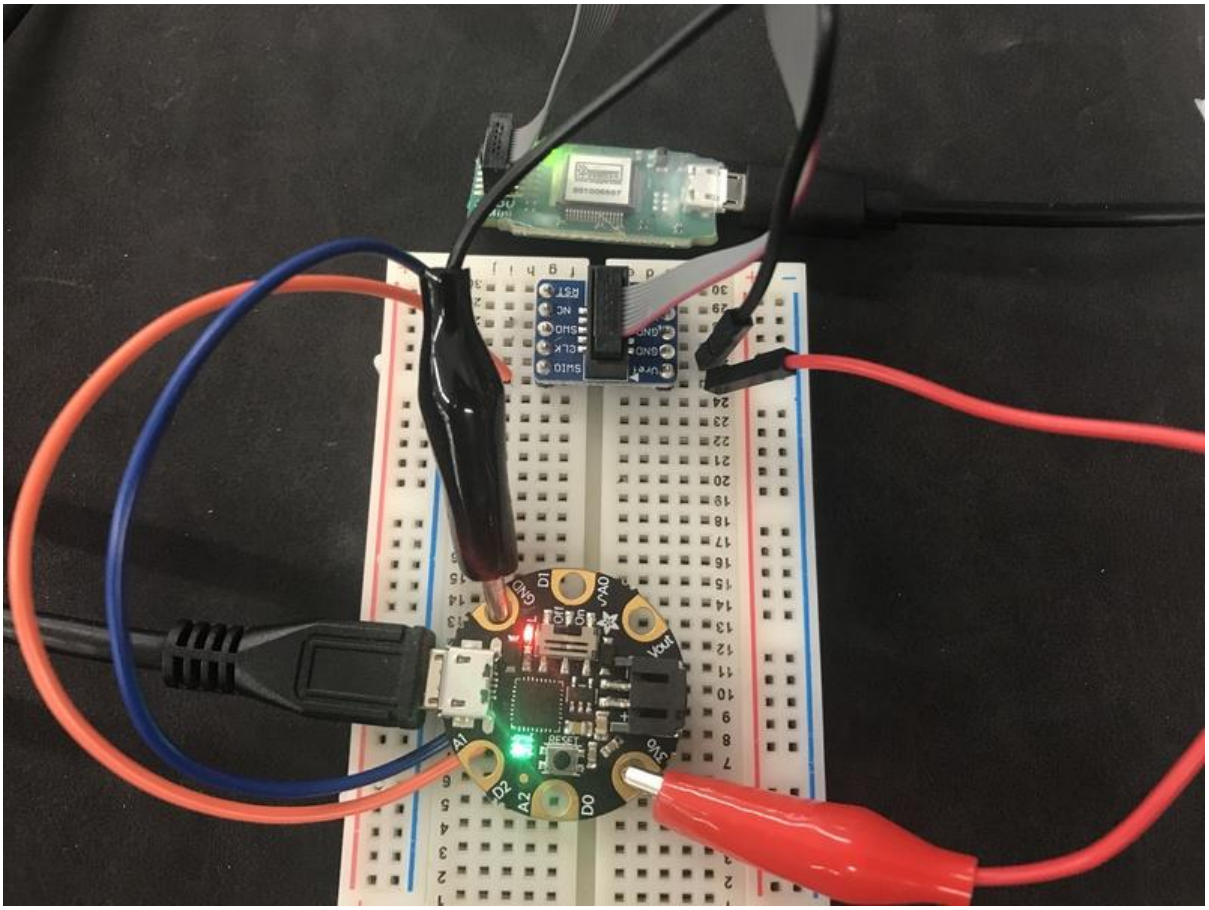
How to Program SAMD Bootloaders

Created by Brent Rubell

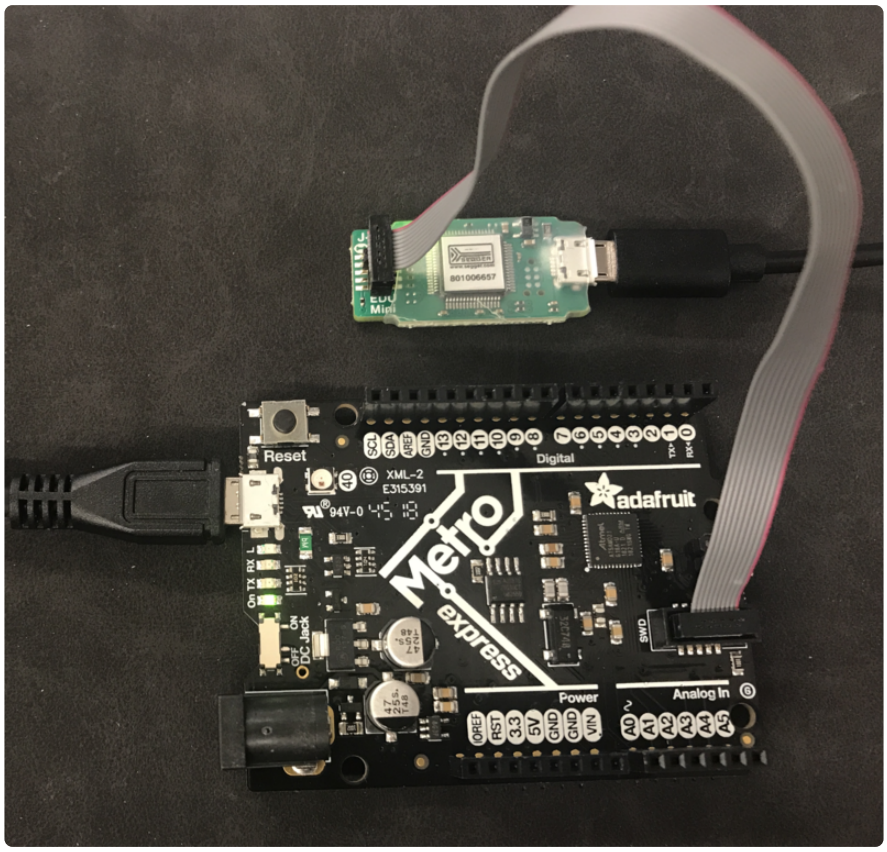


https://learn.adafruit.com/how-to-program-samd-bootloaders

Last updated on 2021-11-15 07:30:43 PM EST

Table of Contents

Overview



If you have a functioning bootloader on your board, you do not need to use the techniques described in this Guide to update the bootloader. Instead, refer to the Guide for your board about how to update.

Do you have a bricked Adafruit SAMD board that won't boot into CircuitPython, or show up as a boot volume? Are you building your own SAMD board and want to flash our UF2-SAMD bootloader onto it?

This guide will cover wiring a J-Link to a SAMD board, flashing the bootloader, and (optionally) installing the latest CircuitPython build.

This process does require extra hardware and some software installation time. It is unfortunate when a microcontroller's firmware is corrupted - it does not happen often. But, rather than buy a new board and have one sitting, this process will get your original board back to 100%.

About the SAMD UF2 Bootloader

You will need to program the Adafruit UF2-SAMD Bootloader (https://adafru.it/Dj0) onto the affected board. Adafruit SAMD21 (M0) and SAMD51 (M4) boards feature an improved bootloader that makes it easier than ever to flash different code onto the microcontroller. This bootloader makes it easy to switch between Microsoft MakeCode, CircuitPython and Arduino.

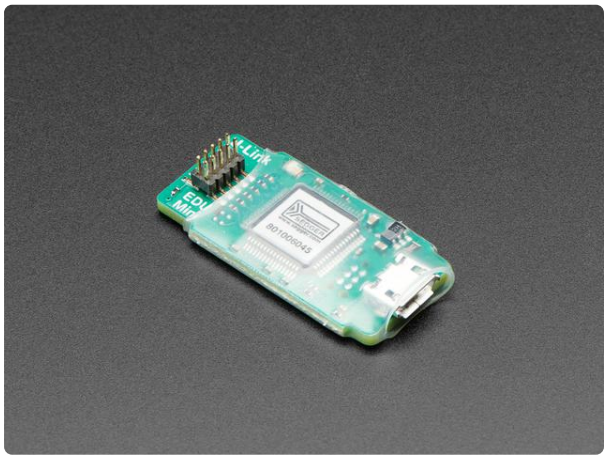
Instead of needing drivers or a separate program for flashing (say, `bossac`, `jlink` or `avrdude`), one can simply drag a UF2 file onto a removable drive.

Parts

To flash the bootloader, you'll need a JTAG/SWD debugger. We suggest the J-Link EDU Mini. This version is smaller (the size of a USB drive) and less expensive than the full-sized J-Link EDU, BUT it is for non-commerical use only.

What does that mean?

Basically, if you're making money (or plan to make money) off your project, you'll need to order the full commercial version, or find a different debugger that suits your needs and budget better. But if you're working on personal, non-commercial projects, such as publishing some open source designs you're not selling yourself, you're good. You don't need to be a student, and you can even be a paid engineer during the week, using this on the weekend for personal non-commercial projects. As long are your intentions are non-commercial, the J-Link EDU is an excellent choice!



SEGGER J-Link EDU Mini - JTAG/SWD Debugger
Doing some serious development on any ARM-based platform, and tired of 'printf' plus an LED to debug? A proper JTAG/SWD HW debugger can make debugging more of a pleasure and...
https://www.adafruit.com/product/3571

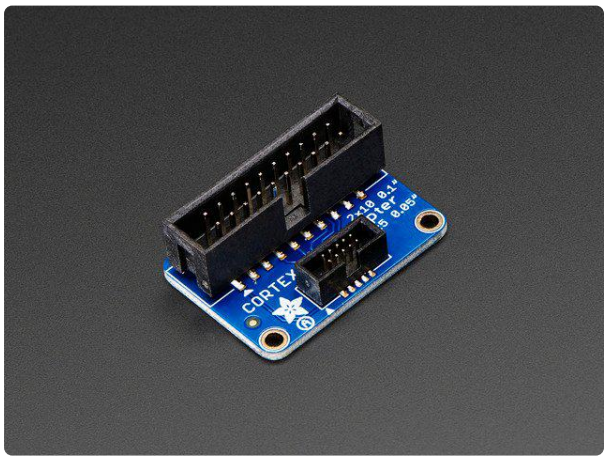
We also carry the full-sized J-Link EDU (https://adafru.it/e9G)and the J-Link base (https://adafru.it/e5q) (this model is for commercial use).

If you're a commercial user (not educational/home hobby) - you must use the commercial J-Link Base

SEGGER J-Link BASE - JTAG/SWD Debugger
The SEGGER J-Link BASE is identical to the cheaper J-Link EDU model except for the terms of...
https://www.adafruit.com/product/2209

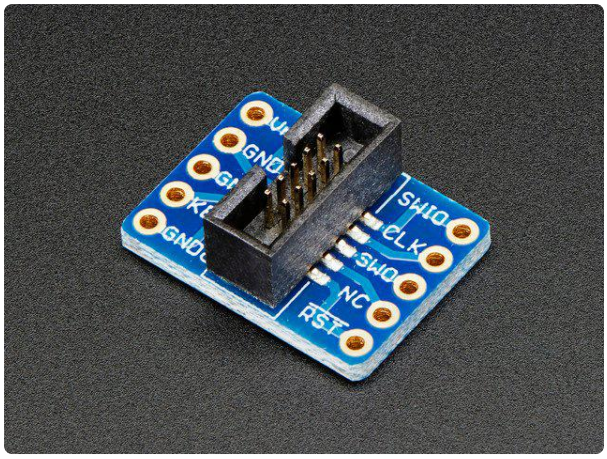
The process for using the J-LINK models is identical, only differing in the software you will install for the specific unit on the next page.

You'll also want to get a JTAG to SWD converter board and SWD cable (not needed for the JLink mini), and a SWD breadboard breakout



JTAG (2x10 2.54mm) to SWD (2x5 1.27mm) Cable Adapter Board
This adapter board is designed for adapting a 'classic' 2x10 (0.1"/2.54mm pitch) JTAG cable to a slimmer 2x5 (0.05"/ 1.27mm pitch) SWD Cable. It's helpful...
https://www.adafruit.com/product/2094

10-pin 2x5 Socket-Socket 1.27mm IDC (SWD) Cable - 150mm long
These little cables are handy when programming or debugging a tiny board that uses 10-pin 1.27mm (0.05") pitch SWD programming connectors. We see these connectors often on ARM...
https://www.adafruit.com/product/1675



SWD (2x5 1.27mm) Cable Breakout Board
This adapter board is designed to make it easier to use ARM dev boards that use slimmer 2x5 (0.05"/1.27mm pitch) SWD cables for programming. It's helpful for using...
https://www.adafruit.com/product/2743

Setup

Installing J-Link

Navigate to the Seger downloads page (https://adafru.it/val) and install the version of the J-Link Software and Documentation Pack for your operating system:

Grab a Bootloader

You'll also want to download a compiled bootloader binary (.bin file) for the board you're recovering. These can be found on the Adafruit/uf2-samdx1 repository (https://adafru.it/D3C):

Adafruit SAMD Bootloader Releases
https://adafru.it/D3C

(Optional) Grab the Latest CircuitPython UF2

If you want to install CircuitPython onto the UF2 bootloader, you'll need a board-specific .uf2 file. Click here to find the latest build for your hardware:

Latest CircuitPython Releases

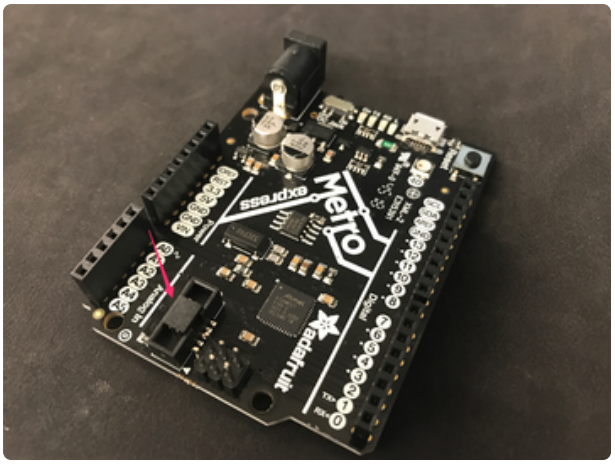
https://adafru.it/Em8

When the UF2 finishes downloading, move the file to your Desktop.

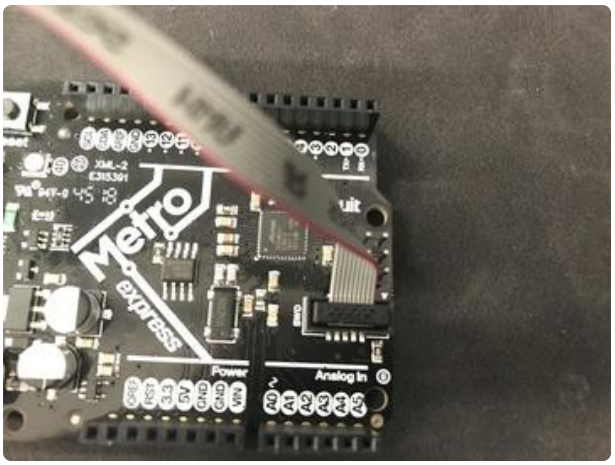
We'll leave this file alone and come back to it when we're ready to load CircuitPython onto our board's boot drive.

Metro M0/M4 Wiring

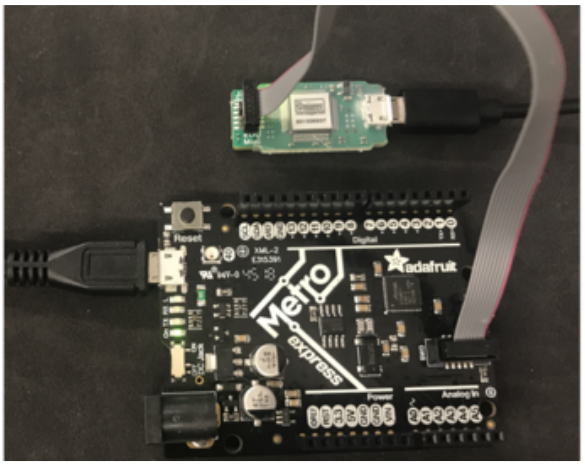
The Adafruit Metro M0 and M4 have a socket for connecting to a SWD cable built-in - you can connect (and disconnect) a SWD cable quickly.



Remove the small plastic cap cap from the SWD socket on the Metro.



Then, connect the SWD cable from the J-Link to the Metro.

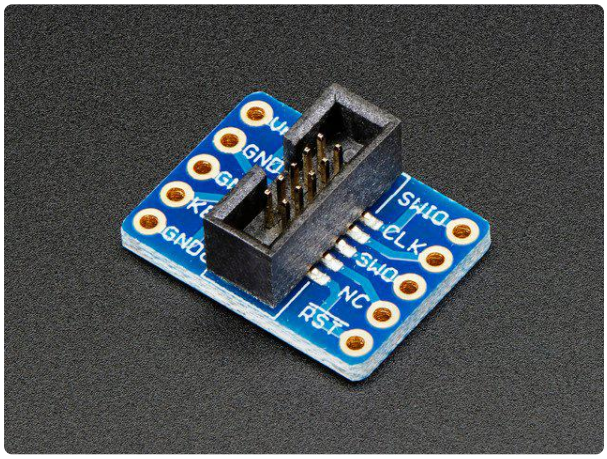


Once the Metro is wired up, plug the J-Link into a USB port on your computer and wait for the status indicator LED to turn green.

Then, plug a USB cable into the Metro, this is required!

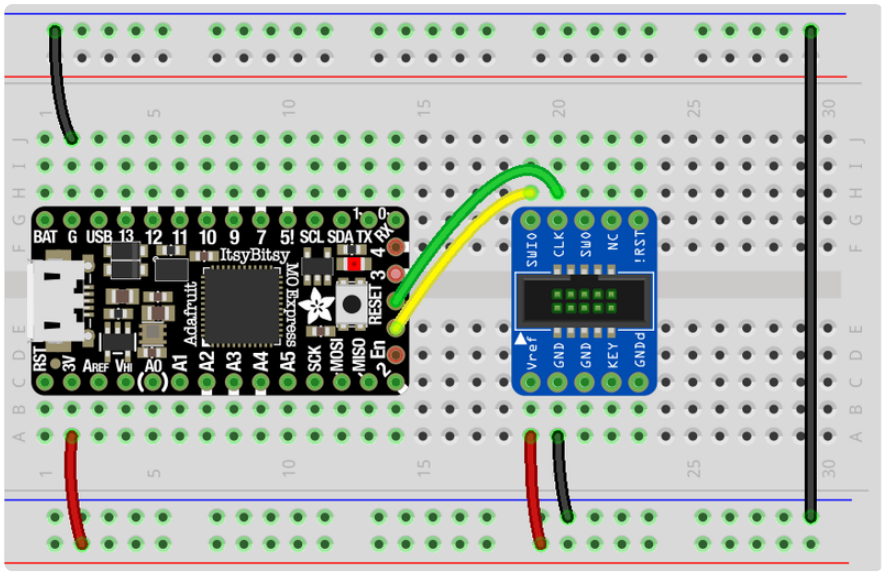
ItsyBitsy M0/M4 Wiring

The ItsyBitsy has pins SWDIO and SWCLK broken out on the back edge of the board. You'll need a SWD cable breakout to connect the J-Link to an ItsyBitsy:



SWD (2x5 1.27mm) Cable Breakout Board
This adapter board is designed to make it easier to use ARM dev boards that use slimmer 2x5 (0.05"/1.27mm pitch) SWD cables for programming. It's helpful for using...
https://www.adafruit.com/product/2743

ItsyBitsy Wiring



Make the following connections between the ItsyBitsy M0/M4 and the SWD Cable Breakout:

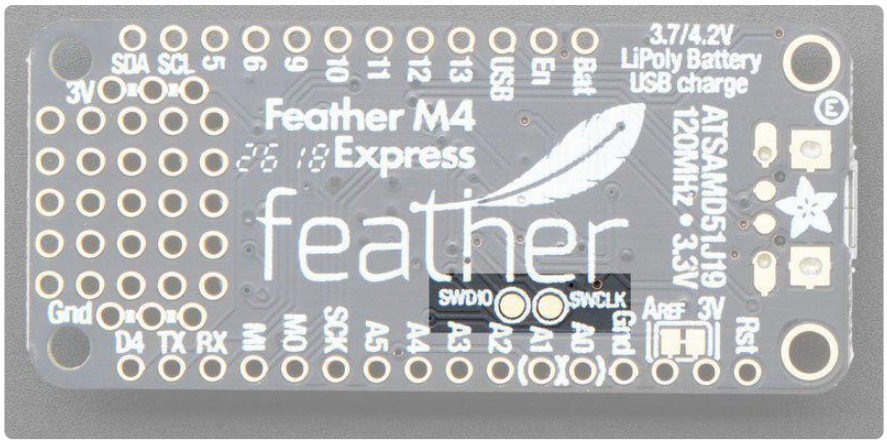
- ItsyBitsy GND to Breakout GND
- ItsyBitsy 3V to Breakout VRef
- ItsyBitsy SWCLK to Breakout CLK
- ItsyBitsy SWDIO to Breakout SWIO

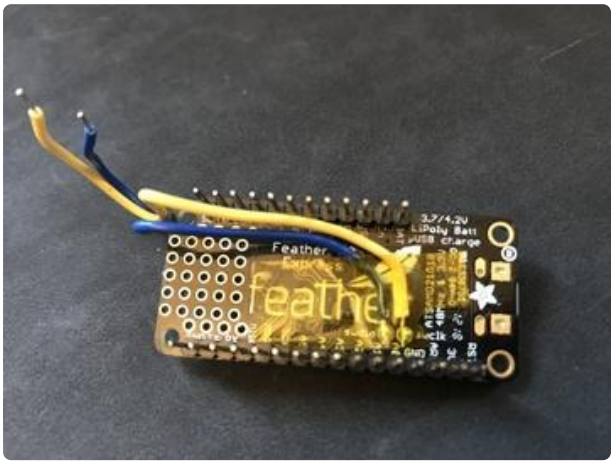
You must also plug in a USB cable to the ItsyBitsy to power it during progamming

---

Feather M0/M4 Wiring

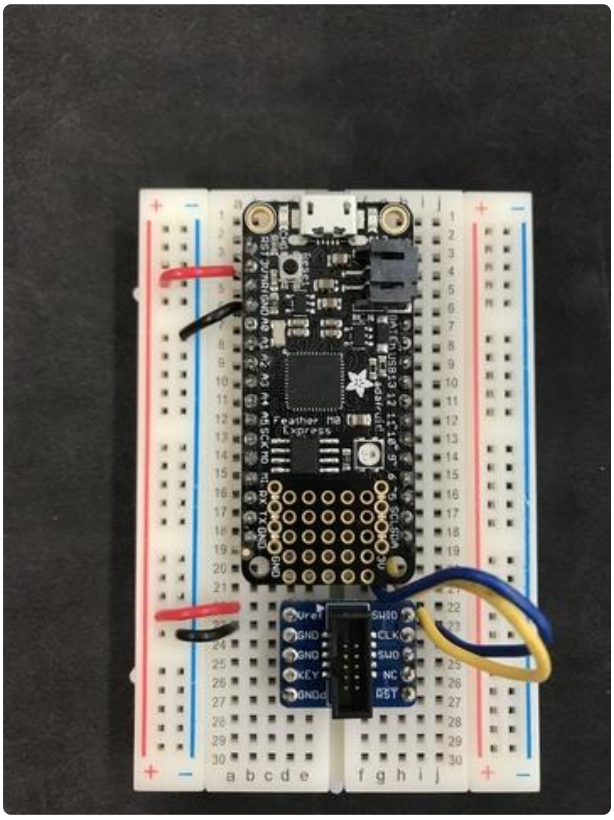
The debugging interface on the Feathers are a little bit harder to get to than the ItsyBitsy and the Metro. They're on the bottom of the board - labeled SWDIO and SW CLK. We'll need to solder a wire to each of them.





Cut and strip two wires. Solder one of them to the SWDIO pad and the other to the SWCLK pad, making sure that the two wires do not touch.

After soldering, add a small piece of tape to secure the connection between the wire and the pad.



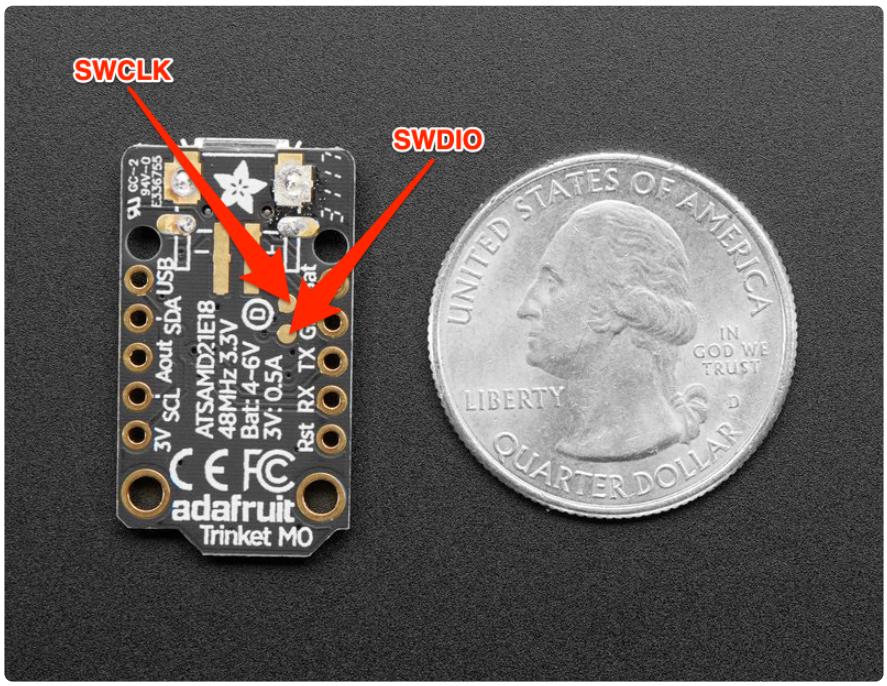
Make the following connections between the Feather and a SWD Breakout:

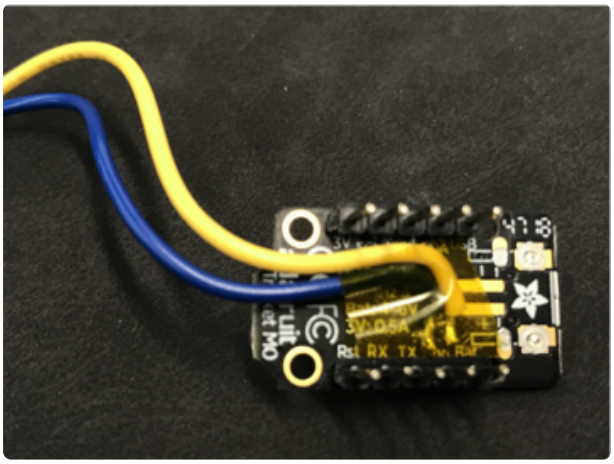
- Feather SWDIO to Breakout SWIO
- Feather SWCLK to Breakout CLK
- Feather 3V to Breakout VRef
- Feather GND to Breakout GND

You must also plug in a USB cable to the Feather to power it during programming

Trinket M0 Wiring

There are two programming pads located at the bottom of the Trinket M0:

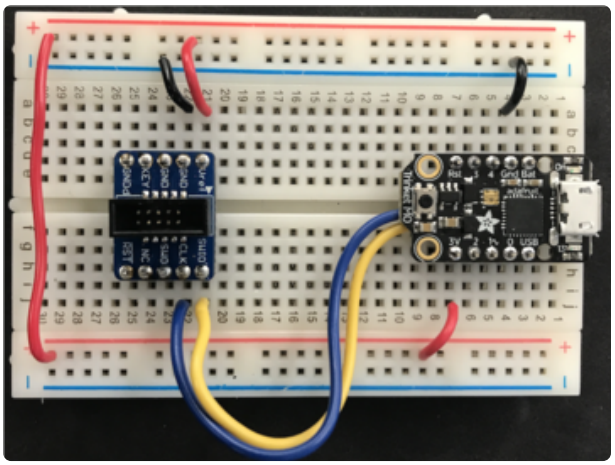




Cut and strip two wires.

Solder one of them to the SWDIO pad and the other to the SWCLK pad, making sure that the two wires do not touch.

After soldering, add a small piece of tape to secure the connection between the wire and the pad.



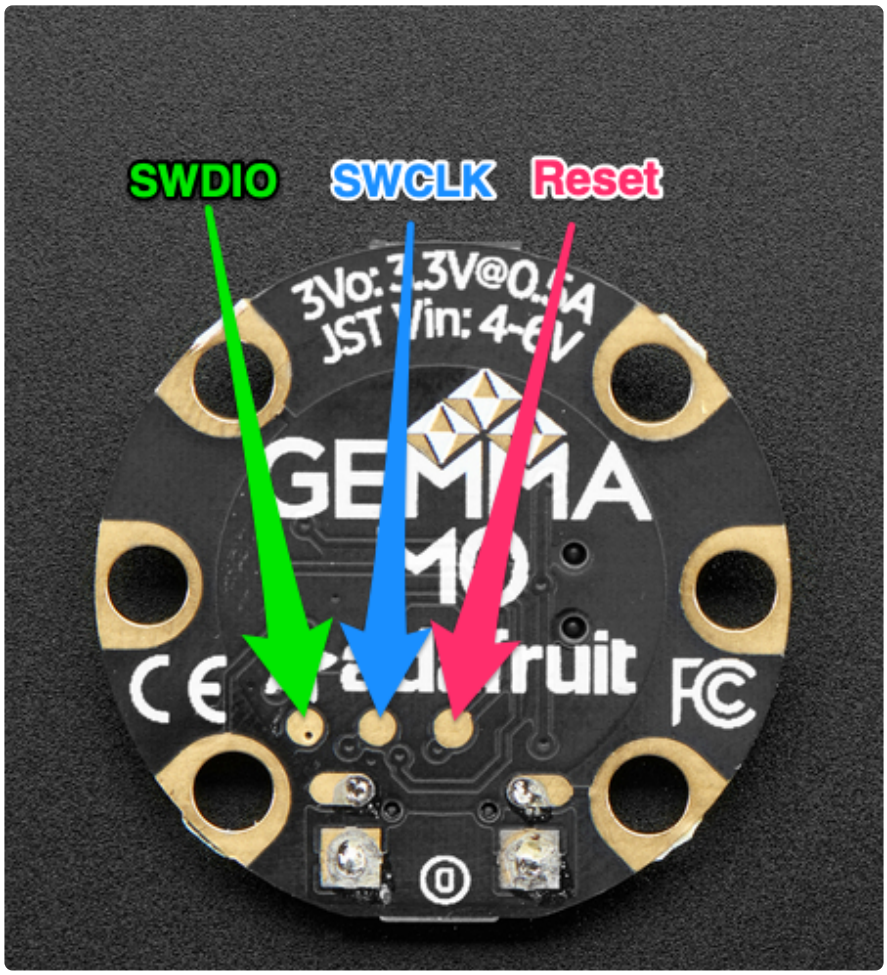
Then, make the following connections between the Trinket and the SWD breakout:

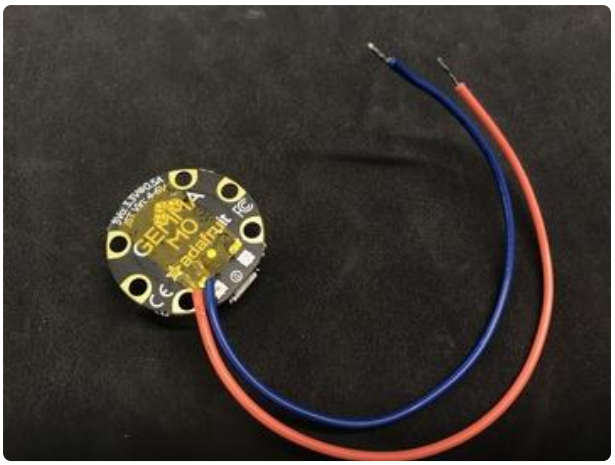
- Trinket SWDIO to Breakout SWIO
- Trinket SWCLK to Breakout CLK
- Trinket 3Vo to Breakout VRef
- Trinket GND to Breakout GND

You must also plug in a USB cable to the Trinket to power it during progamming

Gemma M0 Wiring

On the bottom of the Gemma, there are three small pads used for programming:

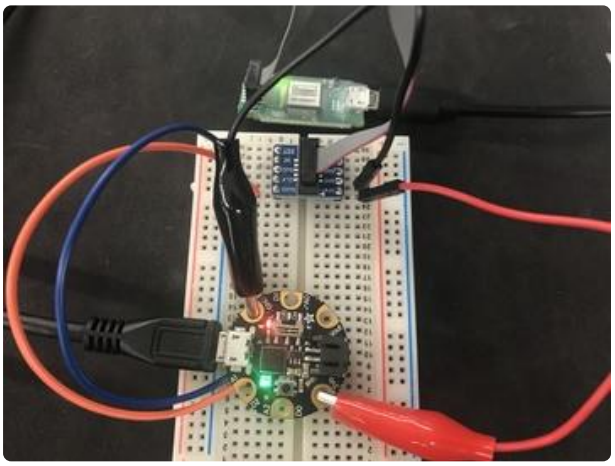




Cut and strip two wires.

Solder one of them to the SWDIO pad and the other to the SWCLK pad, making sure that the two wires do not touch.

After soldering, add a small piece of tape to secure the connection between the wire and the pad.



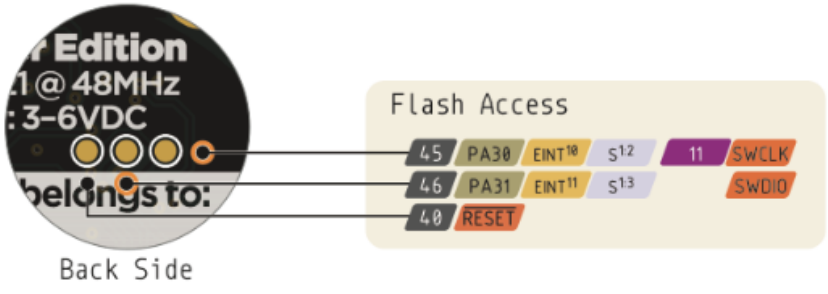
Then, make the following connections between the Gemma and the SWD breakout:

- Gemma SWDIO to Breakout SWIO
- Gemma SWCLK to Breakout CLK
- Gemma 3Vo to Breakout VRef
- Gemma GND to Breakout GND

You can use breadboard-friendly Alligator clips (https://adafru.it/xAV) to attach the Gemma M0's 3Vo and GND pins to the breakout.

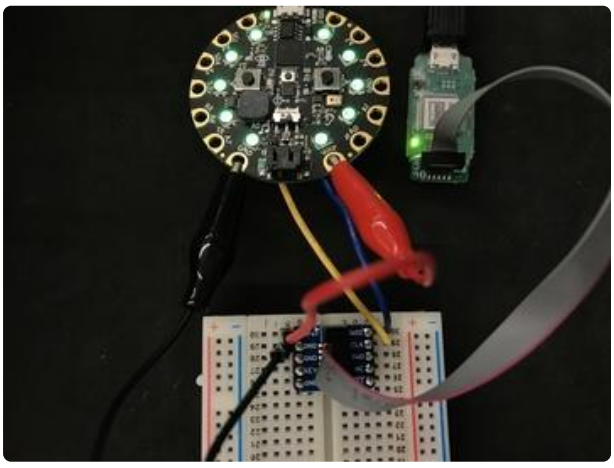
You must also plug in a USB cable to the Gemma to power it during progamming

Circuit Playground Express M0 Wiring



To program the Circuit Playground Express, you'll need to solder directly to the SWCLK and SWDIO pads on the back of the board.

After soldering, use a piece of tape or a dab of hot glue to hold the wires in place so they don't rip off of the pads.



Then, make the following connections between the Circuit Playground Express M0 and the SWD breakout:

- Circuit Playground Express SWDIO to Breakout SWIO
- Circuit Playground Express SWCLK to Breakout CLK
- Circuit Playground Express VOut to Breakout VRef
- Circuit Playground Express GND to Breakout GND

You can use breadboard-friendly Alligator clips (https://adafru.it/xAV) to attach the Circuit Playground Express VOut and GND pins to the breakout.

You must also plug in a USB cable to the Circuit Playground Express to power it during progamming

Programming the Bootloader with Atmel Studio

You'll need to install Atmel Studio 7. This software is free and only works on a Windows host computer.

- If you're running macOS or Linux, you can run Windows in a virtual machine (Parallels, VirtualBox) and install Atmel Studio on the Windows virtual machine.

The download (we recommend the online installer) is available from Microchip's website.

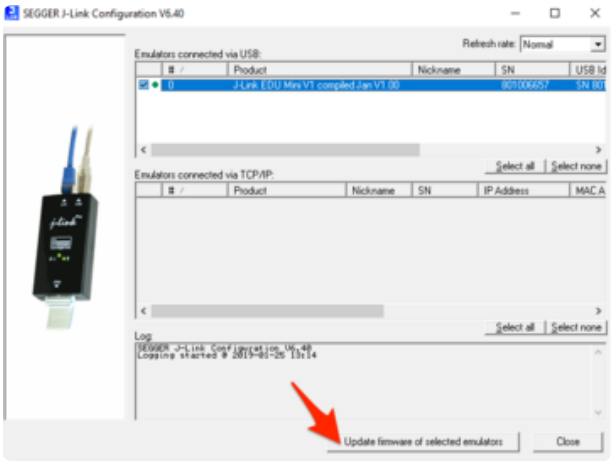
Download Microchip Studio 7

https://adafru.it/TAL

Setting up JLink for Atmel Studio

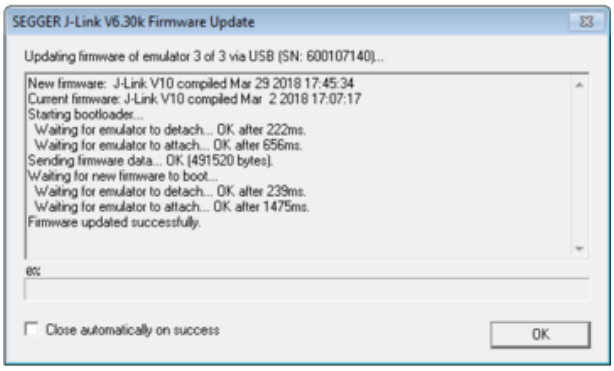
JLinks ship with the firmware they are programmed with from the factory. We'll want to update ours to the latest version.

From the Windows search bar, search for the Jlink Configurator application and launch it.

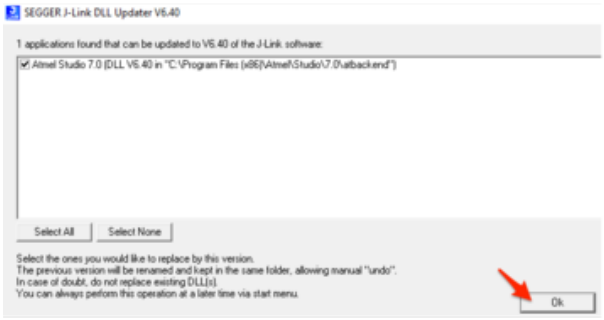


The JLink Configurator tool will show all connected devices. Tick the box next to the JLink you want to update.

Then, click Update Firmware of Selected Emulators.



If a new firmware is available, the JLink will launch a pop-up window, updating the firmware.

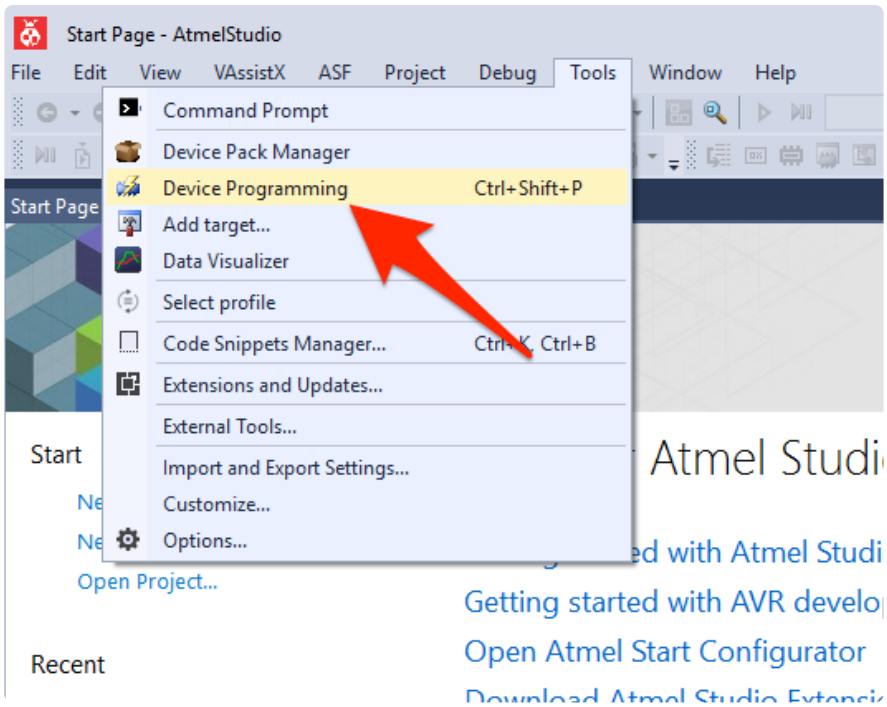


Next, launch the SEGGER JLink DLL Updater from the start menu.

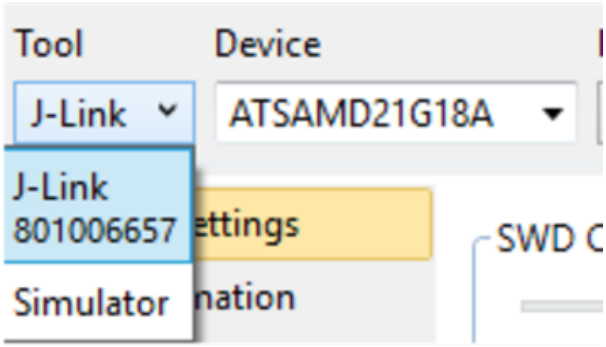
If there is an update available for Atmel Studio 7, tick the checkbox and click OK to update Atmel Studio 7's the latest JLink software.

Verify Connection in Atmel Studio

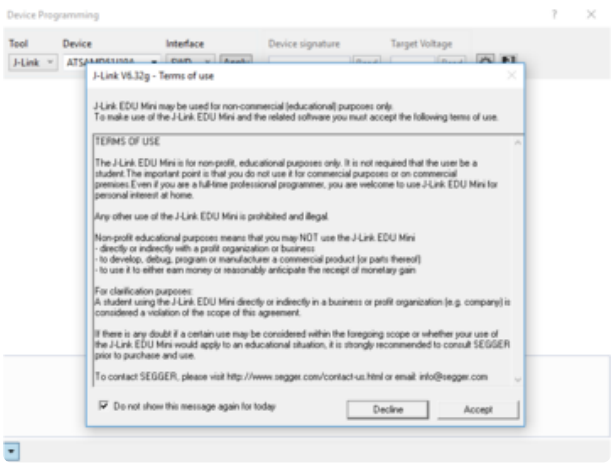
Next, open Atmel Studio. From the toolbar, select Tools -> Device Programming.



The device programming window will open. Before we proceed, make sure your J-Link and board are both plugged into your computer.

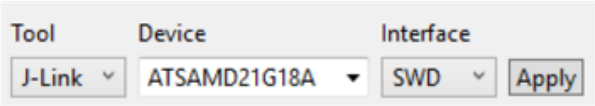


Then, select Tool->J-Link.



- If you're using a J-Link EDU, the terms of use will pop up after selecting the J-Link interface. Click ACCEPT.

Flashing a SAMD21 M0 Board with Atmel Studio



Next, we're going to select the Device (the type of chip you're programming):

- For the Feather M0, Metro M0, Trinket M0, and Circuit Playground Express, select ATSAMD21G18A
- For the Gemma M0 and Trinket M0, select ATSAMD21E18A

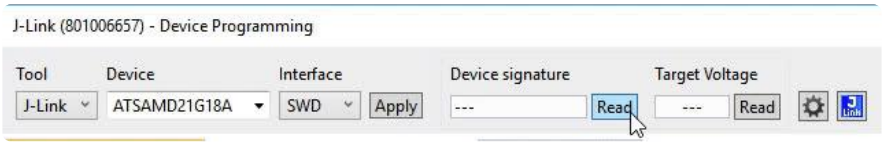
Click Apply.

Make sure your board is plugged into USB, then click Read in the top nav br. The empty fields for Device Signature and Target Voltage will populate.

Make sure these values appear before proceeding!

- If the board is not detected, or your wiring is not detected, Atmel Studio will throw an error that it could not connect to the board. Check your wiring (especially the SWDIO/SWCLK wires), that your USB cables are connected to the computer, and try to connect again.

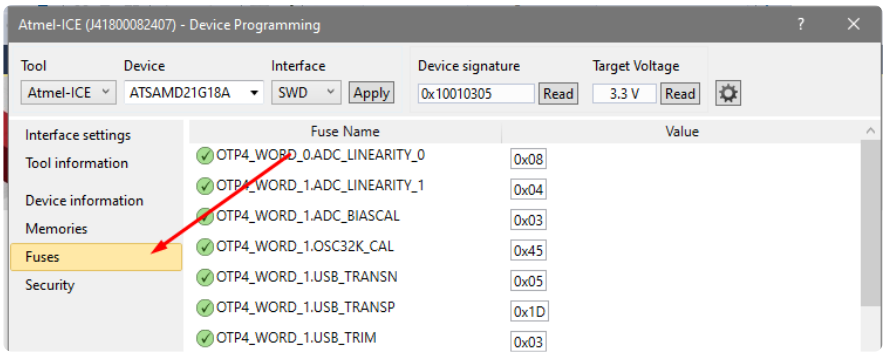
Be sure your USB cable is a USB data cable and not a "cell phone charging" power only cable. The data lines are needed to communicate between your computer and the J-LINK.



Un-set Bootloader Protection Fuse

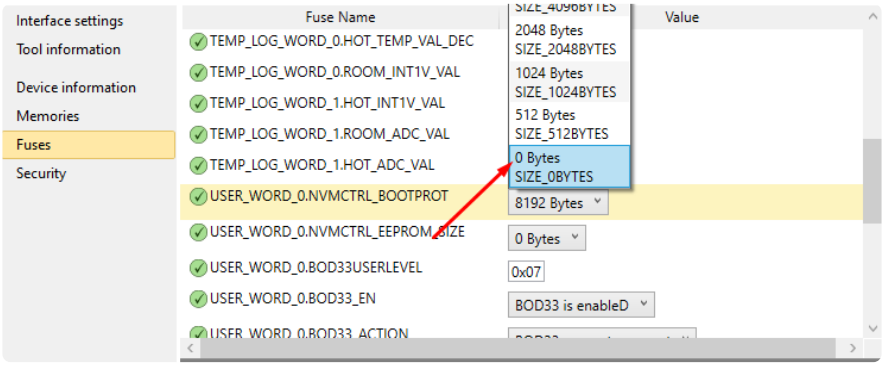
The SAMD21 has a `BOOTPROT` fuse protecting the flash area of the bootloader. You'll want to clear the `BOOTPROT` fuse before flashing the bootloader.

Go to the Fuses page



Click Read in the bottom right section to read the fuses. Look for the BOOTPROT fuse

For SAMD21, you will need to set it to `0x07` or Zero Bytes



Click Program, wait for a confirmation that the fuses have been set. Then, click Verify.

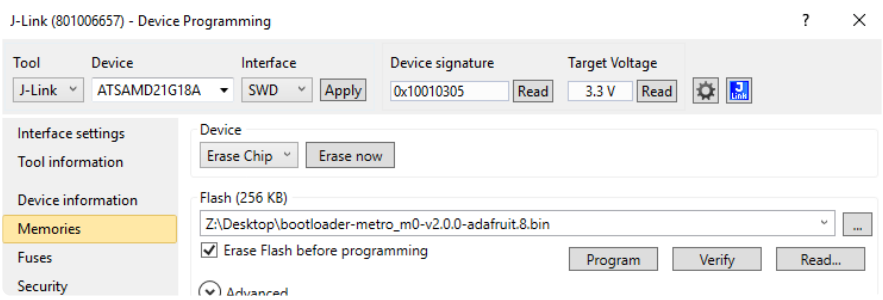
Program Binary File

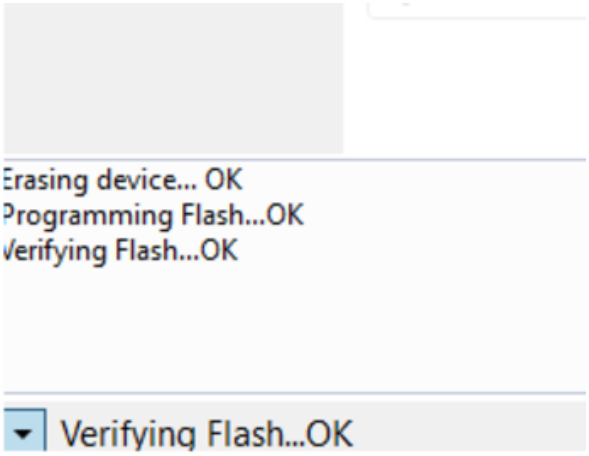
On the sidebar, click Memories.

Select the bootloader for the CircuitPython board you're recovering (the .bin file you downloaded earlier).

Select Erase Flash before programming and Verify flash after programming.

Then, click Program.





After clicking Program, the serial will output:

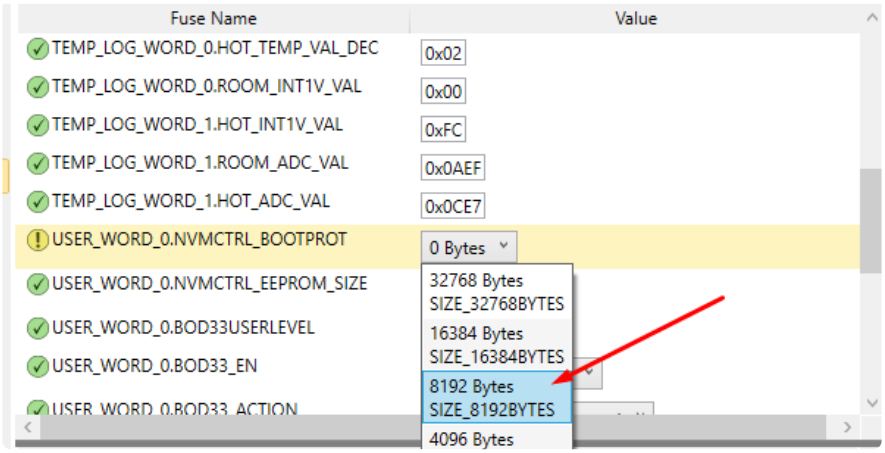
```
Erasing Device...OK
```

```
Programming Flash...OK
```

```
Verifying Flash...OK
```

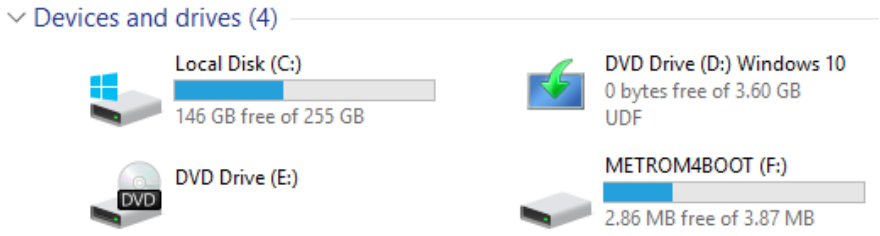
After flashing, you'll need to set the `BOOTPROT` fuse back to a 8kB bootloader size.

From Fuses, set **`BOOTPROT`** to 0x2 or 8KB and click Program



Check Success!

Open Windows Explorer - there should be a new volume mounted on your machine indicating that the board has been programed with the UF2 bootloader:

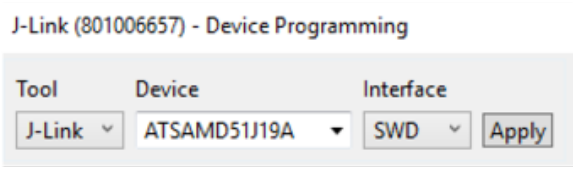


If you want to use the SAMD21 board with Arduino or Microsoft MakeCode, you can stop here.

If you want to use it with CircuitPython, let's continue to installing a CircuitPython build.

Flashing a SAMD51 M4 Board with Atmel Studio

Boards like the Metro M4 and the Grand Central use a different, more powerful chip than the SAMD21, the SAMD51 - the flashing process is a little bit different for these boards.



In the Device Programming window, select the device based off which M4 CircuitPython Board you have.

- Feather M4 and Metro M4, select ATSAMD51J19A
- ItsyBitsy M4, select ATSAMD51G19A
- For the Grand Central M4, select ATSAMD51P20

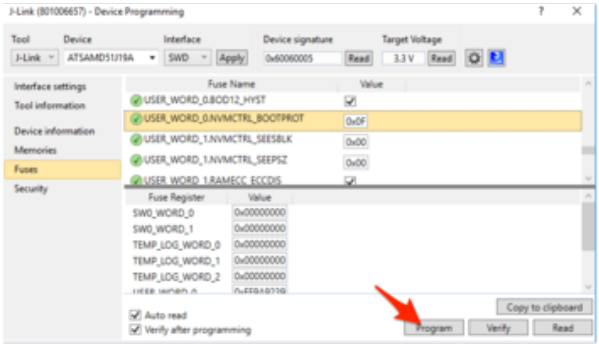
Click Apply.

Make sure your board is plugged into USB, then click Read. The empty fields for Devi ce Signature and Target Voltage will populate.

Make sure these values appear before proceeding!

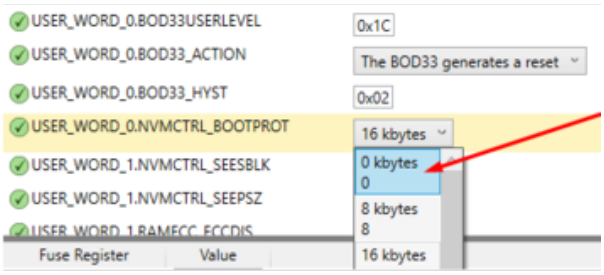
- If the board is not detected, or your wiring is not detected, Atmel Studio will throw an error that it could not connect to the board. Check your wiring (especially the SWDIO/SWCLK wires), that your USB cables are connected to the computer, and try to connect again.

Be sure your USB cable is a USB data cable and not a "cell phone charging" power only cable. The data lines are needed to communicate between your computer and the J-LINK.



The SAMD51 has a `BOOTPROT` fuse protecting the flash area of the bootloader. You'll want to clear the `BOOTPROT` fuse before flashing the bootloader.

- From the Device Programming window, select Fuses.
- Then, change the value of `USER_WORD_0.NVMCTRL_BOOTPROT` to `0x0F` or 0 kbytes
- Click Program
- Then, click VERIFY to verify that the fuse has been set correctly.

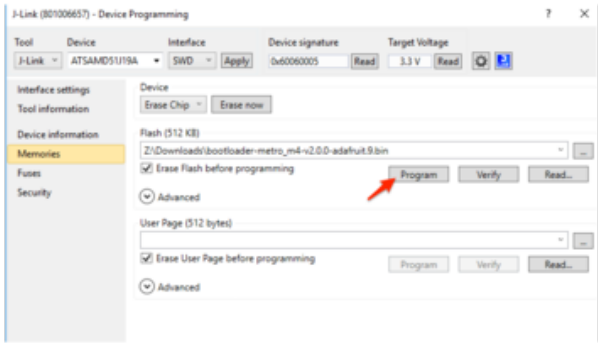


On the sidebar, click Memories.

Select the bootloader for the CircuitPython board you're recovering (the .bin file you downloaded earlier).

Select Erase Flash before programming and Verify flash after programming.

Then, click Program.



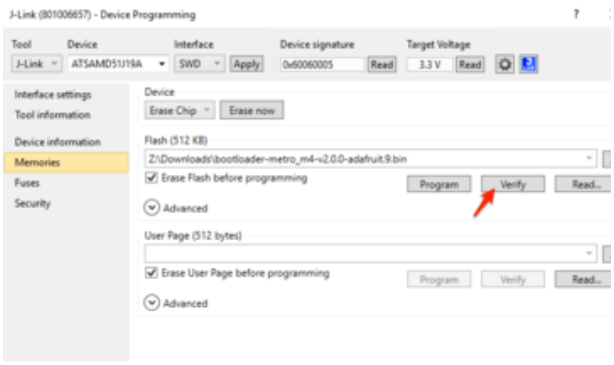
On the sidebar, click Memories.

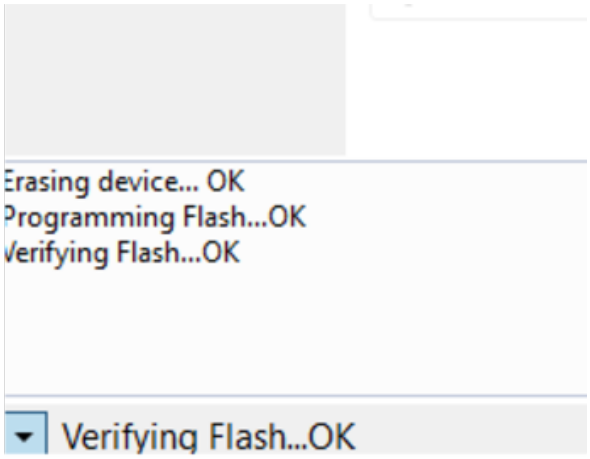
Select the bootloader for the CircuitPython board you're recovering (the .bin file you downloaded earlier).

Select Erase Flash before programming

Click Program.

Then, click Verify





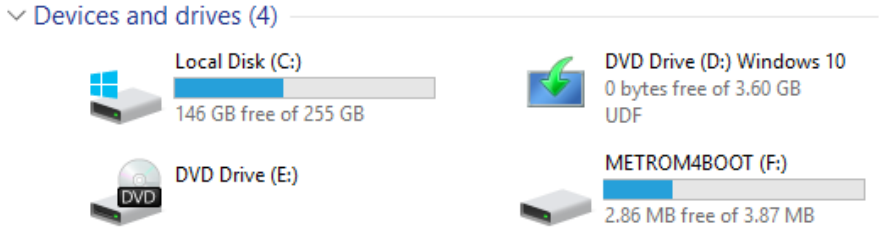
After clicking Program, the serial will output:

```
Erasing Device...OK
```

```
Programming Flash...OK
```

```
Verifying Flash...OK
```

Open Windows Explorer - there should be a new volume mounted on your machine indicating that the board has been programed with the UF2 bootloader:



If you want to use the SAMD51 board with Arduino or Microsoft MakeCode, you can stop here.

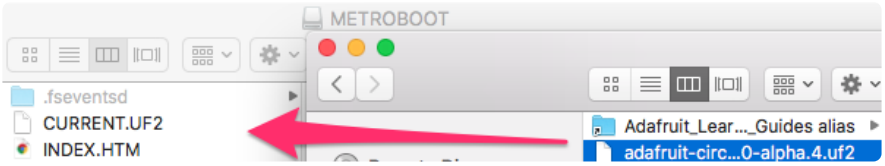
If you want to use it with CircuitPython, let's continue to installing a CircuitPython build.

Installing CircuitPython

A perk of the SAMD UF2 bootloader is the ability to load other firmware (like CircuitPython) onto the board without re-programming the bootloader.

Double tap the Reset button, a new volume should appear on your computer - board NameBOOT (the name of this volume differs between boards).

Drag and drop the CircuitPython board and language-specific UF2 file we downloaded earlier from the desktop to the boardNameBOOT volume.



The volume should briefly disappear, and reappear as CIRCUITPY.

Congrats, your board is un-bricked and running CircuitPython again!