Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

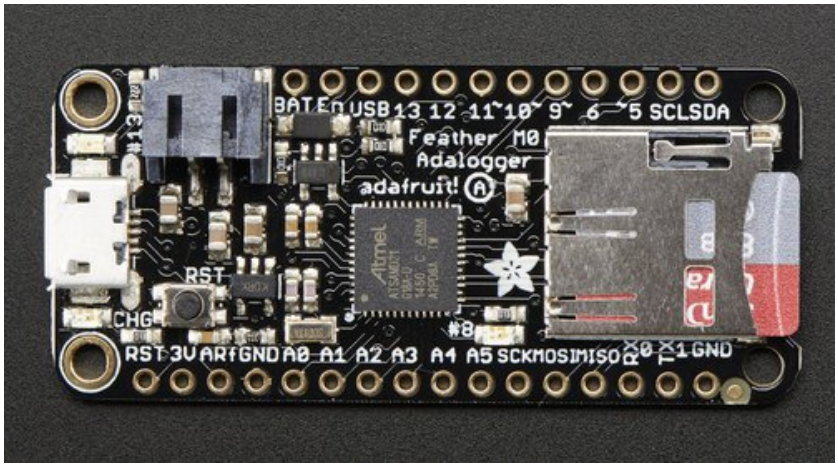
We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

Adafruit Feather M0 Adalogger

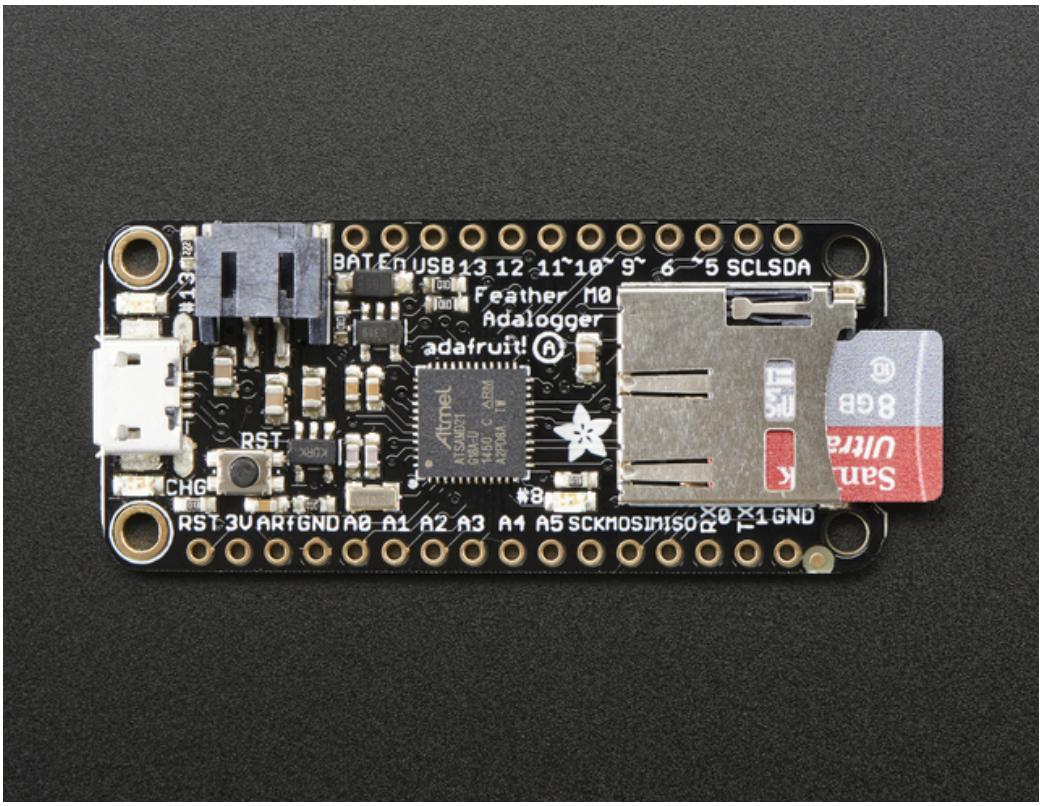
Created by lady ada



Last updated on 2017-02-06 08:03:11 PM UTC

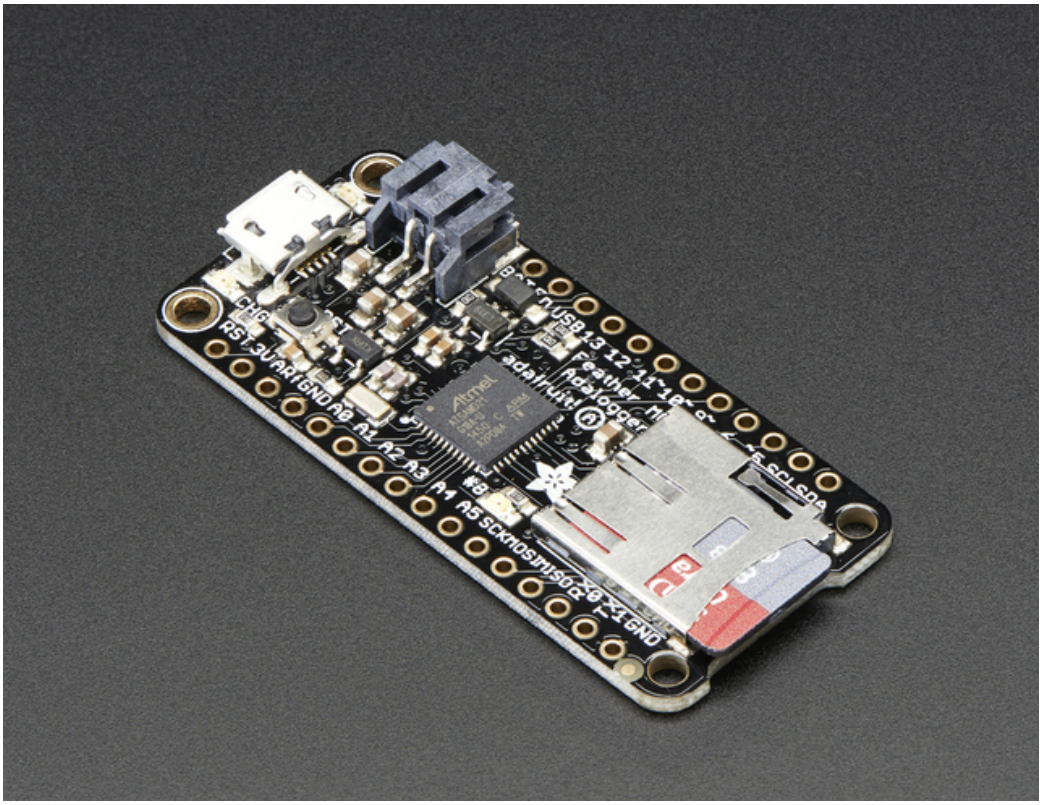
Guide Contents

Overview

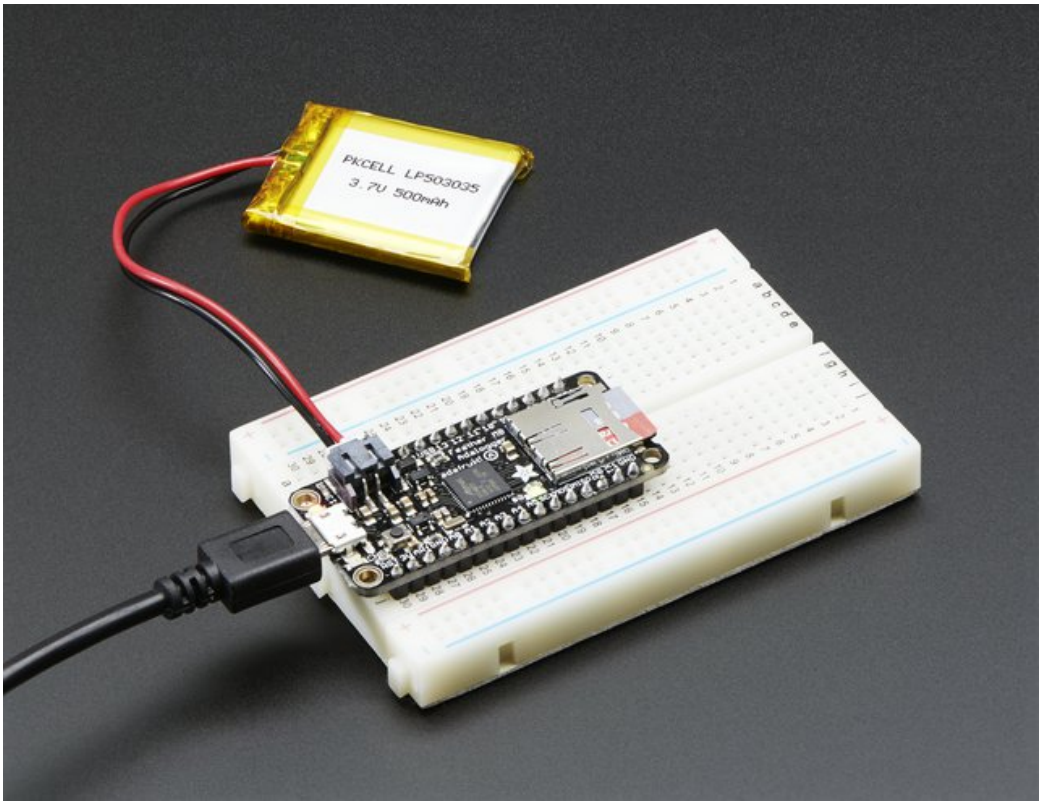


Feather is the new development board from Adafruit, and like it's namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores.

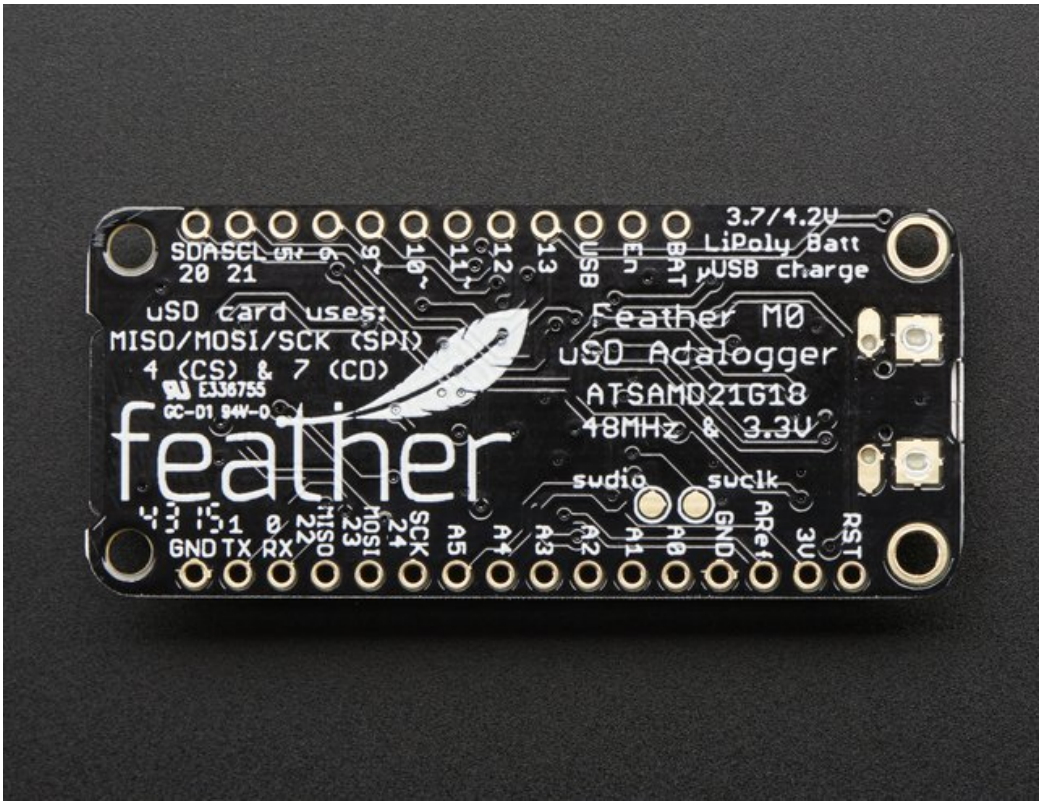
This is the **Adafruit Feather M0 Adalogger** - our take on an 'all-in-one' Cortex M0 datalogger (or data-reader) with built in USB and battery charging. Its an Adafruit Feather M0 with a microSD holder ready to rock! [We have other boards in the Feather family, check'em out here](http://adafru.it/jAQ) (http://adafru.it/jAQ)



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.



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

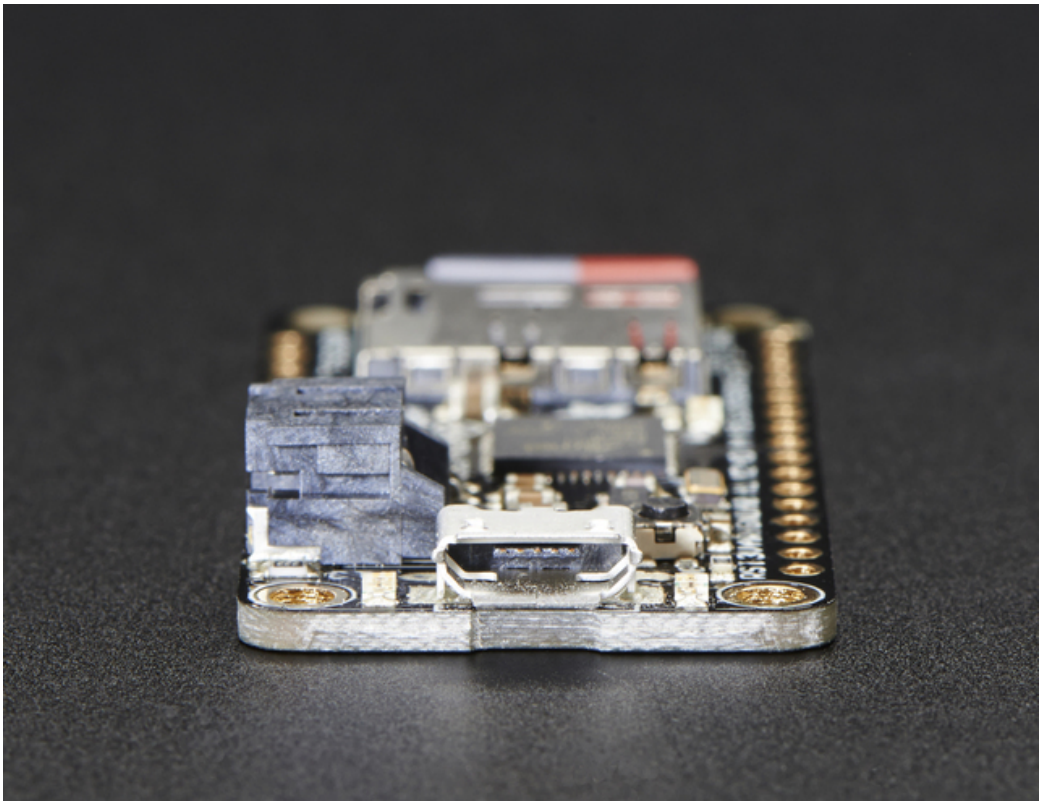


Here's some handy specs! Like all Feather M0's you get:

- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather - 5.3 grams
- ATSAMD21G18 @ 48MHz with 3.3V logic/power
- 256KB of FLASH + 32KB of RAM
- 3.3V regulator with 500mA peak current output
- 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
- Built in 100mA lipoly charger with charging status indicator LED
- Pin #13 red LED for general purpose blinking
- Power/enable pin
- 4 mounting holes
- Reset button

The **Feather M0 Adalogger** uses the extra space left over to add MicroSD + a green LED:

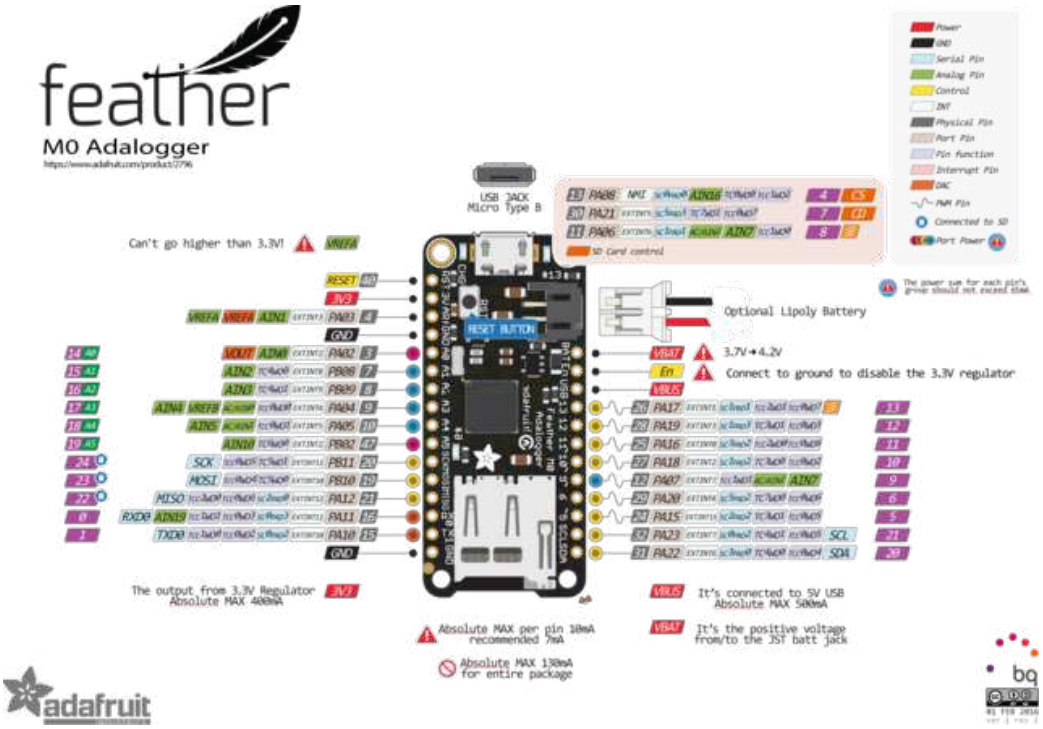
- Pin #8 green LED for your blinking pleasure
- MicroSD card holder for adding as much storage as you could possibly want, for reading or writing.



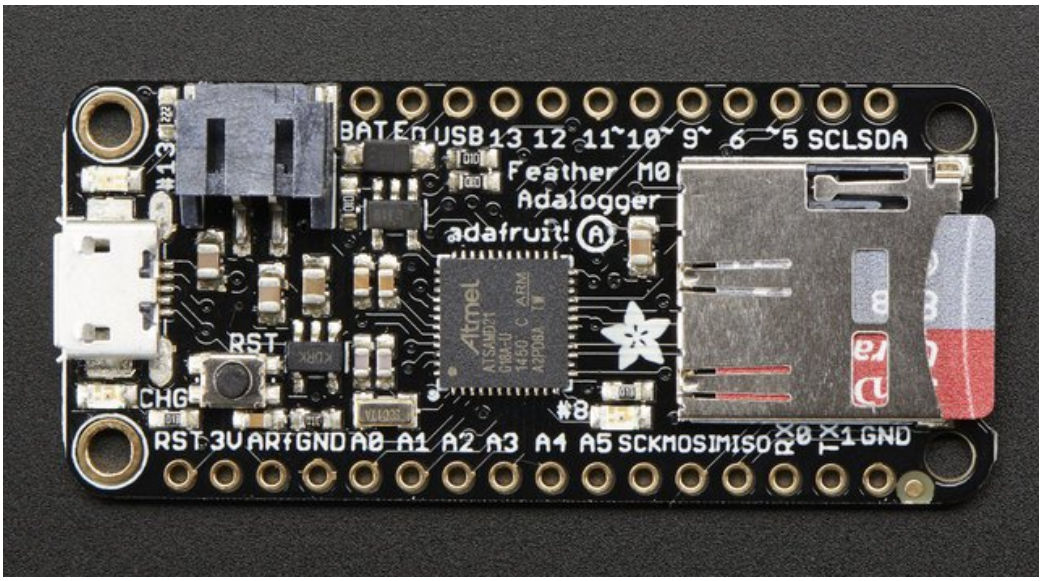
Comes fully assembled and tested, with a USB bootloader that lets you quickly use it with the Arduino IDE. We also toss in some header so you can solder it in and plug into a solderless breadboard. **Lipoly battery, MicroSD card and USB cable not included** (but we do have lots of options in the shop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

Check out our tutorial for all sorts of details, including schematics, files, IDE instructions, and more!

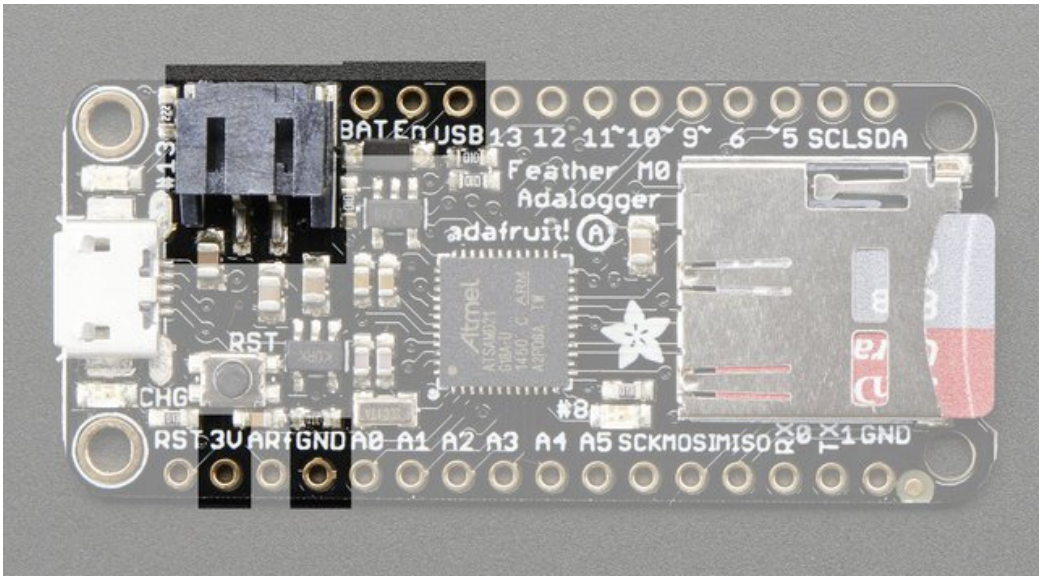
Pinouts



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 500mA peak

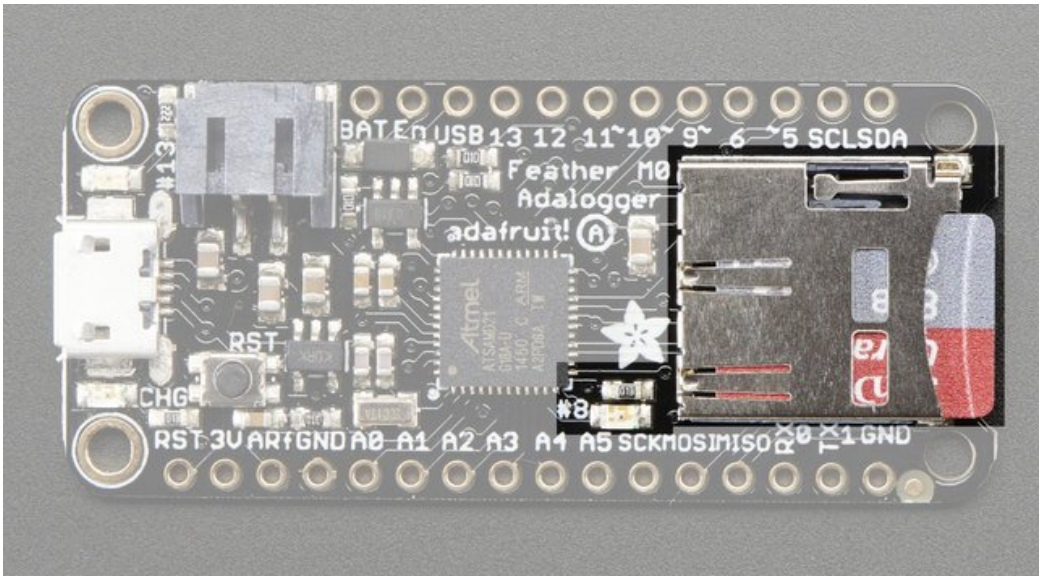
Logic pins

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.

Micro SD Card + Green LED



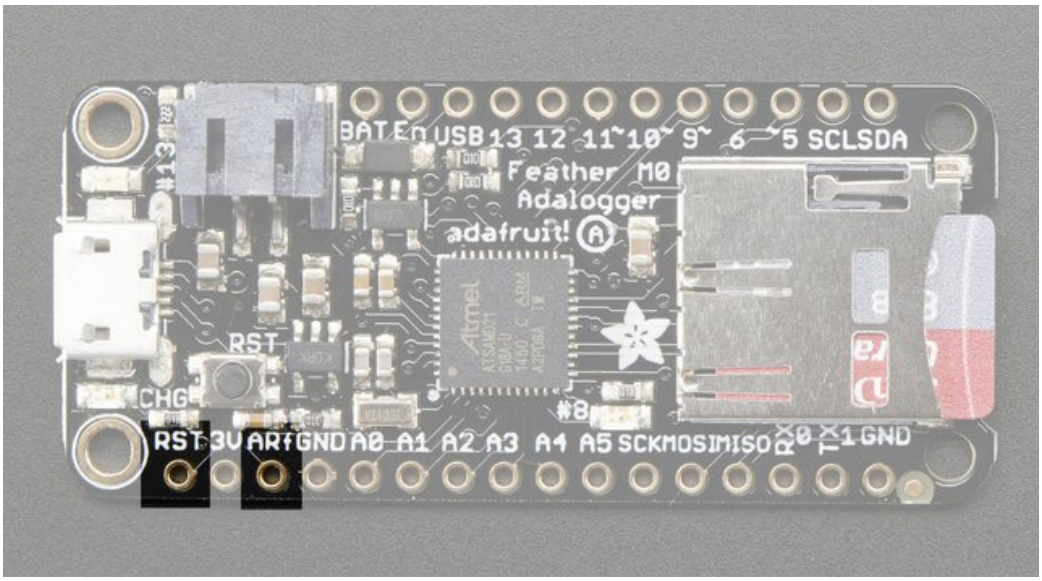
Since not all pins can be brought out to breakouts, due to the small size of the Feather, we use these to control the SD card!

- **#4** - used as the MicroSD card **CS** (chip select) pin
- **#7** - used as the MicroSD card **CD** (card detect) pin. If you want to detect when a card is inserted/removed, configure this pin as an input with a pullup. When the pin reads low (0V) then there is no card inserted. When the pin reads high, then a card is in place. It will not tell you if the card is valid, its just a mechanical switch
- **#8** - This pin was also left over, so we tied it to a green LED, its next to the SD card. It might be handy to blink this LED when writing / reading valid data or some other user-alert!

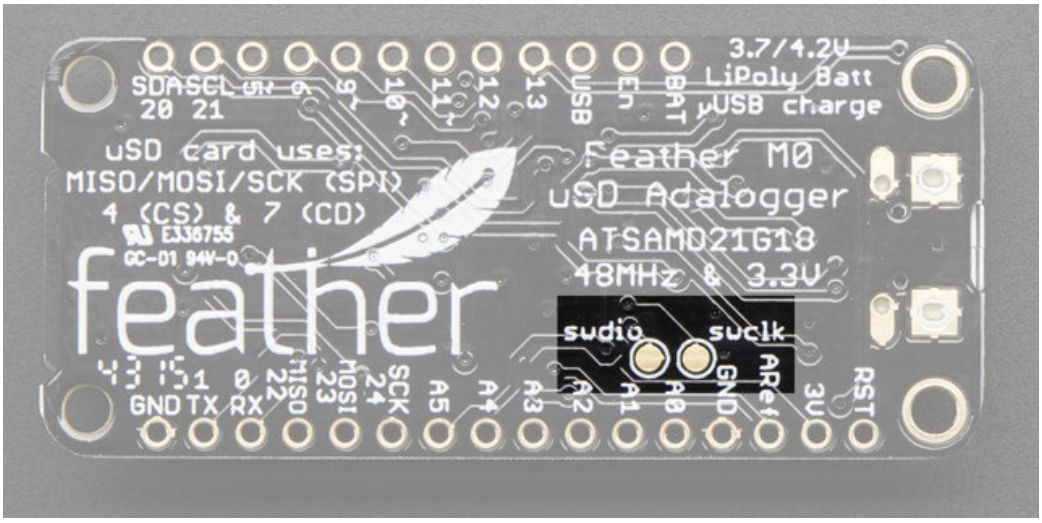
Other Pins!

- **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!



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

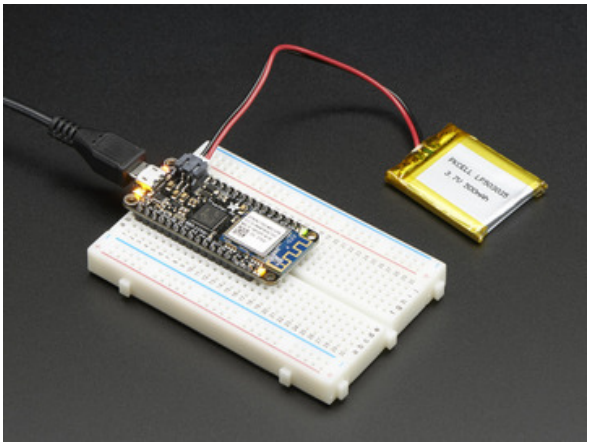
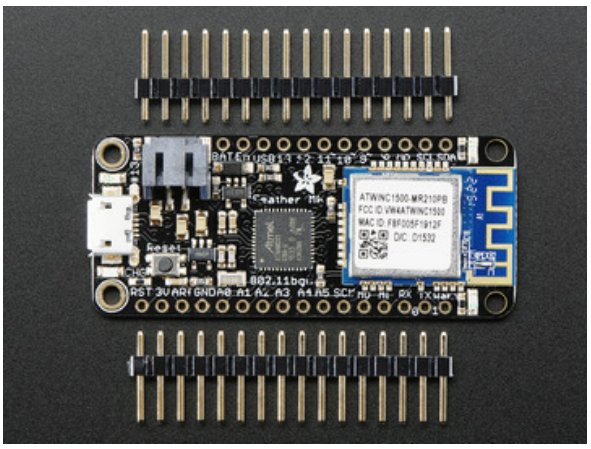


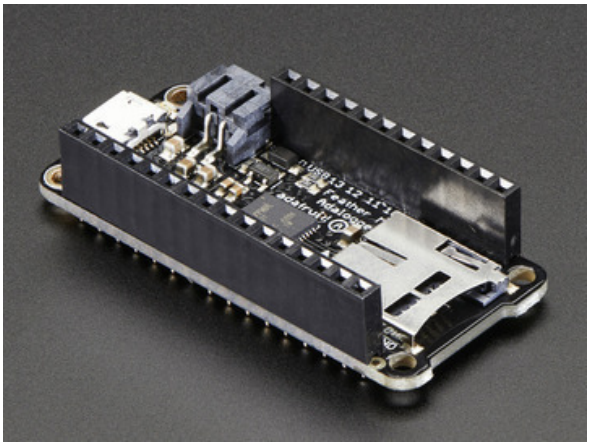
Assembly

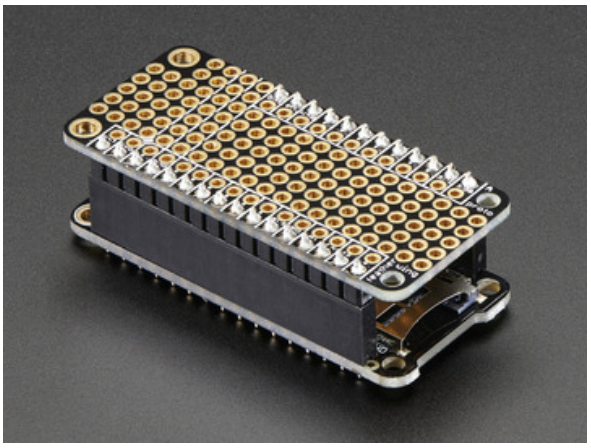
We ship Feathers fully tested but without headers attached - this gives you the most flexibility on choosing how to use and configure your Feather

Header Options!

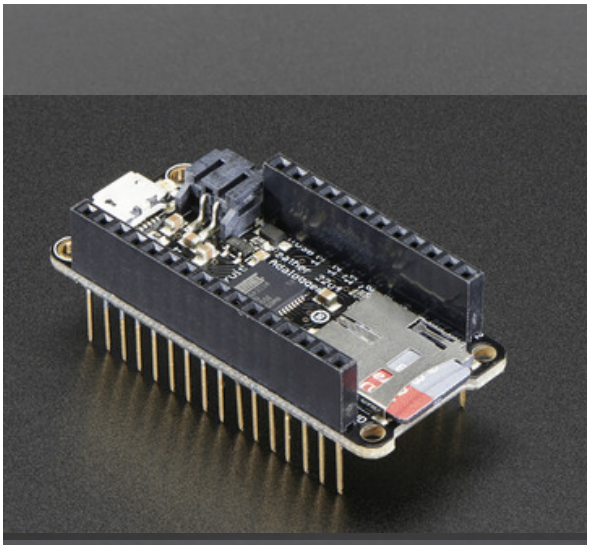
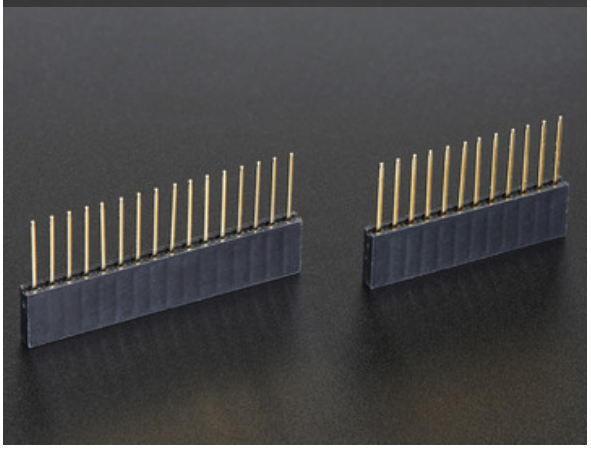
Before you go gung-ho on soldering, there's a few options to consider!

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

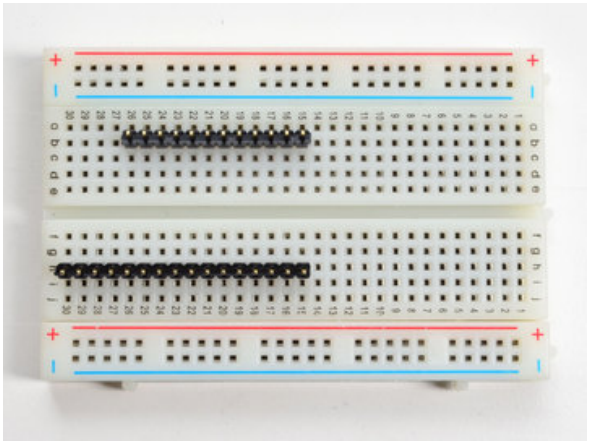
- 

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

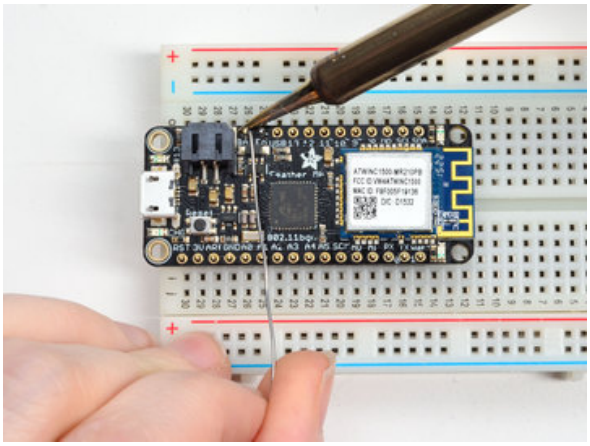
- 
- Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a solderless breadboard *and* plug a featherwing on top. But its a little bulky
- 

Soldering in Plain Headers

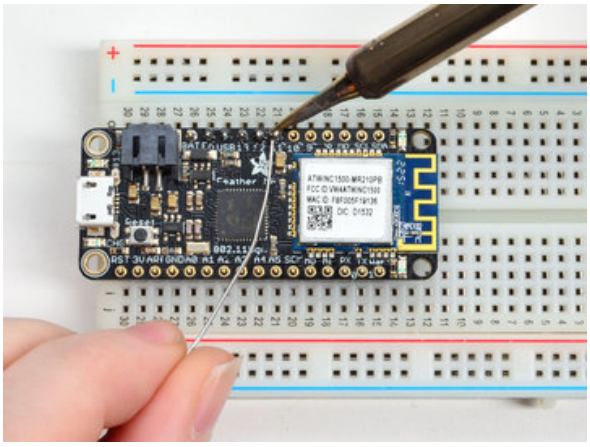
- 

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:

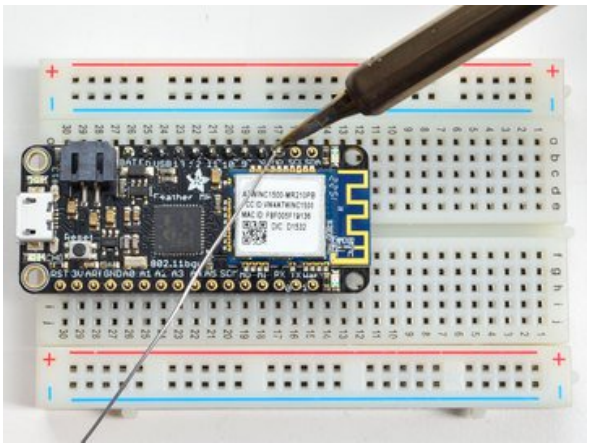
- 

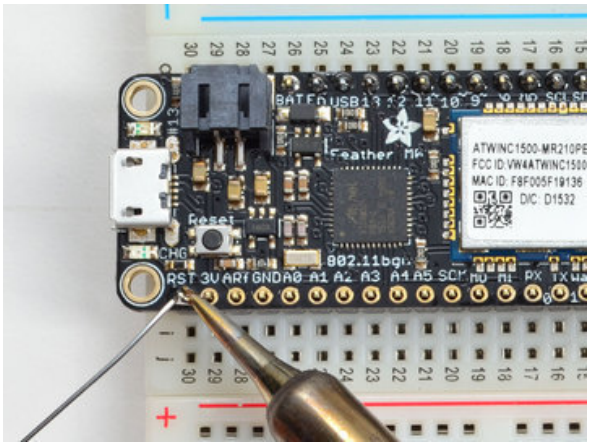
Place the breakout board over the pins so that the short pins poke through the breakout pads

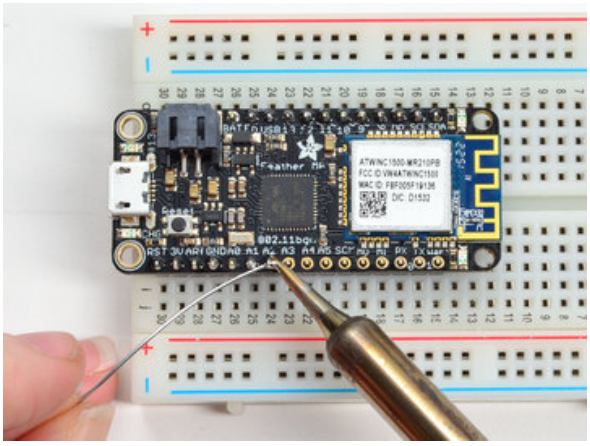
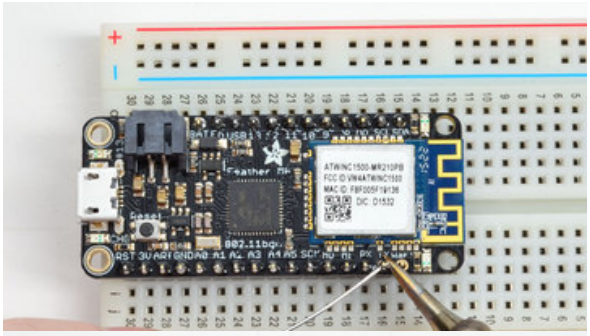
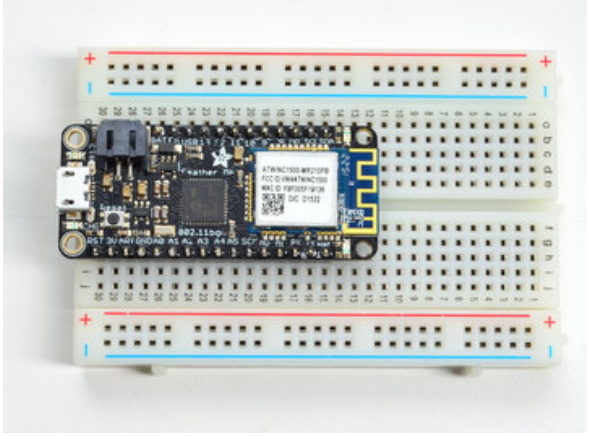
And Solder!

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)).

- 

- 

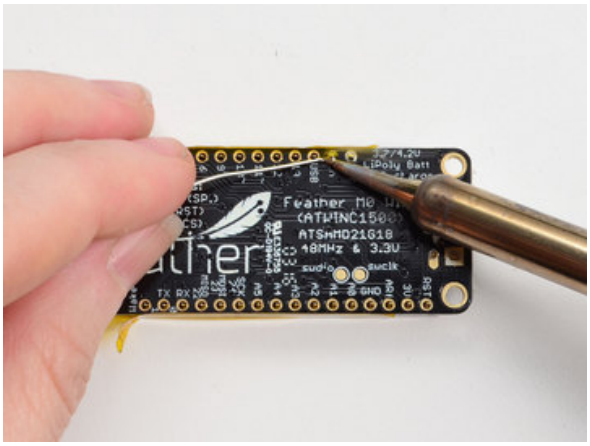
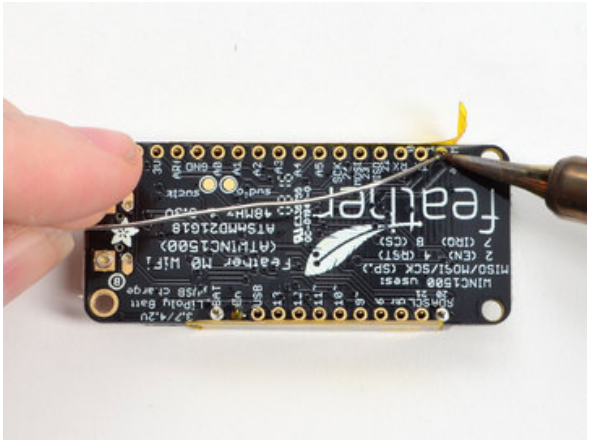
Solder the other strip as well.
- 
- 
- 
You're done! Check your solder joints visually and continue onto the next steps

Soldering on Female Header

- 

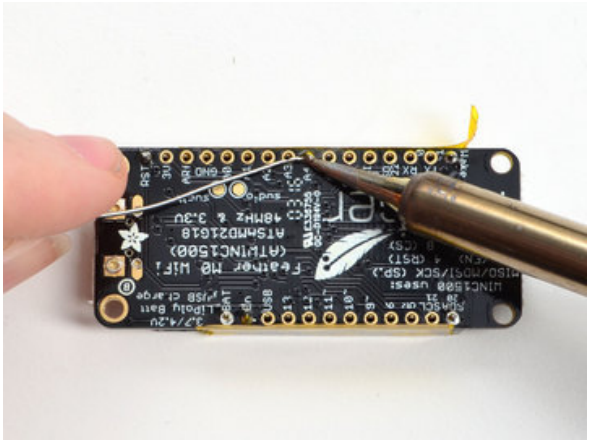
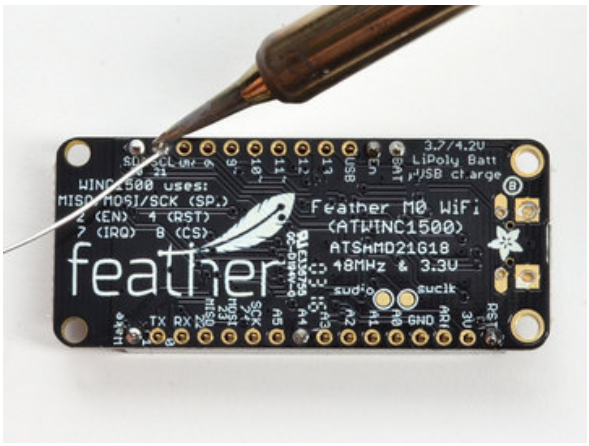
Tape In Place

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!

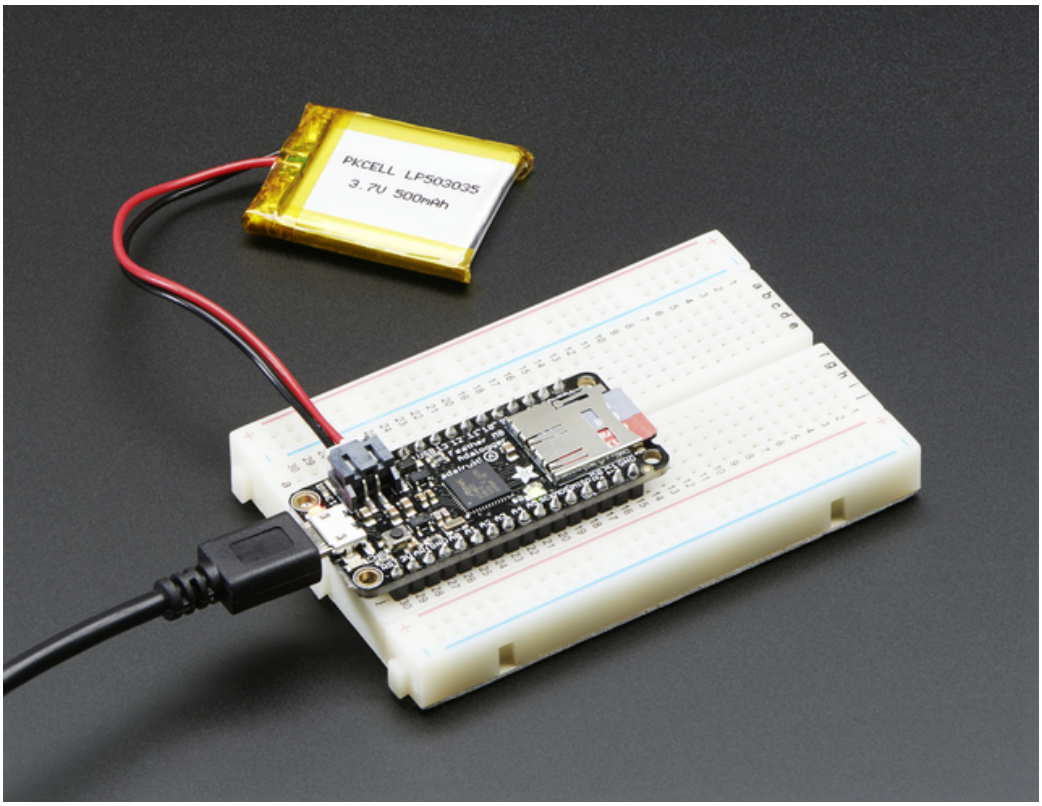
Be sure to solder all pins for reliable electrical contact.

(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

- 
3.7/4.2V
LiPoly Batt
/USB charge
WINC1500 uses:
MISO/MOSI/SCK (SPI)
2 (EN) 4 (RST)
7 (IRQ) 8 (CS)
Feather M0 WiFi
(ATWINC1500)
ATSAMD21G18
48MHz & 3.3V
feather
TX RX
GND
RST
- 

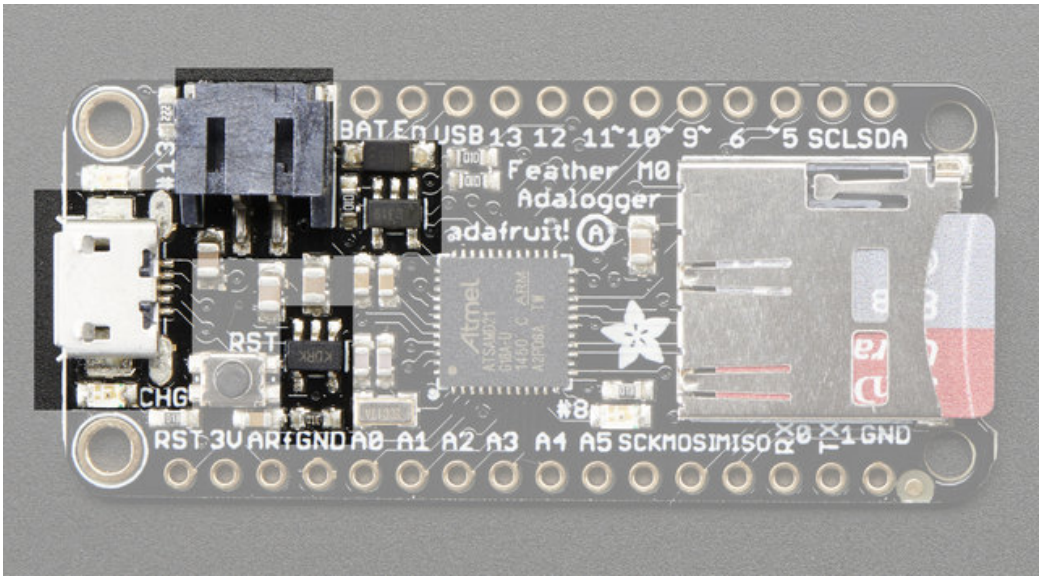
Power Management



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 100mA.** This happens 'hotswap' style so you can always keep the Lipoly connected as a 'backup' power that will only get used when USB power is lost.

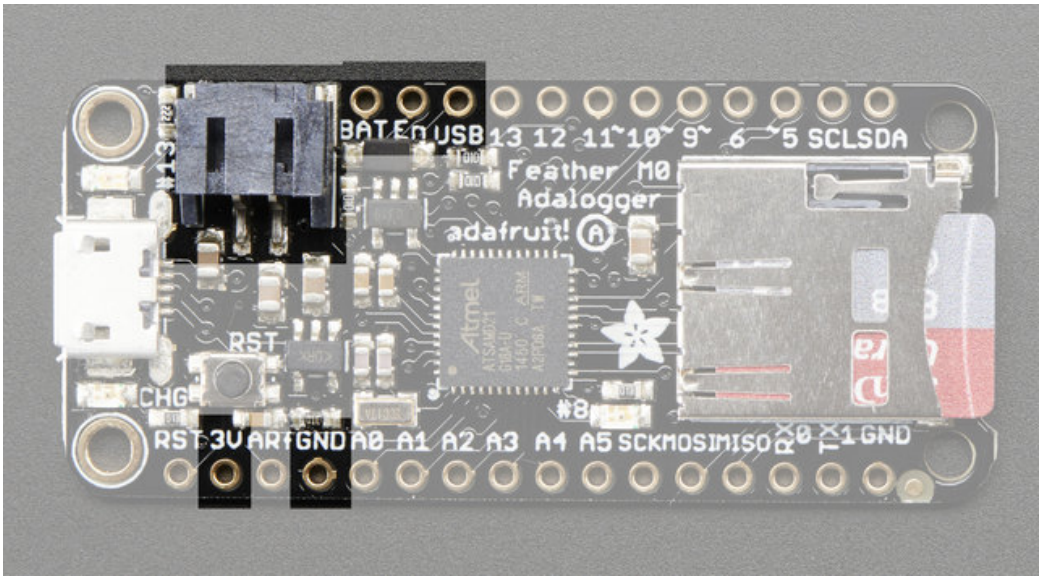
The JST connector polarity is matched to Adafruit LiPoly batteries. Using wrong polarity batteries can destroy your Feather



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 500mA peak regulator. While you can get 500mA from it, you can't do it continuously from 5V as it will overheat the regulator. It's fine for, say, powering an ESP8266 WiFi chip or XBee radio though, since the current draw is 'spikey' & 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

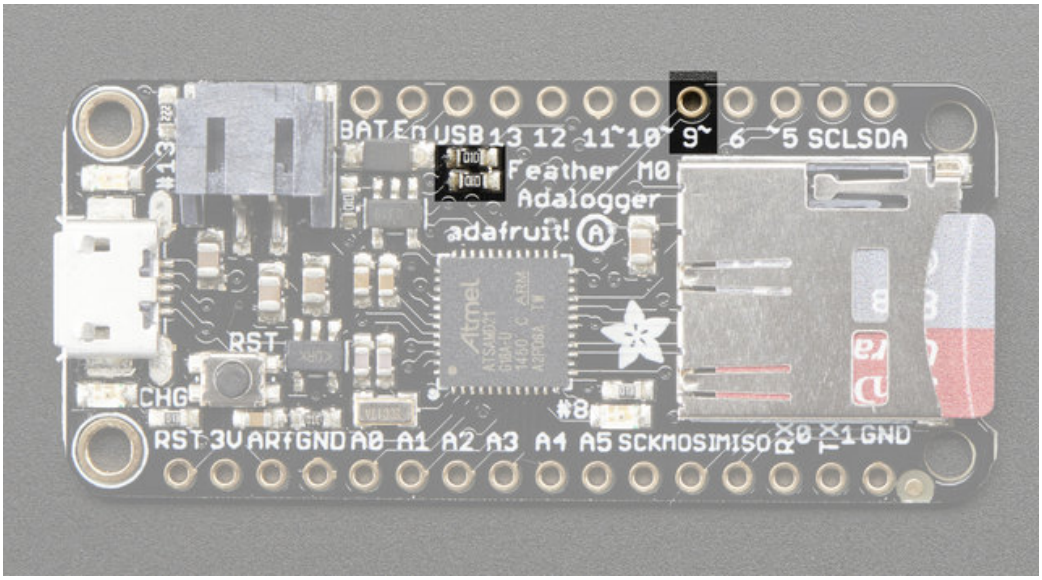
the battery needs recharging. Lipoly batteries are 'maxed out' at 4.2V and stick around 3.7V for much of the battery life, then slowly sink down to 3.2V or so before the protection circuitry cuts it off. By measuring the voltage you can quickly tell when you're heading below 3.7V

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);
```

This voltage will 'float' at 4.2V when no battery is plugged in, due to the lipoly charger output, so its not a good way to detect if a battery is plugged in or not (there is no simple way to detect if a battery is plugged in)



Average Power Draw w/SD Card

The average power draw of the ATSAMD21 + regulator circuitry is 11mA. Both the red and green LED each draw 1mA if you light them up.

Say you are running this sample sketch which logs the analog voltage on A0 to an SD card file once a second.

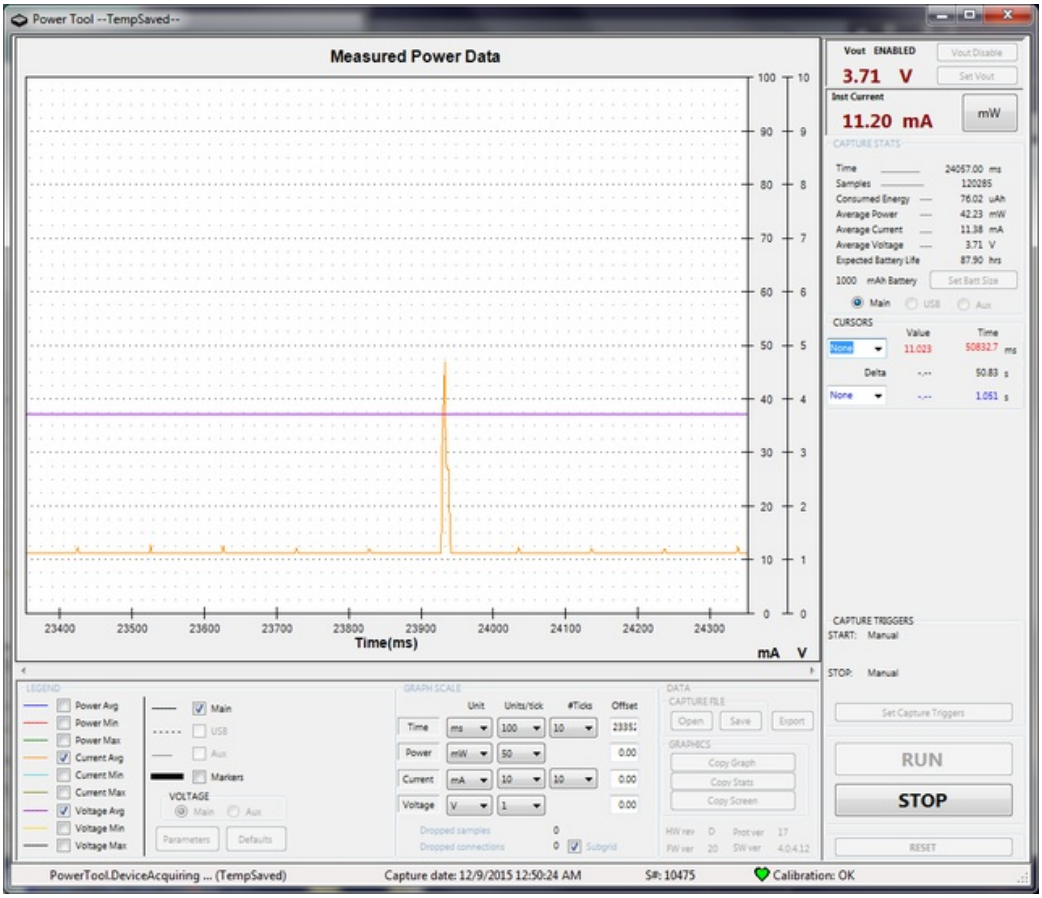
```
#include <SPI.h>
#include <SD.h>
// Set the pins used
#define cardSelect 4
File logfile;
// blink out an error code
void error(uint8_t errno) {
while(1) {
uint8_t i;
```

```
for (i=0; i<errno; i++) {
digitalWrite(13, HIGH);
delay(100);
digitalWrite(13, LOW);
delay(100);
}
for (i=errno; i<10; i++) {
delay(200);
}
}
}
// This line is not needed if you have Adafruit SAMD board package 1.6.2+
// #define Serial SerialUSB
void setup() {
// connect at 115200 so we can read the GPS fast enough and echo without dropping
chars
// also spit it out
Serial.begin(115200);
Serial.println("\r\nAnalog logger test");
pinMode(13, OUTPUT);
// see if the card is present and can be initialized:
if (!SD.begin(cardSelect)) {
Serial.println("Card init. failed!");
error(2);
}
char filename[15];
strcpy(filename, "ANALOG00.TXT");
for (uint8_t i = 0; i < 100; i++) {
filename[6] = '0' + i/10;
filename[7] = '0' + i%10;
// create if does not exist, do not open existing, write, sync after write
if (! SD.exists(filename)) {
break;
}
}
logfile = SD.open(filename, FILE_WRITE);
if( ! logfile ) {
Serial.print("Couldnt create ");
Serial.println(filename);
error(3);
}
Serial.print("Writing to ");
Serial.println(filename);
pinMode(13, OUTPUT);
```

```
pinMode(8, OUTPUT);
Serial.println("Ready!");
}
uint8_t i=0;
void loop() {
digitalWrite(8, HIGH);
logfile.print("A0 = "); logfile.println(analogRead(0));
Serial.print("A0 = "); Serial.println(analogRead(0));
digitalWrite(8, LOW);
delay(100);
}
```

adalogger.ino hosted with ❤ by GitHub view raw

You'll draw 11mA or so for 100ms during the delay(100) line. Since we blink the pin #8 LED, we'll also get a 11mA blip for about 10ms. The data for the SD card is *buffered* which means that whenever we reach 512 bytes of log file that needs to be written, the SD card will actually write the data. When this happens you'll see a 50mA pulse for 10ms. If you use **flush()** to write the log file, this pulse will be much longer, as you have to write the file as well as the file table sectors. Your 50mA spike could end up being 500ms or longer. So basically, keep your file writes to a minimum if you can avoid it!



ENable pin