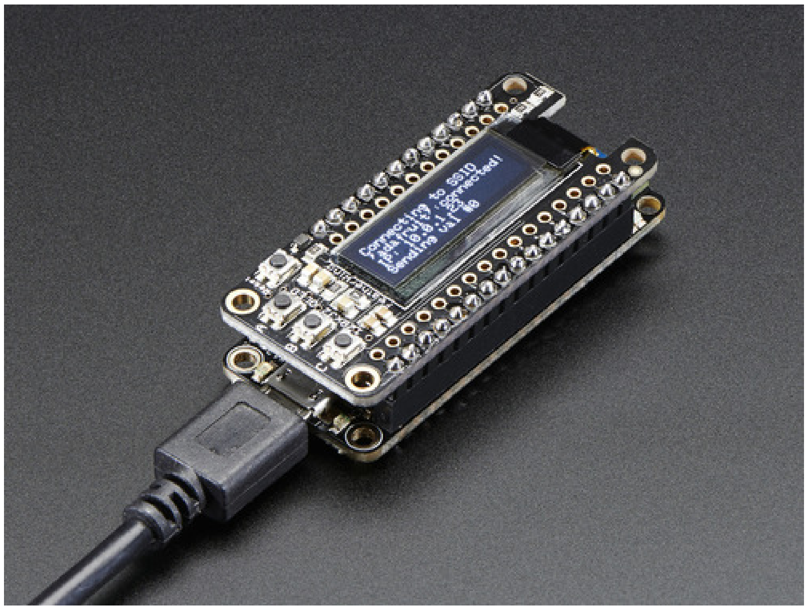
adafruit learning system

Adafruit Feather M0 WiFi with ATWINC1500

Created by lady ada

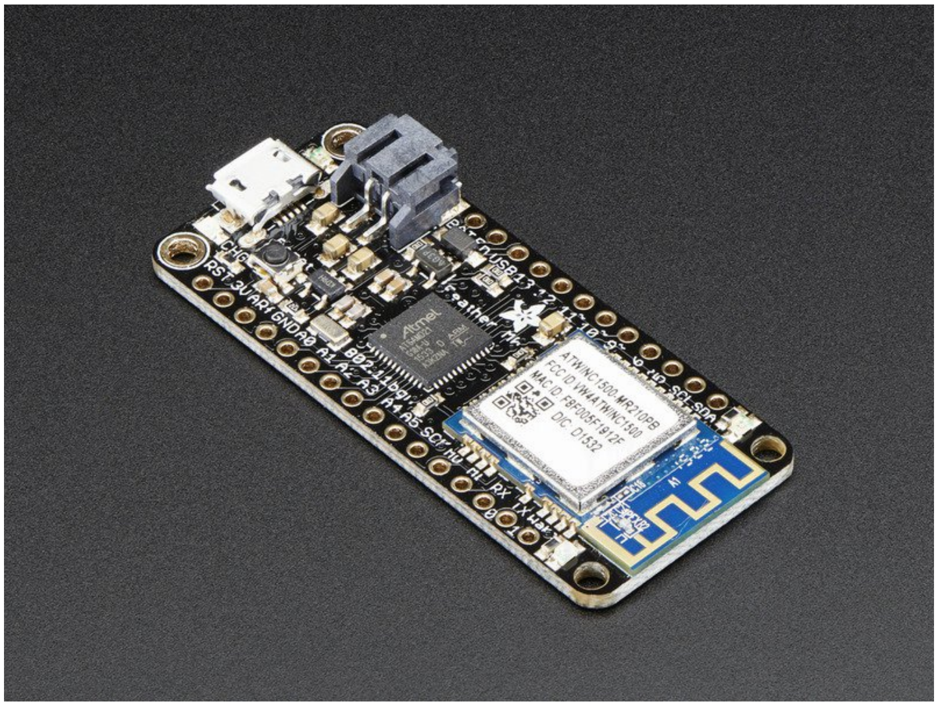


Last updated on 2019-09-05 09:40:59 PM UTC

Overview

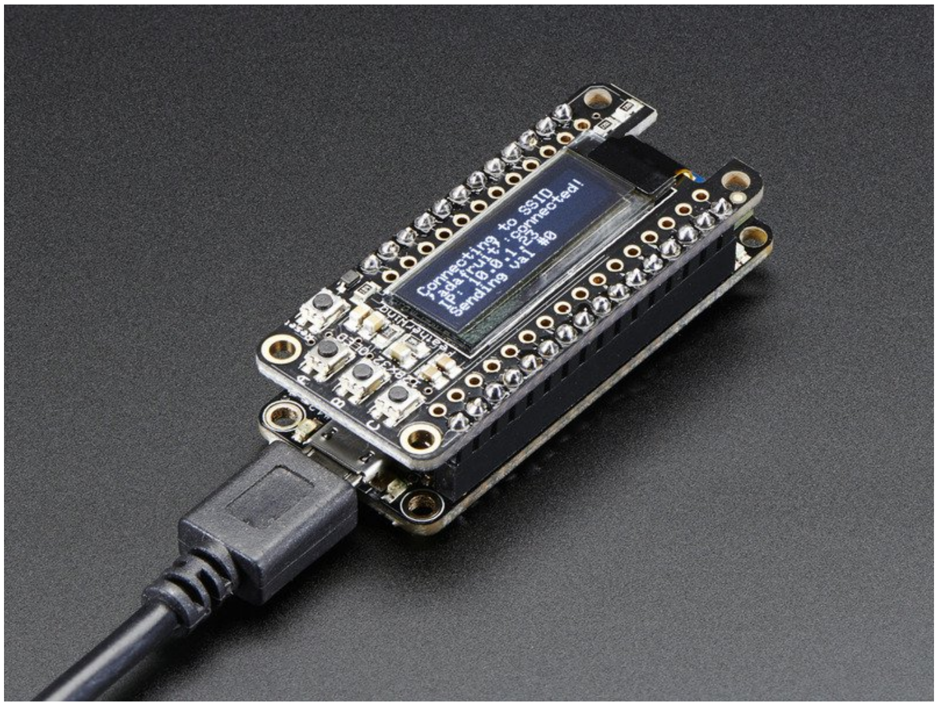
Feather is the new development board from Adafruit, and like its namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores.

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 (https://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.

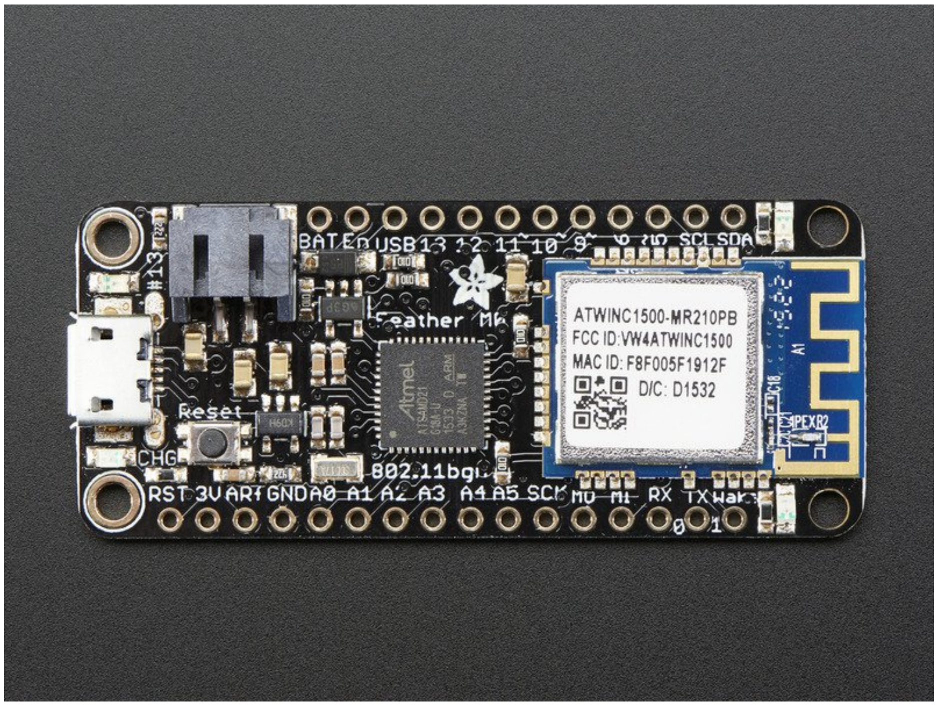
This module works with 802.11b, g, or n networks & supports WEP, WPA and WPA2 encryption. You can connect to your own WiFi networks or create your own with "Soft AP" mode, where it becomes its own access point (we have an example of it creating a webserver that you can then control the Arduino's pins). 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.



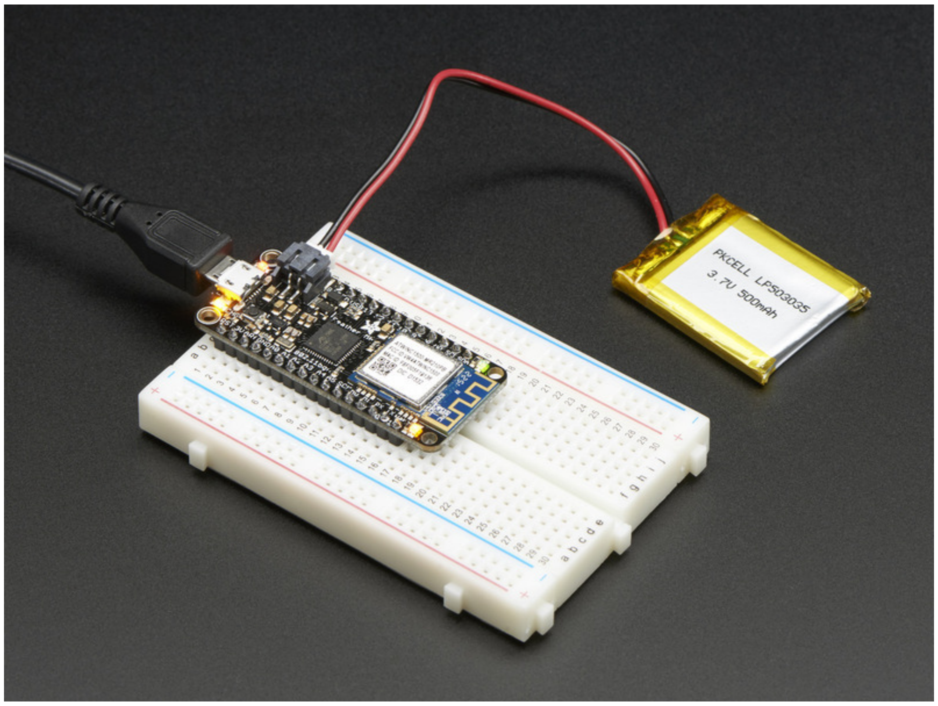
You might be wondering why use this when you can get a HUZZAH Feather? (http://adafru.it/2821) Well, you get

- 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 (https://adafru.it/l3e))
- 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

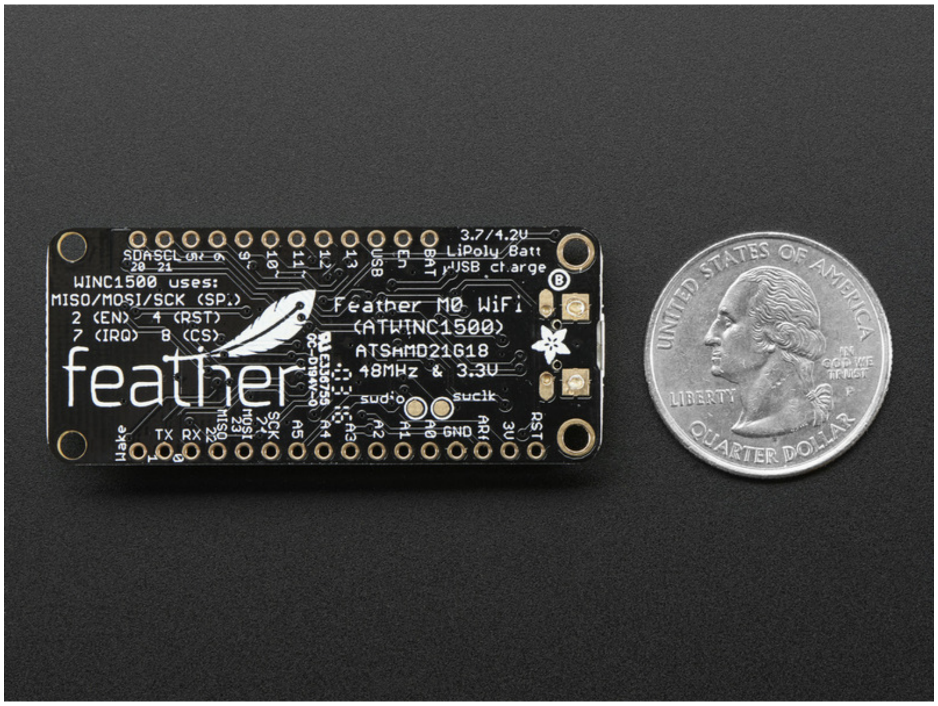
Of course, both WiFi-capable Feathers have their strengths and tradeoffs, & we love both equally!



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). 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.

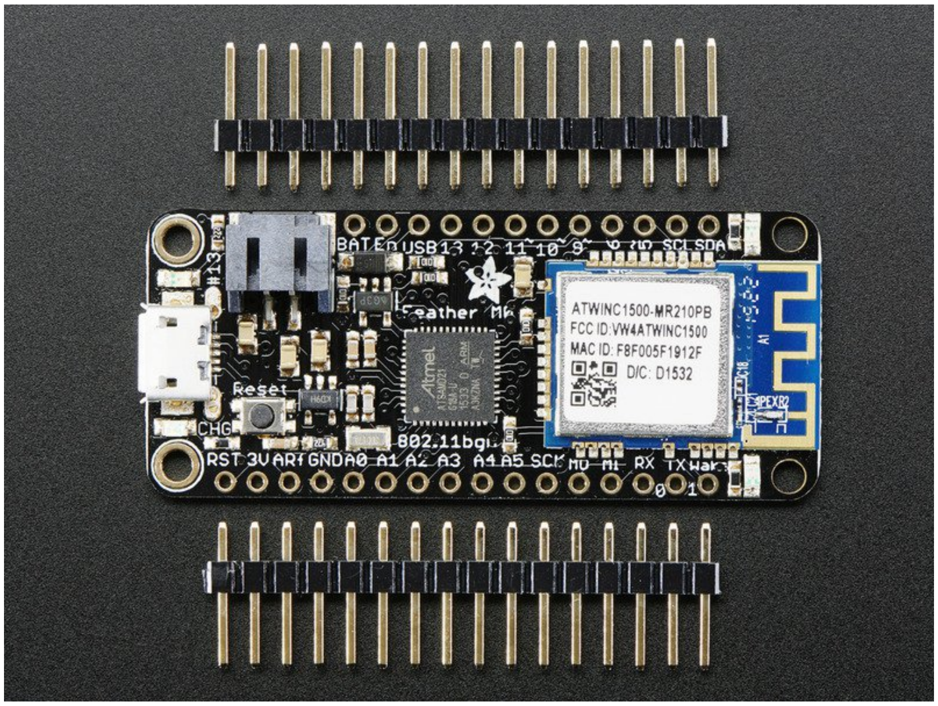


To make it easy to use for portable projects, we added a connector for any of our 3.7V Lithium polymer batteries and built in battery charging. You don't need to use a battery, it will run just fine straight from the micro USB connector. But, if you do have a battery, you can take it on the go, then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



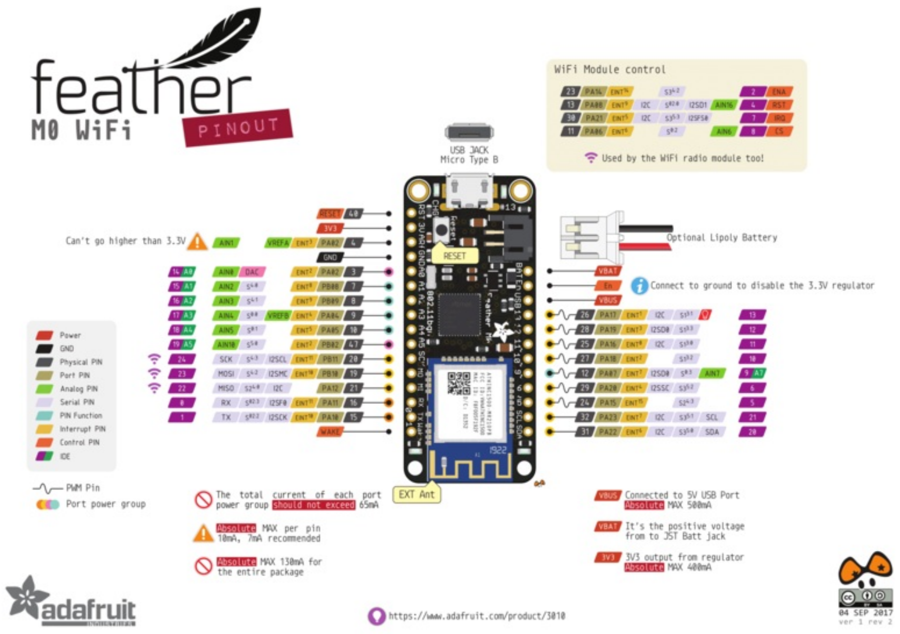
Here's some handy specs! Like all Feather M0's you get:

- 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



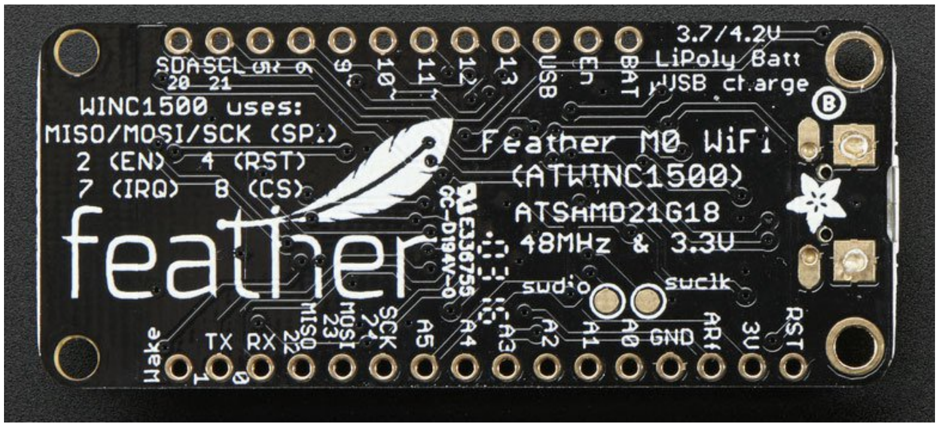
Comes fully assembled and tested, with a USB bootloader that lets you quickly use it with the Arduino IDE. We also toss in some header so you can solder it in and plug into a solderless breadboard. **Lipoly battery (https://adafru.it/e0v) and MicroUSB cable (https://adafru.it/aM5) not included** (but we do have lots of options in the shop if you'd like!)

Pinouts

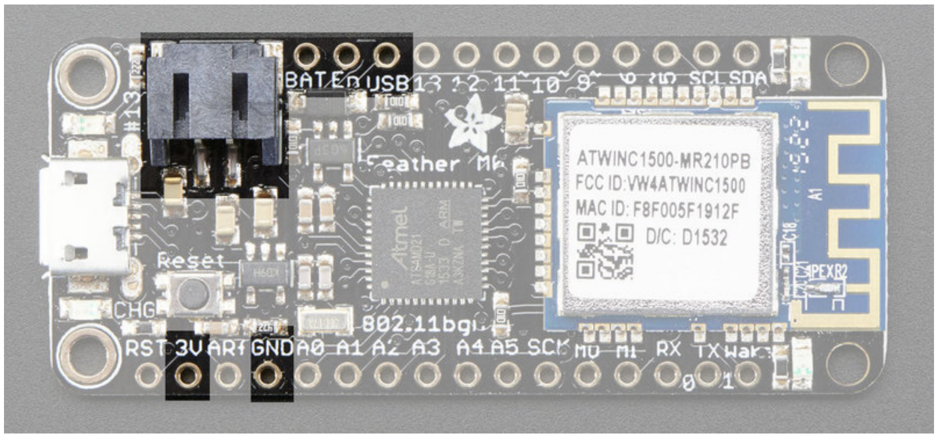


Note AREF in the diagram should be marked PA03 not PA02

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



BAT En USB 13 12 11 10 9 6 5 SCL SDA
ATWINC1500-MR210PB
FCC ID:VW4ATWINC1500
MAC ID: F8F005F1912F
D/C: D1532
Reset
CHG
802.11bgn
RST 3V ARf GND A0 A1 A2 A3 A4 A5 SCK MO MI RX TX Wake
0 1

- **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

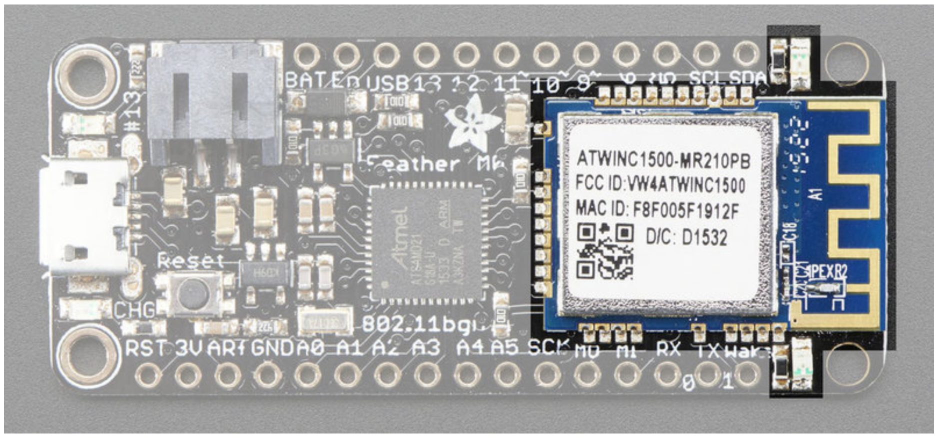
This is the general purpose I/O pin set for the microcontroller.
All logic is 3.3V
Nearly all pins can do PWM output
All pins can be interrupt inputs

- **#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

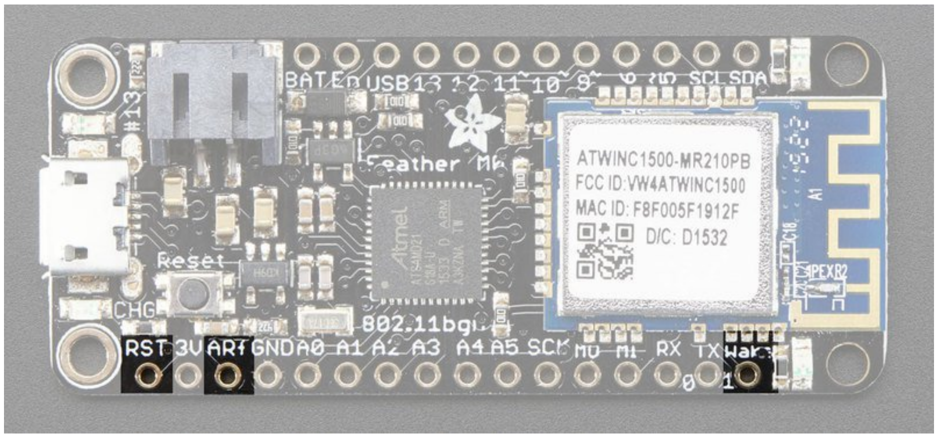


Since not all pins can be brought out to breakouts, due to the small size of the Feather, we use these to control the WiFi module

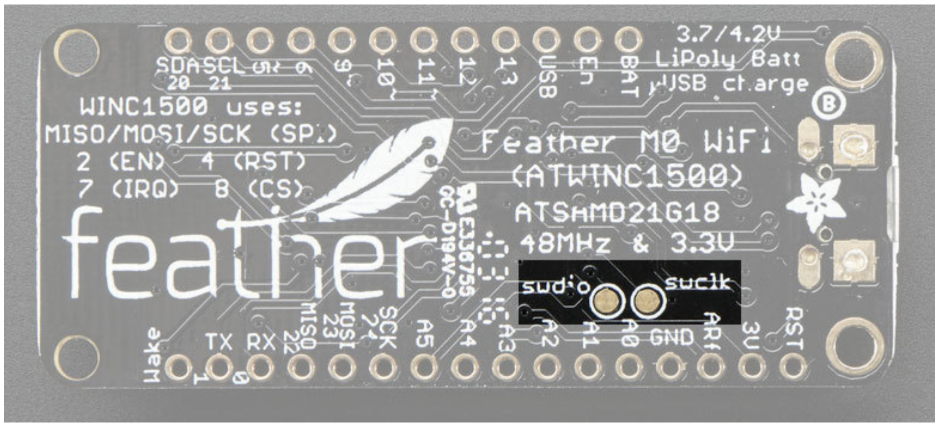
- **#2** - used as the **EN**able pin for the WiFi module, by default pulled down low, set HIGH to enable WiFi
- **#4** - used as the **Reset** pin for the WiFi module, controlled by the library
- **#7** - used as the **IRQ** interrupt request pin for the WiFi module, controlled by the library
- **#8** - used as the **C**hip **S**elect 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.

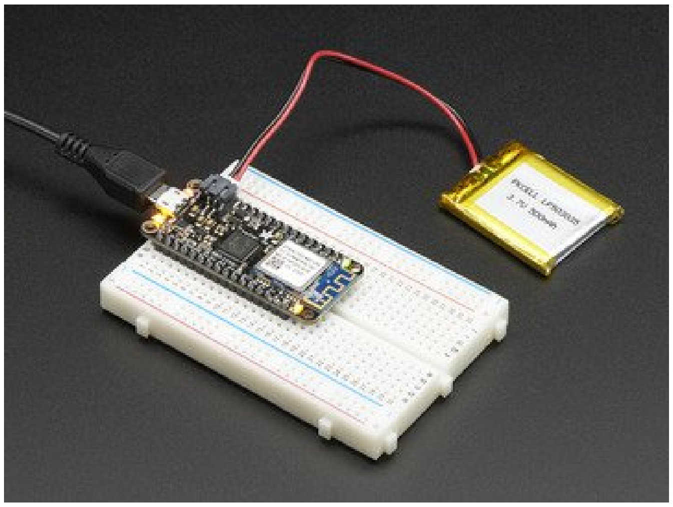


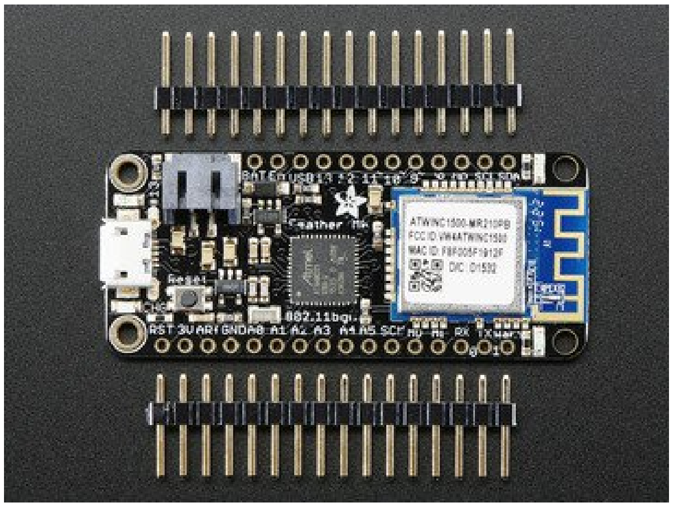
Assembly

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

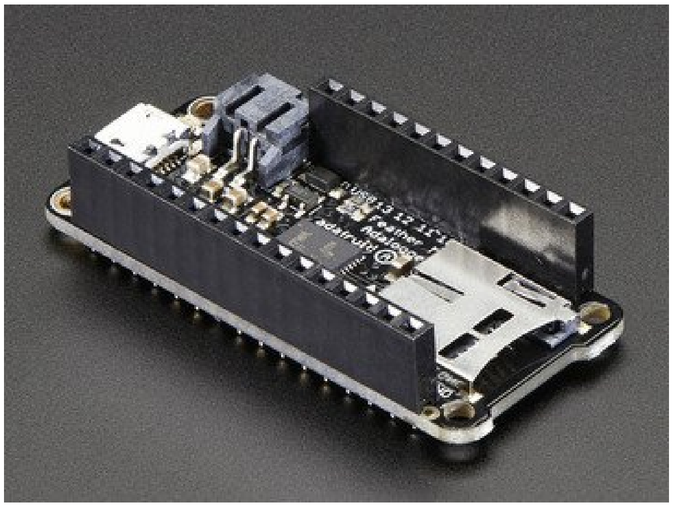
Header Options!

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

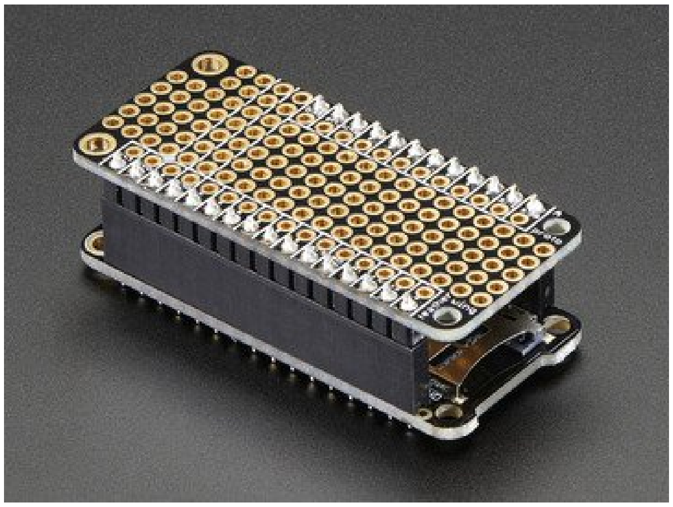


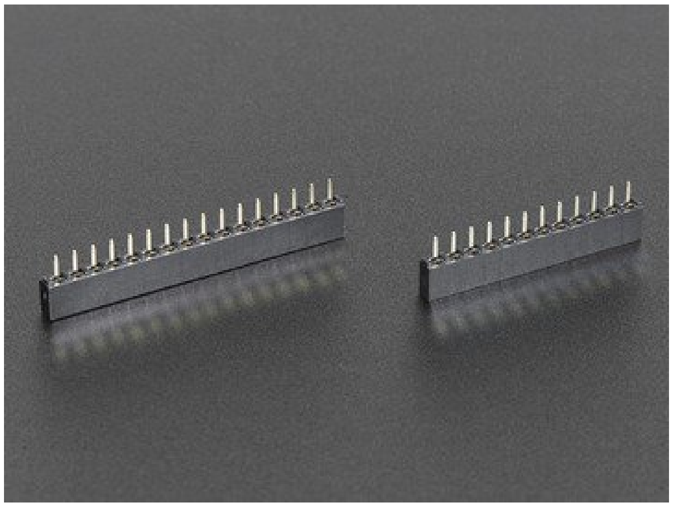


The first option is soldering in plain male headers, this lets you plug in the Feather into a solderless breadboard

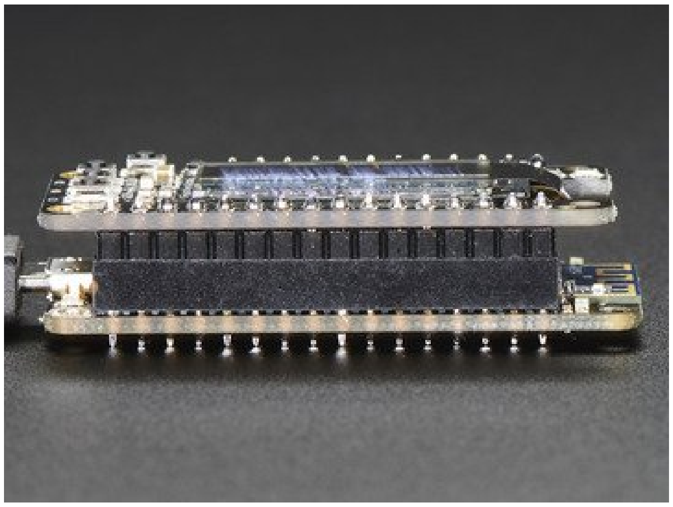


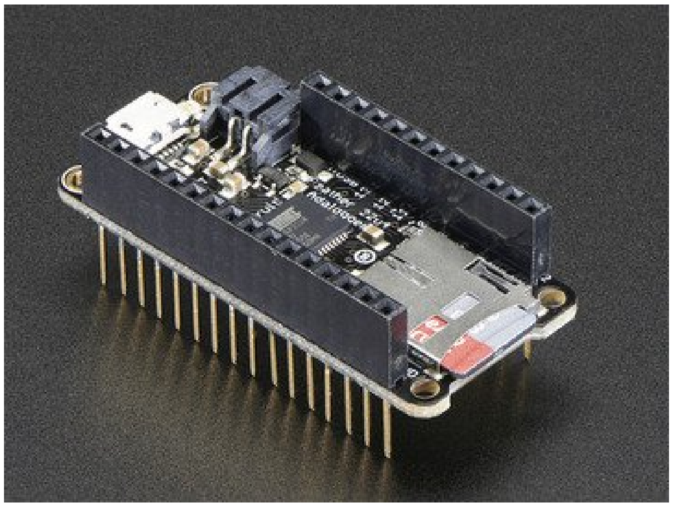
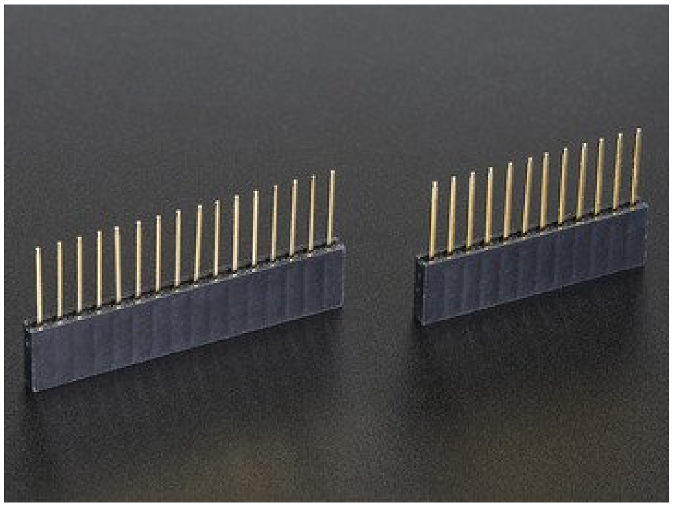
Another option is to go with socket female headers. This won't let you plug the Feather into a breadboard but it will let you attach featherwings very easily





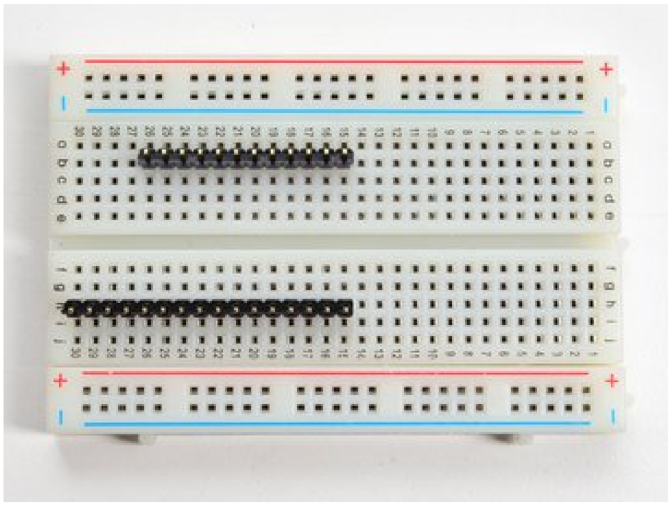
We also have 'slim' versions of the female headers, that are a little shorter and give a more compact shape



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

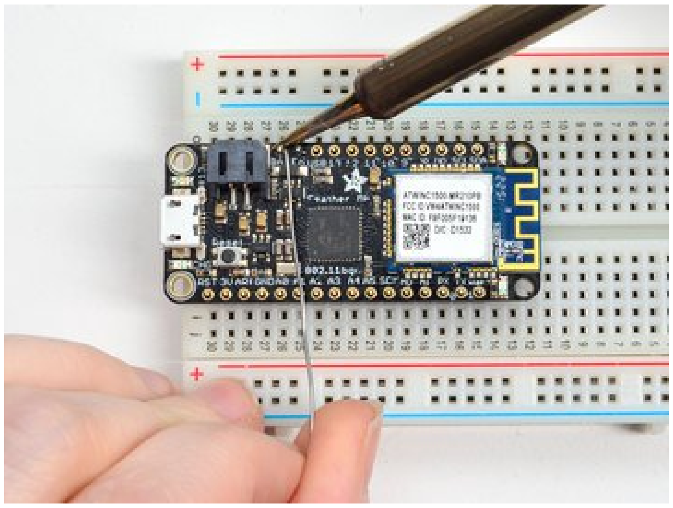


Prepare the header strip:

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

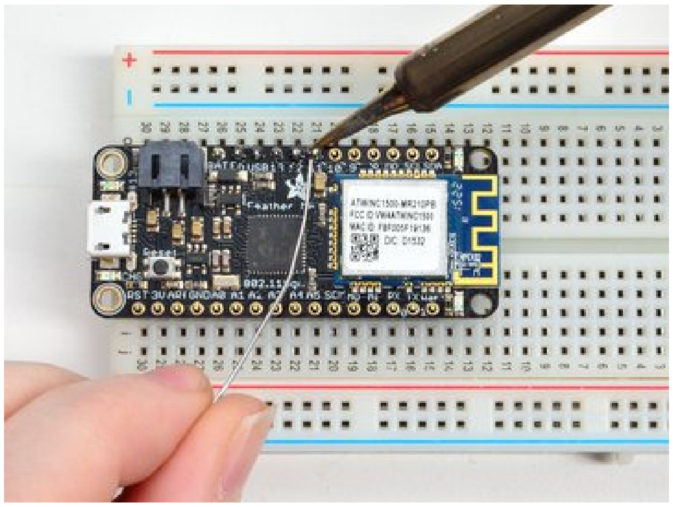


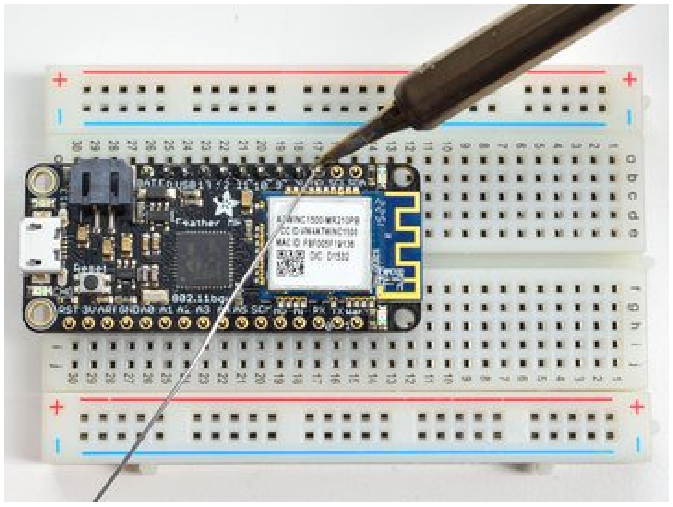
pins poke through the breakout pads

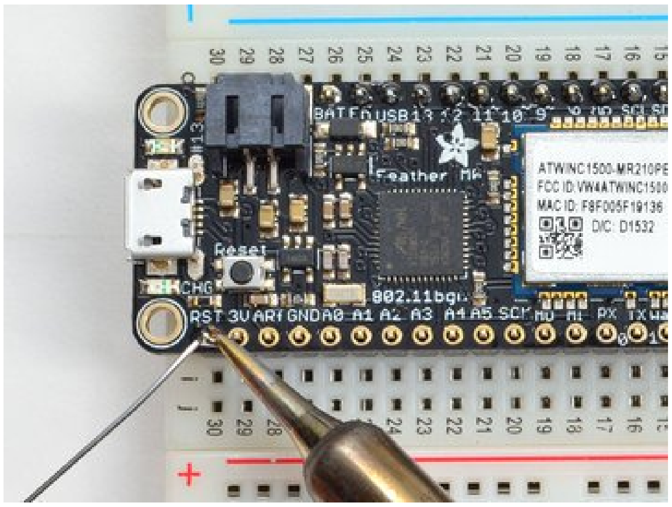
And Solder!

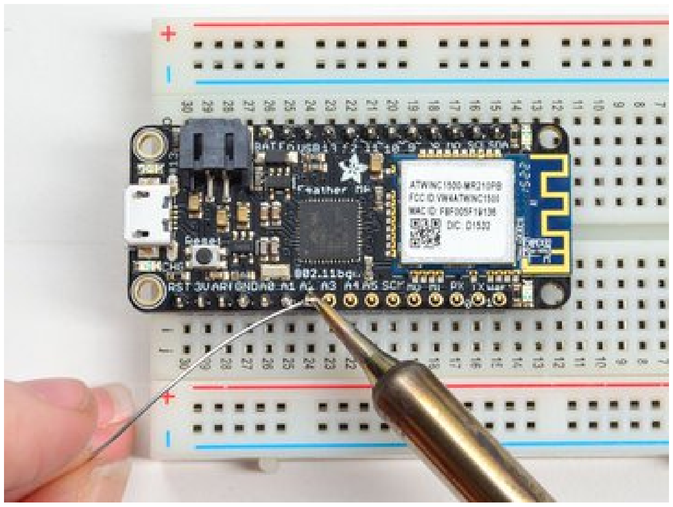
Be sure to solder all pins for reliable electrical contact.

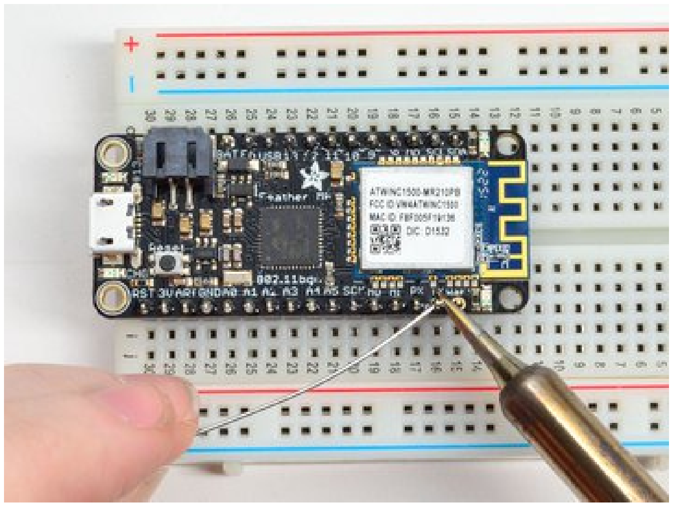
(For tips on soldering, be sure to check out our Guide to Excellent Soldering (https://adafru.it/aTk)*).*

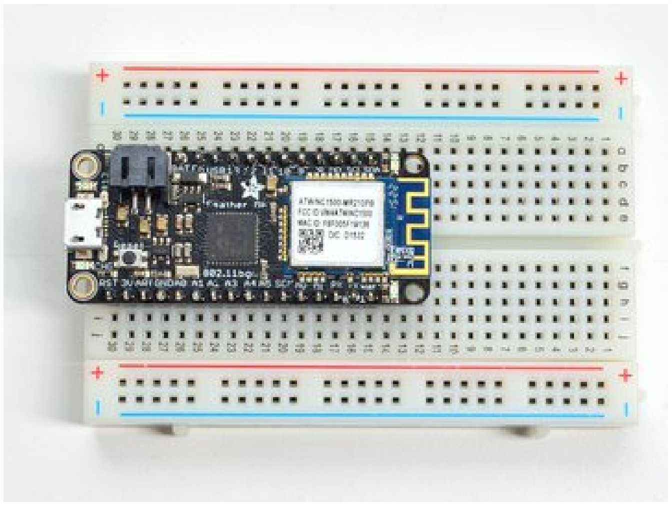





Solder the other strip as well.

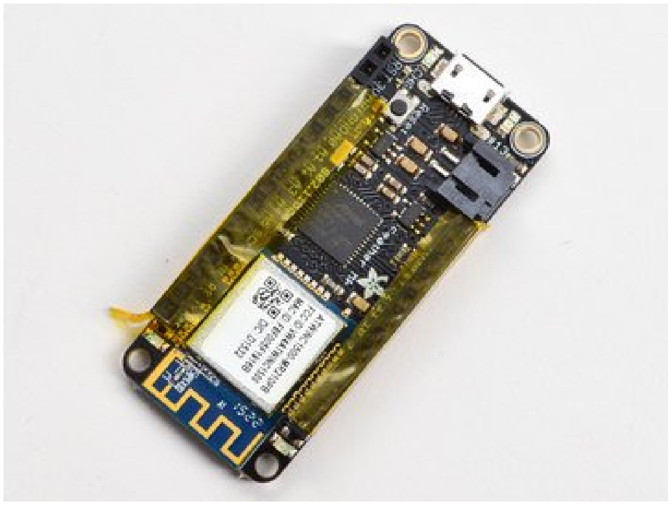






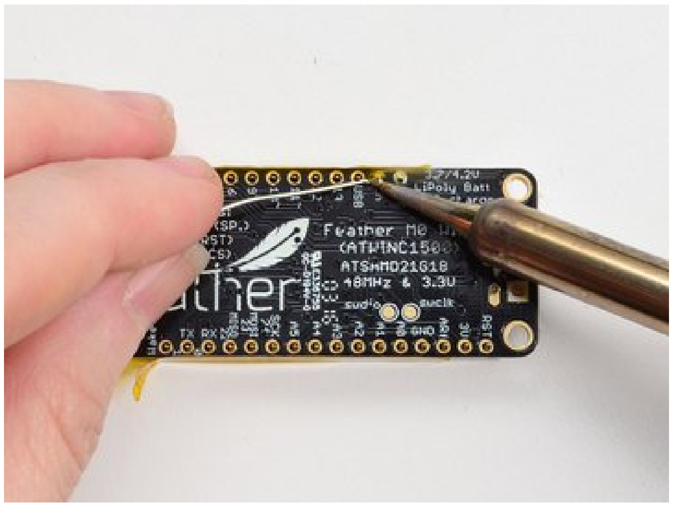
You're done! Check your solder joints visually and continue onto the next steps

Soldering on Female Header



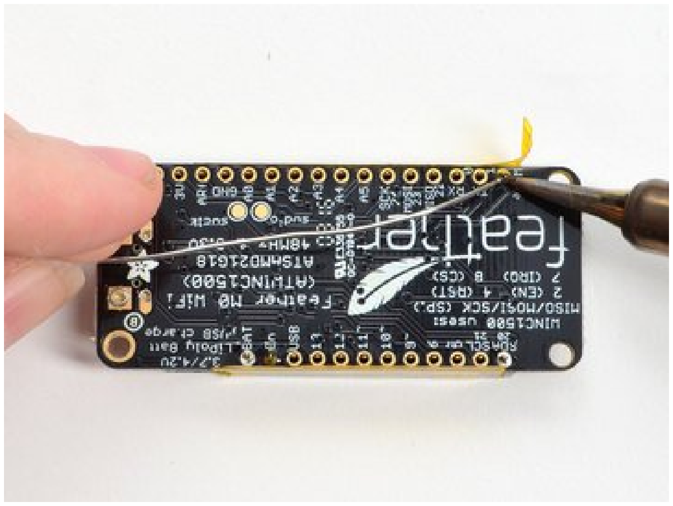
Tape In Place

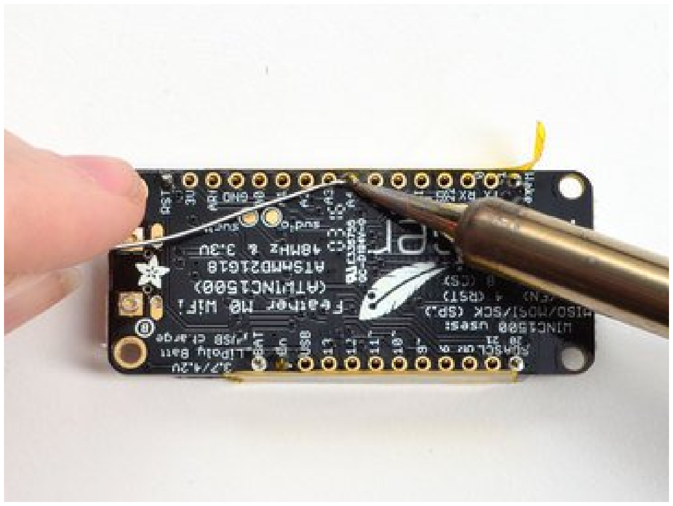
For sockets you'll want to tape them in place so when you flip over the board they don't fall out

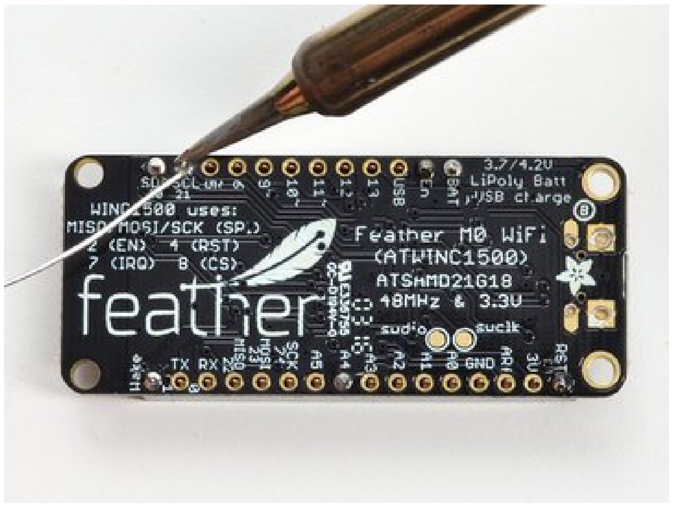


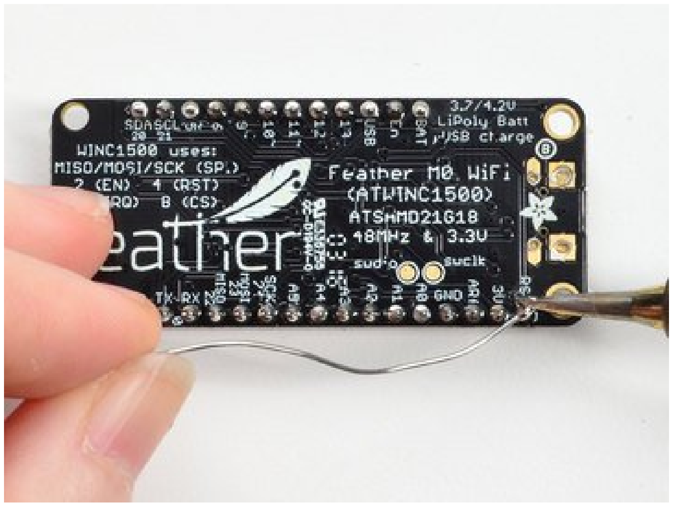
Flip & Tack Solder

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





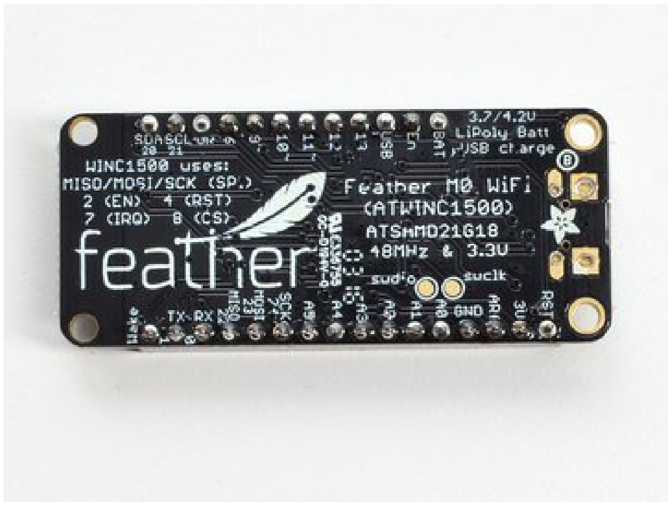




And Solder!

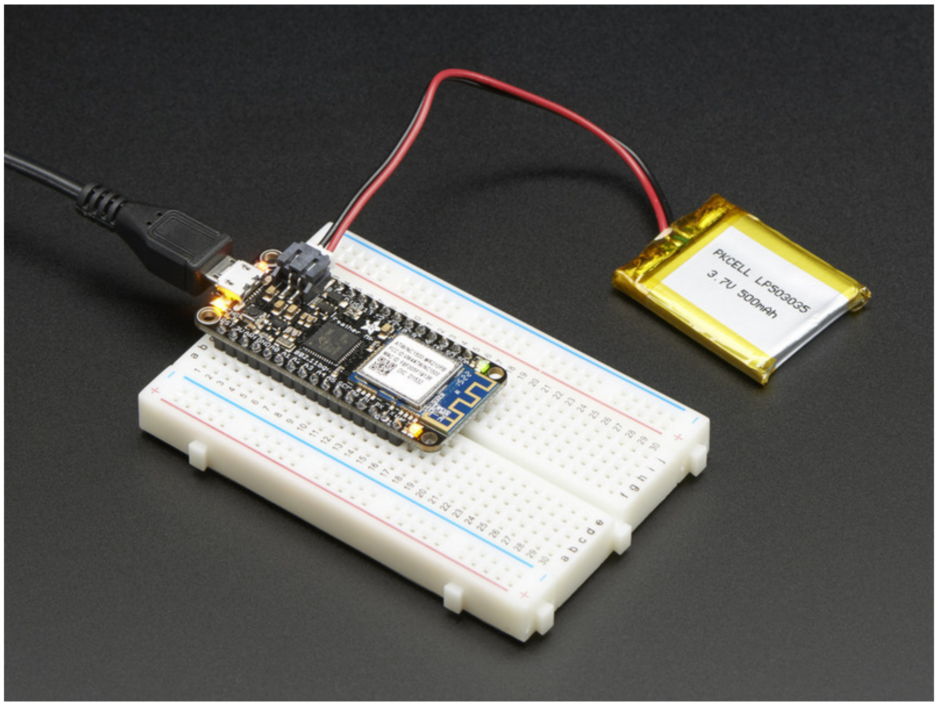
Be sure to solder all pins for reliable electrical contact.

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



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

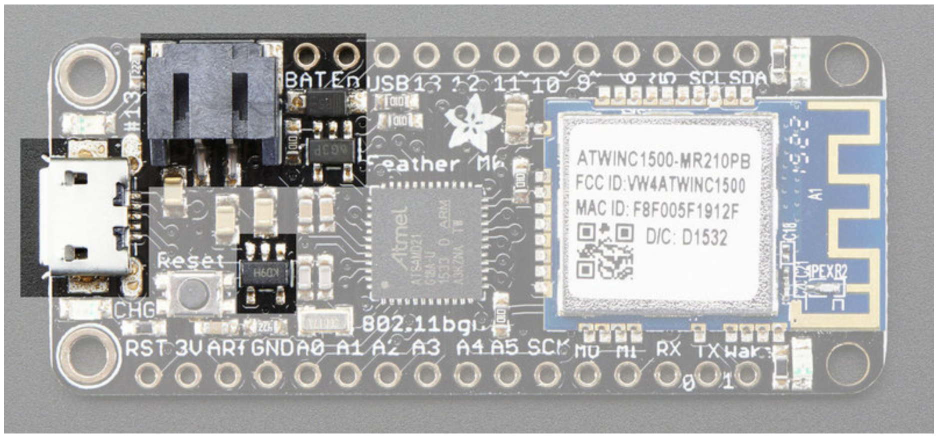
Power Management



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 (LiIon) battery to the JST jack. This will let the Feather run on a rechargable battery. **When the USB power is powered, it will automatically switch over to USB for power, as well as start charging the battery (if attached) at 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

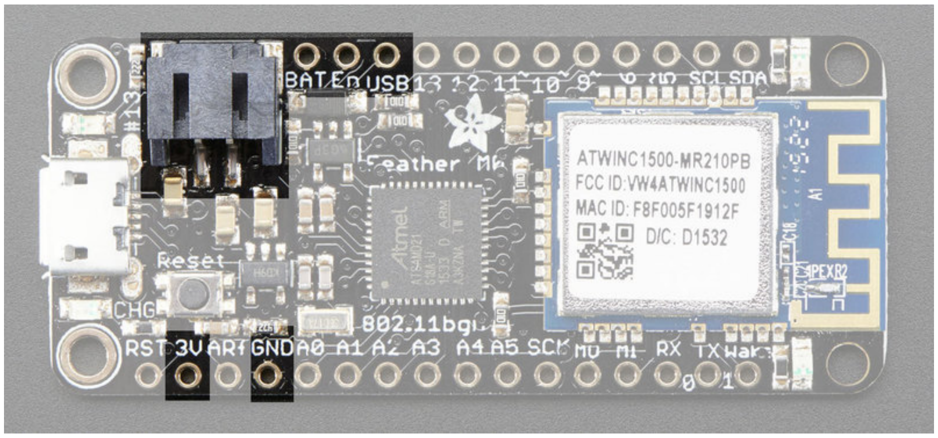


The above shows the Micro USB jack (left), Lipoly JST jack (top left), as well as the 3.3V regulator and changeover diode (just to the right of the JST jack) and the Lipoly charging circuitry (to the right of the Reset button). There's also a **CHG** LED, which will light up while the battery is charging. This LED might also flicker if the battery is not connected.

The charge LED is automatically driven by the Lipoly charger circuit. It will try to detect a battery and is expecting one to be attached. If there isn't one it may flicker once in a while when you use power because it's trying to charge a (non-existant) battery. It's not harmful, and its totally normal!

Power supplies

You have a lot of power supply options here! We bring out the **BAT** pin, which is tied to the lipoly JST connector, as well as **USB** which is the +5V from USB if connected. We also have the **3V** pin which has the output from the 3.3V regulator. We use a 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.



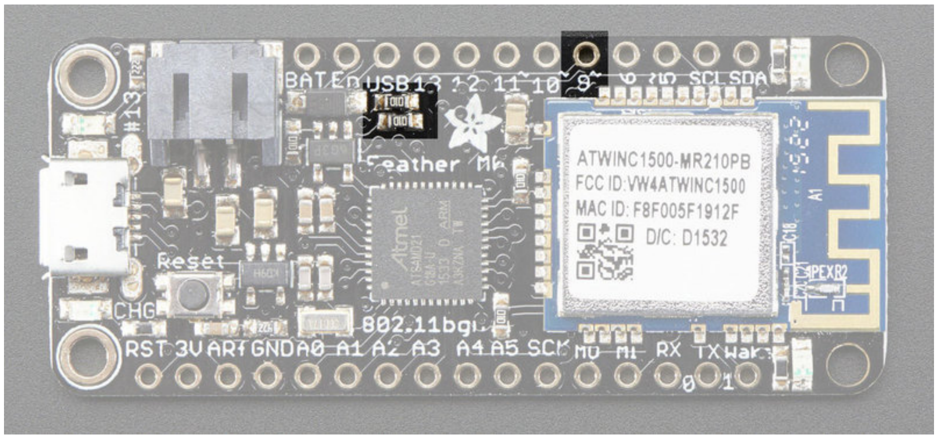
Measuring Battery

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

Alternative Power Options

The two primary ways for powering a feather are a 3.7/4.2V LiPo battery plugged into the JST port *or* a USB power cable.

If you need other ways to power the Feather, here's what we recommend:

- For permanent installations, a 5V 1A USB wall adapter (https://adafru.it/duP) will let you plug in a USB cable for reliable power
- For mobile use, where you don't want a LiPoly, use a USB battery pack! (https://adafru.it/e2q)
- If you have a higher voltage power supply, use a 5V buck converter (https://adafru.it/DHs) and wire it to a USB cable's 5V and GND input (https://adafru.it/DHu)

Here's what you cannot do:

- **Do not use alkaline or NiMH batteries** and connect to the battery port - this will destroy the LiPoly charger and there's no way to disable the charger
- **Do not use 7.4V RC batteries on the battery port** - this will destroy the board

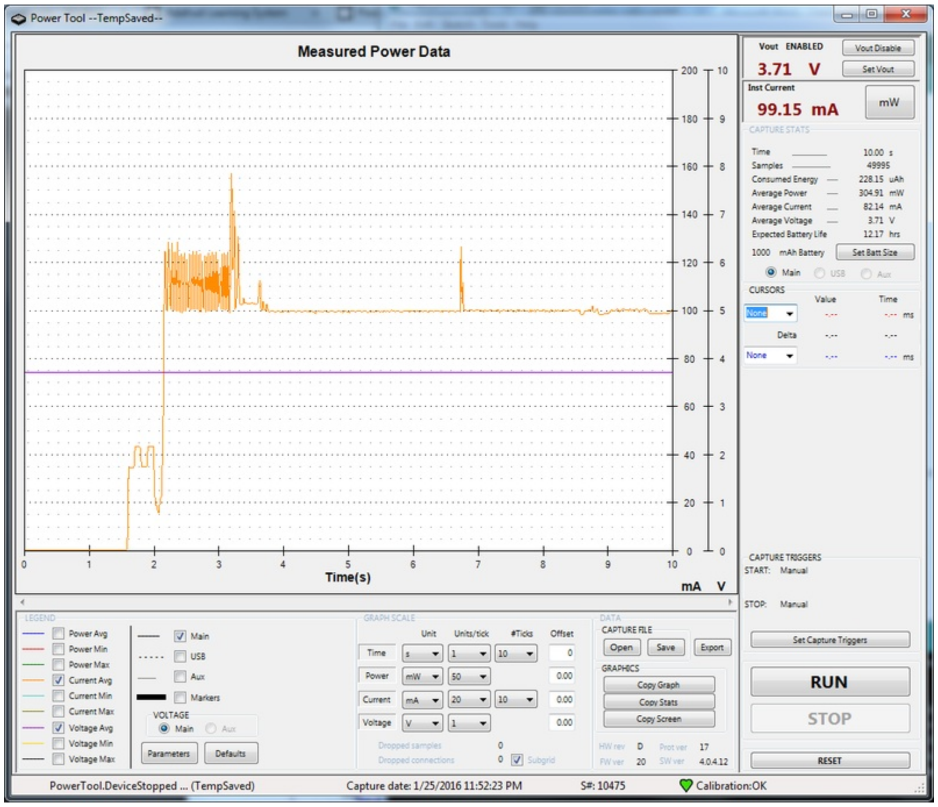
The Feather *is not designed for external power supplies* - this is a design decision to make the board compact and low cost. It is not recommended, but technically possible:

- **Connect an external 3.3V power supply to the 3V and GND pins.** Not recommended, this may cause unexpected behavior and the **EN** pin will no longer. Also this doesn't provide power on **BAT** or **USB** and some Feathers/Wings use those pins for high current usages. You may end up damaging your Feather.

- **Connect an external 5V power supply to the USB and GND pins.** Not recommended, this may cause unexpected behavior when plugging in the USB port because you will be back-powering the USB port, which *could* confuse or damage your computer.

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:



Vout ENABLED
3.71 V
Inst Current
99.15 mA

CAPTURE STATS
Time 10.00 s
Samples 49995
Consumed Energy 228.15 uAh
Average Power 304.91 mW
Average Current 82.14 mA
Average Voltage 3.71 V
Expected Battery Life 12.17 hrs
1000 mAh Battery
(● Main) USB Aux

CAPTURE TRIGGERS
START: Manual
STOP: Manual

PowerTool.DeviceStopped ... (TempSaved) Capture date: 1/25/2016 11:52:23 PM S#: 10475 Calibration:OK

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.

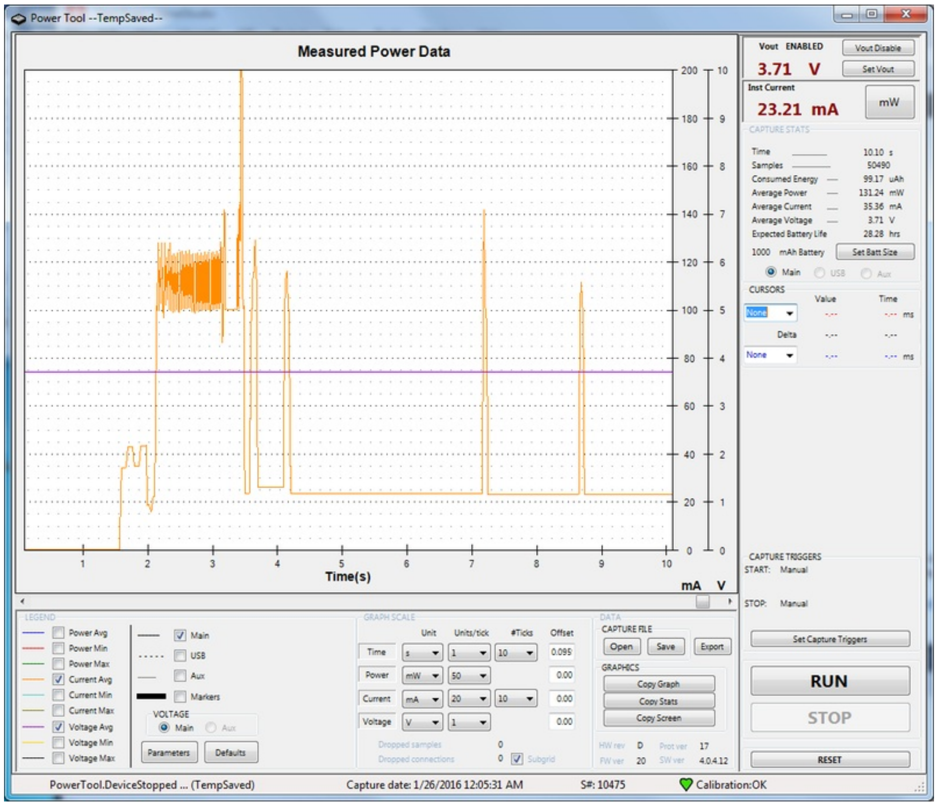
100mA is still quite a bit, you can very easily reduce this by letting the WINC1500 manage its own power:

```
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();
```



Vout ENABLED 3.71 V

Inst Current 23.21 mA

CAPTURE STATS

Time	10.10 s
Samples	50490
Consumed Energy	99.17 uAh
Average Power	131.24 mW
Average Current	35.36 mA
Average Voltage	3.71 V
Expected Battery Life	28.28 hrs

1000 mAh Battery

CAPTURE TRIGGERS START: Manual STOP: Manual

PowerTool.DeviceStopped ... (TempSaved) Capture date: 1/26/2016 12:05:31 AM S#: 10475 Calibration:OK

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

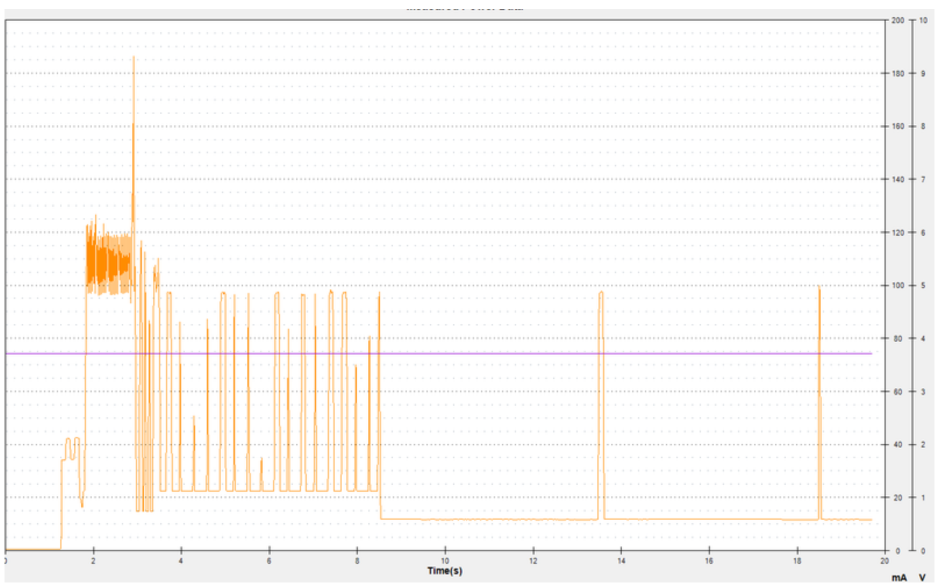
```
WiFi.setSleepMode(M2M_PS_MANUAL, 1);
```

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!


Time(s)
mA
V

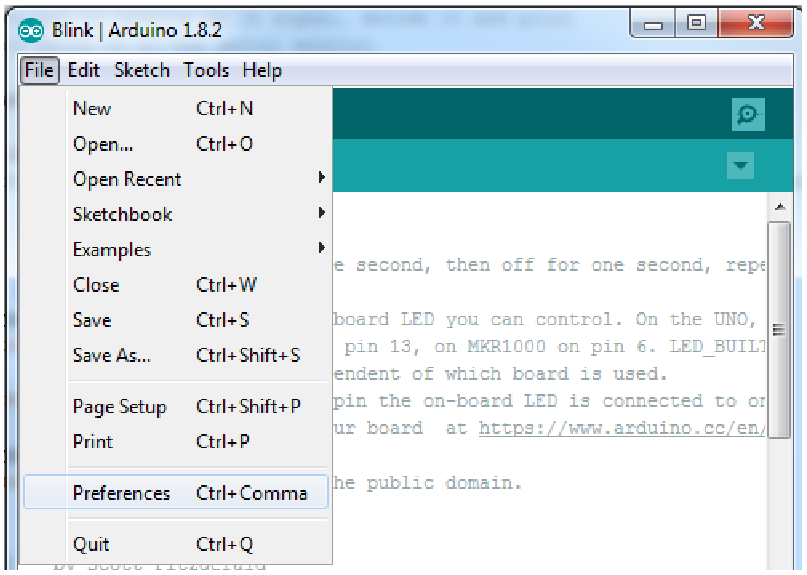
Arduino IDE Setup

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 1.8** or higher for this guide

https://adafru.it/f1P

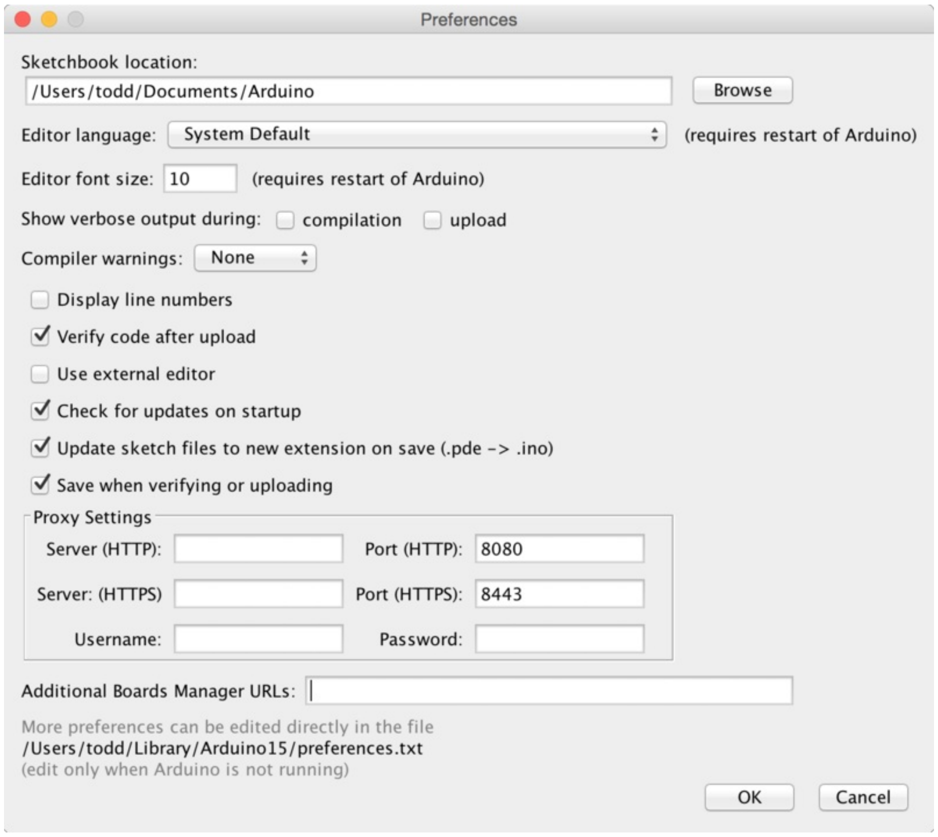
https://adafru.it/f1P

After you have downloaded and installed **the latest version of Arduino IDE**, 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*.



```
e second, then off for one second, repe

board LED you can control. On the UNO,
 pin 13, on MKR1000 on pin 6. LED_BUILT
endent of which board is used.
pin the on-board LED is connected to or
ur board  at https://www.arduino.cc/en/

he public domain.
```

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



Sketchbook location: /Users/todd/Documents/Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 10 (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

✓ Verify code after upload

☐ Use external editor

✓ Check for updates on startup

✓ Update sketch files to new extension on save (.pde -> .ino)

✓ Save when verifying or uploading

Proxy Settings

Server (HTTP):		Port (HTTP):	8080
Server: (HTTPS)		Port (HTTPS):	8443
Username:		Password:	

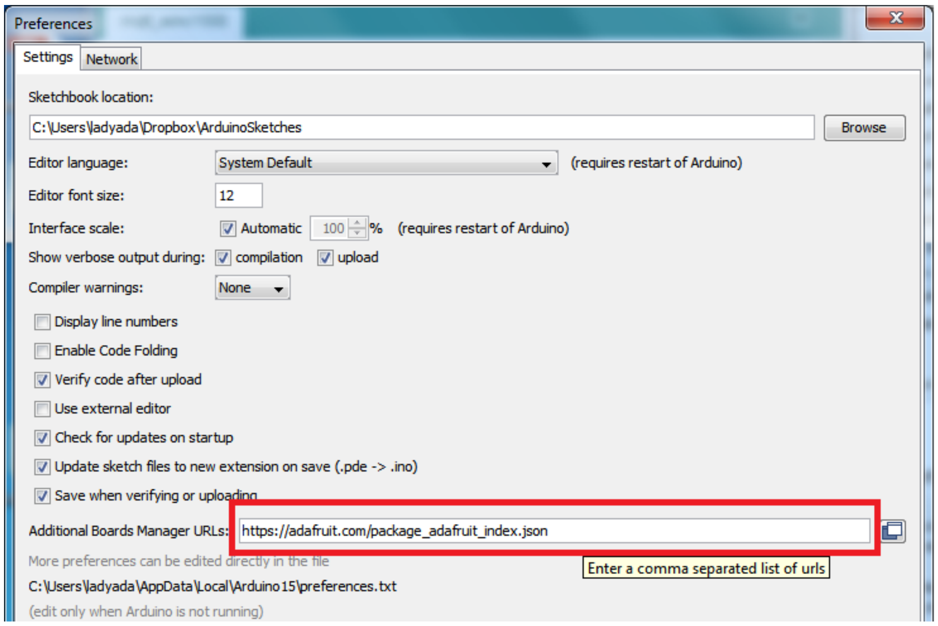
Additional Boards Manager URLs:

More preferences can be edited directly in the file
/Users/todd/Library/Arduino15/preferences.txt
(edit only when Arduino is not running)

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 (https://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
```


Preferences

Sketchbook location: C:\Users\ladyada\Dropbox\ArduinoSketches
Editor language: System Default (requires restart of Arduino)
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Show verbose output during: [✓] compilation [✓] upload
Compiler warnings: None
[ ] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading
Additional Boards Manager URLs: https://adafruit.com/package_adafruit_index.json
Enter a comma separated list of urls
More preferences can be edited directly in the file
C:\Users\ladyada\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

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 and M4, Metro M0 and M4, ItsyBitsy M0 and M4, Circuit Playground Express, Gemma M0 and Trinket M0
- **Arduino Leonardo & Micro MIDI-USB** - This adds MIDI over USB support for the Flora, Feather 32u4, Micro and Leonardo using the arcore project (https://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 (,)

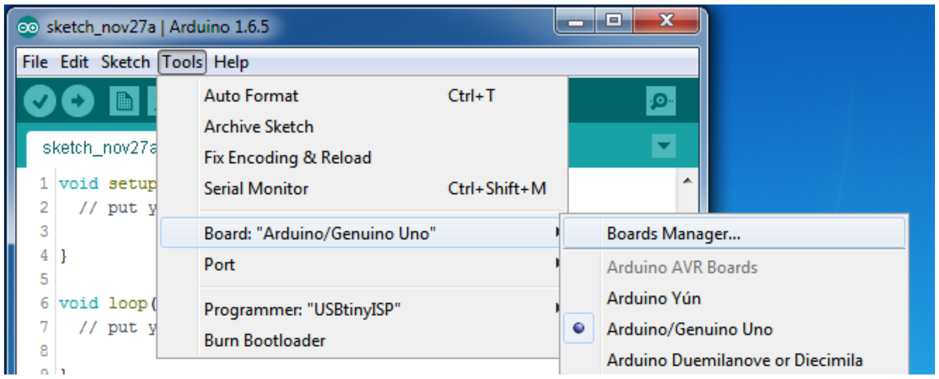
Once done click **OK** to save the new preference settings. Next we will look at installing boards with the Board Manager.

Now continue to the next step to actually install the board support package!

Using with Arduino IDE

The Feather/Metro/Gemma/Trinket M0 and M4 use an ATSAMD21 or ATSAMD51 chip, and 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 and M4, especially devices & sensors that use I2C or SPI.

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



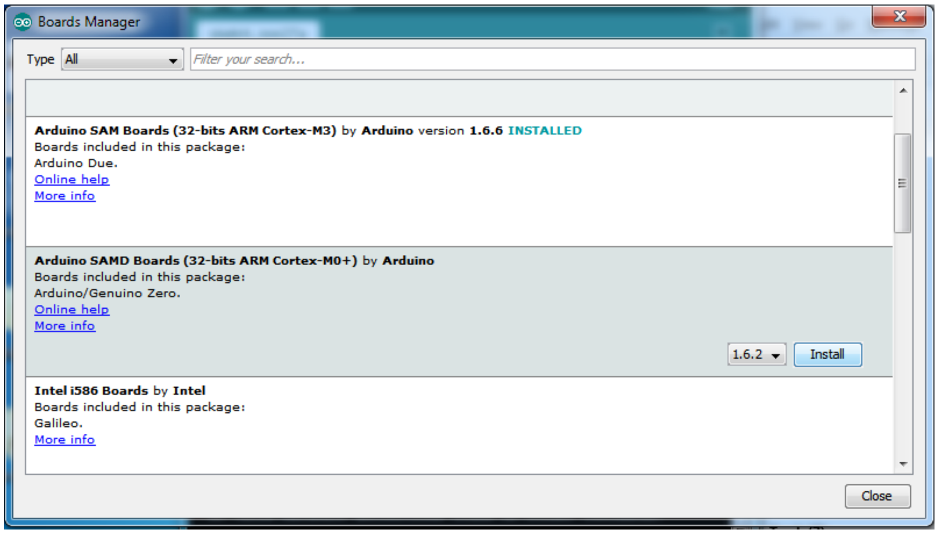
Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select **All**. You will then be able to select and install the boards supplied by the URLs added to the preferences.

Remember you need SETUP the Arduino IDE to support our board packages - see the previous page on how to add adafruit's URL to the preferences

Install SAMD Support

First up, install the latest **Arduino SAMD Boards (**version **1.6.11** or later)

You can type **Arduino SAMD** in the top search bar, then when you see the entry, click **Install**



Arduino SAM Boards (32-bits ARM Cortex-M3) by **Arduino** version **1.6.6** **INSTALLED**
Boards included in this package:
Arduino Due.
Online help
More info

Arduino SAMD Boards (32-bits ARM Cortex-M0+) by **Arduino**
Boards included in this package:
Arduino/Genuino Zero.
Online help
More info

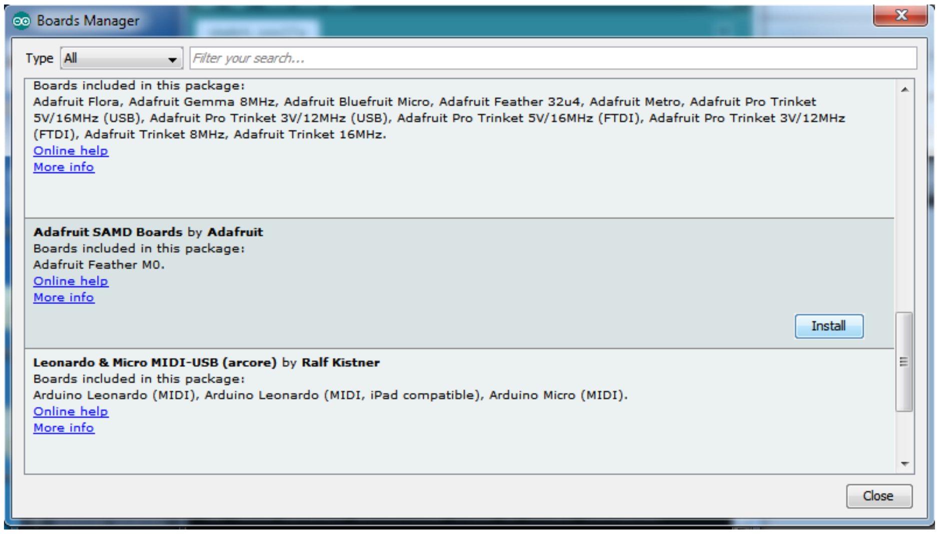
Intel i586 Boards by **Intel**
Boards included in this package:
Galileo.
More info

Install Adafruit SAMD

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

Make sure you have **Type All** selected to the left of the *Filter your search...* box

You can type **Adafruit SAMD** in the top search bar, then when you see the entry, click **Install**



Boards included in this package:
Adafruit Flora, Adafruit Gemma 8MHz, Adafruit Bluefruit Micro, Adafruit Feather 32u4, Adafruit Metro, Adafruit Pro Trinket 5V/16MHz (USB), Adafruit Pro Trinket 3V/12MHz (USB), Adafruit Pro Trinket 5V/16MHz (FTDI), Adafruit Pro Trinket 3V/12MHz (FTDI), Adafruit Trinket 8MHz, Adafruit Trinket 16MHz.
Online help
More info

Adafruit SAMD Boards by **Adafruit**
Boards included in this package:
Adafruit Feather M0.
Online help
More info

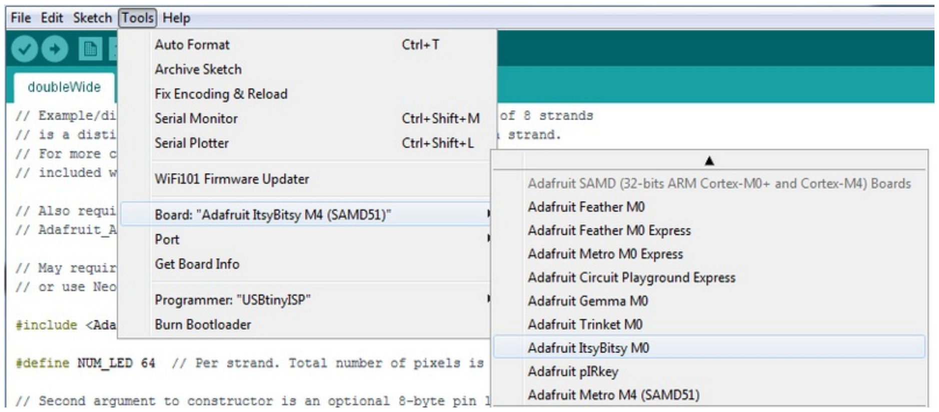
Leonardo & Micro MIDI-USB (arcore) by **Ralf Kistner**
Boards included in this package:
Arduino Leonardo (MIDI), Arduino Leonardo (MIDI, iPad compatible), Arduino Micro (MIDI).
Online help
More info

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.

Select the matching board, the current options are:

- **Feather M0** (for use with any Feather M0 other than the Express)
- **Feather M0 Express**
- **Metro M0 Express**
- **Circuit Playground Express**
- **Gemma M0**
- **Trinket M0**
- **ItsyBitsy M0**
- **Hallowing M0**
- **Crickit M0** (this is for direct programming of the Crickit, which is probably not what you want! For advanced hacking only)
- **Metro M4 Express**
- **ItsyBitsy M4 Express**
- **Feather M4 Express**
- **Trellis M4 Express**
- **Grand Central M4 Express**


File Edit Sketch Tools Help
doubleWide
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Adafruit ItsyBitsy M4 (SAMD51)"
Port
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader
Adafruit SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards
Adafruit Feather M0
Adafruit Feather M0 Express
Adafruit Metro M0 Express
Adafruit Circuit Playground Express
Adafruit Gemma M0
Adafruit Trinket M0
Adafruit ItsyBitsy M0
Adafruit pIRkey
Adafruit Metro M4 (SAMD51)

```
// Example/di                                              of 8 strands
// is a disti                                              strand.
// For more c
// included w

// Also requi
// Adafruit_A

// May requir
// or use Neo

#include <Ada

#define NUM_LED 64  // Per strand. Total number of pixels is

// Second argument to constructor is an optional 8-byte pin l
```

Install Drivers (Windows 7 & 8 Only)

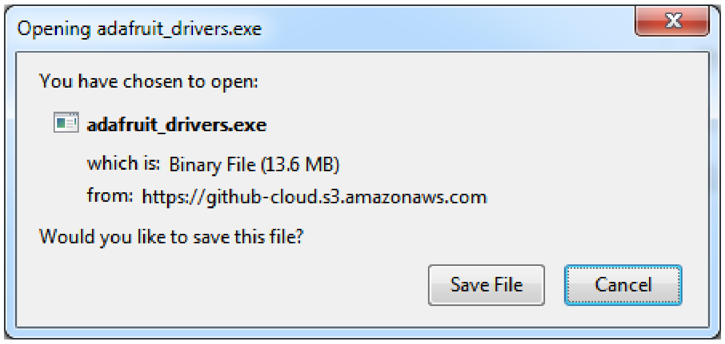
When you plug in the board, you'll need to possibly install a driver

Click below to download our Driver Installer

https://adafru.it/EC0

https://adafru.it/EC0

Download and run the installer



Opening adafruit_drivers.exe

You have chosen to open:

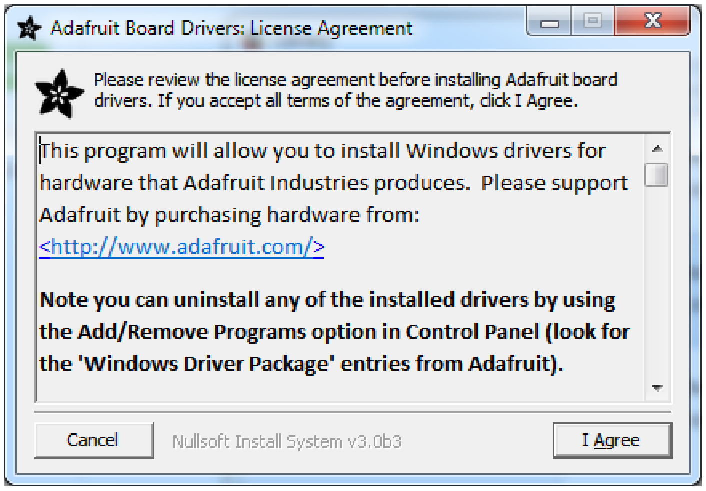
adafruit_drivers.exe

which is: Binary File (13.6 MB)

from: https://github-cloud.s3.amazonaws.com

Would you like to save this file?

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



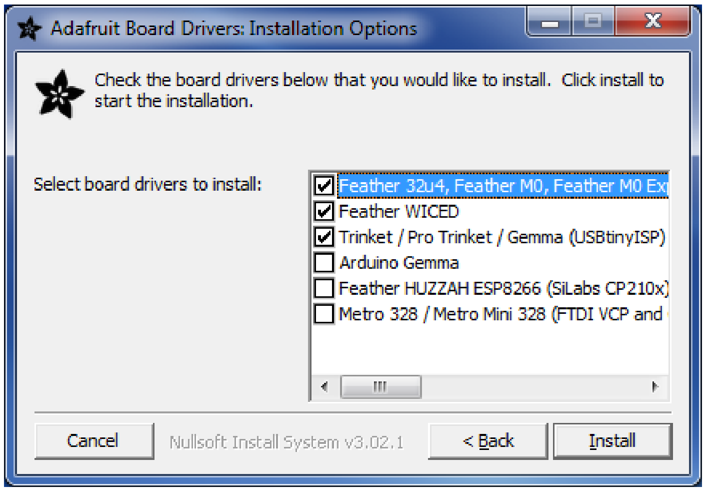
Adafruit Board Drivers: License Agreement

Please review the license agreement before installing Adafruit board drivers. If you accept all terms of the agreement, click I Agree.

This program will allow you to install Windows drivers for hardware that Adafruit Industries produces. Please support Adafruit by purchasing hardware from: <http://www.adafruit.com/>

Note you can uninstall any of the installed drivers by using the Add/Remove Programs option in Control Panel (look for the 'Windows Driver Package' entries from Adafruit).

Select which drivers you want to install, the defaults will set you up with just about every Adafruit board!

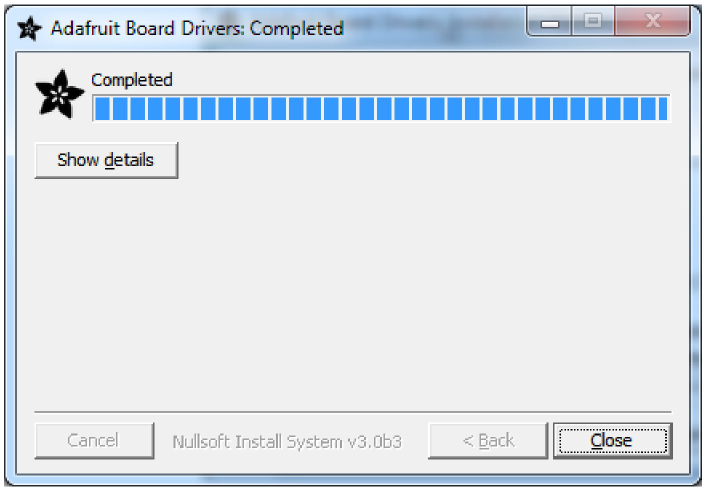

Adafruit Board Drivers: Installation Options

Check the board drivers below that you would like to install. Click install to start the installation.

Select board drivers to install:

- [✓] Feather 32u4, Feather M0, Feather M0 Ex
- [✓] Feather WICED
- [✓] Trinket / Pro Trinket / Gemma (USBtinyISP)
- [ ] Arduino Gemma
- [ ] Feather HUZZAH ESP8266 (SiLabs CP210x)
- [ ] Metro 328 / Metro Mini 328 (FTDI VCP and

Click **Install** to do the installin'

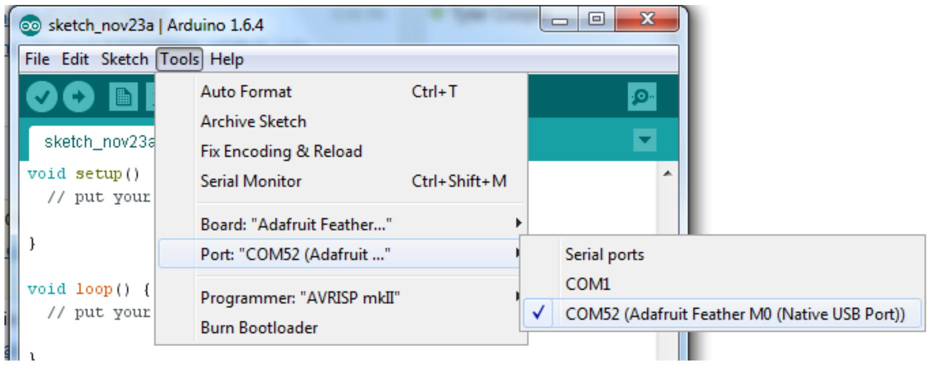

Adafruit Board Drivers: Completed

Completed

Blink

Now you can upload your first blink sketch!

Plug in the M0 or M4 board, 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 drop-down, it'll even be 'indicated' as Trinket/Gemma/Metro/Feather/ItsyBitsy/Trellis!


sketch_nov23a | Arduino 1.6.4

```
void setup() {
  // put your
}

void loop() {
  // put your
```

File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Adafruit Feather..."
Port: "COM52 (Adafruit ..."
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
✓ COM52 (Adafruit Feather M0 (Native USB Port))

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
}
```

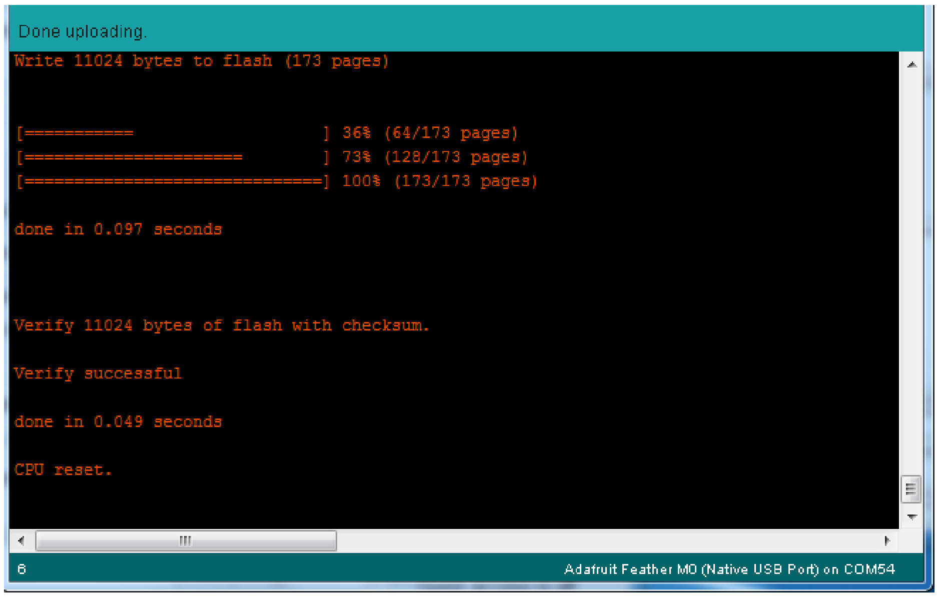
And click upload! That's it, you will be able to see the LED blink rate change as you adapt the **delay()** calls.

If you're using **Trellis M4 Express**, you can go to the next page cause there's no pin 13 LED - so you won't see it blink. Still this is a good thing to test compile and upload!

If you are having issues, make sure you selected the matching Board in the menu that matches the hardware you have in your hand.

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



```
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.
```

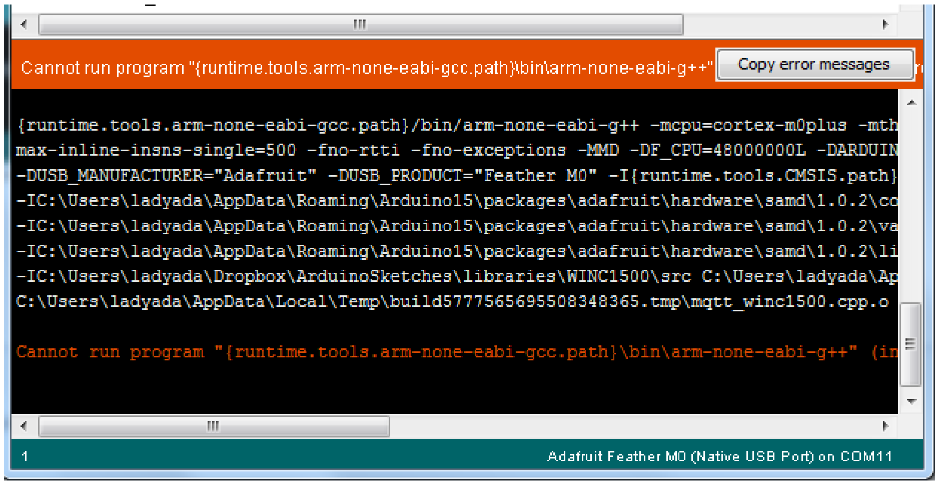
After uploading, you may see a message saying "Disk Not Ejected Properly" about the ...BOOT drive. You can ignore that message: it's an artifact of how the bootloader and uploading work.

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



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

```
{runtime.tools.arm-none-eabi-gcc.path}/bin/arm-none-eabi-g++ -mcpu=cortex-m0plus -mth
max-inline-insns-single=500 -fno-rtti -fno-exceptions -MMD -DF_CPU=48000000L -DARDUIN
-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\co
-IC:\Users\ladyada\AppData\Roaming\Arduino15\packages\adafruit\hardware\samd\1.0.2\va
-IC:\Users\ladyada\AppData\Roaming\Arduino15\packages\adafruit\hardware\samd\1.0.2\li
-IC:\Users\ladyada\Dropbox\ArduinoSketches\libraries\WINC1500\src C:\Users\ladyada\Ap
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
```

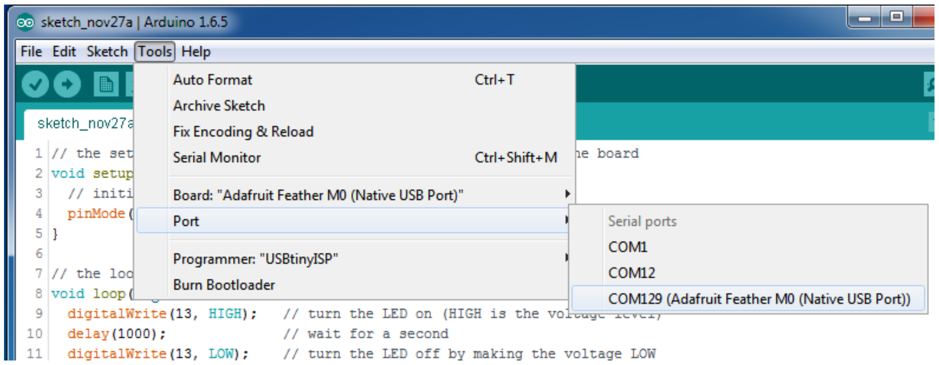
```
1                                    Adafruit Feather M0 (Native USB Port) on COM11
```

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.



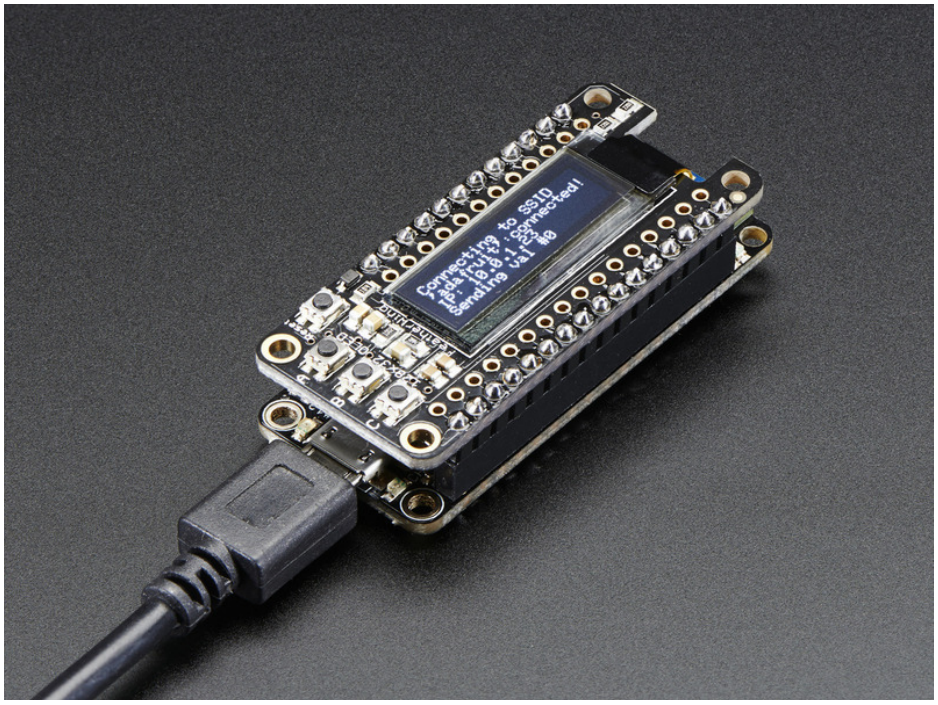
```
// the set                                          he board
void setup
  // initi
  pinMode(
}

// the loo
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
```

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

Follow the steps for installing Adafruit's udev rules on this page. (https://adafru.it/iOE)

Using the WiFi Module

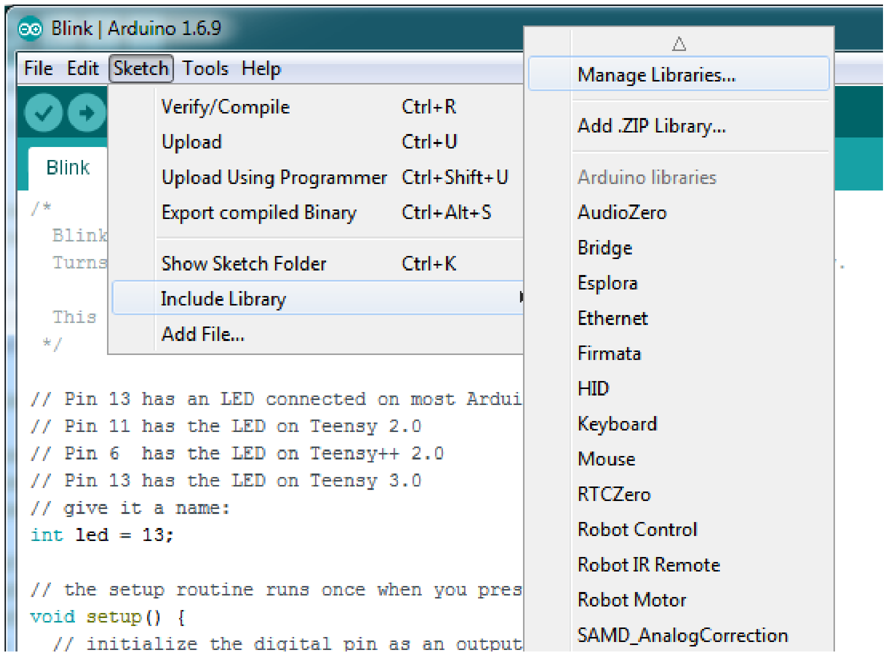


Once you have your Feather working, you probably want to rock out with some Wireless connectivity. Luckily, Atmel & Arduino have written a great library for supporting the WINC1500

Install the Library

We will start by installing the official Arduino WiFi101 library (https://adafru.it/kUF).

We want the latest version so visit the Library Manager



```
/*
  Blink
  Turns
  
  This
 */

// Pin 13 has an LED connected on most Ardui
// Pin 11 has the LED on Teensy 2.0
// Pin 6  has the LED on Teensy++ 2.0
// Pin 13 has the LED on Teensy 3.0
// give it a name:
int led = 13;

// the setup routine runs once when you pres
void setup() {
  // initialize the digital pin as an output
```

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 (https://adafru.it/aYM)!

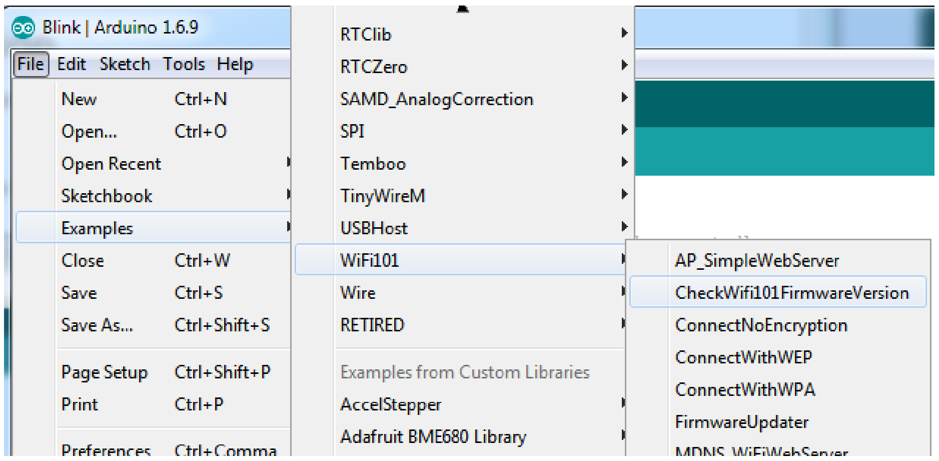
Restart the Arduino IDE.

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.

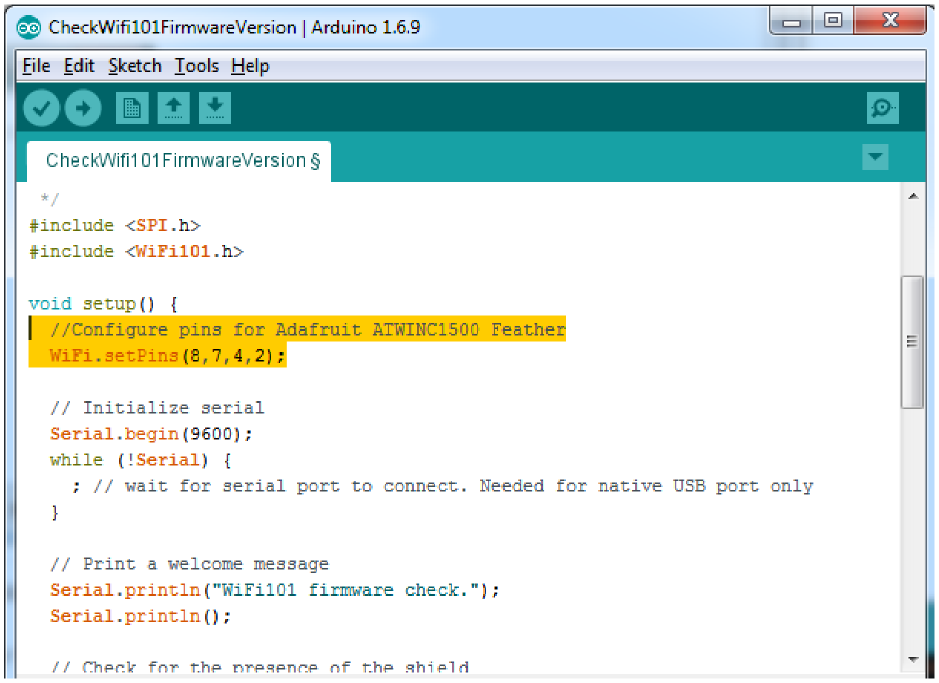
Load up the **WiFi101->CheckWifi101Firmware** sketch



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);
```

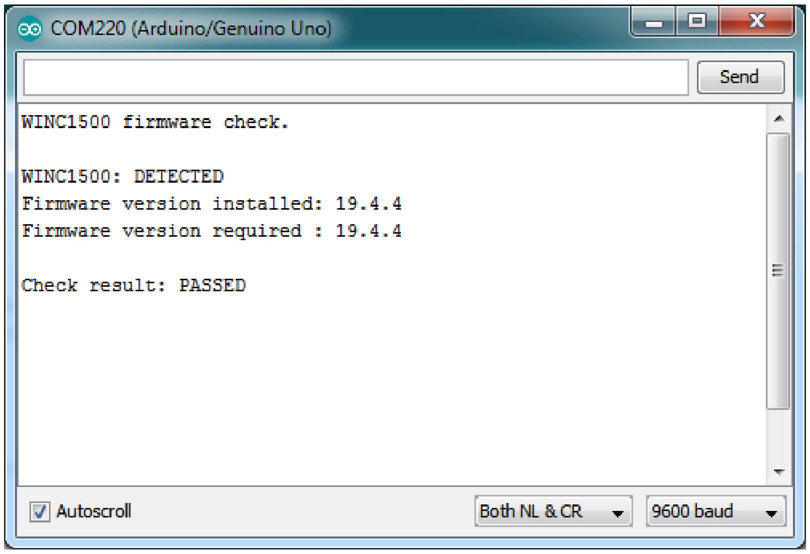
Like so:


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! (https://adafru.it/vfe)



```
WINC1500 firmware check.

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

Check result: PASSED
```

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

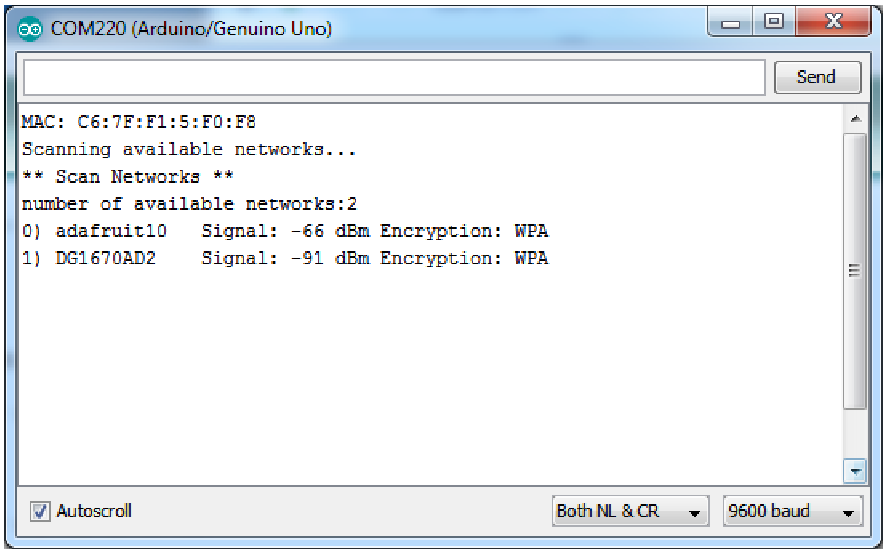
If you get not response, the firmware is either waaay to old, or something is amiss with your wiring!

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()



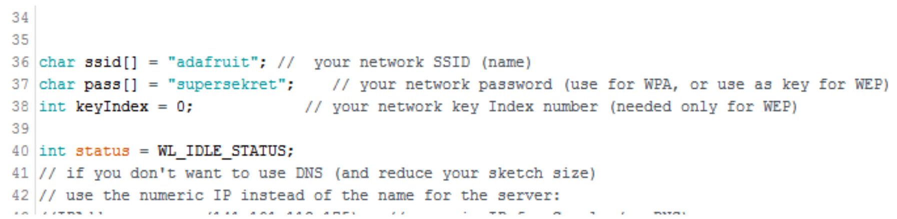
```
MAC: C6:7F:F1:5:F0:F8
Scanning available networks...
** Scan Networks **
number of available networks:2
0) adafruit10   Signal: -66 dBm Encryption: WPA
1) DG1670AD2    Signal: -91 dBm Encryption: WPA
```

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

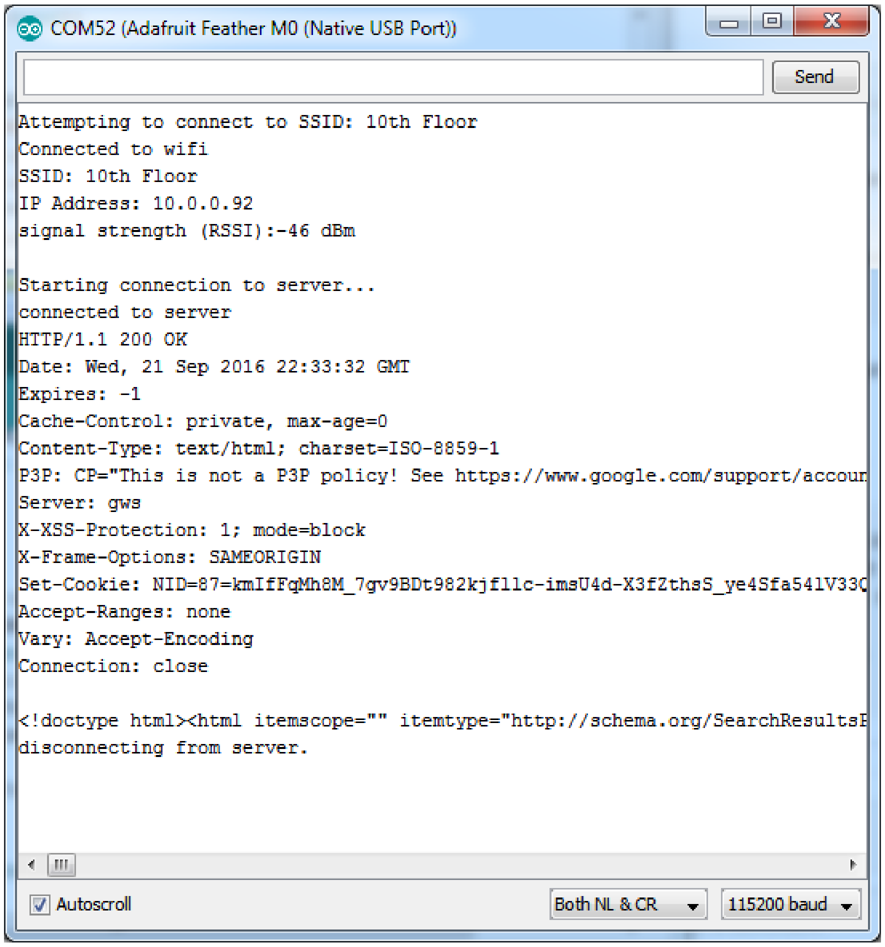


```


char ssid[] = "adafruit"; //  your network SSID (name)
char pass[] = "supersekret";    // your network password (use for WPA, or use as key for WEP)
int keyIndex = 0;            // your network key Index number (needed only for WEP)

int status = WL_IDLE_STATUS;
// if you don't want to use DNS (and reduce your sketch size)
// 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:

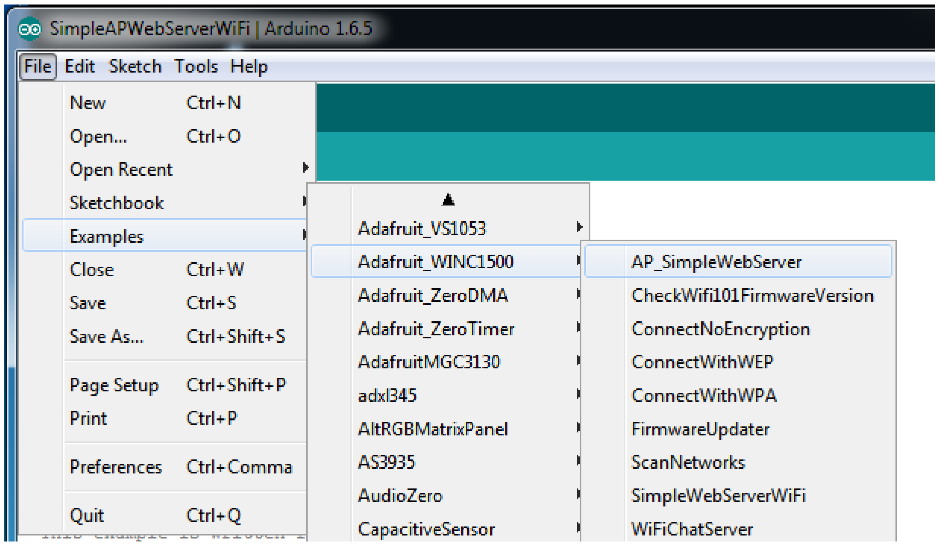


```
Attempting to connect to SSID: 10th Floor
Connected to wifi
SSID: 10th Floor
IP Address: 10.0.0.92
signal strength (RSSI):-46 dBm

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_7gv9BDt982kjfllc-imsU4d-X3fZthsS_ye4Sfa541V33Q
Accept-Ranges: none
Vary: Accept-Encoding
Connection: close

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

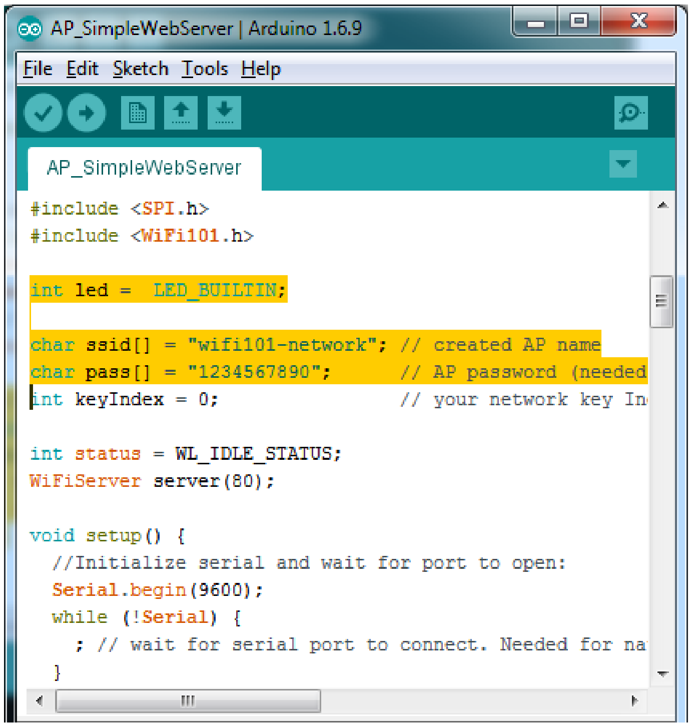
Creating an Access Point + Webserver

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

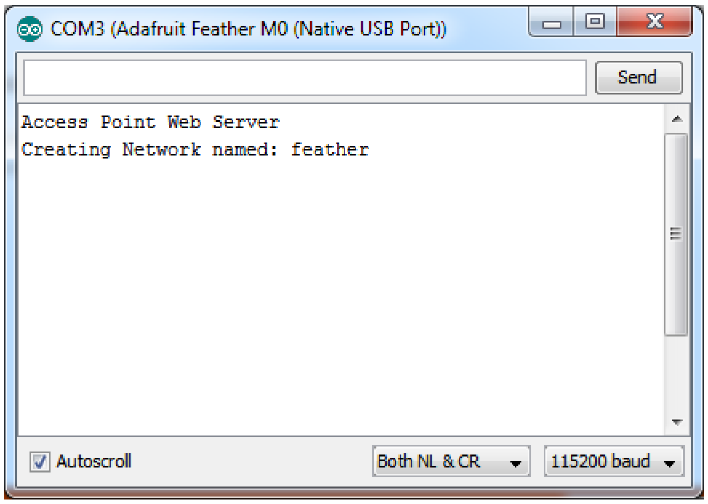


You can change the SSID & LED (`LED_BUILTIN` is #13, the onboard feather LED)


AP_SimpleWebServer | Arduino 1.6.9
File Edit Sketch Tools Help
AP_SimpleWebServer

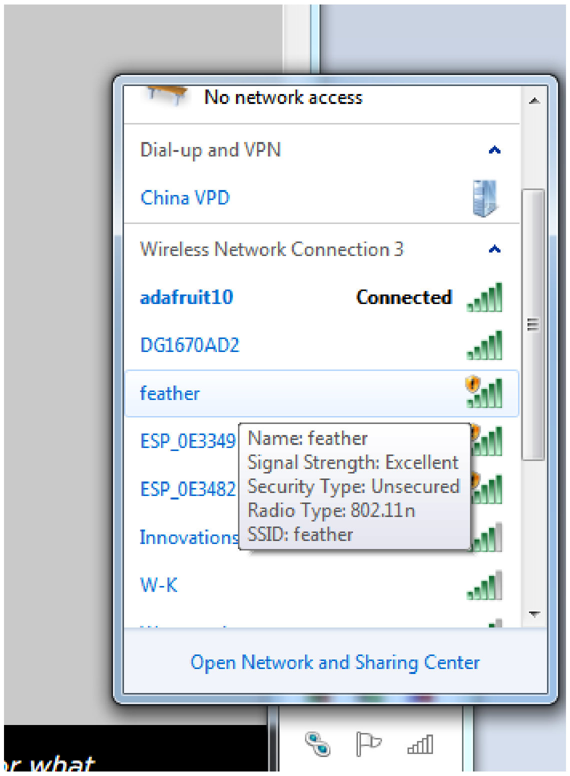
```
#include <SPI.h>
#include <WiFi101.h>

int led =  LED_BUILTIN;

char ssid[] = "wifi101-network"; // created AP name
char pass[] = "1234567890";      // AP password (needed
int keyIndex = 0;                // your network key In

int status = WL_IDLE_STATUS;
WiFiServer server(80);

void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for na
  }
```

Upload and open up the serial console to start the AP



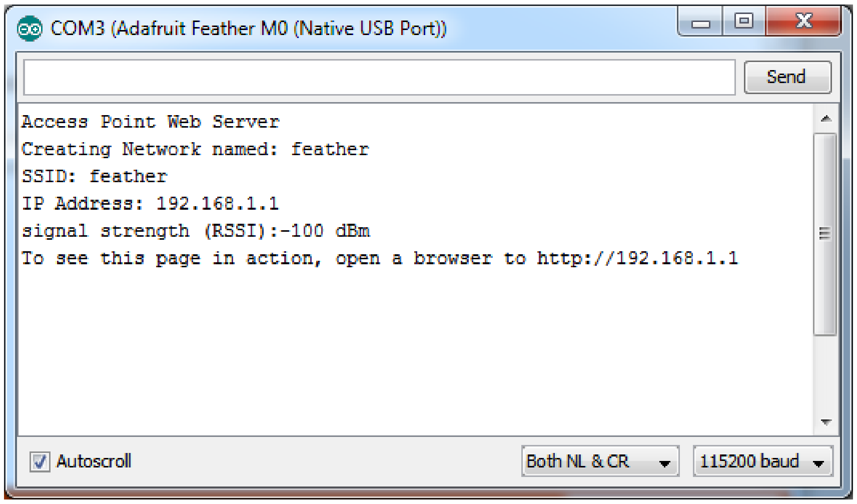
```
Access Point Web Server
Creating Network named: feather
```

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



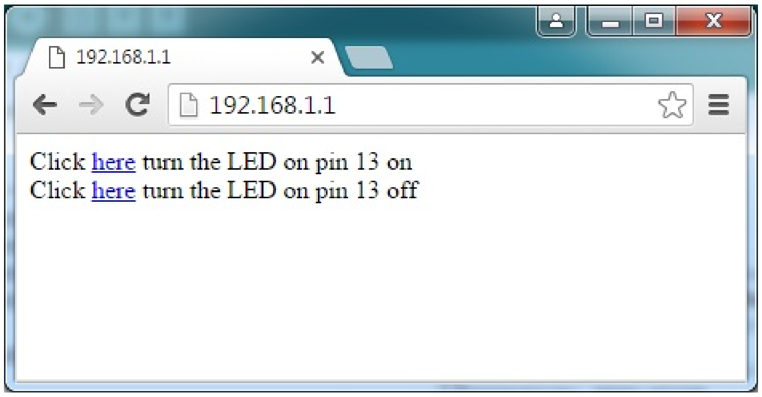
Name: feather
Signal Strength: Excellent
Security Type: Unsecured
Radio Type: 802.11n
SSID: feather

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



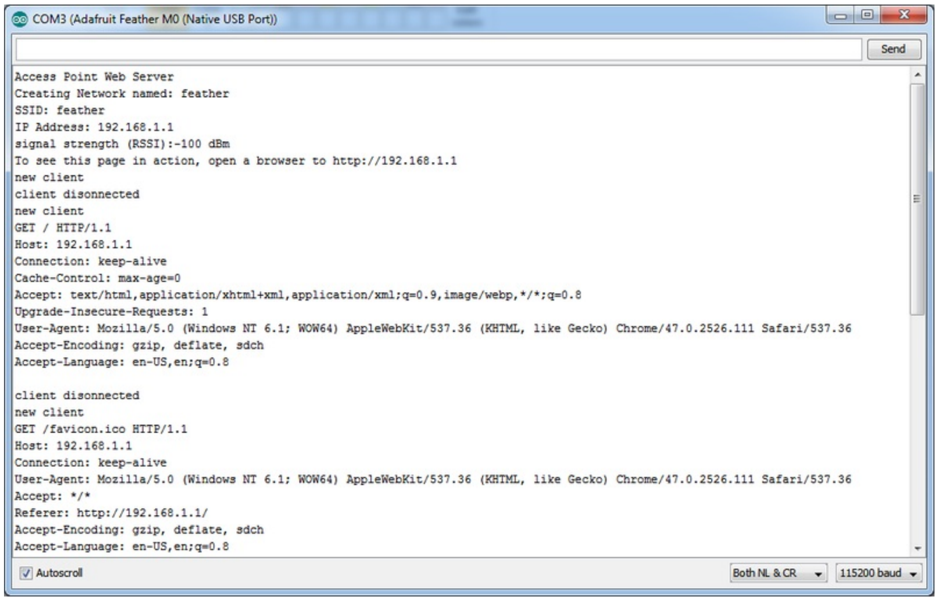
```
Access Point Web Server
Creating Network named: feather
SSID: feather
IP Address: 192.168.1.1
signal strength (RSSI):-100 dBm
To see this page in action, open a browser to http://192.168.1.1
```

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



Click here turn the LED on pin 13 on
Click here turn the LED on pin 13 off

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



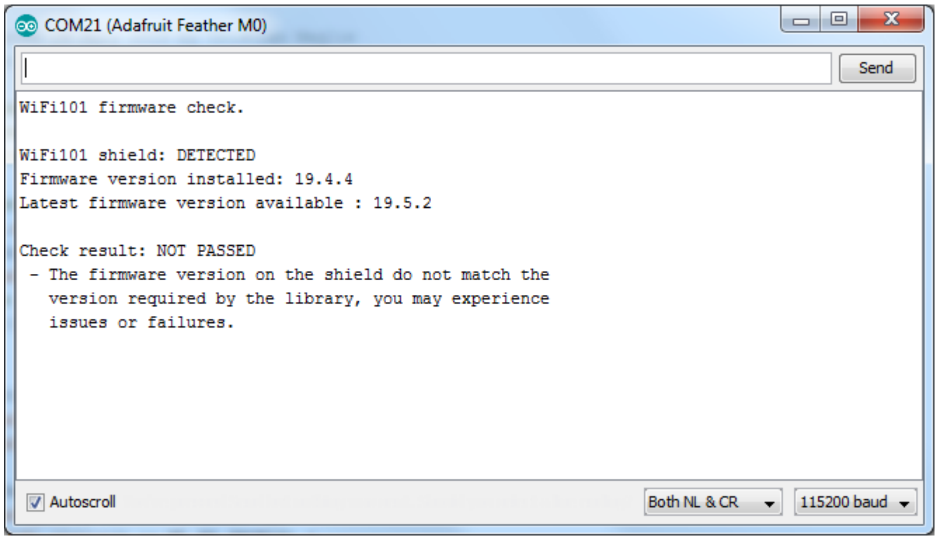
```
Access Point Web Server
Creating Network named: feather
SSID: feather
IP Address: 192.168.1.1
signal strength (RSSI):-100 dBm
To see this page in action, open a browser to http://192.168.1.1
new client
client disconnected
new client
GET / HTTP/1.1
Host: 192.168.1.1
Connection: keep-alive
Cache-Control: 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
```

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

Updating Firmware

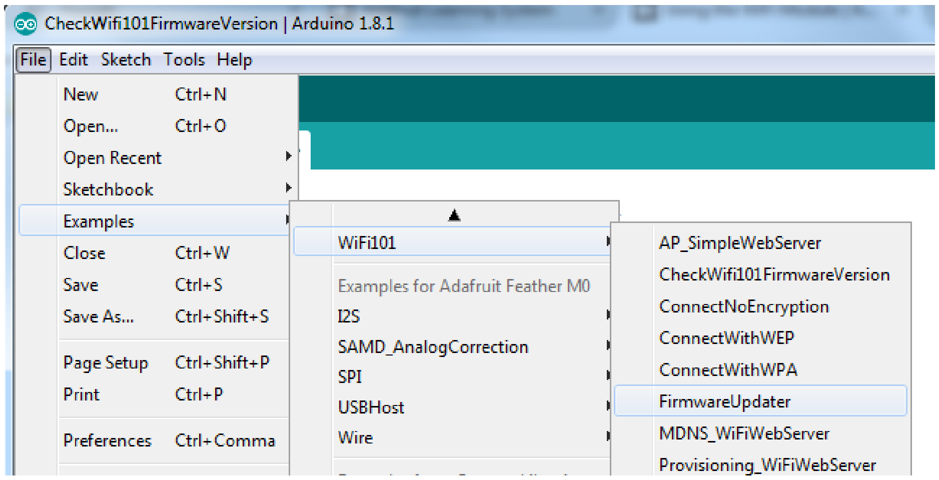
 If you're running 19.5.2 there is no need to update to 19.5.4, despite what the WiFi101 library says it should work just fine and you could brick your device by updating, it is not perfectly foolproof! Thanks :)

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:



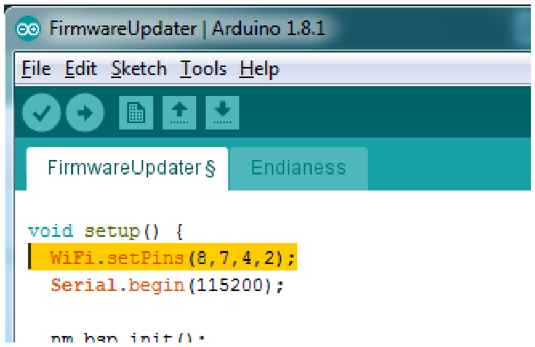
```
WiFi101 firmware check.

WiFi101 shield: DETECTED
Firmware version installed: 19.4.4
Latest firmware version available : 19.5.2

Check result: NOT PASSED
 - The firmware version on the shield do not match the
   version required by the library, you may experience
   issues or failures.
```

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



 If you are using a Feather M0 or WINC1500 breakout, don't forget to update the pins as necessary with setPins()!

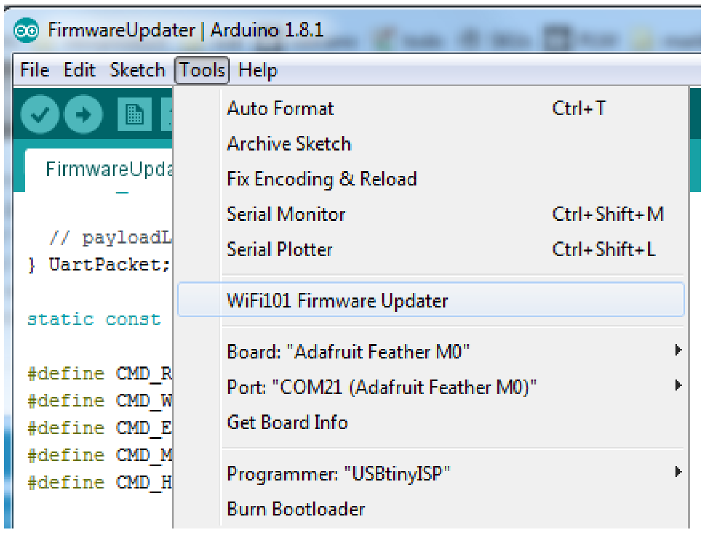
If you are using a WiFi101 or WINC1500 shield, you do not have to add `setPins()` code



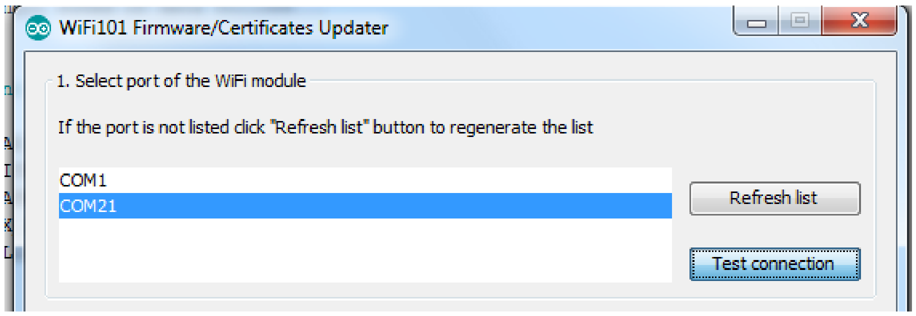
```
void setup() {
  WiFi.setPins(8,7,4,2);
  Serial.begin(115200);
```

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

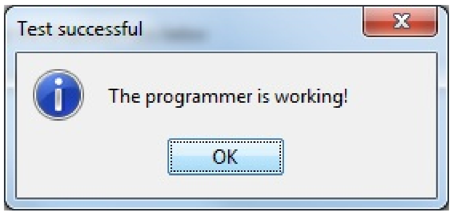
Then select the Updater tool built into the IDE


FirmwareUpdater | Arduino 1.8.1
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Adafruit Feather M0"
Port: "COM21 (Adafruit Feather M0)"
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader

Select the right COM port, and click Test Connection


WiFi101 Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
COM21
Refresh list
Test connection

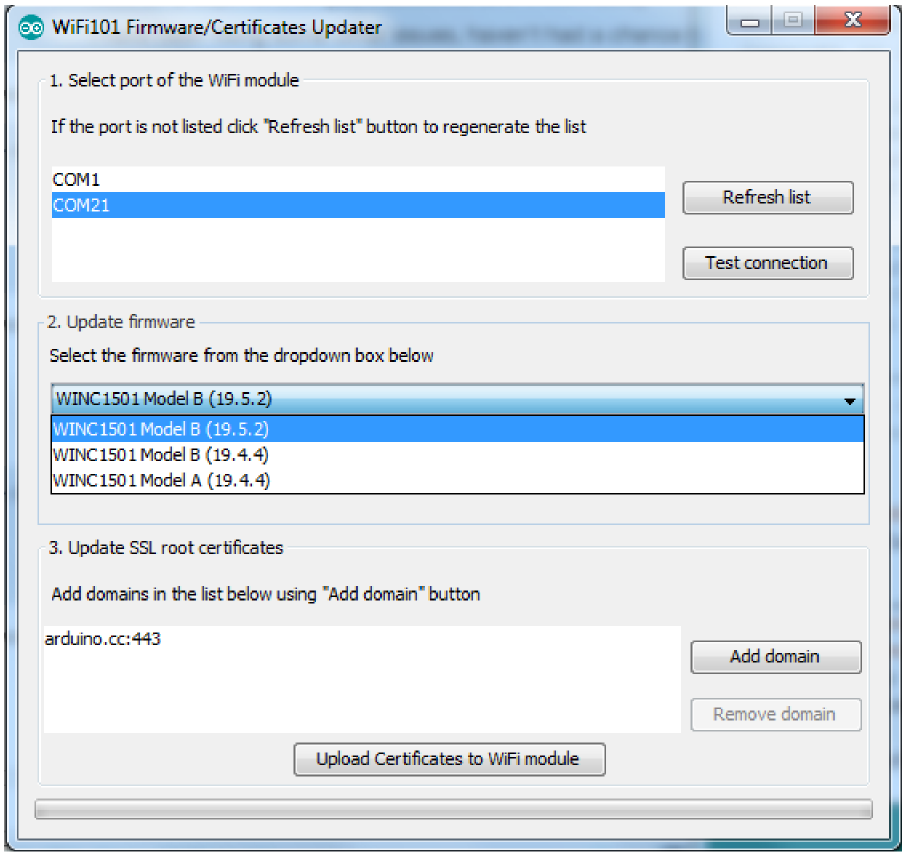
If all is good you'll get a confirmation


Test successful

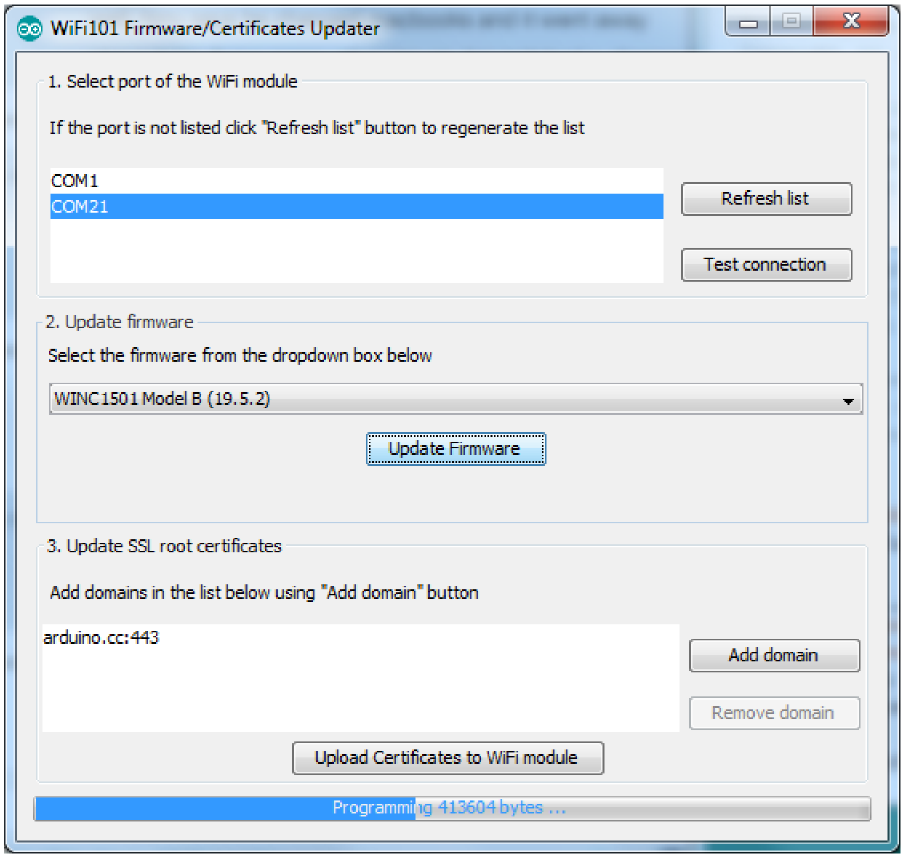
The programmer is working!

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

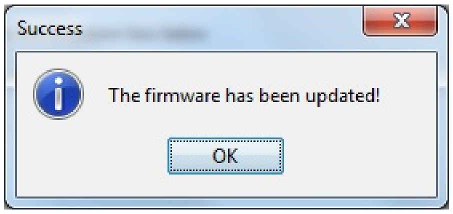
If you don't see the right/matching version you may need to update the IDE


WiFi101 Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
COM21
Refresh list
Test connection
2. Update firmware
Select the firmware from the dropdown box below
WINC1501 Model B (19.5.2)
WINC1501 Model B (19.5.2)
WINC1501 Model B (19.4.4)
WINC1501 Model A (19.4.4)
3. Update SSL root certificates
Add domains in the list below using "Add domain" button
arduino.cc:443
Add domain
Remove domain
Upload Certificates to WiFi module

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


WiFi101 Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
COM21
Refresh list
Test connection
2. Update firmware
Select the firmware from the dropdown box below
WINC1501 Model B (19.5.2)
Update Firmware
3. Update SSL root certificates
Add domains in the list below using "Add domain" button
arduino.cc:443
Add domain
Remove domain
Upload Certificates to WiFi module
Programming 413604 bytes ...

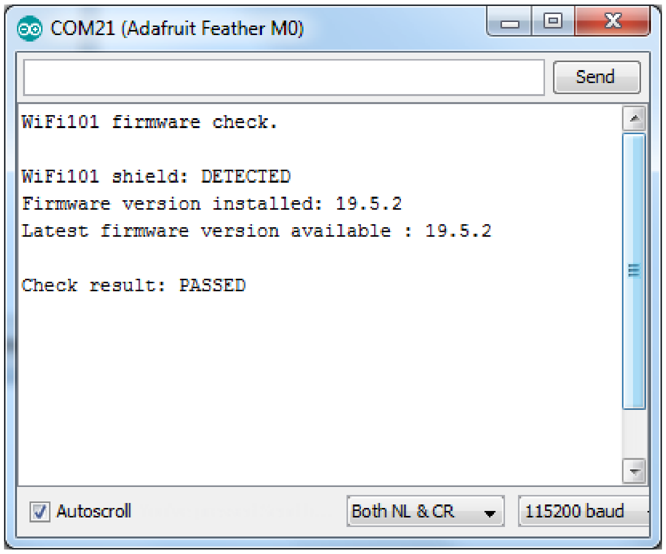
And a minute or two later...



Success

The firmware has been updated!

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



COM21 (Adafruit Feather M0)

```
WiFi101 firmware check.

WiFi101 shield: DETECTED
Firmware version installed: 19.5.2
Latest firmware version available : 19.5.2

Check result: PASSED
```

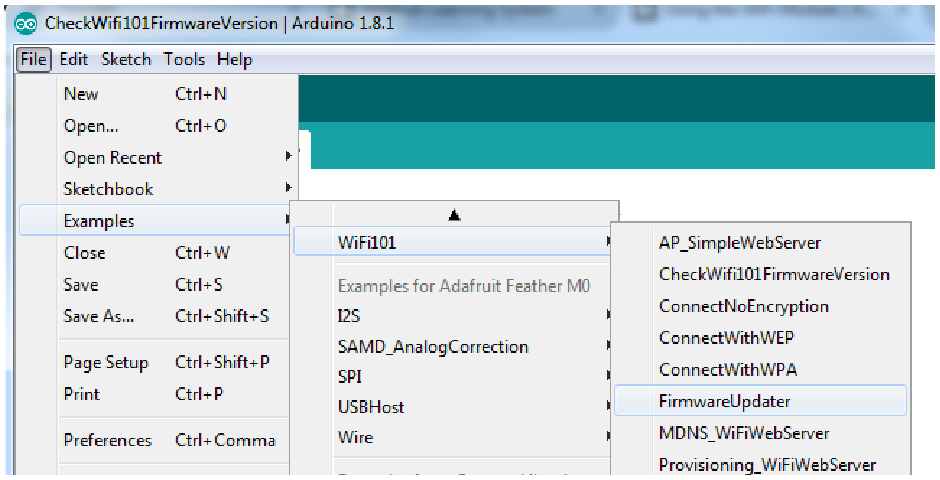
Autoscroll ✓ Both NL & CR 115200 baud

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 isnt included.

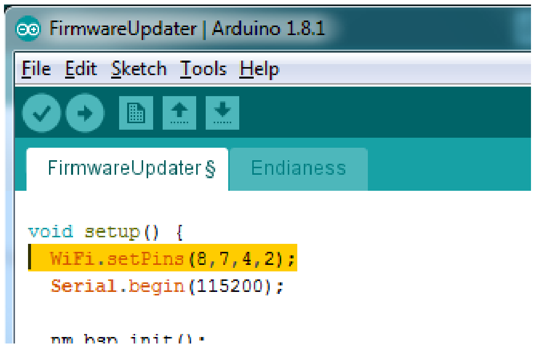
Its quite easy to update the certificates, you'll need to upload some code and run the uploaders but it only has to happen once

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



If you are using a Feather M0 or WINC1500 breakout, don't forget to update the pins as necessary with **setPins**()!

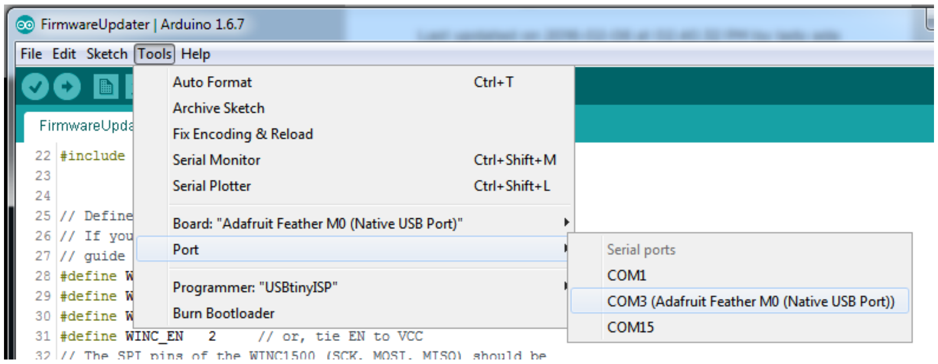
If you are using a WiFi101 or WINC1500 shield, skip this step



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

and upload it!

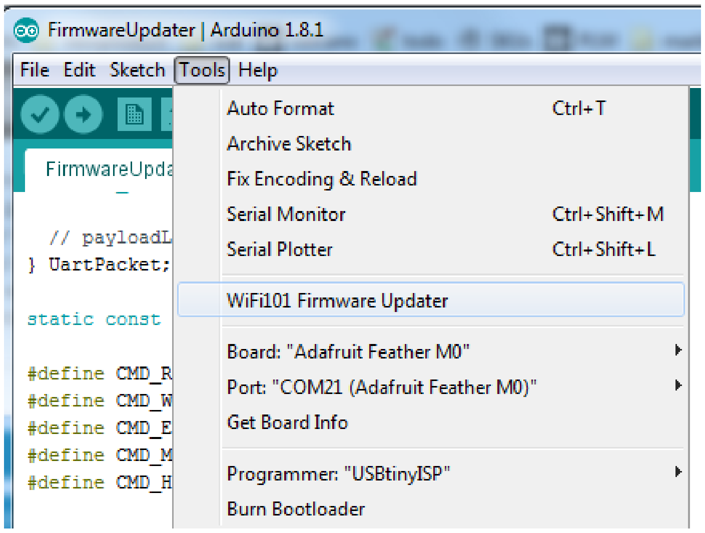
After uploading be sure to note what is the name of the COM or Serial port for the Arduino Zero or Feather...You'll need this for the next step


FirmwareUpdater | Arduino 1.6.7
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
Board: "Adafruit Feather M0 (Native USB Port)"
Port
Programmer: "USBtinyISP"
Burn Bootloader
Serial ports
COM1
COM3 (Adafruit Feather M0 (Native USB Port))
COM15

```
#include
// Define
// If you
// guide
#define W
#define W
#define W
#define WINC_EN   2     // or, tie EN to VCC
// The SPI pins of the WINC1500 (SCK, MOSI, MISO) should be
```

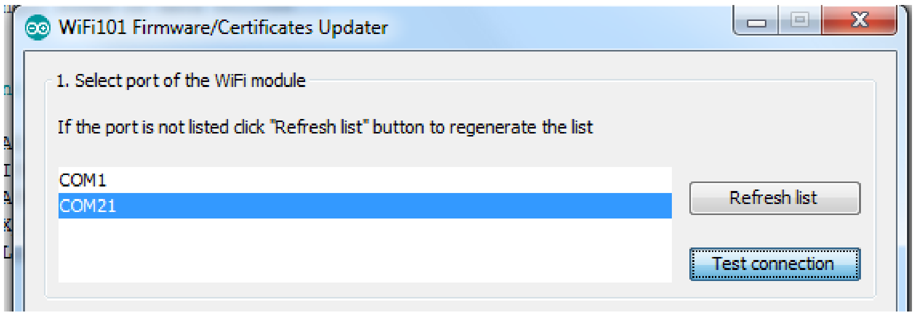
Upload it to your Feather. Make sure the Serial console is not open before or after uploading.

Then select the Updater tool built into the IDE

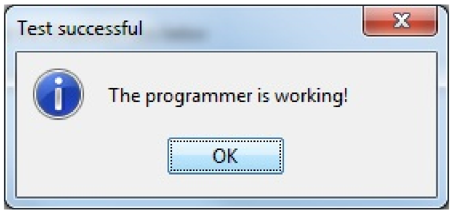

FirmwareUpdater | Arduino 1.8.1
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Adafruit Feather M0"
Port: "COM21 (Adafruit Feather M0)"
Get Board Info
Programmer: "USBtinyISP"
Burn Bootloader

```
  // payloadL
} UartPacket;

static const

#define CMD_R
#define CMD_W
#define CMD_E
#define CMD_M
#define CMD_H
```

Select the right COM port, and click Test Connection

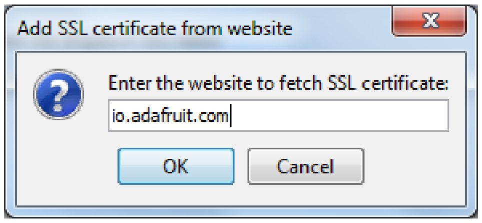

WiFi101 Firmware/Certificates Updater
1. Select port of the WiFi module
If the port is not listed click "Refresh list" button to regenerate the list
COM1
COM21
Refresh list
Test connection

If all is good you'll get a confirmation



The programmer is working!

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

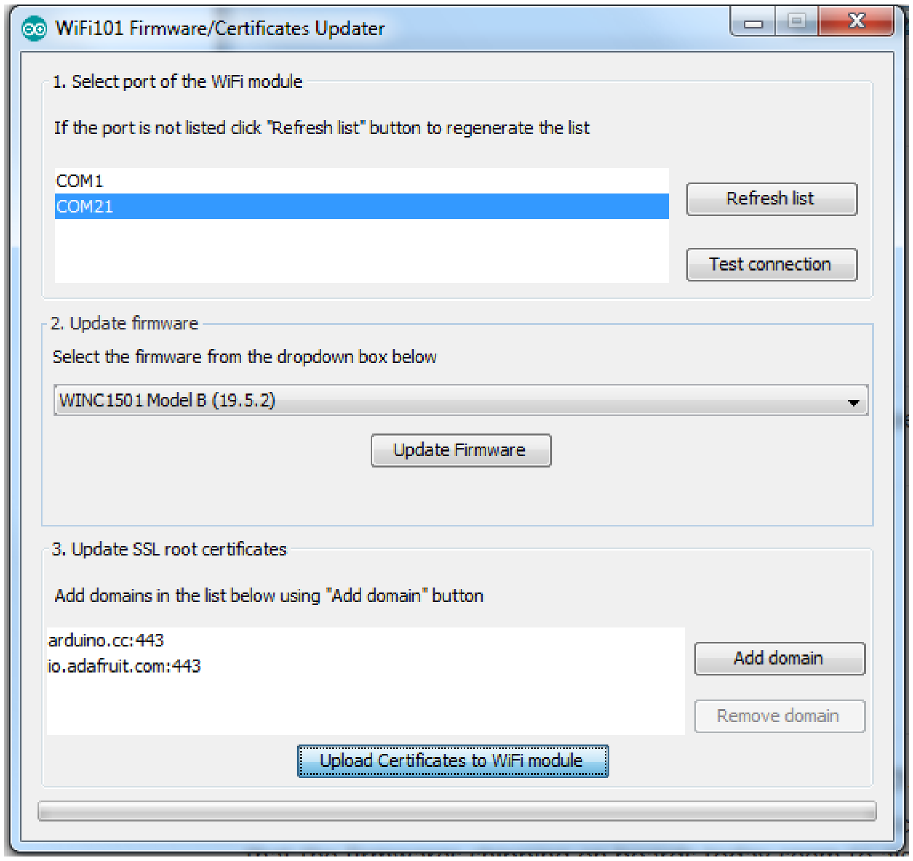


Add SSL certificate from website

Enter the website to fetch SSL certificate:

io.adafruit.com

Then click **Upload Certificates**



WiFi101 Firmware/Certificates Updater

1. Select port of the WiFi module

If the port is not listed click "Refresh list" button to regenerate the list

COM1
COM21

2. Update firmware

Select the firmware from the dropdown box below

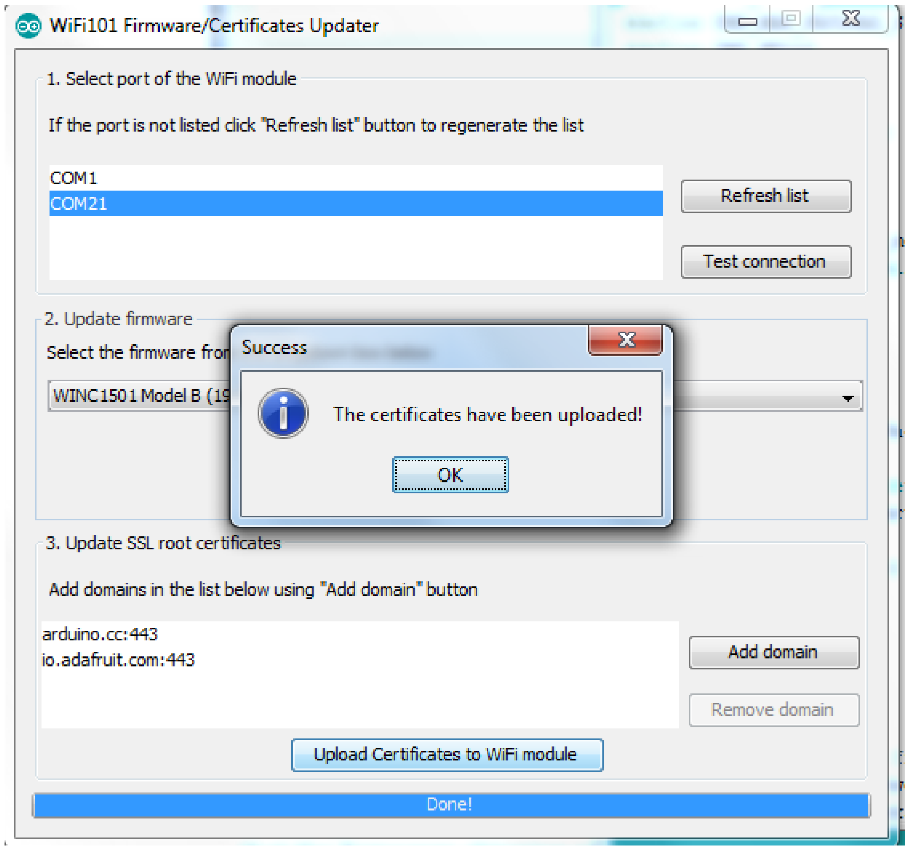
WINC1501 Model B (19.5.2)

3. Update SSL root certificates

Add domains in the list below using "Add domain" button

arduino.cc:443
io.adafruit.com:443

A few moments later...success!


WiFi101 Firmware/Certificates Updater

1. Select port of the WiFi module

If the port is not listed click "Refresh list" button to regenerate the list

COM1
COM21

2. Update firmware

Select the firmware fro

WINC1501 Model B (19

Success

The certificates have been uploaded!

3. Update SSL root certificates

Add domains in the list below using "Add domain" button

arduino.cc:443
io.adafruit.com:443

Done!

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).

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 & M4

The ATSAMD21 and 51 are very nice little chips, but fairly new as Arduino-compatible cores go. **Most** sketches & libraries will work but here's a collection of things we noticed.

The notes below cover a range of Adafruit M0 and M4 boards, but not every rule will apply to every board (e.g. Trinket and Gemma M0 do not have ARef, so you can skip the Analog References note!).

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 on 8-bit AVR chips is the same as the output-selection register.

For M0 & M4 boards, you can't do this anymore! Instead, use:

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

Code written this way still has the benefit of being *backwards compatible with AVR.* You don't need separate versions for the different board types.

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.

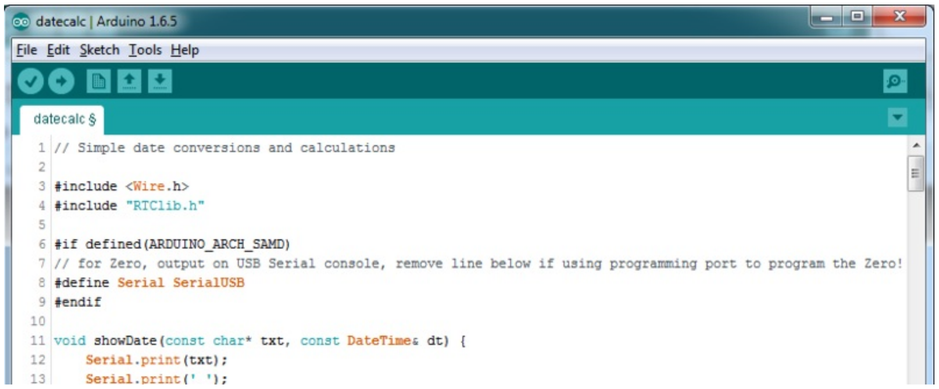
In the Adafruit M0/M4 Core, we fixed it so that **Serial goes to USB so it will automatically work just fine**.

However, on the off chance you are using the official Arduino SAMD core and *not* the Adafruit version (which really, we recommend you use our version because it's been tuned to our boards), and 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:



```
// Simple date conversions and calculations

#include <Wire.h>
#include "RTClib.h"

#if defined(ARDUINO_ARCH_SAMD)
// for Zero, output on USB Serial console, remove line below if using programming port to program the Zero!
#define Serial SerialUSB
#endif

void showDate(const char* txt, const DateTime& dt) {
    Serial.print(txt);
    Serial.print(' ');
```

AnalogWrite / PWM on Feather/Metro M0

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)**

analogWrite() PWM range

On AVR, if you set a pin's PWM with `analogWrite(pin, 255)` it will turn the pin fully HIGH. On the ARM cortex, it will set it to be 255/256 so there will be very slim but still-existing pulses-to-0V. If you need the pin to be fully on, add test code that checks if you are trying to `analogWrite(pin, 255)` and, instead, does a `digitalWrite(pin, HIGH)`

analogWrite() DAC on A0

If you are trying to use `analogWrite()` to control the DAC output on **A0**, make sure you do **not** have a line that sets the pin to output. ***Remove***: `pinMode(A0, OUTPUT)`.

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_AR
 #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 AVRs, clicking **reset** while plugged into USB will launch the bootloader manually, the bootloader will time out after a few seconds. For the M0/M4, 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 (https://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 (https://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

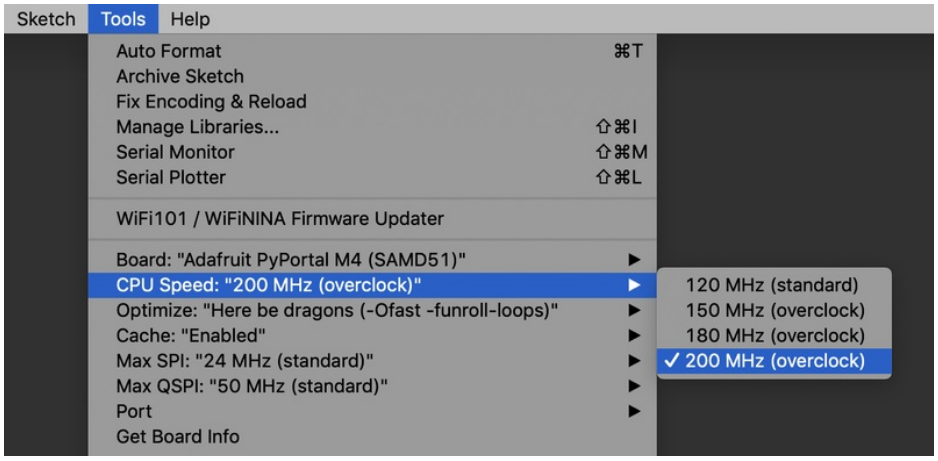
Pretty-Printing out registers

There's *a lot* of registers on the SAMD21, and you often are going through ASF or another framework to get to them. So having a way to see exactly what's going on is handy. This library from drewfish will help a ton!

https://github.com/drewfish/arduino-ZeroRegs (https://adafru.it/Bet)

M4 Performance Options

As of version 1.4.0 of the *Adafruit SAMD Boards* package in the Arduino Boards Manager, some options are available to wring extra performance out of M4-based devices. These are in the *Tools* menu.



All of these performance tweaks involve a degree of uncertainty. There's *no guarantee* of improved performance in any given project, and *some may even be detrimental,* failing to work in part or in whole. If you encounter trouble, **select the default performance settings** and re-upload.

Here's what you get and some issues you might encounter...

CPU Speed (overclocking)

This option lets you adjust the microcontroller core clock...the speed at which it processes instructions...beyond the official datasheet specifications.

Manufacturers often rate speeds conservatively because such devices are marketed for harsh industrial environments...if a system crashes, someone could lose a limb or worse. But most creative tasks are less critical and operate in more comfortable settings, and we can push things a bit if we want more speed.

There is a small but nonzero chance of code **locking up** or **failing to run** entirely. If this happens, try **dialing back the speed by one notch and re-upload**, see if it's more stable.

Much more likely, **some code or libraries may not play well** with the nonstandard CPU speed. For example, currently the NeoPixel library assumes a 120 MHz CPU speed and won't issue the correct data at other settings (this will be worked on). Other libraries may exhibit similar problems, usually anything that strictly depends on CPU timing...you might encounter problems with audio- or servo-related code depending how it's written. **If you encounter such code or libraries, set the CPU speed to the default 120 MHz and re-upload.**

Optimize

There's usually more than one way to solve a problem, some more resource-intensive than others. Since Arduino got its start on resource-limited AVR microcontrollers, the C++ compiler has always aimed for the **smallest compiled program size**. The "Optimize" menu gives some choices for the compiler to take different and often faster approaches, at the expense of slightly larger program size...with the huge flash memory capacity of M4 devices, that's rarely a problem now.

The "**Small**" setting will compile your code like it always has in the past, aiming for the smallest compiled program size.

The "**Fast**" setting invokes various speed optimizations. The resulting program should produce the same results, is slightly larger, and usually (but not always) noticably faster. It's worth a shot!

"**Here be dragons**" invokes some more intensive optimizations...code will be larger still, faster still, but there's a possibility these optimizations could cause unexpected behaviors. *Some code may not work the same as before.* Hence the name. Maybe you'll discover treasure here, or maybe you'll sail right off the edge of the world.

Most code and libraries will continue to function regardless of the optimizer settings. If you do encounter problems, **dial it back one notch and re-upload**.

Cache

This option allows a small collection of instructions and data to be accessed more quickly than from flash memory, boosting performance. It's enabled by default and should work fine with all code and libraries. But if you encounter some esoteric situation, the cache can be disabled, then recompile and upload.

Max SPI and Max QSPI

These should probably be left at their defaults. They're present mostly for our own experiments and can cause **serious headaches**.

Max SPI determines the clock source for the M4's SPI peripherals. Under normal circumstances this allows transfers up to 24 MHz, and should usually be left at that setting. But...if you're using write-only SPI devices (such as TFT or OLED displays), this option lets you drive them faster (we've successfully used 60 MHz with some TFT screens). The caveat is, if using *any* read/write devices (such as an SD card), *this will not work at all...*SPI reads *absolutely* max out at the default 24 MHz setting, and anything else will fail. **Write = OK. Read = FAIL.** This is true *even if your code is using a lower bitrate setting...*just having the different clock source prevents SPI reads.

Max QSPI does similarly for the extra flash storage on M4 "Express" boards. *Very few* Arduino sketches access this storage at all, let alone in a bandwidth-constrained context, so this will benefit next to nobody. Additionally, due to the way clock dividers are selected, this will only provide some benefit when certain "CPU Speed" settings are active. Our PyPortal Animated GIF Display (https://adafru.it/EkO) runs marginally better with it, if using the QSPI flash.

Enabling the Buck Converter on some M4 Boards

If you want to reduce power draw, some of our boards have an inductor so you can use the 1.8V buck converter instead of the built in linear regulator. If the board does have an inductor (see the schematic) you can add the line `SUPC->VREG.bit.SEL = 1;` to your code to switch to it. Note it will make ADC/DAC reads a bit noisier so we don't use it by default. You'll save ~4mA (https://adafru.it/F0H).

Downloads

Datasheets & Files

- Atmel Software Programming guide for WINC1500 (https://adafru.it/ldD) - this is for the underlying ASF codebase that is 'wrapped' in Adafruit_WINC1500 but its still very handy reference
- ATSAMD21 Datasheet (https://adafru.it/ldE) - Its long, but its a good read
- EagleCAD PCB Files on GitHub (https://adafru.it/oeK)

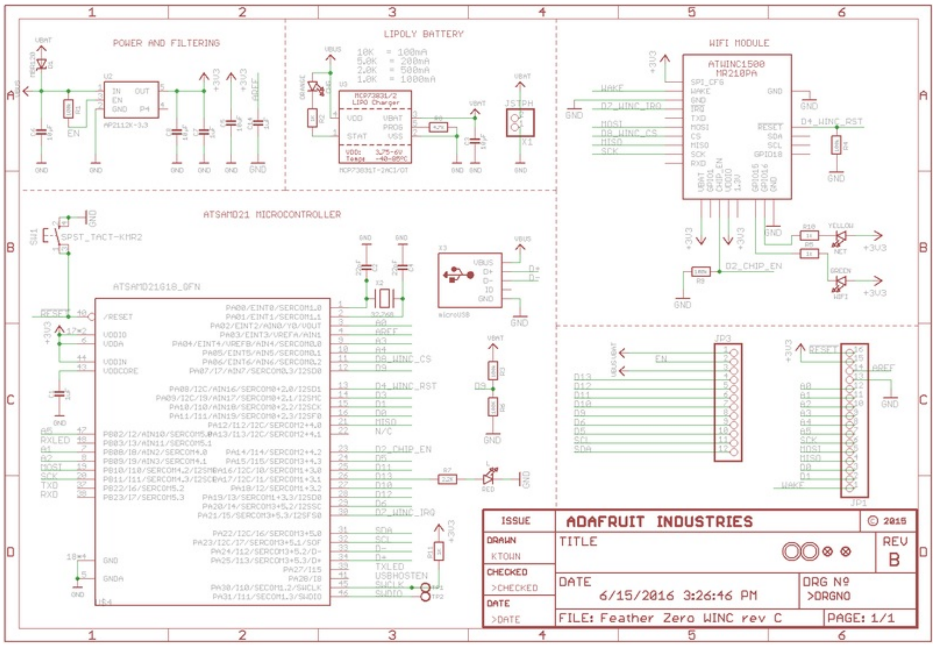
https://adafru.it/z4e

https://adafru.it/z4e

Note AREF in the diagram should be marked PA03 not PA02

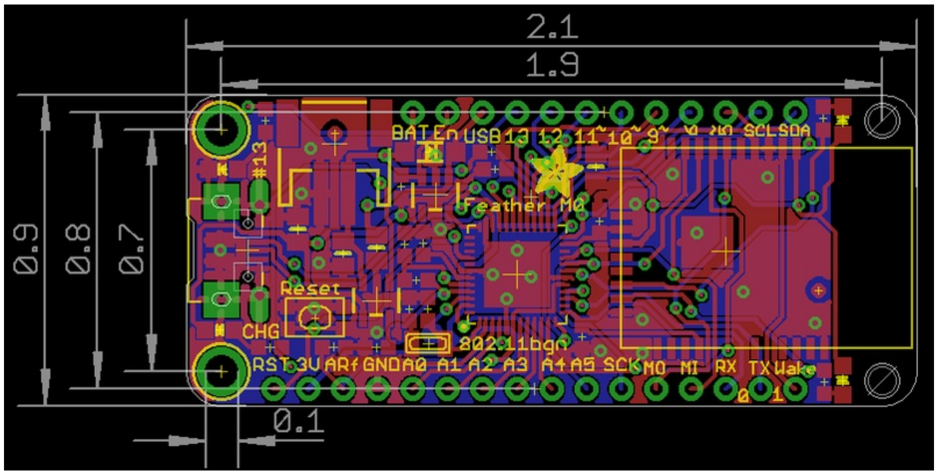
Schematic

Click to enlarge



Fabrication Print

Dimensions in inches


2.1
1.9
0.9
0.8
0.7
0.1
BAT En USB 13 12 11~ 10~ 9~ ~6 ~5 SCL SDA
Feather M0
#13
Reset
CHG
802.11bgn
RST 3V ARf GND A0 A1 A2 A3 A4 A5 SCK MO MI RX TX Wake
0 1

Feather HELP!

Even though this FAQ is labeled for Feather, the questions apply to ItsyBitsy's as well!

My ItsyBitsy/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 Itsy/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!

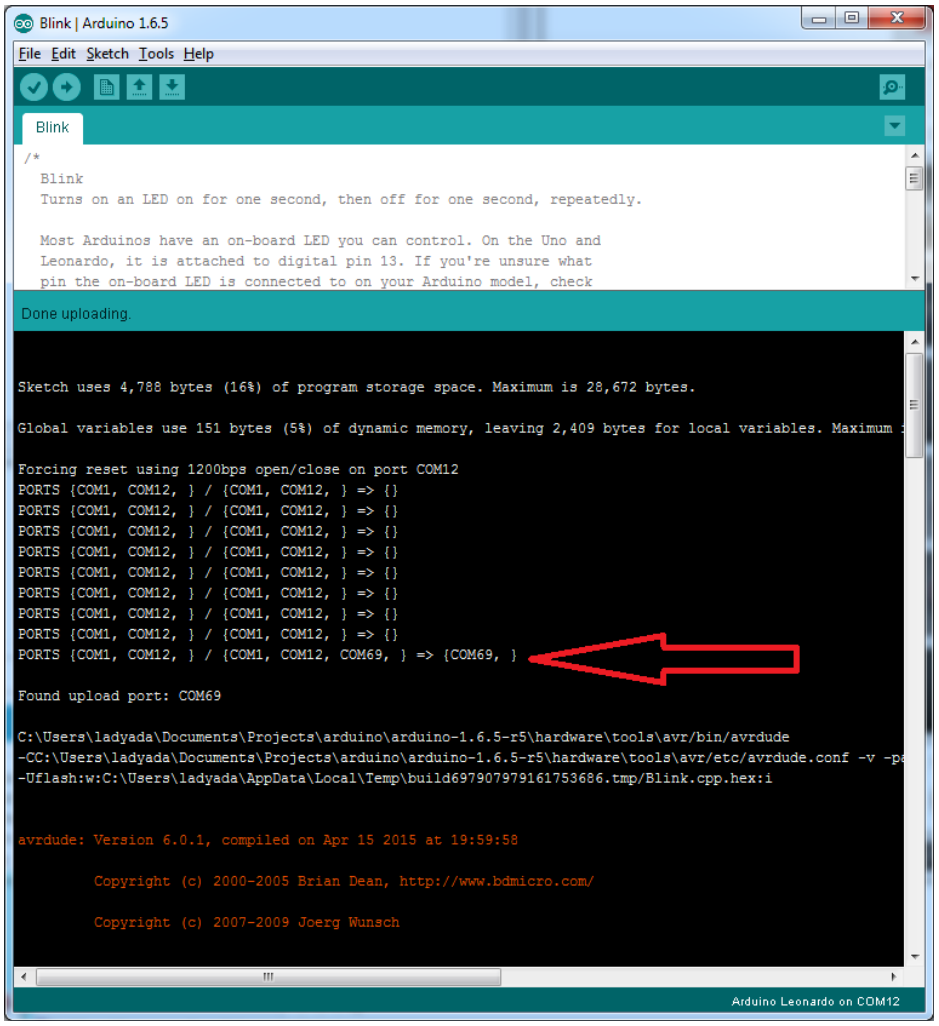
Use only a cable that you **know** is for data syncing

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 Itsy/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 board

1. Turn on **verbose upload** in the Arduino IDE preferences
2. Plug in Itsy or 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, Feather M0, Itsy 32u4 or M0 *(physically 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 board will show up as the Bootloader COM/Serial port
9. The IDE should see the bootloader COM/Serial port and upload properly



```
/*
  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 -pa
-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 Itsy/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

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

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

Theres **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).

I attached some wings to my Feather and now I can't read the battery voltage!

Make sure your Wing doesn't use pin #9 which is the analog sense for the lipo battery!

The yellow LED Is flickering on my Feather, but no battery is plugged in, why is that?

The charge LED is automatically driven by the Lipoly charger circuit. It will try to detect a battery and is expecting one to be attached. If there isn't one it may flicker once in a while when you use power because it's trying to charge a (non-existant) battery.

It's not harmful, and its totally normal!