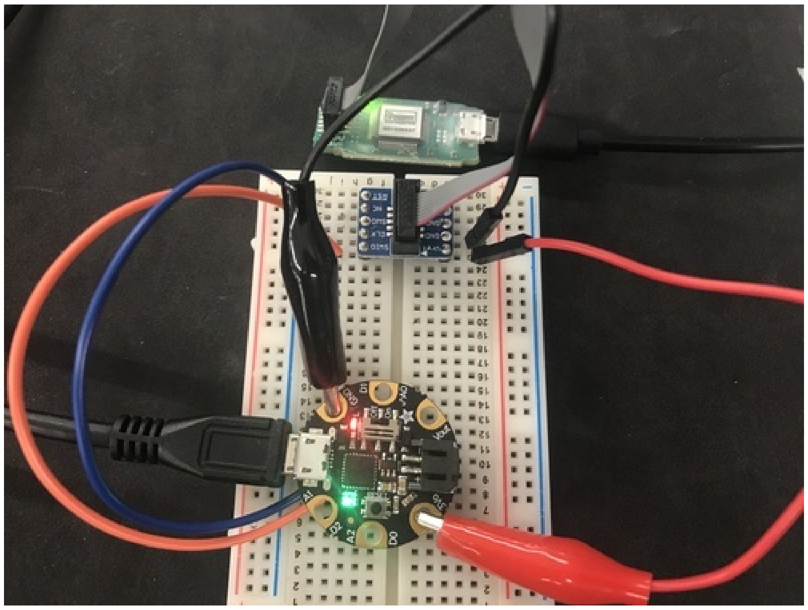
adafruit learning system

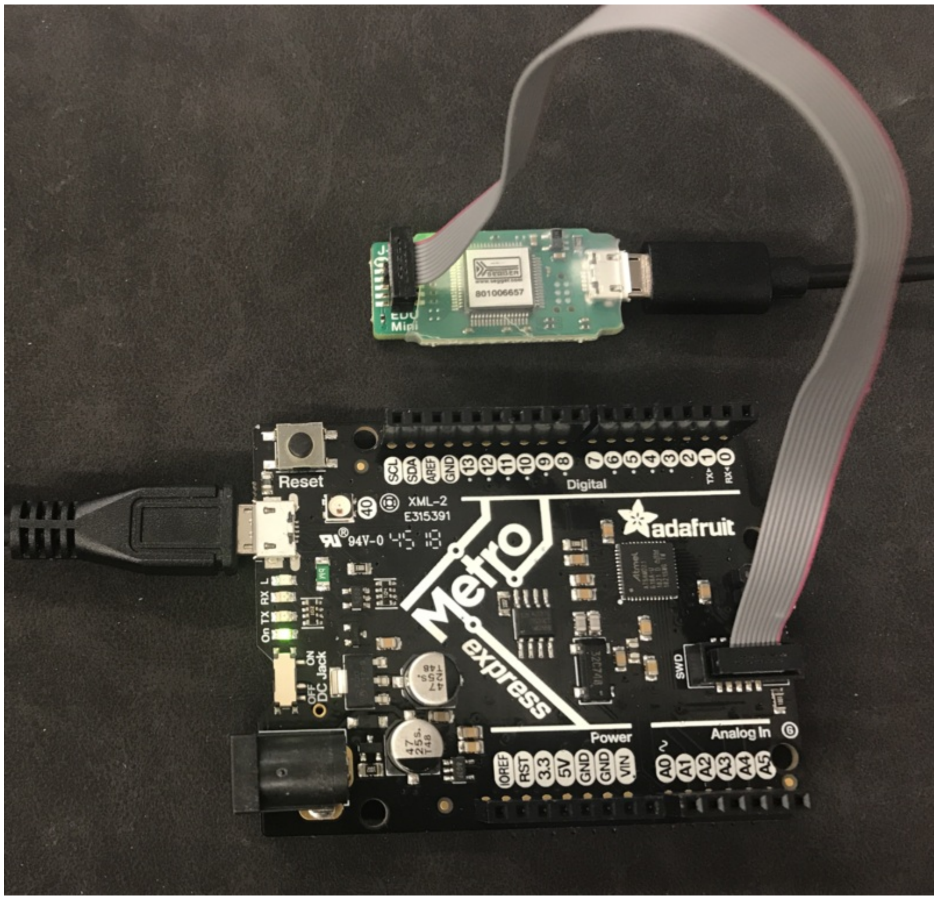
How to Program SAMD Bootloaders

Created by Brent Rubell



Last updated on 2019-08-08 07:24:37 PM UTC

Overview



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.

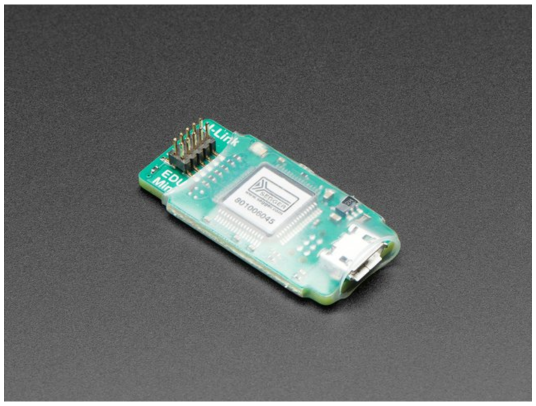
Instead of needing drivers or a separate program for flashing (say, `bossac`, `jlink` or `avrdude`), one can simply ***drag a UF2 file onto a removable drive***.

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

$19.95
IN STOCK

ADD TO CART

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

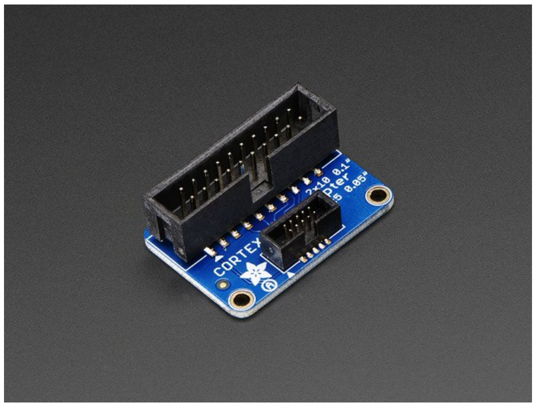
SEGGER J-Link BASE - JTAG/SWD Debugger

OUT OF STOCK

OUT OF STOCK

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

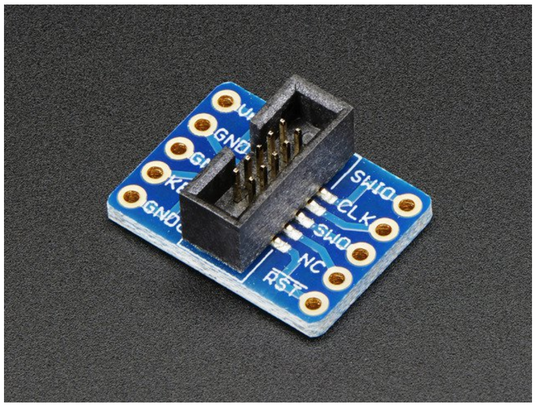
$4.95
IN STOCK

ADD TO CART

10-pin 2x5 Socket-Socket 1.27mm IDC (SWD) Cable - 150mm long

$2.95
IN STOCK

ADD TO CART



SWD (2x5 1.27mm) Cable Breakout Board

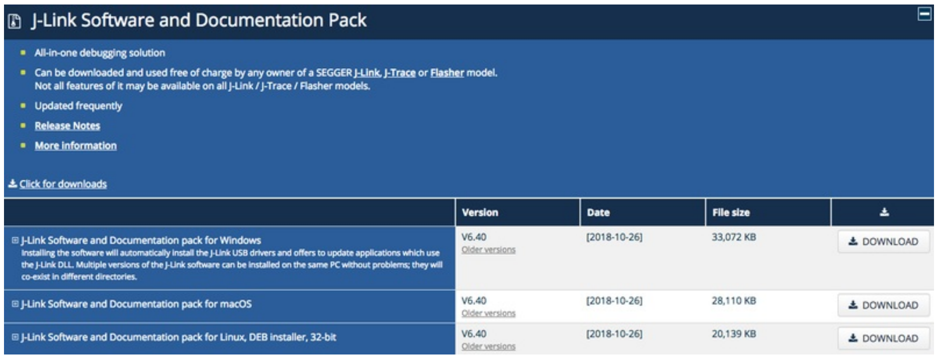
$1.95
IN STOCK

ADD TO CART

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:



J-Link Software and Documentation Pack

- All-in-one debugging solution
- Can be downloaded and used free of charge by any owner of a SEGGER J-Link, J-Trace or Flasher model.
  Not all features of it may be available on all J-Link / J-Trace / Flasher models.
- Updated frequently
- Release Notes
- More information

Click for downloads

	Version	Date	File size	
J-Link Software and Documentation pack for Windows Installing the software will automatically install the J-Link USB drivers and offers to update applications which use the J-Link DLL. Multiple versions of the J-Link software can be installed on the same PC without problems; they will co-exist in different directories.	V6.40 Older versions	[2018-10-26]	33,072 KB	DOWNLOAD
J-Link Software and Documentation pack for macOS	V6.40 Older versions	[2018-10-26]	28,110 KB	DOWNLOAD
J-Link Software and Documentation pack for Linux, DEB installer, 32-bit	V6.40 Older versions	[2018-10-26]	20,139 KB	DOWNLOAD

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):

https://adafru.it/D3C

https://adafru.it/D3C

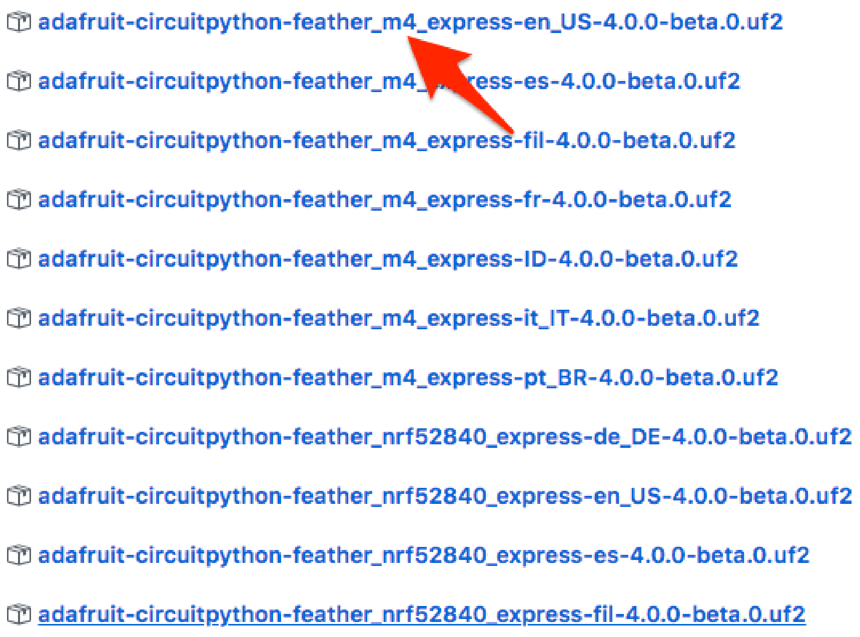
(Optional) Grab the Latest CircuitPython UF2

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:

https://adafru.it/tBa

https://adafru.it/tBa

Click the file corresponding to the CircuitPython board you're recovering. Check the two letter code for your language: **en**glish, **es**pagnol, **fr**ench, **fil**ipino (tagalog) - we'll be using the **en**glish build but instructions are the same for other languages.



adafruit-circuitpython-feather_m4_express-en_US-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-es-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-fil-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-fr-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-ID-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-it_IT-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_m4_express-pt_BR-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_nrf52840_express-de_DE-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_nrf52840_express-en_US-4.0.0-beta.0.uf2

adafruit-circuitpython-feather_nrf52840_express-es-4.0.0-beta.0.uf2

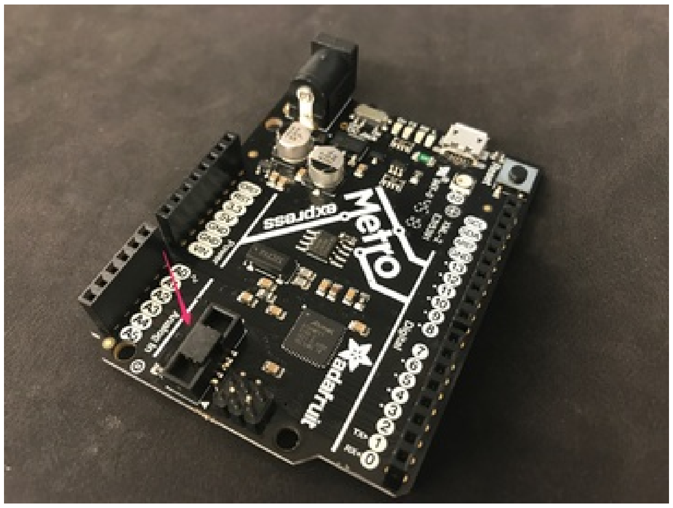
adafruit-circuitpython-feather_nrf52840_express-fil-4.0.0-beta.0.uf2

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

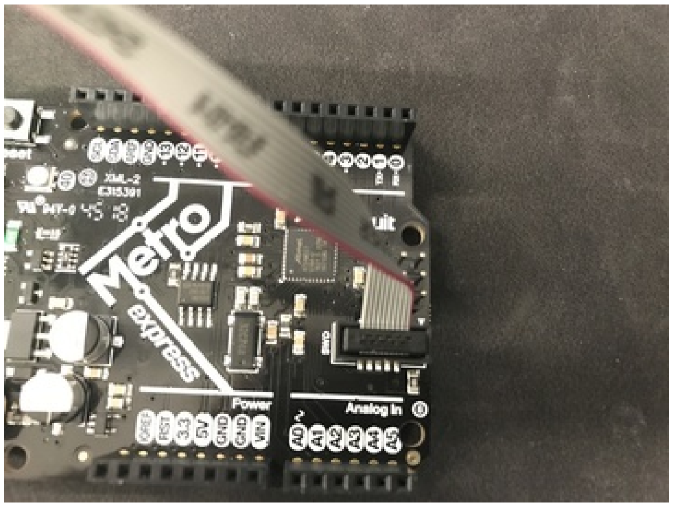
We'll leave this file alone and come back to it when we're ready to load CircuitPython onto our board's boot drive.

Metro M0/M4 Wiring

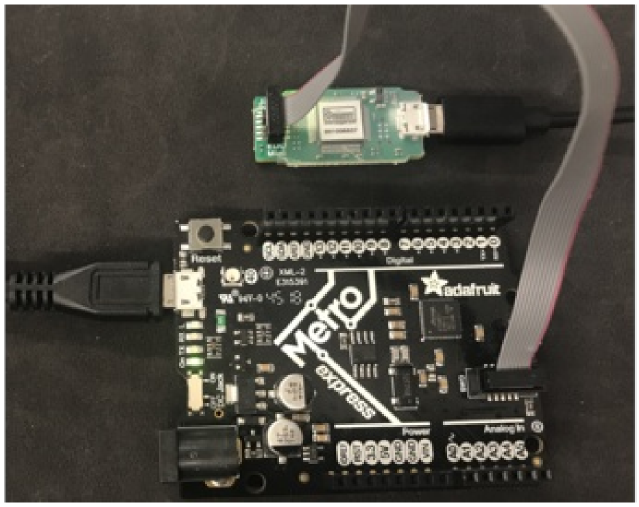
The Adafruit Metro M0 and M4 have a socket for connecting to a SWD cable built-in - you can connect (and disconnect) a SWD cable quickly.



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.

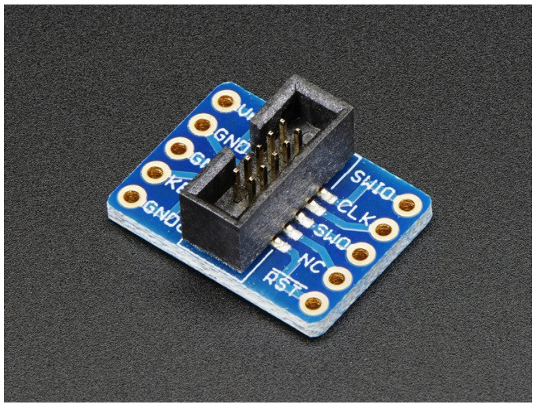


Once the Metro is wired up, **plug the J-Link into a USB port** on your computer and wait for the status indicator LED to turn green.

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:

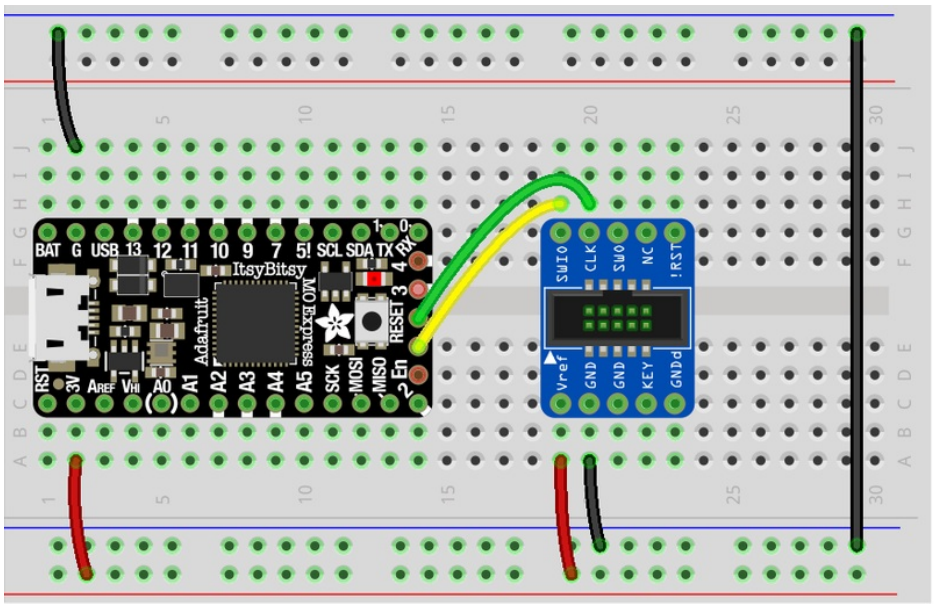


SWD (2x5 1.27mm) Cable Breakout Board

$1.95
IN STOCK

ADD TO CART

ItsyBitsy Wiring



BAT G USB 13 12 11 10 9 7 5! SCL SDA TX RX
ItsyBitsy
M0 Express
Adafruit
RESET
RST 3V AREF VHI A0 A1 A2 A3 A4 A5 SCK MOSI MISO En
SWIO CLK SWO NC !RST
Vref GND GND KEY GNDd

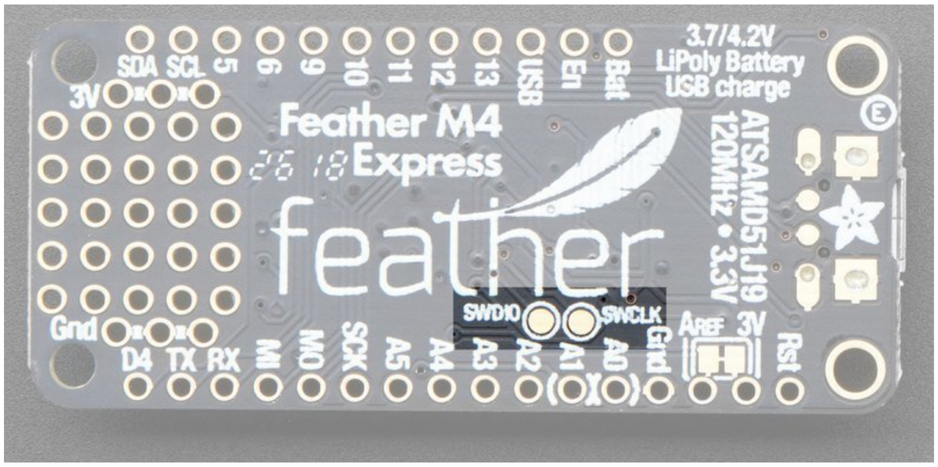
Make the following connections between the ItsyBitsy M0/M4 and the SWD Cable Breakout:

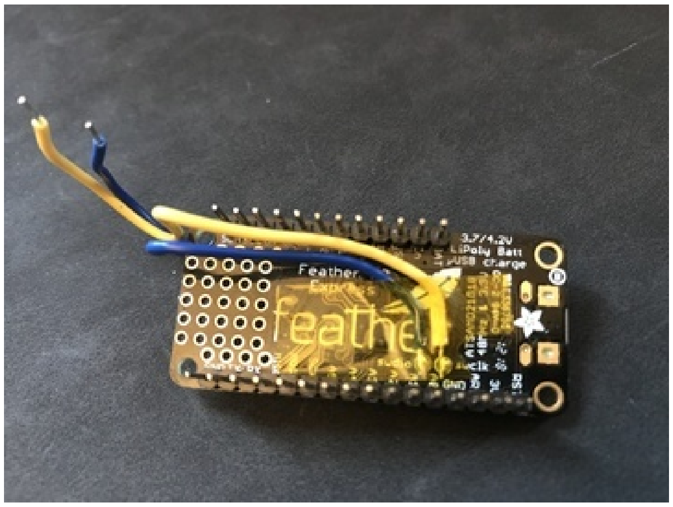
- **ItsyBitsy GND** to **Breakout GND**
- **ItsyBitsy 3V** to **Breakout VRef**
- **ItsyBitsy SWCLK** to **Breakout CLK**
- **ItsyBitsy SWDIO** to **Breakout SWIO**

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

Feather M0/M4 Wiring

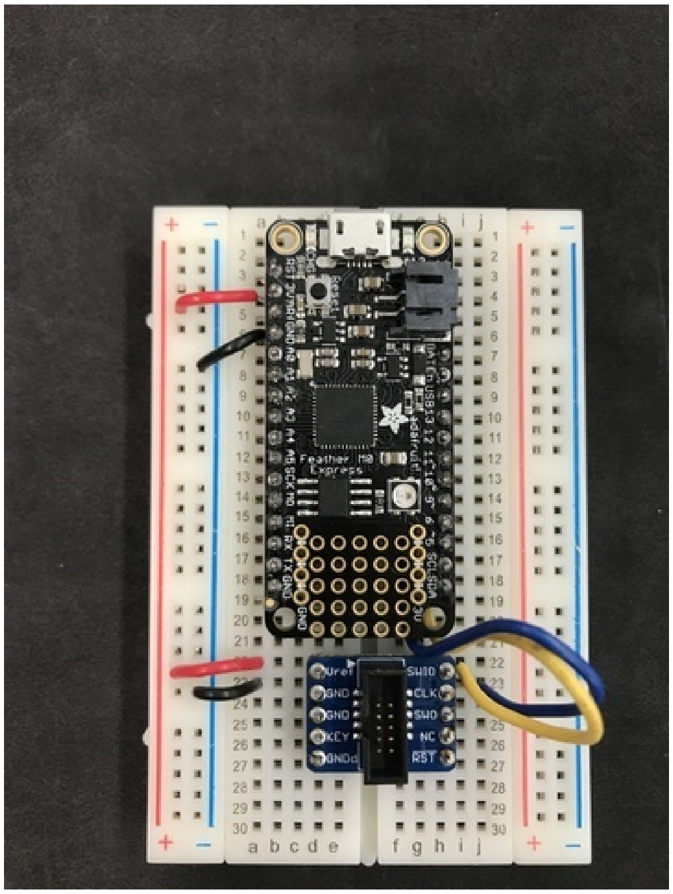
The debugging interface on the Feathers are a little bit harder to get to than the ItsyBitsy and the Metro. They're on the bottom of the board - labeled **SWDIO** and **SWCLK**. We'll need to solder a wire to each of them.





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.



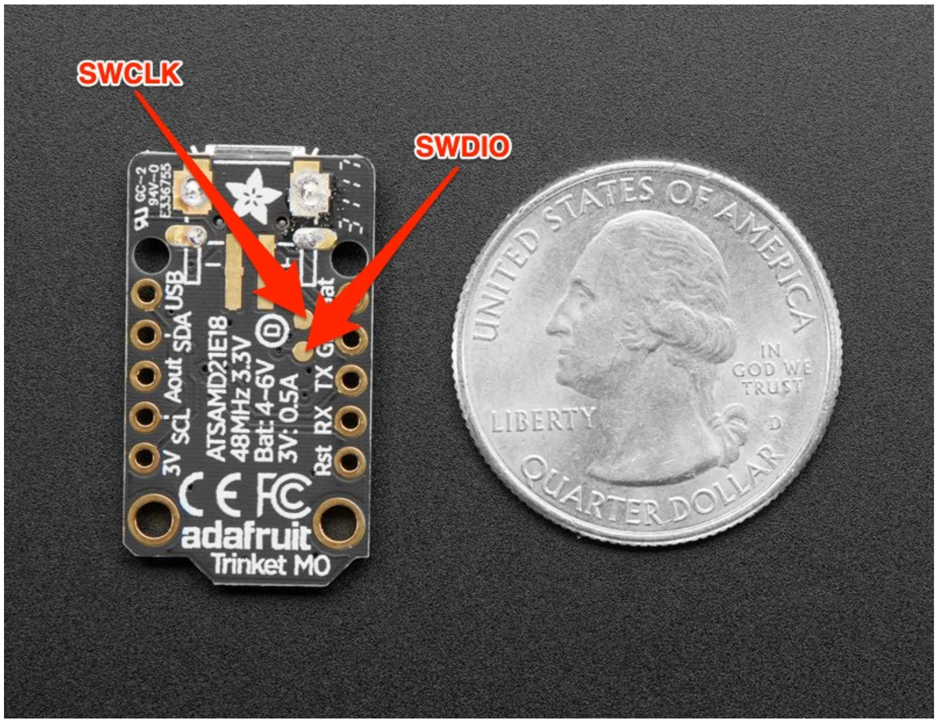
Make the following connections between the Feather and a SWD Breakout:

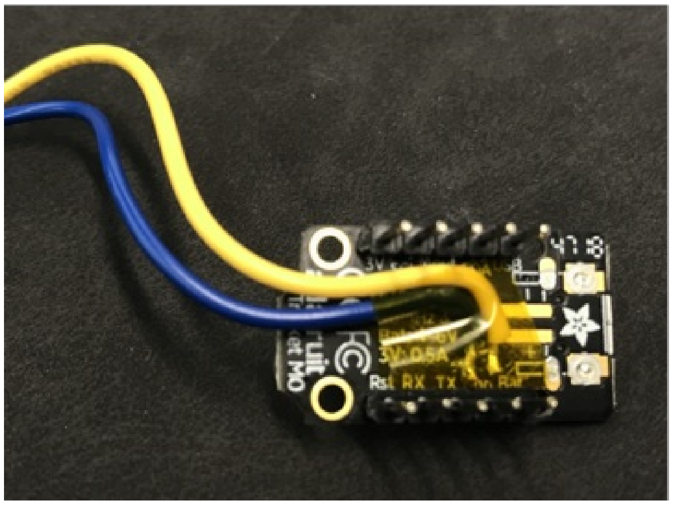
- **Feather SWDIO** to **Breakout SWIO**
- **Feather SWCLK** to **Breakout CLK**
- **Feather 3V** to **Breakout VRef**
- **Feather GND** to **Breakout GND**

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:

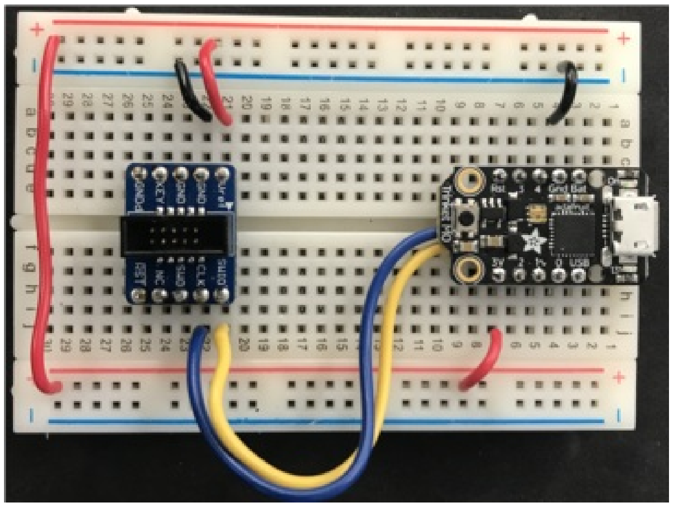




Cut and strip two wires.

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.



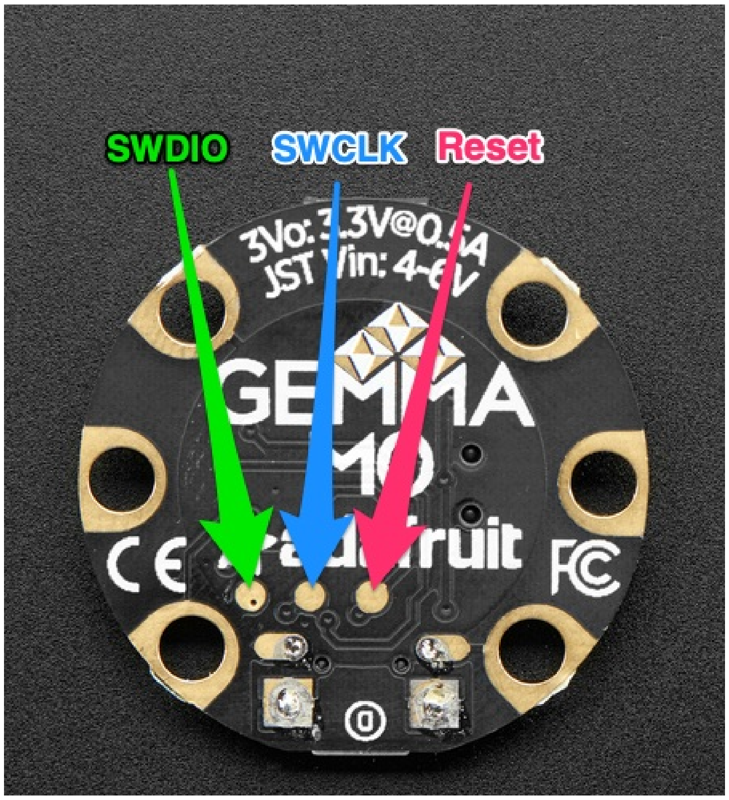
Then, make the following connections between the Trinket and the SWD breakout:

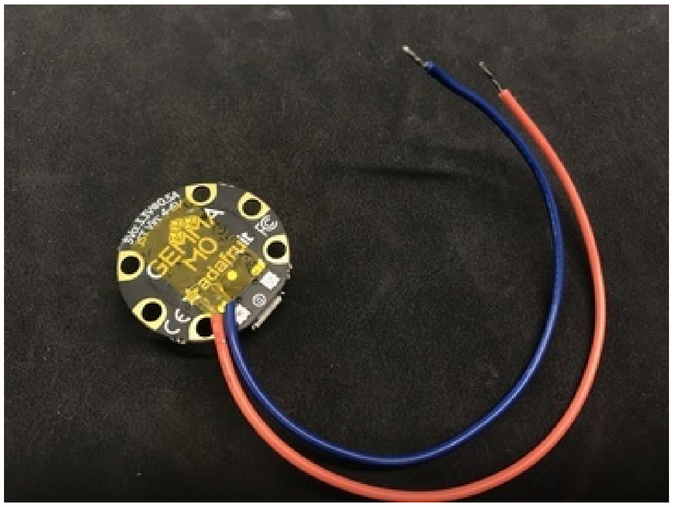
- **Trinket SWDIO** to **Breakout SWIO**
- **Trinket SWCLK** to **Breakout CLK**
- **Trinket 3Vo** to **Breakout VRef**
- **Trinket GND** to **Breakout GND**

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:

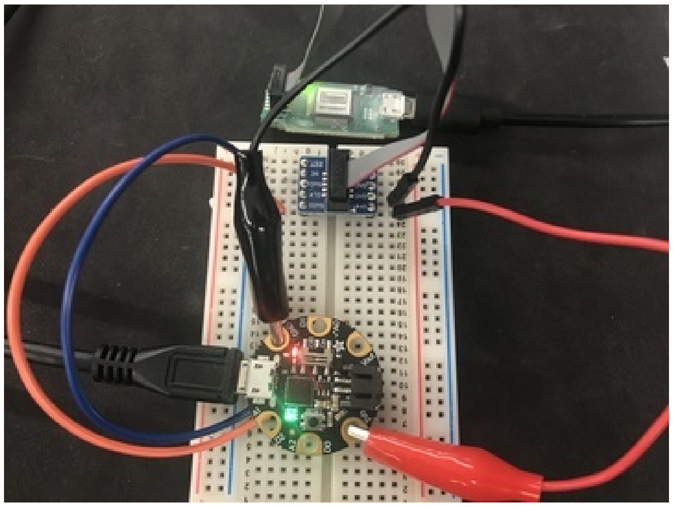




Cut and strip two wires.

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.



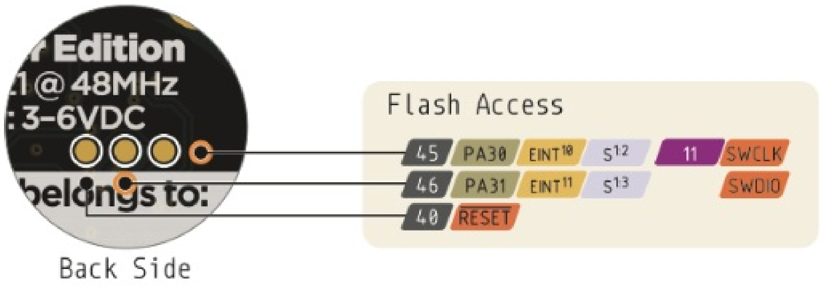
Then, make the following connections between the Gemma and the SWD breakout:

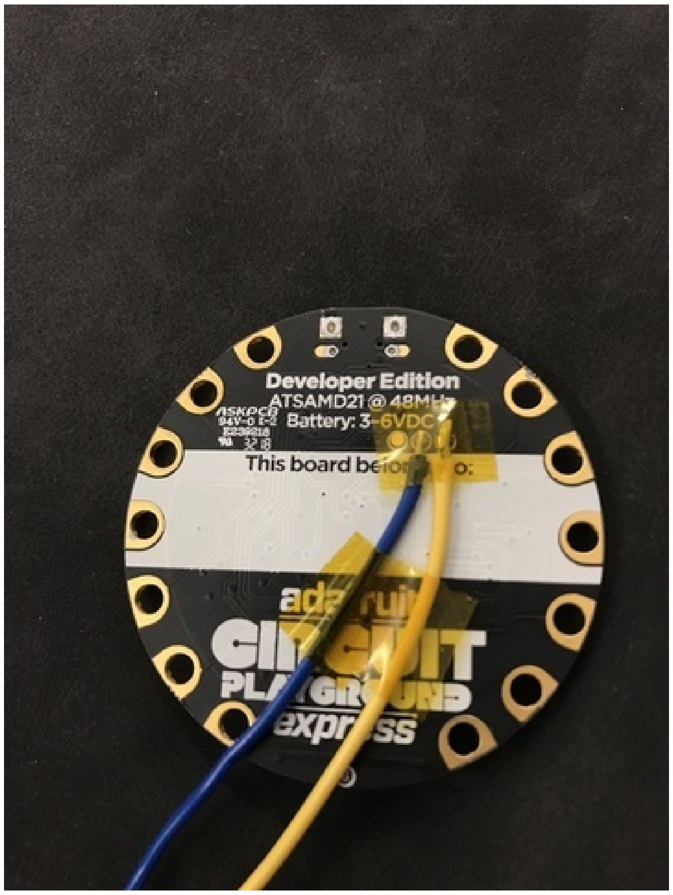
- **Gemma SWDIO** to **Breakout SWIO**
- **Gemma SWCLK** to **Breakout CLK**
- **Gemma 3Vo** to **Breakout VRef**
- **Gemma GND** to **Breakout GND**

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

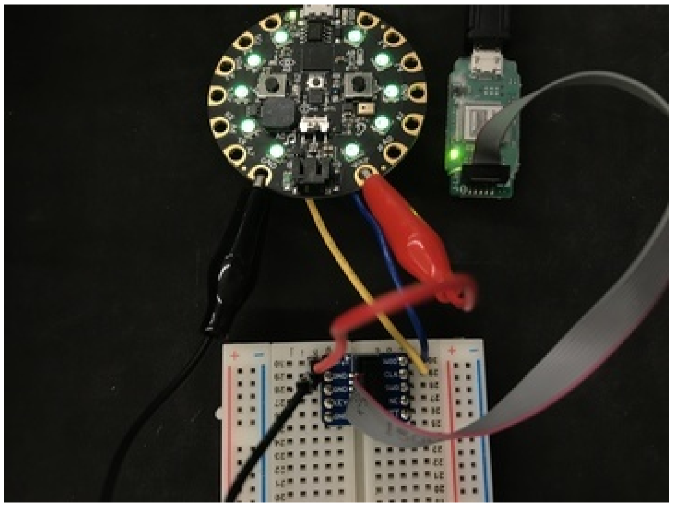
Circuit Playground Express M0 Wiring





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.

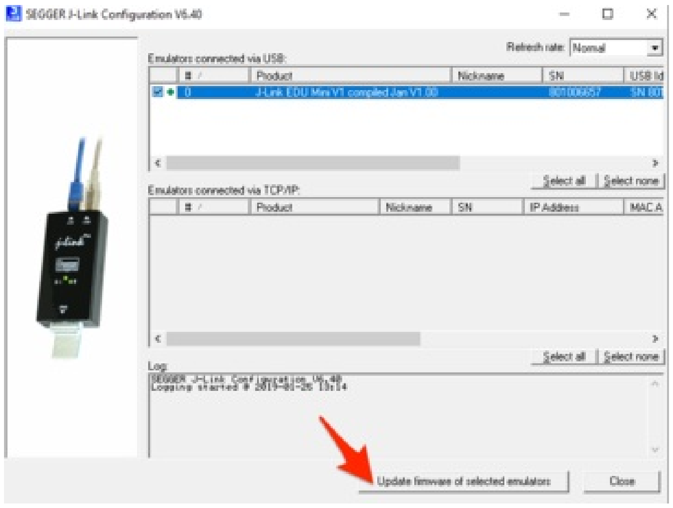
https://adafru.it/DJ0

https://adafru.it/DJ0

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.


SEGGER J-Link Configuration V6.40
Refresh rate: Normal
Emulators connected via USB:
Select all
Select none
Emulators connected via TCP/IP:
Update firmware of selected emulators
Close

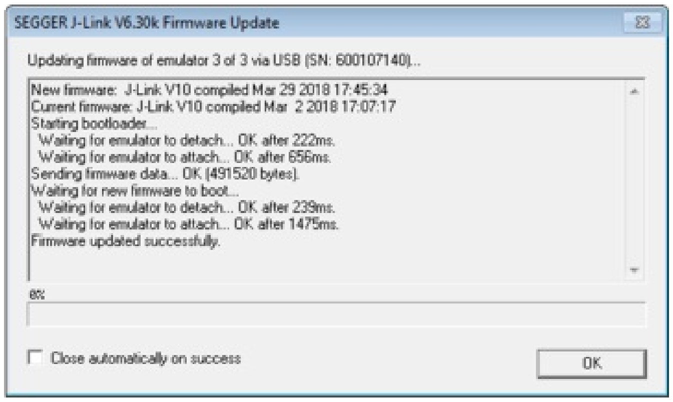
	#	Product	Nickname	SN	USB Id
✓	0	J-Link EDU Mini V1 compiled Jan V1.00		801006657	SN 801

```
SEGGER J-Link Configuration V6.40
Logging started @ 2019-01-26 13:14
```

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.

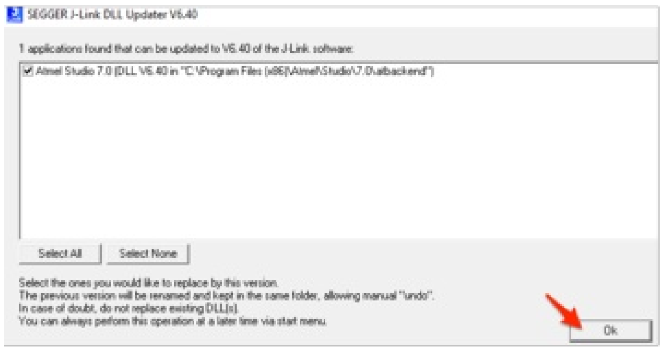

SEGGER J-Link V6.30k Firmware Update
0%
Close automatically on success
OK

```
Updating firmware of emulator 3 of 3 via USB (SN: 600107140)...

New firmware:  J-Link V10 compiled Mar 29 2018 17:45:34
Current firmware: J-Link V10 compiled Mar  2 2018 17:07:17
Starting bootloader...
 Waiting for emulator to detach... OK after 222ms.
 Waiting for emulator to attach... OK after 656ms.
Sending firmware data... OK (491520 bytes).
Waiting for new firmware to boot...
 Waiting for emulator to detach... OK after 239ms.
 Waiting for emulator to attach... OK after 1475ms.
Firmware updated successfully.
```

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

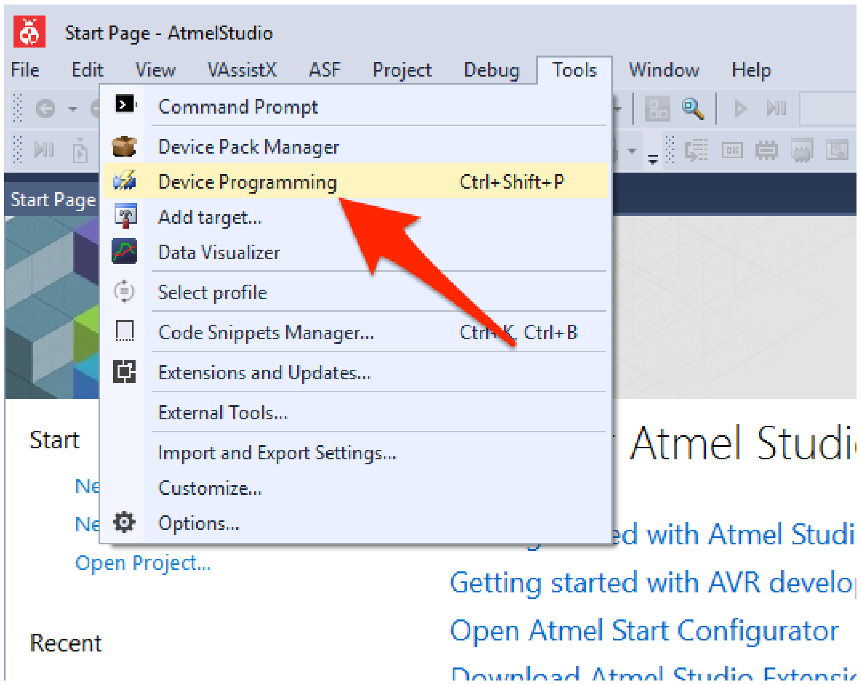
If there is an update available for Atmel Studio 7, **tick the checkbox** and click **OK** to update Atmel Studio 7's the latest JLink software.

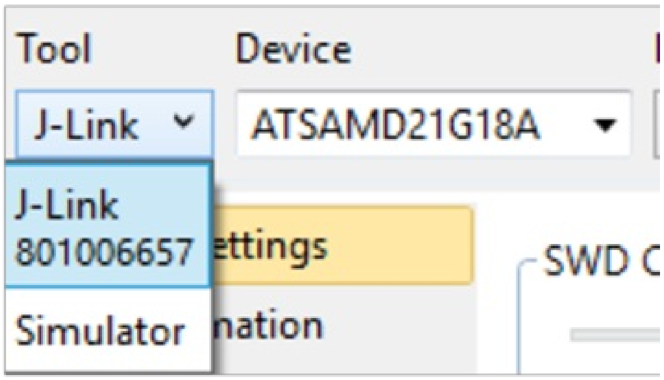


```
1 applications found that can be updated to V6.40 of the J-Link software:
☑ Atmel Studio 7.0 (DLL V6.40 in "C:\Program Files (x86)\Atmel\Studio\7.0\atbackend")

Select the ones you would like to replace by this version.
The previous version will be renamed and kept in the same folder, allowing manual "undo".
In case of doubt, do not replace existing DLL(s).
You can always perform this operation at a later time via start menu.
```

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

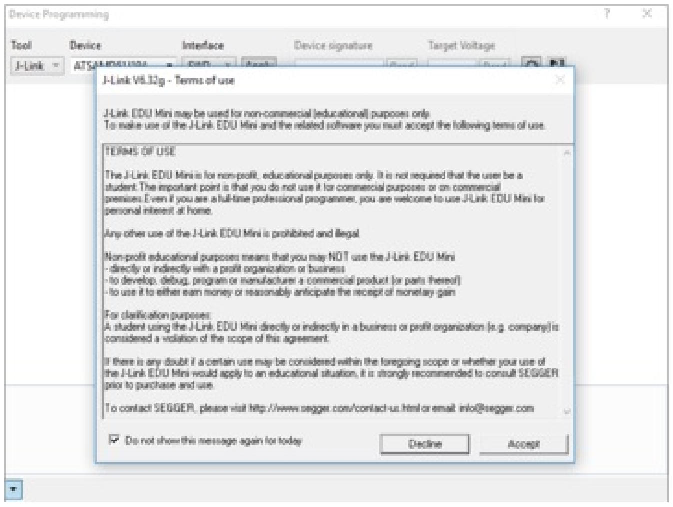




The device programming window will open. Before we proceed, make sure your J-Link and board are both plugged into your computer.

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**.



J-Link V6.32g - Terms of use

J-Link EDU Mini may be used for non-commercial (educational) purposes only.
To make use of the J-Link EDU Mini and the related software you must accept the following terms of use.

TERMS OF USE

The J-Link EDU Mini is for non-profit, educational purposes only. It is not required that the user be a student. The important point is that you do not use it for commercial purposes or on commercial premises. Even if you are a full-time professional programmer, you are welcome to use J-Link EDU Mini for personal interest at home.

Any other use of the J-Link EDU Mini is prohibited and illegal.

Non-profit educational purposes means that you may NOT use the J-Link EDU Mini
- directly or indirectly with a profit organization or business
- to develop, debug, program or manufacturer a commercial product (or parts thereof)
- to use it to either earn money or reasonably anticipate the receipt of monetary gain

For clarification purposes:
A student using the J-Link EDU Mini directly or indirectly in a business or profit organization (e.g. company) is considered a violation of the scope of this agreement.

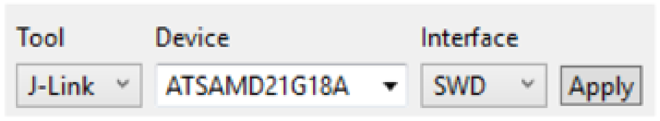
If there is any doubt if a certain use may be considered within the foregoing scope or whether your use of the J-Link EDU Mini would apply to an educational situation, it is strongly recommended to consult SEGGER prior to purchase and use.

To contact SEGGER, please visit http://www.segger.com/contact-us.html or email info@segger.com

[✓] Do not show this message again for today

Decline   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**

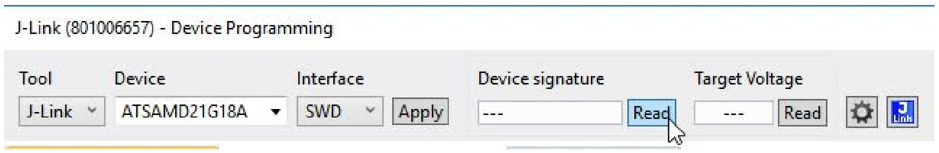
Click Apply.

Make sure your board is plugged into USB, then click **Read**. 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.



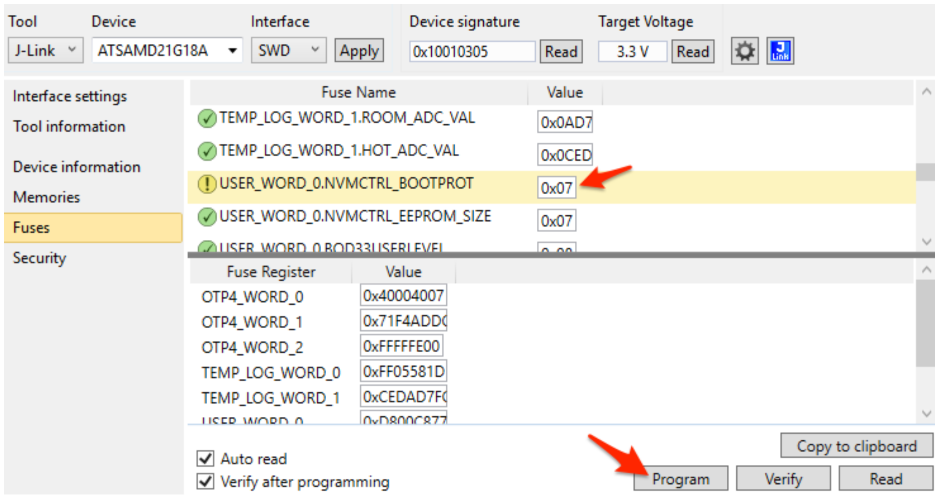
J-Link (801006657) - Device Programming

Tool	Device	Interface	Device signature	Target Voltage
J-Link	ATSAMD21G18A	SWD	---	---

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.

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

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



Tool	Device	Interface	Device signature	Target Voltage
J-Link	ATSAMD21G18A	SWD	0x10010305	3.3 V

Fuse Name	Value
TEMP_LOG_WORD_1.ROOM_ADC_VAL	0x0AD7
TEMP_LOG_WORD_1.HOT_ADC_VAL	0x0CED
USER_WORD_0.NVMCTRL_BOOTPROT	0x07
USER_WORD_0.NVMCTRL_EEPROM_SIZE	0x07
USER_WORD_0.BOD33USERLEVEL	0x08

Fuse Register	Value
OTP4_WORD_0	0x40004007
OTP4_WORD_1	0x71F4ADD(
OTP4_WORD_2	0xFFFFFE00
TEMP_LOG_WORD_0	0xFF05581D
TEMP_LOG_WORD_1	0xCEDAD7F(
USER_WORD_0	0xD800C877

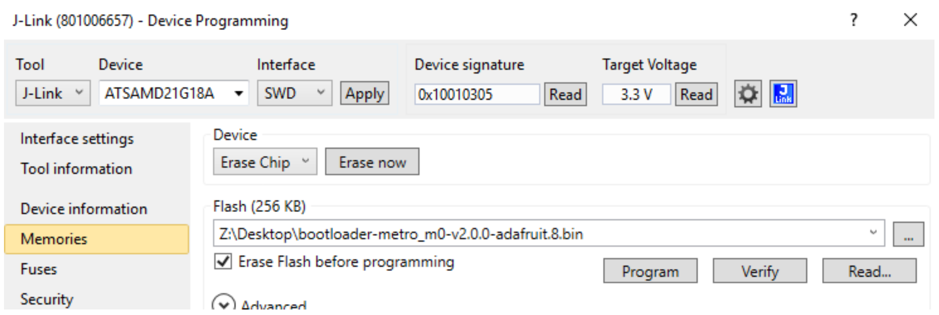
☑ Auto read
☑ Verify after programming

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**.


J-Link (801006657) - Device Programming

Tool	Device	Interface	Device signature	Target Voltage
J-Link	ATSAMD21G18A	SWD	0x10010305	3.3 V

Interface settings
Tool information
Device information
Memories
Fuses
Security

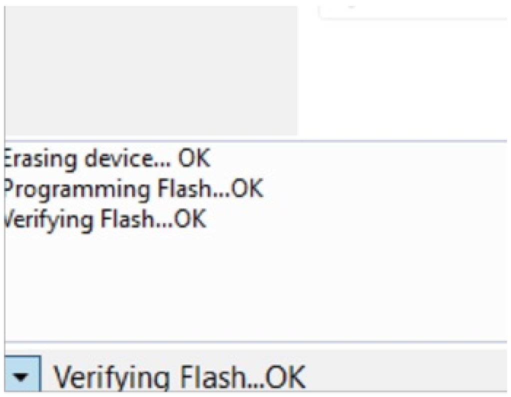
Device: Erase Chip — Erase now

Flash (256 KB): Z:\Desktop\bootloader-metro_m0-v2.0.0-adafruit.8.bin

☑ Erase Flash before programming

Program | Verify | Read...

Advanced

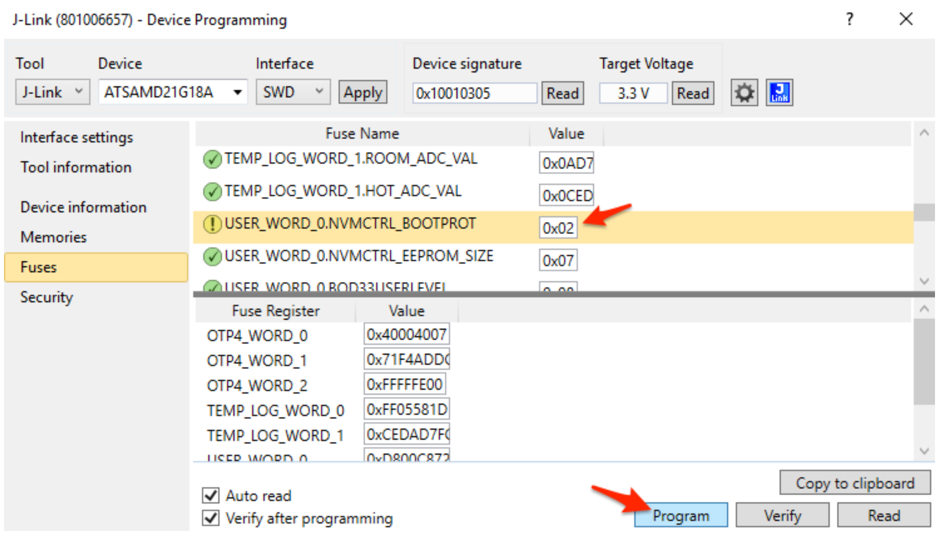

```
Erasing device... OK
Programming Flash...OK
Verifying Flash...OK
```
Verifying Flash...OK

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 to a 8kB bootloader size.

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


J-Link (801006657) - Device Programming

Tool	Device	Interface	Device signature	Target Voltage
J-Link	ATSAMD21G18A	SWD	0x10010305	3.3 V

Interface settings
Tool information
Device information
Memories
Fuses
Security

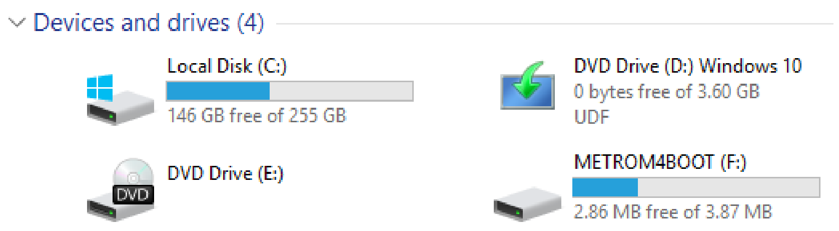
Fuse Name	Value
TEMP_LOG_WORD_1.ROOM_ADC_VAL	0x0AD7
TEMP_LOG_WORD_1.HOT_ADC_VAL	0x0CED
USER_WORD_0.NVMCTRL_BOOTPROT	0x02
USER_WORD_0.NVMCTRL_EEPROM_SIZE	0x07
USER_WORD_0.BOD33USERLEVEL	0x08

Fuse Register	Value
OTP4_WORD_0	0x40004007
OTP4_WORD_1	0x71F4ADD(
OTP4_WORD_2	0xFFFFFE00
TEMP_LOG_WORD_0	0xFF05581D
TEMP_LOG_WORD_1	0xCEDAD7F(
USER_WORD_0	0xD800C872

☑ Auto read
☑ Verify after programming

Copy to clipboard | Program | Verify | Read

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


Devices and drives (4)

Local Disk (C:)
146 GB free of 255 GB

DVD Drive (D:) Windows 10
0 bytes free of 3.60 GB
UDF

DVD Drive (E:)

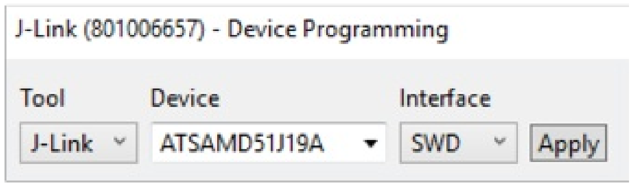
METROM4BOOT (F:)
2.86 MB free of 3.87 MB

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.


J-Link (801006657) - Device Programming

Tool: J-Link
Device: ATSAMD51J19A
Interface: SWD

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

- Feather M4 and Metro M4, and the Itsy-Bitsy M4 **select ATSAMD51J19A**
- For the Grand Central M4, **select ATSAMD51P20**

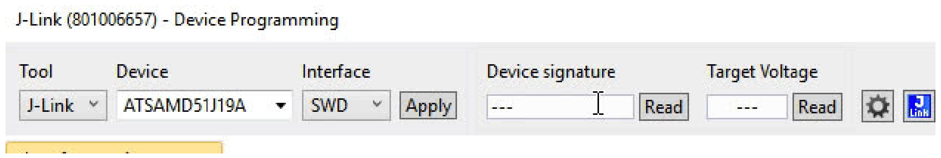
Click Apply.

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

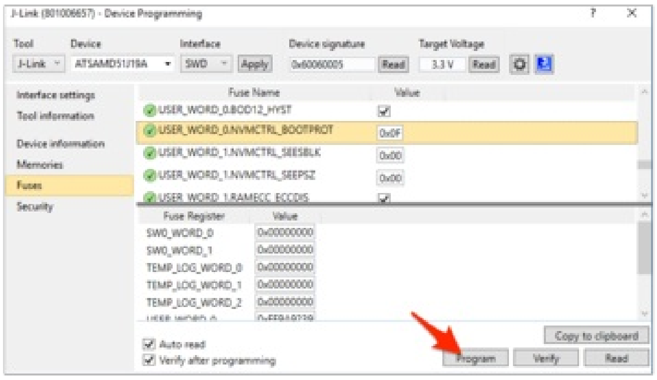
Make sure these values appear before proceeding!

- 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.


J-Link (801006657) - Device Programming

Tool: J-Link
Device: ATSAMD51J19A
Interface: SWD
Device signature: ---
Target Voltage: ---



J-Link (801006657) - Device Programming

Tool: J-Link | Device: ATSAMD51J19A | Interface: SWD | Device signature: 0x60060005 | Target Voltage: 3.3 V

Fuse Name	Value
USER_WORD_0.BOD12_HYST	✓
USER_WORD_0.NVMCTRL_BOOTPROT	0x0F
USER_WORD_1.NVMCTRL_SEESBLK	0x00
USER_WORD_1.NVMCTRL_SEEPSZ	0x00
USER_WORD_1.RAMECC_ECCDIS	✓

Fuse Register	Value
SW0_WORD_0	0x00000000
SW0_WORD_1	0x00000000
TEMP_LOG_WORD_0	0x00000000
TEMP_LOG_WORD_1	0x00000000
TEMP_LOG_WORD_2	0x00000000

☑ Auto read
☑ Verify after programming

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.

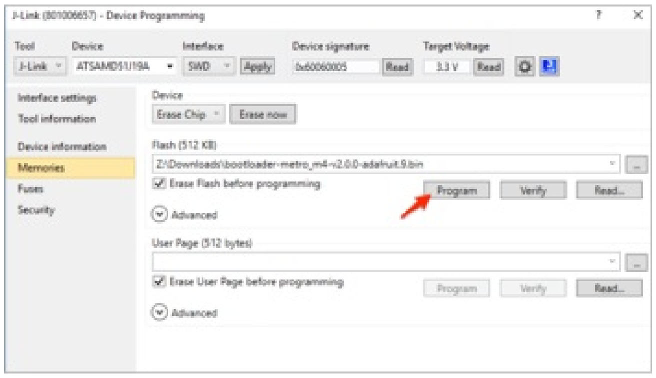
- From the Device Programming window, select **Fuses**.
- Then, **change the value of** `USER_WORD_0.NVMCTRL_BOOTPROT` **to** **`0x0F`**
- **Click Program**
- Then, click **VERIFY** to verify that the fuse has been set correctly.

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**.


J-Link (801006657) - Device Programming

Tool: J-Link
Device: ATSAMD51J19A
Interface: SWD
Device signature: 0x60060005
Target Voltage: 3.3 V

Device: Erase Chip

Flash (512 KB): Z:\Downloads\bootloader-metro_m4-v2.0.0-adafruit.9.bin
☑ Erase Flash before programming

User Page (512 bytes)
☑ Erase User Page before programming

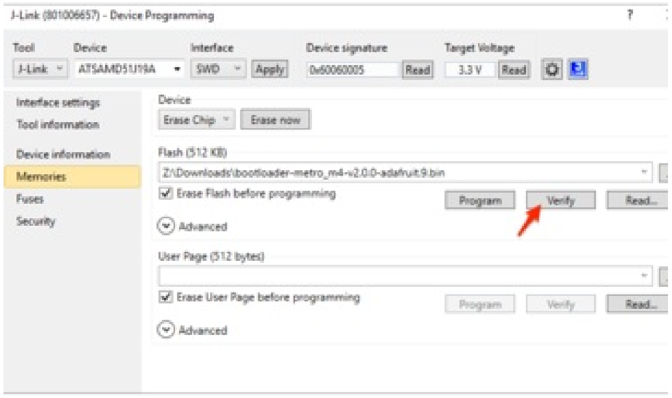
On the sidebar, **click Memories.**

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

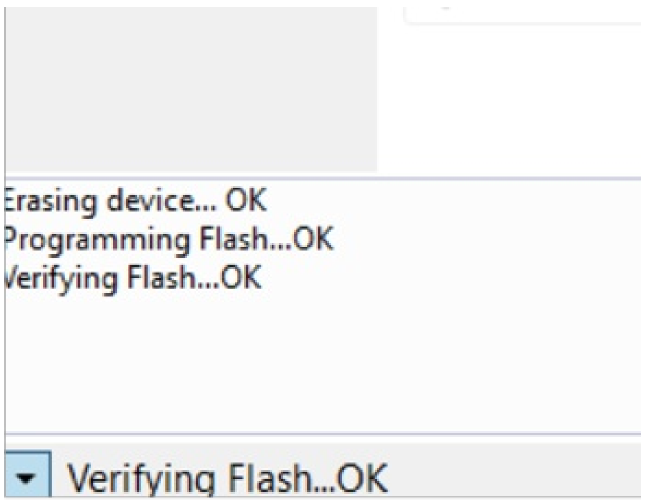

J-Link (801006657) - Device Programming

Tool: J-Link
Device: ATSAMD51J19A
Interface: SWD
Device signature: 0x60060005
Target Voltage: 3.3 V

Device: Erase Chip

Flash (512 KB): Z:\Downloads\bootloader-metro_m4-v2.0.0-adafruit.9.bin
☑ Erase Flash before programming

User Page (512 bytes)
☑ Erase User Page before programming



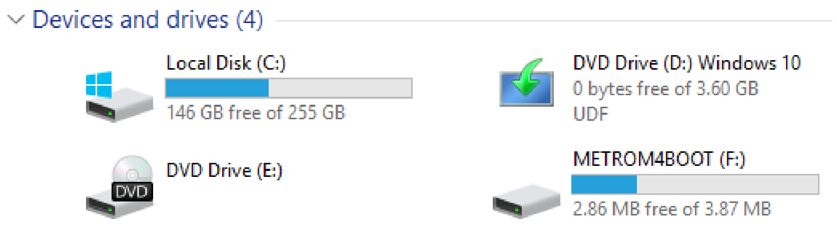
```
Erasing device... OK
Programming Flash...OK
Verifying Flash...OK
```

Verifying Flash...OK

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:



Devices and drives (4)

Local Disk (C:)
146 GB free of 255 GB

DVD Drive (D:) Windows 10
0 bytes free of 3.60 GB
UDF

DVD Drive (E:)

METROM4BOOT (F:)
2.86 MB free of 3.87 MB

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

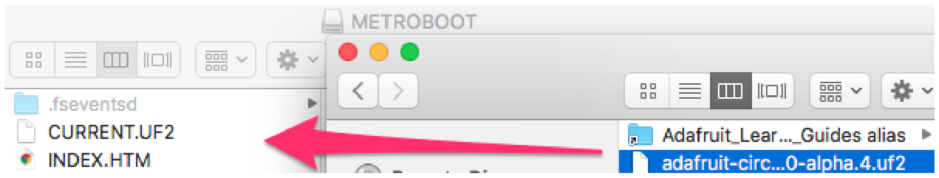
If you want to use it with CircuitPython, let's continue to installing a CircuitPython build.

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 - **boardNameBOOT** (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!