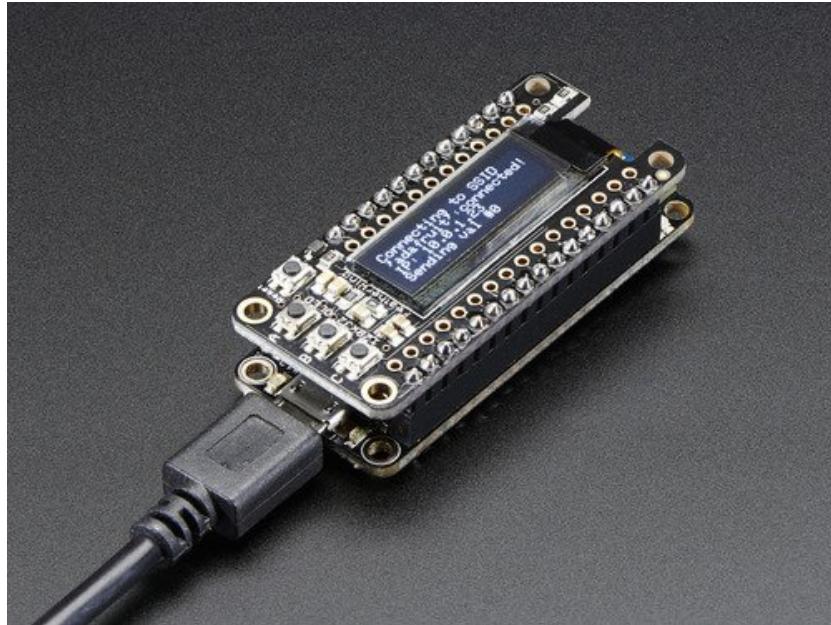




Adafruit Feather M0 WiFi with ATWINC1500

Created by lady ada



Last updated on 2016-04-22 12:25:31 PM EDT

Guide Contents

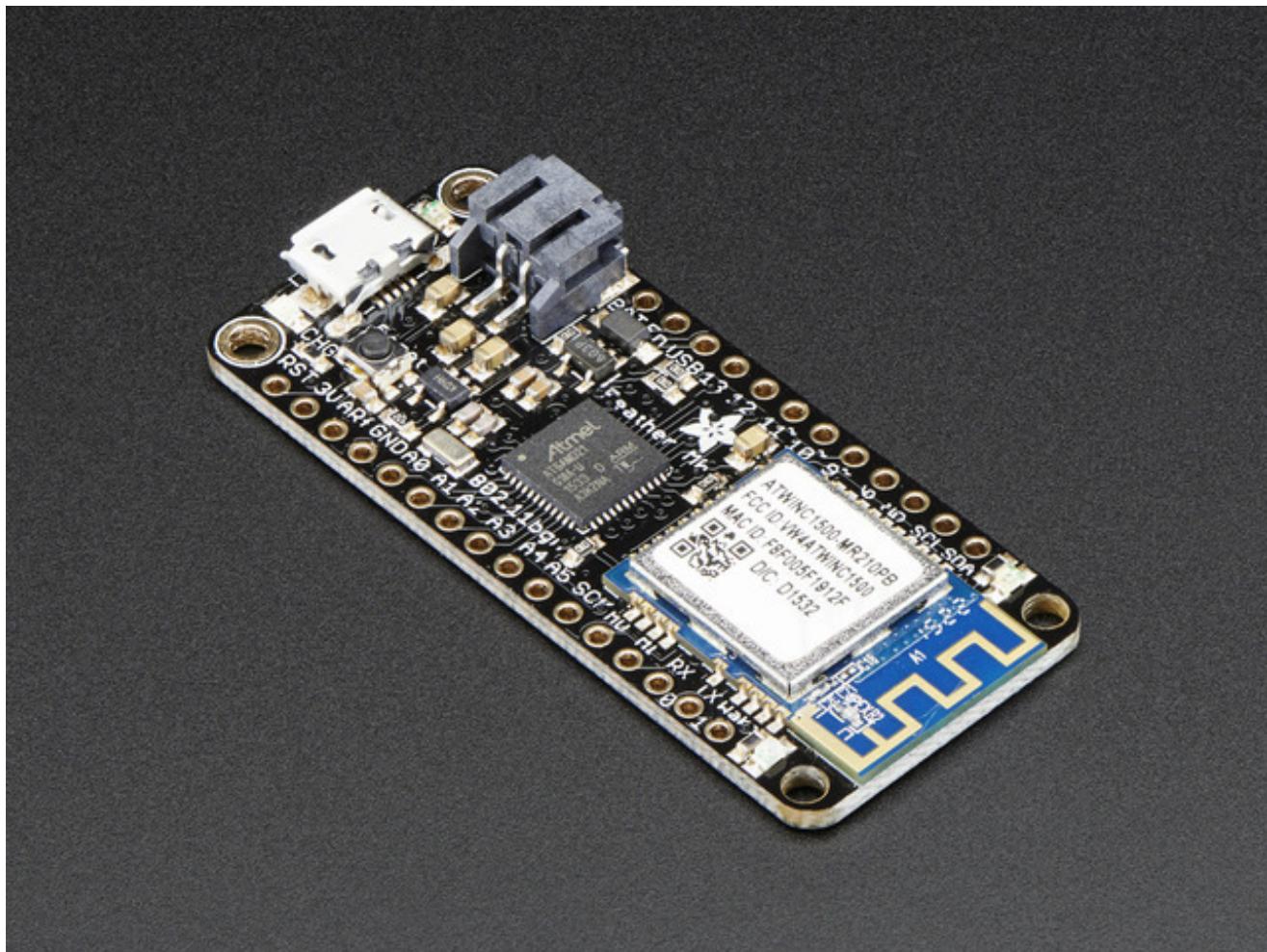
Guide Contents	2
Overview	4
Pinouts	10
Power Pins	10
Logic pins	11
WiFi Module & LEDs	12
Other Pins!	12
Assembly	14
Header Options!	14
Soldering in Plain Headers	15
Prepare the header strip:	15
Add the breakout board:	16
And Solder!	16
Soldering on Female Header	17
Tape In Place	17
Flip & Tack Solder	18
And Solder!	18
Power Management	20
Battery + USB Power	20
Power supplies	21
Measuring Battery	22
ENable pin	22
Power Usage & Saving with WiFi	22
Arduino IDE Setup	26
Using with Arduino IDE	29
Install SAMD Support	29
Install Adafruit SAMD	30
Install Drivers (Windows Only)	31
Blink	33
Sucessful Upload	34
Compilation Issues	35
Manually bootloading	36

Ubuntu & Linux Issue Fix	36
Using the WiFi Module	38
Download the Adafruit Library	38
Check Connections & Version	39
Scanning WiFi	40
Connect & Read Webpage	41
Creating an Access Point + Webserver	42
Updating SSL Certificates	48
Command Line Usage	51
Windows	51
Mac OS X (Command Line) Usage	52
GUI Usage	52
Manually Adding Certificates	54
Certificate Format	59
Adapting Sketches to M0	63
Analog References	63
Pin Outputs & Pullups	63
Serial vs SerialUSB	63
AnalogWrite / PWM	64
Missing header files	64
Bootloader Launching	65
Aligned Memory Access	65
Floating Point Conversion	65
How Much RAM Available?	65
Storing data in FLASH	66
FAQ	67
Downloads	68
Datasheets	68

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.

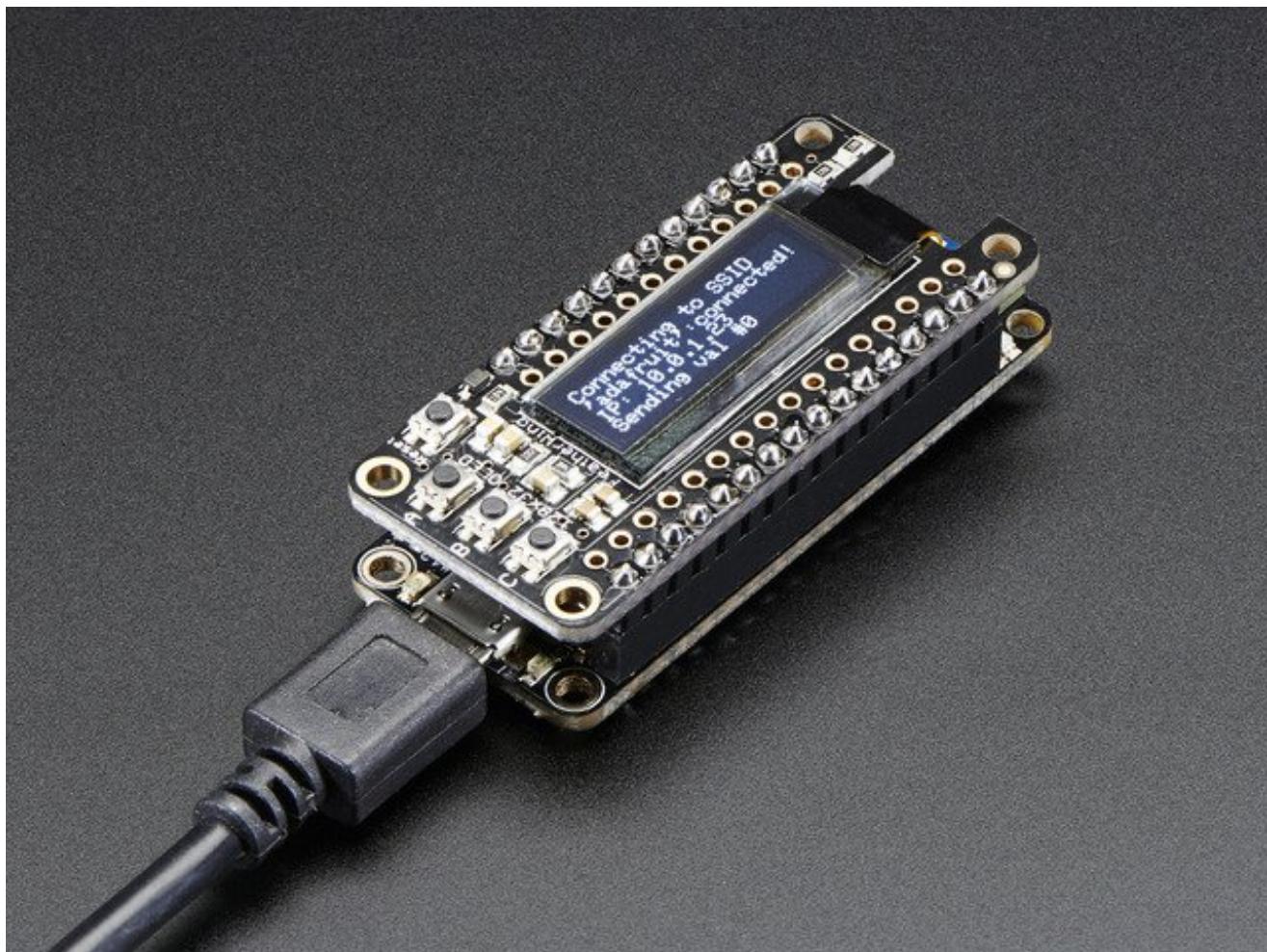
This is the **Adafruit Feather M0 WiFi** w/ATWINC1500 - our take on an 'all-in-one' Arduino-compatible + high speed, reliable WiFi with built in USB and battery charging. Its an Adafruit Feather M0 [with a WiFi module](#) (<http://adafru.it/2999>), ready to rock! [We have other boards in the Feather family, check'em out here](#) (<http://adafru.it/l7B>).



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 capabilites, 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

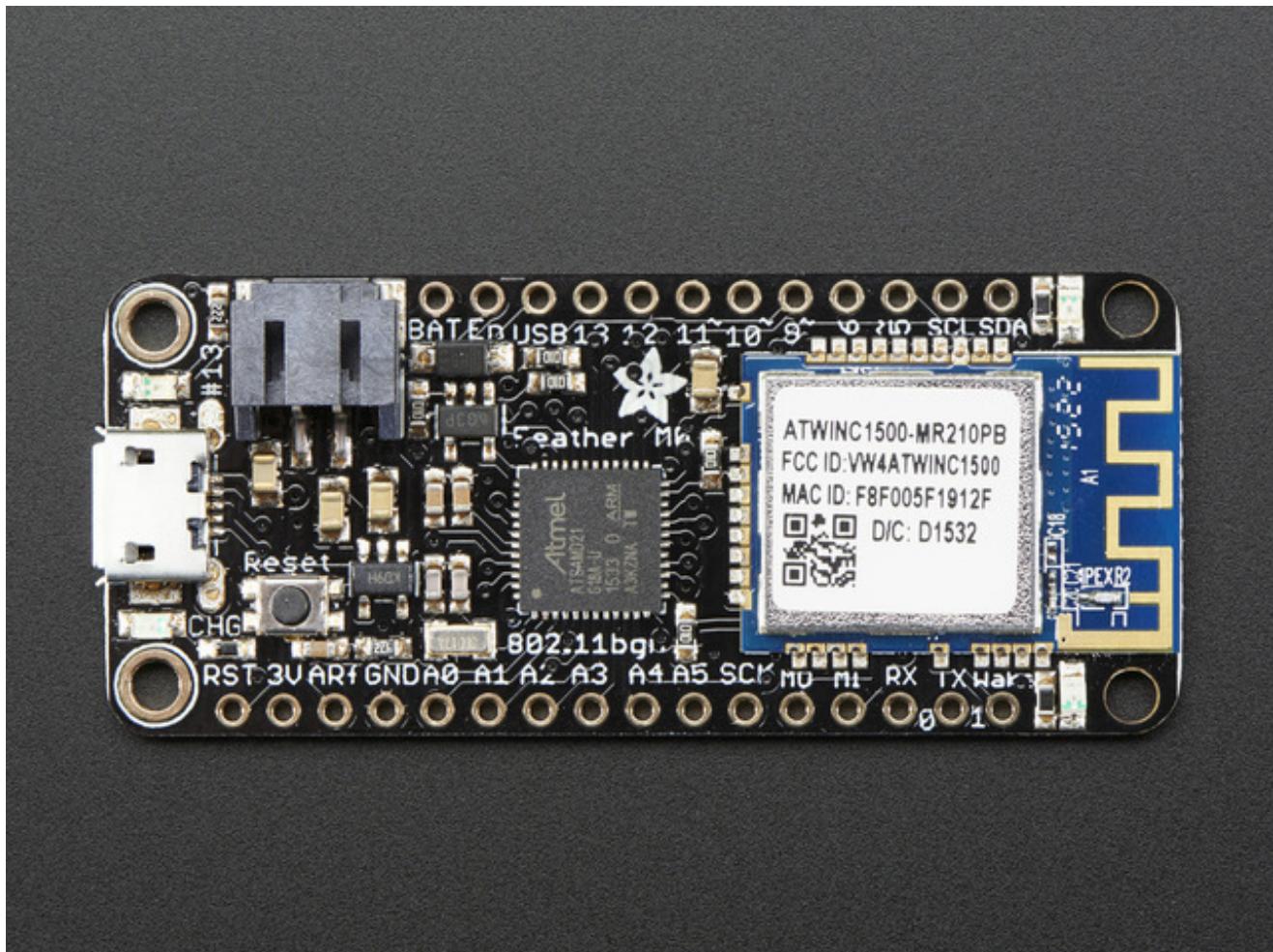
own access point (we have an example of it creating a webserver that you can then control the Arduino's pins). You can clock it as fast as 12MHz for speedy, reliable packet streaming. And scanning/connecting to networks is very fast, just a second or two.



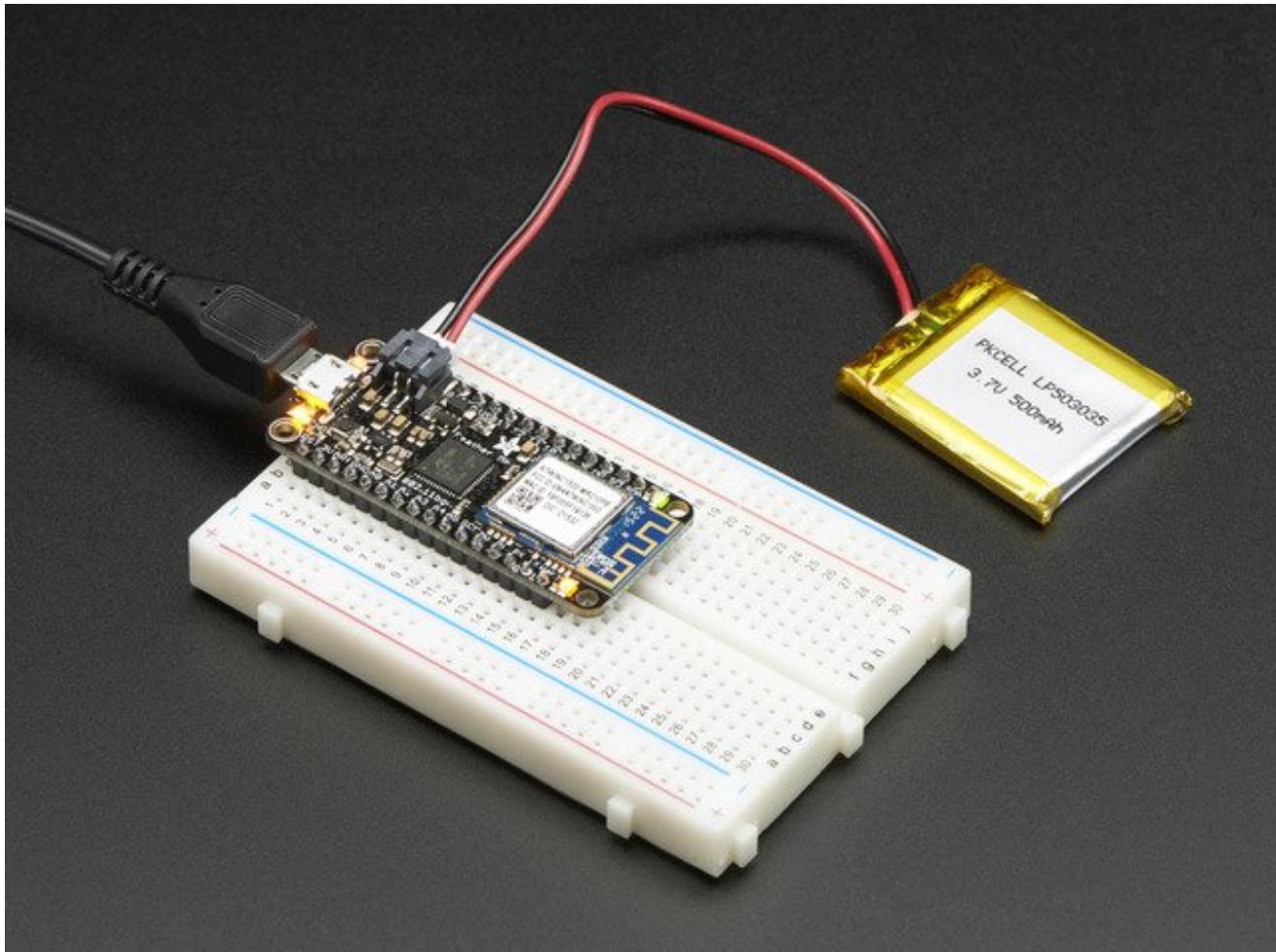
You might be wondering why use this [when you can get a HUZZAH Feather?](http://adafru.it/2821) (<http://adafru.it/2821>) Well, you get

- 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://adafru.it/I3e) (<http://adafru.it/I3e>))
- The ATWINC has much lower power usage, about 12mA for the WINC & 10mA for the ATSAMD21 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

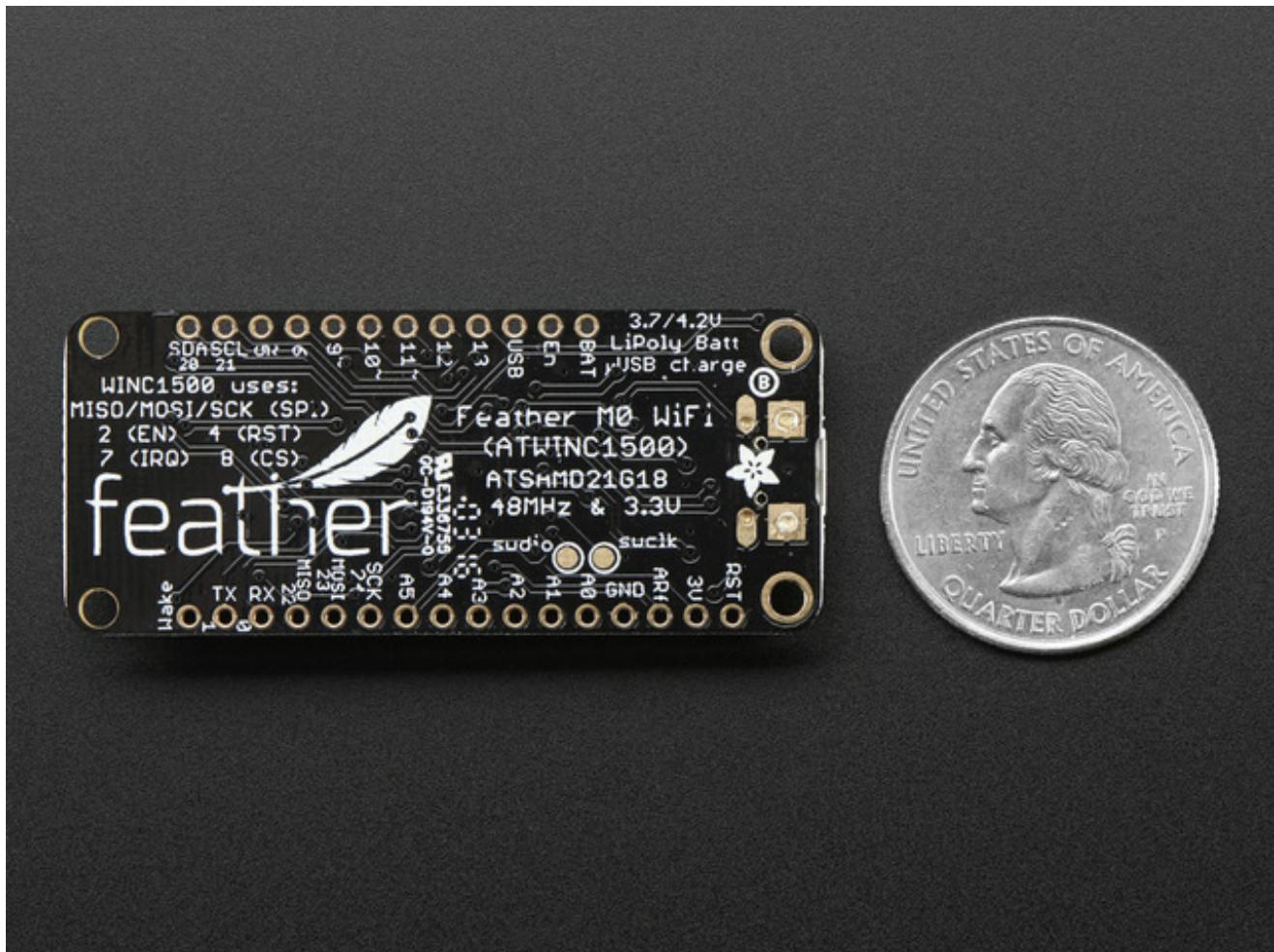
Of course, both WiFi-capable Feathers have their strengths and tradeoffs, & we love both equally!



At the Feather M0's heart is an ATSAMD21G18 ARM Cortex M0 processor, clocked at 48 MHz and at 3.3V logic, the same one used in the new [Arduino Zero](http://adafru.it/2843) (<http://adafru.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.

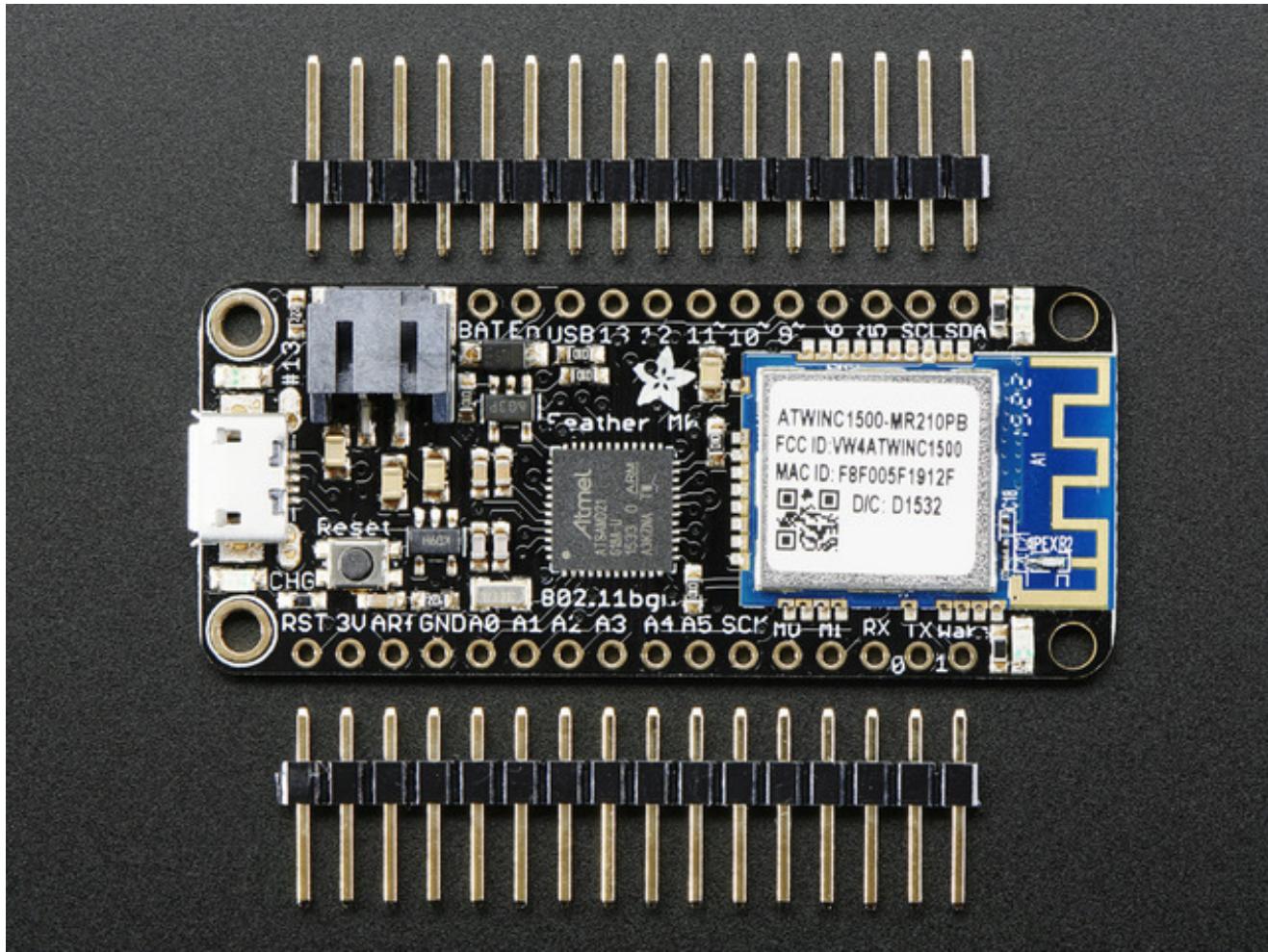


To make it easy to use for portable projects, we added a connector for any of our 3.7V Lithium polymer batteries and built in battery charging. You don't need to use a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then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

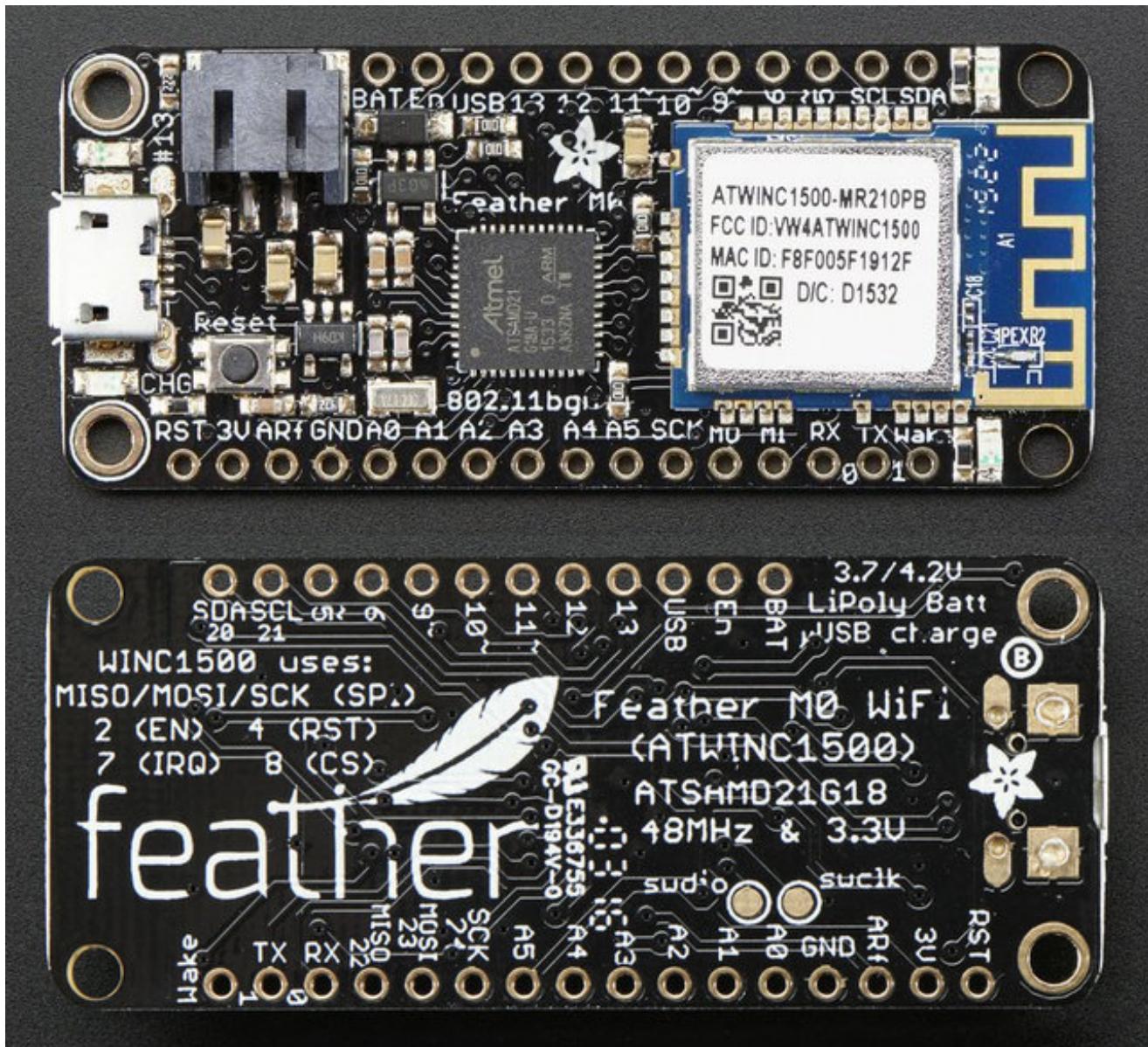
- Measures 2.1" x 0.9" x 0.3" (53.65mm x 23mm x 8mm) without headers soldered in. Note it is 0.1" longer than most Feathers
- Light as a (large?) feather - 6.1 grams
- ATSAMD21G18 @ 48MHz with 3.3V logic/power
- 256KB FLASH, 32KB SRAM, No EEPROM
- 3.3V regulator (AP2112K-3.3) with 600mA peak current 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



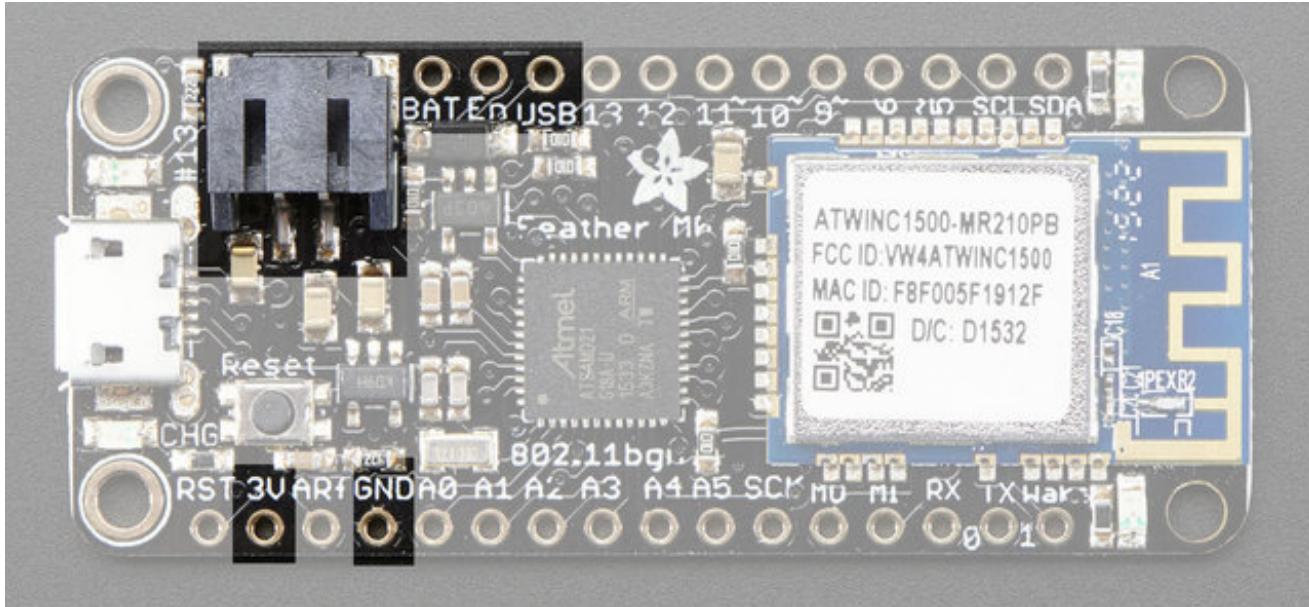
Comes fully assembled and tested, with a USB bootloader that lets you quickly use it with the Arduino IDE. We also toss in some header so you can solder it in and plug into a solderless breadboard. [Lipoly battery](http://adafruit.it/e0v) (<http://adafruit.it/e0v>) and [MicroUSB cable](http://adafruit.it/aM5) (<http://adafruit.it/aM5>) **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

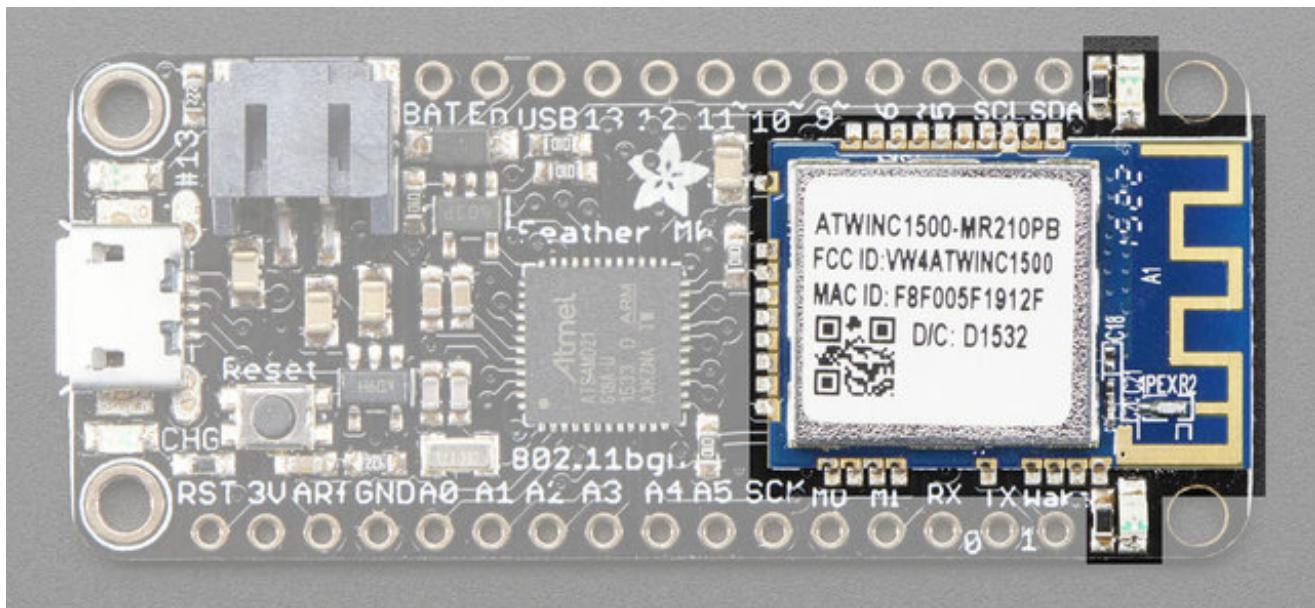
All pins can do PWM output

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

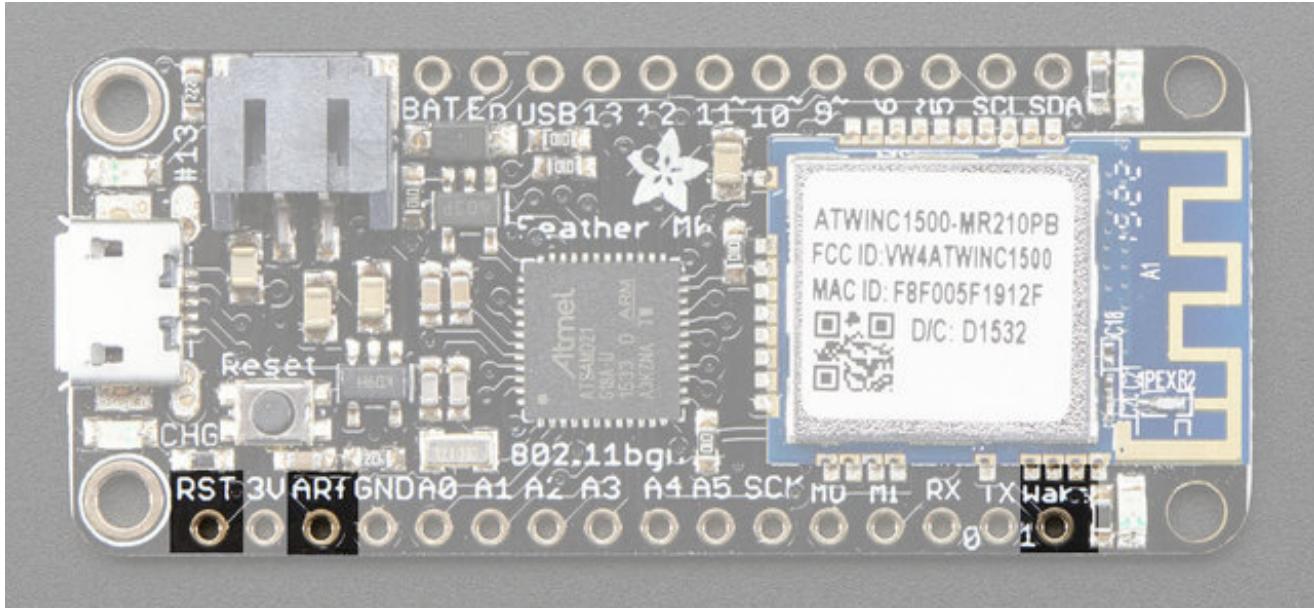
- **#2** - used as the **ENable** pin for the WiFi module, by default pulled down low, set HIGH to enable WiFi
- **#4** - used as the **Reset** pin for the WiFi module, controlled by the library
- **#7** - used as the **IRQ** interrupt request pin for the WiFi module, controlled by the library
- **#8** - 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!

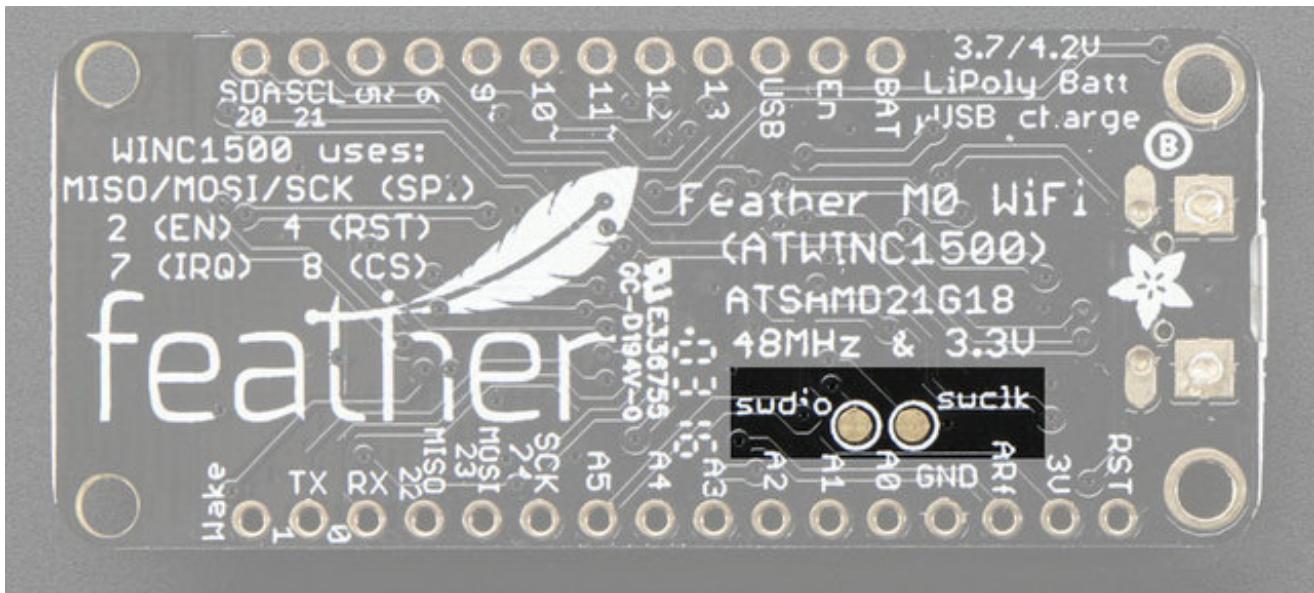
- **RST** - this is the Reset pin, tie to ground to manually reset the AVR, as well as launch the bootloader manually
- **ARef** - the analog reference pin. Normally the reference voltage is the same as the chip logic

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.

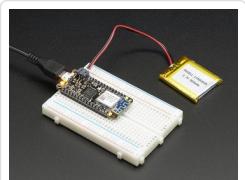


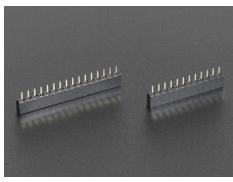
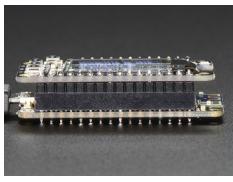
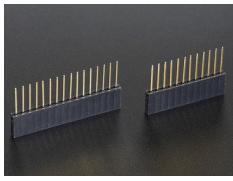
Assembly

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

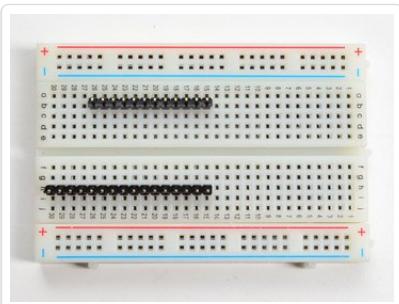
Header Options!

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

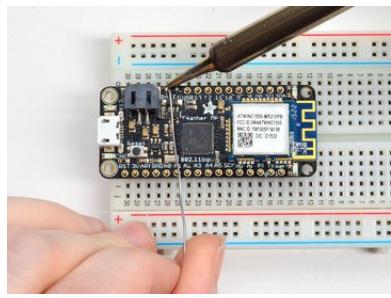
-  We also have 'slim' versions of the female headers, that are a little shorter and give a more compact shape
- 
-  Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a solderless breadboard *and* plug a featherwing on top. But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

Soldering in Plain Headers



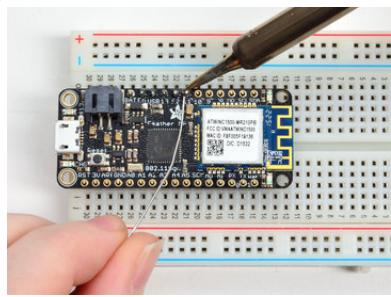
Prepare the header strip:

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:

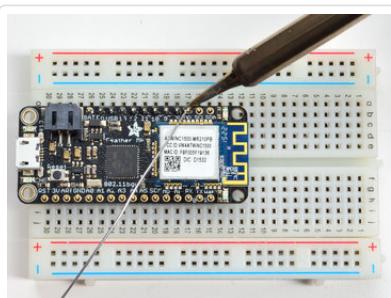
Place the breakout board over the pins so that the short pins poke through the breakout pads

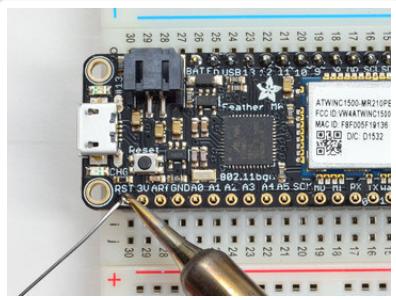


And Solder!

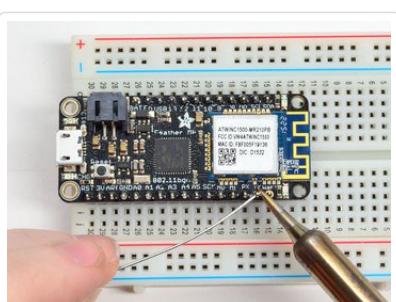
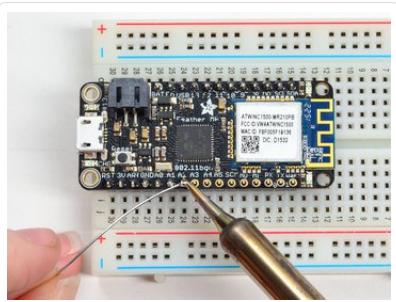
Be sure to solder all pins for reliable electrical contact.

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

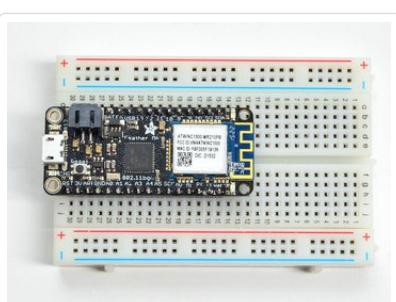




Solder the other strip as well.



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



Soldering on Female Header

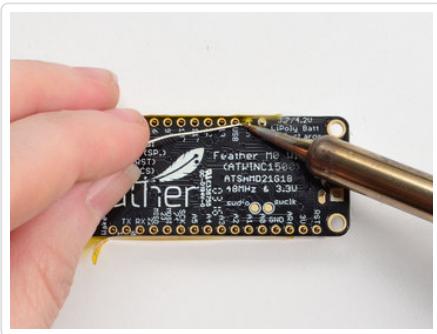
-



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

-



-

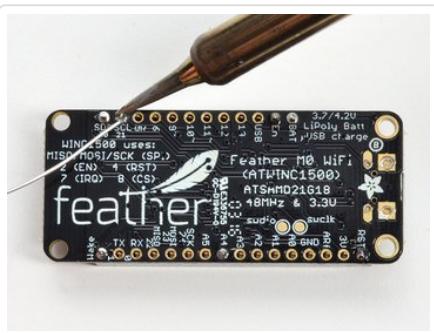
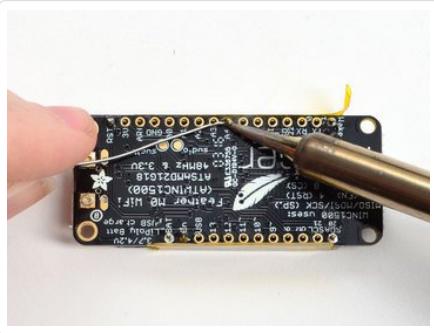


Flip & Tack Solder

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>)).

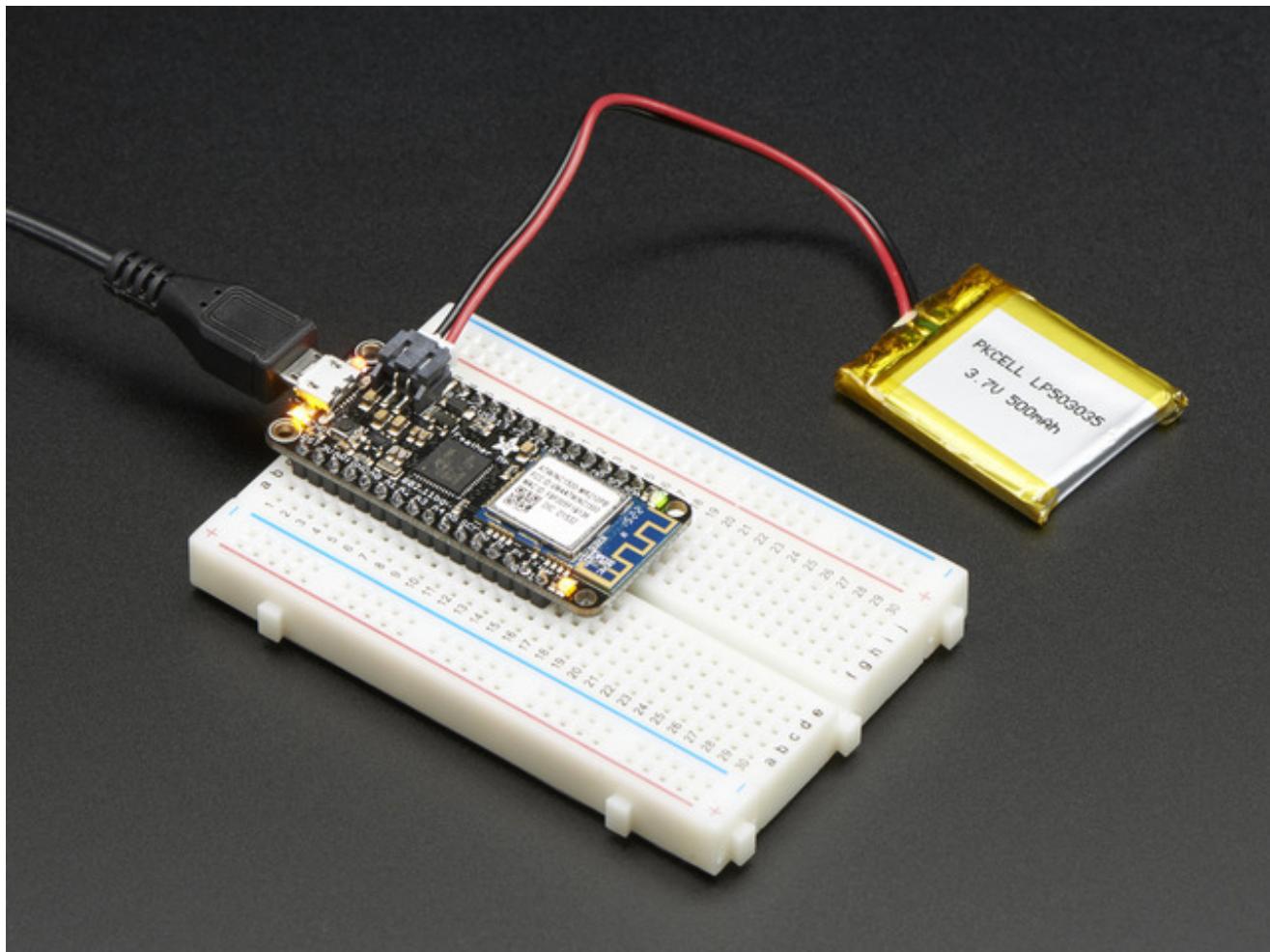




You're done! Check your solder joints visually and continue onto the next steps



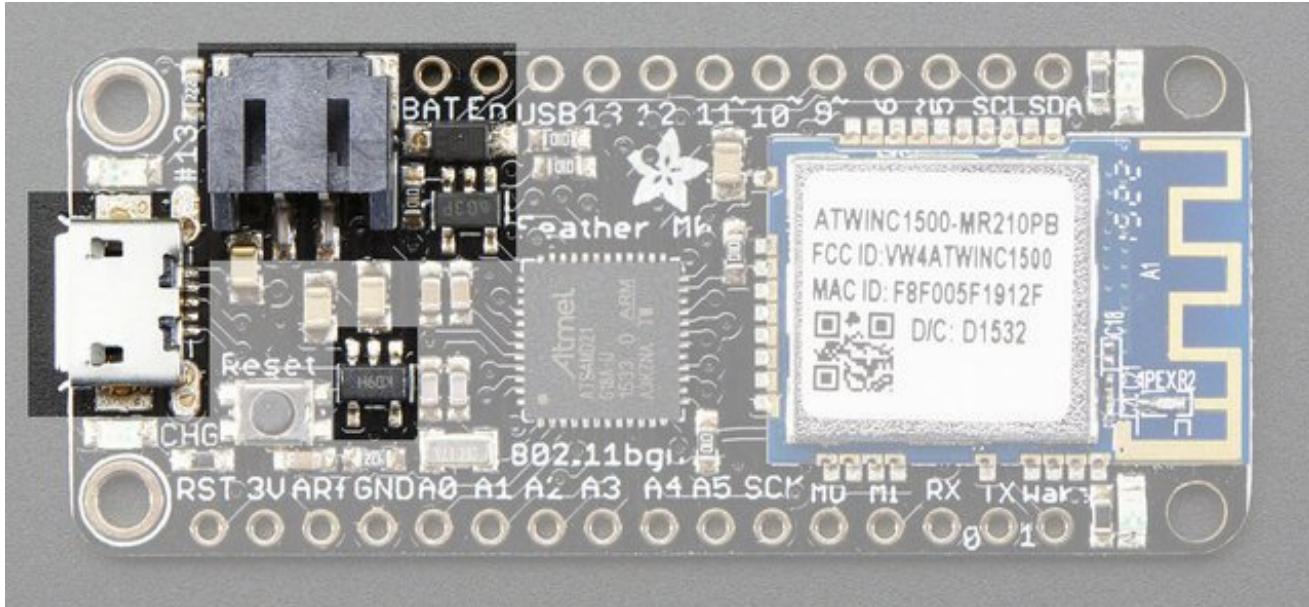
Power Management



Battery + USB Power

We wanted to make the Feather 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.

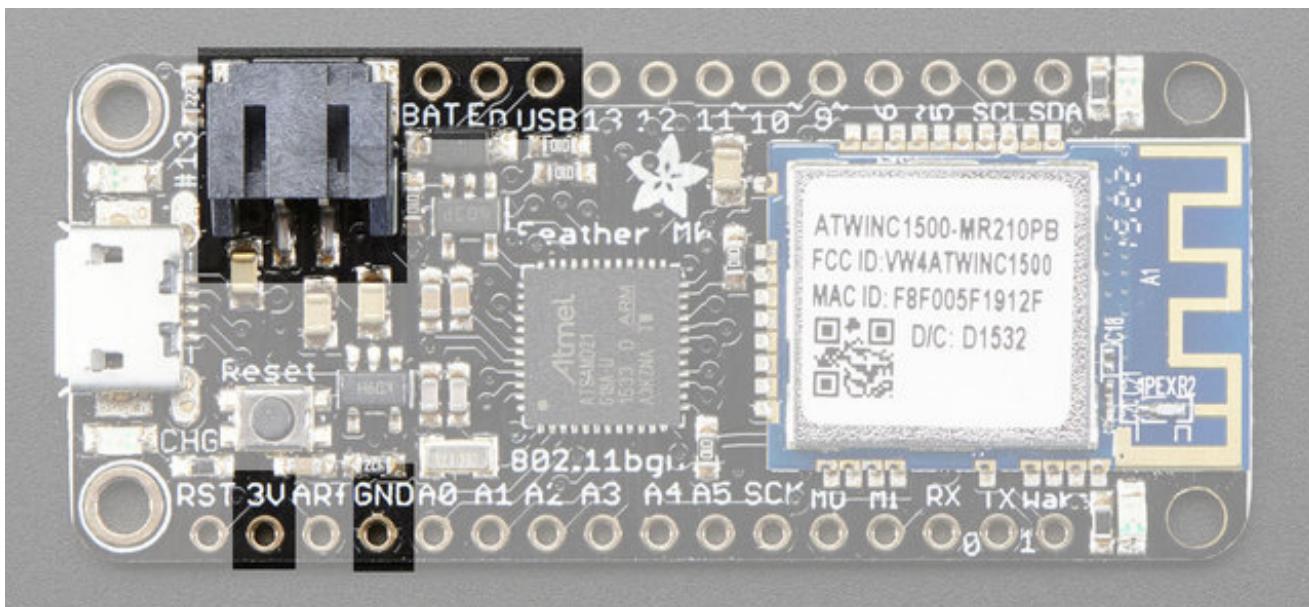
The JST connector polarity is matched to Adafruit LiPoly batteries. Using wrong polarity batteries can destroy your Feather



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.



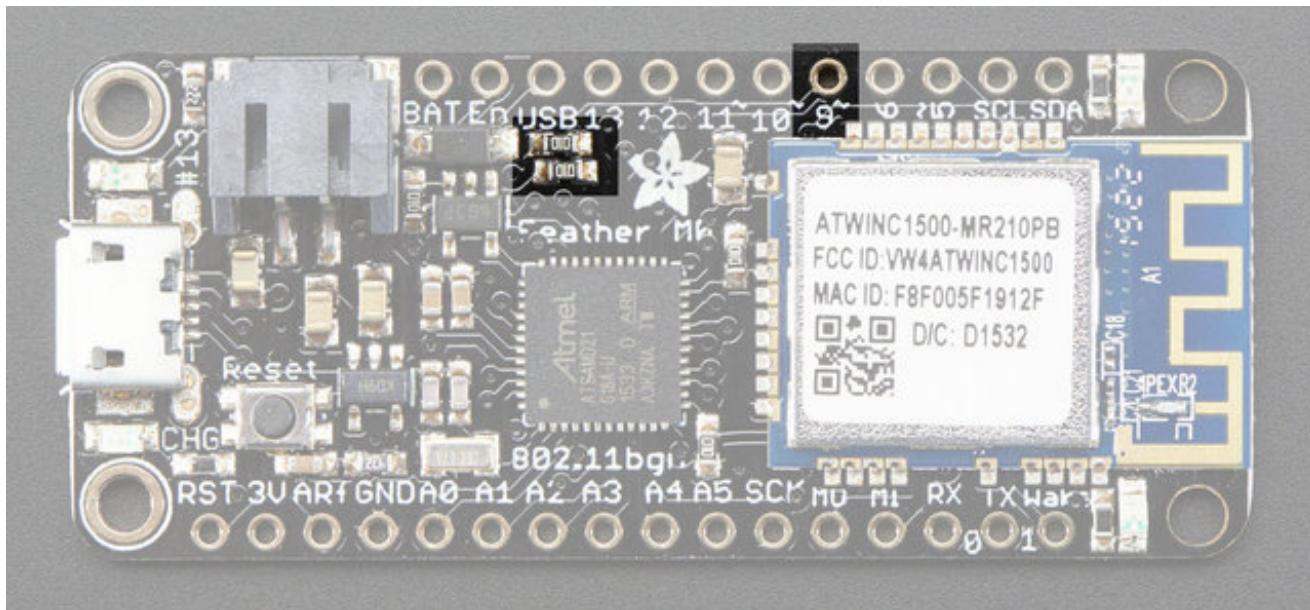
Measuring Battery

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);
```



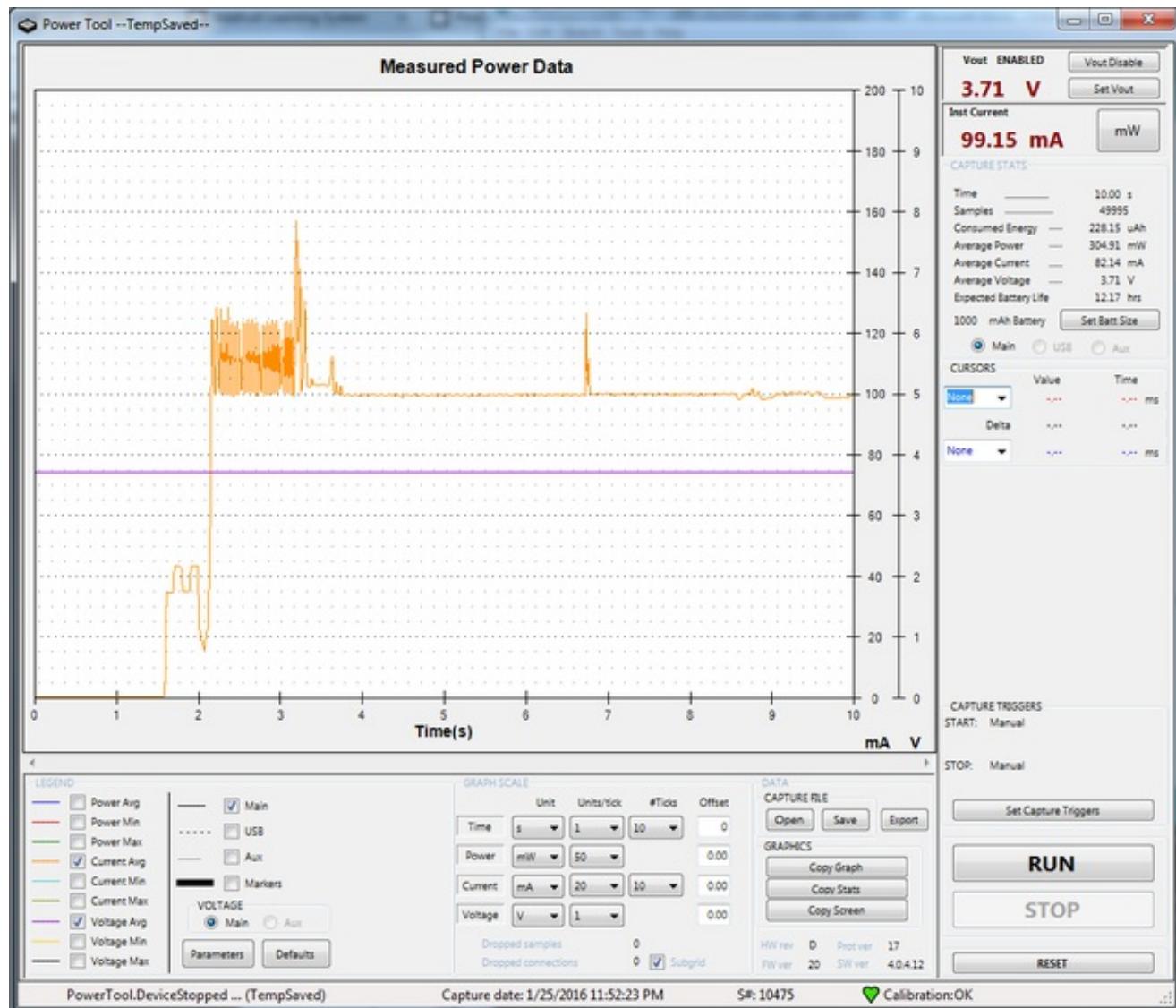
ENable pin

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

sents a packet every 5 seconds or so:

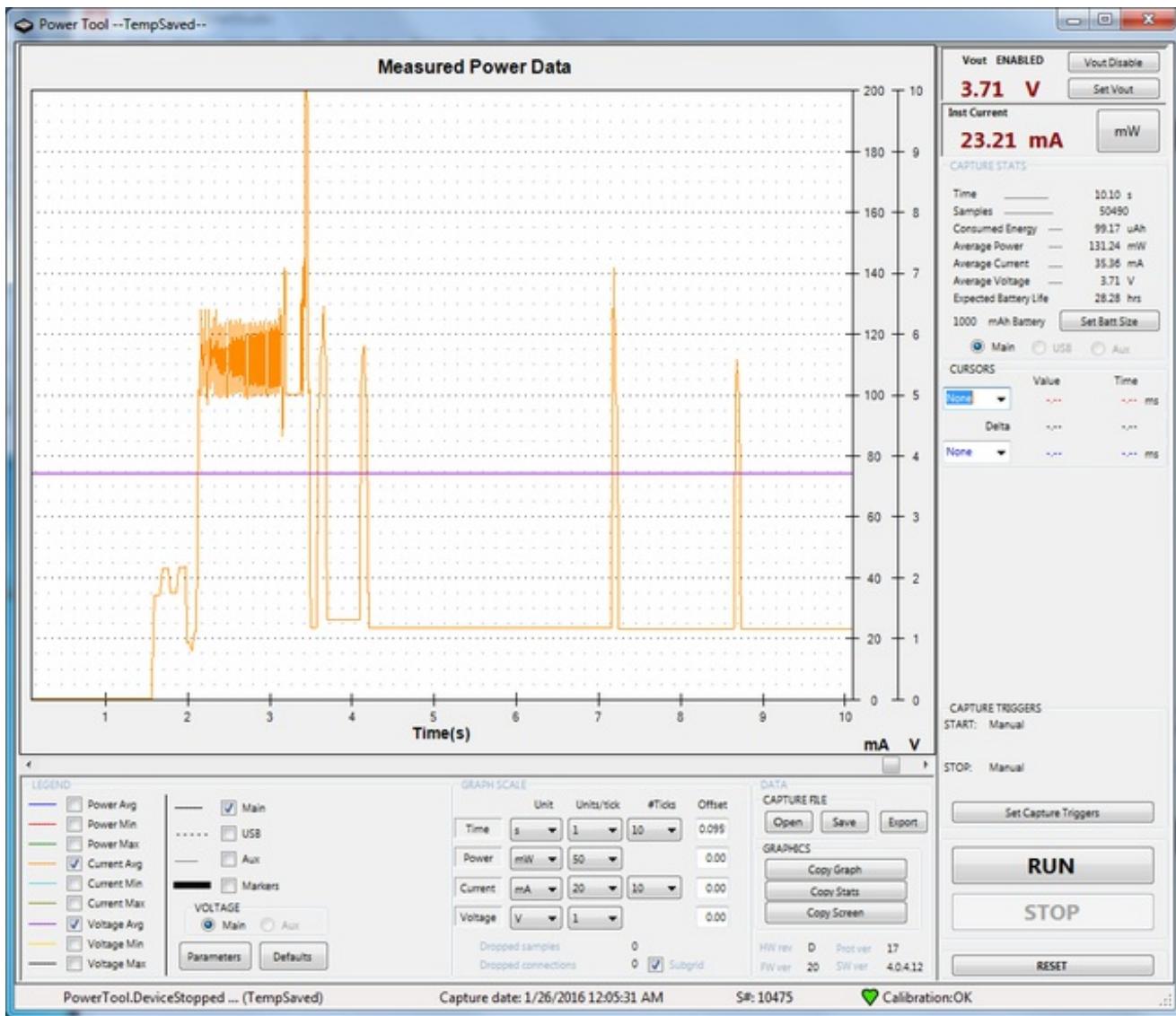


You can see the chip launch at about 1.5 seconds, then turn on the WiFi and at about 2s make the SSID connection and MQTT connection. The average current is about 100ms afterwards, and a packet spikes up to ~130mA at the 7 second mark.

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 nothing's going on, shut down unneeded parts. You don't 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)



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

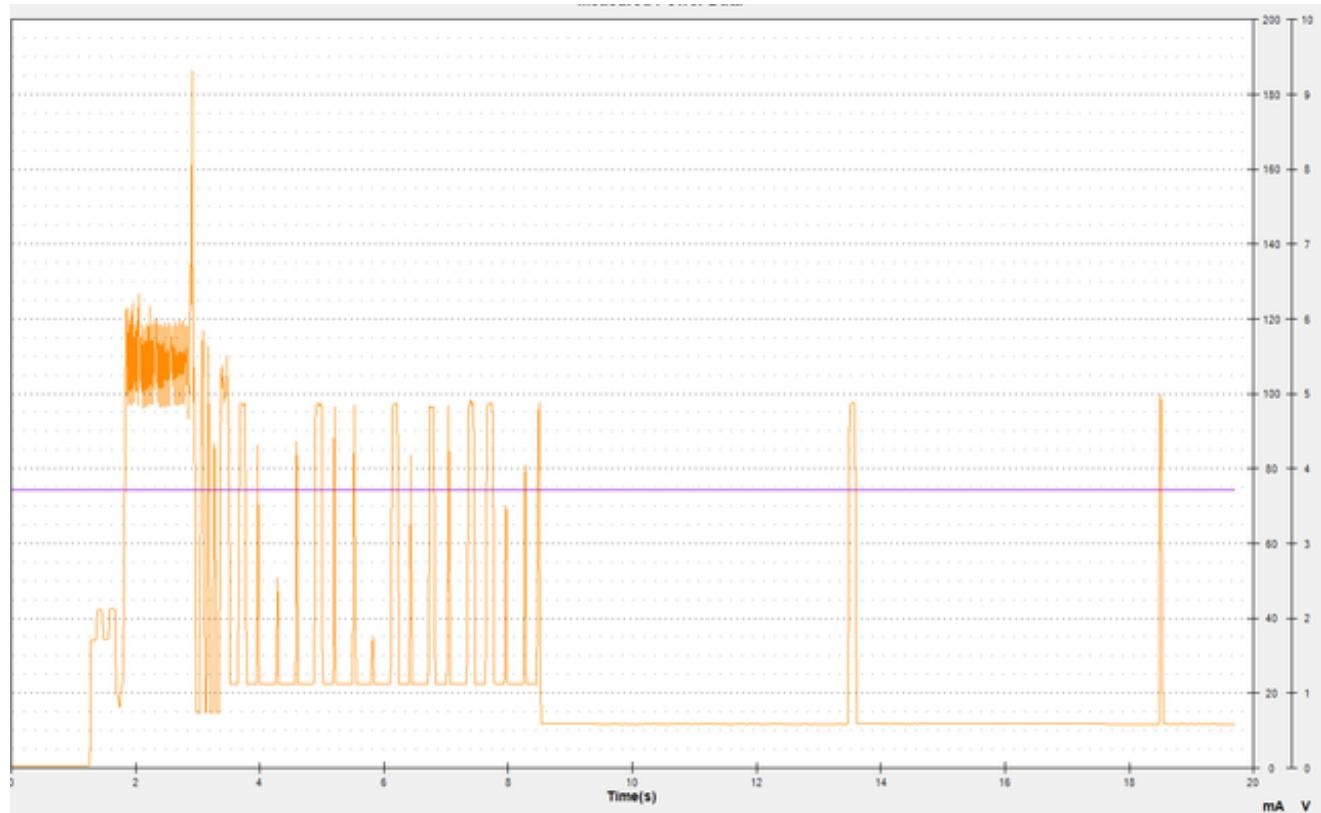
```
WiFi.setSleepMode(M2M_PS_MANUAL, 1);
```

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 spiky 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!



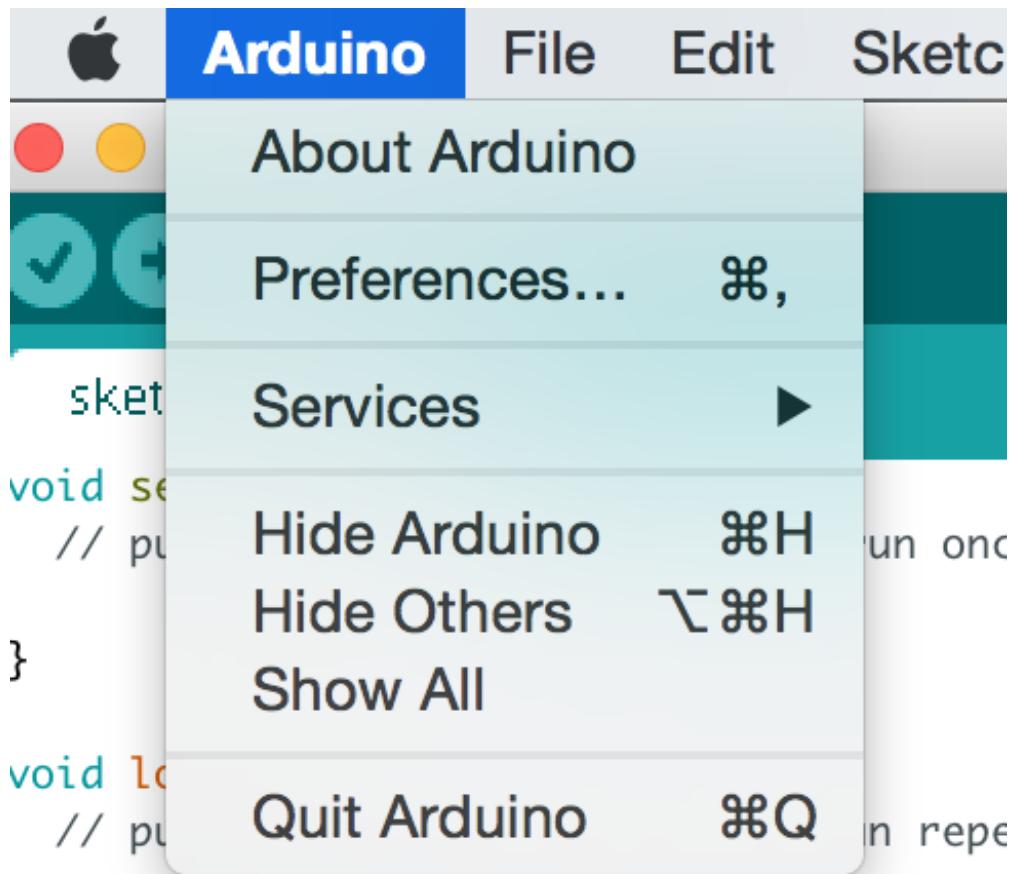
Arduino IDE Setup

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.

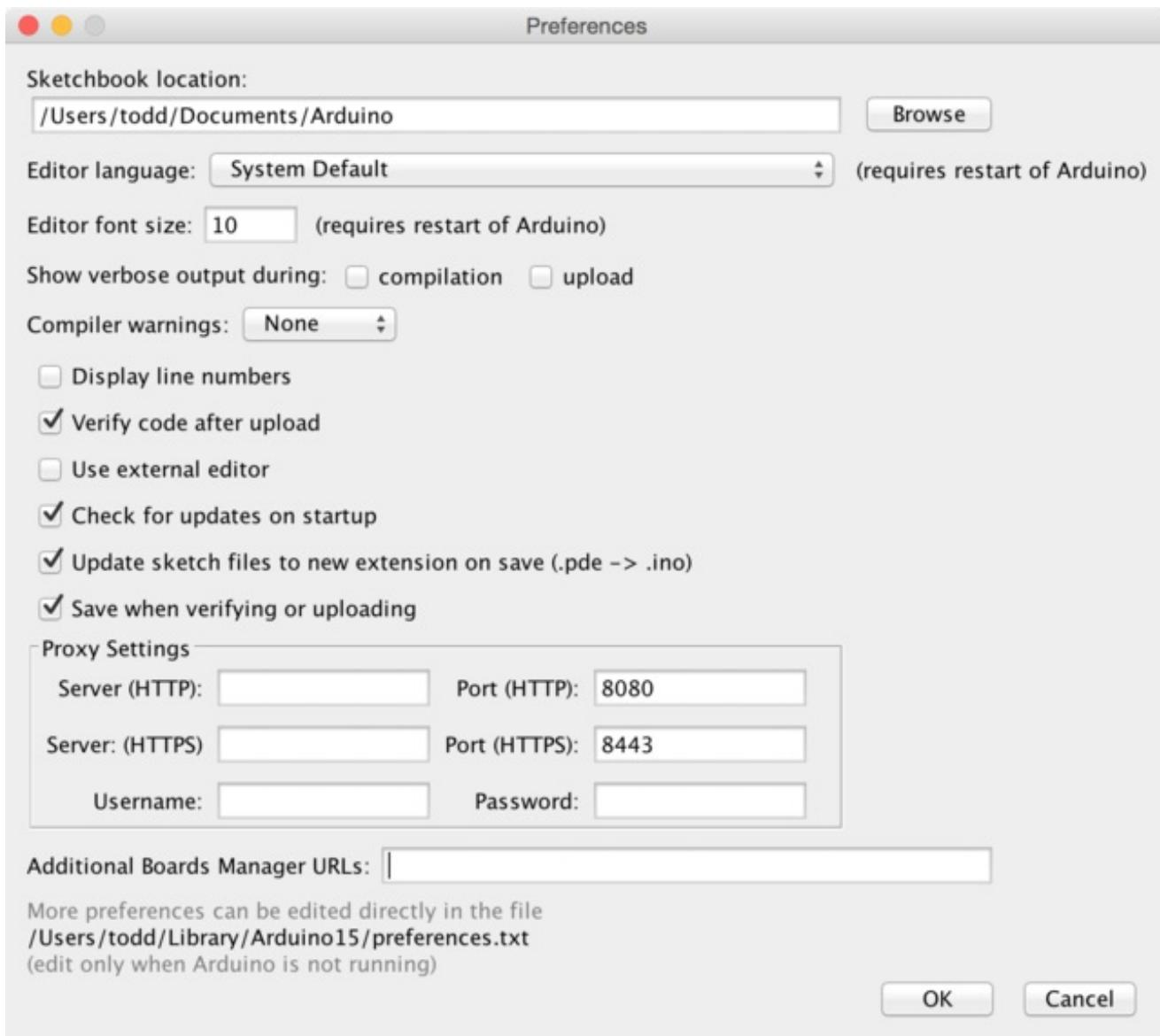
[Arduino IDE v1.6.4+ Download](https://www.arduino.cc/en/Main/Software)

<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*.



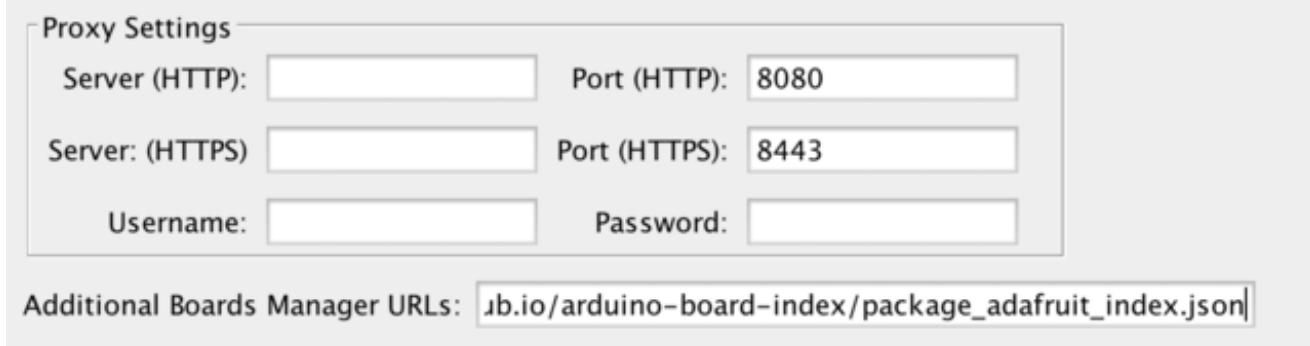
A dialog will pop up just like the one shown below.



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>). 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
```



Make sure you remove the `apt.adafruit.com` proxy setting from the Arduino preferences if you have previously added it.

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:

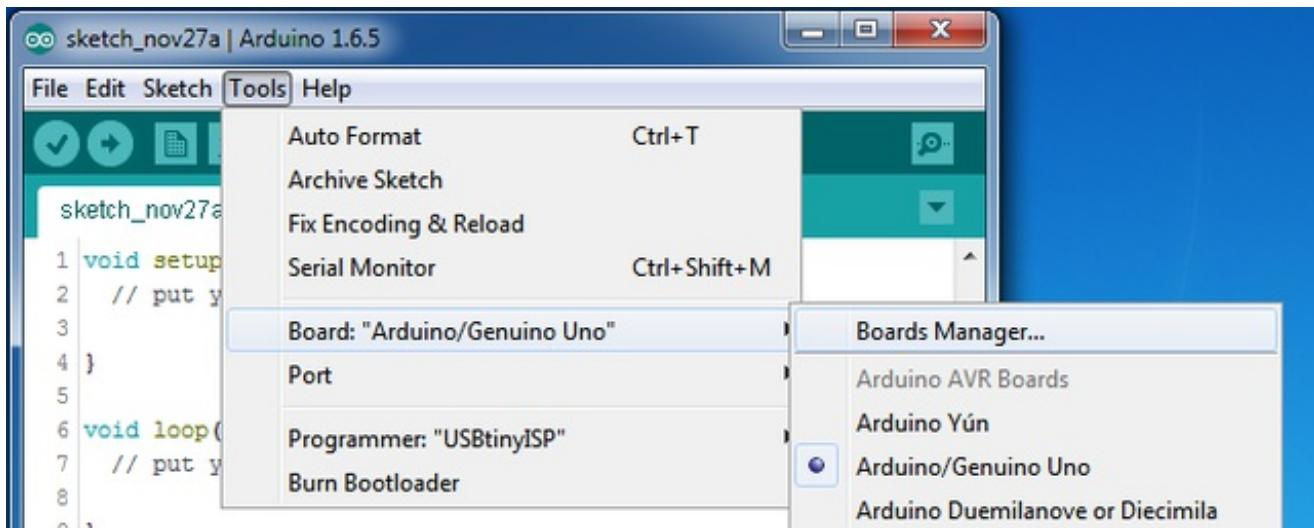
- **Adafruit AVR Boards** - Includes support for Flora, Gemma, Feather 32u4, Trinket, & Trinket Pro.
- **Adafruit SAMD Boards** - Includes support for Feather M0
- **Arduino Leonardo & Micro MIDI-USB** - This adds MIDI over USB support for the Flora, Feather 32u4, Micro and Leonardo using the [arcore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

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 ATSAMD21 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.

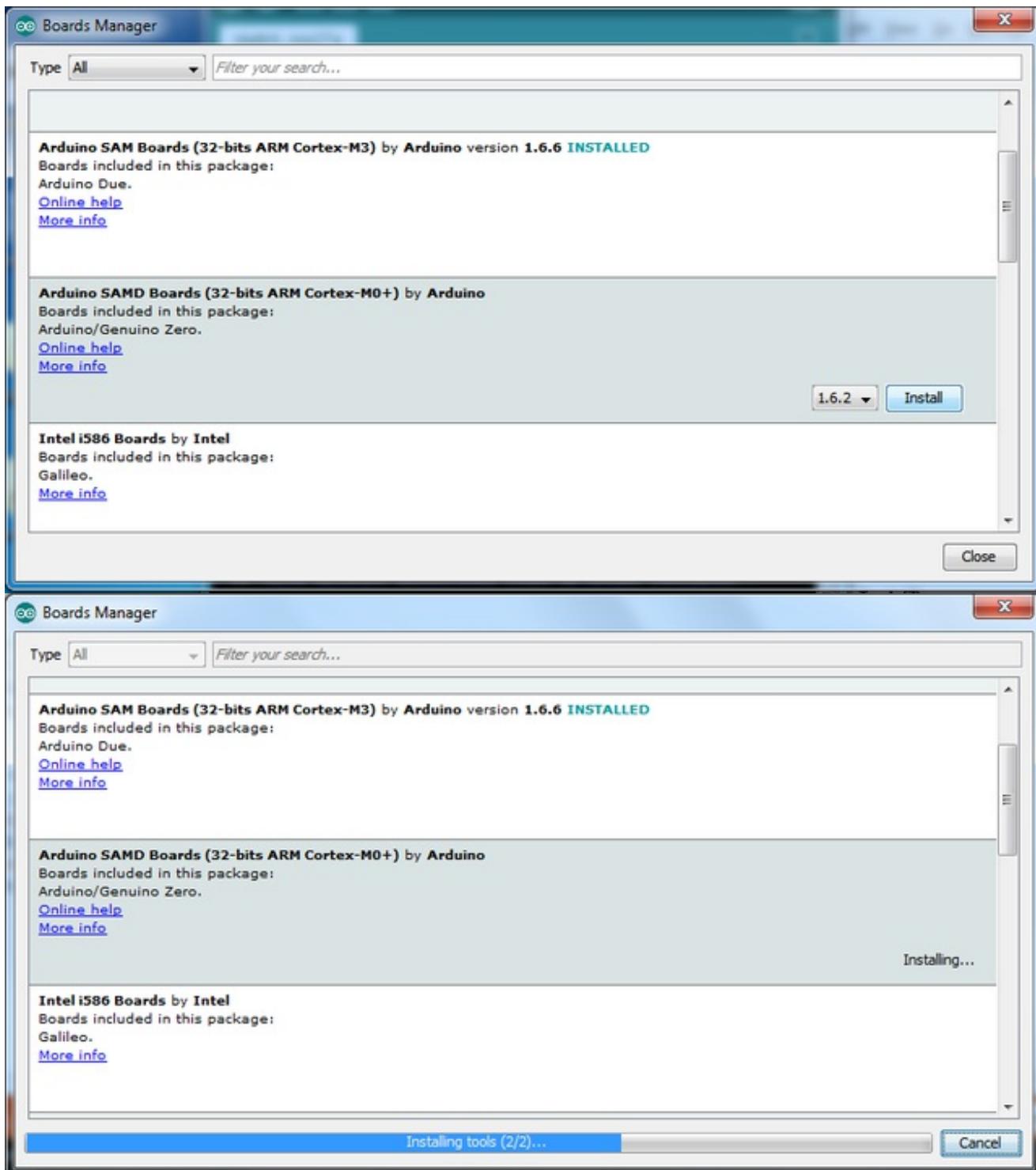
Now that you have added the appropriate URLs to the Arduino IDE preferences, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.



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.

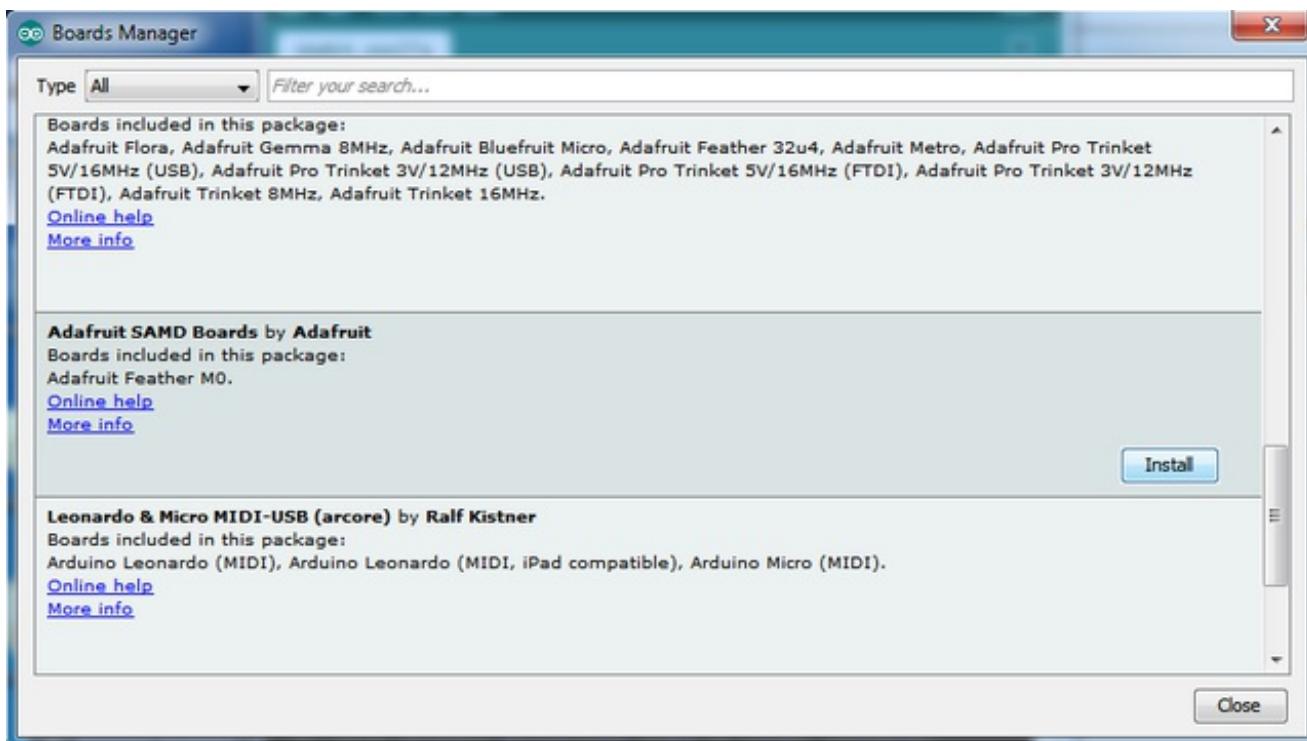
Install SAMD Support

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



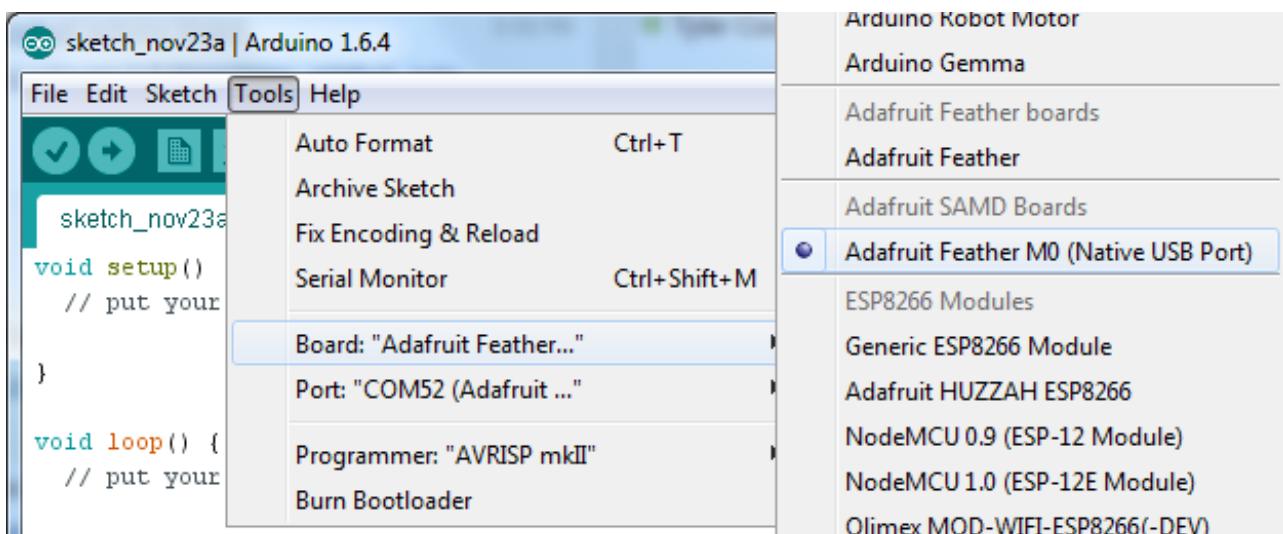
Install Adafruit SAMD

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)

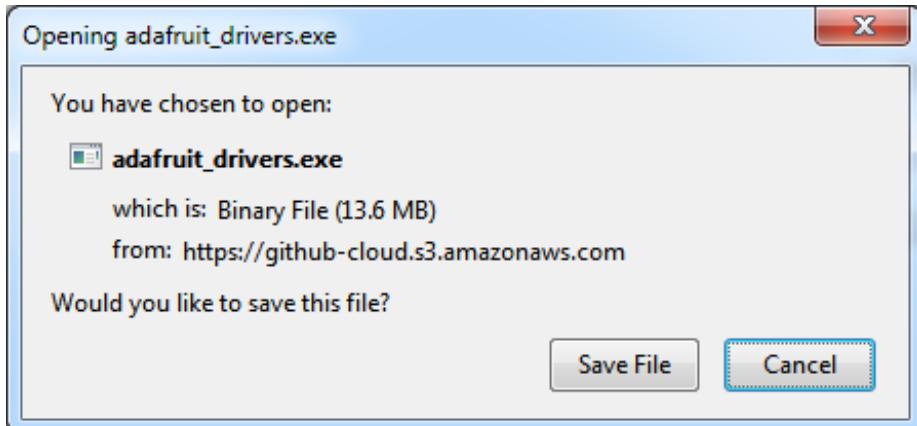
When you plug in the Feather, you'll need to possibly install a driver

Click below to download our Driver Installer

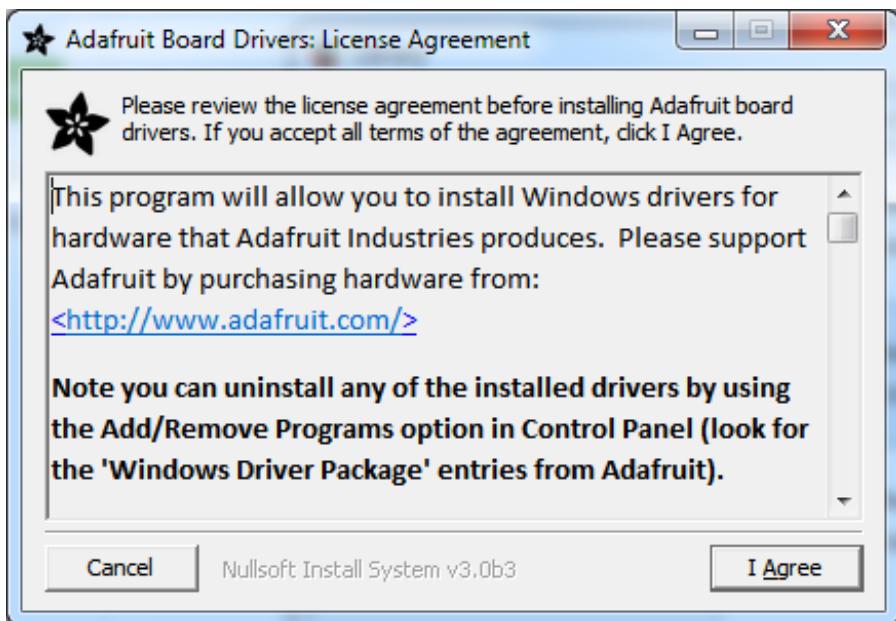
[Download Adafruit Driver Installer](#)

<http://adafru.it/mai>

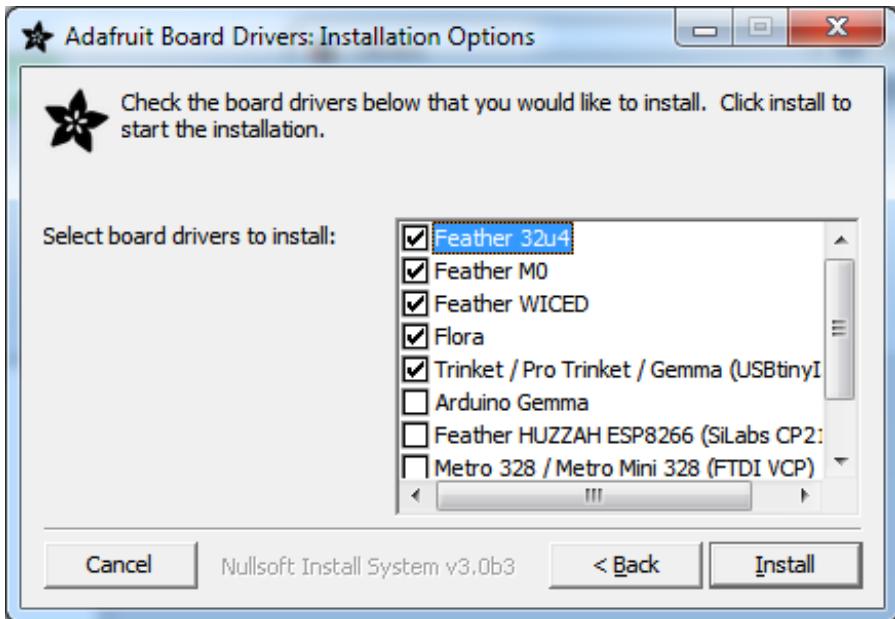
Download and run the installer



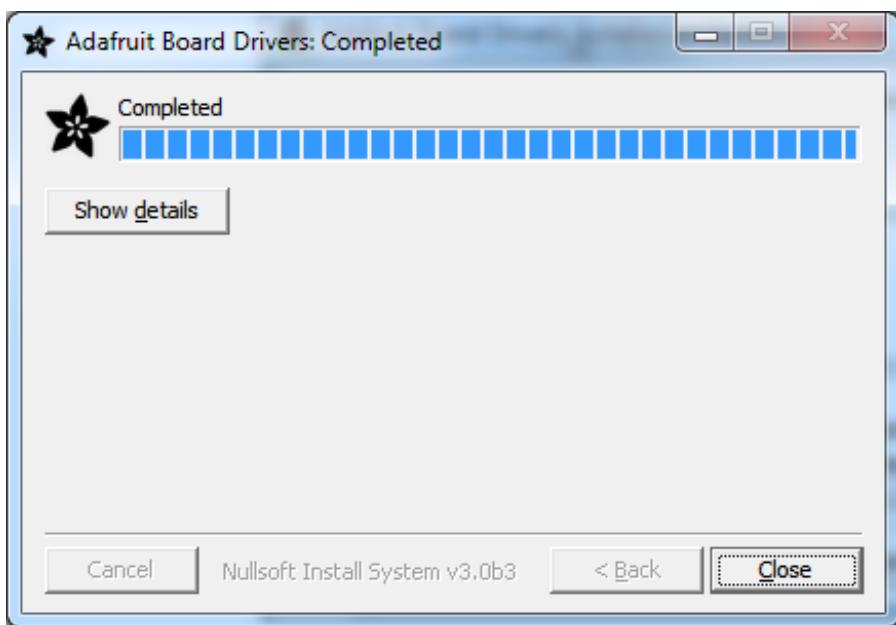
Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



Select which drivers you want to install:



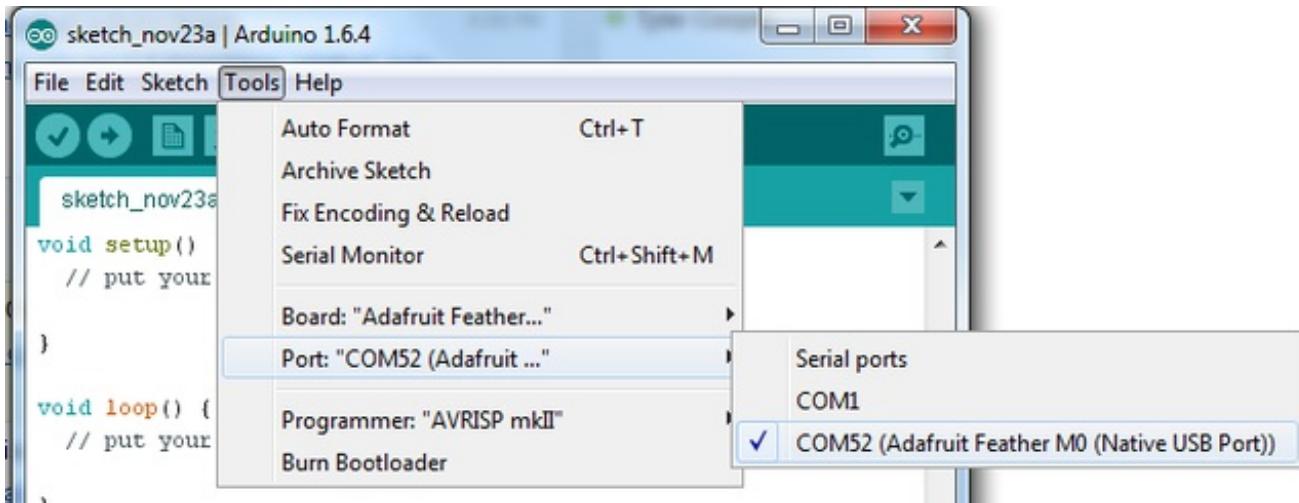
Click **Install** to do the installin'



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.

Sucessful 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

```
Done uploading.  
Write 11024 bytes to flash (173 pages)  
  
[=====] 36% (64/173 pages)  
[=====] 73% (128/173 pages)  
[=====] 100% (173/173 pages)  
  
done in 0.097 seconds  
  
  
Verify 11024 bytes of flash with checksum.  
  
Verify successful  
  
done in 0.049 seconds  
  
CPU reset.  
  
Adafruit Feather M0 (Native USB Port) on COM54
```

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

The screenshot shows the Arduino IDE's Serial Monitor window. The text in the window reads:

```
Cannot run program "{runtime.tools.arm-none-eabi-gcc.path}\bin\arm-none-eabi-g++" (in directory C:\Users\ladyada\Dropbox\ArduinoSketches\libraries\WINC1500\src)
{runtime.tools.arm-none-eabi-gcc.path}\bin\arm-none-eabi-g++ -mcpu=cortex-m0plus -mthumb -fno-max-inline-insns-single=500 -fno-rtti -fno-exceptions -MMD -DF_CPU=48000000L -DARDUINO -DUSB_MANUFACTURER="Adafruit" -DUSB_PRODUCT="Feather M0" -I{runtime.tools\CMSIS.path} -IC:\Users\ladyada\AppData\Roaming\Arduino15\packages\adafruit\hardware\samd\1.0.2\cores\arduino -IC:\Users\ladyada\AppData\Roaming\Arduino15\packages\adafruit\hardware\samd\1.0.2\variants\feather_m0 -IC:\Users\ladyada\AppData\Roaming\Arduino15\packages\adafruit\hardware\samd\1.0.2\libraries\WINC1500 C:\Users\ladyada\Dropbox\ArduinoSketches\libraries\WINC1500\src C:\Users\ladyada\AppData\Local\Temp\build5777565695508348365.tmp\mqtt_winc1500.cpp.o
Cannot run program "{runtime.tools.arm-none-eabi-gcc.path}\bin\arm-none-eabi-g++" (in directory C:\Users\ladyada\Dropbox\ArduinoSketches\libraries\WINC1500\src)
```

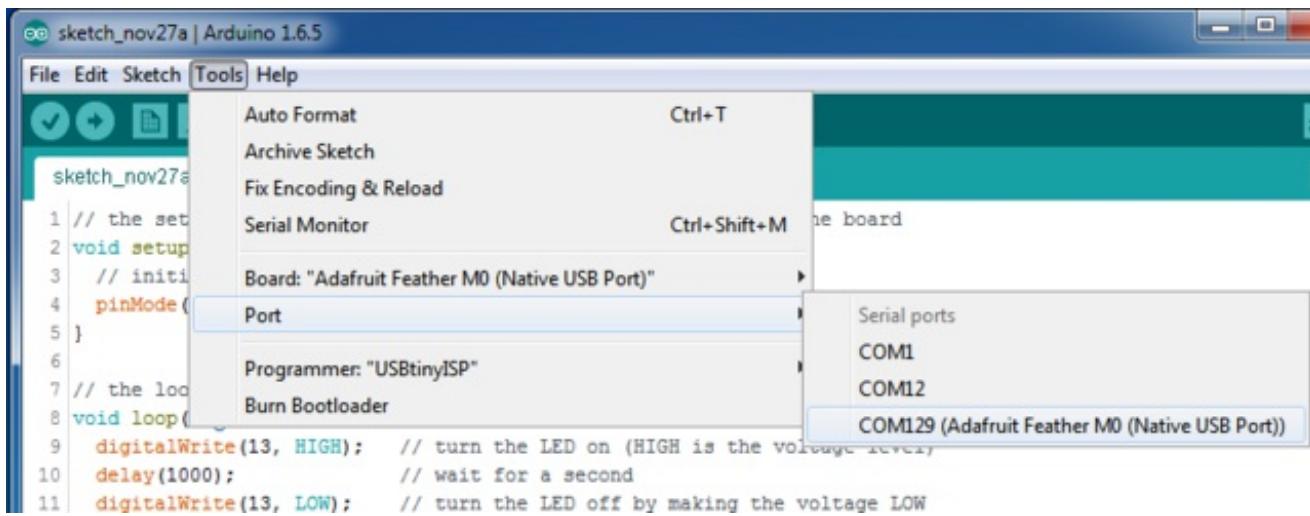
The status bar at the bottom of the window indicates: Adafruit Feather M0 (Native USB Port) on COM11.

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.



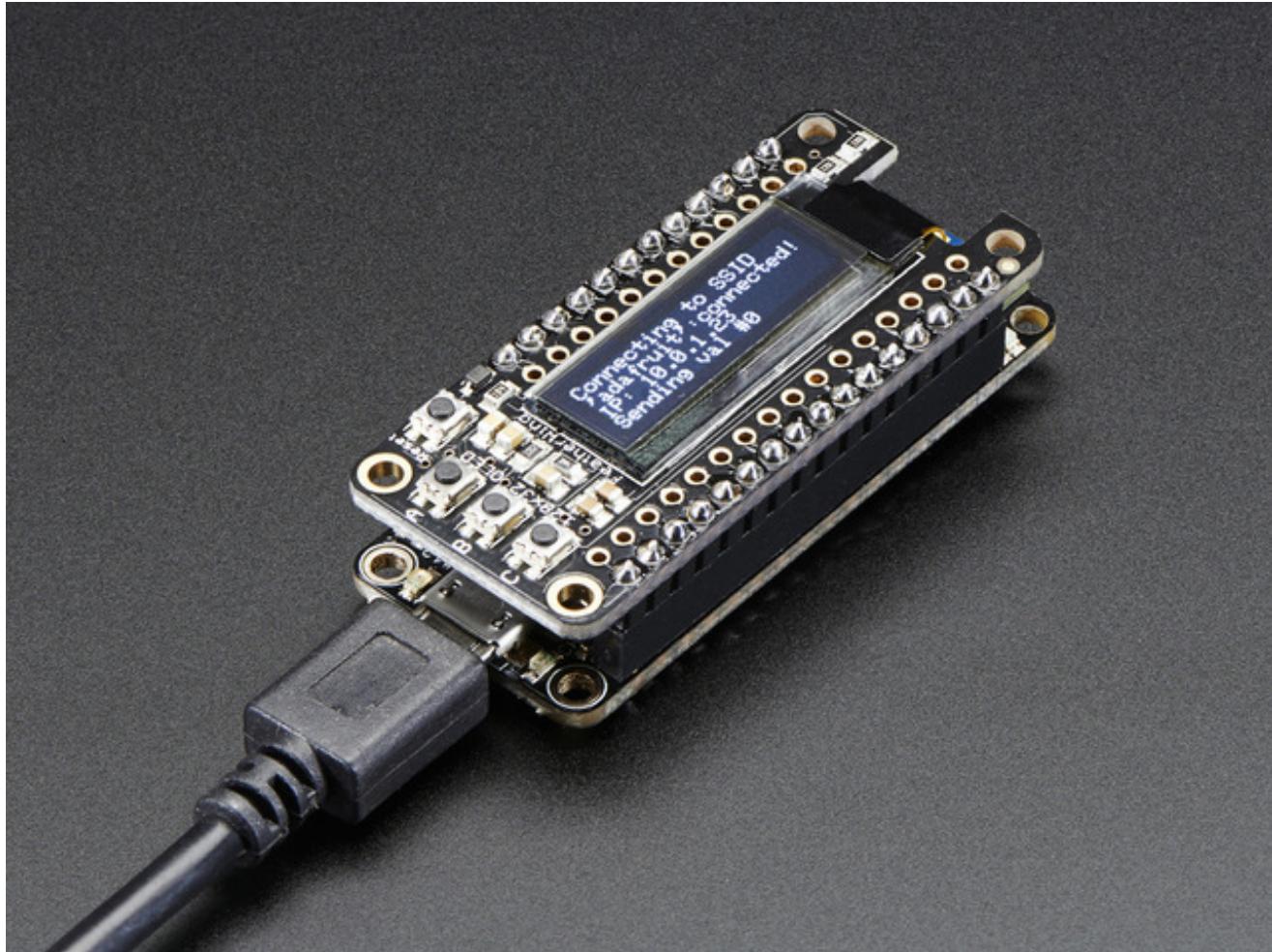
You may need to go back and reselect the 'normal' USB serial port next time you want to use the normal upload.

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/fP6) (<http://adafru.it/fP6>)

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

Download the Adafruit Library

We will start by downloading the Adafruit_ATWINC1500 Library, available from [our GitHub repository](#) (<http://adafru.it/kUE>).

This library is a light fork of the official Arduino Wifi101 library, (<http://adafru.it/kUF>) the only real changes are to allow changes to the default pin usage.

You can download the latest ZIP file by clicking the button below.

[Download Adafruit_WINC1500 Library](#)

<http://adafru.it/kVa>

Rename the uncompressed folder **Adafruit_WINC1500**. Check that the **Adafruit_WINC1500** folder contains a folder named **src** and **examples** and a file named **library.properties**

Place the **Adafruit_WINC1500** library folder your **sketchbookfolder/libraries/** folder. You may need to create the libraries subfolder if its your first library. Restart the IDE. You can figure out your **sketchbookfolder** by opening up the Preferences tab in the Arduino IDE.

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

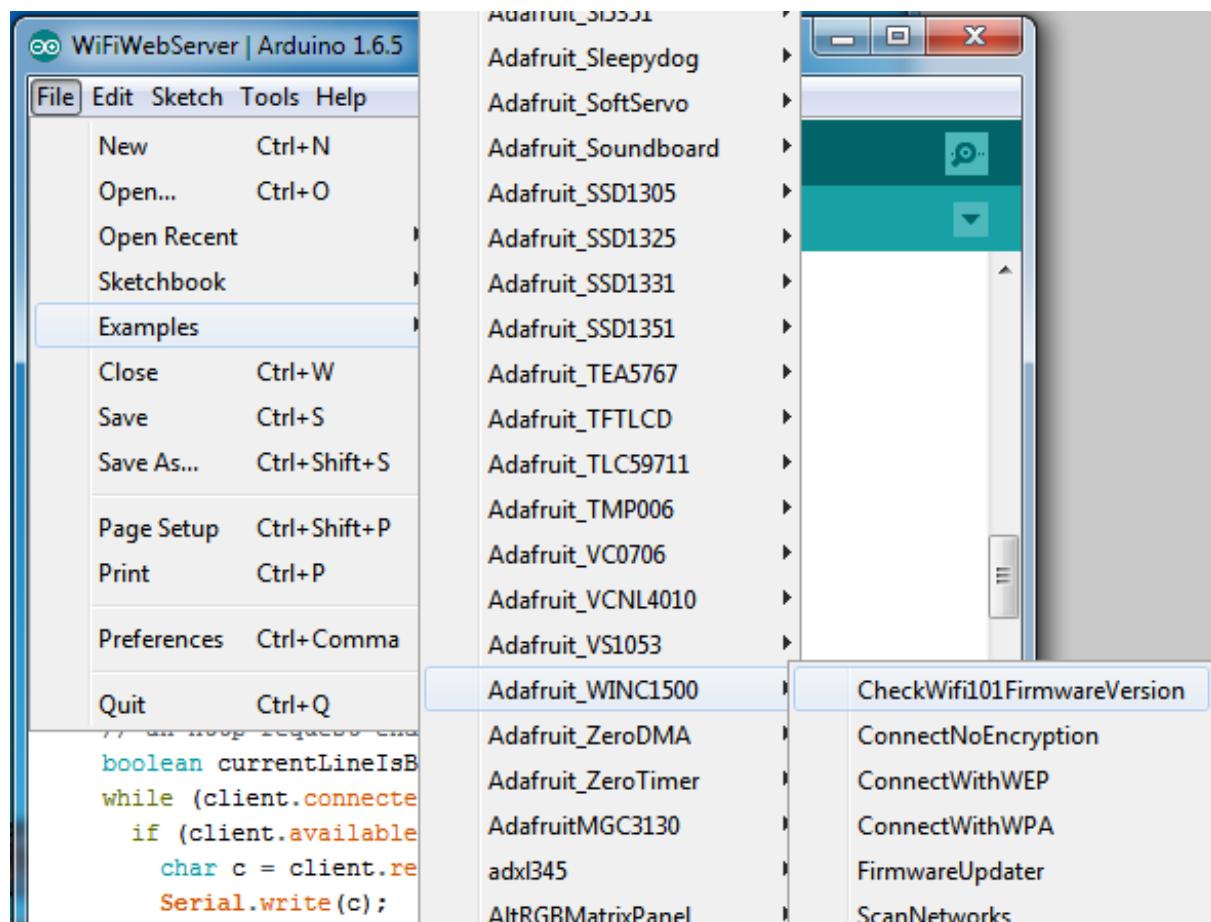
Restart the Arduino IDE.

You may need to use Arduino 1.6.5 or later

Check Connections & Version

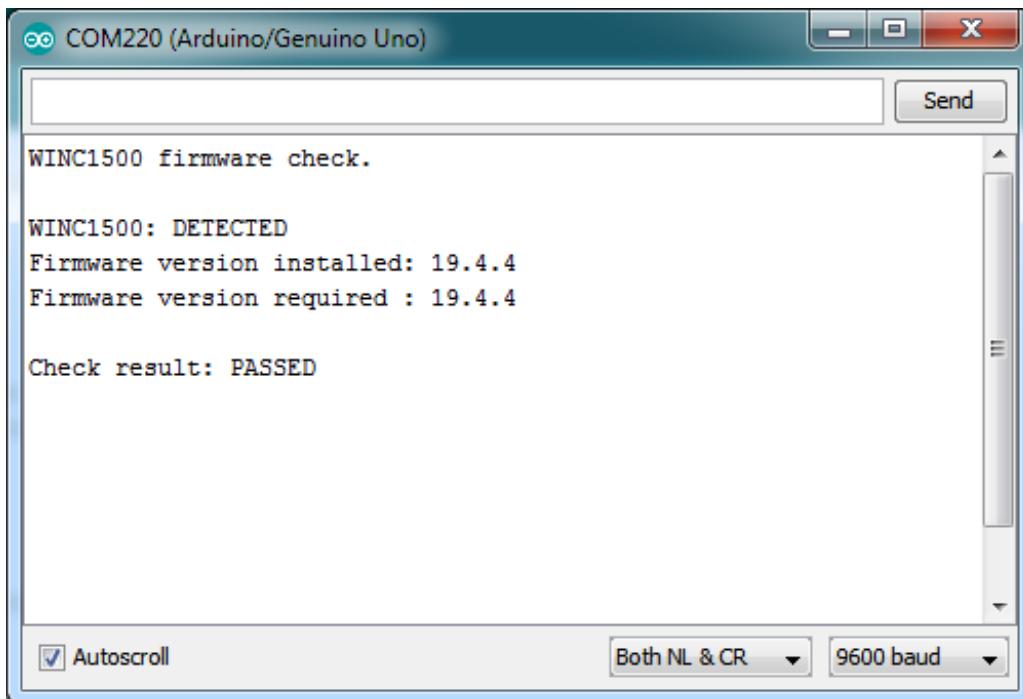
Before we start, its important to verify you have the right setup & firmware version.

Load up the **Adafruit_WINC1500->CheckWifi101Firmware** sketch



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

You should see that the firmware is 19.4.4



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

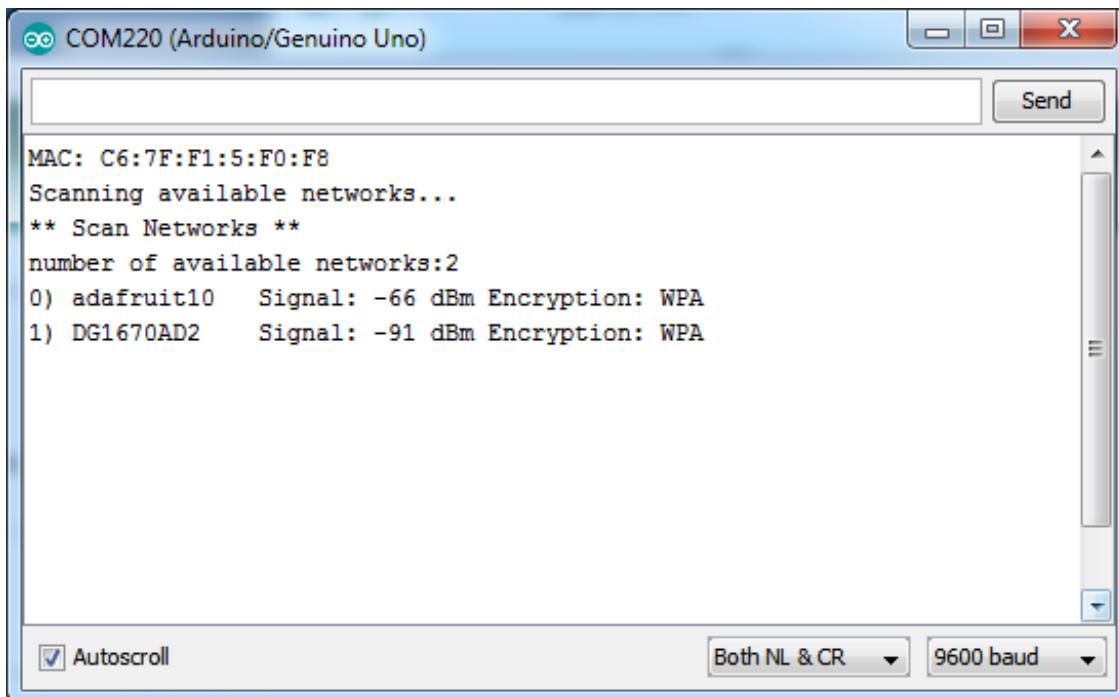
If you have version 19.3 or less, the firmware is too old

If you get no response, the firmware is either way too old, or something is amiss with your wiring!

Scanning WiFi

Now that you have the right firmware version, let's scan for networks!

Run the **Adafruit_WINC1500->ScanNetworks** example to see a list of available visible networks



COM220 (Arduino/Genuino Uno)

```
MAC: C6:7F:F1:5:F0:F8
Scanning available networks...
** Scan Networks **
number of available networks:2
0) adafruit10    Signal: -66 dBm Encryption: WPA
1) DG1670AD2    Signal: -91 dBm Encryption: WPA
```

Autoscroll Both NL & CR 9600 baud

Connect & Read Webpage

OK finally you get to connect and read some data!

Open up the **Adafruit_WINC1500->WiFi101WebClient** example, then 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:  
.....
```

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

```
WINC1500 Web client test
Attempting to connect to SSID: adafruit10
Connected to wifi
SSID: adafruit10
IP Address: 10.0.1.150
signal strength (RSSI):-78 dBm

Starting connection to server...
connected to server
HTTP/1.1 200 OK
Date: Wed, 20 Jan 2016 23:10:53 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: close
Set-Cookie: __cfduid=d9b99643e6b6eb86e76f6f2d5a016f24d1453331452; expires=Thu, 19-Jan-17 23:10:52 GMT; pa
Accept-Ranges: bytes
Access-Control-Allow-Credentials: true
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept, Accept-Encoding, Authorizat
Access-Control-Allow-Methods: GET, POST, OPTIONS
Access-Control-Max-Age: 1728000
Cf-Railgun: direct (starting new WAN connection)
Last-Modified: Thu, 27 Jun 2013 14:13:27 GMT
Vary: Accept-Encoding
Server: cloudflare-nginx
CF-RAY: 267e748cfcf90ef7-EWR

4a
This is a test of the CC3000 module!
If you can read this, its working :)

0

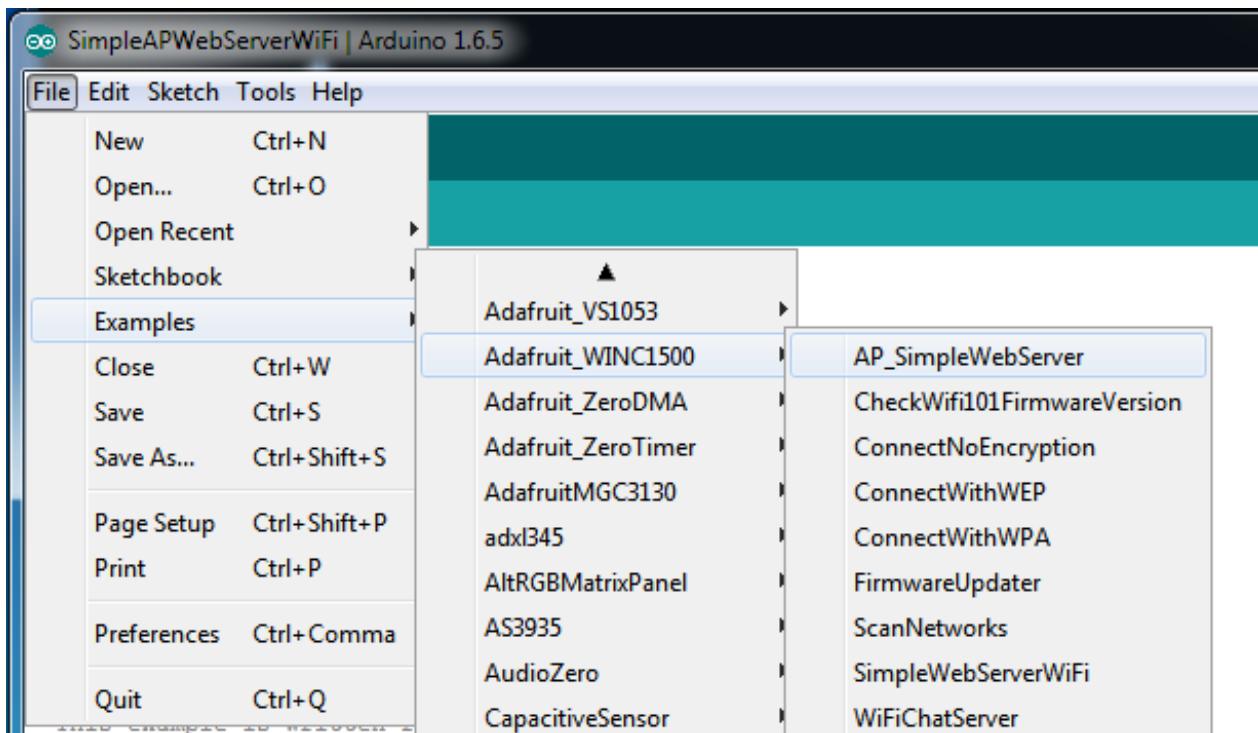
disconnecting from server.
```

Autoscroll Both NL & CR 9600 baud

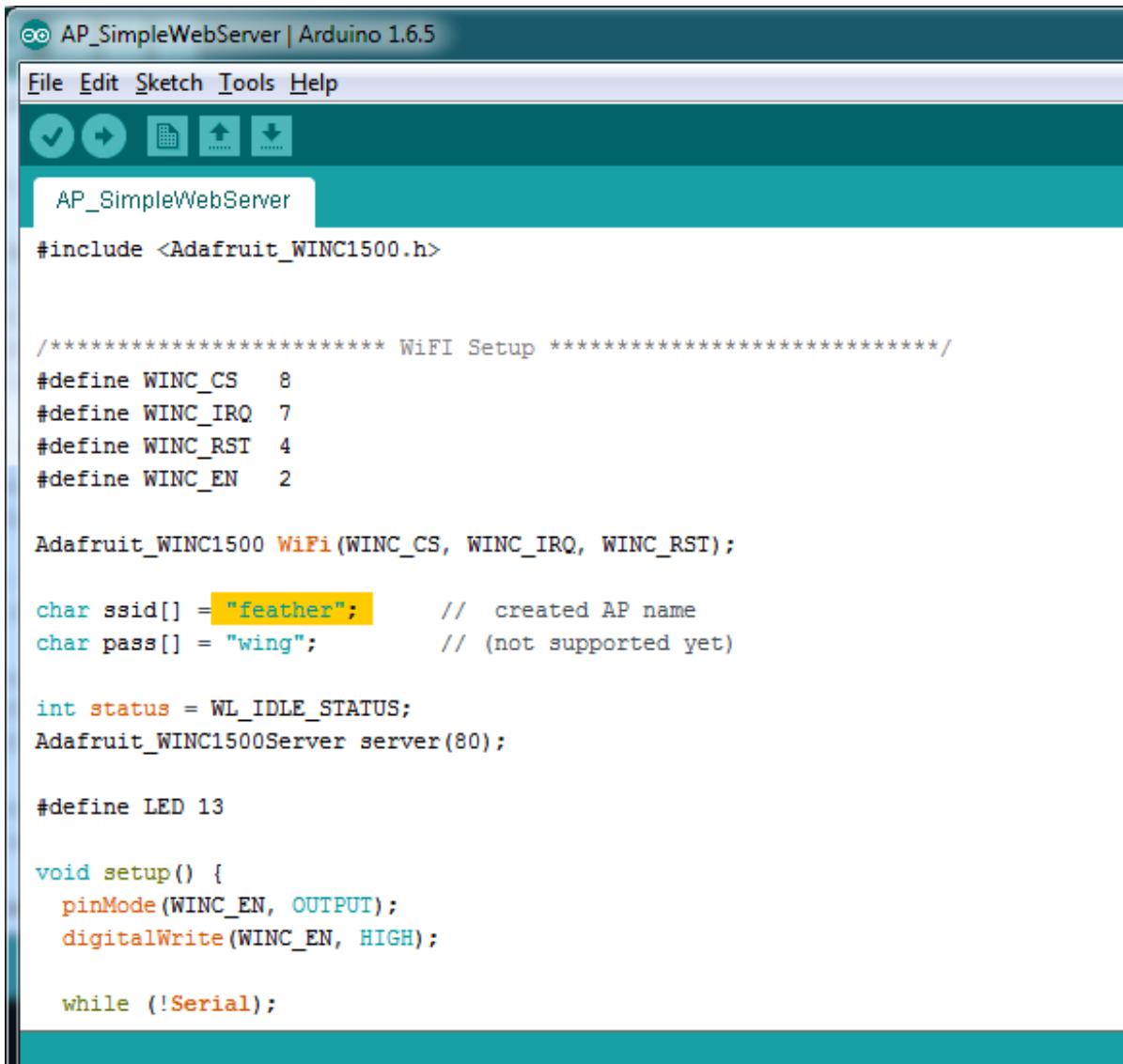
Creating an Access Point + Webserver

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 **Adafruit_WINC1500->AP_SimpleWebServer** example



You can change the SSID (at this time the password isn't used) & LED (13 is the onboard feather LED)



The screenshot shows the Arduino IDE interface with the title bar "AP_SimpleWebServer | Arduino 1.6.5". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu is a toolbar with icons for save, upload, and download. The main code area is titled "AP_SimpleWebServer". The code itself is as follows:

```
#include <Adafruit_WINC1500.h>

/***************** WiFi Setup *****/
#define WINC_CS    8
#define WINC_IRQ   7
#define WINC_RST   4
#define WINC_EN    2

Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);

char ssid[] = "feather";          // created AP name
char pass[] = "wing";           // (not supported yet)

int status = WL_IDLE_STATUS;
Adafruit_WINC1500Server server(80);

#define LED 13

void setup() {
  pinMode(WINC_EN, OUTPUT);
  digitalWrite(WINC_EN, HIGH);

  while (!Serial);
```

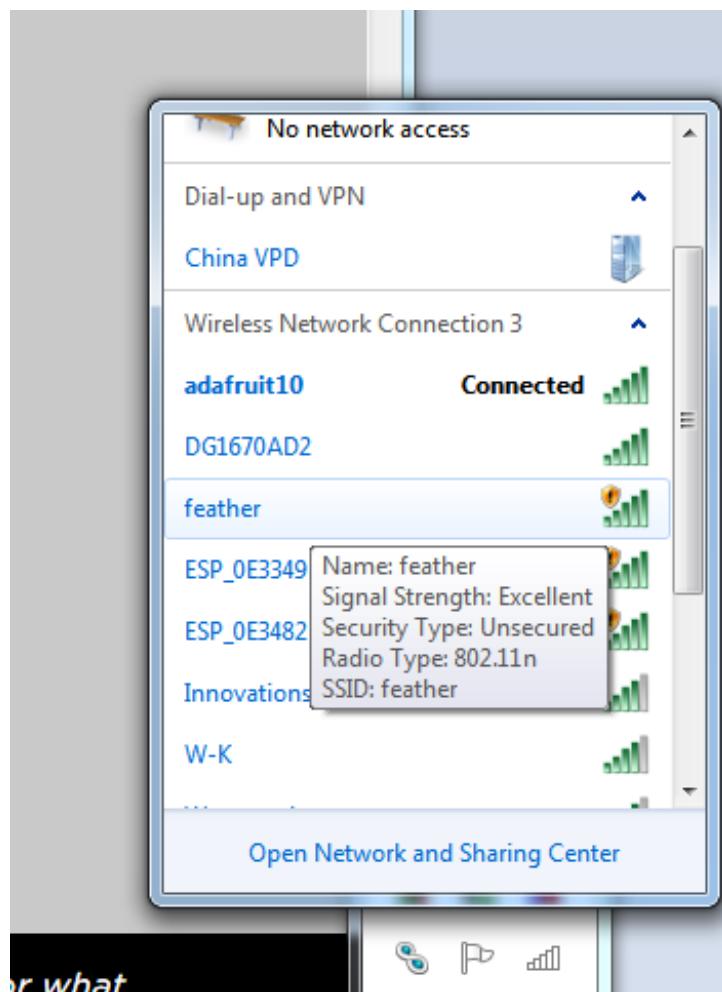
Upload and open up the serial console to start the AP

COM3 (Adafruit Feather M0 (Native USB Port))

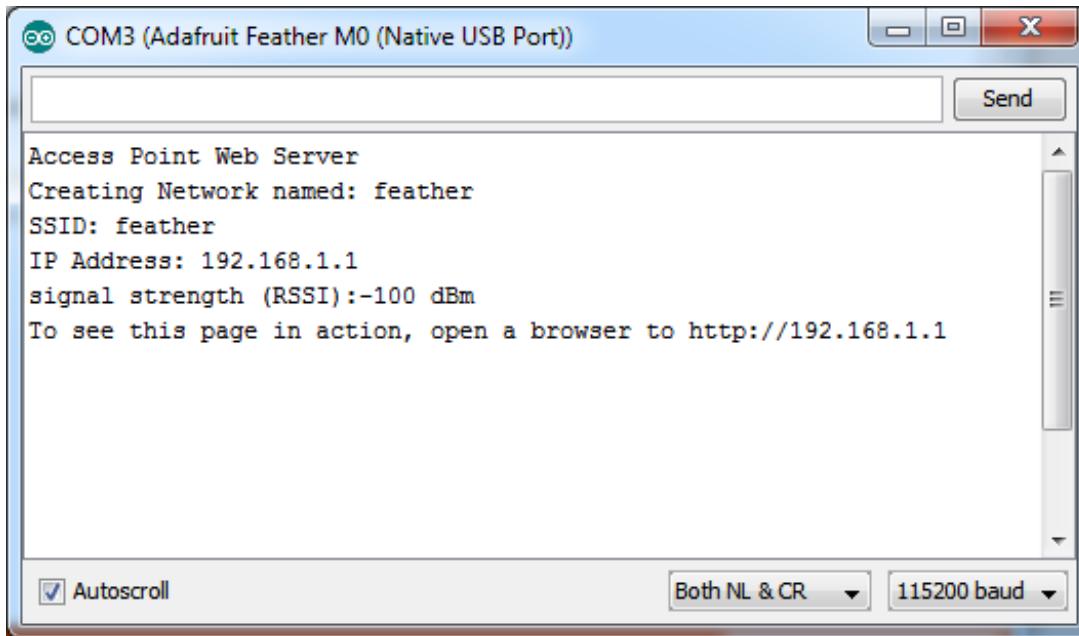
```
Access Point Web Server
Creating Network named: feather
```

Autoscroll Both NL & CR 115200 baud

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

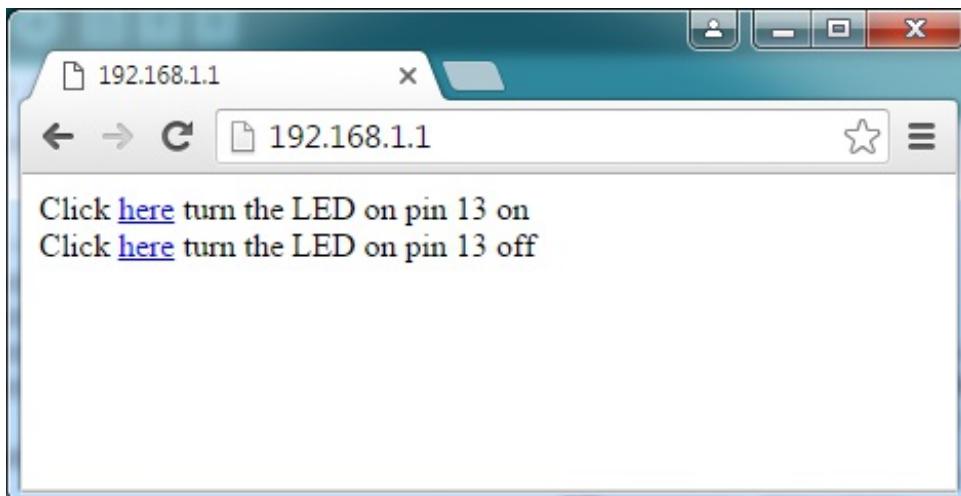


COM3 (Adafruit Feather M0 (Native USB Port))

```
Access Point Web Server
Creating Network named: feather
SSID: feather
IP Address: 192.168.1.1
signal strength (RSSI):-100 dBm
To see this page in action, open a browser to http://192.168.1.1
```

Autoscroll Both NL & CR 115200 baud

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

```
COM3 (Adafruit Feather M0 (Native USB Port))  
Send  
  
Access Point Web Server  
Creating Network named: feather  
SSID: feather  
IP Address: 192.168.1.1  
signal strength (RSSI):-100 dBm  
To see this page in action, open a browser to http://192.168.1.1  
new client  
client disconnected  
new client  
GET / HTTP/1.1  
Host: 192.168.1.1  
Connection: keep-alive  
Cache-Control: max-age=0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36  
Accept-Encoding: gzip, deflate, sdch  
Accept-Language: en-US,en;q=0.8  
  
client disconnected  
new client  
GET /favicon.ico HTTP/1.1  
Host: 192.168.1.1  
Connection: keep-alive  
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.111 Safari/537.36  
Accept: */*  
Referer: http://192.168.1.1/  
Accept-Encoding: gzip, deflate, sdch  
Accept-Language: en-US,en;q=0.8  
  
 Autoscroll Both NL & CR 115200 baud
```

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

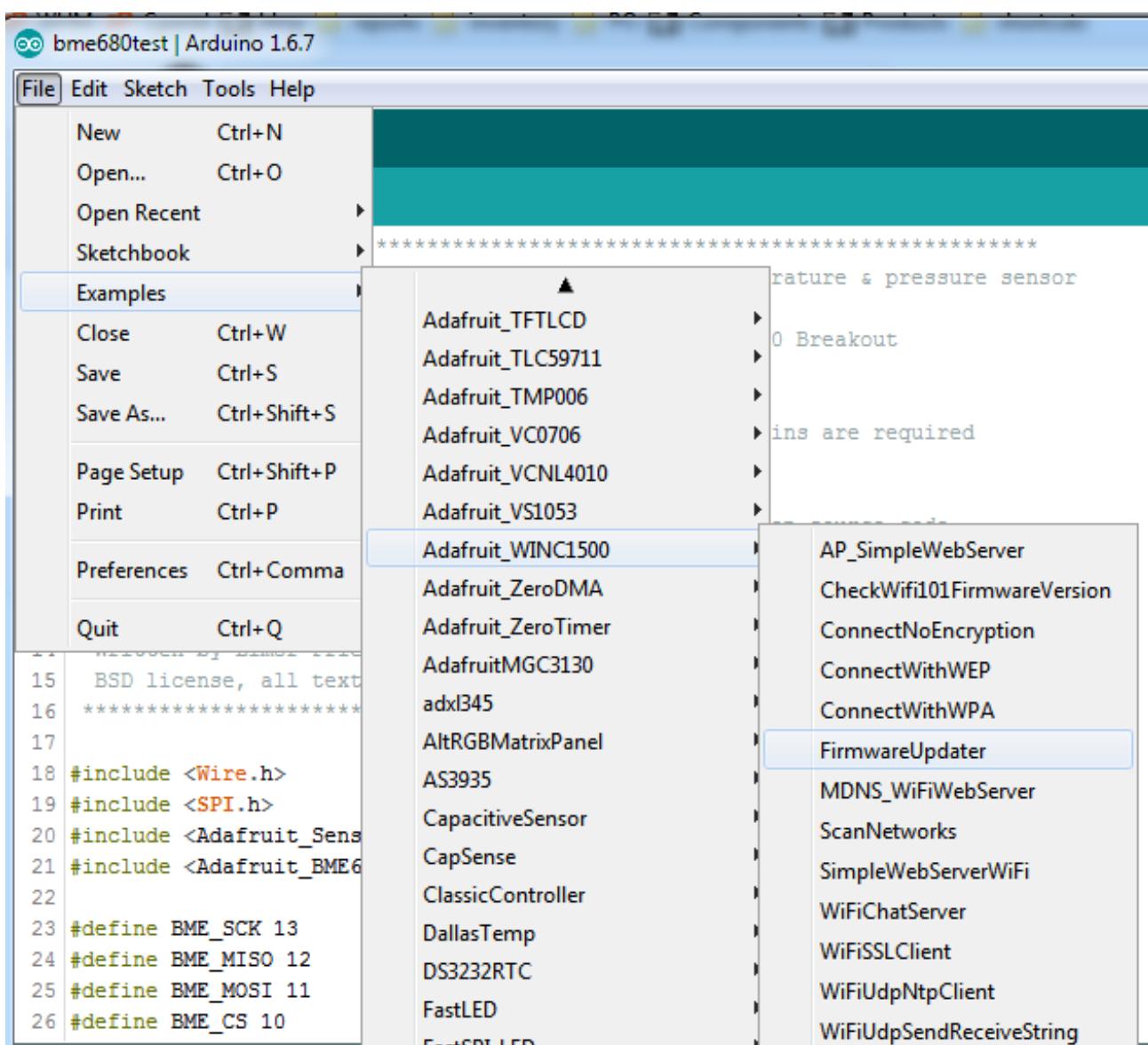
Updating SSL Certificates

Do not use the updater to update the WINC1500 firmware, you could brick it. Only use it for updating SSL certs

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 uploaders but it only has to happen once

Start out by uploading the **FirmwareUpdater** sketch from **Adafruit_WINC1500**



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

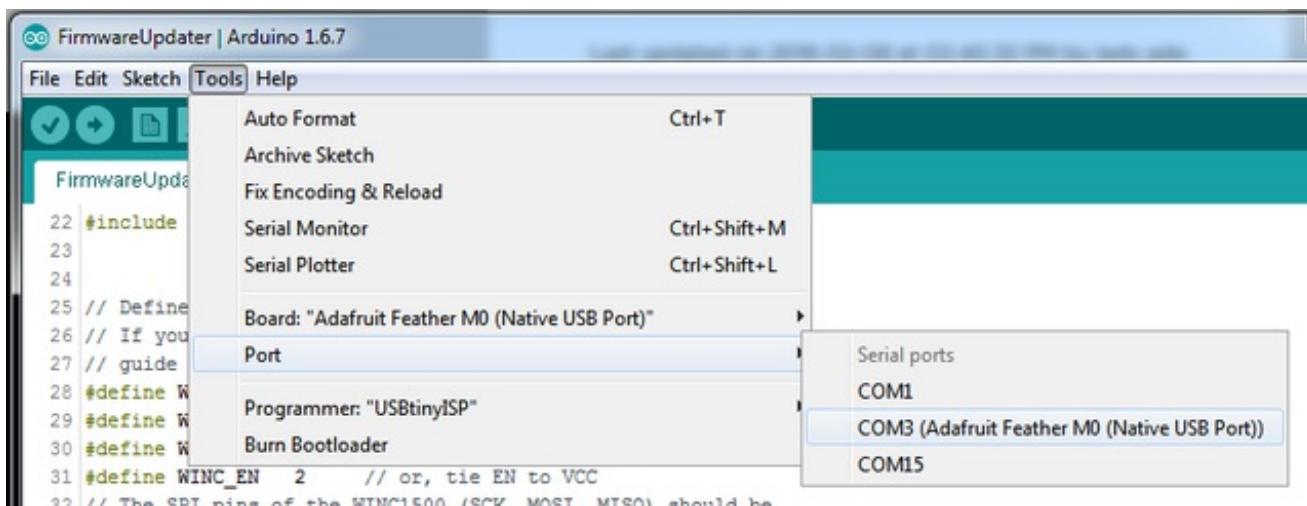
The screenshot shows the Arduino IDE interface with the title bar "FirmwareUpdater | Arduino 1.6.7". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar has icons for save, upload, and refresh. The main area displays the "FirmwareUpdater" tab selected, containing the following code:

```
22 #include <spi_flash/include/spi_flash.h>
23
24
25 // Define the WINC1500 board connections below.
26 // If you're following the Adafruit WINC1500 board
27 // guide you don't need to modify these:
28 #define WINC_CS 8
29 #define WINC_IRQ 7
30 #define WINC_RST 4
31 #define WINC_EN 2 // or, tie EN to VCC
32 // The SPI pins of the WINC1500 (SCK, MOSI, MISO) should be
33 // connected to the hardware SPI port of the Arduino.
34 // On an Uno or compatible these are SCK = #13, MISO = #12, MOSI = #11.
35 // On an Arduino Zero use the 6-pin ICSP header, see:
36 // https://www.arduino.cc/en/Reference/SPI
37
38 // Setup the WINC1500 connection with the pins above and the default hardware SPI.
39 Adafruit_WINC1500 WiFi(WINC_CS, WINC_IRQ, WINC_RST);
40
```

The status bar at the bottom shows "25 - 31" on the left and "Arduino/Genuine Uno on COM11" on the right.

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



Now download or clone the [WiFi101 Firmware Updater repository](http://adafru.it/leT) (<http://adafru.it/leT>) from github, you can just click here to grab the latest Zip

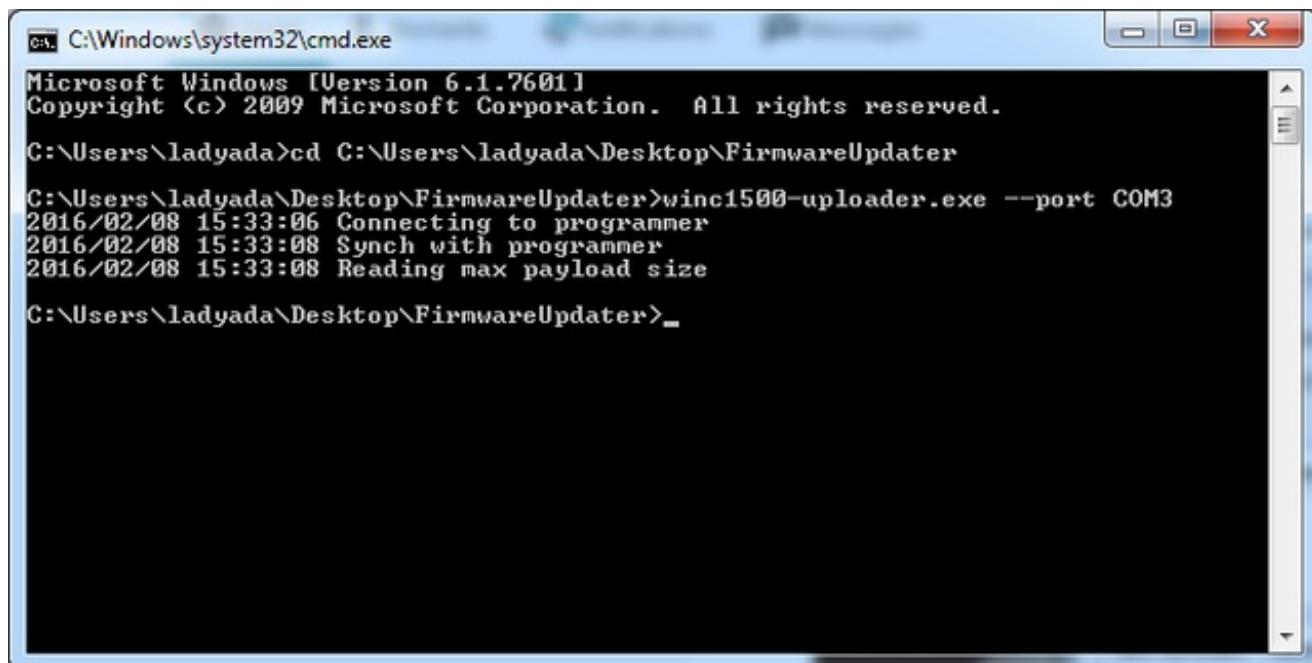
Command Line Usage

Windows

Uncompress it on your desktop. Now use powershell, command or terminal to **cd** to the uncompressed directory and run

winc1500-uploader --port *serialport*

for example, on windows, **winc1500-uploader --port COM3**

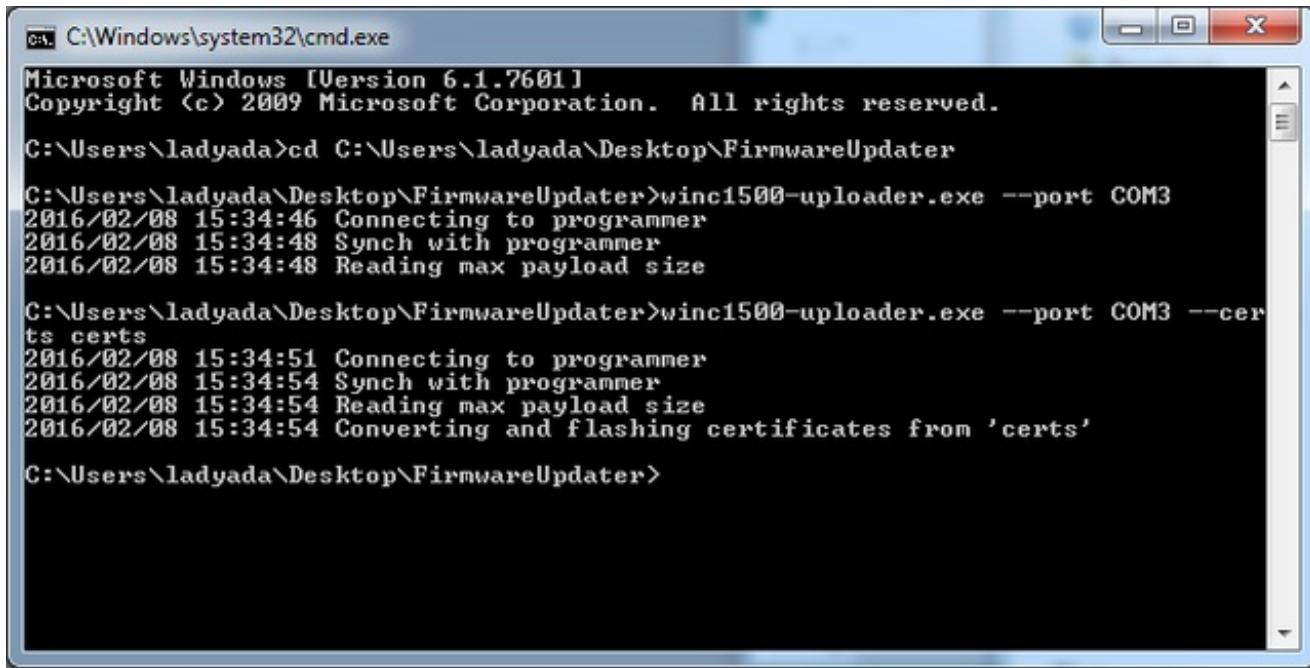


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ladyada>cd C:\Users\ladyada\Desktop\FirmwareUpdater
C:\Users\ladyada\Desktop\FirmwareUpdater>winc1500-uploader.exe --port COM3
2016/02/08 15:33:06 Connecting to programmer
2016/02/08 15:33:08 Synch with programmer
2016/02/08 15:33:08 Reading max payload size

C:\Users\ladyada\Desktop\FirmwareUpdater>_
```

You should see that it was able to read the max payload size. Next up just run the same command but add **--certs certs** to upload all the certificates in the certs directory



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\ladyada>cd C:\Users\ladyada\Desktop\FirmwareUpdater

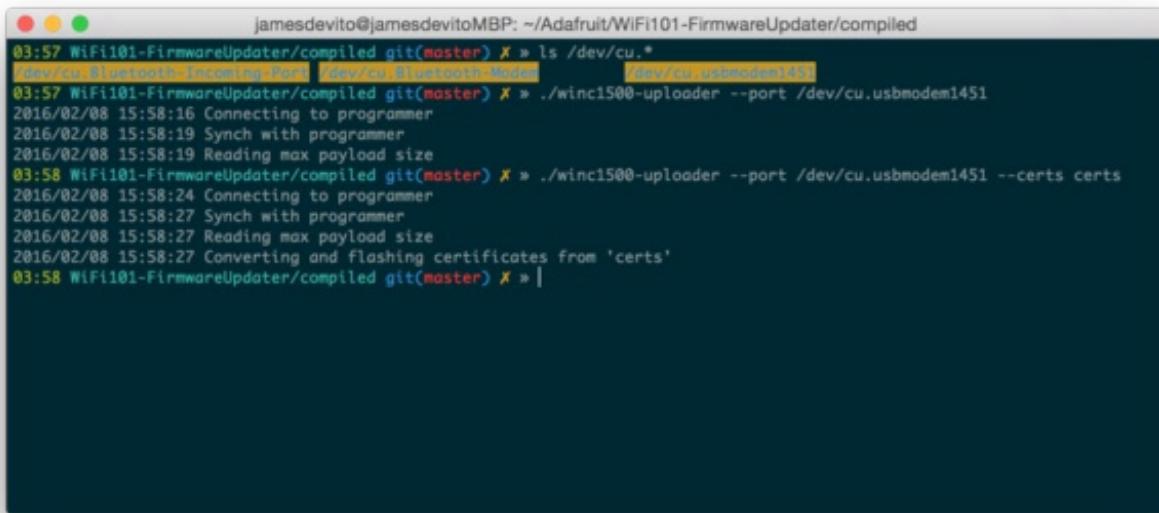
C:\Users\ladyada\Desktop\FirmwareUpdater>winc1500-uploader.exe --port COM3
2016/02/08 15:34:46 Connecting to programmer
2016/02/08 15:34:48 Synch with programmer
2016/02/08 15:34:48 Reading max payload size

C:\Users\ladyada\Desktop\FirmwareUpdater>winc1500-uploader.exe --port COM3 --certs certs
2016/02/08 15:34:51 Connecting to programmer
2016/02/08 15:34:54 Synch with programmer
2016/02/08 15:34:54 Reading max payload size
2016/02/08 15:34:54 Converting and flashing certificates from 'certs'

C:\Users\ladyada\Desktop\FirmwareUpdater>
```

Mac OS X (Command Line) Usage

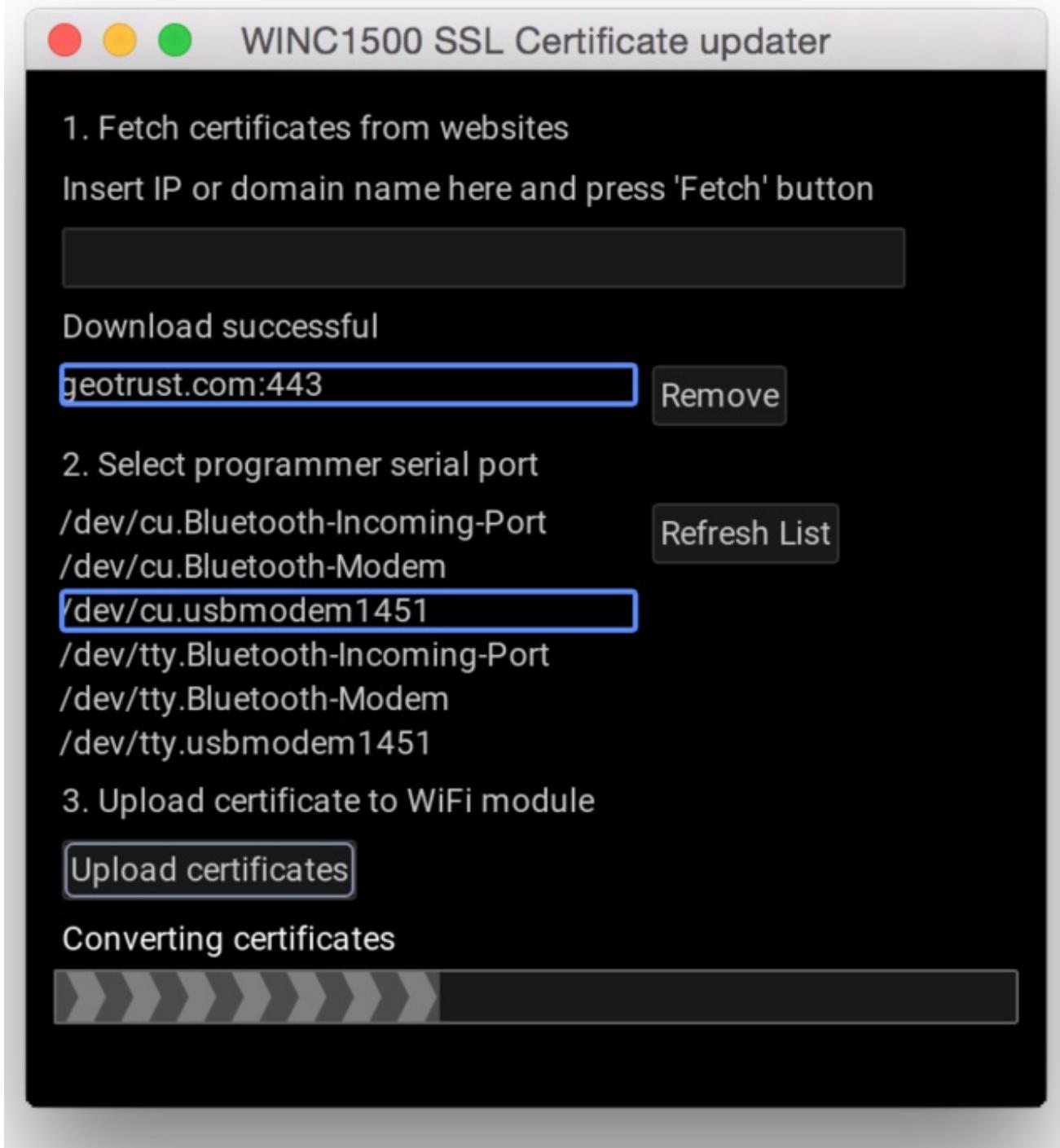
With Mac, you can use command line, its essentially the same as above except the serial port will be something like **/dev/cu.usbserialnnnn** You can figure it out by running **ls /dev/cu.*** to list all serial devices, ignore the Bluetooth modem interfaces

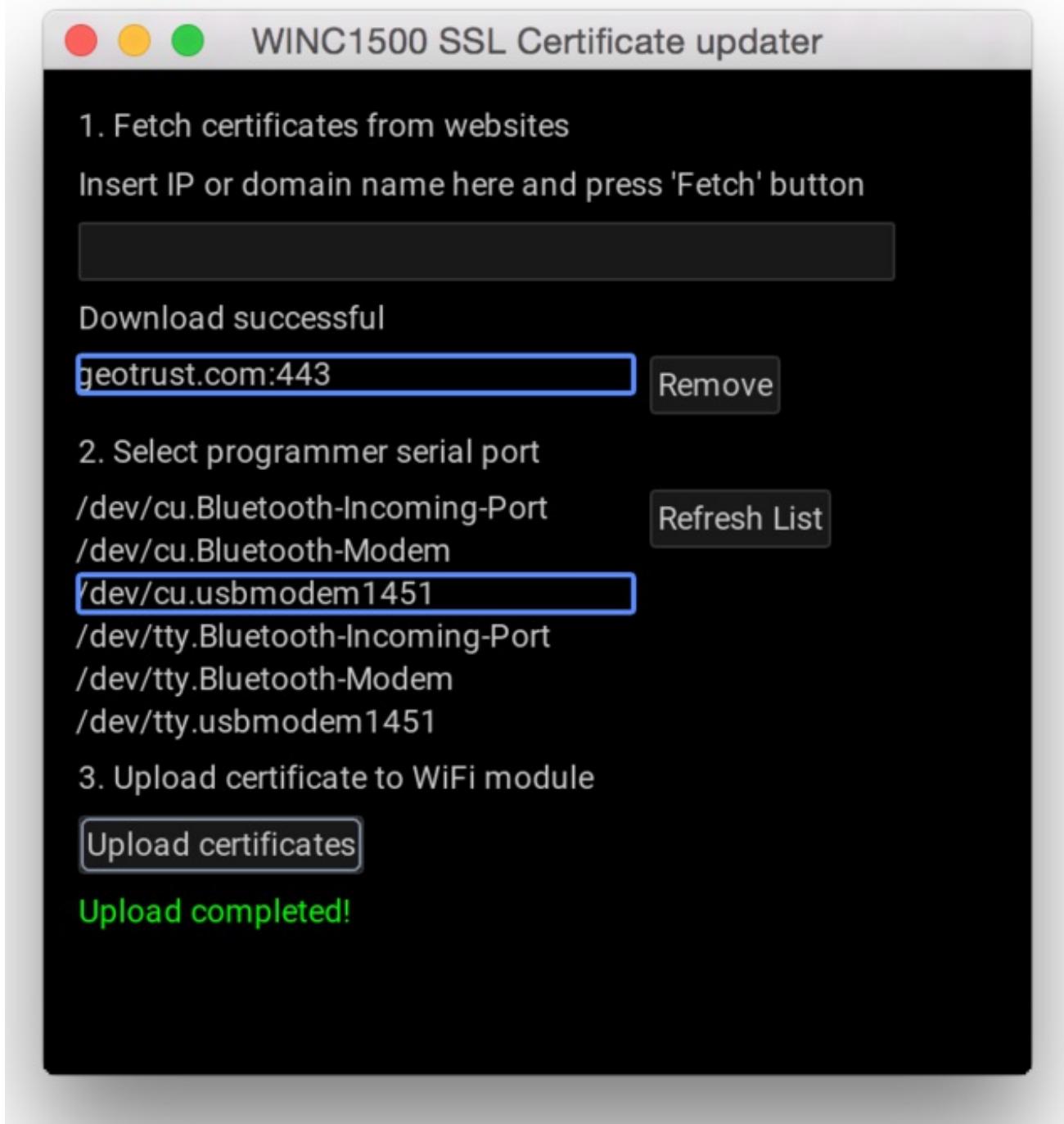


```
jamesdevito@jamesdevitoMBP: ~/Adafruit/WiFi101-FirmwareUpdater/compiled
03:57 WiFi101-FirmwareUpdater/compiled git[master] ✘ > ls /dev/cu.*
/dev/cu_Bluetooth-Modem /dev/cu_Bluetooth-Modem (/dev/cu_usbmodem1451)
03:57 WiFi101-FirmwareUpdater/compiled git[master] ✘ > ./winc1500-uploader --port /dev/cu.usbmodem1451
2016/02/08 15:58:16 Connecting to programmer
2016/02/08 15:58:19 Synch with programmer
2016/02/08 15:58:19 Reading max payload size
03:58 WiFi101-FirmwareUpdater/compiled git[master] ✘ > ./winc1500-uploader --port /dev/cu.usbmodem1451 --certs certs
2016/02/08 15:58:24 Connecting to programmer
2016/02/08 15:58:27 Synch with programmer
2016/02/08 15:58:27 Reading max payload size
2016/02/08 15:58:27 Converting and flashing certificates from 'certs'
03:58 WiFi101-FirmwareUpdater/compiled git[master] ✘ > |
```

GUI Usage

You can also use the GUI which is nice and will also let you fetch the certificate and upload it directly. If you don't need any particular site's cert just put in www.google.com



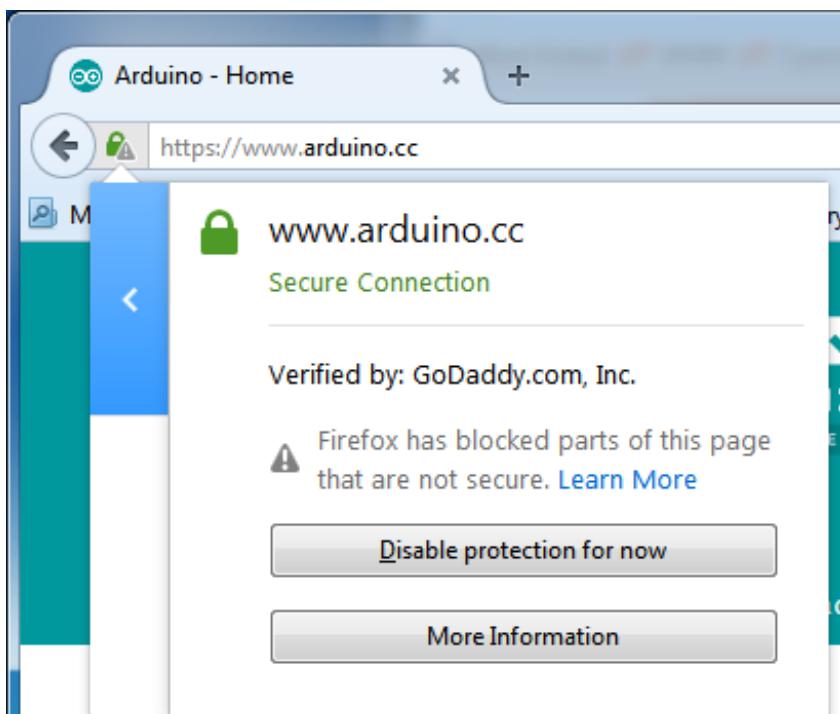


Manually Adding Certificates

You can upload other certificates, make sure they are in DER format (<http://adafru.it/leV>) (binary, not ascii!) and end the name with .cer

Name	Date modified	Type
AddTrustExternalCARoot.cer	2/8/2016 3:23 PM	Security Certificate
BaltimoreCyberTrustRoot.cer	2/8/2016 3:23 PM	Security Certificate
Digicert_Root.cer	2/8/2016 3:23 PM	Security Certificate
FreeRadius_Root.cer	2/8/2016 3:23 PM	Security Certificate
GoDaddyRootCertificateAuthority-G2.cer	2/8/2016 3:23 PM	Security Certificate
NMA_Root.cer	2/8/2016 3:23 PM	Security Certificate
PROWL_Root.cer	2/8/2016 3:23 PM	Security Certificate
Radius_Root.cer	2/8/2016 3:23 PM	Security Certificate
VeriSignClass3PublicPrimaryCertification...	2/8/2016 3:23 PM	Security Certificate

To figure out what certificate you need, go to the page you're trying to connect to, using your browser. Then click on the lock (it may be in a different location) to make sure you're using **https** and its secured. Then click **More Information**



When you get the details popup, click on **View Certificate**

Page Info - https://anprice-iothub.azure-devices.net/

General Permissions Security

Website Identity

Website: **anprice-iothub.azure-devices.net**
Owner: **This website does not supply ownership information.**
Verified by: **Microsoft Corporation**

[View Certificate](#)

Privacy & History

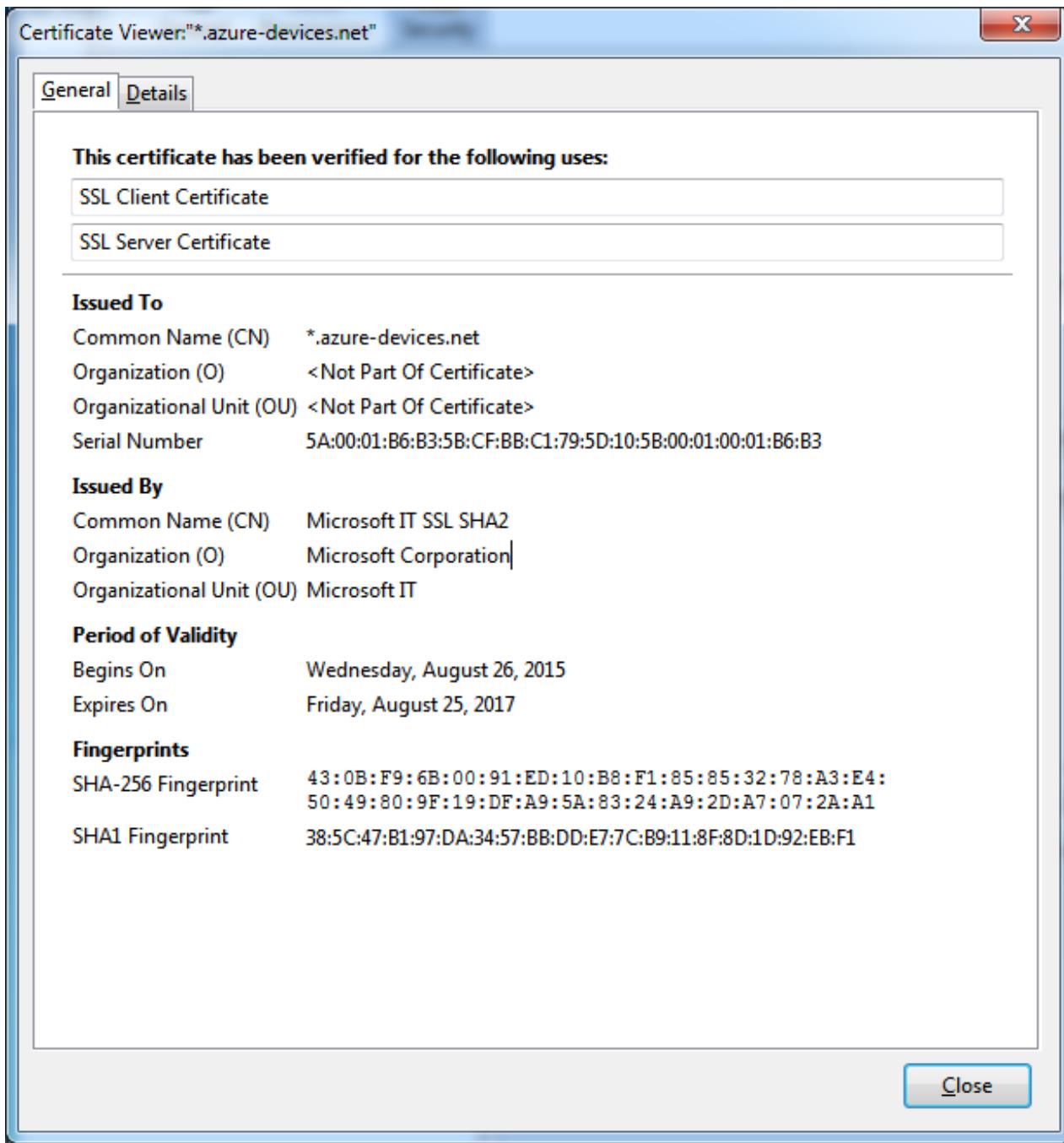
Have I visited this website prior to today? **No**
Is this website storing information (cookies) on my computer? **No** [View Cookies](#)
Have I saved any passwords for this website? **No** [View Saved Passwords](#)

Technical Details

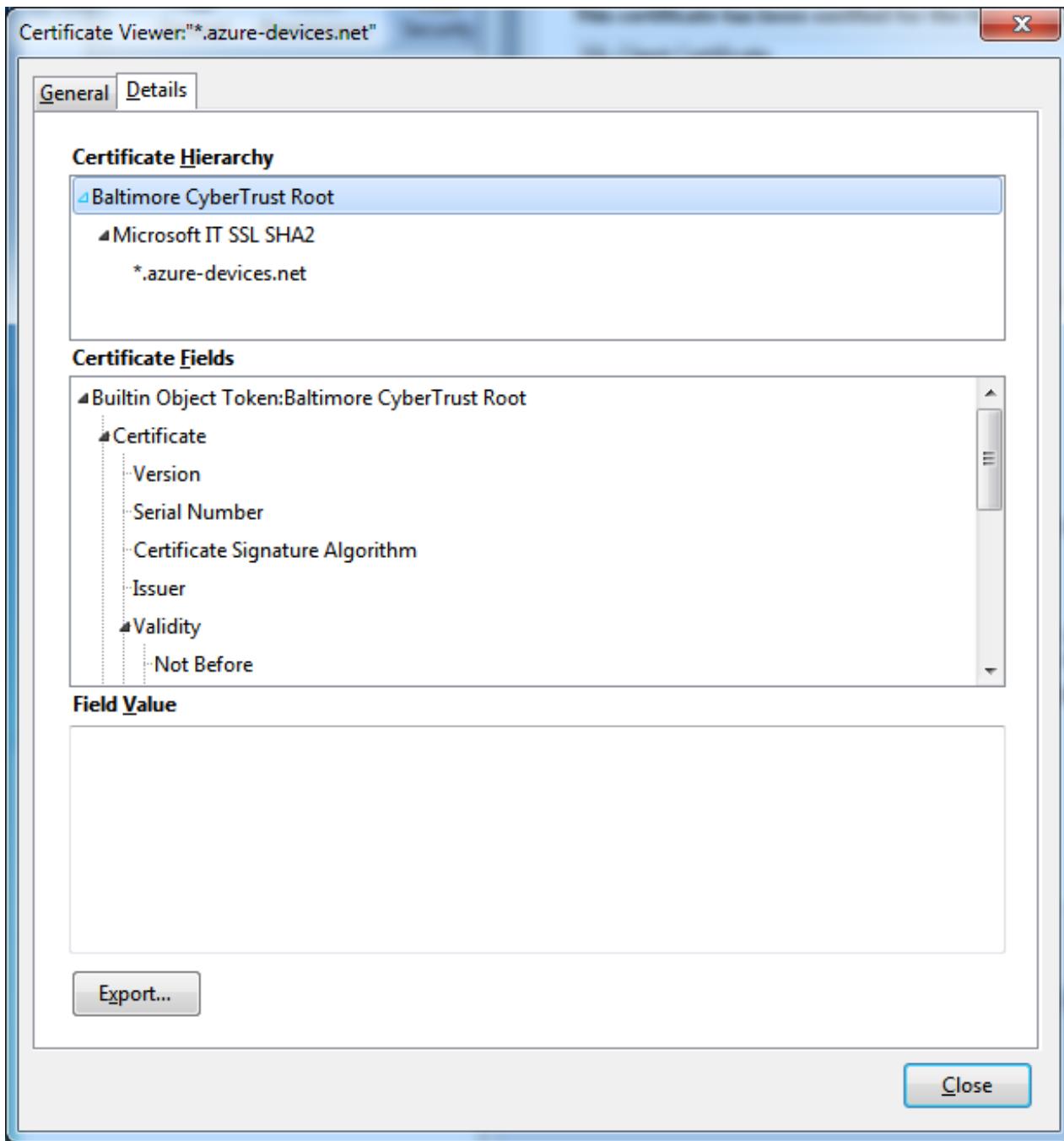
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, 256 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

[Help](#)

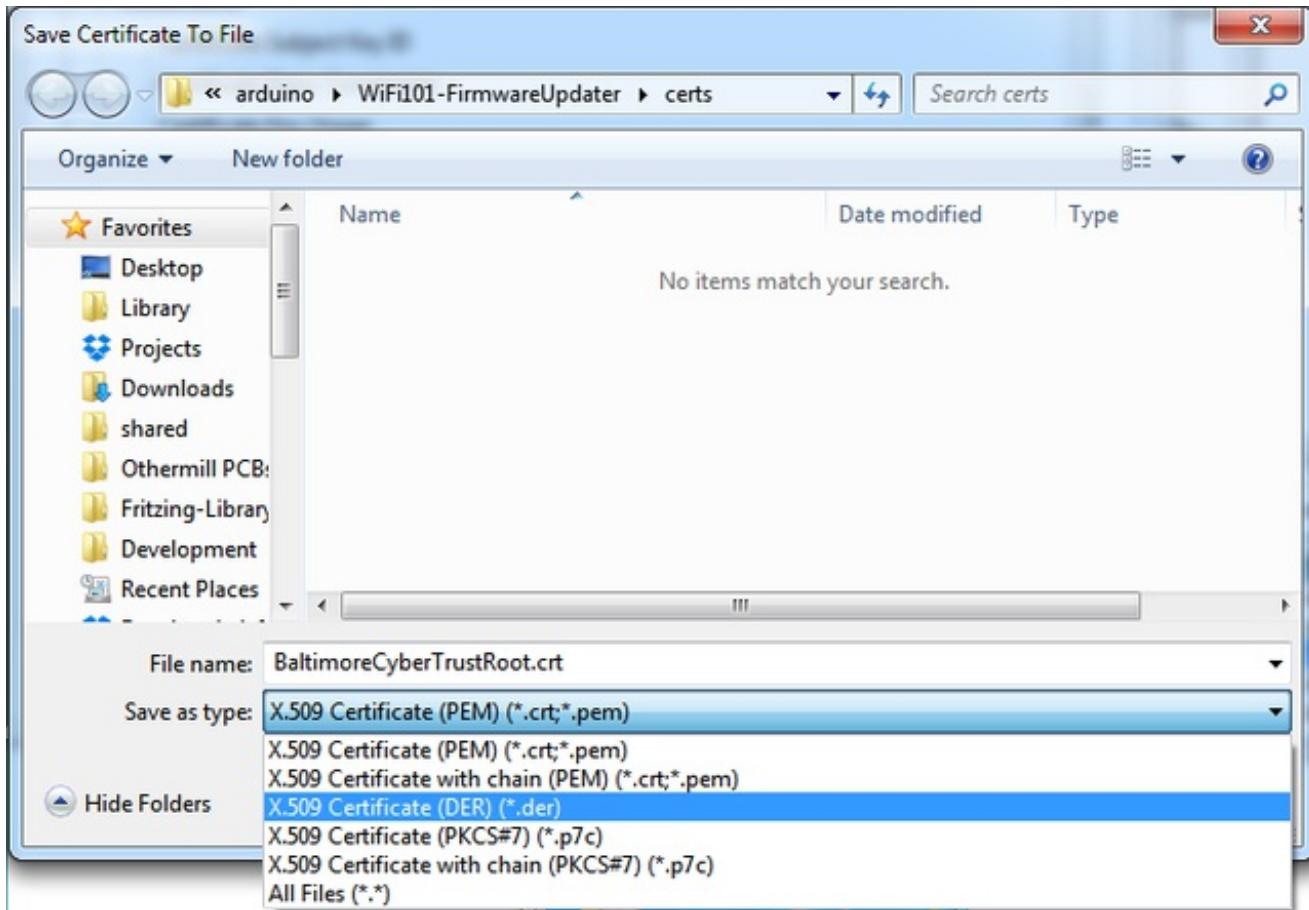
Almost done, once you see the view of the certificate, click on **Details**



Finally, you can see the Root Certificate, its at the top of **Certificate Hierarchy** now click **Export**



and export/save it to the **certs** directory, in **DER** format



Certificate Format

Open up the certificate in a text editor, if you see this you have an ascii certificate **which is not what you want!**

The screenshot shows the XEmacs editor window with the title "X BaltimoreCyberTrustRoot.crt - XEmacs". The menu bar includes File, Edit, View, Cmds, Tools, Options, Buffers, and Help. The toolbar contains icons for Open, Dired, Save, Print, Cut, Copy, Paste, Undo, Spell, Replace, Mail, Info, Compile, Debug, and News. The main buffer area displays a certificate file named "BaltimoreCyberTrustRoot.crt". The content of the buffer is as follows:

```
-----BEGIN CERTIFICATE-----
MIIDdzCCA1+gAwIBAgIEAgAAuTANBgkqhkiG9w0BAQUFADBaMQswCQYDVQQGEwJJ
RTEsMBAGA1UEChMjQmFsdGltb3J1MRMwEQYDVQQLEwpDeWJlc1RydXNOMSiwiAYD
VQQDEx1CYWx0aW1vcnUgQ3liZXJUcnVzdCBSb290MB4XDTAwMDUxMjE4NDYwMFoX
DTI1MDUxMjIxNTkwMFowWjELMAkGA1UEBhMCSUuXejAQBgNVBAoTCUJhbHRpbW9y
ZTETMBEGA1UECxMKQ3liZXJUcnVzdDEiMCAGA1UEAxMZQmFsdGltb3J1IEN5YmVy
VHJ1c3QgUm9vdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKMEuyKr
mD1X6CZymrV51Cni4eiVgLGw41uOKymaZN+hXe2wCQVt2yguzmKiYv60iNoS6zjr
I23AQSSBuNId9Mcj8e6uYilagnnc+gRQKfRzMpijS3ljwumUNKoUMMo6vWrJYeK
mpYcqWe4PwzV9/1SEy/CG9VwcPCPwBLKBsua4dnKM3p31vjsoffoREJIE9LAwqSu
XmD+tqYF/LTdBi1kC1FkYmGP1pWPgkAx9XbIGevOF6uvUA65ehD5f/xTabz5OTZy
dc93Uk3zyZAsuT3lySNTPx8kmCfcB5kpvcY67Oduhjpr13RjM71oGDHweII2v/ye
j10qhqdNkNwnGjkCAwEAaNFMEMwhQYDVR0OBByEFOWdWTCCR1jMrPoIVDaGezq1
BE3wMBIGA1UdEwEB/wQIMAYBAf8CAQMwDgYDVR0PAQH/BAQDAgEGMA0GCSqGSIB3
DQEBBQUAA4IBAQCDF2O5G9RaEIFoN27TyclhAO992T9Ldcw46QQF+vaKSm2eT92
9hkTI7gQCv1YpNRhcLOEYWoSihfVCr3FvDB81ukMJY2GQE/szKN+OMY3EU/t3Wgx
jkzSswF07r51XgdIGn9w/xZchMB5hbgF/X++ZRGjd8ACtPhSNzkElakxehi/oCr0
Epn3oWC4xe9Z2etciefC7IpJ5OCBRLbf1wbWsAY71k5h+3zvDyny67G7fyUIhz
ksLi4xaNmjICq44Y3ekQEe5+NauQrz4wlHrQMz2nZQ/1/I6eYs9HRCwBXbsdtTLS
R9I4LtD+gdwyah617jzV/OeBHRnDJELqYzmp
-----END CERTIFICATE-----
```

At the bottom of the buffer, the status bar shows "Raw: T-----XEmacs: BaltimoreCyberTrustRoot.crt (Fundamental PenDel) ----L22--A1".

Instead, make sure its in binary format like this (it should be a jumble of characters)

X BaltimoreCyberTrustRoot.der - XEmacs

File Edit View Cmds Tools Options Buffers Help

Open Dired Save Print Cut Copy Paste Undo Spell AB₂ Replace Mail Info Compile Debug News

BaltimoreCyberTrustRoo... BaltimoreCyberTrustRoo...

```

D\202^Cw0\202^B_ ^C^B^A^B^B^D^B^@^@^0
^F      *\206H\206+
^A^A^E^E^@Z1^K0      ^F^CU^D^F^S^BIE1^R0^P^F^CU^D
^S      Baltimore1^S0^Q^F^CU^D^K^S
CyberTrust1"0 ^F^CU^D^C^S^YBaltimore CyberTrust Root0^^^W
000512184600Z^W
250512235900Z0Z1^K0      ^F^CU^D^F^S^BIE1^R0^P^F^CU^D
^S      Baltimore1^S0^Q^F^CU^D^K^S
CyberTrust1"0 ^F^CU^D^C^S^YBaltimore CyberTrust Root0\202^A"0
^F      *\206H\206+
^A^A^E^E^@C\202^A^O^@0\202^A
^B\202^A^A^@^D" «\230=Wè&r\232uyÔ) ááé\225\200±ä[\216+)\232d8;]í"      ^EmÛ(.í
bcbp\210Ú^Rë8ë!\235ÀA+^AR\210wÓ^\"217ç^@^210uj      csè^Q@SÑIÊb\215-å\217^Kç
;PÖ^PÄ(éö<^t\207\212\232\226^@g,?^LÖ^ùR^S/Â^[\Öppö\217À^RÉ^FË\232áÜÈ3zwÖsi^ñhDBHç
^SÖÀÄm@^p^;^Eü^Y^GY^BÖY^X\230cöYcà\220^L])^Fzö\205eeÖ^C@^/204>_y^Uiiaù96ruïwRç
MóÉ\220, ^=åÉ#S?^$_\230!^\G\231)çE:icn\206:k\227tc3^h^X18x\215vçü\236\216]*\206Sç
M\220Ü'Z9^B^C^A^@^AEE0C0^] ^F^CU^] ^N^D^V^D^Tå\235Y0\202GXî-ú^HT6\206{:u^DM80^R^ç
F^CU^] ^S^A^A^y^D^H0^F^A^A^y^B^A^C0^N^F^CU^] ^O^A^A^y^D^D^C^B^A^F0
^F      *\206H\206+
^A^A^E^E^@^C\202^A^A^@^L]\216äoQhB^E Y»O'%\204^C^dý-^Oä^P^WéU) ) y?vö^Y^S#ç
,^P
ùX^öap^Daj^R\212^WÖ
^A^H0|Öé^L%^\215\206@Oii^z^8^Z^QOíÍh1\216Lò^Atí^u^GH^Z^?pü^V\204Ày\205, ^Eý^?^e^ç
QE^OÀ^B^øR79^DÖ@1z^Xç *ö^R\231^EE\202ä^ç\235\236uÈ\236].È\236N^H^TKmýpmk^Zc^sdç
^=^_Íöö\237.»[·òP\210s\222Åää^V\215\2322^B«\216^Xé^P^Qi~5«\220->0\224zD3=§e^Oç
öü\216\236bIGD,^A]»^]u2ÖGö8.Ðp\201Ü2j^mu<Öüç\201]^YÄ$Béç9ç

```

Raw:t-----XEmacs: BaltimoreCyberTrustRoot.der (Fundamental PenDel)-----L1--All

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://adafru.it/mgf>)).

The supported ciphers are:

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256

- 1. TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- 1. TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- 1. TLS_RSA_WITH_AES_128_CBC_SHA256
- 1. TLS_RSA_WITH_AES_128_CBC_SHA

Adapting Sketches to M0

The ATSAMD21 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 is `analogReference(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:



The screenshot shows the Arduino IDE interface with a sketch titled "datecalc". The code includes #include <Wire.h> and #include "RTClib.h". It contains a conditional compilation block for SAMD architecture, defining Serial as SerialUSB if ARDUINO_ARCH_SAMD is defined. The sketch also includes a showDate function that prints a string and a space character to the serial port.

```
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(" ");
14 }
```

AnalogWrite / PWM

We've noticed that some PWM outputs are not working with the current SAMD core, its something that is being worked on!

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 wont 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/lFS>)

How Much RAM Available?

The ATSAMD21G18 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 \$3FFF Then it is in FLASH

FAQ

My Feather M0 won't enumerate anymore and can't be programmed, help!

If something happens and your Feather M0 won't enumerate as a USB serial device, like perhaps you were exploring the ATSAMD21's peripherals and accidentally misconfigured something, don't worry you can try a few things to revive it back to normal.

- First try pressing the reset button twice like a mouse double click. The LED should start pulsing red to inform you the chip has entered its bootloader and is waiting a program upload.
- The Feather M0 will show up as a new COM port device (if you haven't already installed the Windows drivers, do that!)
- Select the new bootloader COM port in the Arduino IDE
- Now try uploading a simple blink example from the Arduino IDE to see if that gets the board back into a good state.

If you don't get a pulsing red LED and the board doesn't enumerate as a serial device then something has happened to the bootloader. Unfortunately the best option in this case is to connect to the single-wire debug test points on the back of the board (the SWDIO & SWCLK pads) and manually reprogram the bootloader using a J-Link or ST-Link ARM programmer.

Downloads

Datasheets

- [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 Adafruit_WINC1500 but its still very handy reference
- [ATSAMD21 Datasheet](http://adafru.it/lDE) (<http://adafru.it/lDE>) - Its long, but its a good read