.html";
  Serial.print("Requesting URL: ");
  Serial.println(url);
```

```
  // This will send the request to the server
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  delay(500);

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");
}
```

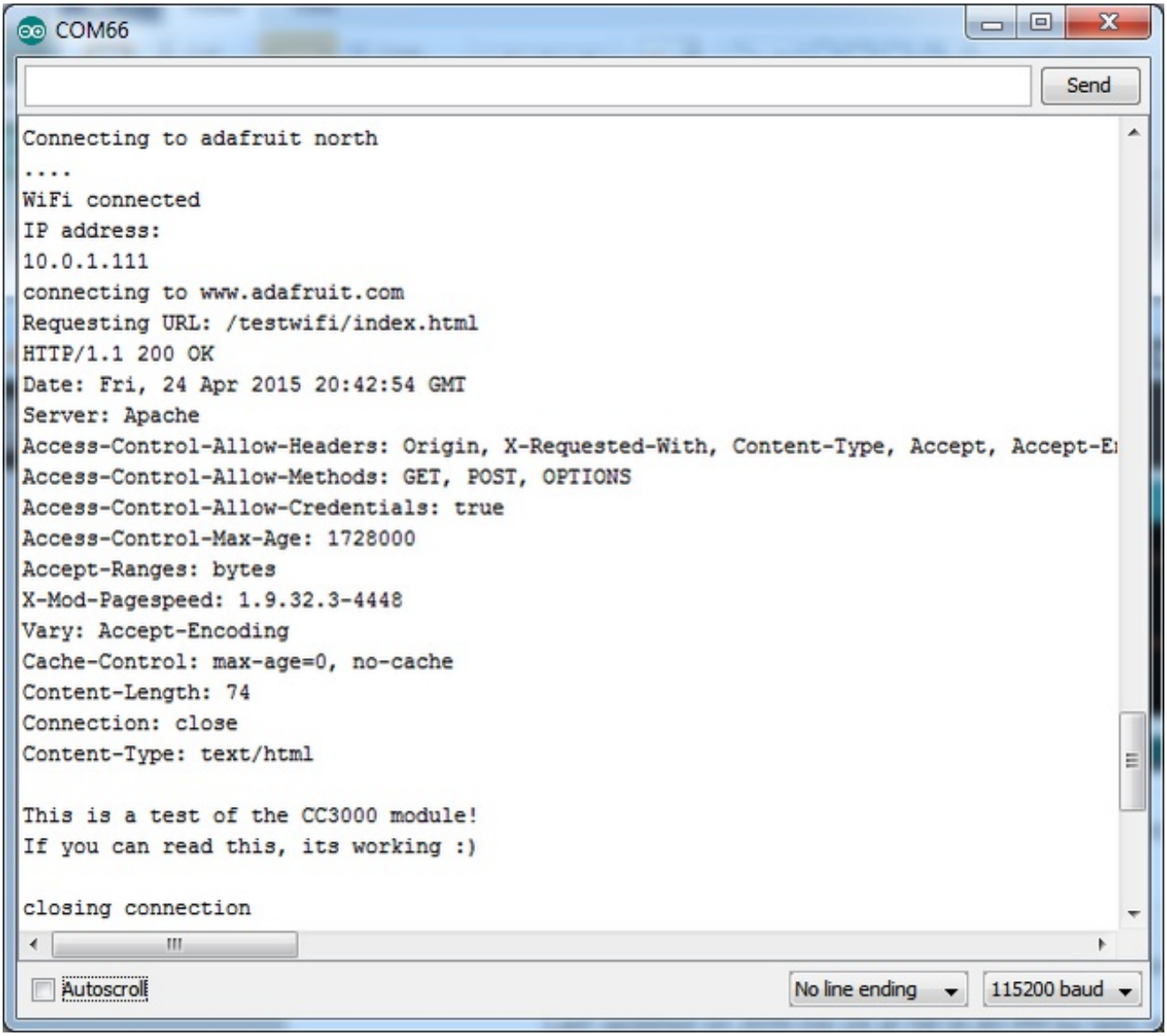
Dont forget to update

```
const char* ssid     = "yourssid";
const char* password = "yourpassword";
```

to your access point and password, then upload the same way: get into bootload mode, then upload code via IDE



Open up the IDE serial console at 115200 baud to see the connection and webpage printout!



That's it, pretty easy!

This page was just to get you started and test out your module. For more information, check out the ESP8266 port github repository (http://adafru.it/eSH) for much more up-to-date documentation!

F.A.Q.

When I connect stuff to some of the pins, the Huzzah stops working. Whats up with that?
The ESP8266 uses some of the pins as 'boot mode' pins so on boot they must be set to certain values:

- **CH_PD** should be always pulled high (it will disable the entire module if low)
- **RST** should be always pulled high (it will disable the entire module if low)
- **GPIO 0** sets whether the bootloader is active, it must be pulled HIGH during power up/reset for the user program to run. If its pulled LOW, it will activate the bootloader. The built-in red LED on #0 pulls it up
- **GPIO 2** must be pulled high on power up/reset.
- **GPIO 15** must be pulled low on power up/reset.

My Huzzah board keeps crashing and resetting, whats up with that?
The most common reason for crashes is power failure. Make sure you're powering the Huzzah with a good ~5V power supply, and if you're using a USB-Serial cable, that its plugged into the mainboard of your computer or through a **powered** hub!

I can't seem to find the Serial port on my computer for the Feather HUZZAH?
Don't forget to install the CP2104 VCP drivers (http://adafru.it/jCs) for your computer, they are required!

So, I'm getting a 'no such file' error compiling for ESP8266 on my Mac
If your error message looks like this:

fork/exec /Users/xxxxxxx/Library/Arduino15/packages/esp8266/tools/xtensa-lx106-elf-gcc/1.20.0-26-gb404fb9-2/bin/xtensa-lx106-elf-g++: no such file or directory
Error compiling.

To fix this problem, do this:

1. Open the Boards Manager in the Arduino IDE
2. Uninstall the ESP8266 support
3. go to your ~LIbrary folder (in the Finder, select "Go::Go to folder:, and enter **~Library**). Find the folder Arduino15.
4. In the Arduino15 folder, go into packages, and delete the folder **esp8266**
5. Go back to the Arduino IDE, and install ESP8266 board support.
6. Now go back to the Finder, and check that you have the xtensa-lx106-elf-g++ file in the path ***Arduino15/packages/esp8266/tools/xtensa-lx106-elf-gcc/1.20.0-26-gb404fb9-2/bin/xtensa-lx106-elf-g++***
7. That's it!

Downloads

Datasheets

- AP2112K-3.3V regulator onboard (http://adafru.it/mdb)
- CP2104 USB-to-Serial converter (http://adafru.it/jCr)

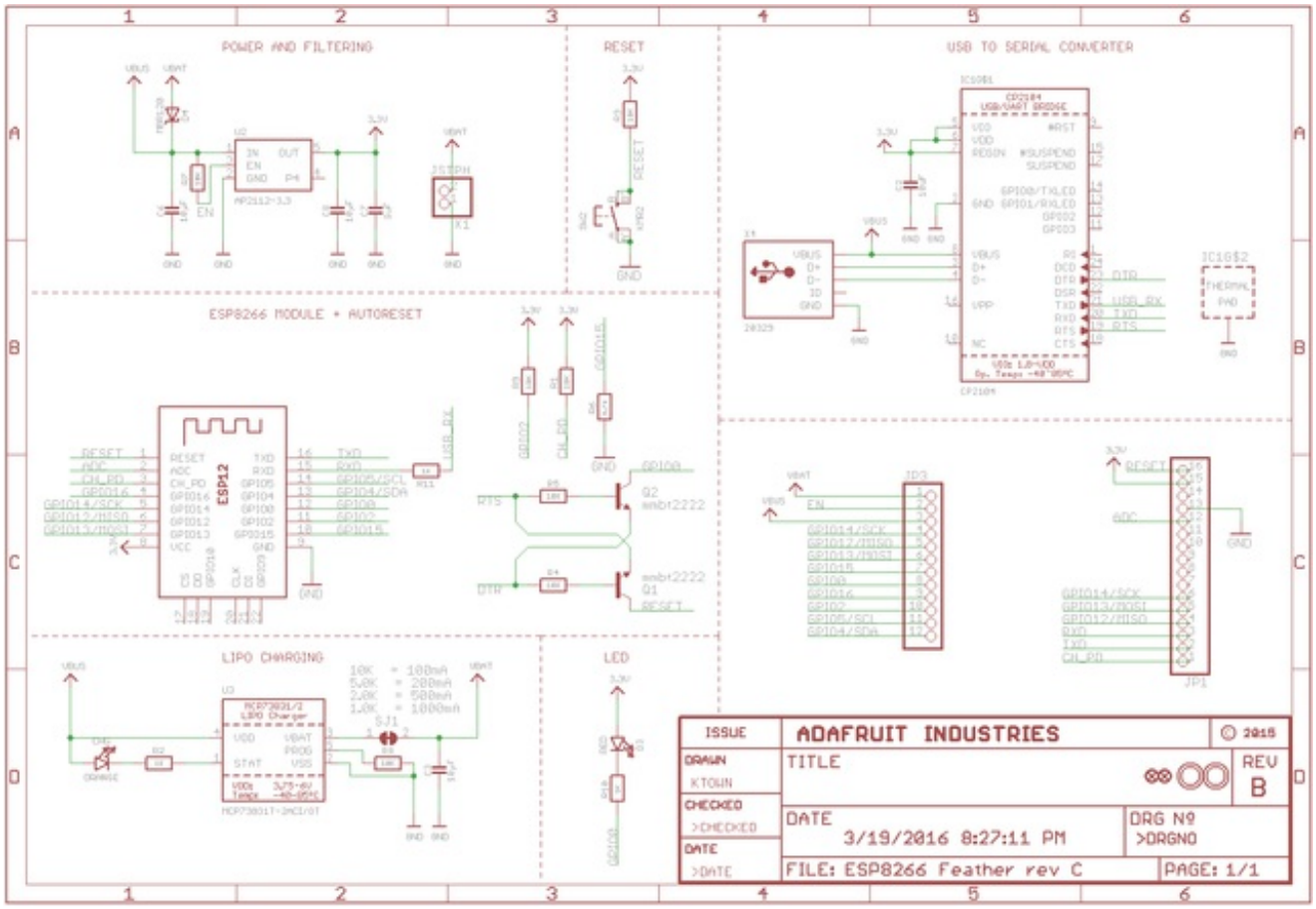
More info about the ESP8266

- ESP8266 specification sheet (http://adafru.it/f1E)
- FCC test report (http://adafru.it/f1S) for the module used on this breakout
- CE test report for the module used on this breakout (http://adafru.it/f1U)
- Huuuuge amount of information on http://www.esp8266.com/ (http://adafru.it/f1F) community forum!
- NodeMCU (Lua for ESP8266) webpage (http://adafru.it/f1G) with examples and documentation on the Lua framework
- Arduino IDE support for ESP8266 (http://adafru.it/eSH)

Don't forget to visit esp8266.com for the latest and greatest in ESP8266 news, software and gossip! (http://adafru.it/f1F)

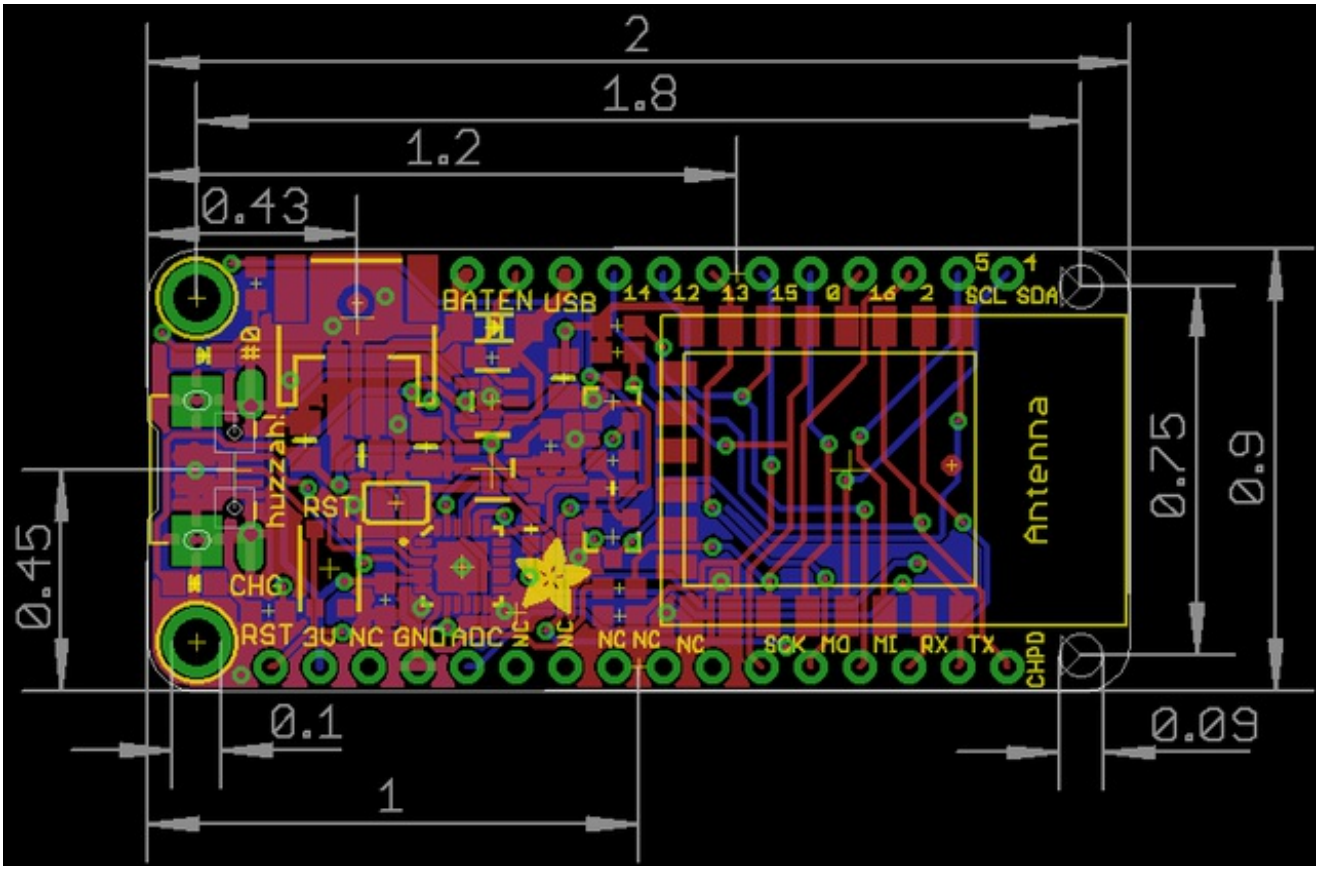
Schematic

Click to enlarge



Fabrication Print

Dimensions in inches


2
1.8
1.2
0.43
0.45
0.1
1
0.09
0.75
0.9
BATEN USB
11 12 13 15 0 16 2 5 4
SCL SDA
Antenna
RST
CHG
RST 3V NC GND ADC NC NC NC NC NC
SCK MO MI RX TX
CHPD