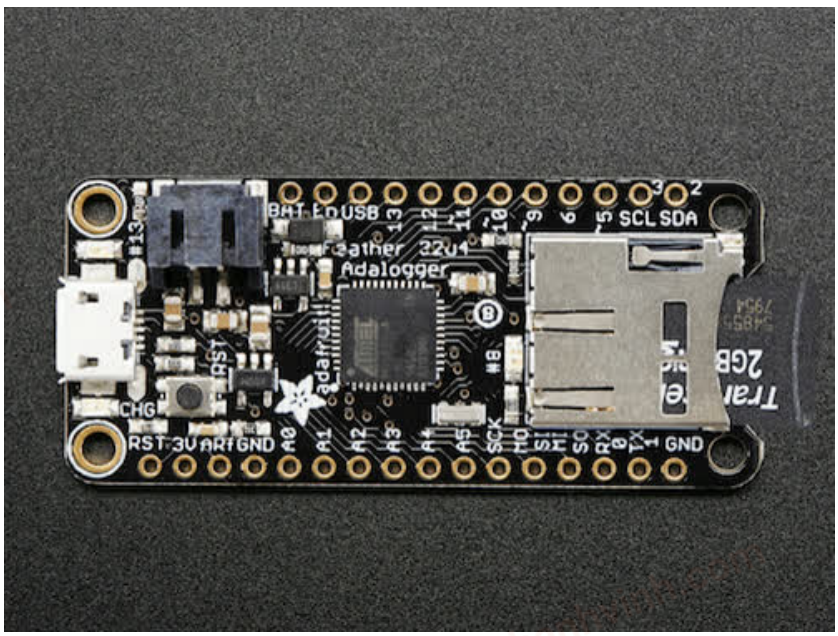
chanhvinh.com

Adafruit Feather 32u4 Adalogger

Created by lady ada



Last updated on 2016-09-21 01:19:38 AM UTC

chanhvinh.com

Guide Contents

Guide Contents 2
Overview 4
Pinouts 9
Power Pins 10
Logic pins 11
Micro SD Card + Green LED 11
Other Pins! 12
Assembly 13
Header Options! 13
Soldering in Plain Headers 16
- Prepare the header strip: 16
- Add the breakout board: 17
- And Solder! 17

Soldering on Female Header 19
- Tape In Place 20
- Flip & Tack Solder 21
- And Solder! 22

Power Management 25
Battery + USB Power 25
Power supplies 26
Measuring Battery 27
ENable pin 28
Arduino IDE Setup 29
- https://adafruit.github.io/arduino-board-index/package_adafruit_index.json 30

Using with Arduino IDE 32
Install Drivers (Windows Only) 34
Blink 36
Manually bootloading 37
- Ubuntu & Linux Issue Fix 38

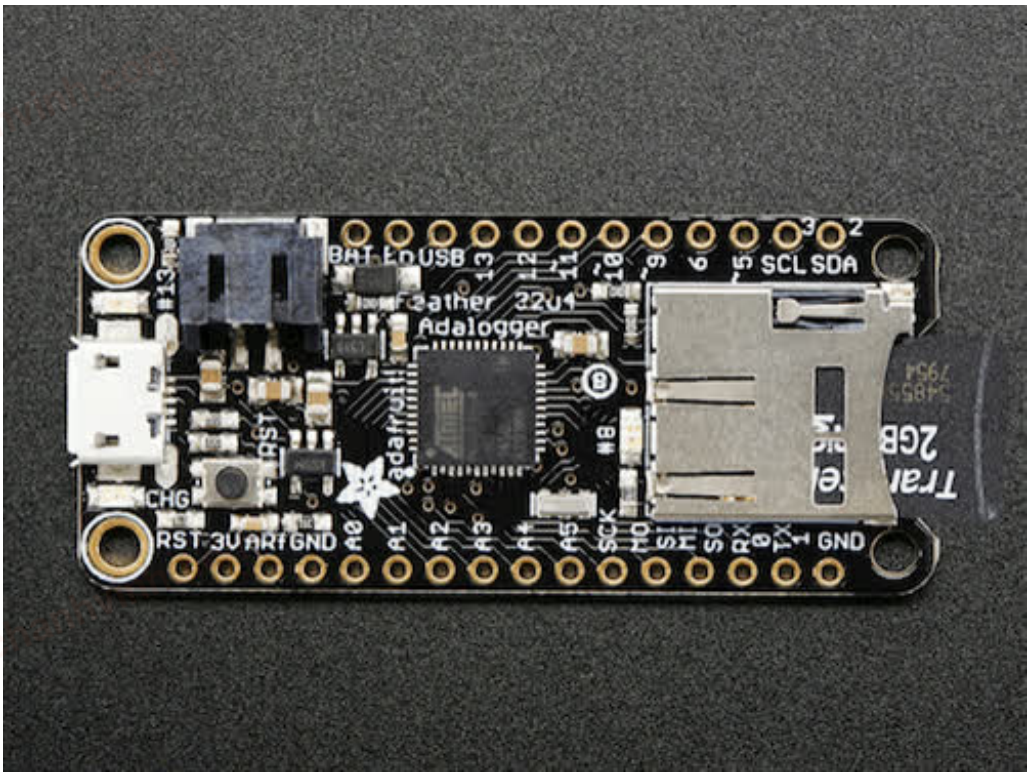
HELP! 39
Using the SD Card 42
Example logging sketch 43

chanhvinh.com

Downloads 47
Schematic 47
Fabrication Print 47

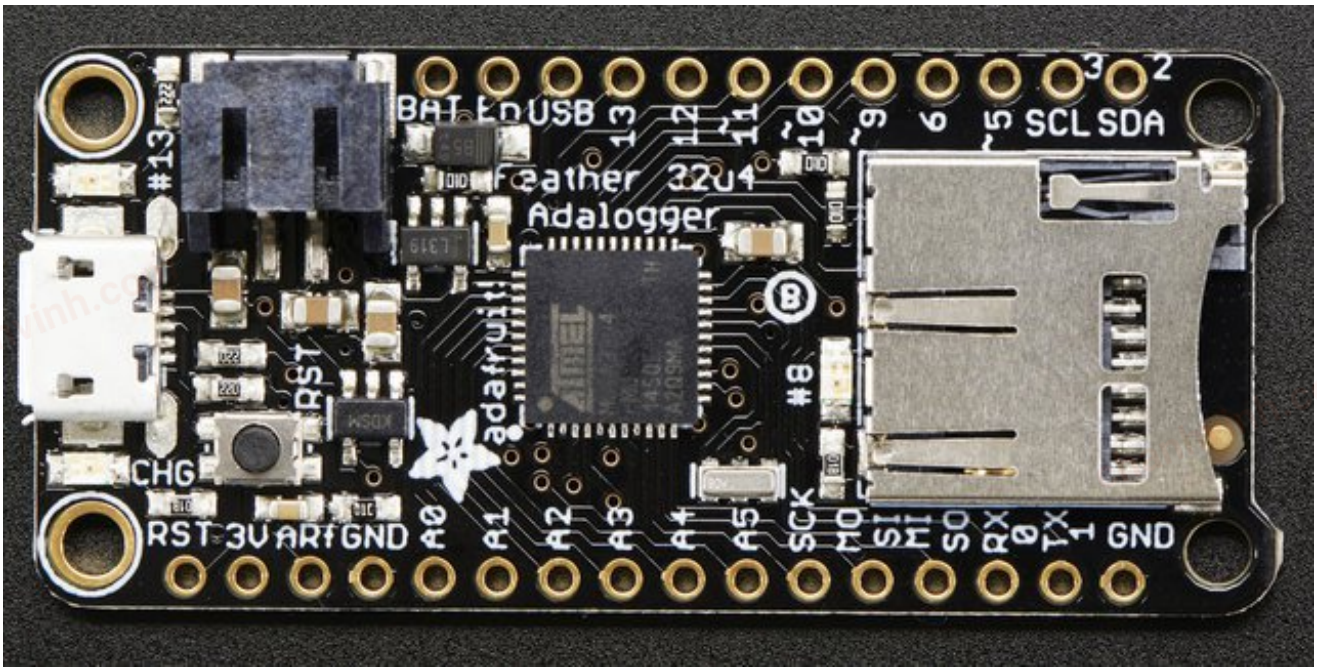
chanhvinh.com

Overview

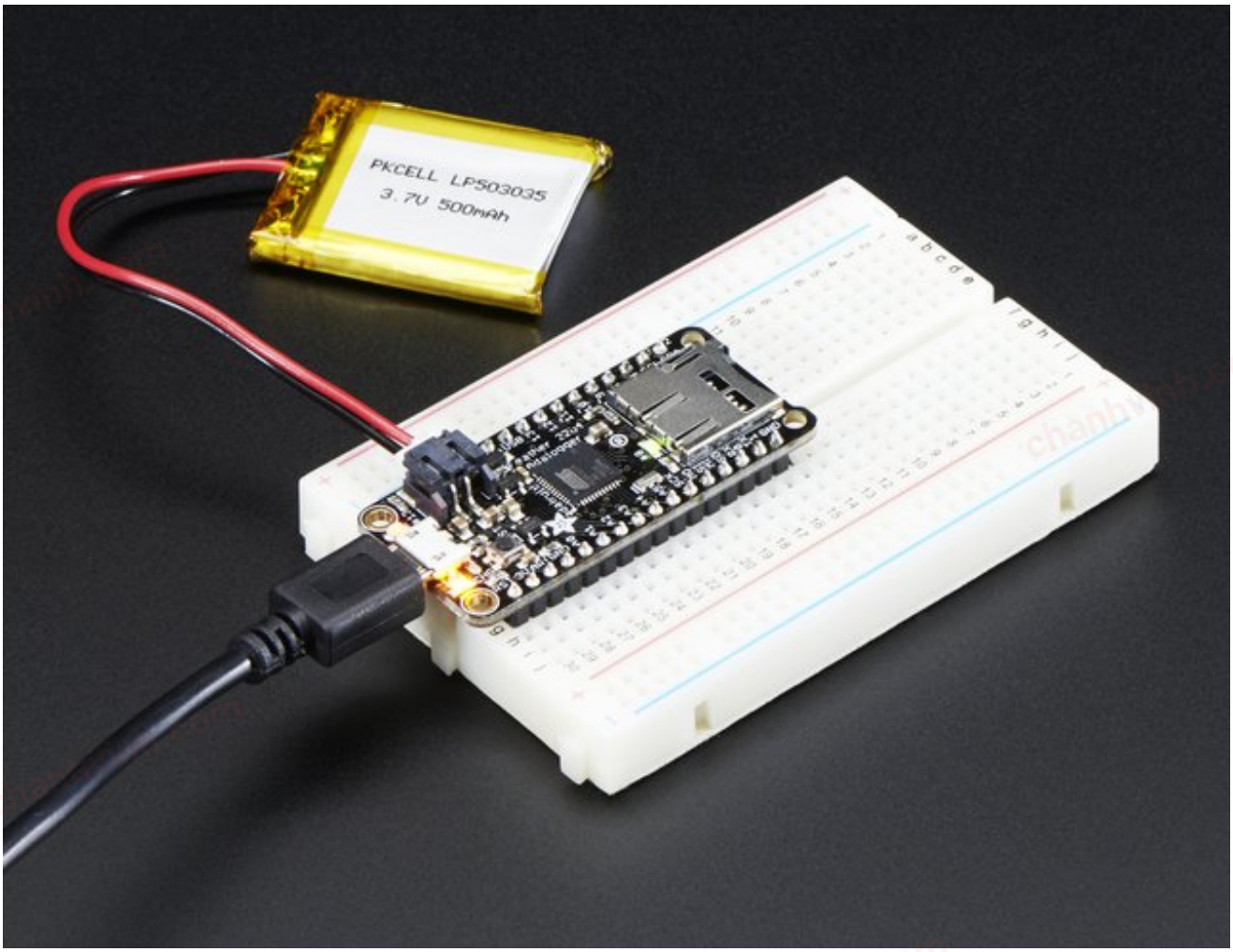


Feather is the new development board from Adafruit, and like it's namesake it is thin, light, and lets you fly! We designed Feather to be a new standard for portable microcontroller cores.

This is the **Adafruit Feather 32u4 Adalogger** - our take on an 'all-in-one' datalogger (or data-reader) with built in USB and battery charging. Its an Adafruit Feather 32u4 with a microSD holder ready to rock!



At the Feather 32u4's heart is at ATmega32u4 clocked at 8 MHz and at 3.3V logic, a chip setup we've had tons of experience with as [it's the same as the Flora](http://adafru.it/dVl). This chip has 32K of flash and 2K of RAM, with built in USB so not only does it have a USB-to-Serial program & debug capability built in with no need for an FTDI-like chip, it can also act like a mouse, keyboard, USB MIDI device, etc.



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.



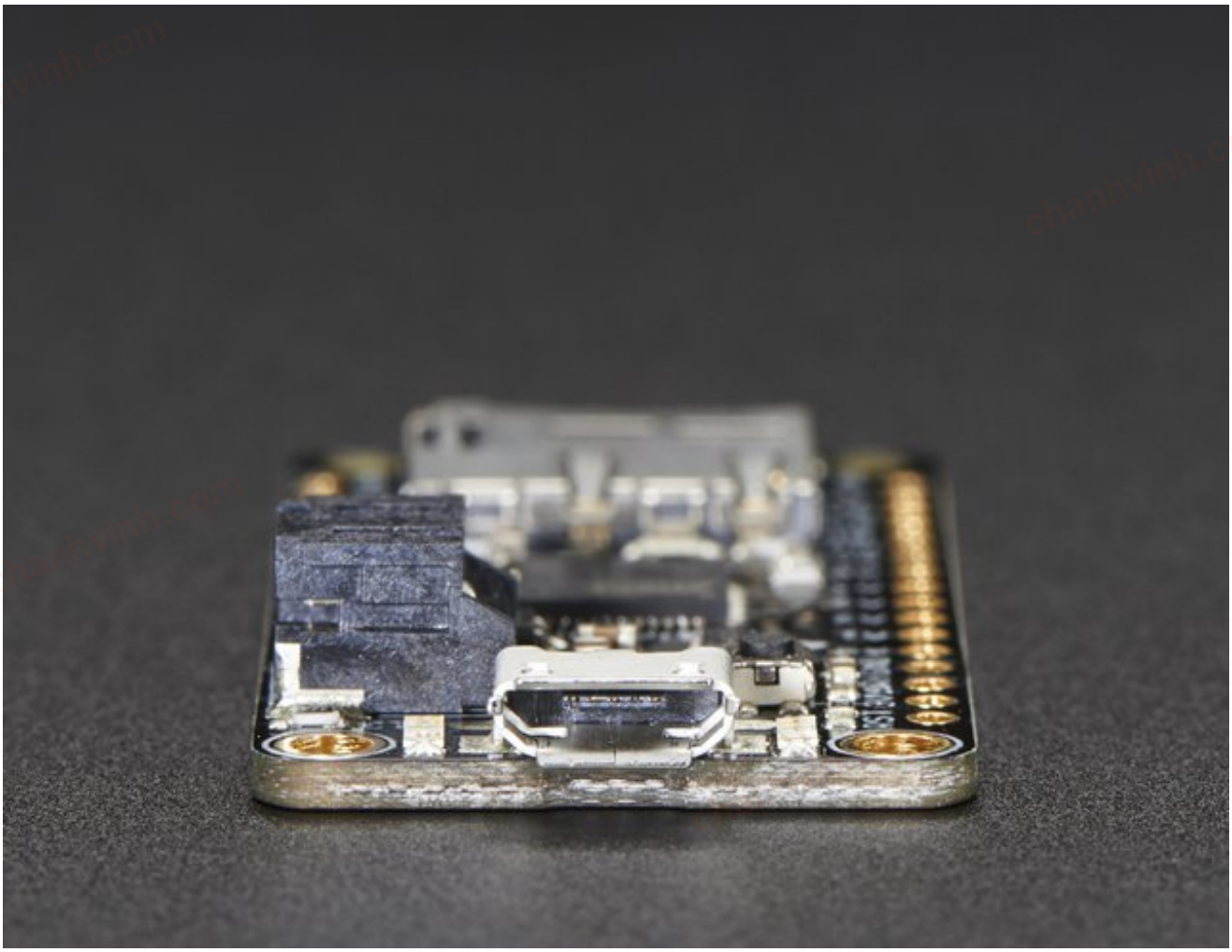
Here's some handy specs! Like all Feather 32u4's you get:

- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm) without headers soldered in
- Light as a (large?) feather - 5.1 grams
- ATmega32u4 @ 8MHz with 3.3V logic/power
- 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 32u4 Adalogger** uses the extra space left over to add MicroSD + a green

LED:

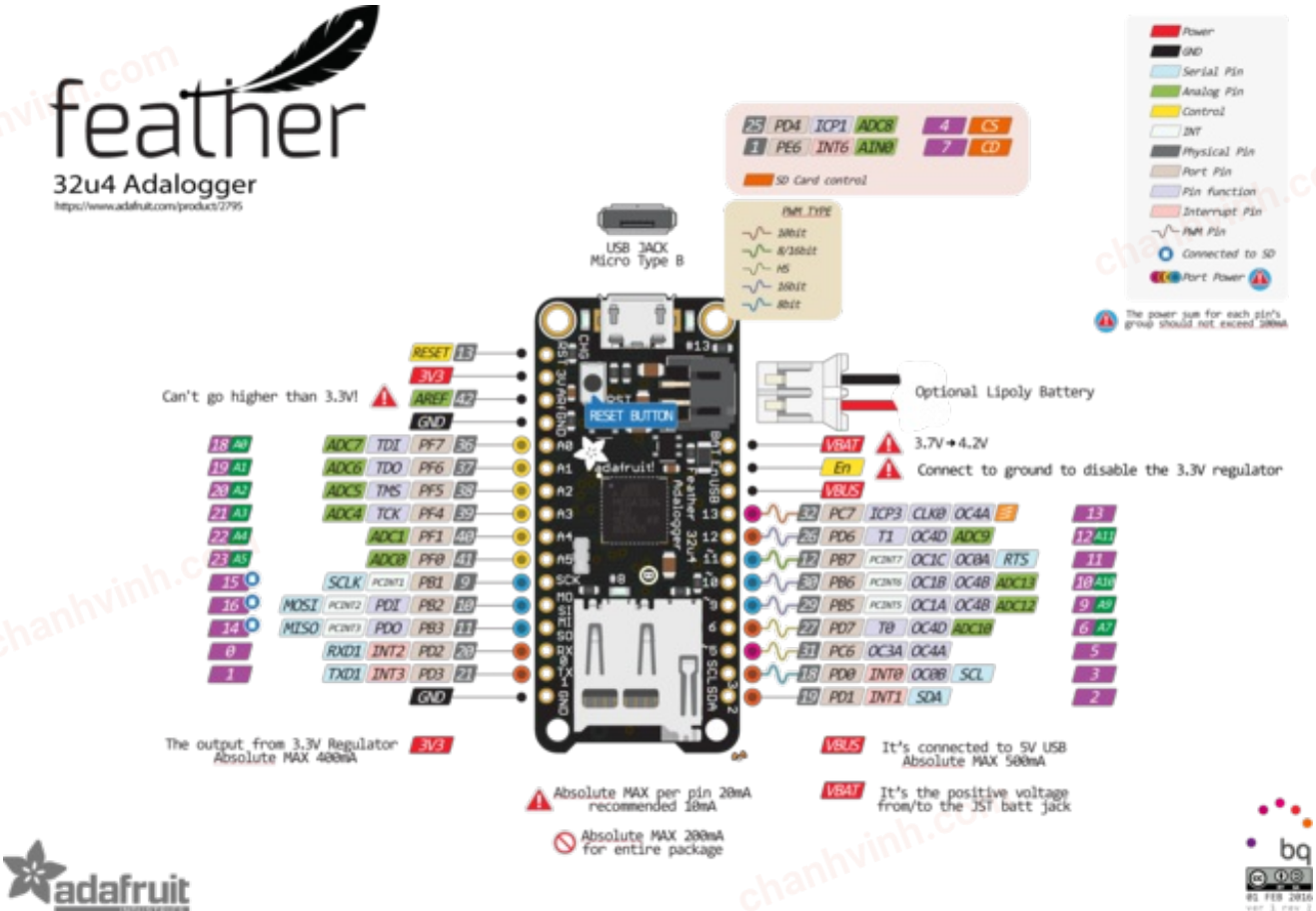
- Pin #8 green LED for your blinking pleasure
- MicroSD card holder for adding as much storage as you could possibly want, for reading or writing.

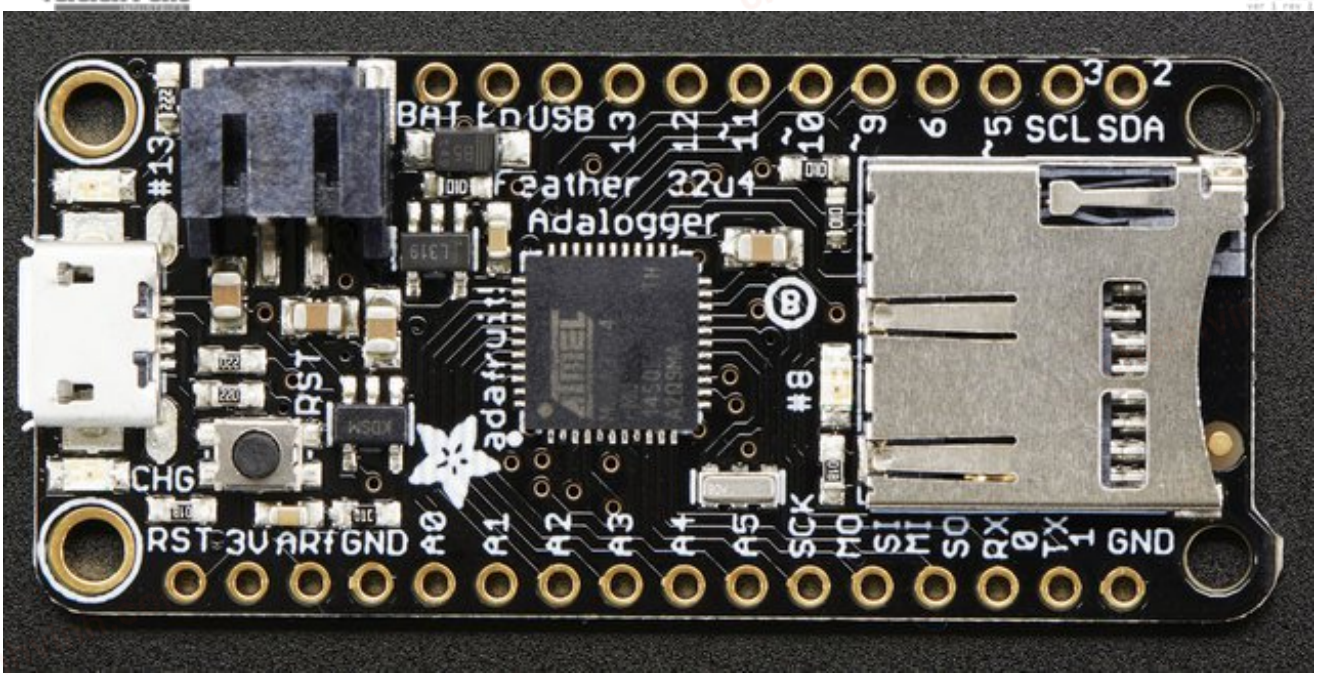


Comes fully assembled and tested, with a USB bootloader that lets you quickly use it with the Arduino IDE. We also toss in some header so you can solder it in and plug into a solderless breadboard. **Lipoly battery, MicroSD card and USB cable not included** (but we do have lots of options in the shop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

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

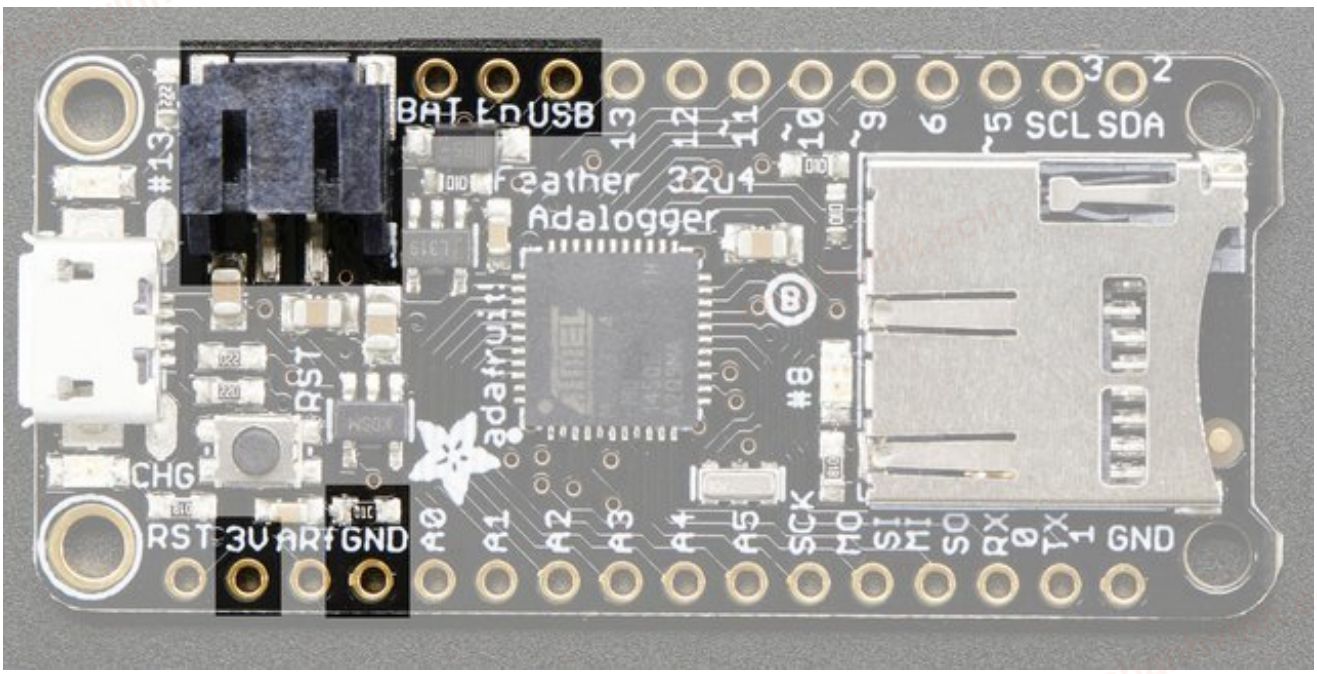
Pinouts


feather
32u4 Adalogger
https://www.adafruit.com/product/2795
USB JACK
Micro Type B
RESET BUTTON
Optional Lipoly Battery
Can't go higher than 3.3V!
The output from 3.3V Regulator
Absolute MAX 400mA
Absolute MAX per pin 20mA
recommended 10mA
Absolute MAX 200mA
for entire package
3.7V→4.2V
Connect to ground to disable the 3.3V regulator
It's connected to 5V USB
Absolute MAX 500mA
It's the positive voltage
from/to the JST batt jack
SD Card control
adafruit


BAT
USB
SCL SDA
Feather 32u4
Adalogger
RST
CHG
adafruit
RST 3V ARF GND A0 A1 A2 A3 A4 A5 SCK MO MI SO RX TX GND



Power Pins



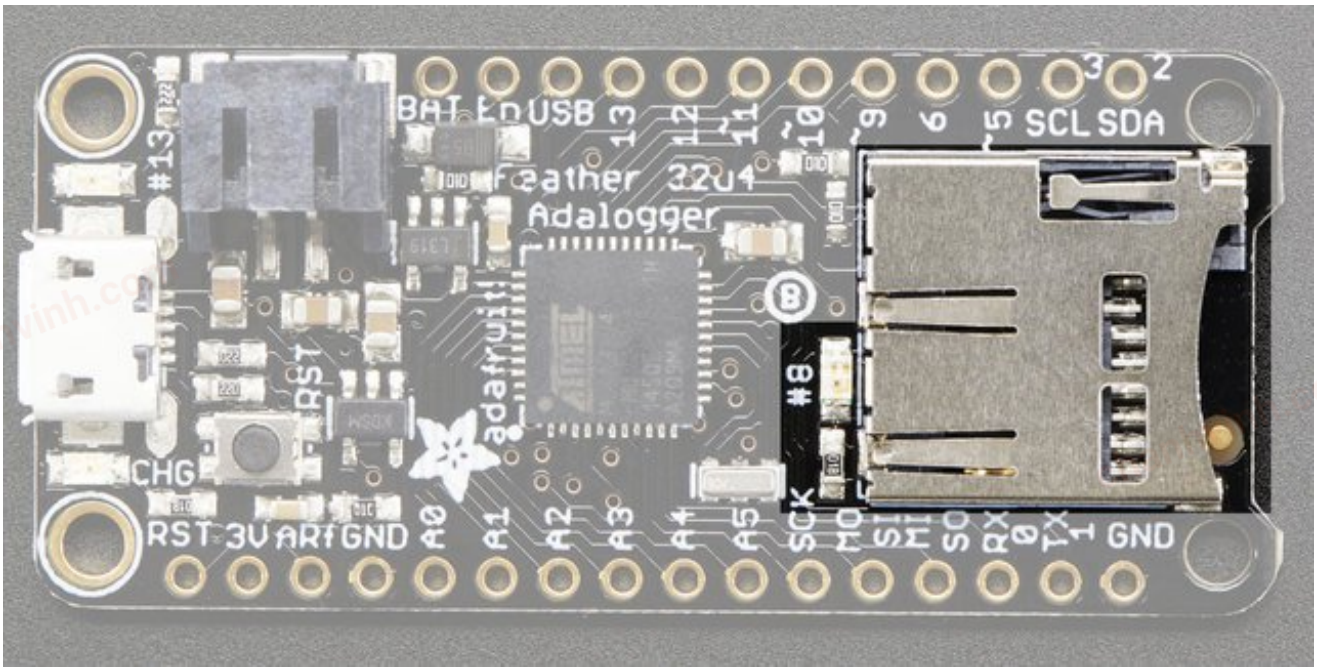
- **GND** - this is the common ground for all power and logic
- **BAT** - this is the positive voltage to/from the JST jack for the optional Lipoly battery
- **USB** - this is the positive voltage to/from the micro USB jack if connected
- **EN** - this is the 3.3V regulator's enable pin. It's pulled 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

- **#0** / **RX** - GPIO #0, also receive (input) pin for **Serial1** and Interrupt #2
- **#1** / **TX** - GPIO #1, also transmit (output) pin for **Serial1** and Interrupt #3
- **#2** / **SDA** - GPIO #2, 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. Also Interrupt #1
- **#3** / **SCL** - GPIO #3, 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. Can also do PWM output and act as Interrupt #0.
- **#5** - GPIO #5, can also do PWM output
- **#6** - GPIO #6, can also do PWM output and analog input **A7**
- **#9** - GPIO #9, also analog input **A9** and can do PWM output. 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, also analog input **A10** and can do PWM output.
- **#11** - GPIO #11, can do PWM output.
- **#12** - GPIO #12, also analog input **A11**
- **#13** - GPIO #13, can do PWM output and is connected to the **red LED** next to the USB jack
- **A0 thru A5** - These are each analog input as well as digital I/O pins.
- **SCK/MOSI/MISO** - These are the hardware SPI pins, **used by the microSD card too!** You can use them as everyday GPIO pins if the SD card is not inserted. However, we really recommend keeping them free as they should be kept available for the SD. If they are used, make sure its with a device that will kindly share the SPI bus! Also used to reprogram the chip with an AVR programmer if you need.

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!

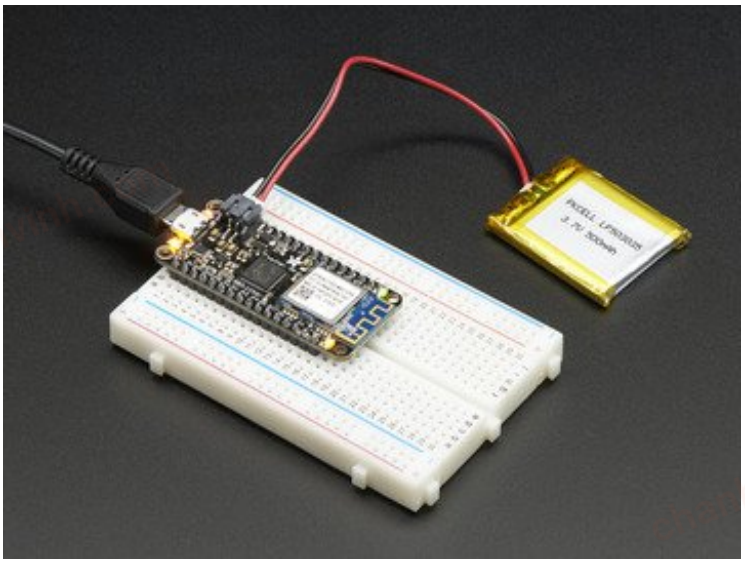
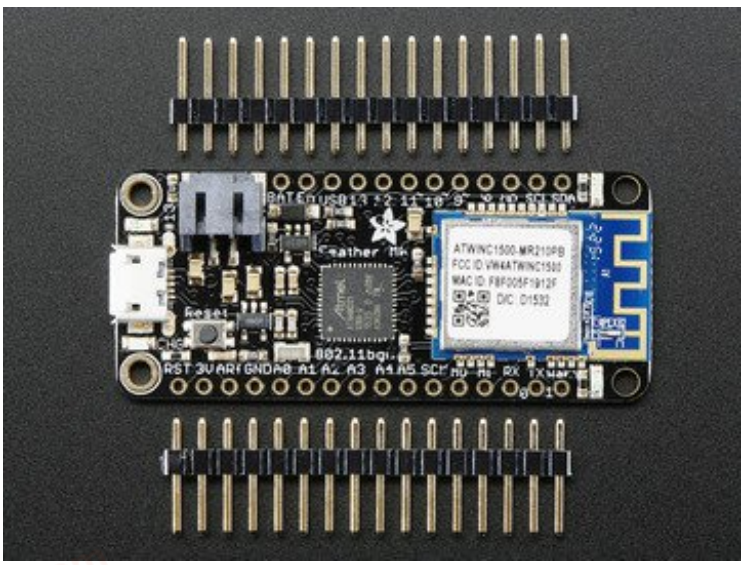
- **RST** - this is the Reset pin, tie to ground to manually reset the AVR, as well as launch the bootloader manually
- **ARef** - the analog reference pin. Normally the reference voltage is the same as the chip logic voltage (3.3V) but if you need an alternative analog reference, connect it to this pin and select the external AREF in your firmware. Can't go higher than 3.3V!

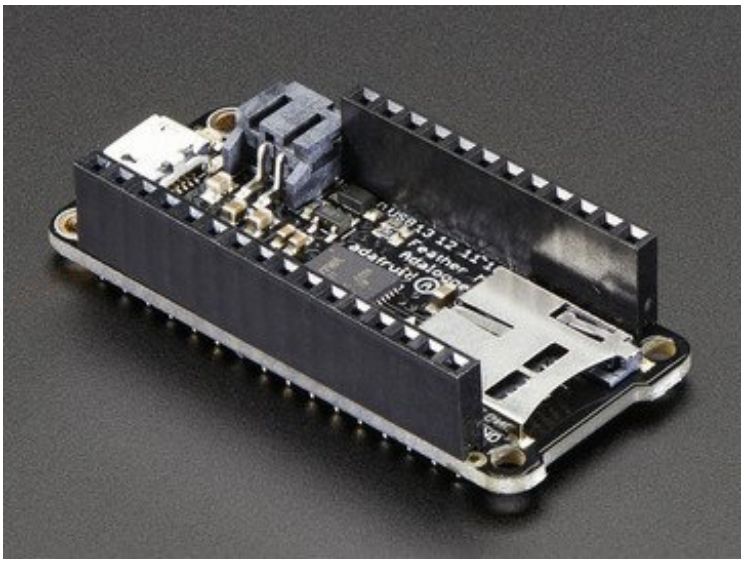
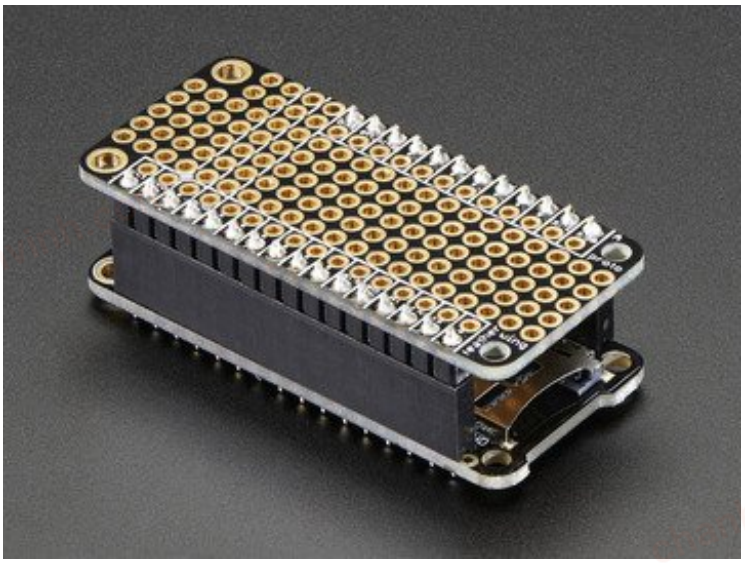
Assembly

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

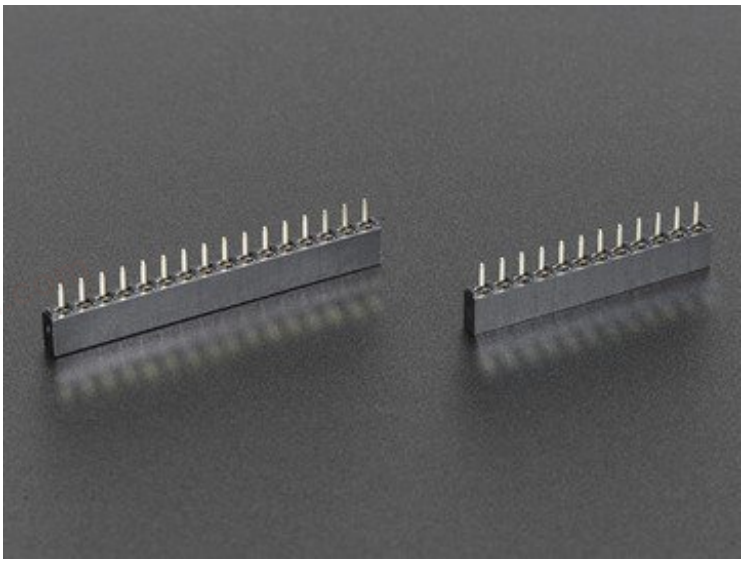
Header Options!

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

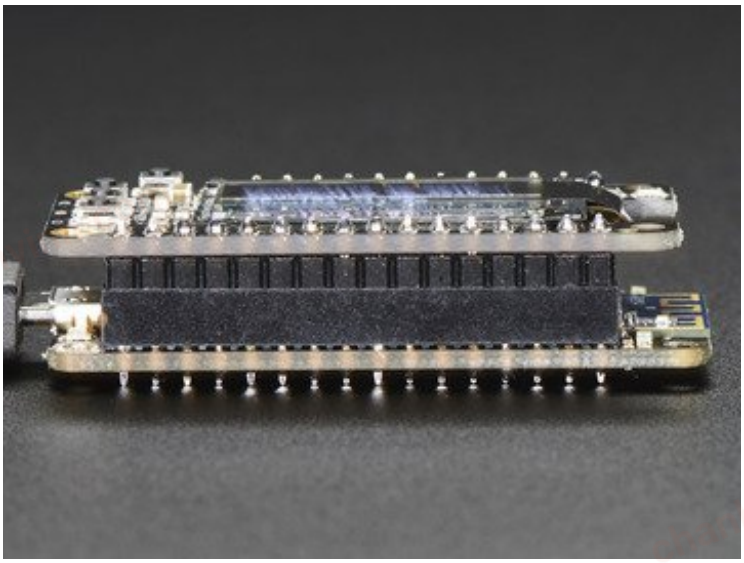
- 
The first option is soldering in plain male headers, this lets you plug in the Feather into a solderless breadboard
- 

- 
- 

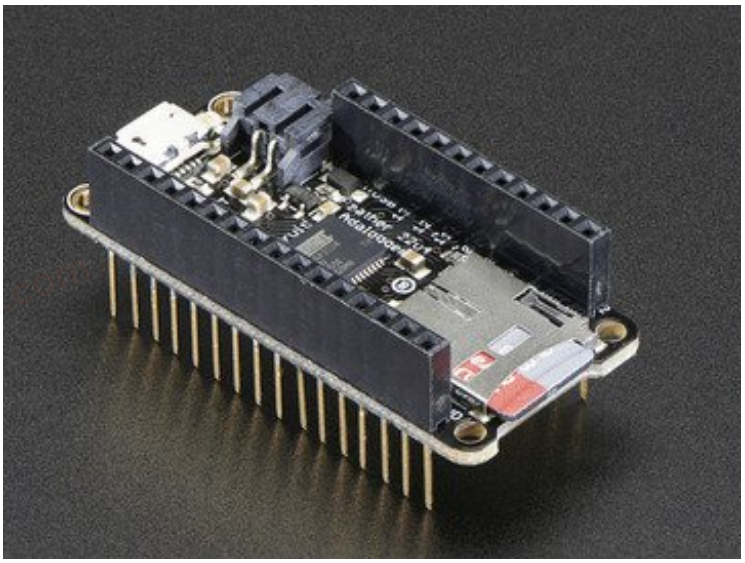
Another option is to go with socket female headers. This won't let you plug the Feather into a breadboard but it will let you attach featherwings very easily

- 

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

- 

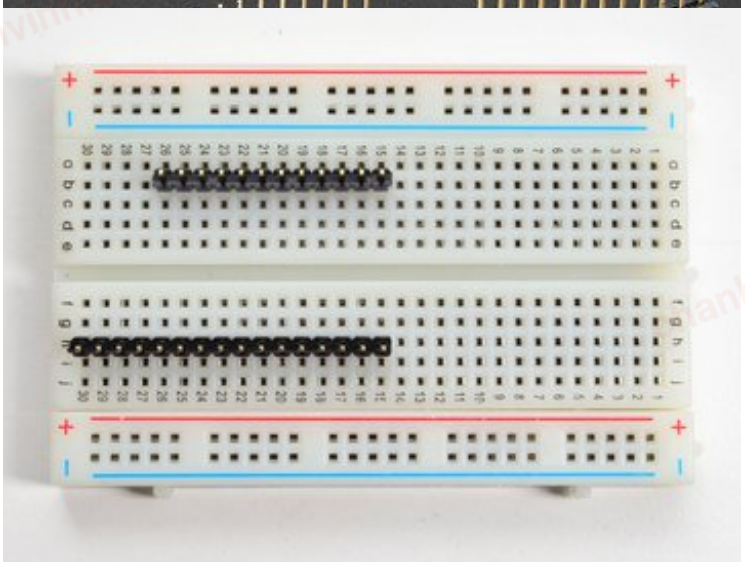
Finally, there's the "Stacking Header" option. This one is sort of the best-of-both-worlds. You get the ability to plug into a

- 

solderless breadboard *and* plug a featherwing on top. But its a little bulky

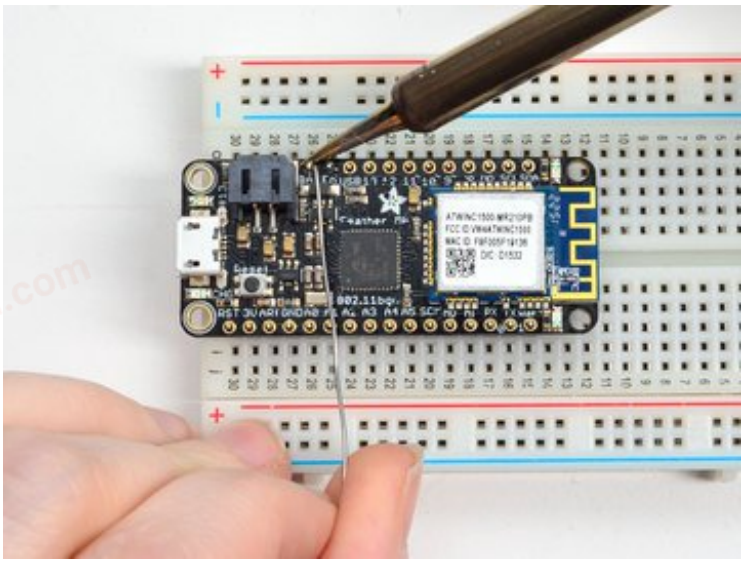
- 

Soldering in Plain Headers

- 

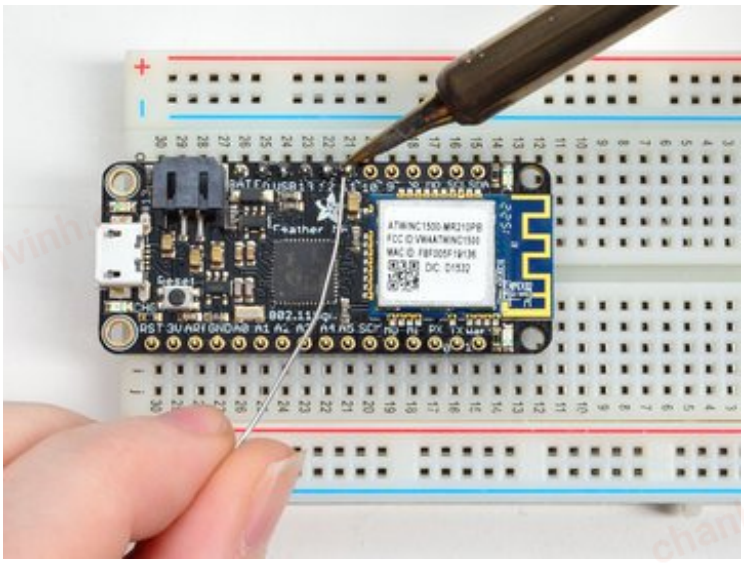
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**

- 

Add the breakout board:

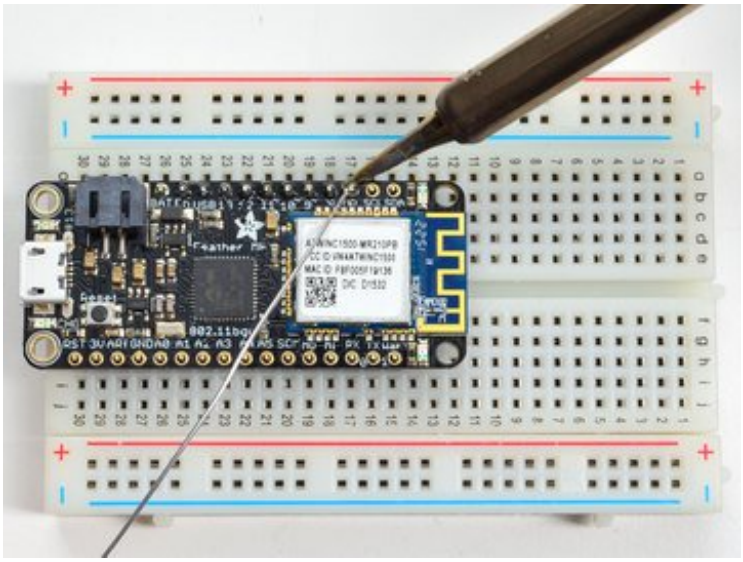
Place the breakout board over the pins so that the short pins poke through the breakout pads

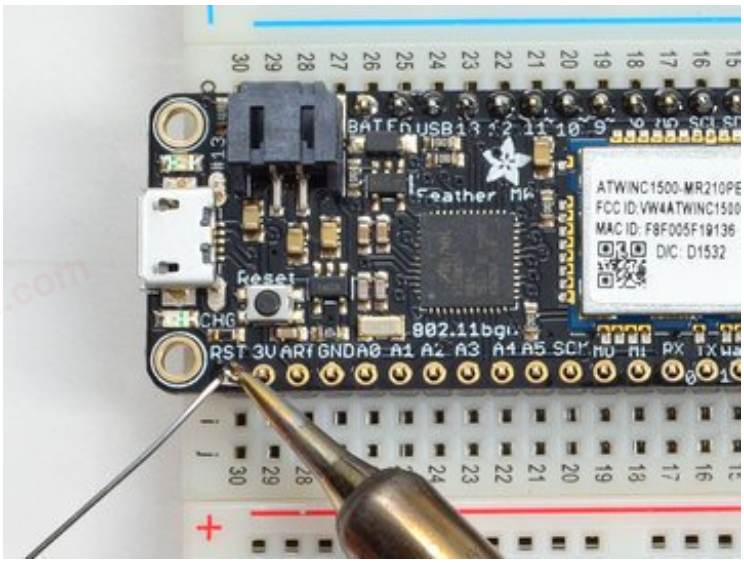
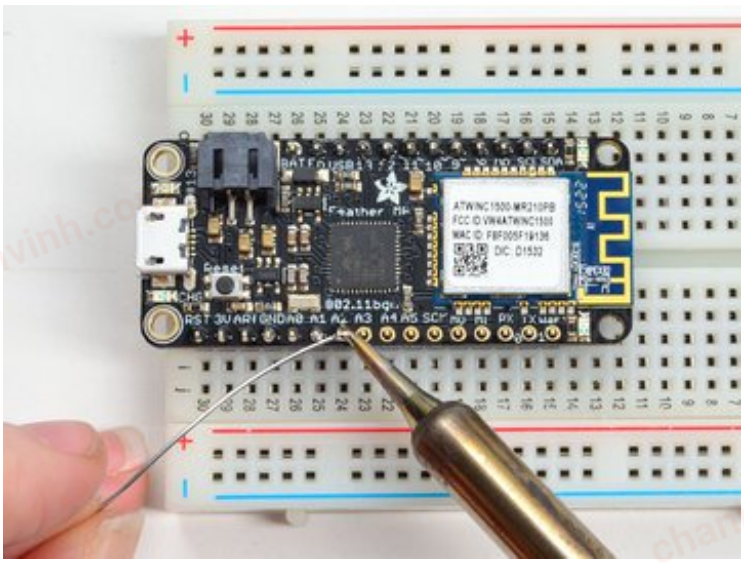
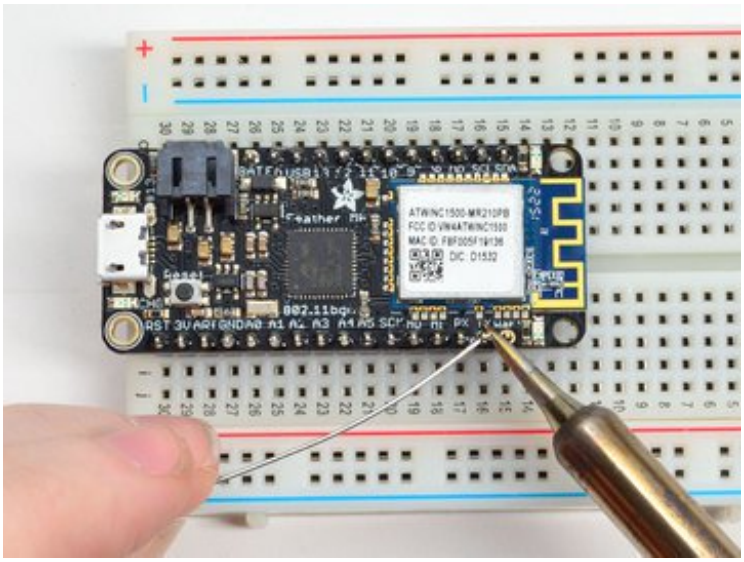
- 

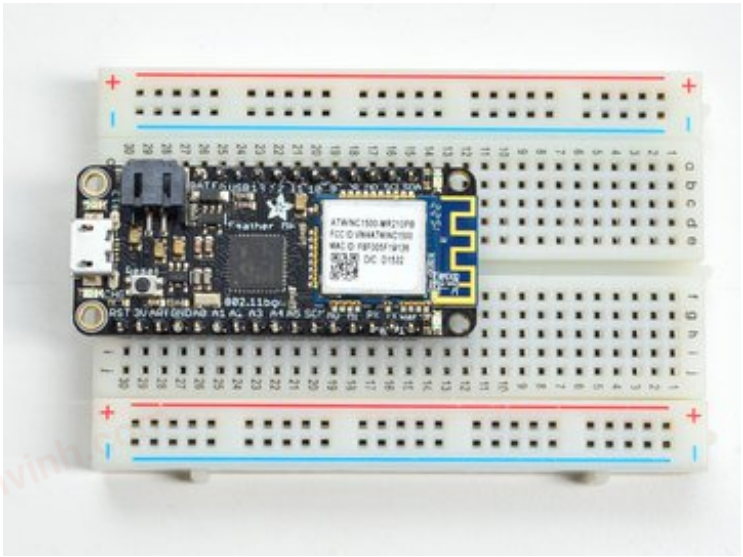
And Solder!

Be sure to solder all pins for reliable electrical contact.

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

- 

- 
- 
  Solder the other strip as well.
- 

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

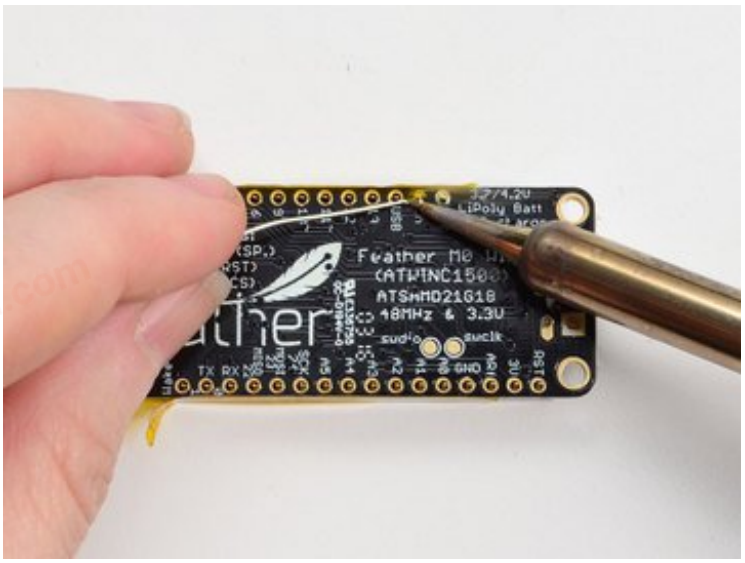
Soldering on Female Header

- 

Tape In Place

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

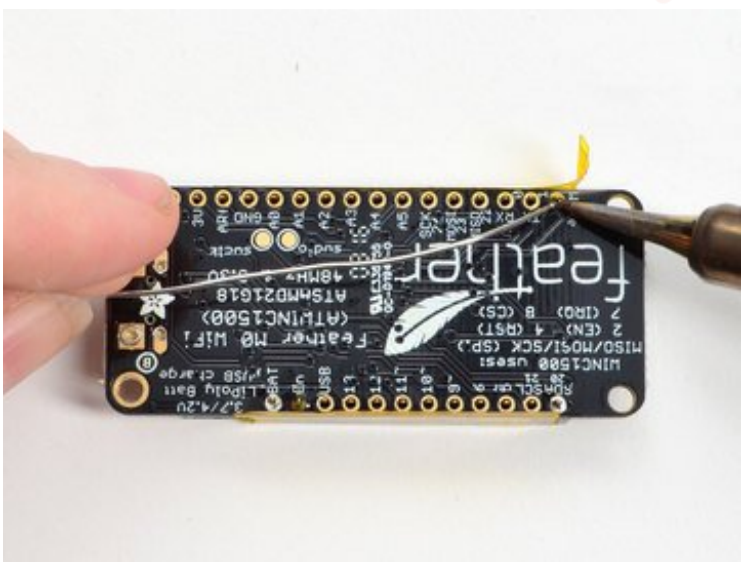
chanhvinh.com

- 

Flip & Tack Solder

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

- 

- 

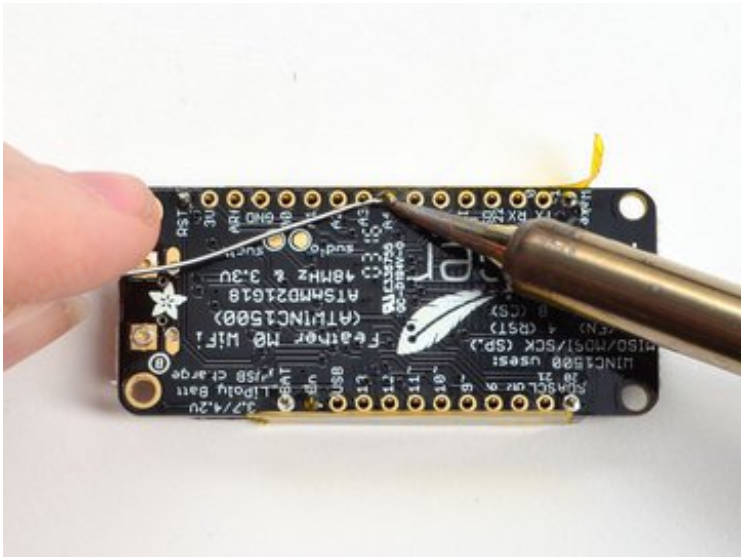
chanhvinh.com

chanhvinh.com

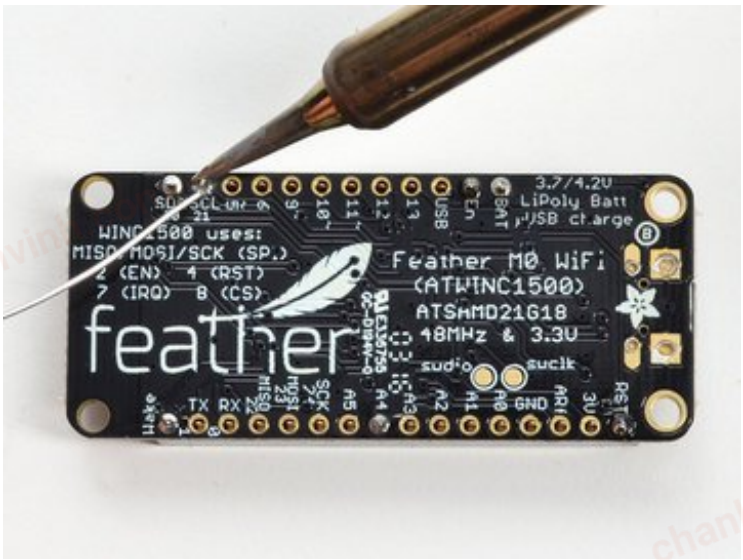
And Solder!

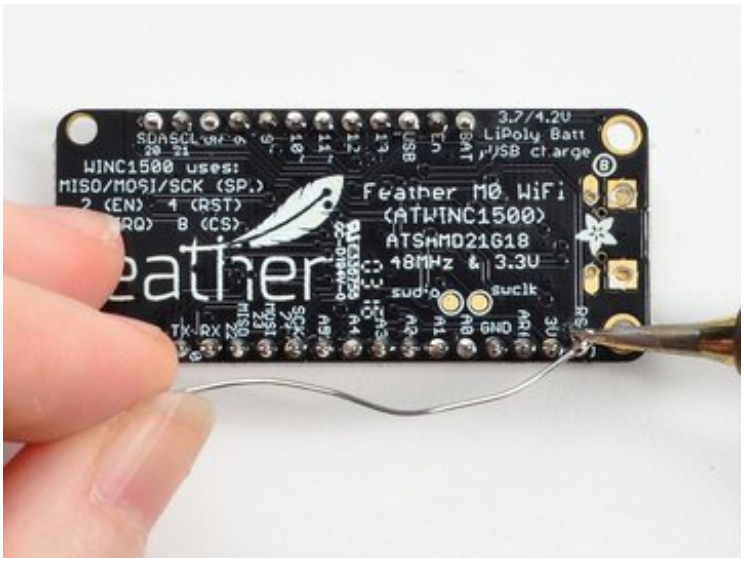
Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to

- 

check out our [Guide to Excellent Soldering](http://adafru.it/aTk).

- 

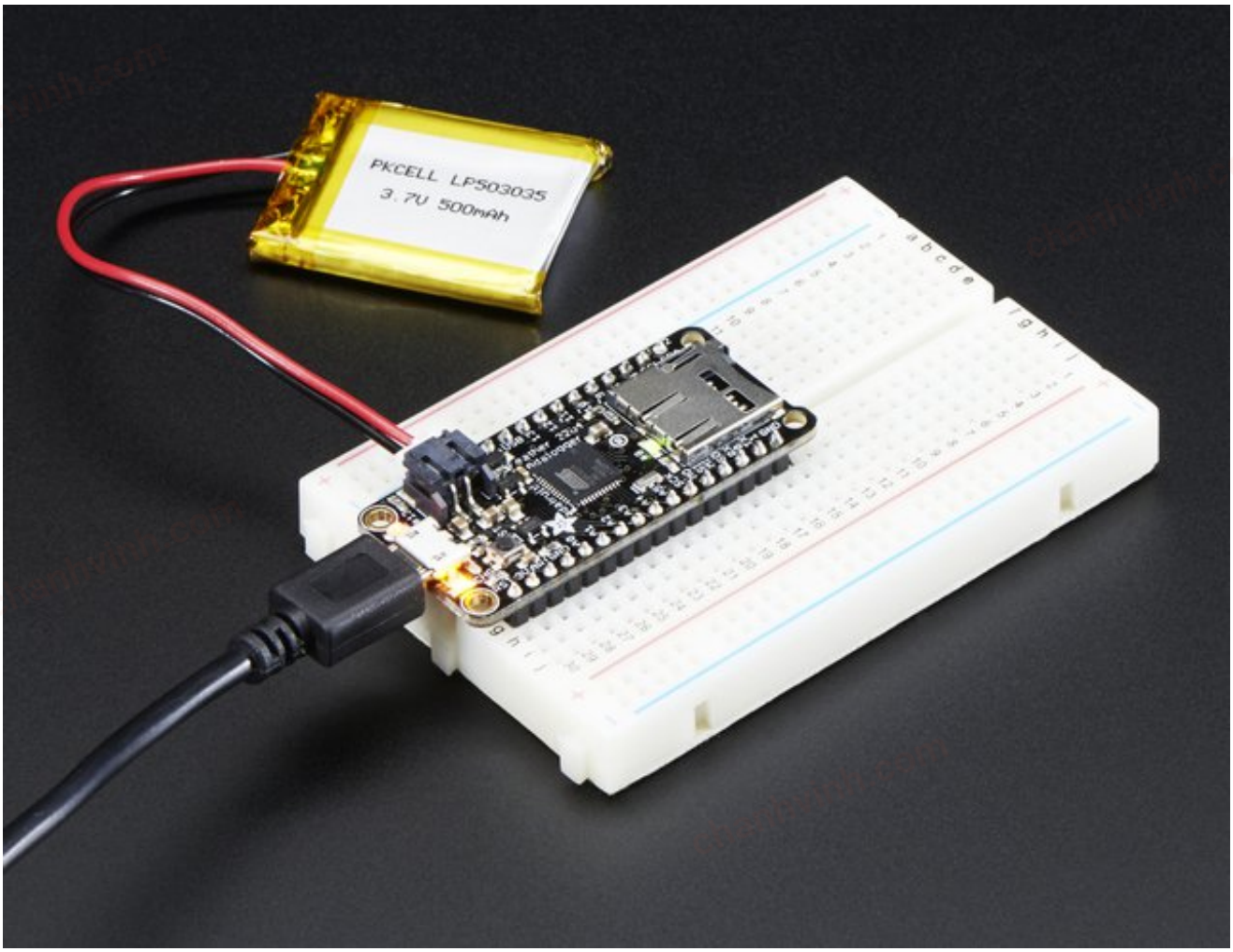
- 

- 

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

- 

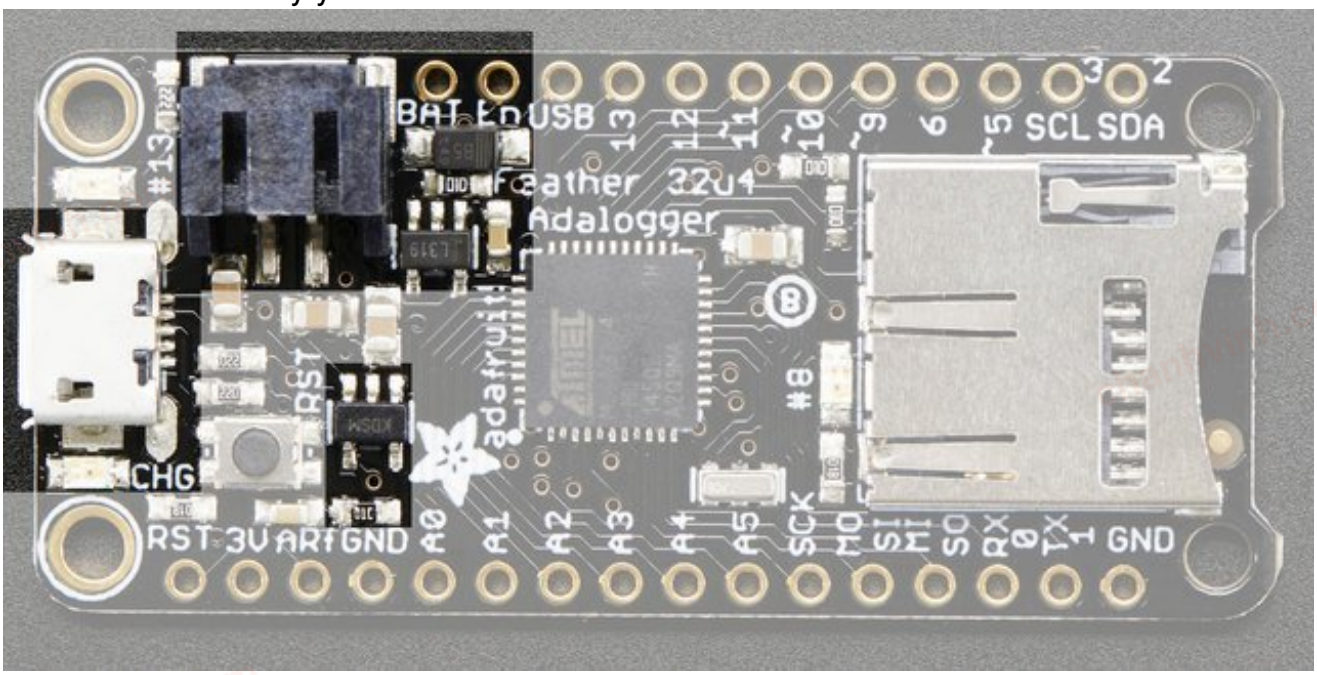
Power Management



Battery + USB Power

We wanted to make the Feather easy to power both when connected to a computer as well as via battery. There's **two ways to power** a Feather. You can connect with a MicroUSB cable (just plug into the jack) and the Feather will regulate the 5V USB down to 3.3V. You can also connect a 4.2/3.7V Lithium Polymer (Lipo/Lipoly) or Lithium Ion (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.

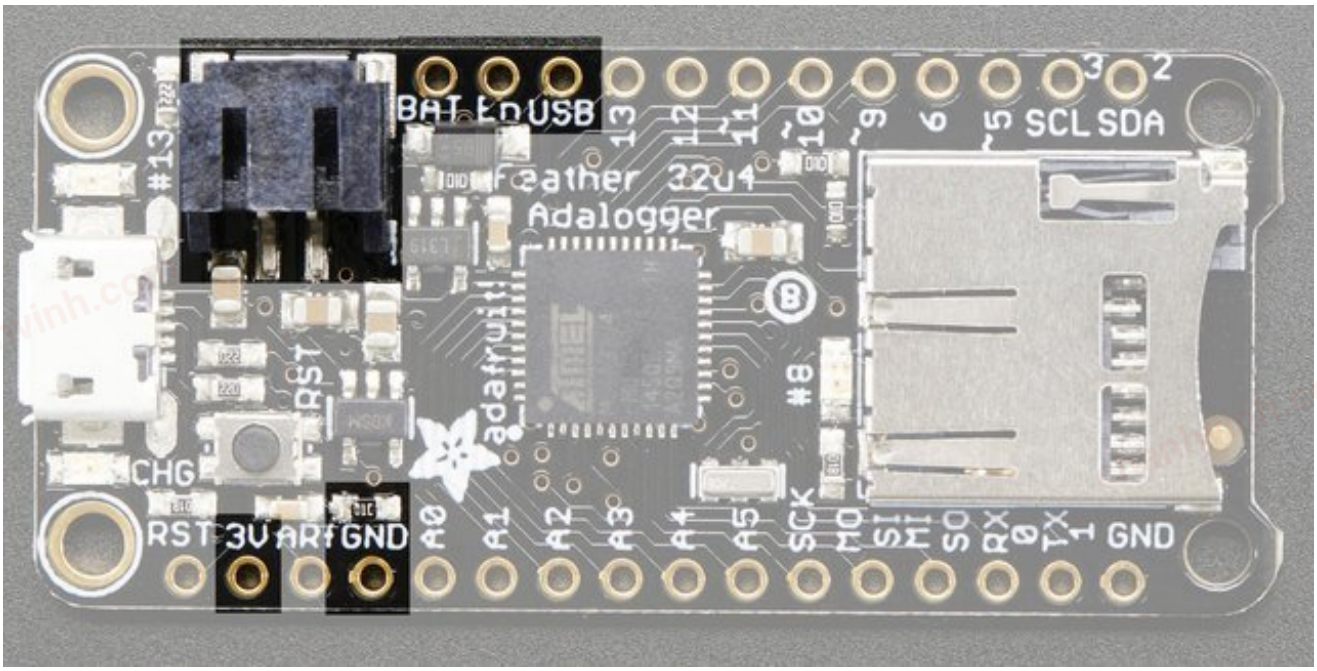
The JST connector polarity is matched to Adafruit LiPoly batteries. Using wrong polarity batteries can destroy your Feather



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 AP2112. 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 'spiky' & sporadic.



Measuring Battery

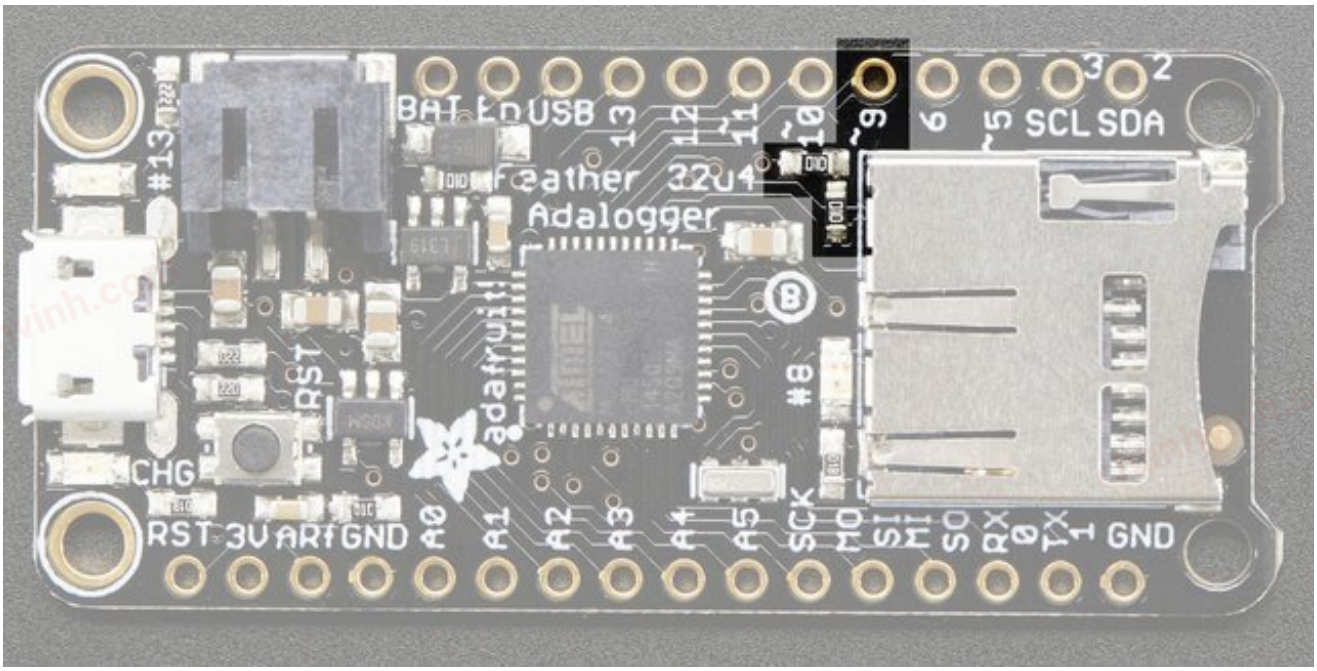
If you're running off of a battery, chances are you wanna know what the voltage is at! That way you can tell when the battery needs recharging. Lipoly batteries are 'maxed out' at 4.2V and stick around 3.7V for much of the battery life, then slowly sink down to 3.2V or so before the protection circuitry 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 A9

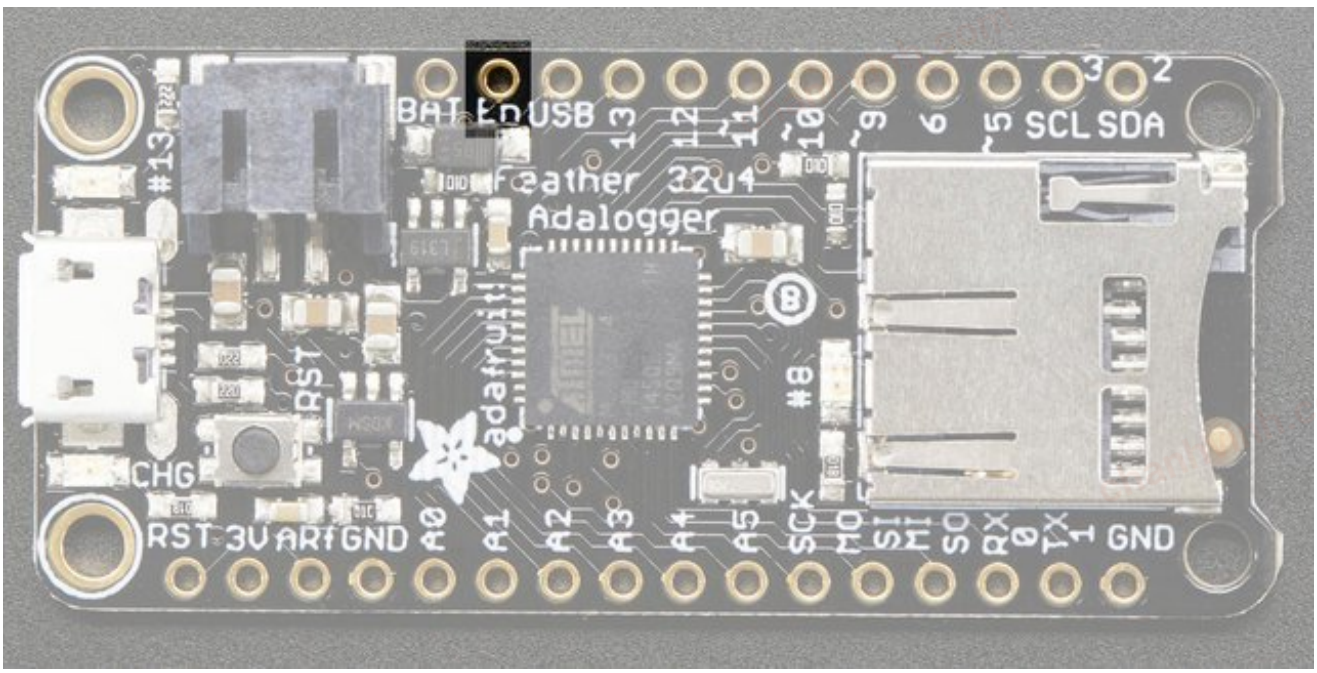
float measuredvbat = analogRead(VBATPIN);
measuredvbat *= 2;    // we divided by 2, so multiply back
measuredvbat *= 3.3;  // Multiply by 3.3V, our reference voltage
measuredvbat /= 1024; // convert to voltage
Serial.print("VBat: " ); Serial.println(measuredvbat);
```

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)



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

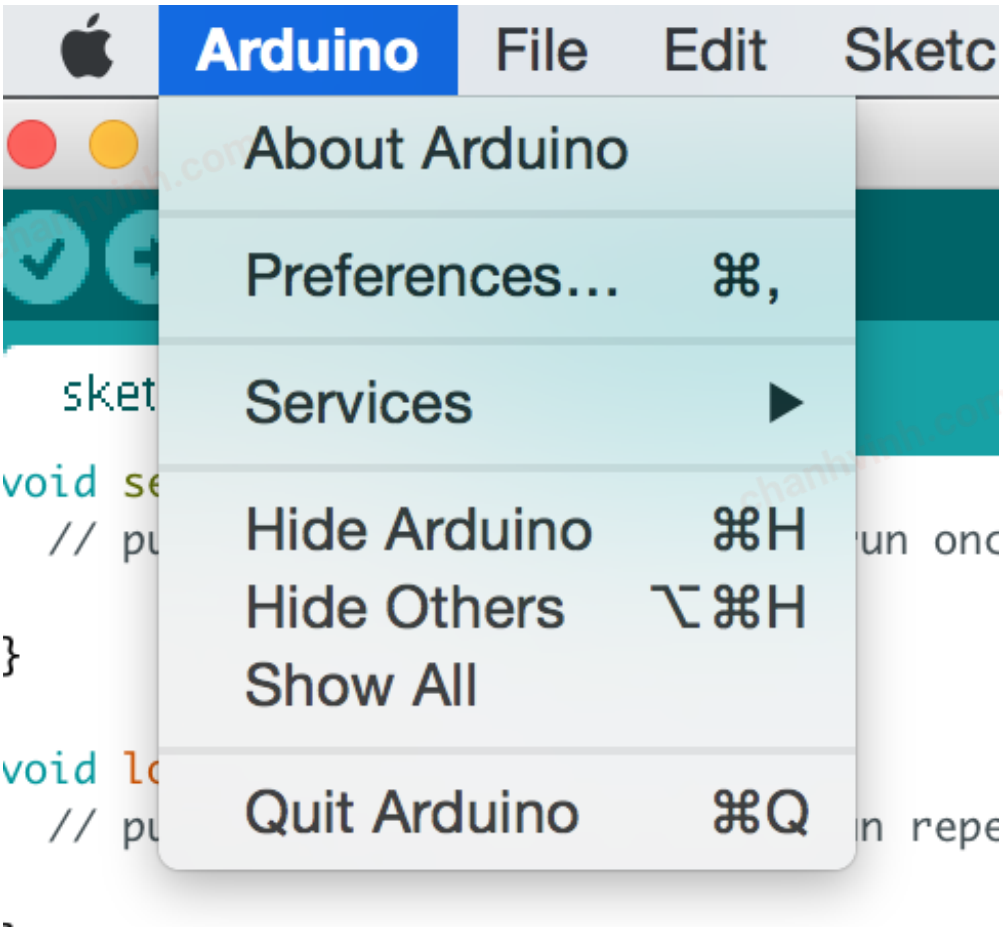


Arduino IDE Setup

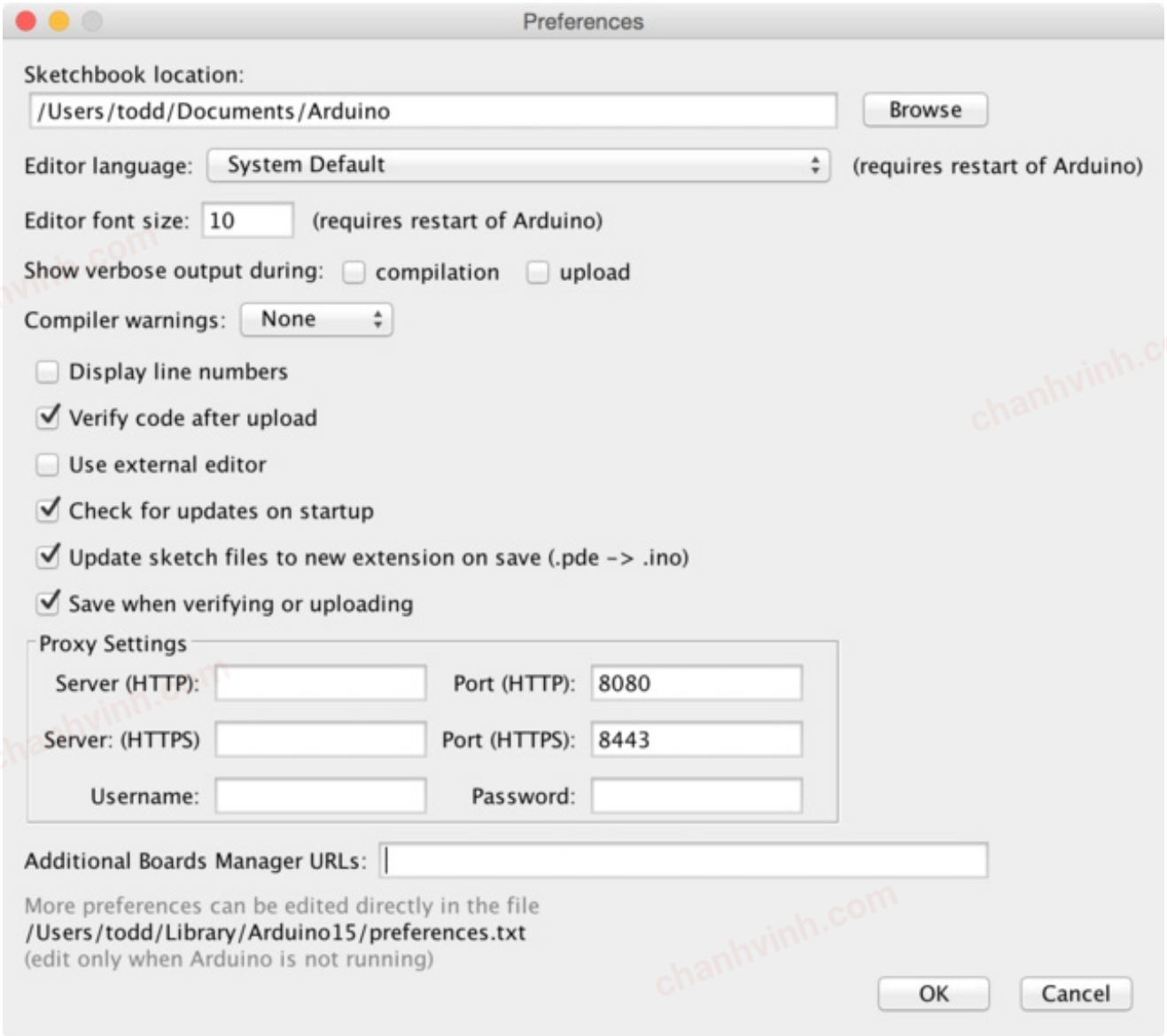
The first thing you will need to do is to download the latest release of the Arduino IDE. You will need to be using **version 1.6.4** or higher for this guide.

Arduino IDE v1.6.4+ Download
http://adafru.it/f1P

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



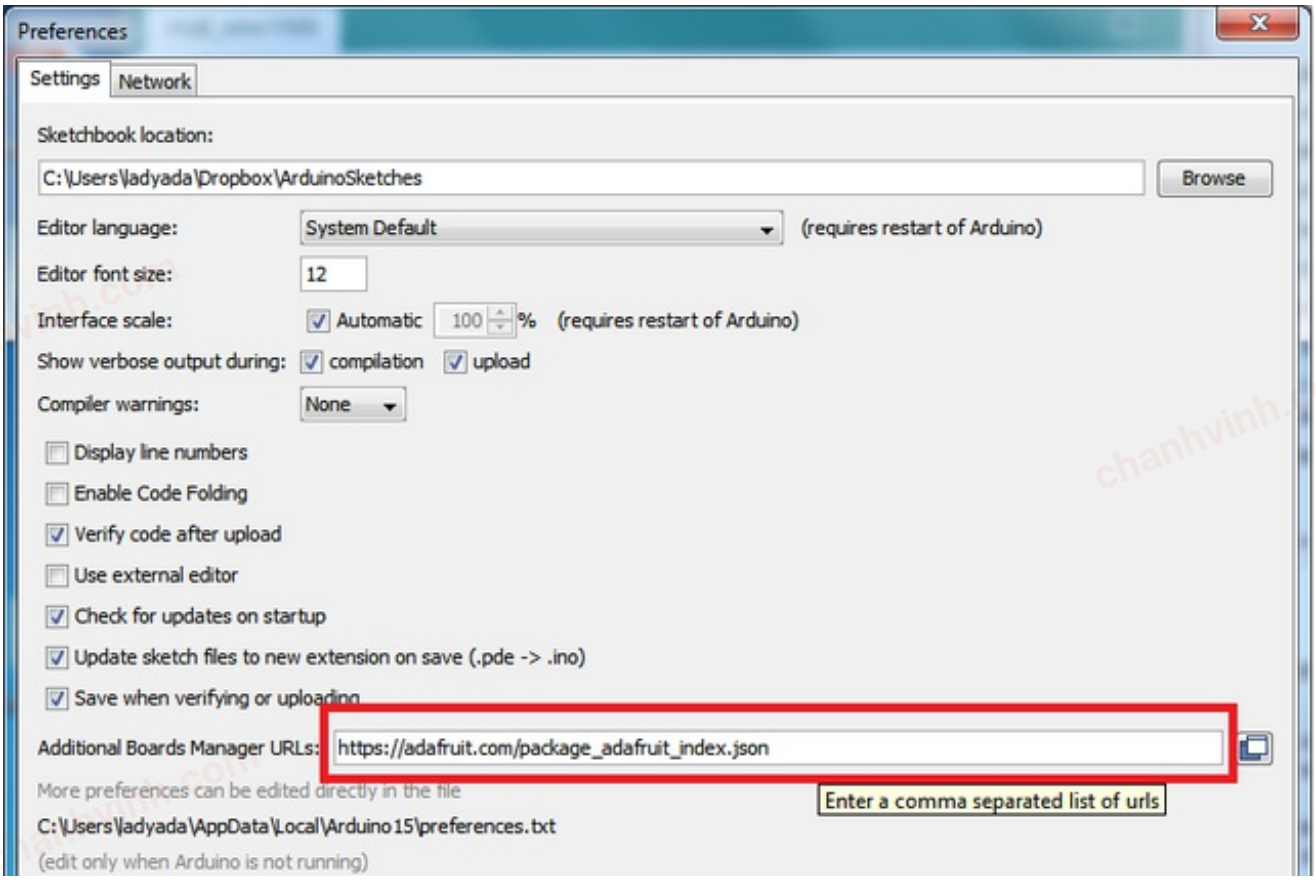
A dialog will pop up just like the one shown below.



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

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

https://adafruit.github.io/arduino-board-index/package_adafruit_index.json



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

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

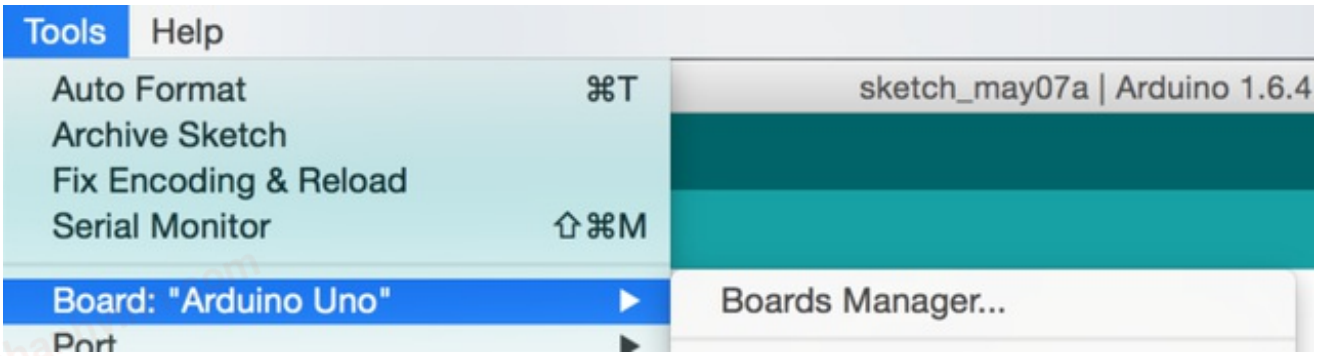
If you have multiple boards you want to support, say ESP8266 and Adafruit, have both URLs in the text box separated by a comma (,)

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

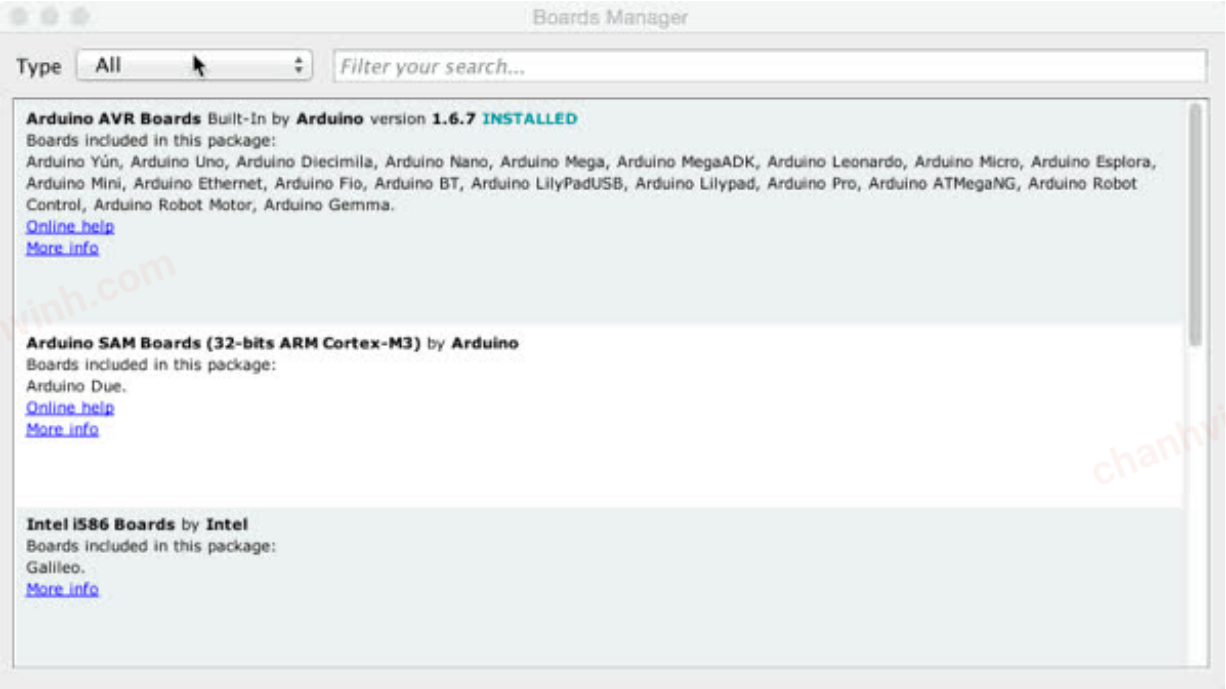
Using with Arduino IDE

Since the Feather 32u4 uses an ATmega32u4 chip running at 8 MHz, you can pretty easily get it working with the Arduino IDE. Many libraries (including the popular ones like NeoPixels and display) work great with the '32u4 and 8 MHz clock speed.

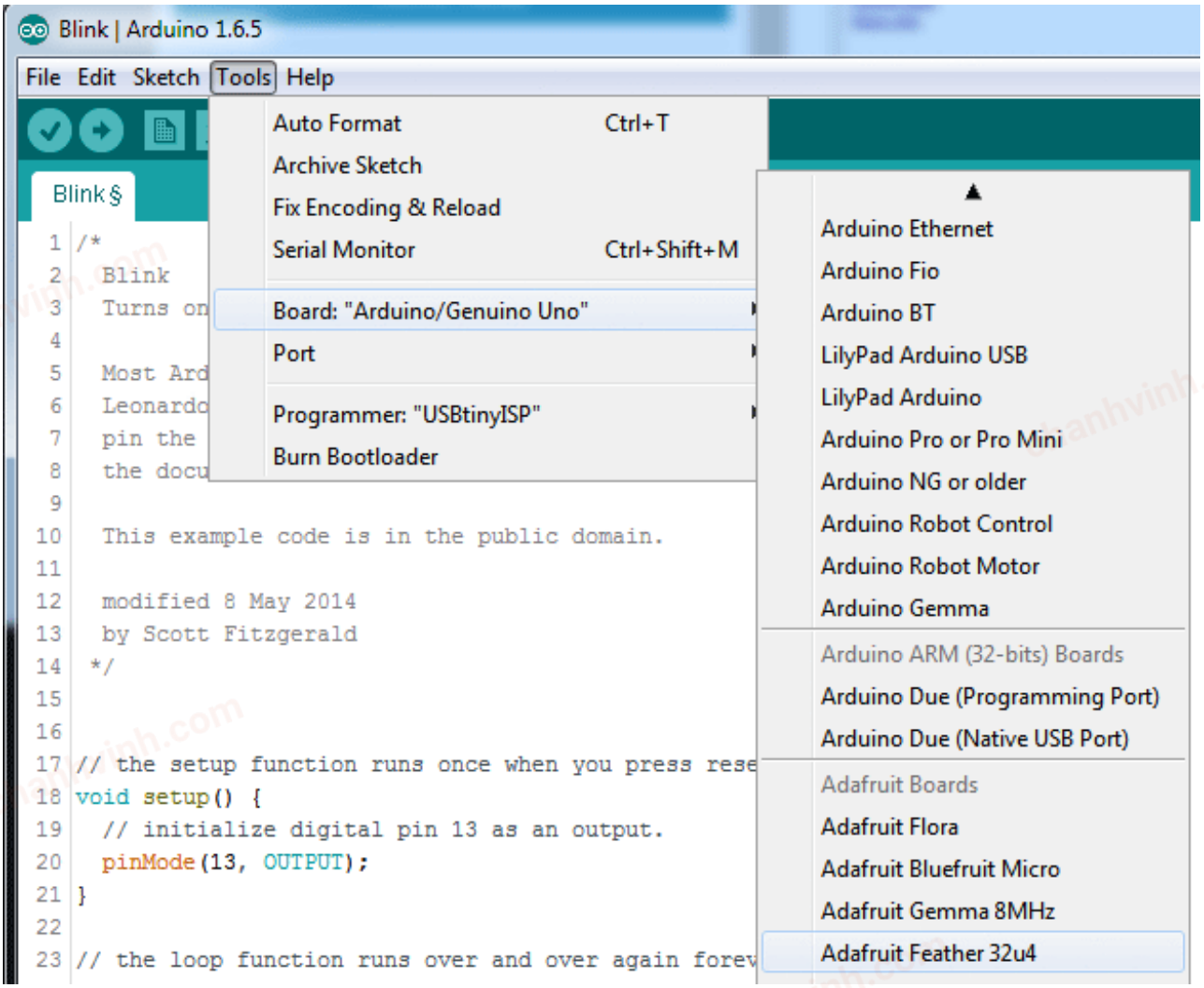
Now that you have added the appropriate URLs to the Arduino IDE preferences, you can open the **Boards Manager** by navigating to the **Tools->Board** menu.



Once the Board Manager opens, click on the category drop down menu on the top left hand side of the window and select **Contributed**. You will then be able to select and install the boards supplied by the URLs added to the prefrences. In the example below, we are installing support for ***Adafruit AVR Boards***, but the same applies to all boards installed with the Board Manager.



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



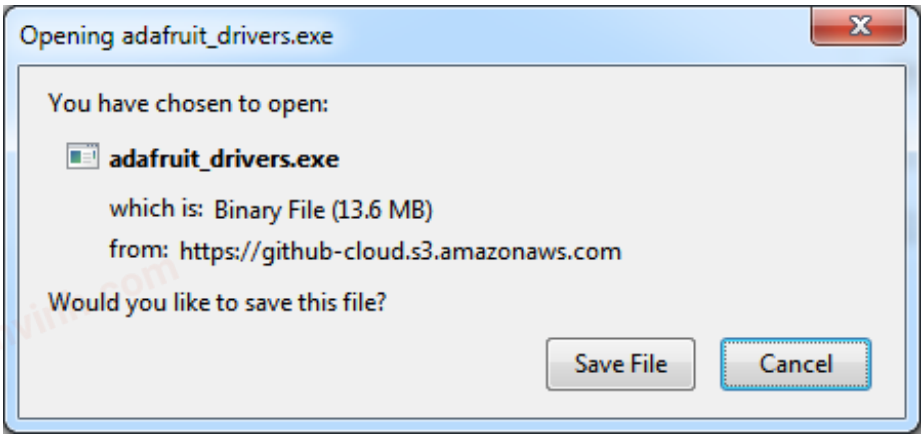
Install Drivers (Windows Only)

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

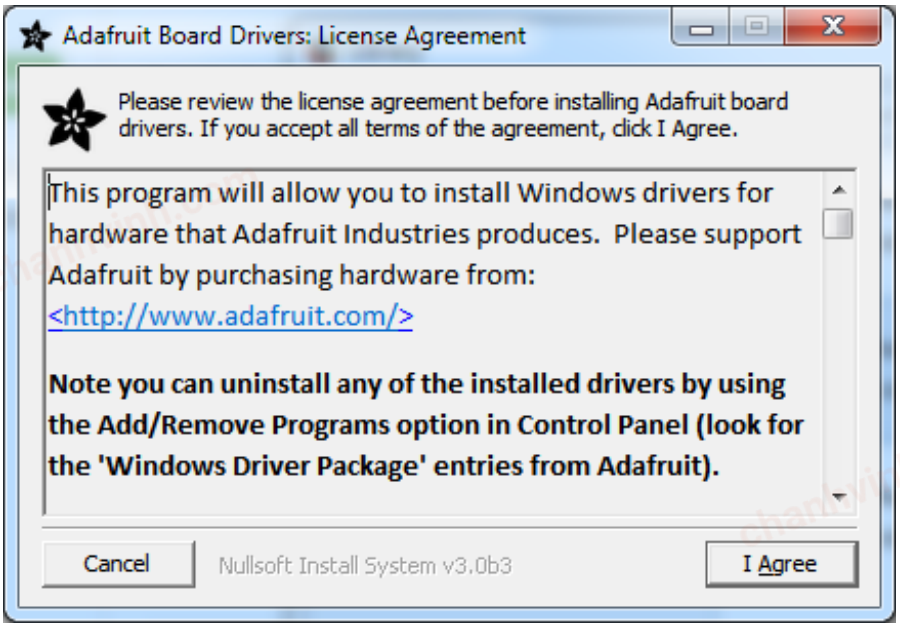
Click below to download our Driver Installer

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

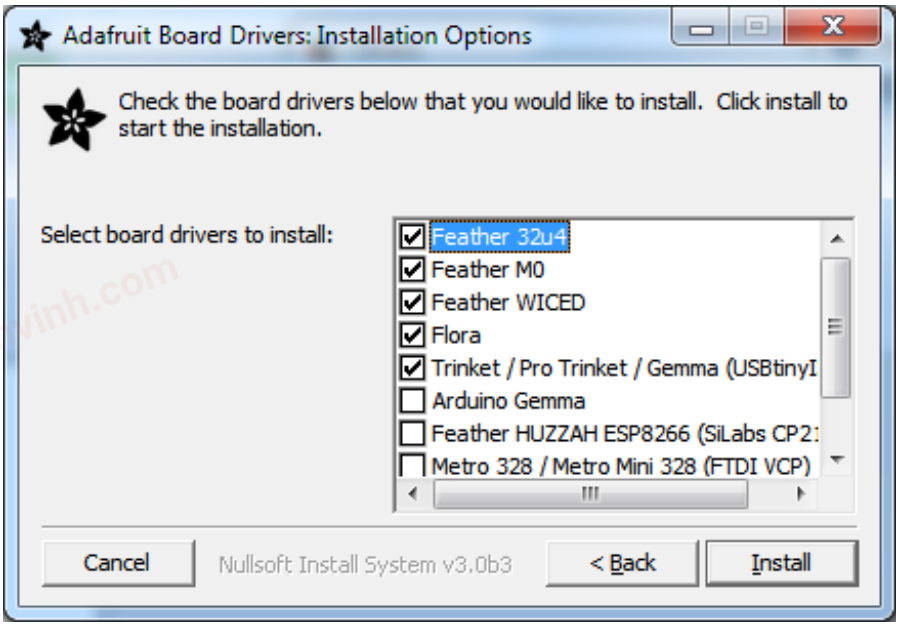
Download and run the installer



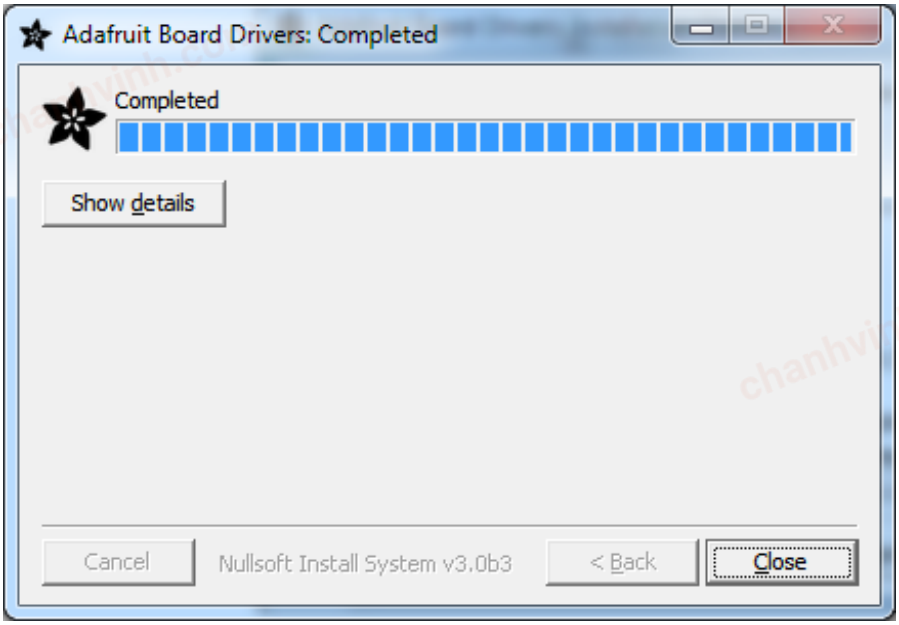
Run the installer! Since we bundle the SiLabs and FTDI drivers as well, you'll need to click through the license



Select which drivers you want to install:



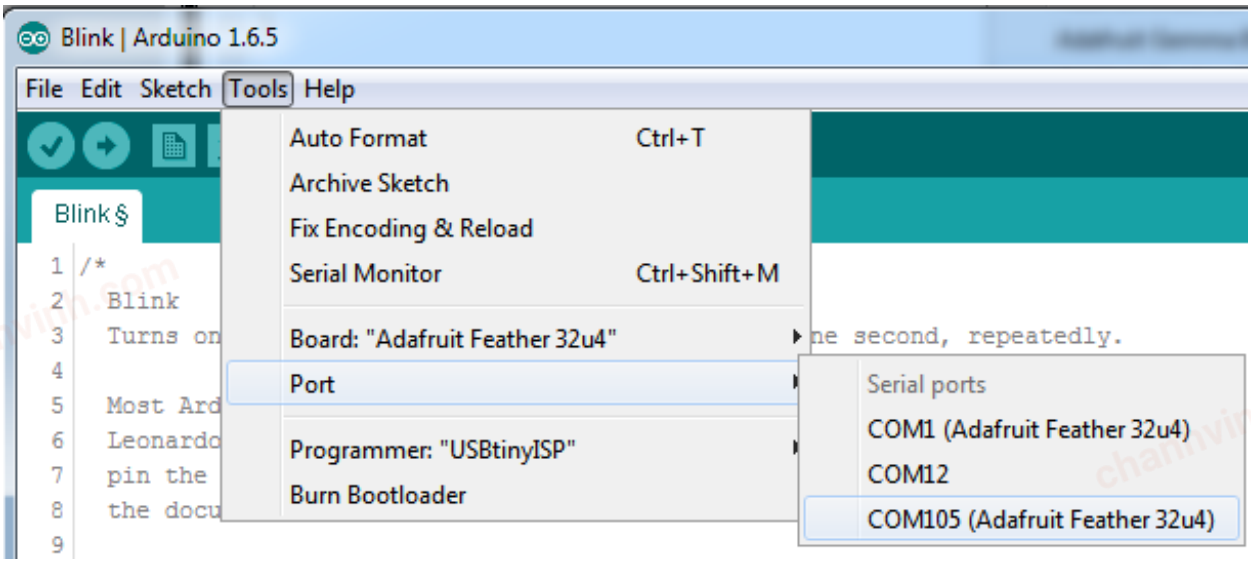
Click **Install** to do the installin'



Blink

Now you can upload your first blink sketch!

Plug in the Feather 32u4 and wait for it to be recognized by the OS (just takes a few seconds). It will create a serial/COM port, you can now select it from the dropdown, it'll even be 'indicated' as Feather 32u4!



Now load up the Blink example

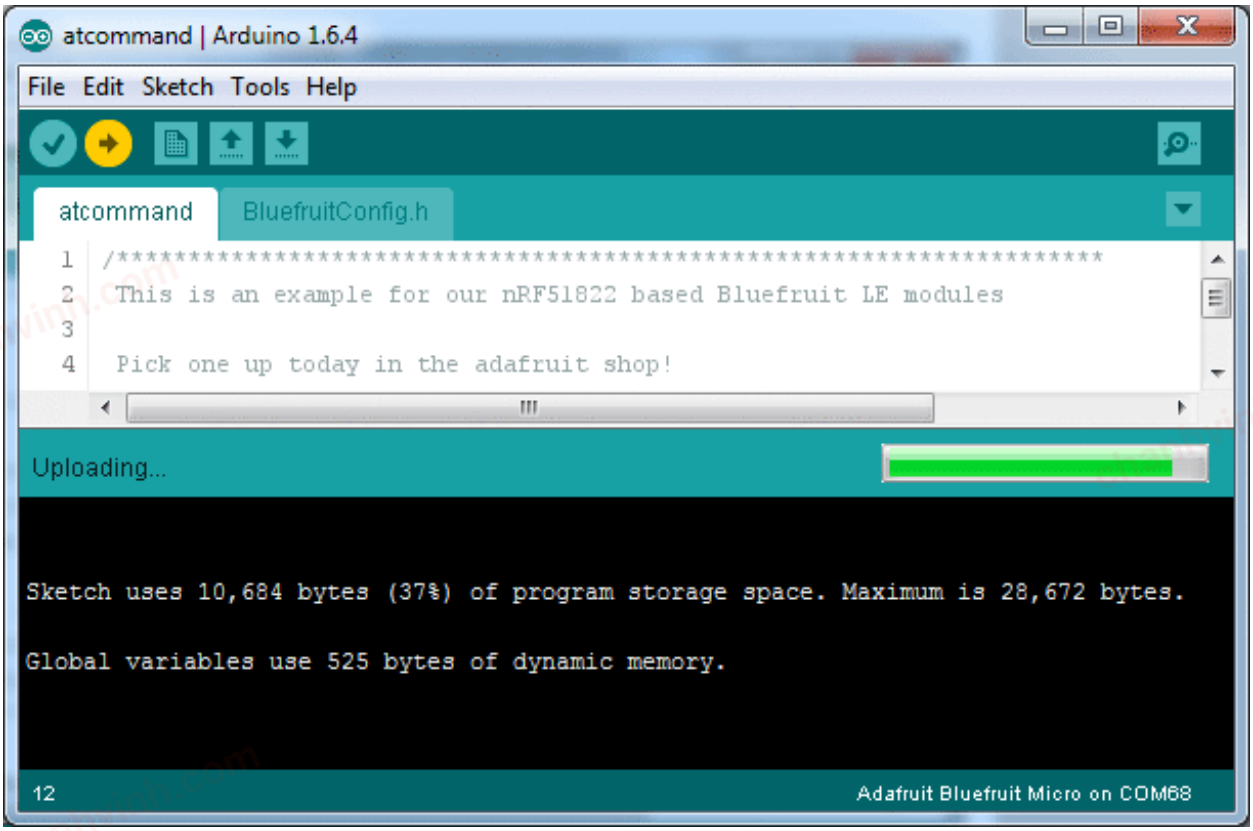
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// 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.

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 to get back into the bootloader. The red LED will pulse, so you know that its in bootloader mode. Do the reset button press right as the Arduino IDE says its attempting to upload the sketch, when you see the Yellow Arrow lit and the **Uploading...** text in the status bar.



Don't click the reset button **before** uploading, unlike other bootloaders you want this one to run at the time Arduino is trying to upload

Ubuntu & Linux Issue Fix

Note if you're using Ubuntu 15.04 (or perhaps other more recent Linux distributions) there is an issue with the modem manager service which causes the Bluefruit LE micro to be difficult to program. If you run into errors like "device or resource busy", "bad file descriptor", or "port is busy" when attempting to program then you are hitting this issue. (http://adafru.it/fP6)

The fix for this issue is to make sure Adafruit's custom udev rules are applied to your system. One of these rules is made to configure modem manager not to touch the Bluefruit Micro board and will fix the programming difficulty issue. Follow the steps for installing Adafruit's udev rules on this page. (http://adafru.it/iOE)

HELP!

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

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

Seriously, we get upwards of 5 complaints a day that turn out to be due to charge-only cables

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

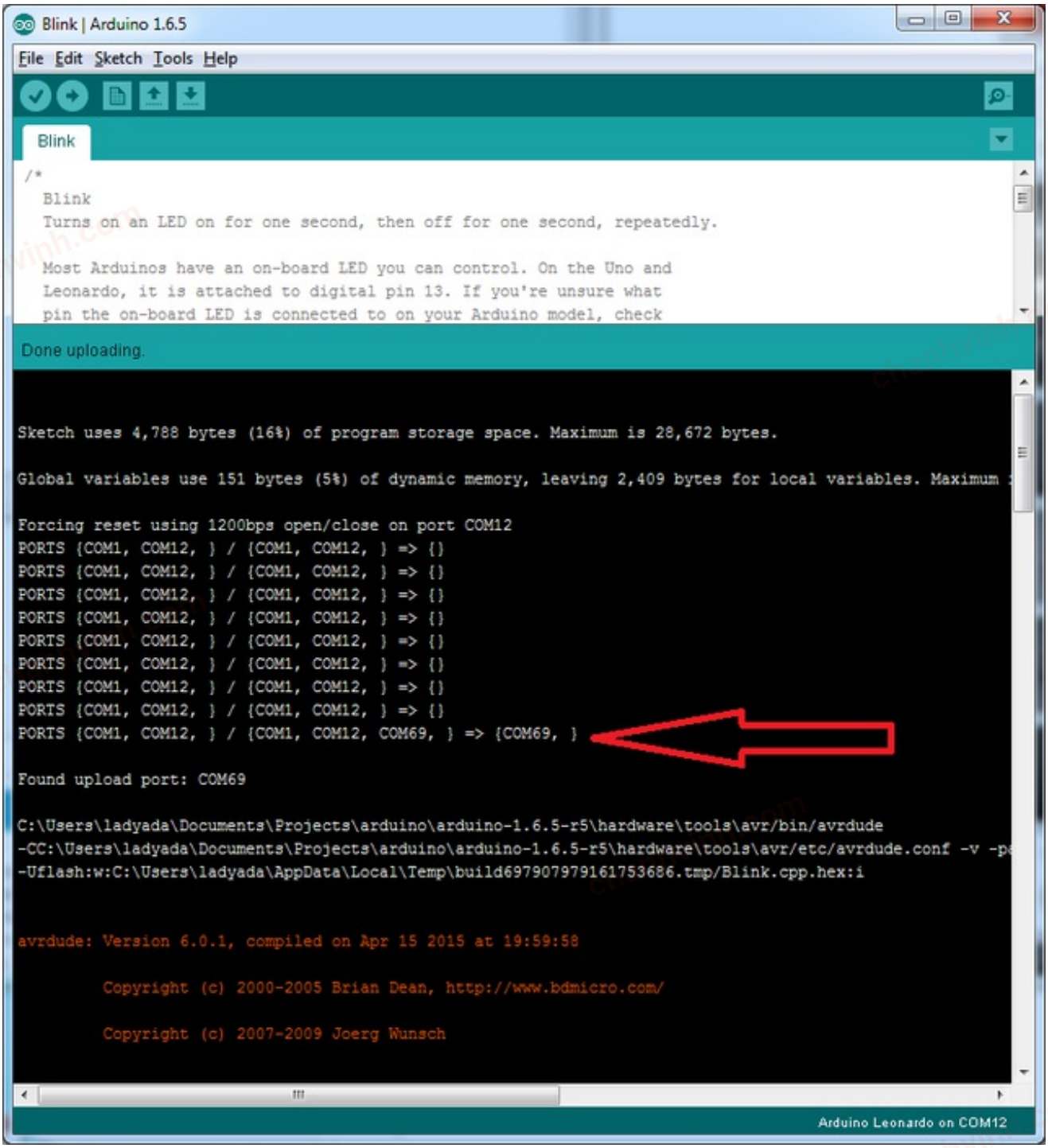
If you have any charge-only cables, cut them in half throw them out. We are serious! They tend to be low quality in general, and will only confuse you and others later, just get a good data+charge USB cable

Ack! I "did something" and now when I plug in the Feather 32u4 it doesn't show up as a device anymore so I cant upload to it or fix it...

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

1. Turn on **verbose upload** in the Arduino IDE preferences
2. Plug in feather 32u4, 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
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, press and release the reset button, you'll see the red pulsing LED that tells you its now in bootloading mode**
8. The Feather 32u4 will show up as the Bootloader COM/Serial port
9. The IDE should see the bootloader COM/Serial port and upload properly

For Feather M0, same as above, but *double click* the reset button, you'll see the red pulsing LED that tells you its bootloading



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

This seems to happen when people select the wrong board from the Arduino Boards menu. Make sure you select **Feather 32u4** for ATMega32u4 based boards! Do not use anything else, do not use the 32u4 breakout board line.

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

Theres **two** COM ports you can have with the 32u4, 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

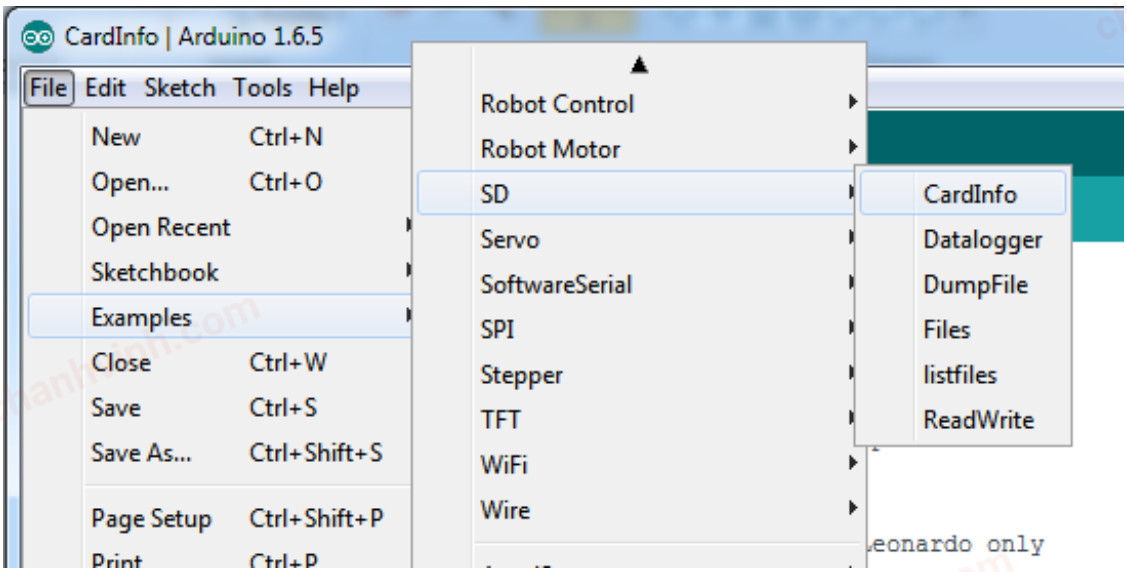
This is likely because the bootloader is not kicking in and you are accidentally**trying to upload to the wrong COM port**

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

Using the SD Card

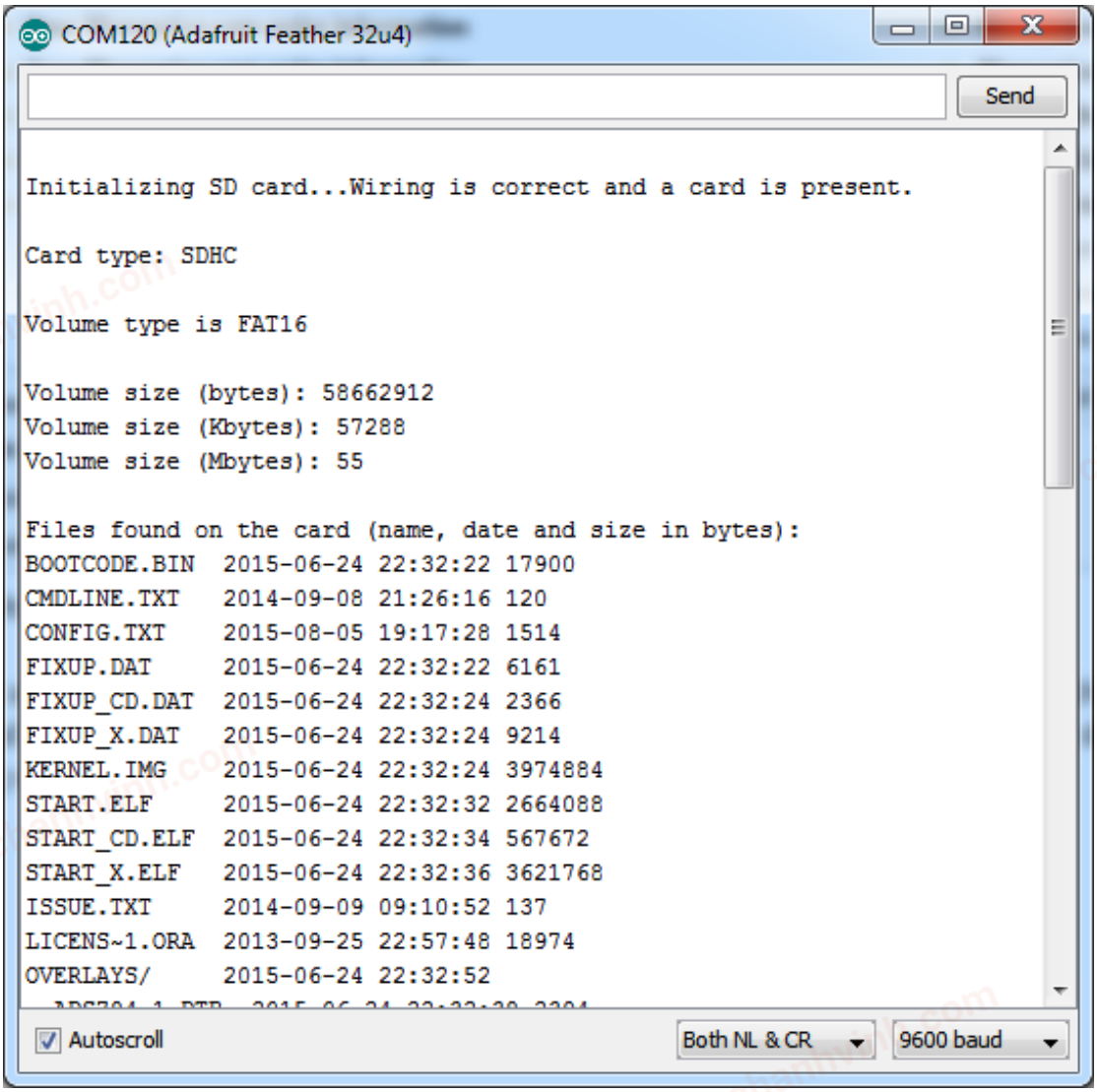
Once you have your Feather working, you probably want to rock out with some SD card reading and writing! Luckily, the Arduino IDE has an SD card library that works great, and it even comes with the IDE!

You can start with **CardInfo** which is very detailed



Luckily many of the default examples already have **chipSelect = 4** so you can just upload immediately. For other sketches, do check to make sure that CS is set to 4! The SD card uses hardware SPI for the remaining pins.

Anyhow, open up the serial console and you'll get all this info including a list of files



Once you have that working, check out the other examples, such the**Datalogger** example (saving analog data to SD card) and **Dumpfile** example (reading back data from an SD card)

Example logging sketch

If you want to try saving data to the SD card in the simplest sketch, try this example. You can adjust the **delay()** to set how often analog data is read from pin**A0** and saved to the SD card. The red LED will blink if there's an error, and the green LED will blink when data is written to the SD card.

Note that to save power, we *buffer* the data, so you will only 'save' data truely every 50 datapoints (512 total characters written)

```
#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

If you really want to make sure you save every data point, put a

```
logfile.flush();
```

right after the **logfile.print**'s however this will cause the adalogger to draw a lot more

power, maybe about 3x as much on average (30mA avg rather than about 10mA)

chanhvinh.com

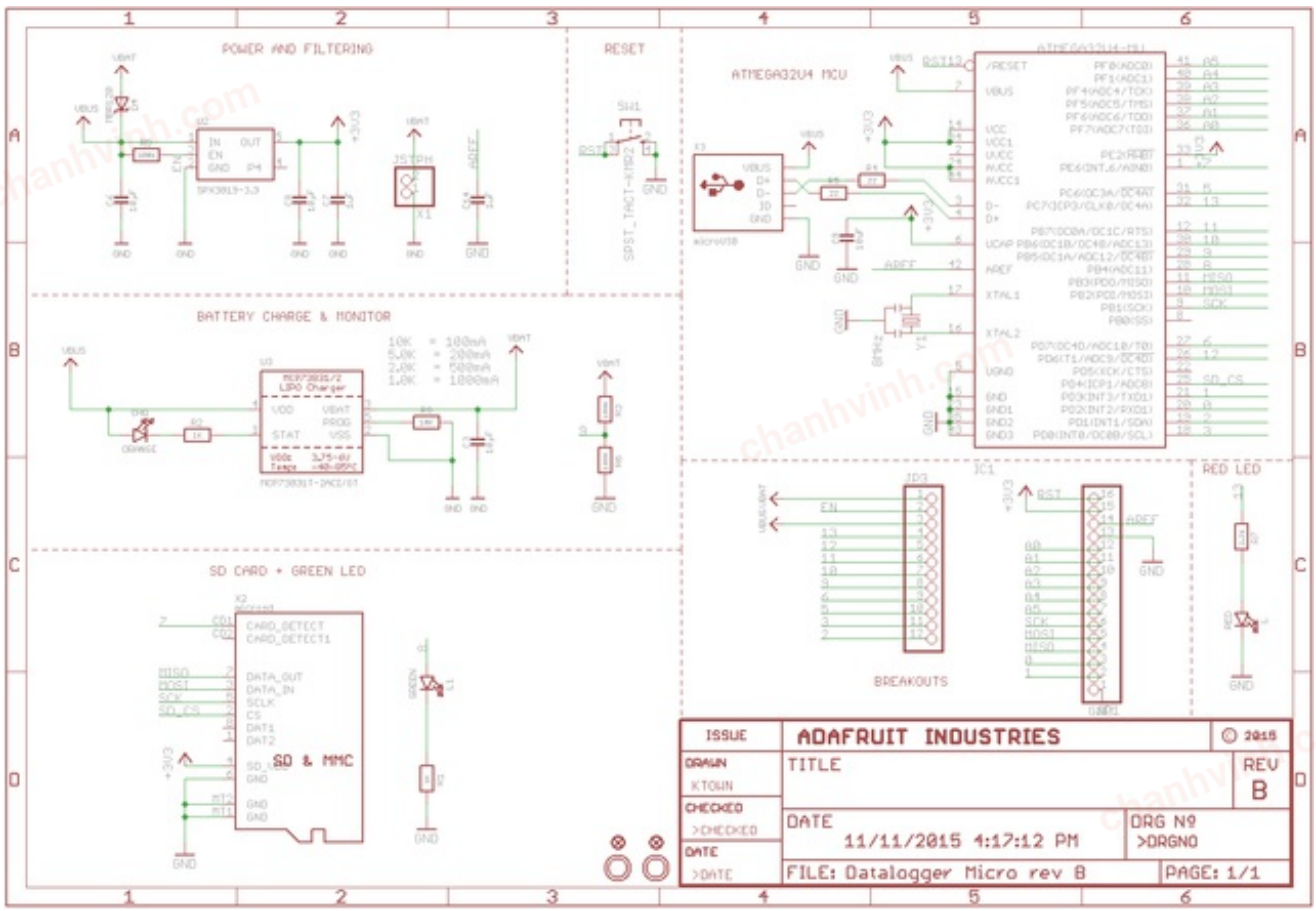
Downloads

Feather 32u4 Adalogger Pinout Diagram
http://adafru.it/lMc

- Frizting object in the Adafruit Fritzing Library (http://adafru.it/aP3)
- EagleCAD PCB files in GitHub (http://adafru.it/qxf)

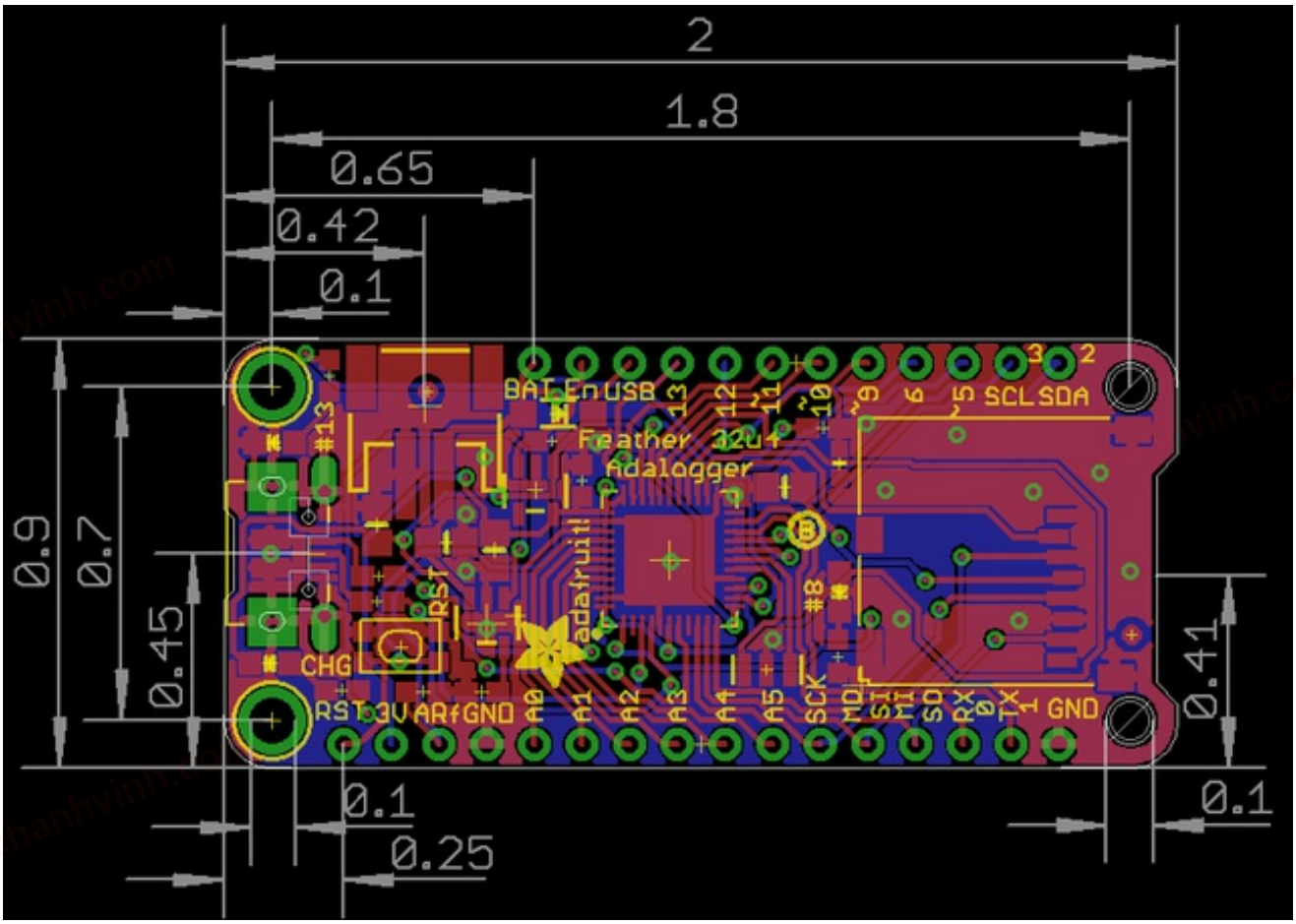
Schematic

Click to enlarge



Fabrication Print

Dimensions in inches



chanhvinh.com