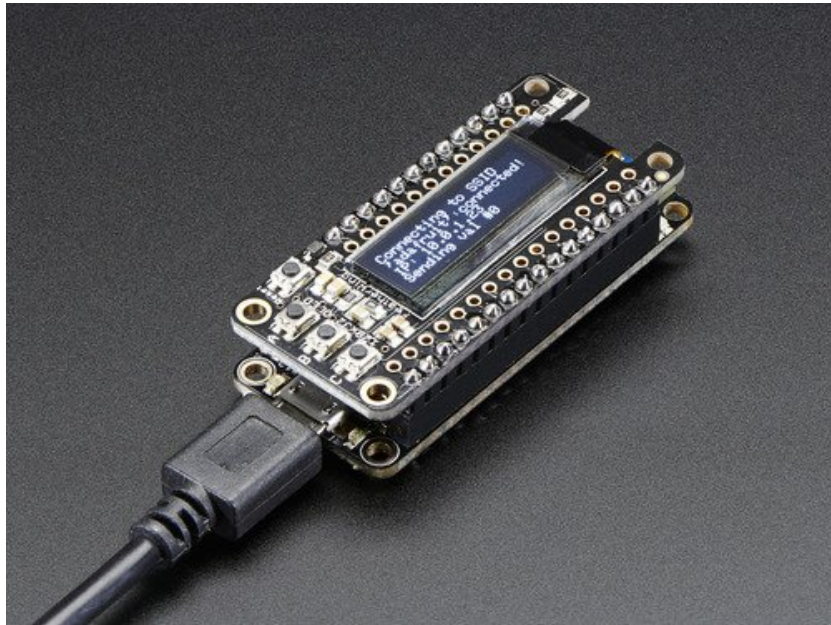


□

## Adafruit Feather M0 WiFi with ATWINC1500

Created by lady\_ada



Last updated on 2017-03-31 12:02:55 AM UTC

# Guide Contents

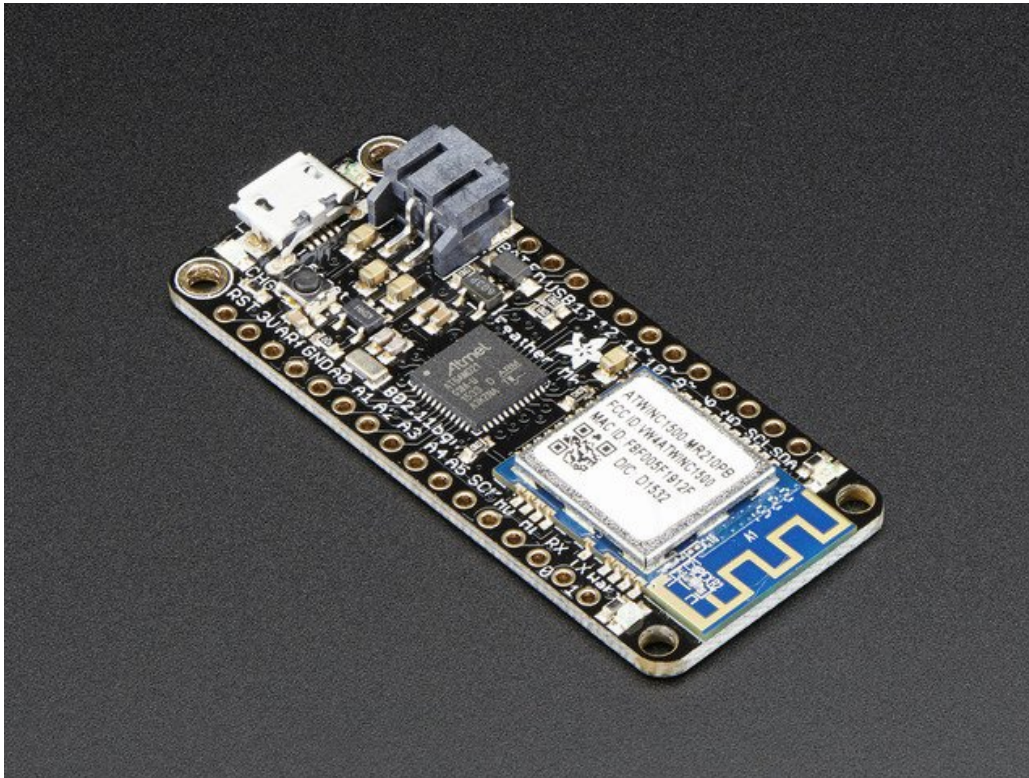
Guide Contents	2
Overview	4
Pinouts	9
Power Pins	9
Logic pins	10
WiFi Module & LEDs	10
Other Pins!	11
Assembly	13
Header Options!	13
Soldering in Plain Headers	15
Prepare the header strip:	15
Add the breakout board:	16
And Solder!	16
Soldering on Female Header	18
Tape In Place	18
Flip & Tack Solder	19
And Solder!	20
Power Management	22
Battery + USB Power	22
Power supplies	23
Measuring Battery	23
ENable pin	24
Power Usage & Saving with WiFi	24
Arduino IDE Setup	28
<a href="https://adafruit.github.io/arduino-board-index/package_adafruit_index.json">https://adafruit.github.io/arduino-board-index/package_adafruit_index.json</a>	29
Using with Arduino IDE	31
Install SAMD Support	31
Install Adafruit SAMD	32
Install Drivers (Windows Only)	33
Blink	34
Sucessful Upload	35
Compilation Issues	35

Manually bootloading	36
Ubuntu & Linux Issue Fix	36
Using the WiFi Module	37
Install the Library	37
Check Connections & Version	38
Scanning WiFi	39
Connect & Read Webpage	40
Creating an Access Point + Webserver	41
Updating Firmware	46
Updating SSL Certificates	51
Adapting Sketches to M0	55
Analog References	55
Pin Outputs & Pullups	55
Serial vs SerialUSB	55
AnalogWrite / PWM	56
Missing header files	57
Bootloader Launching	57
Aligned Memory Access	57
Floating Point Conversion	57
How Much RAM Available?	58
Storing data in FLASH	58
Feather HELP!	59
Downloads	62
Datasheets & Files	62
Schematic	62
Fabrication Print	62

# Overview

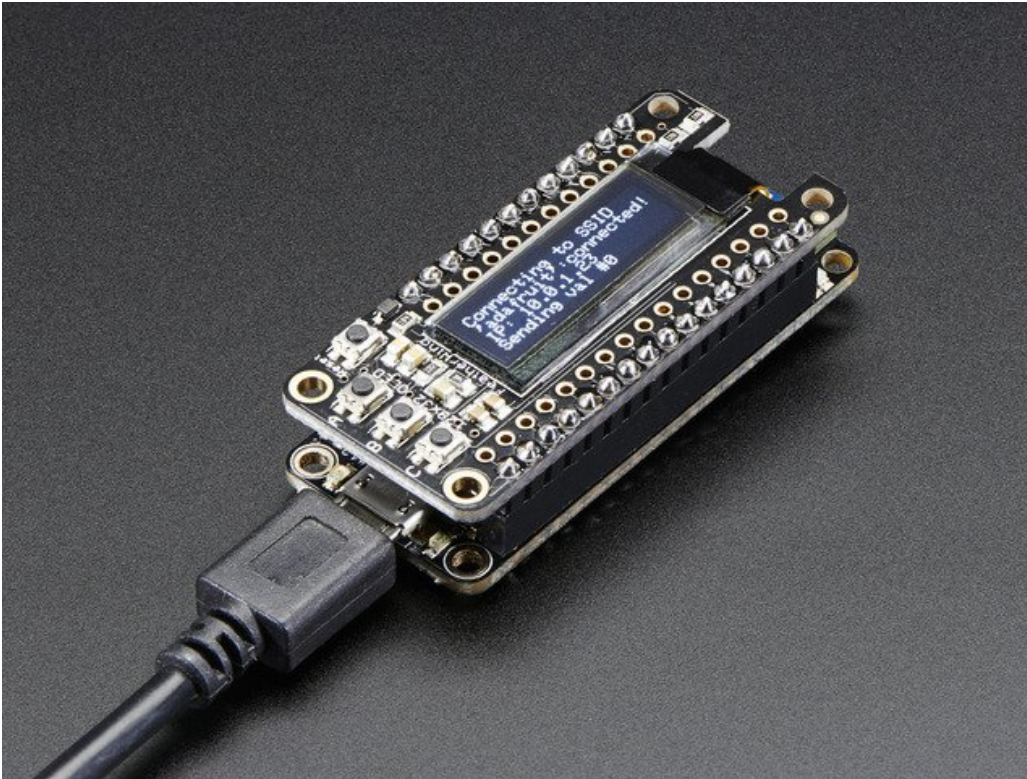
Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores.

This is the **Adafruit Feather M0 WiFi** w/ATWINC1500 - our take on an 'all-in-one' Arduino-compatible + high speed, reliable WiFi with built in USB and battery charging. Its an Adafruit Feather M0 [with a WiFi module](http://adafru.it/2999) (<http://adafru.it/2999>), ready to rock! [We have other boards in the Feather family, check'em out here](http://adafru.it/17B) (<http://adafru.it/17B>).



Connect your Feather to the Internet with this fine new FCC-certified WiFi module from Atmel. This 802.11bgn-capable WiFi module is the best new thing for networking your devices, with built-in low-power management capabilities, Soft-AP, SSL support and rock solid performance. We were running our adafruit.io MQTT demo for a full weekend straight with no hiccups (it would have run longer but we had to go to work, so we unplugged it). This module is very fast & easy to use in comparison to other WiFi modules we've used in the past.

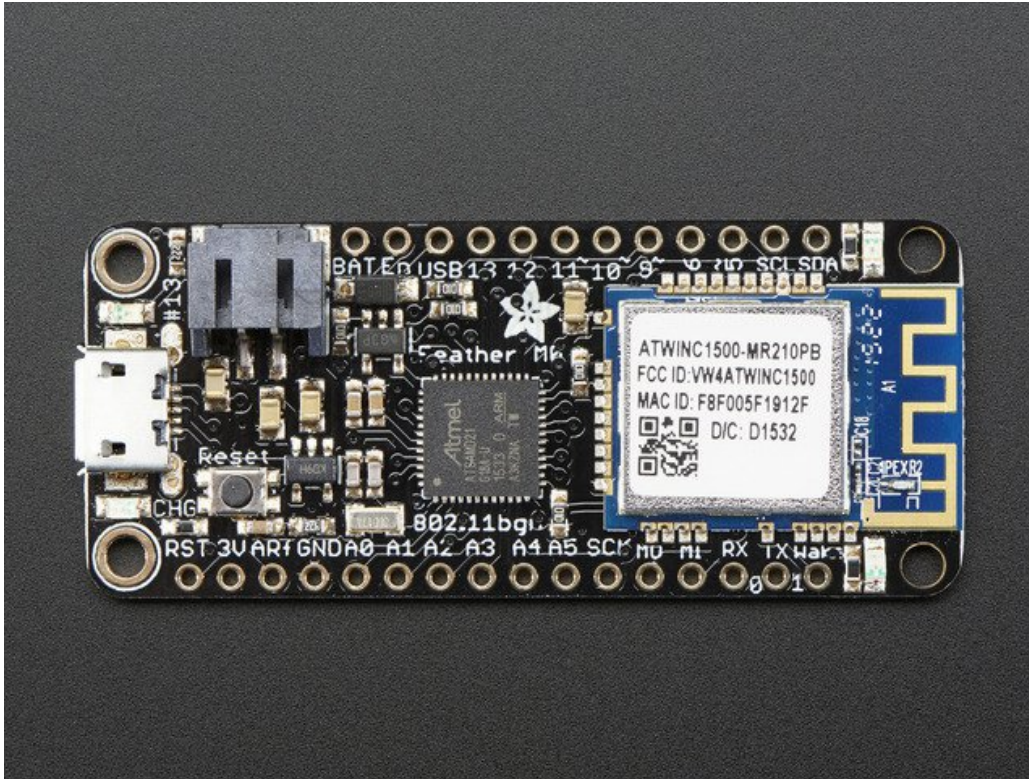
This module works with 802.11b, g, or n networks & supports WEP, WPA and WPA2 encryption. You can connect to your own WiFi networks or create your own with "Soft AP" mode, where it becomes its own access point (we have an example of it creating a webserver that you can then control the Arduino's pins). You can clock it as fast as 12MHz for speedy, reliable packet streaming. And scanning/connecting to networks is very fast, just a second or two.



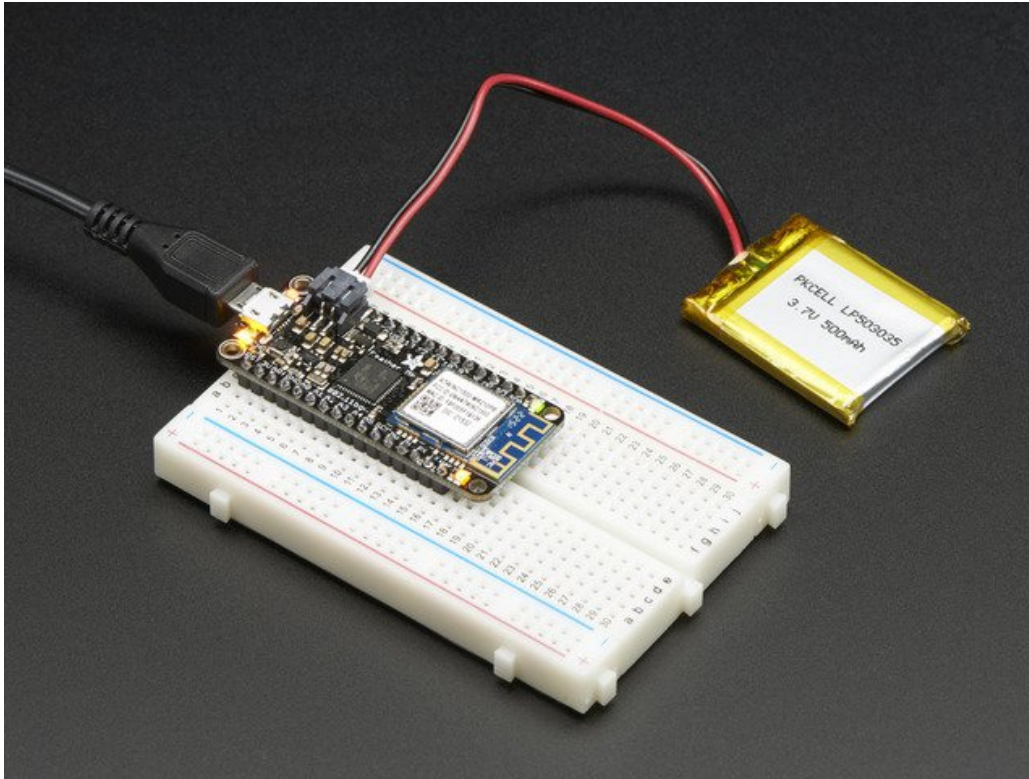
You might be wondering why use this [when you can get a HUZZAH Feather?](http://adafruit.it/2821) (<http://adafruit.it/2821>) Well, you get

- A highly-capable Cortex M0+ processor with ton more I/O pins, lots of 12-bit ADCs, a 10-bit DAC, 6 total SERCOMs that can each do SPI, I2C or UART (3 are used by the existing interfaces, leaving you 3), plenty of timers, PWMs, DMA, native USB, and more ([check out the Datasheet](http://adafruit.it/l3e) (<http://adafruit.it/l3e>))
- The ATWINC has much lower power usage, about 12mA for the WINC & 10mA for the ATSAM21 with auto-powermanagement on for the WiFi and no power management for the ARM. With manual power management, you can get the WiFi module to down to ~2mA by putting it to sleep. This is compared to the ESP's ~70mA average current draw, and whose deep sleep mode requires a WDT reset.
- We also found that we could stream more reliably (less 'bursty') with the ATWINC, although altogether the ESP has higher throughput.
- You also dont have to 'yield' all the time to the WiFi core, since its a separate chip. You get full reign of the processor and timing

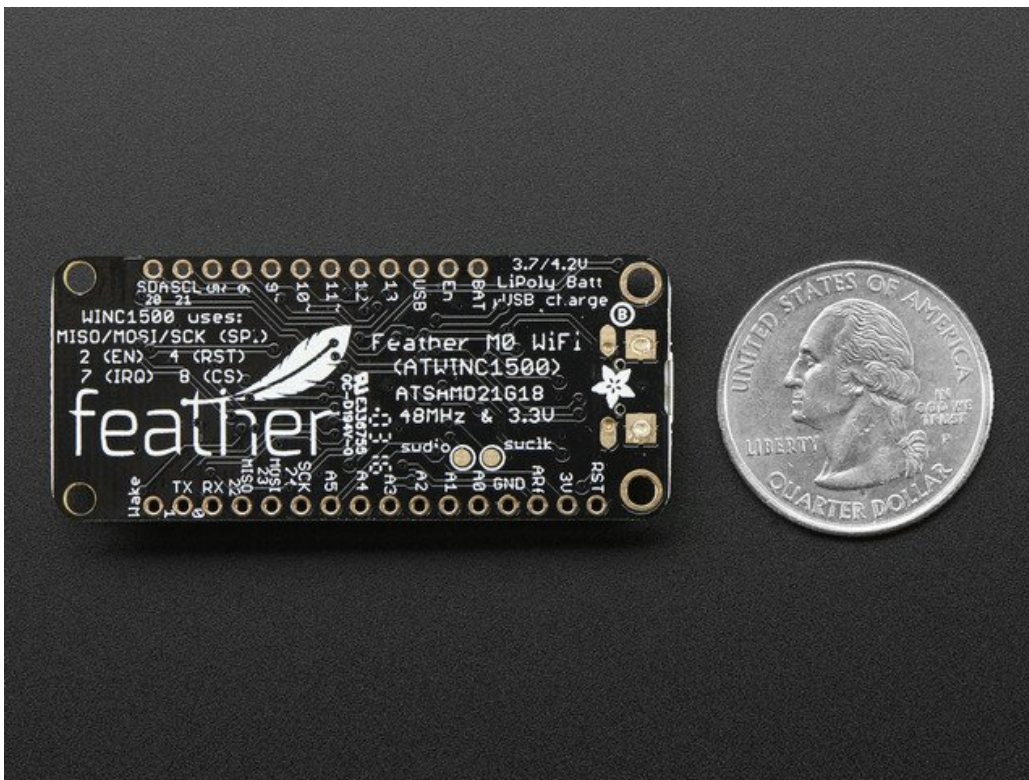
Of course, both WiFi-capable Feathers have their strengths and tradeoffs, & we love both equally!



At the Feather M0's heart is an ATSAM21G18 ARM Cortex M0 processor, clocked at 48 MHz and at 3.3V logic, the same one used in the new [Arduino Zero](http://adafruit.it/2843) (<http://adafruit.it/2843>). This chip has a whopping 256K of FLASH (8x more than the Atmega328 or 32u4) and 32K of RAM (16x as much)! This chip comes with built in USB so it has USB-to-Serial program & debug capability built in with no need for an FTDI-like chip. For advanced users who are comfortable with ASF, the SWDIO/SWCLK pins are available on the bottom, and when connected to a CMSIS-DAP debugger can be used to use Atmel Studio for debugging.

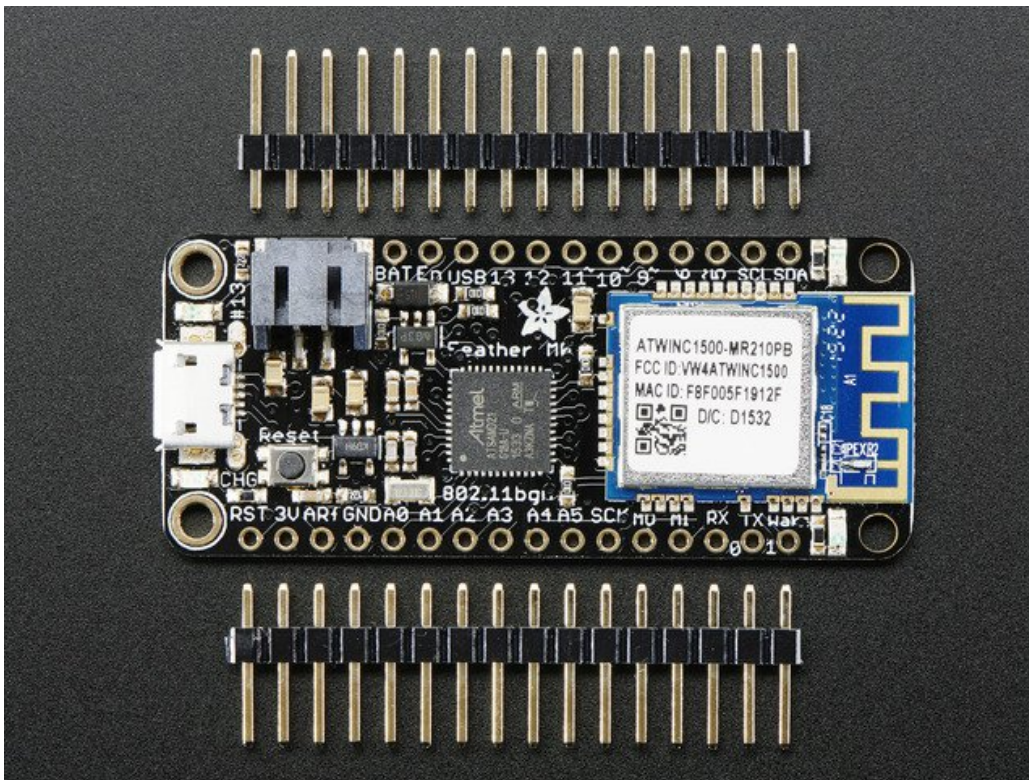


To make it easy to use for portable projects, we added a connector for any of our 3.7V Lithium polymer batteries and built in battery charging. You don't need to use a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then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



## Here's some handy specs! Like all Feather M0's you get:

- Measures 2.1" x 0.9" x 0.3" (53.65mm x 23mm x 8mm) without headers soldered in. Note it is 0.1" longer than most Feathers
- Light as a (large?) feather - 6.1 grams
- ATSAM21G18 @ 48MHz with 3.3V logic/power
- 256KB FLASH, 32KB SRAM, No EEPROM
- 3.3V regulator (AP2112K-3.3) with 600mA peak current output, WiFi can draw 300mA peak during xmit
- USB native support, comes with USB bootloader and serial port debugging
- You also get tons of pins - 20 GPIO pins
- Hardware Serial, hardware I2C, hardware SPI support
- 8 x PWM pins
- 10 x analog inputs
- 1 x analog output
- Built in 200mA lipoly charger with charging status indicator LED
- Pin #13 red LED for general purpose blinking
- Power/enable pin
- 4 mounting holes
- Reset button

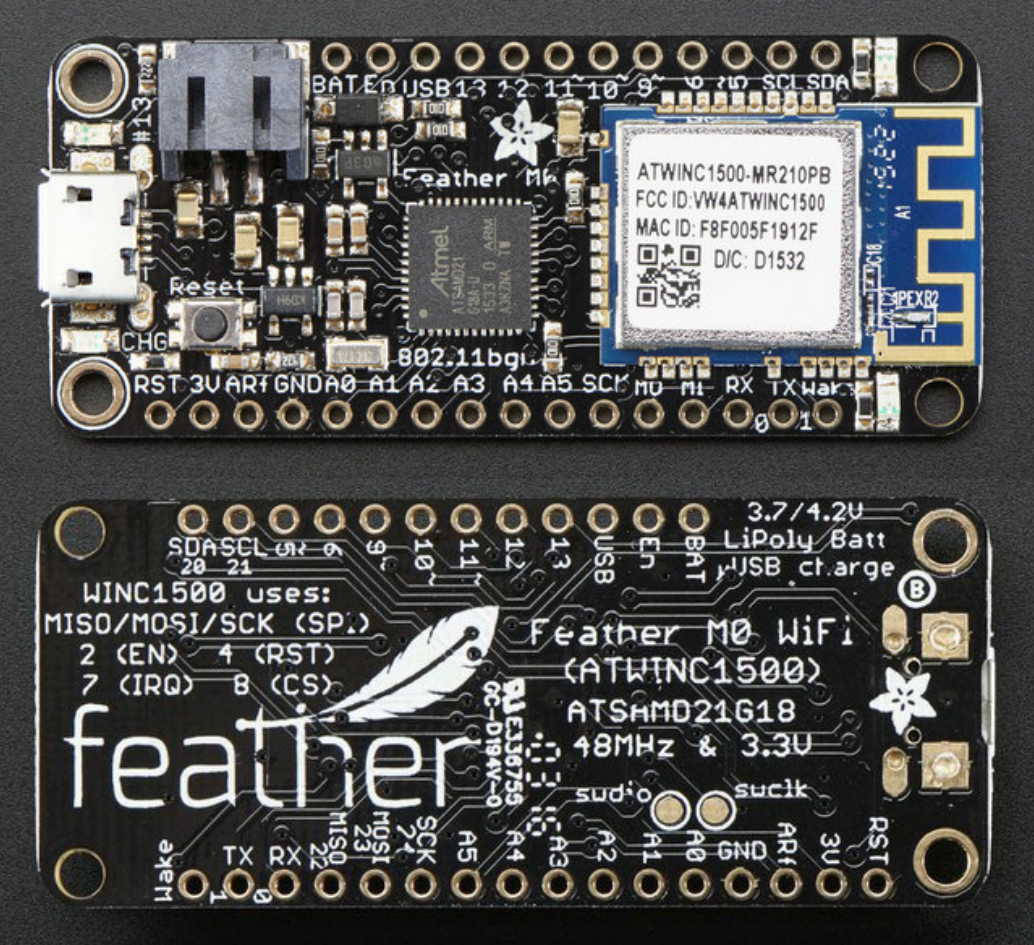


Comes fully assembled and tested, with a USB bootloader that lets you quickly use it with the Arduino IDE. We also toss in some header so you can solder it in and plug into a solderless breadboard. [Lipoly battery \(http://adafruit.com/products/2512\)](http://adafruit.com/products/2512) and [MicroUSB cable \(http://adafruit.com/products/2513\)](http://adafruit.com/products/2513) not included (but we do have lots of options in the shop if you'd like!)

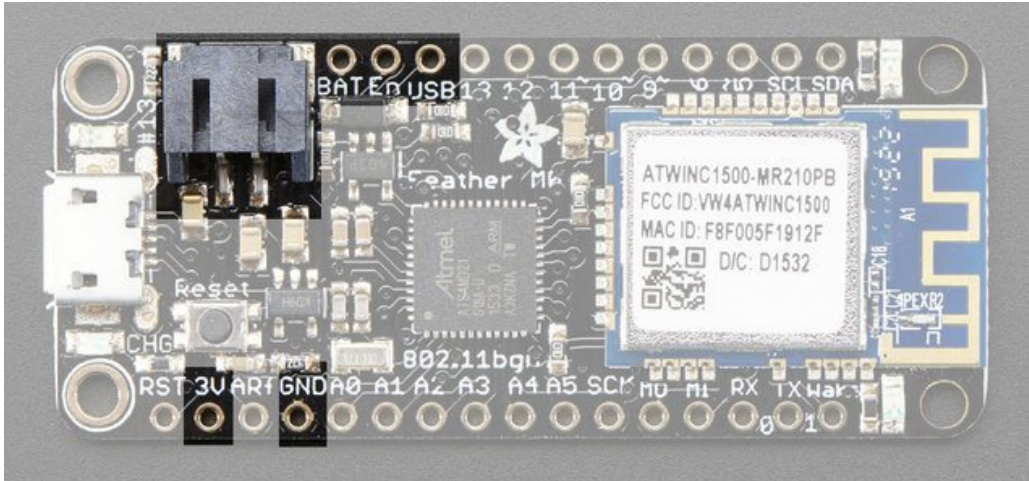


# Pinouts

The Feather M0 Adalogger is chock-full of microcontroller goodness. There's also a lot of pins and ports. We'll take you a tour of them now!



# 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 600mA peak

## Logic pins

This is the general purpose I/O pin set for the microcontroller.

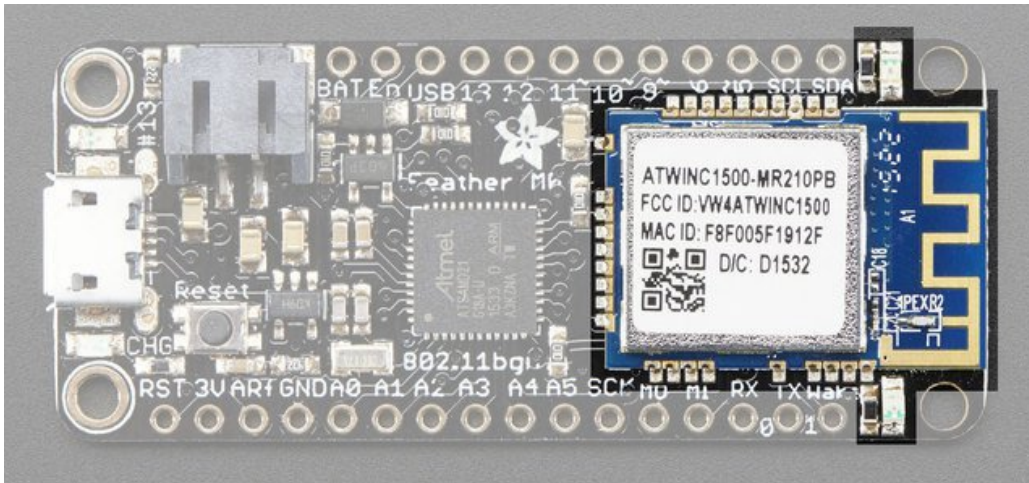
**All logic is 3.3V**

**All pins can do PWM output**

**All pins can be interrupt inputs**

- **#0 / RX** - GPIO #0, also receive (input) pin for **Serial1** (hardware UART), also can be analog input
- **#1 / TX** - GPIO #1, also transmit (output) pin for **Serial1**, also can be analog input
- **#20 / SDA** - GPIO #20, also the I2C (Wire) data pin. There's no pull up on this pin by default so when using with I2C, you may need a 2.2K-10K pullup.
- **#21 / SCL** - GPIO #21, also the I2C (Wire) clock pin. There's no pull up on this pin by default so when using with I2C, you may need a 2.2K-10K pullup.
- **#5** - GPIO #5
- **#6** - GPIO #6
- **#9** - GPIO #9, also analog input **A7**. This analog input is connected to a voltage divider for the lipoly battery so be aware that this pin naturally 'sits' at around 2VDC due to the resistor divider
- **#10** - GPIO #10
- **#11** - GPIO #11
- **#12** - GPIO #12
- **#13** - GPIO #13 and is connected to the **red LED** next to the USB jack
- **A0** - This pin is analog *input* **A0** but is also an analog *output* due to having a DAC (digital-to-analog converter). You can set the raw voltage to anything from 0 to 3.3V, unlike PWM outputs this is a true analog output
- **A1 thru A5** - These are each analog input as well as digital I/O pins.
- **SCK/MOSI/MISO** (GPIO 24/23/22) - These are the hardware SPI pins, you can use them as everyday GPIO pins (but recommend keeping them free as they are best used for hardware SPI connections for high speed)

## WiFi Module & LEDs

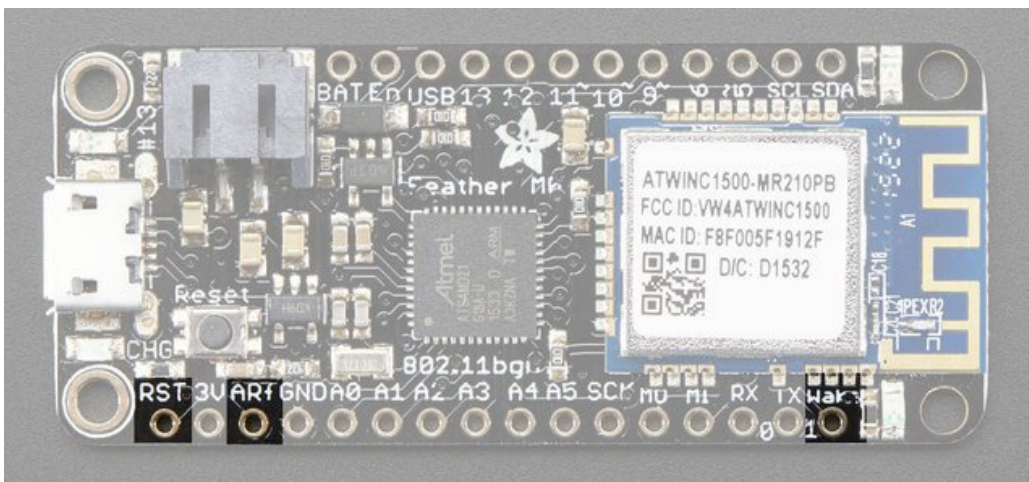


Since not all pins can be brought out to breakouts, due to the small size of the Feather, we use these to control the WiFi module

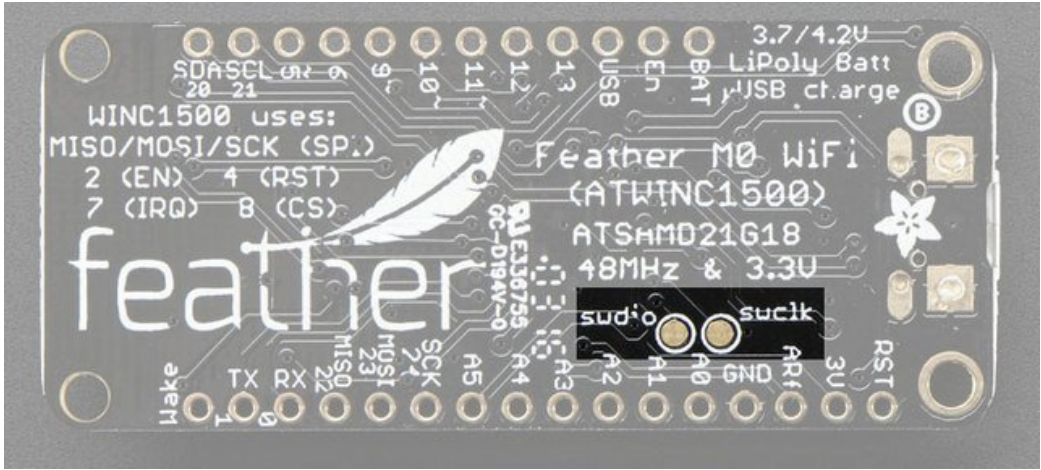
- **#2** - used as the **EN**able pin for the WiFi module, by default pulled down low, set HIGH to enable WiFi
- **#4** - used as the **Reset** pin for the WiFi module, controlled by the library
- **#7** - used as the **IRQ** interrupt request pin for the WiFi module, controlled by the library
- **#8** - used as the **Chip Select** pin for the WiFi module, used to select it for SPI data transfer
- **MOSI / MISO / SCK** - the SPI pins are also used for WiFi module communication
- **Green LED** - the top LED, in green, will light when the module has connected to an SSID
- **Yellow LED** - the bottom LED, in yellow, will blink during data transfer

## Other Pins!

- **RST** - this is the Reset pin, tie to ground to manually reset the AVR, as well as launch the bootloader manually
- **AREf** - the analog reference pin. Normally the reference voltage is the same as the chip logic voltage (3.3V) but if you need an alternative analog reference, connect it to this pin and select the external AREF in your firmware. Can't go higher than 3.3V!
- **Wake** - connected to the Wake pin on the WiFi module, not used at this time but it's there if you want it



**SWCLK & SWDIO** - These pads on the bottom are used to program the chip. They can also be connected to an SWD debugger.

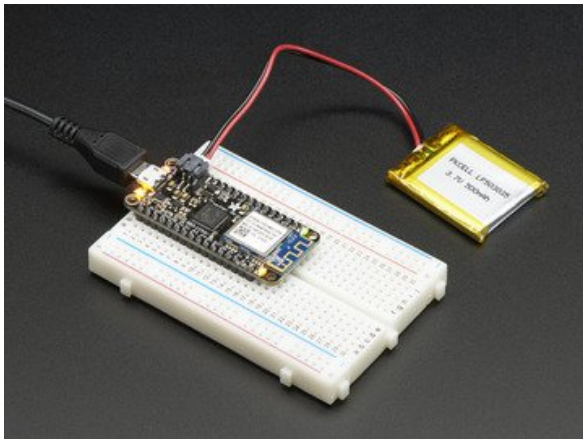


# Assembly

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

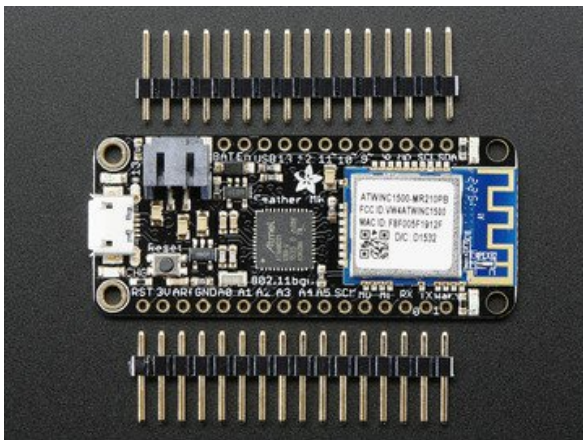
## Header Options!

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

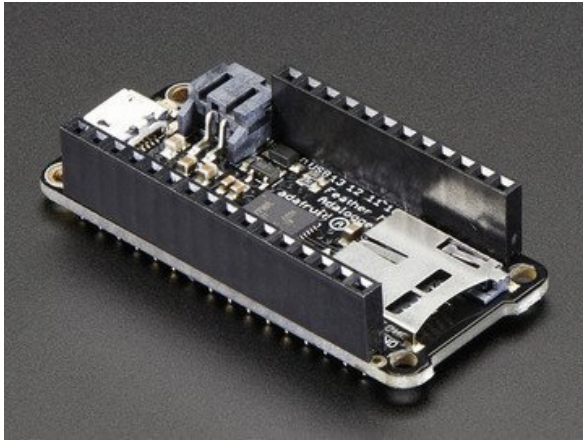


- 

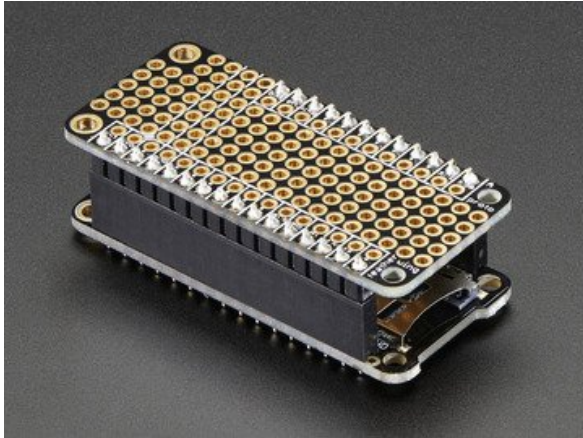
The first option is soldering in plain male headers, this lets you plug in the Feather into a solderless breadboard



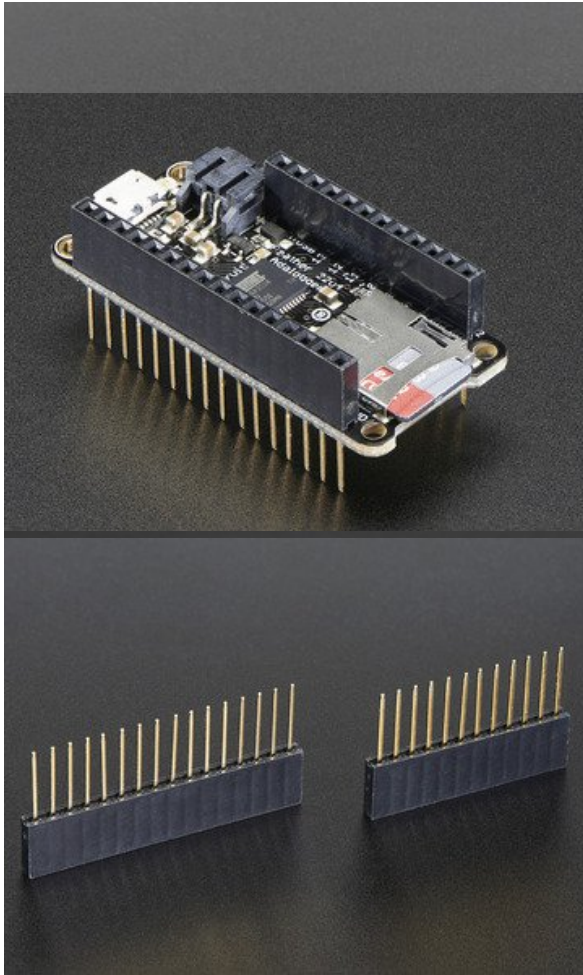
-



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

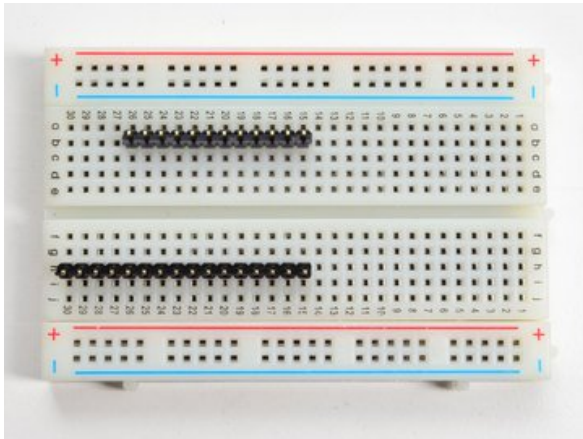


We also have 'slim' versions of the female headers, that are a little shorter and give a more compact shape



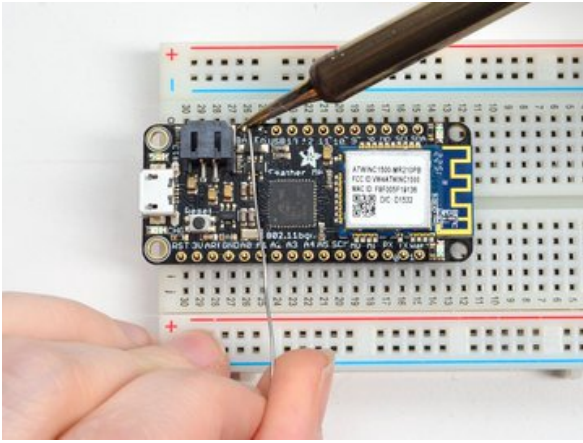
Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a solderless breadboard *and* plug a featherwing on top. But its a little bulky

## Soldering in Plain Headers



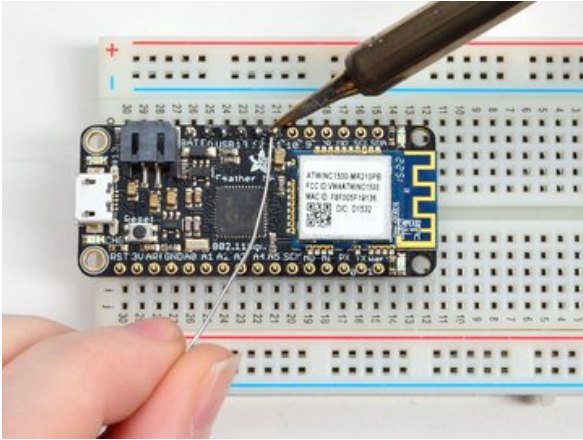
### Prepare the header strip:

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:

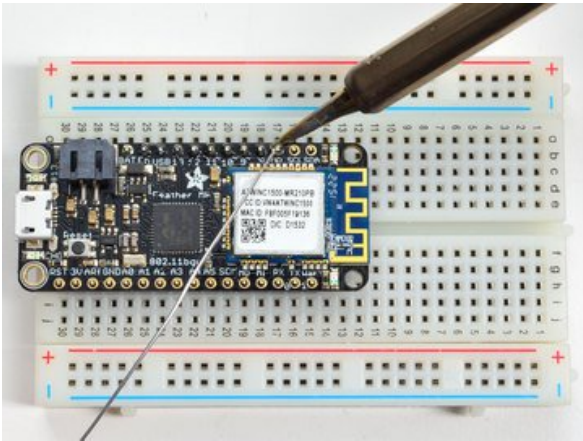
Place the breakout board over the pins so that the short pins poke through the breakout pads



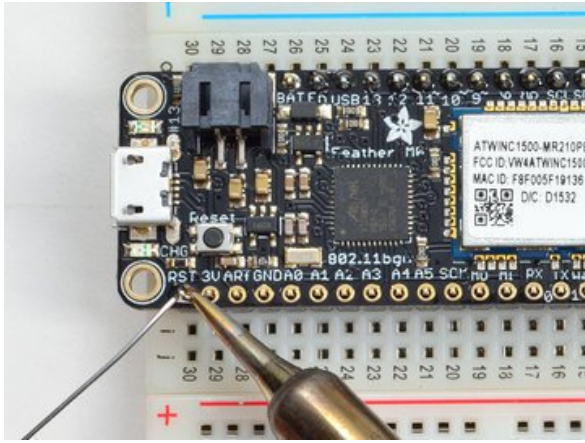
## And Solder!

Be sure to solder all pins for reliable electrical contact.

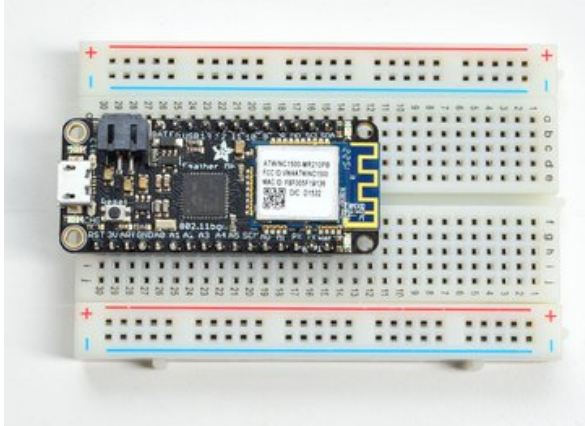
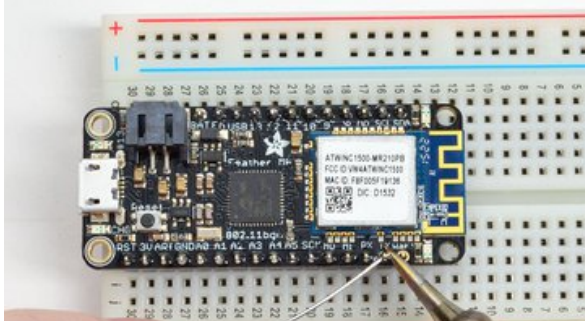
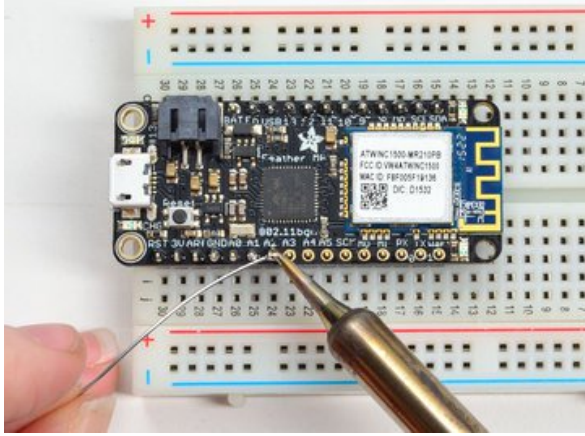
(For tips on soldering, be sure to check out our [Guide to Excellent Soldering](http://adafruit.it/aTk) (<http://adafruit.it/aTk>)).







Solder the other strip as well.



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

## Soldering on Female Header

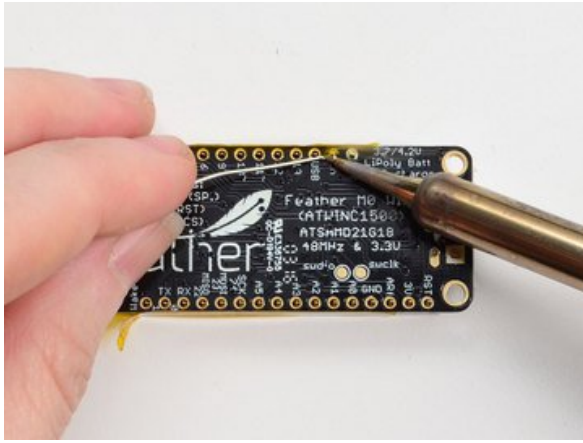


### Tape In Place

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

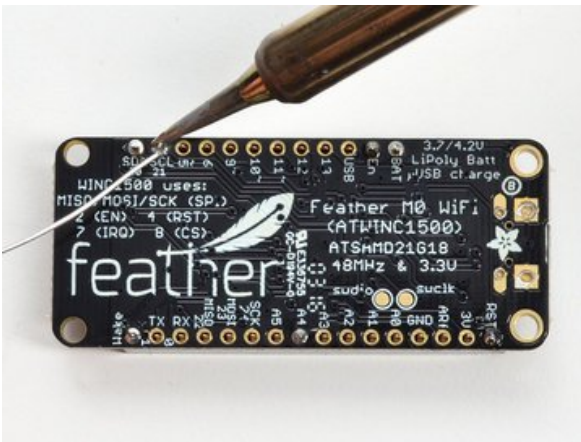
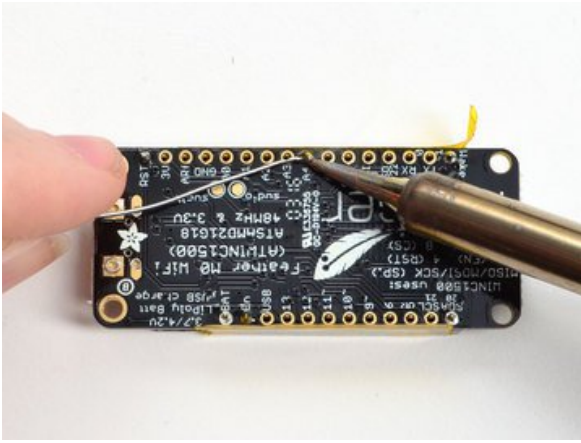
After flipping over, solder one or two points on each strip, to 'tack' the header in place



## 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://adafruit.it/aTk) (<http://adafruit.it/aTk>)).



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

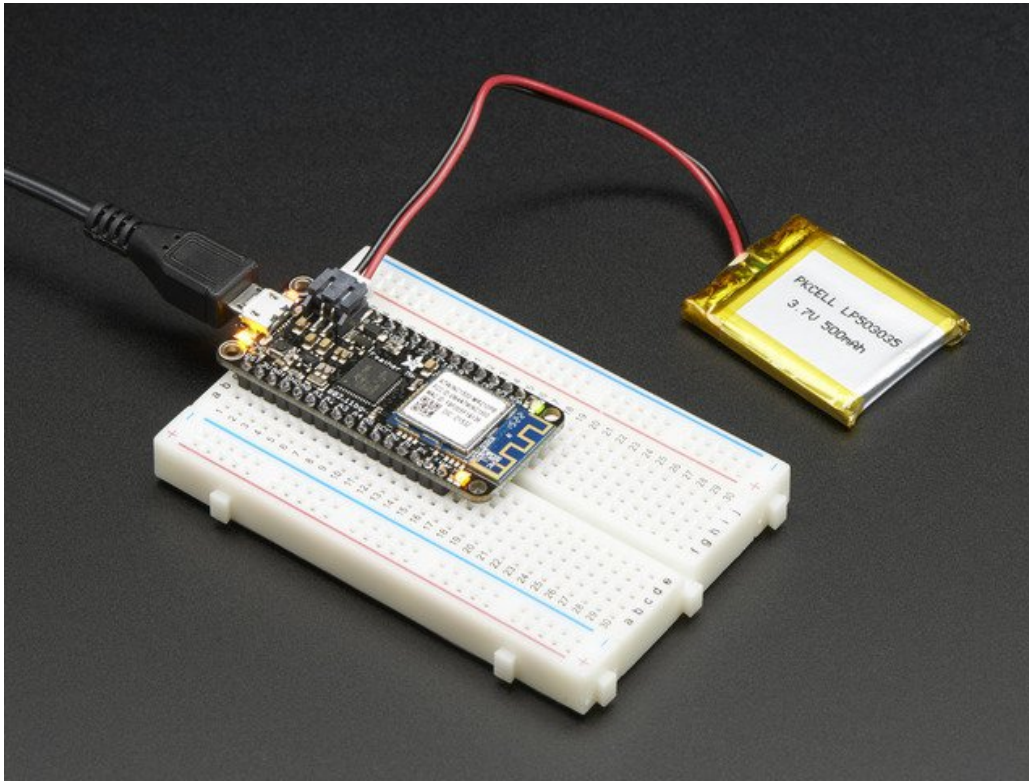


•



•

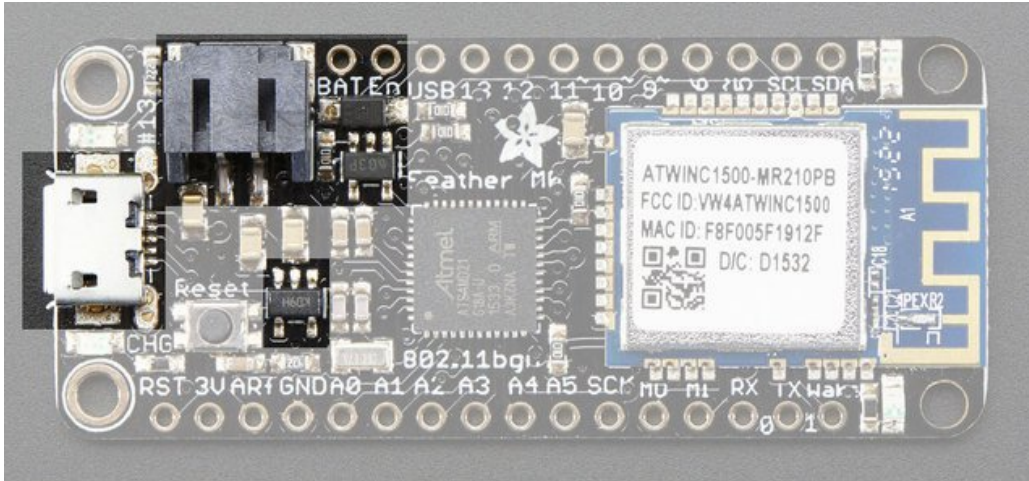
# Power Management



## Battery + USB Power

We wanted to make the Feather 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 (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) at 200mA.** 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.

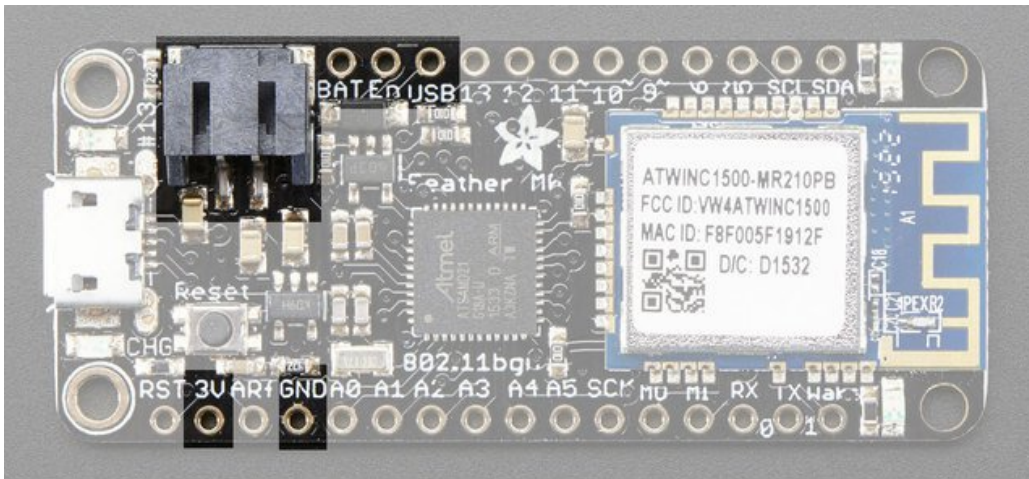
The JST connector polarity is matched to Adafruit LiPoly batteries. Using wrong polarity batteries can destroy your Feather



The above shows the Micro USB jack (left), Lipoly JST jack (top left), as well as the 3.3V regulator and changeover diode (just to the right of the JST jack) and the Lipoly charging circuitry (to the right of the Reset button). There's also a **CHG** LED, 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 600mA peak AP2112K-33. While you can get 600mA from it, you can't do it continuously from 5V as it will overheat the regulator. It's fine for, say, powering the attached WiFi chip or XBee radio though, since the current draw is 'spiky' & sporadic.



## Measuring Battery

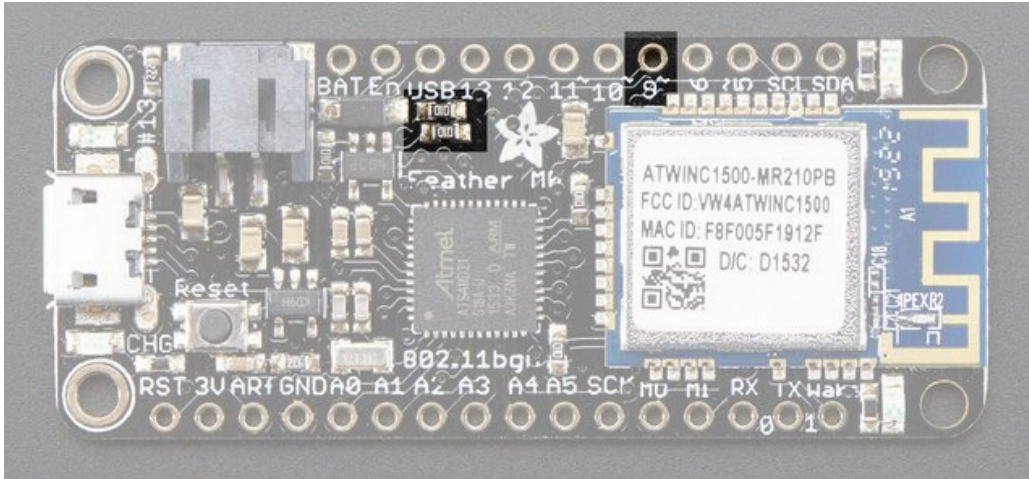
If you're running off of a battery, chances are you wanna know what the voltage is at! That way you can tell when the battery needs recharging. Lipoly batteries are 'maxed out' at 4.2V and stick around 3.7V for much of the battery life, then slowly sink down to 3.2V or so before the protection circuitry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

To make this easy we stuck a double-100K resistor divider on the **BAT** pin, and connected it to **D9** (a.k.a analog #7

**A7).** You can read this pin's voltage, then double it, to get the battery voltage.

```
#define VBATPIN A7
```

```
float measuredvbat = analogRead(VBATPIN);  
measuredvbat *= 2; // we divided by 2, so multiply back  
measuredvbat *= 3.3; // Multiply by 3.3V, our reference voltage  
measuredvbat /= 1024; // convert to voltage  
Serial.print("VBat: "); Serial.println(measuredvbat);
```



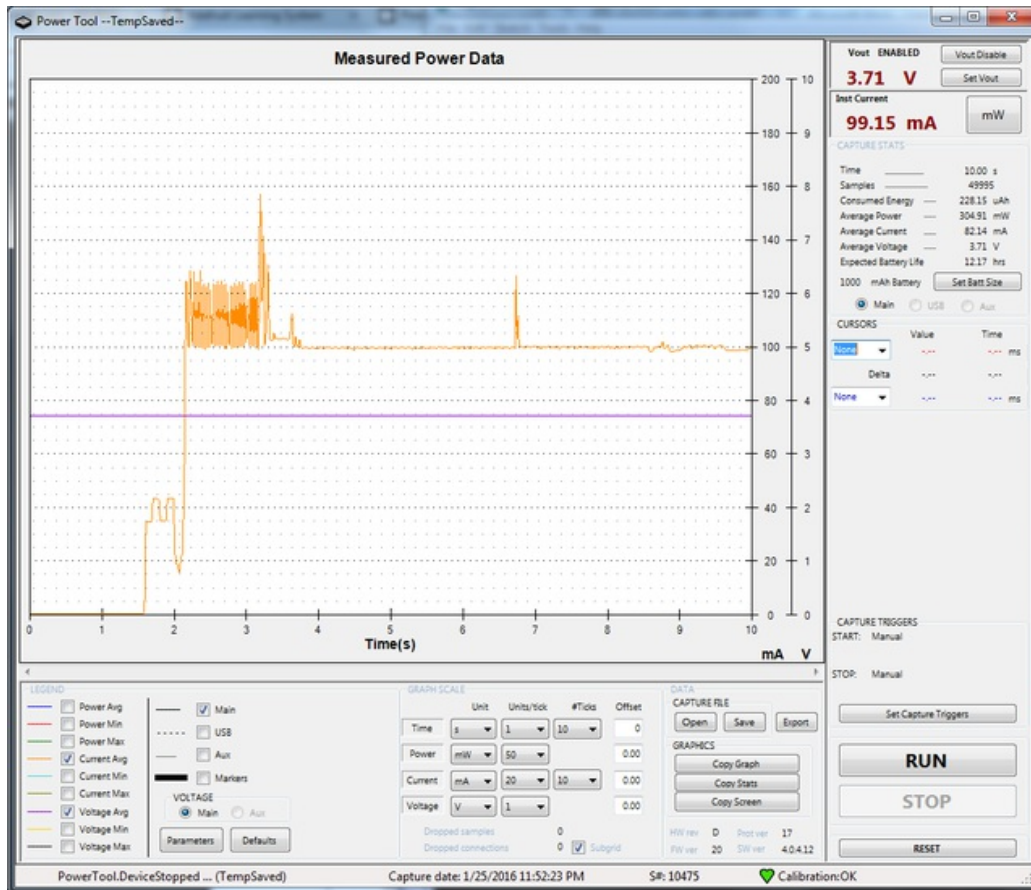
## ENable pin

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

## Power Usage & Saving with WiFi

WiFi is a very power-hungry protocol. During transmit and SSID association, you'll see high power usages. For example, here is an MQTT demo running where it connects to the WPA SSID and then sends a packet every 5 seconds or so:





You can see the chip launch at about 1.5 seconds, then turn on the WiFi and at about 2s make the SSID connection and MQTT connection. The average current is about 100ms afterwards, and a packet spikes up to ~130mA at the 7 second mark.

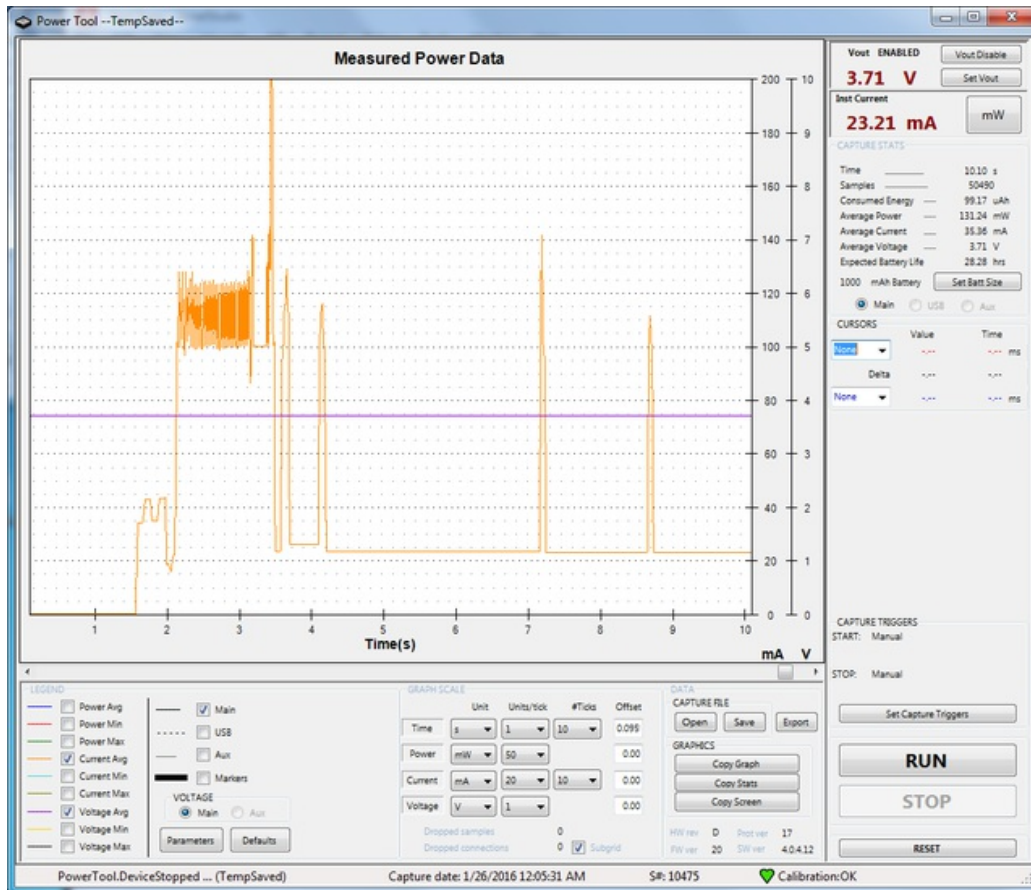
100mA is still quite a bit, you can very easily reduce this by letting the WINC1500 manage its own power:

```
WiFi.setSleepMode(M2M_PS_H_AUTOMATIC, 1); // go into power save mode when possible!
```

When this line is added, it lets the WINC1500 know that when nothings going on, shut down unneeded parts. You dont have to manage the power modes, and the power will drop down nearly instantly to about 22mA average (there's still spikes during transmit of course)

If you're using the Arduino WiFi101 library, call this instead to enable automatic sleep:

```
WiFi.lowPowerMode();
```



Note that 10mA or so is for the ATSAMd chip, so that means you've got ~12mA for the WiFi module.

If you want ultra-low power you can manage the WINC1500 module your own with

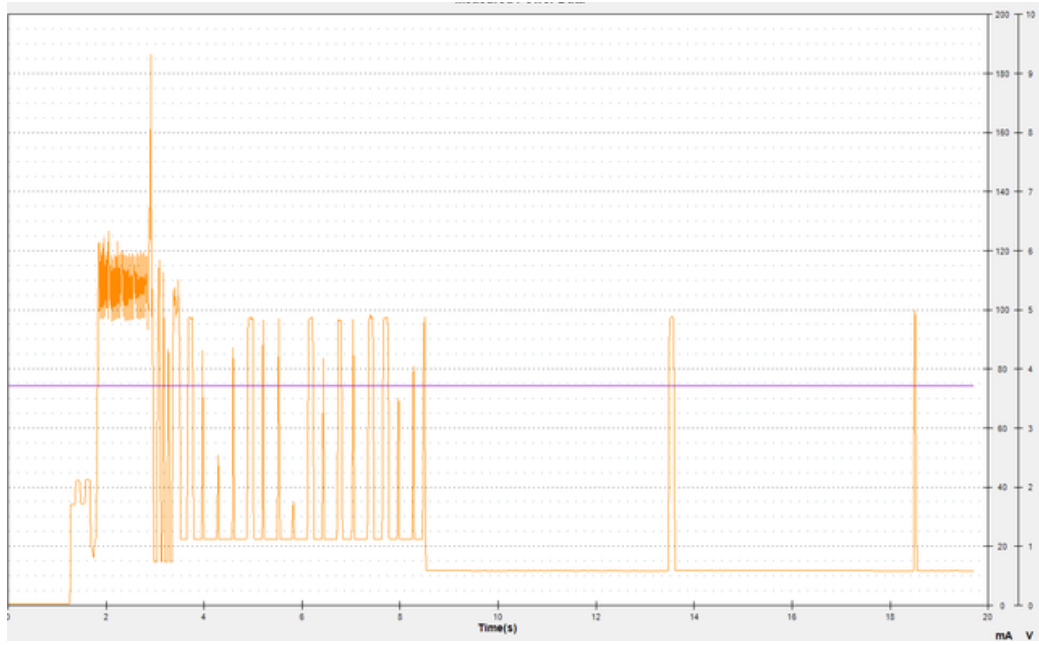
```
WiFi.setSleepMode(M2M_PS_MANUAL, 1);
```

And then when you want it to go to sleep call:

```
WiFi.requestSleep(sleeptimeinmilliseconds)
```

With this mode, you can get much much lower power when you call the requestSleepmode (basically 1-2mA) and still have an active live WiFi connection...but, when not actively sleeping the power usage seems higher (see that spikey part between seconds 3 and 8.5)

A mix of the two may give you the best performance. And don't forget that the SAMD21 is going to draw 10mA so put the main chip to sleep too if you want to get to very low power sleep modes!

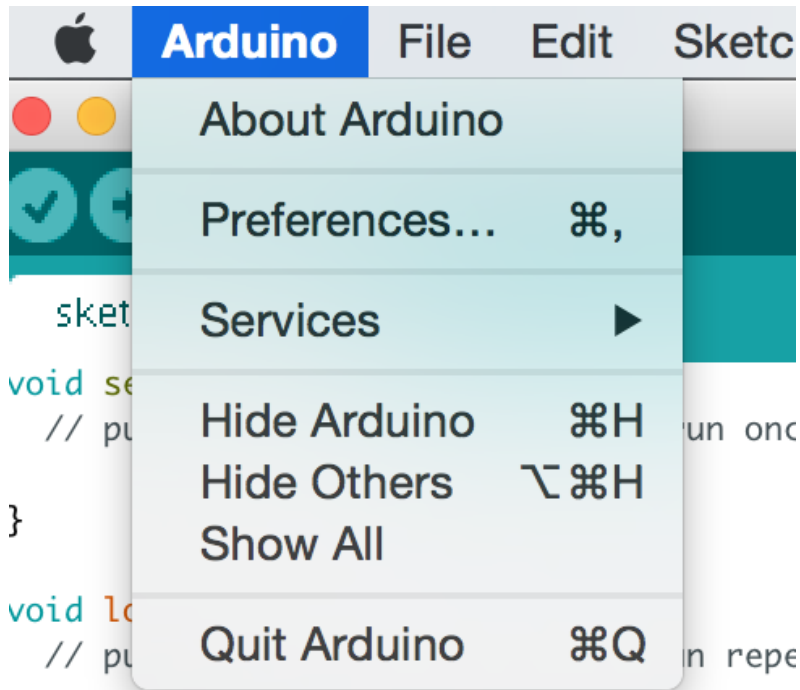


# Arduino IDE Setup

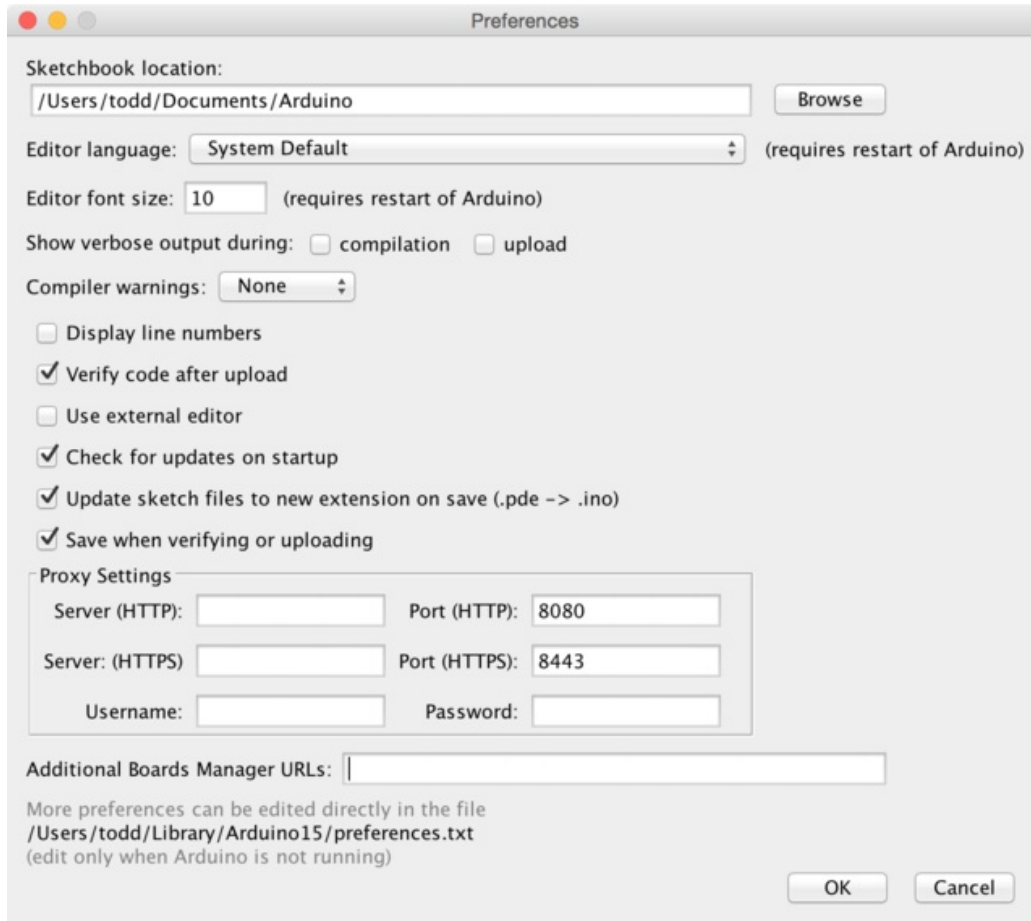
The first thing you will need to do is to download the latest release of the Arduino IDE. You will need to be using **version 1.6.4** or higher for this guide.

[Arduino IDE v1.6.4+ Download](http://adafru.it/f1P)  
<http://adafru.it/f1P>

After you have downloaded and installed **v1.6.4**, you will need to start the IDE and navigate to the **Preferences** menu. You can access it from the **File** menu in *Windows* or *Linux*, or the **Arduino** menu on *OS X*.



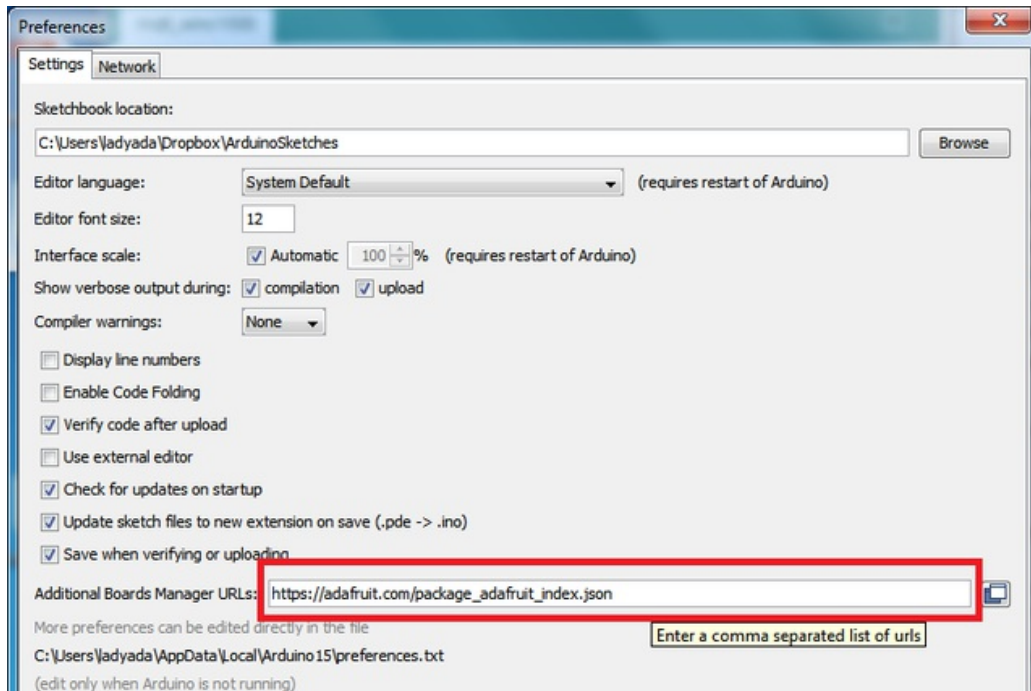
A dialog will pop up just like the one shown below.



We will be adding a URL to the new **Additional Boards Manager URLs** option. The list of URLs is comma separated, and *you will only have to add each URL once*. New Adafruit boards and updates to existing boards will automatically be picked up by the Board Manager each time it is opened. The URLs point to index files that the Board Manager uses to build the list of available & installed boards.

To find the most up to date list of URLs you can add, you can visit the list of [third party board URLs on the Arduino IDE wiki](http://adafru.it/f7U) (<http://adafru.it/f7U>). We will only need to add one URL to the IDE in this example, but **you can add multiple URLs by separating them with commas**. Copy and paste the link below into the **Additional Boards Manager URLs** option in the Arduino IDE preferences.

**[https://adafruit.github.io/arduino-board-index/package\\_adafruit\\_index.json](https://adafruit.github.io/arduino-board-index/package_adafruit_index.json)**



Here's a short description of each of the Adafruit supplied packages that will be available in the Board Manager when you add the URL:

- **Adafruit AVR Boards** - Includes support for Flora, Gemma, Feather 32u4, Trinket, & Trinket Pro.
- **Adafruit SAMD Boards** - Includes support for Feather M0
- **Arduino Leonardo & Micro MIDI-USB** - This adds MIDI over USB support for the Flora, Feather 32u4, Micro and Leonardo using the [arcore project \(http://adafru.it/eSI\)](http://adafru.it/eSI).

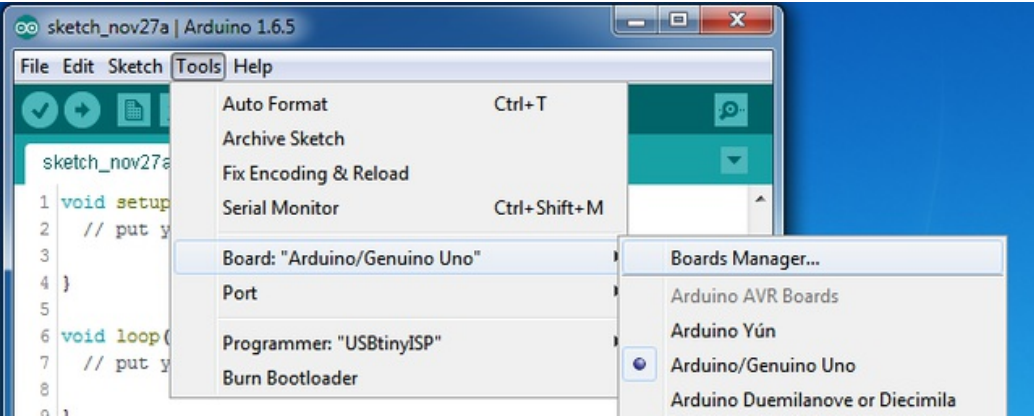
If you have multiple boards you want to support, say ESP8266 and Adafruit, have both URLs in the text box separated by a comma (,)

Once done click **OK** to save the new preference settings. Next we will look at installing boards with the Board Manager.

# Using with Arduino IDE

Since the Feather M0 uses an ATSAM21 chip running at 48 MHz, you can pretty easily get it working with the Arduino IDE. Most libraries (including the popular ones like NeoPixels and display) will work with the M0, especially devices & sensors that use i2c or SPI.

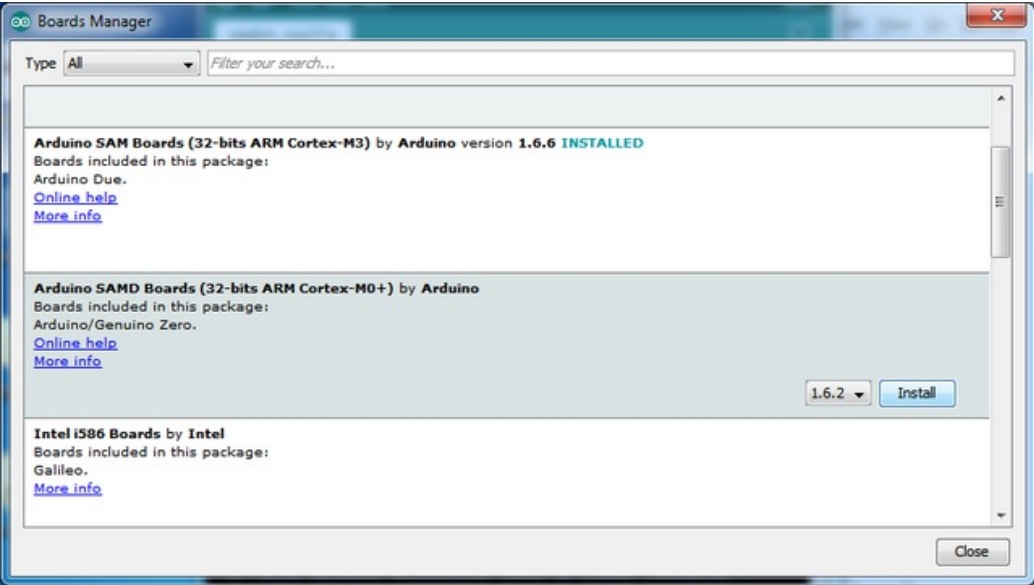
Now that you have added the appropriate URLs to the Arduino IDE preferences, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.

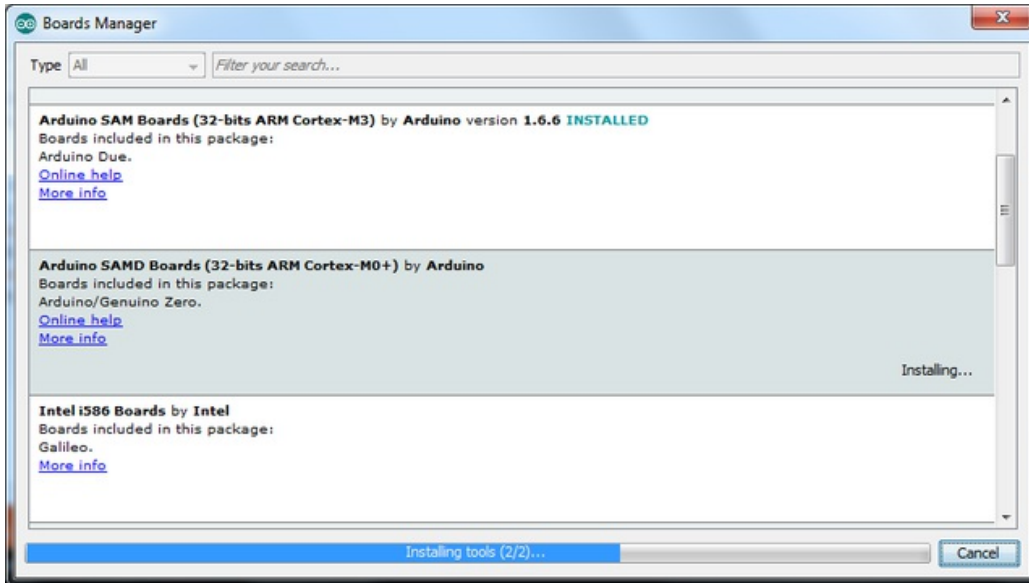


Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select **Contributed**. You will then be able to select and install the boards supplied by the URLs added to the preferences.

# Install SAMD Support

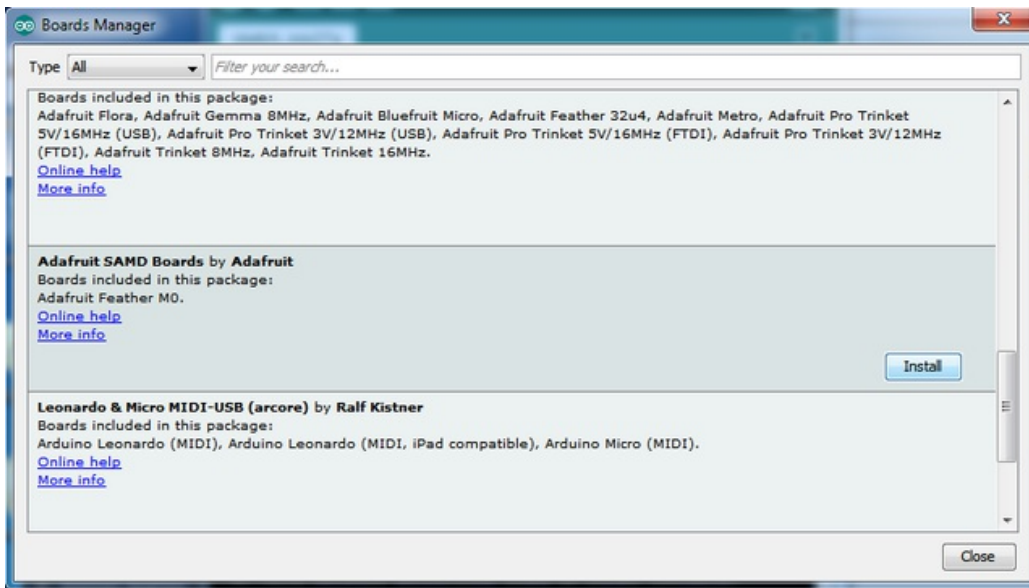
First up, install the **Arduino SAMD Boards** version **1.6.2** or later





## Install Adafruit SAMD

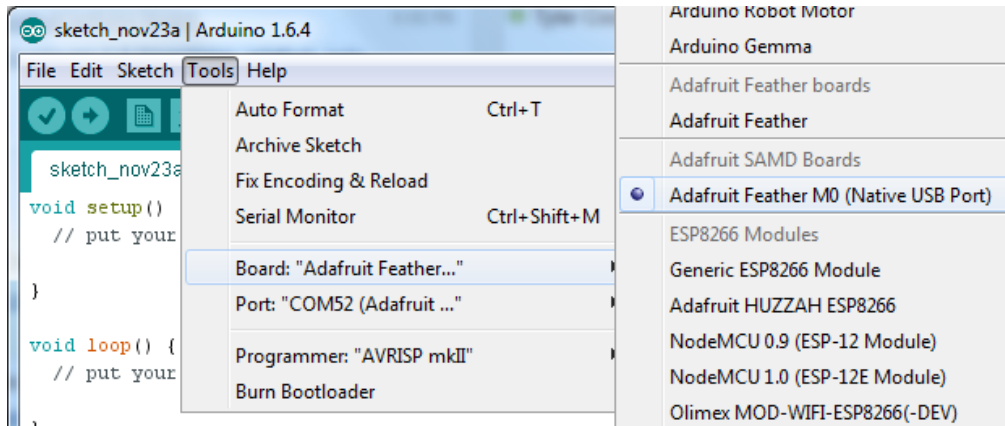
Next you can install the Adafruit SAMD package to add the board file definitions



Even though in theory you don't need to - I recommend rebooting the IDE

**Quit and reopen the Arduino IDE** to ensure that all of the boards are properly installed. You should now be able to select and upload to the new boards listed in the **Tools->Board** menu.





## Install Drivers (Windows Only)

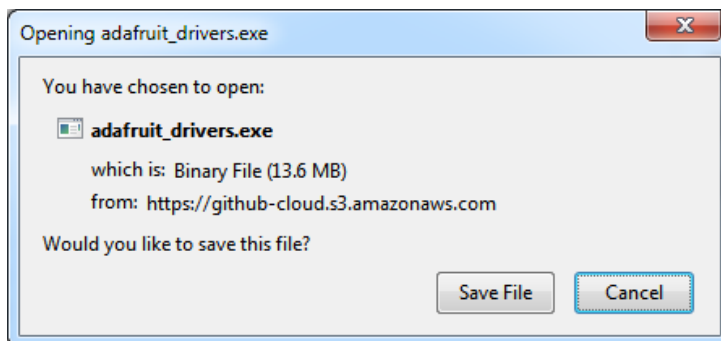
When you plug in the Feather, you'll need to possibly install a driver

Click below to download our Driver Installer

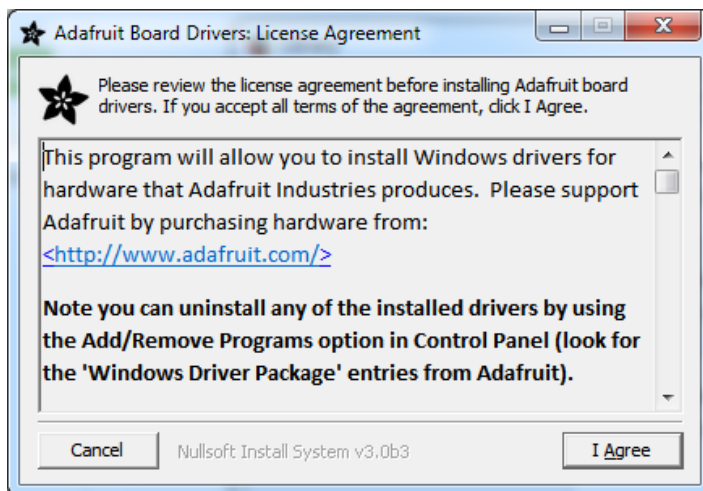
[Download Adafruit Driver Installer](http://adafru.it/mai)

<http://adafru.it/mai>

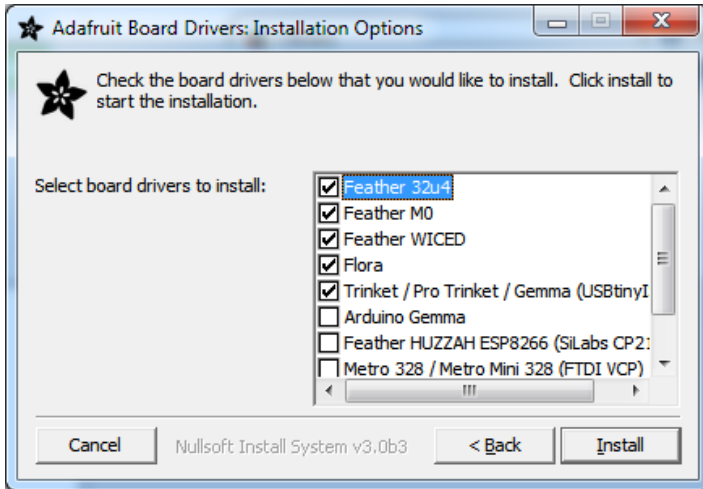
Download and run the installer



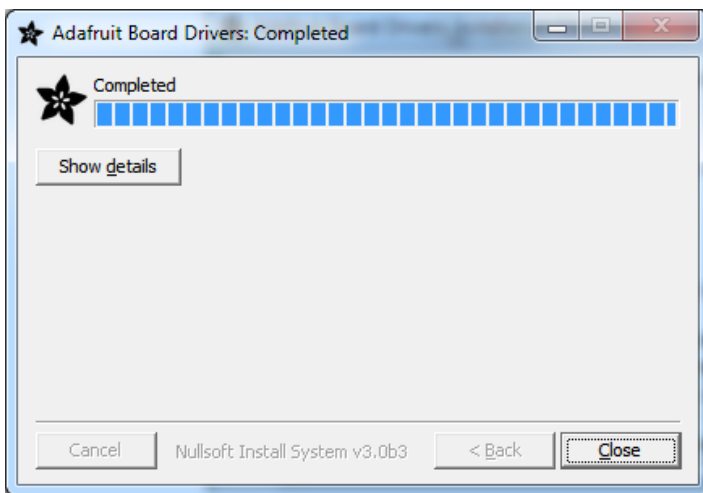
Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



Select which drivers you want to install:



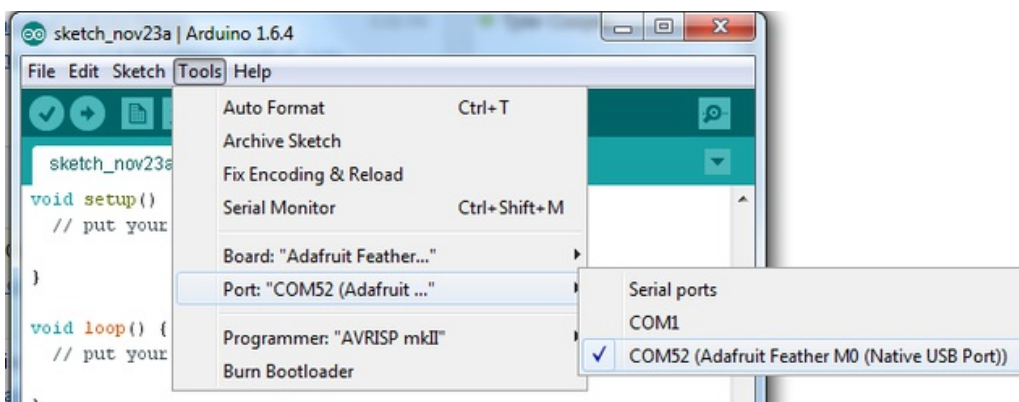
Click **Install** to do the installin'



## Blink

Now you can upload your first blink sketch!

Plug in the Feather M0 and wait for it to be recognized by the OS (just takes a few seconds). It will create a serial/COM port, you can now select it from the dropdown, it'll even be 'indicated' as Feather M0!



Now load up the Blink example

```

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

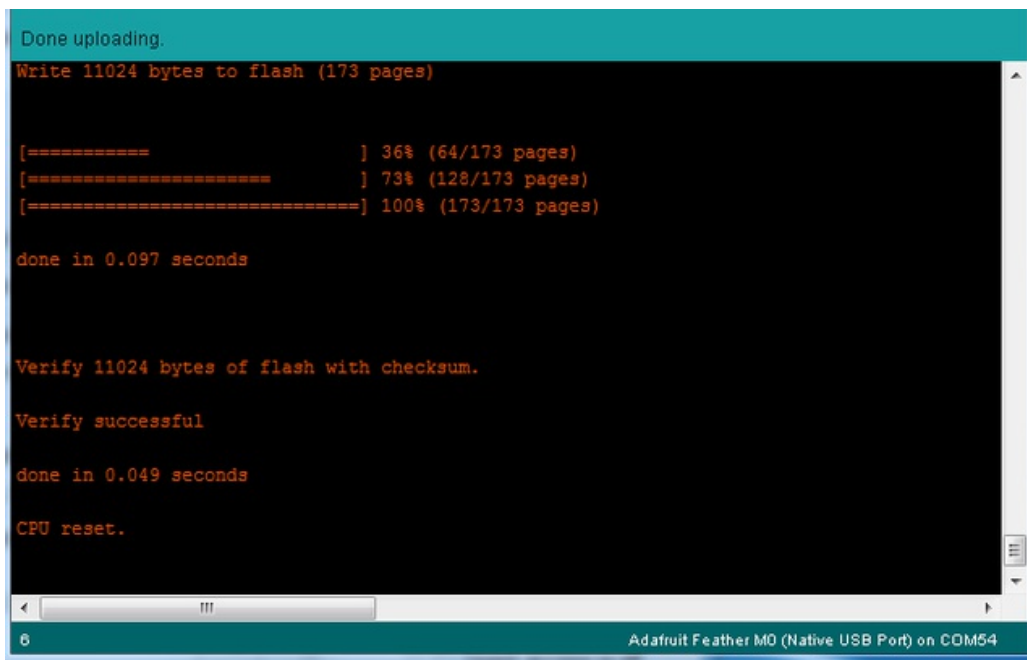
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}

```

And click upload! That's it, you will be able to see the LED blink rate change as you adapt the `delay()` calls.

## Successful Upload

If you have a successful upload, you'll get a bunch of red text that tells you that the device was found and it was programmed, verified & reset

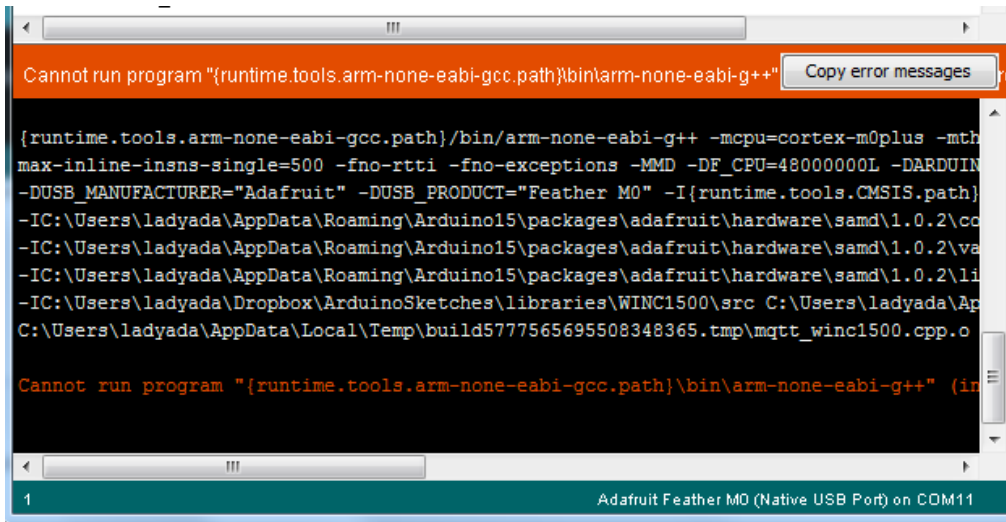


## Compilation Issues

If you get an alert that looks like

**Cannot run program "{runtime.tools.arm-none-eabi-gcc.path}\bin\arm-non-eabi-g++"**

Make sure you have installed the **Arduino SAMD** boards package, you need *both* Arduino & Adafruit SAMD board packages

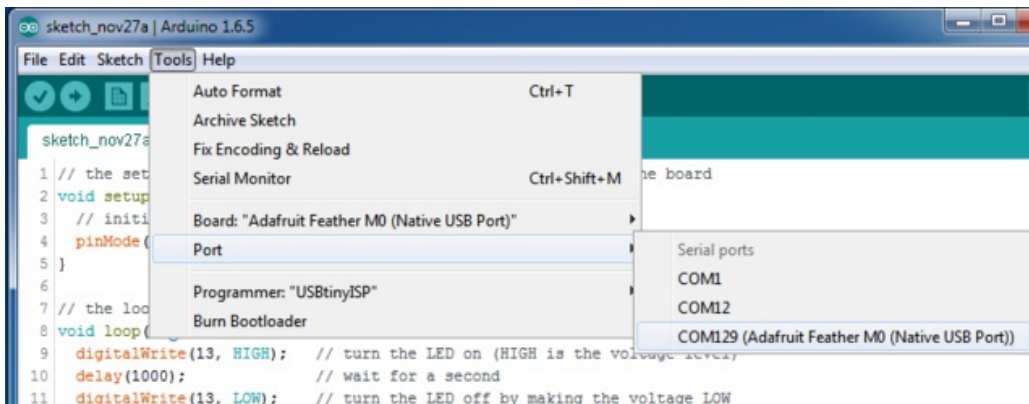


## Manually bootloading

If you ever get in a 'weird' spot with the bootloader, or you have uploaded code that crashes and doesn't auto-reboot into the bootloader, click the **RST** button **twice** (like a double-click) to get back into the bootloader.

**The red LED will pulse, so you know that its in bootloader mode.**

Once it is in bootloader mode, you can select the newly created COM/Serial port and re-try uploading.



You may need to go back and reselect the 'normal' USB serial port next time you want to use the normal upload.

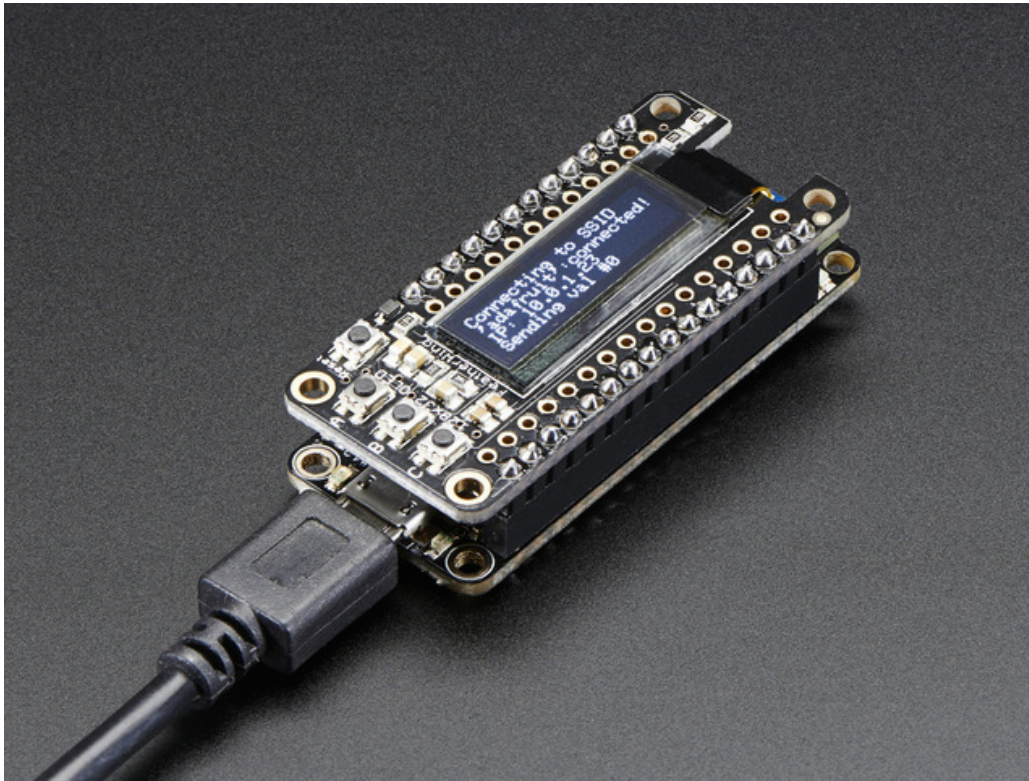
## Ubuntu & Linux Issue Fix

Note if you're using Ubuntu 15.04 (or perhaps other more recent Linux distributions) there is an issue with the modem manager service which causes the Bluefruit LE micro to be difficult to program. If you run into errors like "device or resource busy", "bad file descriptor", or "port is busy" when attempting to program then [you are hitting this issue](http://adafru.it/sHE). (<http://adafru.it/sHE>)

The fix for this issue is to make sure Adafruit's custom udev rules are applied to your system. One of these rules is made to configure modem manager not to touch the Feather board and will fix the programming difficulty issue.

[Follow the steps for installing Adafruit's udev rules on this page](http://adafru.it/iOE). (<http://adafru.it/iOE>)

## Using the WiFi Module

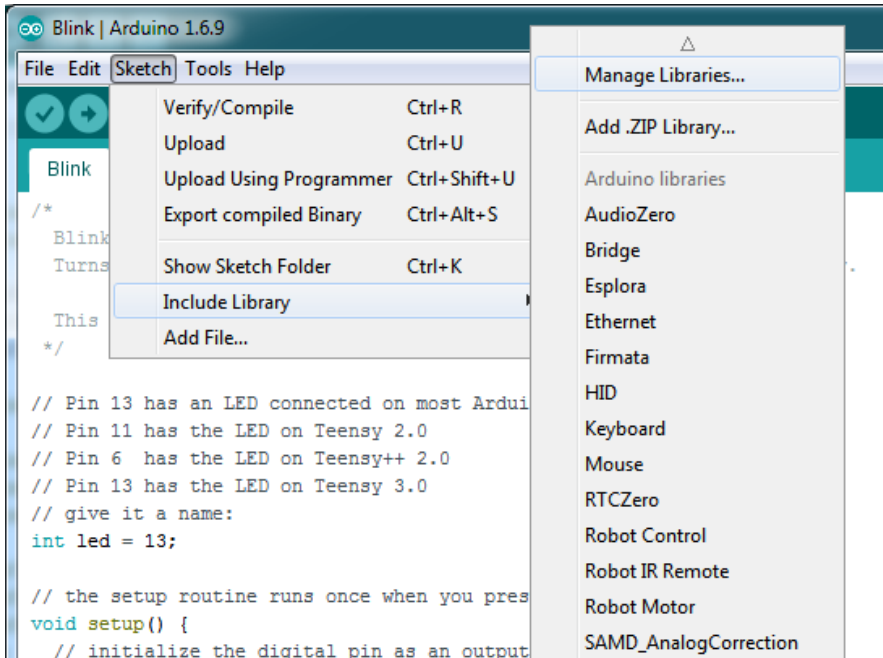


Once you have your Feather working, you probably want to rock out with some Wireless connectivity. Luckily, Atmel & Arduino have written a great library for supporting the WINC1500

## Install the Library

We will start by installing the [official Arduino WiFi101 library](http://adafru.it/kUF) (<http://adafru.it/kUF>).

We want the latest version so visit the Library Manager



Type in wifi101 and when the library comes up, click **Install** or **Update** to make sure its the most recent one!

If you're not familiar with installing Arduino libraries, please visit our tutorial [All About Arduino Libraries](http://adafruit.com/learn/arduino/understanding-arduino-libraries) (<http://adafru.it/aYM>)!

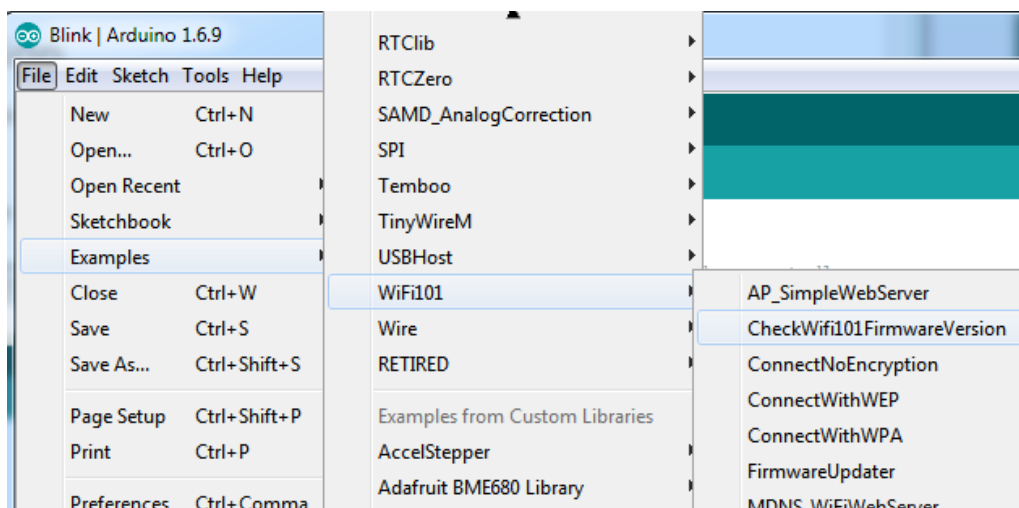
Restart the Arduino IDE.

You may need to use Arduino 1.6.5 or later

## Check Connections & Version

Before we start, its important to verify you have the right setup & firmware version.

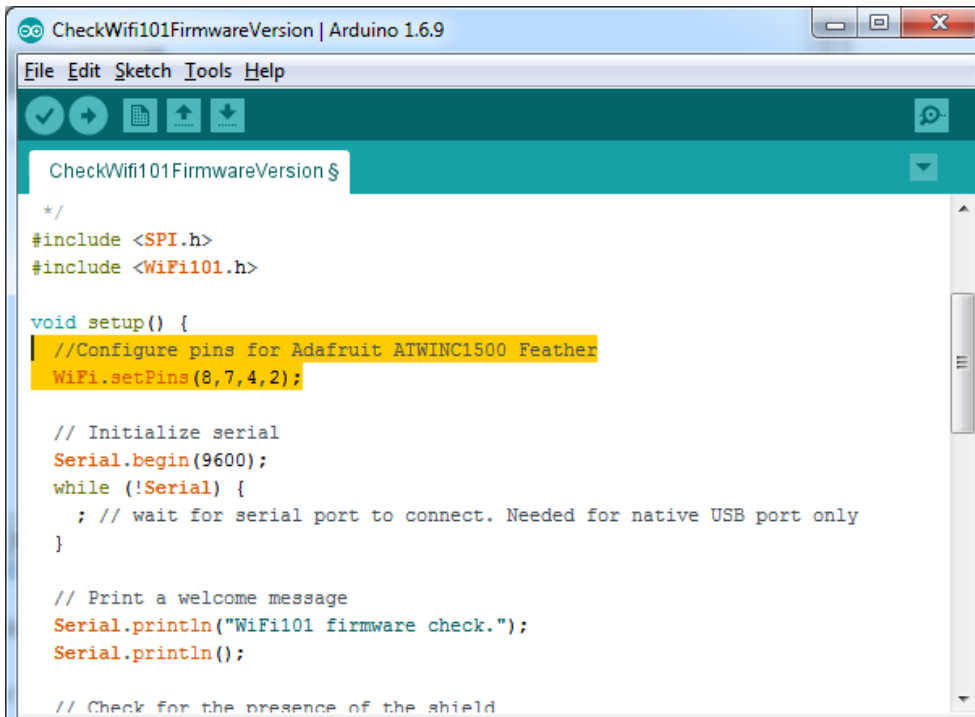
Load up the **WiFi101->CheckWifi101Firmware** sketch



Note that to use the official Arduino WiFi101 Library, we must configure it to use the pins specific to the ATWINC1500 Feather. With each example sketch, you'll need to add `WiFi.setPins(8,7,4,2);` to the top of the setup function!

```
//Configure pins for Adafruit ATWINC1500 Feather
WiFi.setPins(8,7,4,2);
```

Like so:



```
CheckWifi101FirmwareVersion | Arduino 1.6.9
File Edit Sketch Tools Help
CheckWifi101FirmwareVersion $
*/
#include <SPI.h>
#include <WiFi101.h>

void setup() {
  //Configure pins for Adafruit ATWINC1500 Feather
  WiFi.setPins(8,7,4,2);

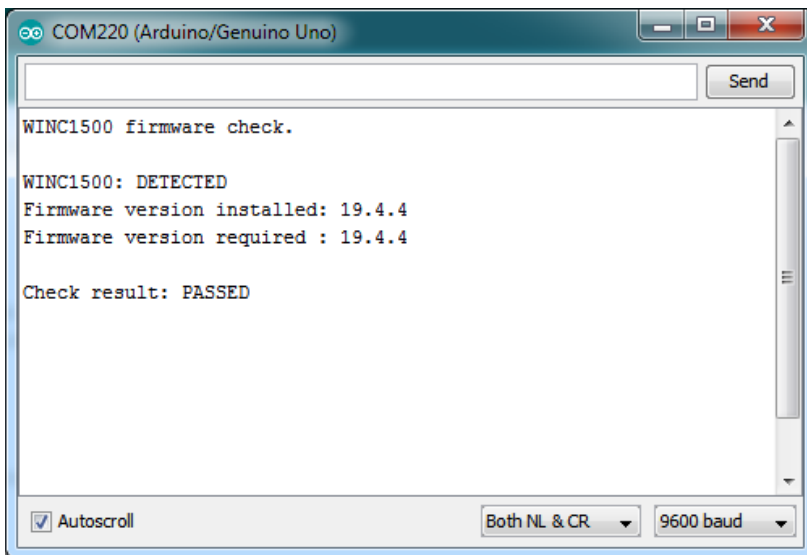
  // Initialize serial
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  // Print a welcome message
  Serial.println("WiFi101 firmware check.");
  Serial.println();

  // Check for the presence of the shield
```

Upload to your Arduino and open up the Serial Console at 9600 baud:

You should see the firmware version. If your firmware hasn't **PASSED**, [use the firmware updater to get the latest, its easy!](http://adafru.it/vfe) (<http://adafru.it/vfe>)



```
COM220 (Arduino/Genuino Uno)
WINC1500 firmware check.

WINC1500: DETECTED
Firmware version installed: 19.4.4
Firmware version required : 19.4.4

Check result: PASSED

Autoscroll Both NL & CR 9600 baud
```

If you have version 19.3 or less, the firmware is too old

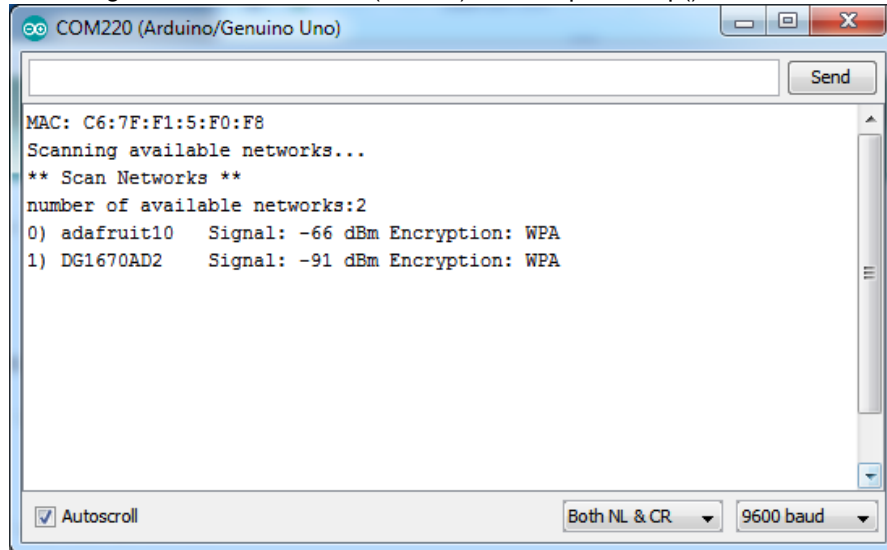
If you get not response, the firmware is either waaay to old, or something is amiss with your wiring!

## Scanning WiFi

Now that you have the right firmware version, lets scan for network!

Run the **WiFi101->ScanNetworks** example to see a list of available visible networks

Don't forget to add `WiFi.setPins(8,7,4,2)` at the top of `setup()`



## Connect & Read Webpage

OK finally you get to connect and read some data!

Open up the **WiFi101->WiFiWebClient** example

Edit the **ssid** and **pass** variables to contain your network and password

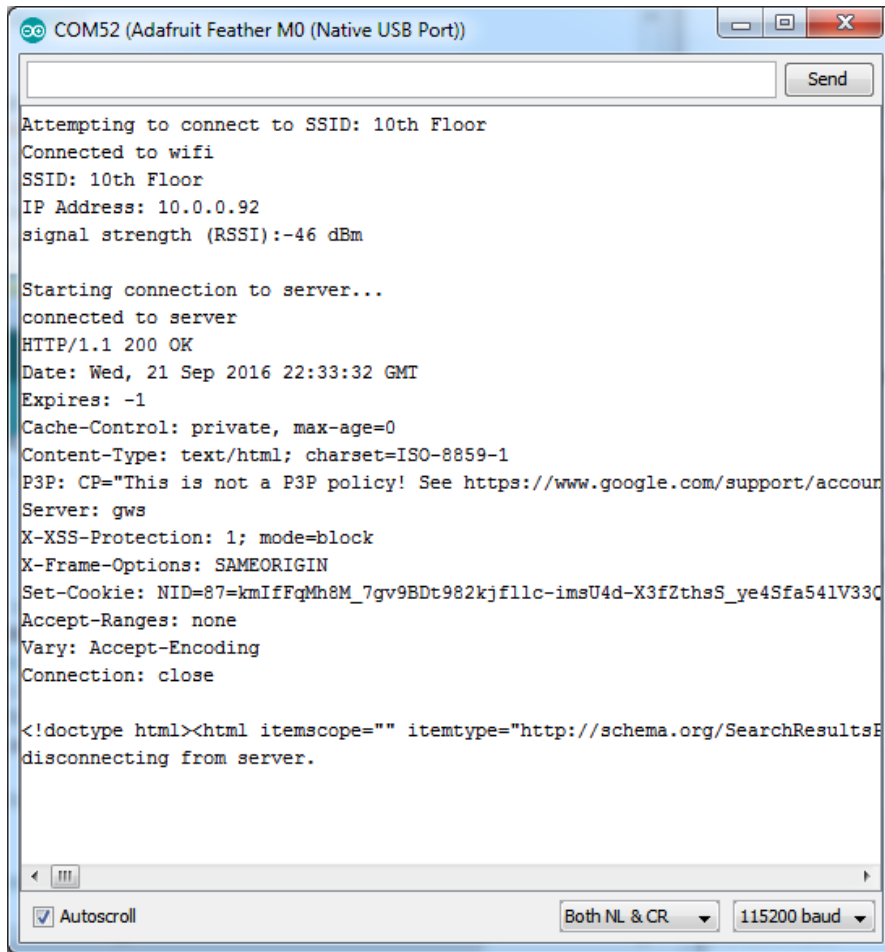
```
34
35
36 char ssid[] = "adafruit"; // your network SSID (name)
37 char pass[] = "supersekret"; // your network password (use for WPA, or use as key for WEP)
38 int keyIndex = 0; // your network key Index number (needed only for WEP)
39
40 int status = WL_IDLE_STATUS;
41 // if you don't want to use DNS (and reduce your sketch size)
42 // use the numeric IP instead of the name for the server:
```

Add the following lines at the top of `setup()`

```
//Configure pins for Adafruit ATWINC1500 Feather
WiFi.setPins(8,7,4,2);
```

It will connect to the website in **server** and read the **webpage** manually:





```
COM52 (Adafruit Feather M0 (Native USB Port))
Attempting to connect to SSID: 10th Floor
Connected to wifi
SSID: 10th Floor
IP Address: 10.0.0.92
signal strength (RSSI):-46 dBm

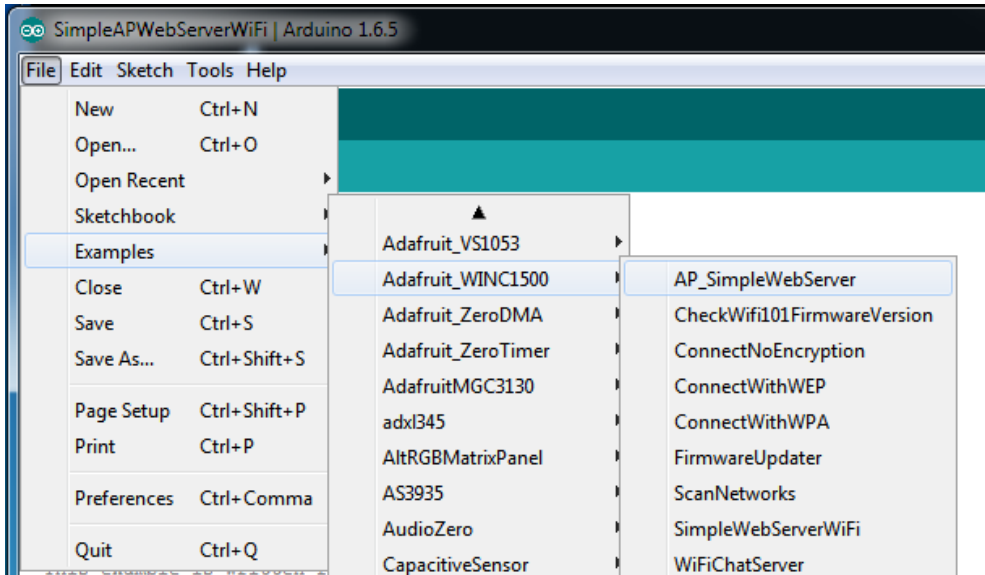
Starting connection to server...
connected to server
HTTP/1.1 200 OK
Date: Wed, 21 Sep 2016 22:33:32 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See https://www.google.com/support/accoun
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=87=kmIfFqMh8M_7gv9BDt982kjf11c-imsU4d-X3fZthsS_ye4Sfa541V33C
Accept-Ranges: none
Vary: Accept-Encoding
Connection: close

<!doctype html><html itemscope="" itemtype="http://schema.org/SearchResultsE
disconnecting from server.
```

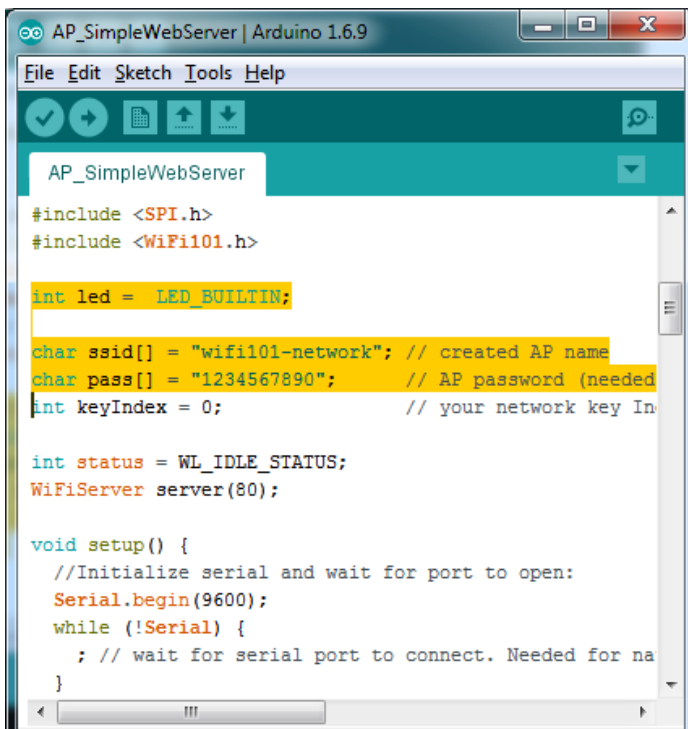
## Creating an Access Point + Webserver

This demo will let you create a new WiFi AP with the Feather M0 which you can connect to from any WiFi capable device. It will also create a Server so you can connect and turn on/off the onboard LED

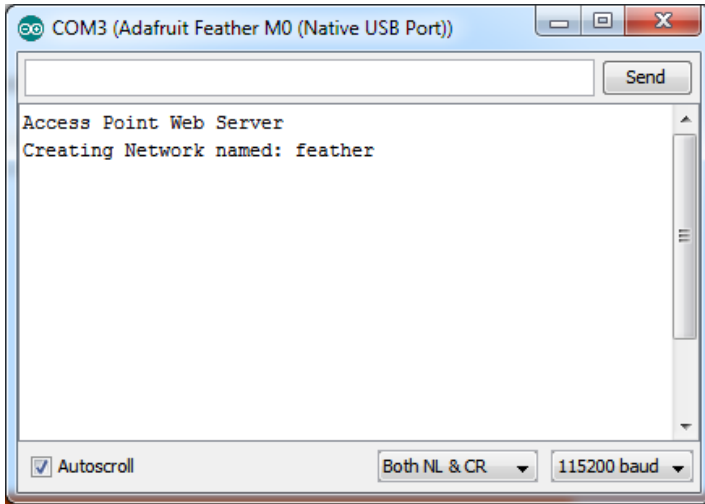
Launch the **WiFi101->AP\_SimpleWebServer** example



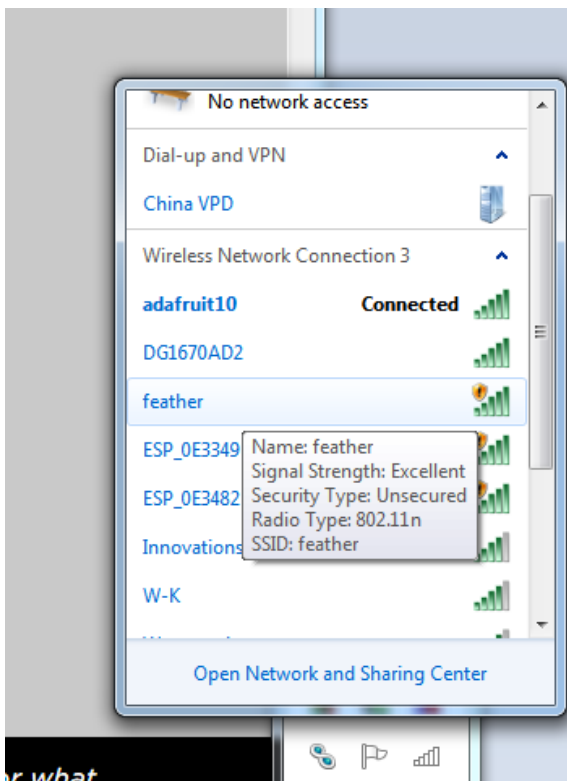
You can change the SSID & LED (LED\_BUILTIN is #13, the onboard feather LED)



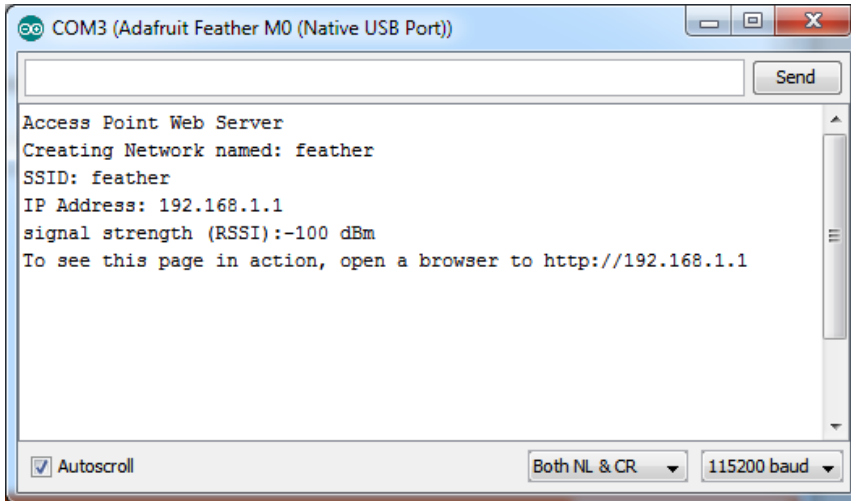
Upload and open up the serial console to start the AP



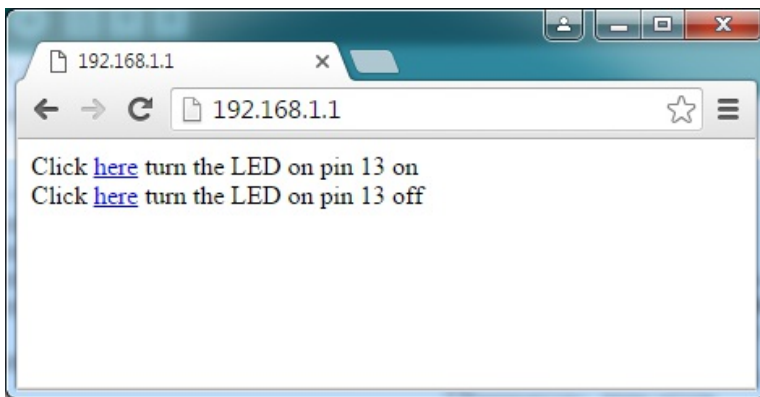
Your computer will see the new AP and you should connect to it



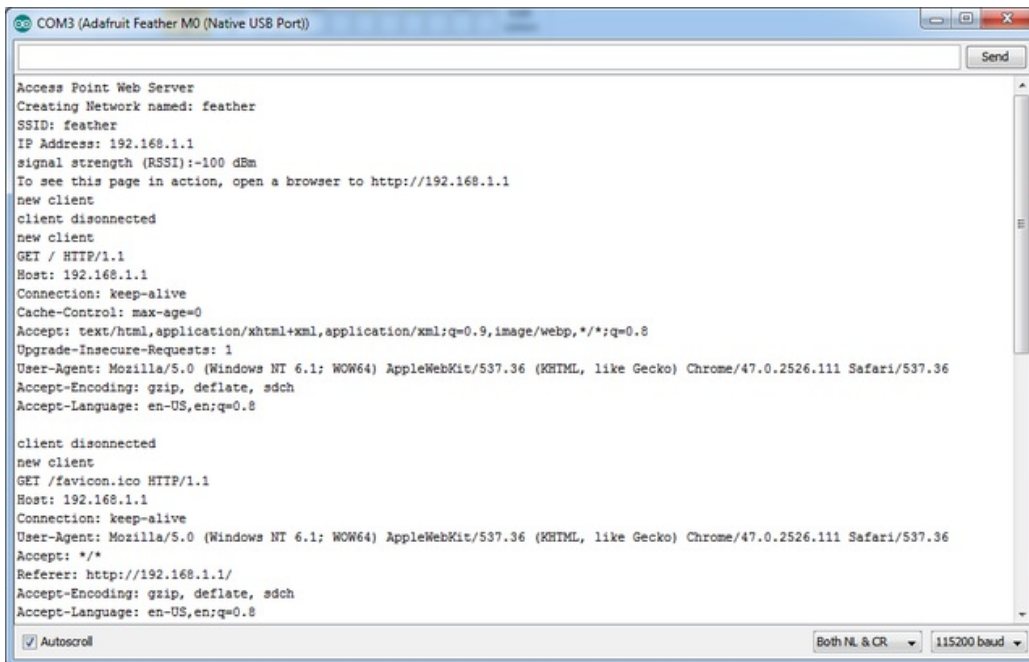
Back over at the serial console, the Feather will have started up a server, it will print out the IP address and instructions



Go to the IP address and you will see the mini webpage, click on the links to turn on/off the LED



In the serial console you will see the data received from the webbrowser client

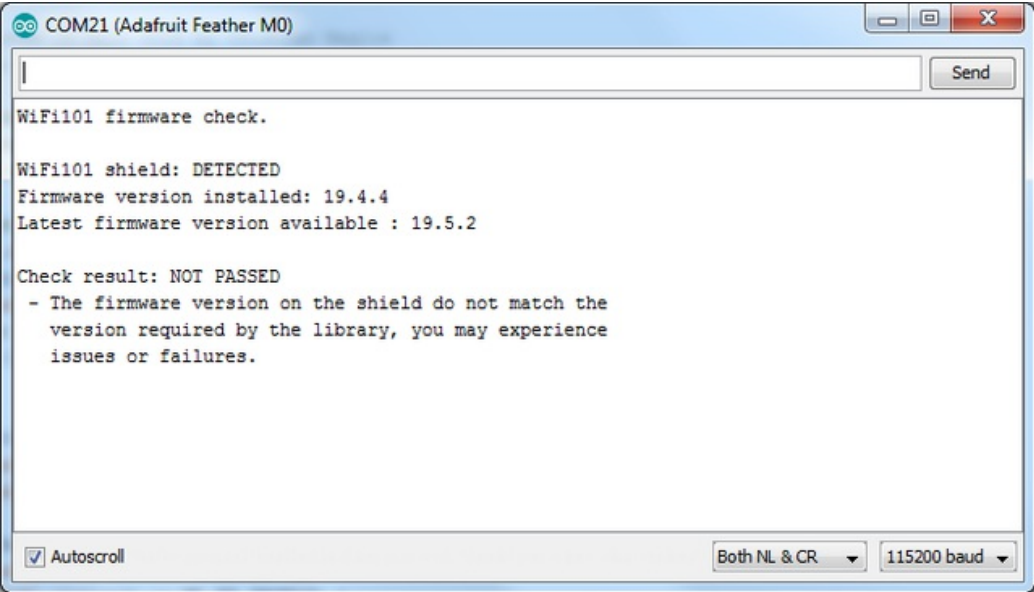


That's it! pretty easy, huh? There's other examples you can try such as server mode, UDP data transmission & SSL

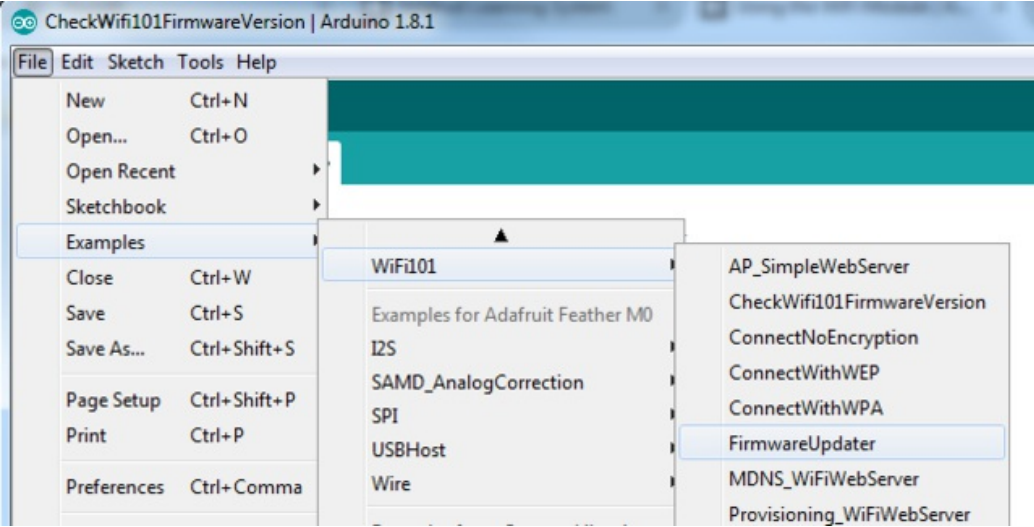


# Updating Firmware

As new versions of the WiFi101 library come out, you may end up getting a complaint that the library and WINC1500 firmware are out of sync:



No problem - you can update the firmware through your Arduino/compatible! Start by loading up the **FirmwareUpdater** sketch



update the pins as necessary, we have the default for use with the Feather M0 or WINC1500 breakout

```
FirmwareUpdater | Arduino 1.8.1
File Edit Sketch Tools Help

FirmwareUpdater $ Endianess

uint32_t address;
uint32_t arg1;
uint16_t payloadLength;

// payloadLength bytes of data follows...
} UartPacket;

static const int MAX_PAYLOAD_SIZE = 1024;

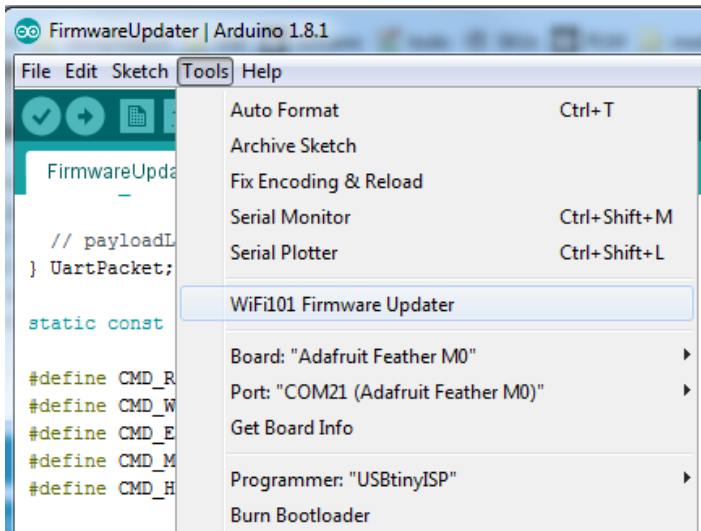
#define CMD_READ_FLASH      0x01
#define CMD_WRITE_FLASH    0x02
#define CMD_ERASE_FLASH    0x03
#define CMD_MAX_PAYLOAD_SIZE 0x50
#define CMD_HELLO          0x99

void setup() {
  WiFi.setPins(8,7,4,2);
  Serial.begin(115200);

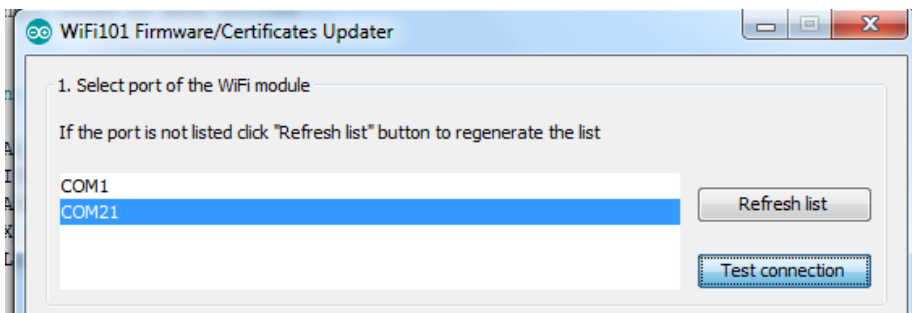
  pinModeInit();
}
```

Upload it to your board. Make sure the Serial console is not open before or after uploading.

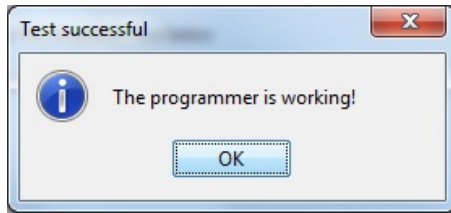
Then select the Updater tool built into the IDE



Select the right COM port, and click Test Connection

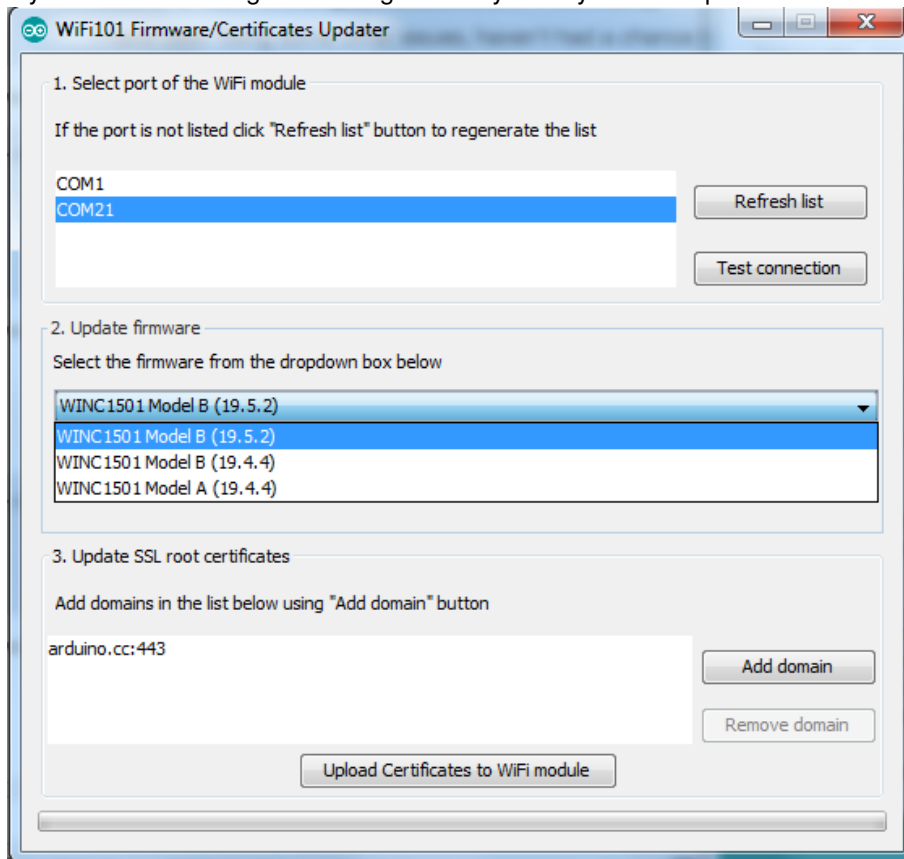


If all is good you'll get a confirmation



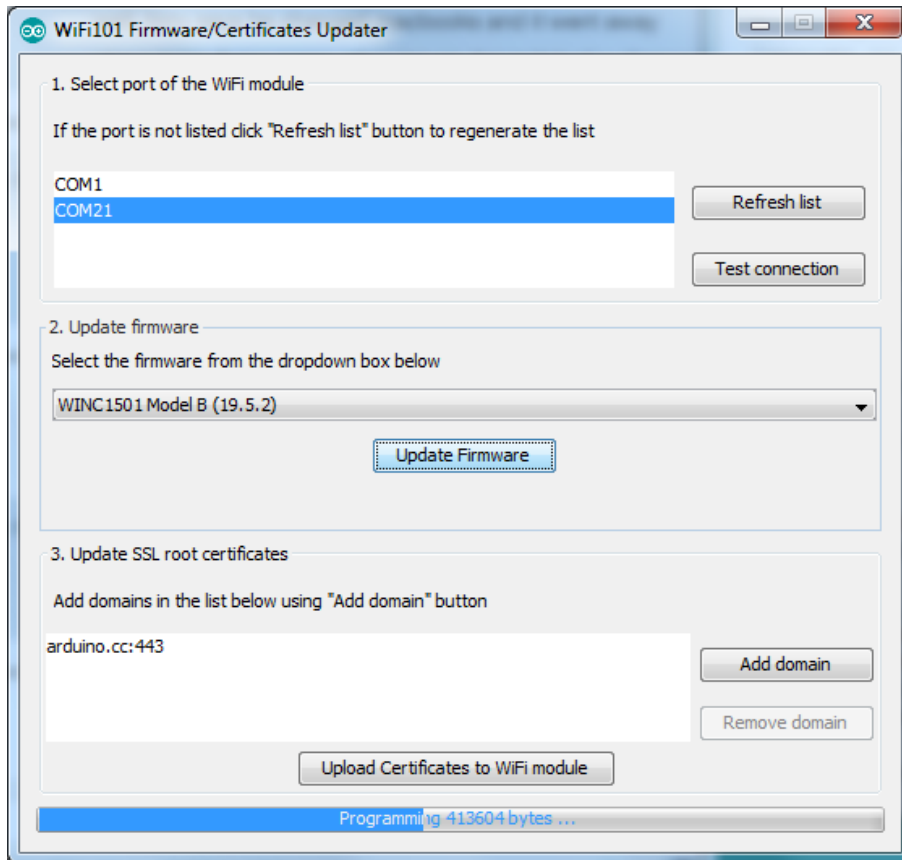
Next, select the firmware - we of course recommend the latest version!

If you don't see the right/matching version you may need to update the IDE

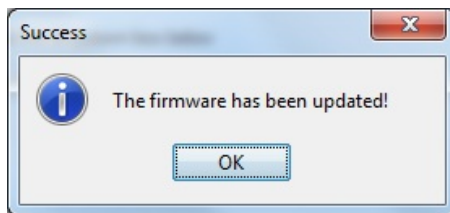


Once you feel ready - make sure the USB cable is connected solidly! Click **Update Firmware**

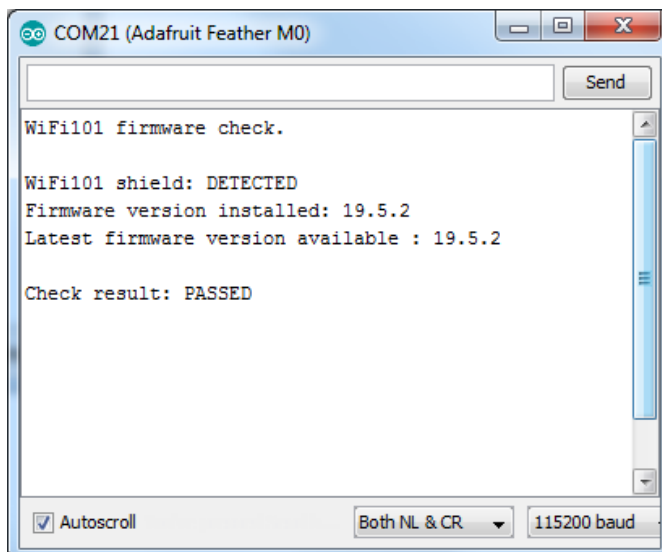




And a minute or two later...



Now you're ready to rock! Reload the Firmware Check sketch from before, this time you will see:



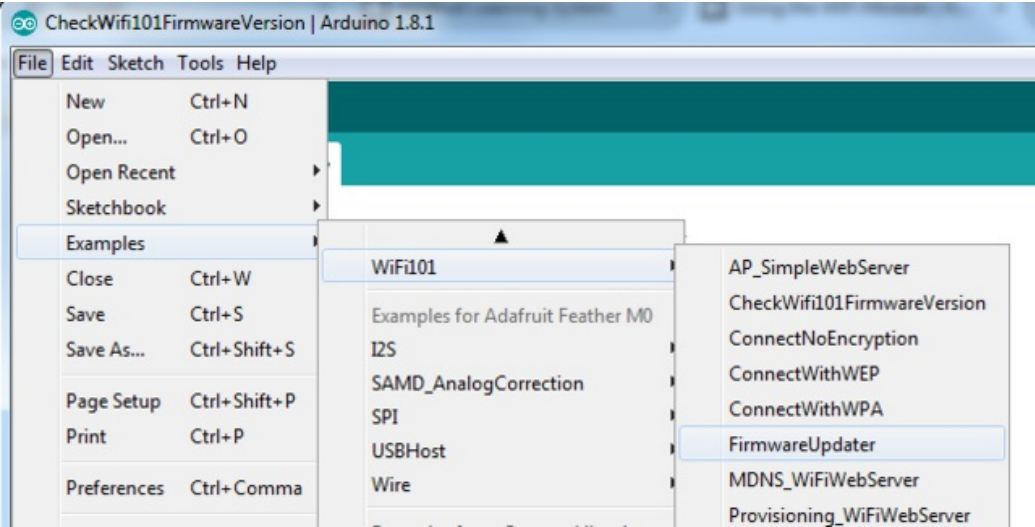


# Updating SSL Certificates

If you're trying to connect to a computer or service via SSL and the connection is failing, you may need to update the certificates built into the WINC1500. By default it comes with many of the most popular SSL certificates but you may bump into a site that requires one that isn't included.

It's quite easy to update the certificates, you'll need to upload some code and run the uploader but it only has to happen once.

Start out by uploading the **FirmwareUpdater** sketch from **WiFi101**



update the pins as necessary, we have the default for use with the Feather M0 or WINC1500 breakout

```
Arduino IDE: FirmwareUpdater | Arduino 1.8.1
File Edit Sketch Tools Help
FirmwareUpdater $ Endianess
uint32_t address;
uint32_t arg1;
uint16_t payloadLength;

// payloadLength bytes of data follows...
} UartPacket;

static const int MAX_PAYLOAD_SIZE = 1024;

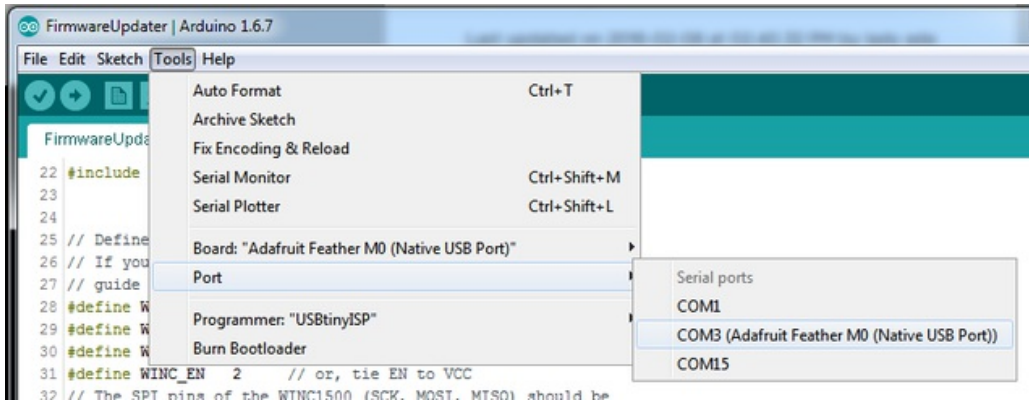
#define CMD_READ_FLASH      0x01
#define CMD_WRITE_FLASH     0x02
#define CMD_ERASE_FLASH     0x03
#define CMD_MAX_PAYLOAD_SIZE 0x50
#define CMD_HELLO           0x99

void setup() {
  WiFi.setPins(8, 7, 4, 2);
  Serial.begin(115200);

  pinMode(LED_BUILTIN, OUTPUT);
}
```

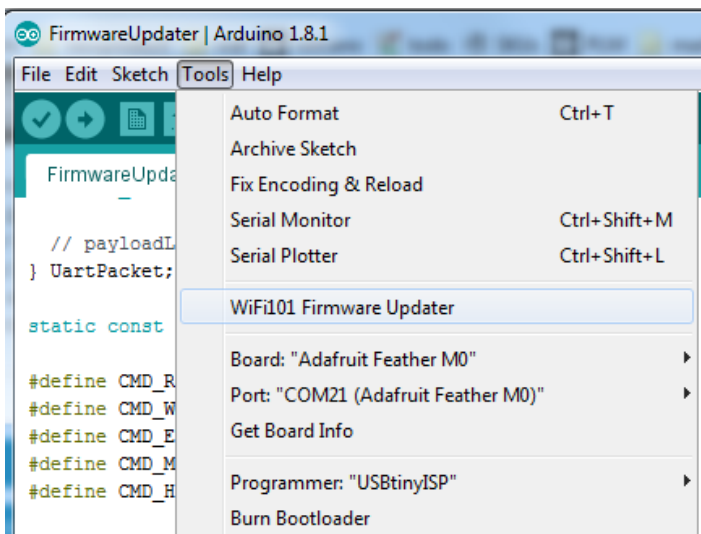
and upload it!

After uploading be sure to note what is the name of the COM or Serial port for the Arduino Zero or Feather...You'll need this for the next step

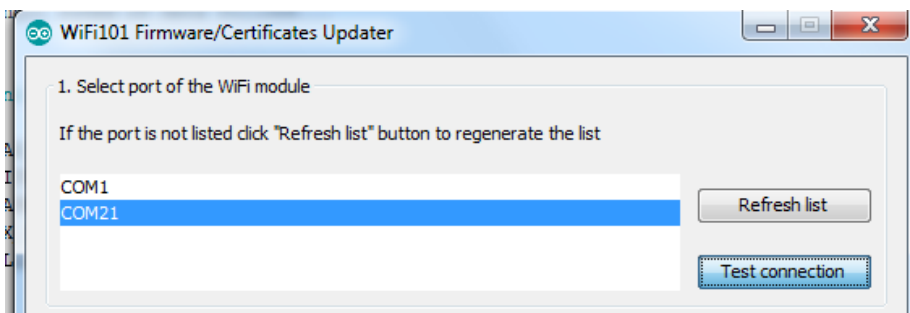


Upload it to your Feather. Make sure the Serial console is not open before or after uploading.

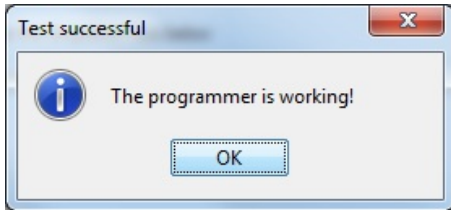
Then select the Updater tool built into the IDE



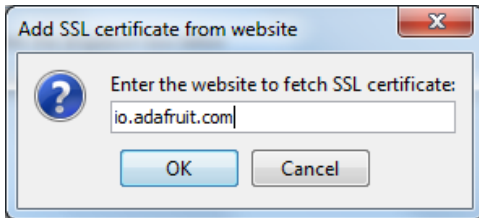
Select the right COM port, and click Test Connection



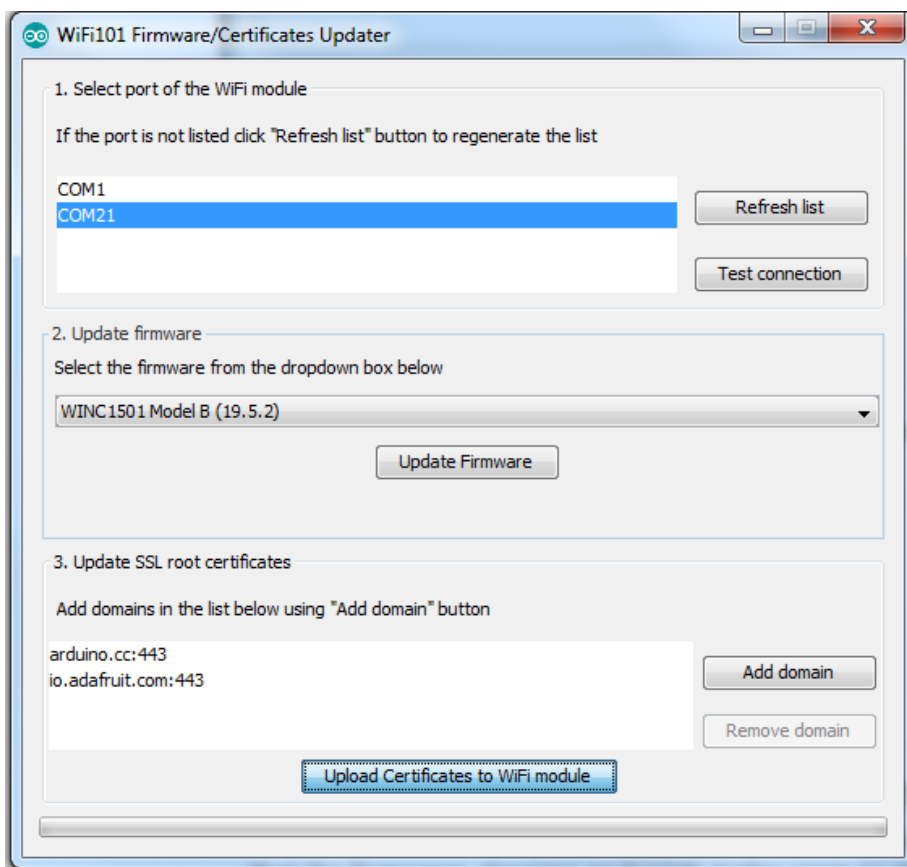
If all is good you'll get a confirmation



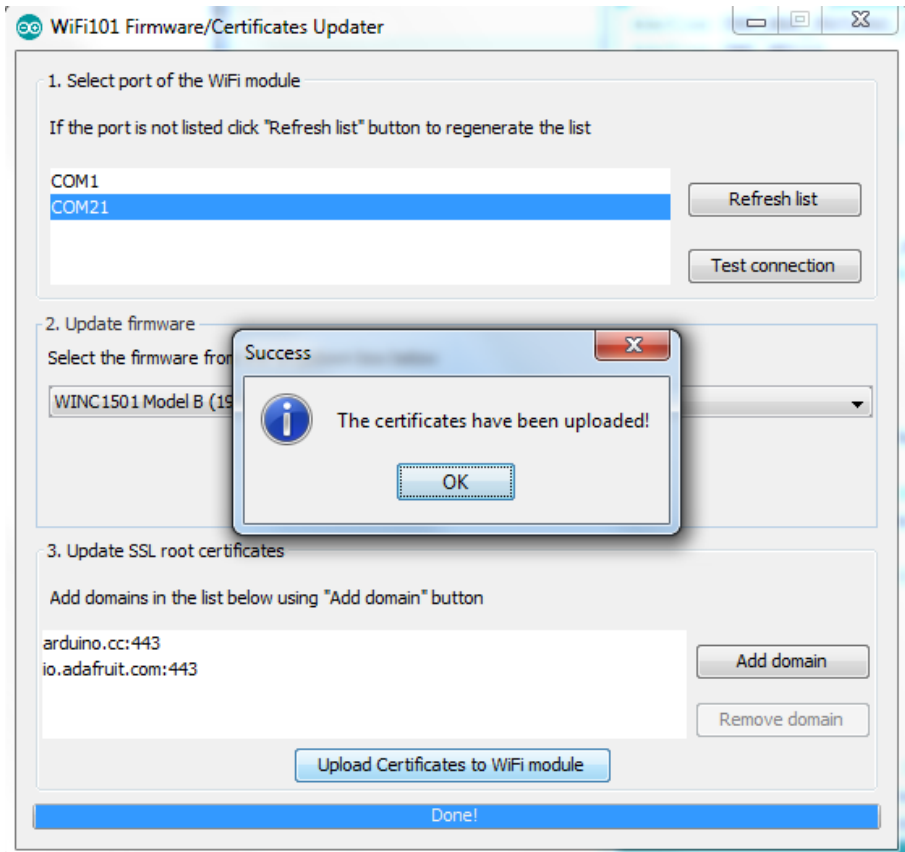
Now at the bottom of the page, click **Add Domain** and type in the URL of the site you want to access:



Then click **Upload Certificates**



A few moments later...success!



What SSL/TLS support is available with the WINC1500?

Officially Atmel lists TLS 1.0 & 1.1, however we have noticed that the firmwares shipping on boards today seem to also support TLS 1.2 (verified by checking the results of [www.howsmyssl.com](http://www.howsmyssl.com) (<http://adafru.it/mgf>)).

The supported ciphers are:

- TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA

# Adapting Sketches to M0

The ATSAM21 is a very nice little chip but its fairly new as Arduino-compatible cores go **Most** sketches & libraries will work but here's a few things we noticed!

## Analog References

If you'd like to use the **ARef** pin for a non-3.3V analog reference, the code to use `isAnalogReference(AR_EXTERNAL)` (it's `AR_EXTERNAL` not `EXTERNAL`)

## Pin Outputs & Pullups

The old-style way of turning on a pin as an input with a pullup is to use

```
pinMode(pin, INPUT)
digitalWrite(pin, HIGH)
```

This is because the pullup-selection register is the same as the output-selection register.

For the M0, you can't do this anymore! Instead, use

```
pinMode(pin, INPUT_PULLUP)
```

which has the benefit of being backwards compatible with AVR.

## Serial vs SerialUSB

99.9% of your existing Arduino sketches use **Serial.print** to debug and give output. For the Official Arduino SAMD/M0 core, this goes to the Serial5 port, which isn't exposed on the Feather. The USB port for the Official Arduino M0 core, is called **SerialUSB** instead.

In the Adafruit M0 Core, we fixed it so that Serial goes to USB when you use a Feather M0 so it will automatically work just fine.

However, on the off chance you are using the official Arduino SAMD core & you want your Serial prints and reads to use the USB port, use **SerialUSB** instead of Serial in your sketch

If you have existing sketches and code and you want them to work with the M0 without a huge find-replace, put

```
#if defined(ARDUINO_SAMD_ZERO) && defined(SERIAL_PORT_USBVIRTUAL)
  // Required for Serial on Zero based boards
  #define Serial SERIAL_PORT_USBVIRTUAL
#endif
```

**right above the first** function definition in your code. For example:

A screenshot of the Arduino IDE window titled "datecalc | Arduino 1.6.5". The window shows a sketch with the following code:

```
1 // Simple date conversions and calculations
2
3 #include <Wire.h>
4 #include "RTClib.h"
5
6 #if defined(ARDUINO_ARCH_SAMD)
7 // for Zero, output on USB Serial console, remove line below if using programming port to program the Zero!
8 #define Serial SerialUSB
9 #endif
10
11 void showDate(const char* txt, const DateTime dt) {
12     Serial.print(txt);
13     Serial.print(' ');
```

## AnalogWrite / PWM

After looking through the SAMD21 datasheet, we've found that some of the options listed in the multiplexer table don't exist on the specific chip used in the Feather M0.

For all SAMD21 chips, there are two peripherals that can generate PWM signals: The Timer/Counter (TC) and Timer/Counter for Control Applications (TCC). Each SAMD21 has multiple copies of each, called 'instances'.

Each TC instance has one count register, one control register, and two output channels. Either channel can be enabled and disabled, and either channel can be inverted. The pins connected to a TC instance can output identical versions of the same PWM waveform, or complementary waveforms.

Each TCC instance has a single count register, but multiple compare registers and output channels. There are options for different kinds of waveform, interleaved switching, programmable dead time, and so on.

The biggest members of the SAMD21 family have five TC instances with two 'waveform output' (WO) channels, and three TCC instances with eight WO channels:

- TC[0-4],WO[0-1]
- TCC[0-2],WO[0-7]

And those are the ones shown in the datasheet's multiplexer tables.

The SAMD21G used in the Feather M0 only has three TC instances with two output channels, and three TCC instances with eight output channels:

- TC[3-5],WO[0-1]
- TCC[0-2],WO[0-7]

Tracing the signals to the pins broken out on the Feather M0, the following pins can't do PWM at all:

- **Analog pin A5**

The following pins can be configured for PWM without any signal conflicts as long as the SPI, I2C, and UART pins keep their protocol functions:

- **Digital pins 5, 6, 9, 10, 11, 12, and 13**
- **Analog pins A3 and A4**

If only the SPI pins keep their protocol functions, you can also do PWM on the following pins:

- **TX and SDA (Digital pins 1 and 20)**



# Missing header files

there might be code that uses libraries that are not supported by the M0 core. For example if you have a line with

```
#include <util/delay.h>
```

you'll get an error that says

```
fatal error: util/delay.h: No such file or directory
#include <util/delay.h>
      ^
compilation terminated.
Error compiling.
```

In which case you can simply locate where the line is (the error will give you the file name and line number) and 'wrap it' with `#ifdef`'s so it looks like:

```
#if !defined(ARDUINO_ARCH_SAM) && !defined(ARDUINO_ARCH_SAMD) && !defined(ESP8266) && !defined(ARDUINO_ARCH_STM32F2)
#include <util/delay.h>
#endif
```

The above will also make sure that header file isn't included for other architectures

If the `#include` is in the arduino sketch itself, you can try just removing the line.

# Bootloader Launching

For most other AVR's, clicking **reset** while plugged into USB will launch the bootloader manually, the bootloader will time out after a few seconds. For the M0, you'll need to *double click* the button. You will see a pulsing red LED to let you know you're in bootloader mode. Once in that mode, it won't time out! Click reset again if you want to go back to launching code

# Aligned Memory Access

This is a little less likely to happen to you but it happened to me! If you're used to 8-bit platforms, you can do this nice thing where you can typecast variables around. e.g.

```
uint8_t mybuffer[4];
float f = (float)mybuffer;
```

You can't be guaranteed that this will work on a 32-bit platform because **mybuffer** might not be aligned to a 2 or 4-byte boundary. The ARM Cortex-M0 can only directly access data on 16-bit boundaries (every 2 or 4 bytes). Trying to access an odd-boundary byte (on a 1 or 3 byte location) will cause a Hard Fault and stop the MCU. Thankfully, there's an easy work around ... just use `memcpy`!

```
uint8_t mybuffer[4];
float f;
memcpy(f, mybuffer, 4)
```

# Floating Point Conversion

Like the AVR Arduinos, the M0 library does not have full support for converting floating point numbers to ASCII strings. Functions like `sprintf` will not convert floating point. Fortunately, the standard AVR-LIBC library includes the `dtostrf` function which can handle the conversion for you.

Unfortunately, the M0 run-time library does not have `dtostrf`. You may see some references to using `#include <avr/dtostrf.h>` to get `dtostrf` in your code. And while it will compile, it does **not** work.

Instead, check out this thread to find a working `dtostrf` function you can include in your code:

<http://forum.arduino.cc/index.php?topic=368720.0> (<http://adafru.it/IFS>)

## How Much RAM Available?

The ATSAM21G18 has 32K of RAM, but you still might need to track it for some reason. You can do so with this handy function:

```
extern "C" char *sbrk(int i);

int FreeRam () {
  char stack_dummy = 0;
  return &stack_dummy - sbrk(0);
}
```

Thx to <http://forum.arduino.cc/index.php?topic=365830.msg2542879#msg2542879> (<http://adafru.it/m6D>) for the tip!

## Storing data in FLASH

If you're used to AVR, you've probably used **PROGMEM** to let the compiler know you'd like to put a variable or string in flash memory to save on RAM. On the ARM, its a little easier, simply add **const** before the variable name:

```
const char str[] = "My very long string";
```

That string is now in FLASH. You can manipulate the string just like RAM data, the compiler will automatically read from FLASH so you dont need special progmem-knowledgeable functions.

You can verify where data is stored by printing out the address:

```
Serial.print("Address of str $"); Serial.println((int)&str, HEX);
```

If the address is \$2000000 or larger, its in SRAM. If the address is between \$0000 and \$3FFFF Then it is in FLASH

# Feather HELP!

My Feather stopped working when I unplugged the USB!

A lot of our example sketches have a

```
while (!Serial);
```

line in setup(), to keep the board waiting until the USB is opened. This makes it a lot easier to debug a program because you get to see all the USB data output. If you want to run your Feather without USB connectivity, delete or comment out that line

My Feather never shows up as a COM or Serial port in the Arduino IDE

## A vast number of Feather 'failures' are due to charge-only USB cables

We get upwards of 5 complaints a day that turn out to be due to charge-only cables!

Use only a cable that you **know** is for data syncing

If you have any charge-only cables, cut them in half throw them out. We are serious! They tend to be low quality in general, and will only confuse you and others later, just get a good data+charge USB cable

Ack! I "did something" and now when I plug in the Feather, it doesn't show up as a device anymore so I cant upload to it or fix it...

No problem! You can 'repair' a bad code upload easily. Note that this can happen if you set a watchdog timer or sleep mode that stops USB, or any sketch that 'crashes' your Feather

1. Turn on **verbose upload** in the Arduino IDE preferences
2. Plug in feather 32u4/M0, it won't show up as a COM/serial port that's ok
3. Open up the Blink example (Examples->Basics->Blink)
4. Select the correct board in the Tools menu, e.g. Feather 32u4 or Feather M0 (check your board to make sure you have the right one selected!)
5. Compile it (make sure that works)
6. Click Upload to attempt to upload the code
7. The IDE will print out a bunch of COM Ports as it tries to upload **During this time, double-click the reset button, you'll see the red pulsing LED that tells you its now in bootloading mode**
8. The Feather will show up as the Bootloader COM/Serial port
9. The IDE should see the bootloader COM/Serial port and upload properly

```
Blink | Arduino 1.6.5
File Edit Sketch Tools Help
Blink
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  Done uploading.

Sketch uses 4,788 bytes (16%) of program storage space. Maximum is 28,672 bytes.
Global variables use 151 bytes (5%) of dynamic memory, leaving 2,409 bytes for local variables. Maximum
Forcing reset using 1200bps open/close on port COM12
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, } => {}
PORTS {COM1, COM12, } / {COM1, COM12, COM69, } => {COM69, }
Found upload port: COM69

C:\Users\ladyada\Documents\Projects\arduino\arduino-1.6.5-r5\hardware\tools\avr\bin\avrdude
-CC:\Users\ladyada\Documents\Projects\arduino\arduino-1.6.5-r5\hardware\tools\avr\etc\avrdude.conf -v -p
-Uflash:w:C:\Users\ladyada\AppData\Local\Temp\build697907979161753686.tmp/Blink.cpp.hex:i

avrdude: Version 6.0.1, compiled on Apr 15 2015 at 19:59:58

Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/

Copyright (c) 2007-2009 Joerg Wunsch

Arduino Leonardo on COM12
```

I can't get the Feather USB device to show up - I get "USB Device Malfunctioning" errors!

This seems to happen when people select the wrong board from the Arduino Boards menu.

If you have a Feather 32u4 (look on the board to read what it is you have) Make sure you select **Feather 32u4** for ATmega32u4 based boards! Do not use anything else, do not use the 32u4 breakout board line.

If you have a Feather M0 (look on the board to read what it is you have) Make sure you select **Feather M0** - do not use 32u4 or Arduino Zero

I'm having problems with COM ports and my Feather 32u4/M0

There's **two** COM ports you can have with the 32u4/M0, one is the **user port** and one is the **bootloader port**. They are not the same COM port number!

When you upload a new user program it will come up with a user com port, particularly if you use Serial in your user program.

**If you crash your user program, or have a program that halts or otherwise fails, the user com port can disappear.**

**When the user COM port disappears, Arduino will not be able to automatically start the bootloader and upload new software.**

So you will need to help it by performing the click-during upload procedure to re-start the bootloader, and upload something that is known working like "Blink"

I don't understand why the COM port disappears, this does not happen on my Arduino UNO!

UNO-type Arduinos have a *seperate* serial port chip (aka "FTDI chip" or "Prolific PL2303" etc etc) which handles all serial port capability seperately than the main chip. This way if the main chip fails, you can always use the COM port.

M0 and 32u4-based Arduinos do not have a seperate chip, instead the main processor performs this task for you. It allows for a lower cost, higher power setup...but requires a little more effort since you will need to 'kick' into the bootloader manually once in a while

I'm trying to upload to my 32u4, getting "avrdude: butterfly\_recv(): programmer is not responding" errors

This is likely because the bootloader is not kicking in and you are accidentally **trying to upload to the wrong COM port**

The best solution is what is detailed above: manually upload Blink or a similar working sketch by hand by manually launching the bootloader

I'm trying to upload to my Feather M0, and I get this error "Connecting to programmer: .avrdude: butterfly\_recv(): programmer is not responding"

You probably don't have Feather M0 selected in the boards drop-down. Make sure you selected Feather M0.

I'm trying to upload to my Feather and i get this error "avrdude: ser\_recv(): programmer is not responding"

You probably don't have Feather M0 / Feather 32u4 selected in the boards drop-down. Make sure you selected Feather M0 (or Feather 32u4).

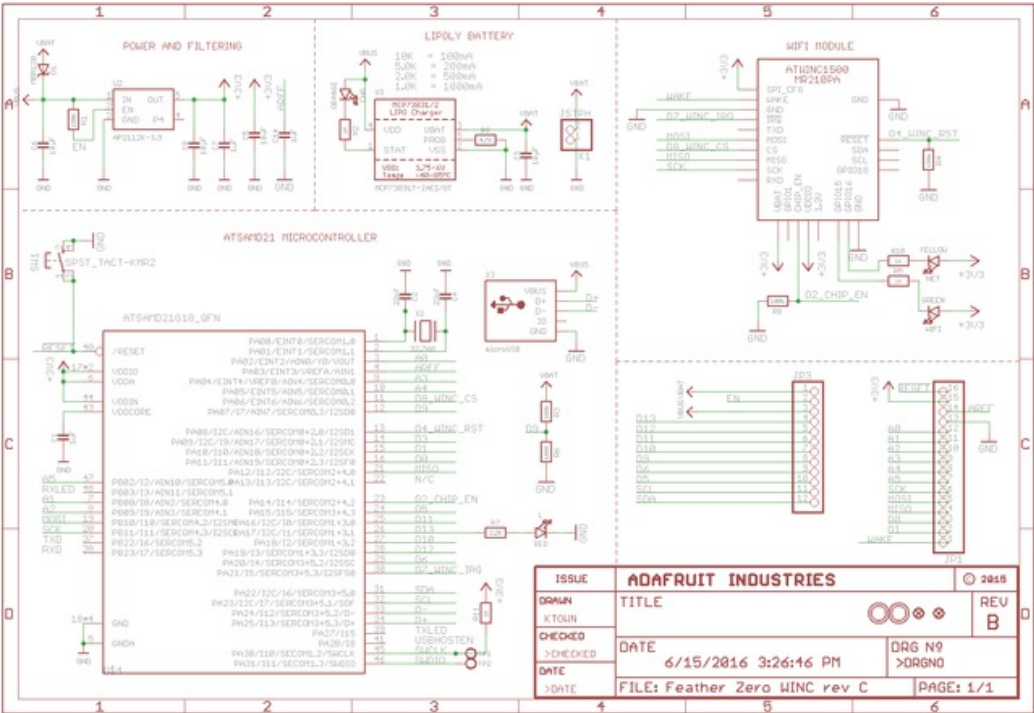
# Downloads

# Datasheets & Files

- [Atmel Software Programming guide for WINC1500](http://adafru.it/ldD) (http://adafru.it/ldD) - this is for the underlying ASF codebase that is 'wrapped' in Adafuit\_WINC1500 but its still very handy reference
- [ATSAMD21 Datasheet](http://adafru.it/ldE) (http://adafru.it/ldE) - Its long, but its a good read
- [EagleCAD PCB Files on GitHub](http://adafru.it/oeK) (http://adafru.it/oeK)
- [Fritzing object in the Adafuit Fritzing library](http://adafru.it/aP3) (http://adafru.it/aP3)

# Schematic

Click to enlarge



# Fabrication Print

Dimensions in inches

