# NeoPixel Flame Torch

Created by Ruiz Brothers

https://learn.adafruit.com/neopixel-flame-torch

Last updated on 2024-03-08 02:38:34 PM EST

# Table of Contents

# Overview

## Props & Electronics

Build a 3D printed prop with a realistic looking flame! Use NeoPixels and GEMMA M0 to create a faux yet realistic looking torch. Hidden inside the torch is a mini fan and silk fabric. Use CircuitPython to easily program GEMMA M0 and NeoPixels.

## 3D Printing & Crafting

We designed and 3d printed the prop to house all of the electronics and features a modular design that allows parts to easily be swapped out. The flame is made from pieces of a silk that are cut and taped to a crossbar fitted over the neopixel jewel.

Inspired by Nick Dimelow's Medieval Torch project (https://adafru.it/M-B).

## Prerequisite Guides

Take a moment to walk through the following guides.

- GEMMA M0 Introduction (https://adafru.it/yev)
- NeoPixel Uber Guide (https://adafru.it/oe1)
- Slide switch JST Adapters (https://adafru.it/vej)

## Hardware

List of required hardware.

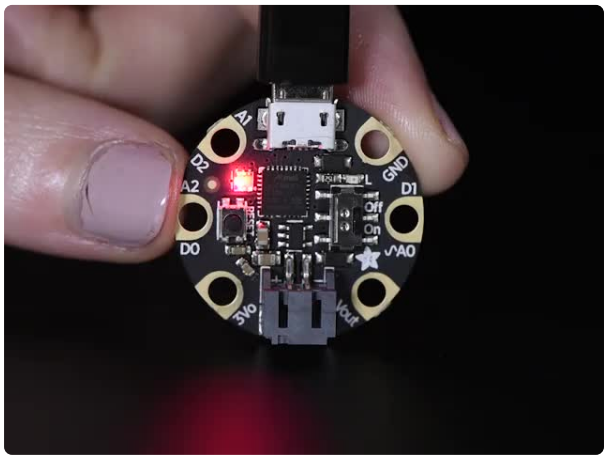
- 4x M3 x 30mm screws

- 12x M3 hex nuts
- 3x M2.5 x 6mm
- 3x M2.5 hex nuts

## Parts from Adafruit

List of parts used to build the project.

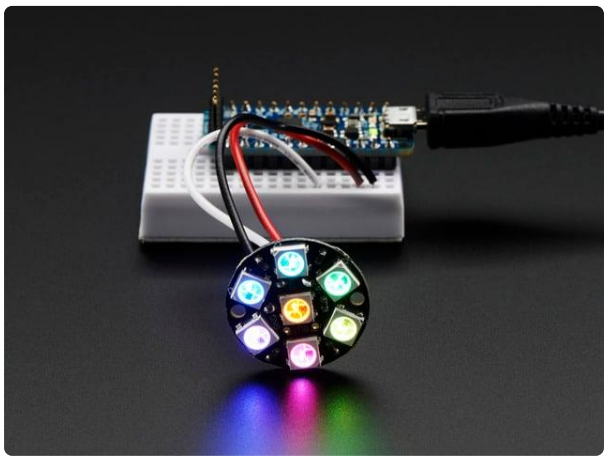
Adafruit GEMMA M0 (http://adafru.it/3501)
7x NeoPixel Jewel (http://adafru.it/2226)
Adafruit 5V MiniBoost (http://adafru.it/4654)
2200mAh battery (http://adafru.it/1781)
5VDC Fan (https://adafru.it/Lnd)
Slide switch (http://adafru.it/805)



Adafruit GEMMA M0 - Miniature wearable electronic platform

The Adafruit Gemma M0 is a super small microcontroller board, with just enough built-in to create many simple projects. It may look small and cute: round, about the...

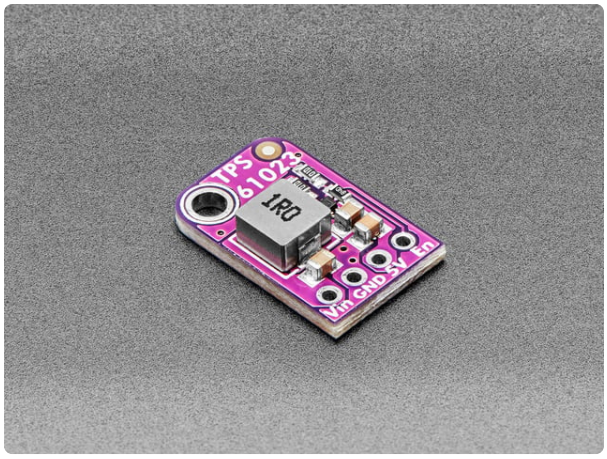
https://www.adafruit.com/product/3501



NeoPixel Jewel - 7 x 5050 RGB LED with Integrated Drivers

Be the belle of the ball with the NeoPixel Jewel!  We fit seven of our tiny 5050 (5mm x 5mm) smart RGB LEDs onto a beautiful, round PCB with mounting holes and a...

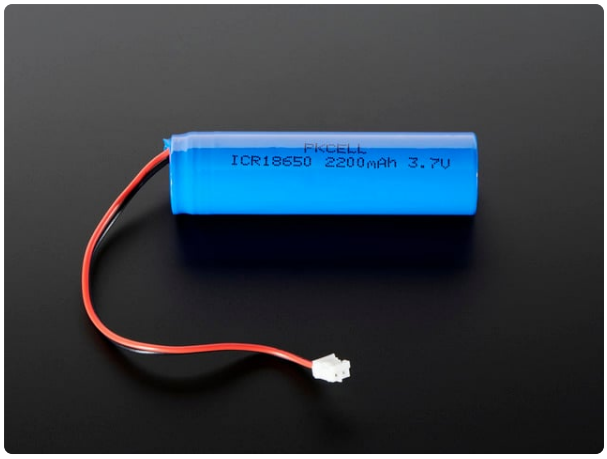
https://www.adafruit.com/product/2226




Adafruit MiniBoost 5V @ 1A - TPS61023

This adorable little board will come in very handy whenever you need a good amount of 5V power. It's the size of a linear regulator, but it's actually a mini-booster! Input...

https://www.adafruit.com/product/4654




Lithium Ion Cylindrical Battery - 3.7v 2200mAh

Need a big battery for your project? This lithium-ion battery contains a 2200mAh and a protection circuit that provides over-voltage, under-voltage, and over-current protection. Yet,...

https://www.adafruit.com/product/1781

**1 x** Silk Dancing Scarves — https://amzn.to/32vlmdB

Silk Square Scarf with Hemmed Edges - 6 Colors(24" x 24")

---

**1 x** Slide Switch — https://www.adafruit.com/product/805

Slide Switch

---

**1 x** 5VDC Mini Fan — https://www.digikey.com/product-detail/en/cui-devices/CFM-5010-03-10/102-4012-ND/6194920

CFM-5010-03-10

---

**1 x** 10-Wire Ribbon Cable — https://www.adafruit.com/product/3890

28AWG Silicone Cover Stranded Core Wire

---

**1 x** 3-pin JST cable — https://www.adafruit.com/product/4046

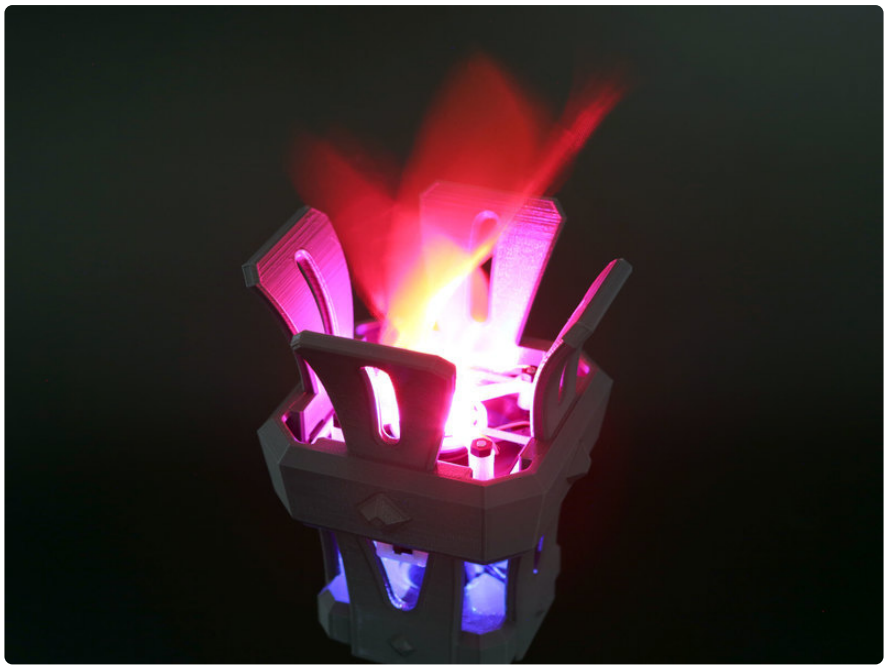
JST PH 3-Pin Socket to Color Coded Cable - 200mm

---

**1 x** 3-pin JST cable — https://www.adafruit.com/product/4336

Plug to Plug 100mm long

---

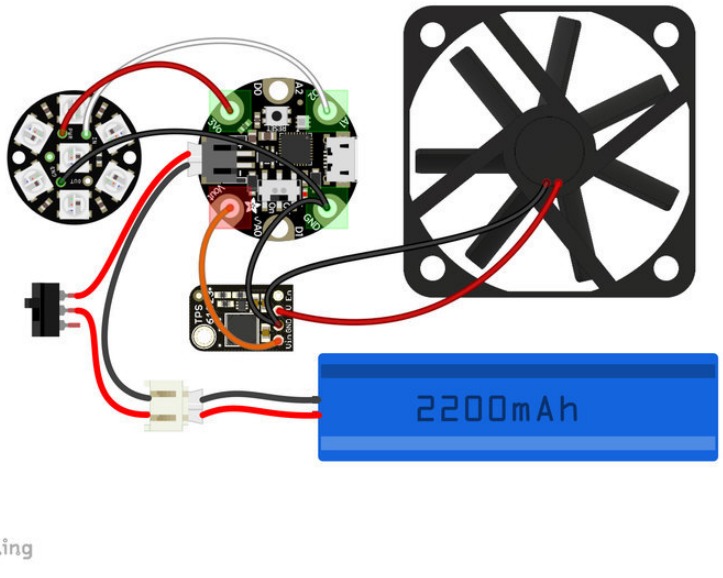


# Circuit Diagram

The diagram below provides a visual reference for wiring of the components. This diagram was created using the software package Fritzing (https://adafru.it/oEP).

## Adafruit Library for Fritzing

Use Adafruit's Fritzing parts library to create circuit diagrams for your projects. Download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts (https://adafru.it/AYZ).



## Wired Connections

NeoPixel Jewel

- **5V** from Jewel to **3V** on GEMMA
- **GND** from Jewel to **GND** on GEMMA
- **DIN** from Jewel to **D2** on GEMMA

5V MiniBoost

- **VIN** from MiniBoost to **VOUT** on GEMMA
- **GND** from MiniBoost to **GND** on GEMMA

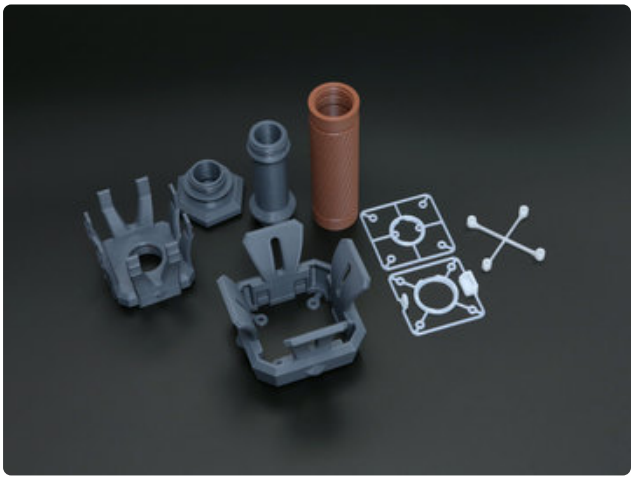
5VDC Fan

- **Voltage** from fan to **5V** on MiniBoost
- **Ground** from fan to **GND** on MiniBoost

## Powering

The Adafruit board can be powered via USB or JST using a 3.7v lipo battery. In this project, a 2200mAh lipo battery is used. The battery is rechargeable with a USB lipo charger. The switch is wired to a JST adapter.

# 3D Printing



## Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

**File names**

handle.stl
pommel.stl
collar.stl
torch-bottom.stl
torch-top.stl
gemma-mount.stl
jewel-mount.stl
flame-bar.stl
battery-clip.stl

**Fusion 360 Share Link**

https://adafru.it/M-E

**Download CAD files from PrusaPrinters**

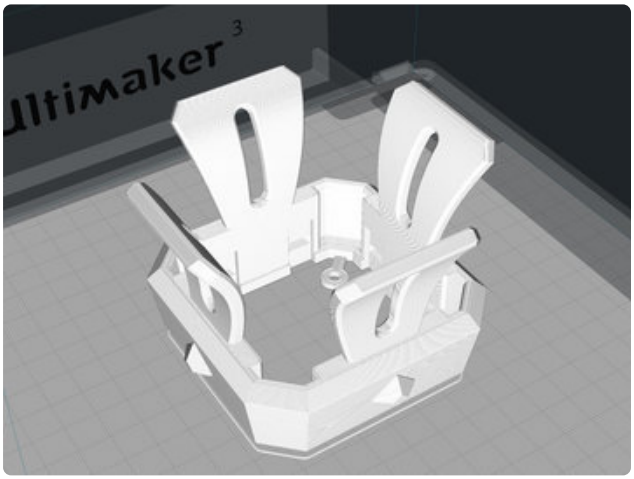
https://adafru.it/M-F

**Download CAD files from Thingiverse**

https://adafru.it/N0a

## CAD Assembly

The GEMMA M0, 5V MiniBoost, switch and NeoPixel Jewel are mounted to both sides of the 5V fan using M3 x 30mm long screws. The GEMMA M0 is press fitted into a PCB mount which is secured to the bottom of the torch cage. The fan is mounted over the GEMMA PCB mount. The NeoPixel Jewel is secured to a separate PCB mount using 2x M2.5 x 6mm long screws and nuts. The NeoPixel Jewel PCB mount is secured on top of the 5V fan. The 2200mAh battery is hidden inside the handle.

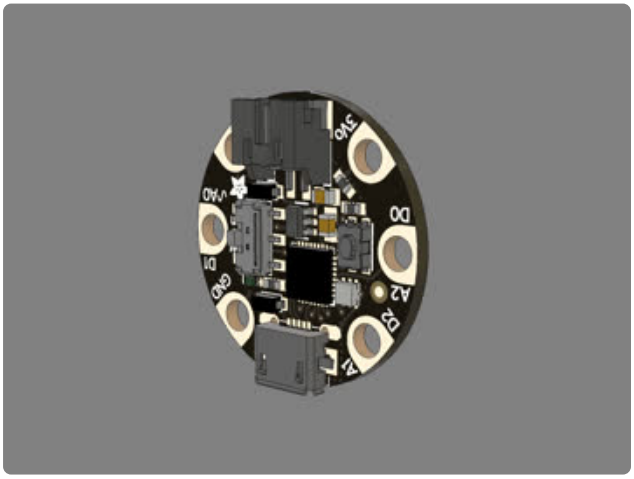



## Slicing Parts

No supports are required. Slice with setting for PLA material.

The parts were sliced using CURA using the slice settings below.

PLA filament 220c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed



## Design Source Files

The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo (https://adafru.it/AW8).

## Installing Top and Bottom Cage

Insert the fingers from the bottom half into the slots on the top half.

## Secure Top and Bottom Cage

Carefully press the fingers inward and lightly apply pressure. Slowly fit the fingers into the slots. With the nubs fitted through, firmly press together to snap fit.

## Cage Disassembly

The cage halves are designed to be removable. Squeeze all of the fingers inward on the bottom half and gently push the top up and pull them part.

# CircuitPython

As we continue to develop CircuitPython and create new releases, we will stop supporting older releases. If you are running an older version of CircuitPython, you need to update. Click the button below to download the latest!

CircuitPython is a derivative of MicroPython designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. The Gemma M0 is the first board that comes pre-loaded with CircuitPython. Simply copy and edit files on the **CIRCUITPY** drive to iterate.

Your Gemma M0 already comes with CircuitPython but maybe there's a new version, or you overwrote your Gemma M0 with Arduino code! In that case, see the below for how to reinstall or update CircuitPython. Otherwise you can skip this and go straight to the next page

If you've already plugged your board into your computer, you should see a drive called **CIRCUITPY**. The drive will contain a few files. If you want to make a 'backup' of the current firmware on the device, drag-off and save the **CURRENT.UF2** file. Other that that you can ignore the **index.htm** and **info_uf2.txt** files. They cannot be deleted and are only for informational purposes.

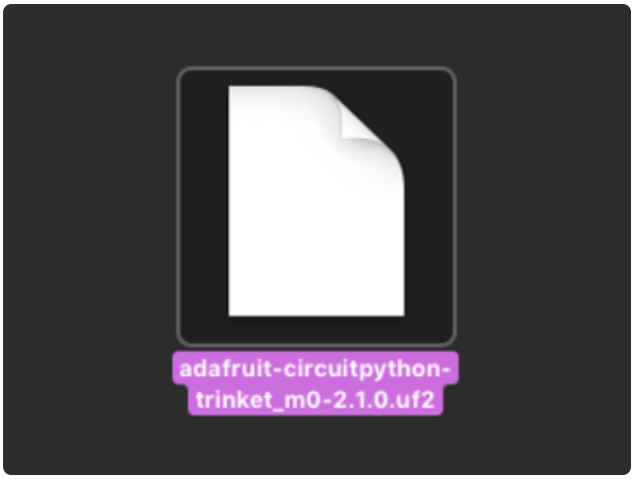
**If you have already plugged in your board, start by ejecting or "safely remove" the CIRCUITPY drive**. This is a good practice to get into. Always eject before unplugging or resetting your board!

## Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

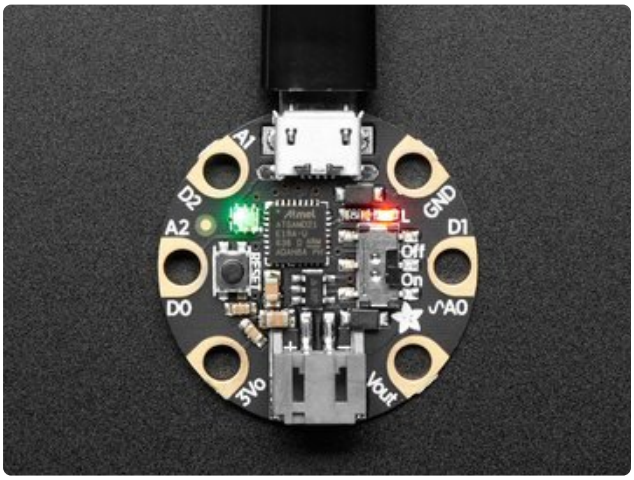
**Download the latest version of CircuitPython for this board via CircuitPython.org**

https://adafru.it/Eme




**Click the link above and download the latest UF2 file.**

Download and save it to your desktop (or wherever is handy).

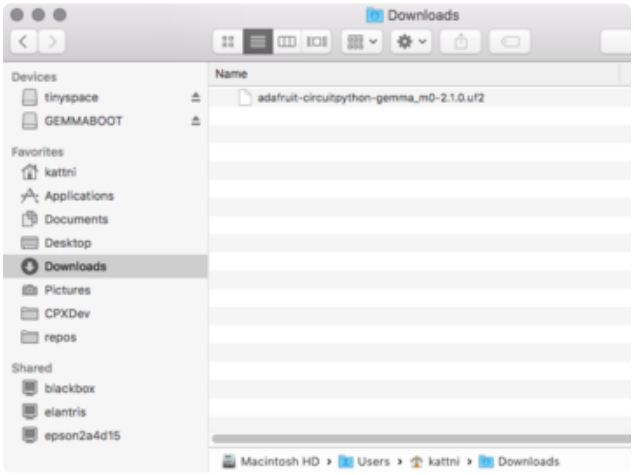



Plug your Gemma into your computer using a known-good USB cable.

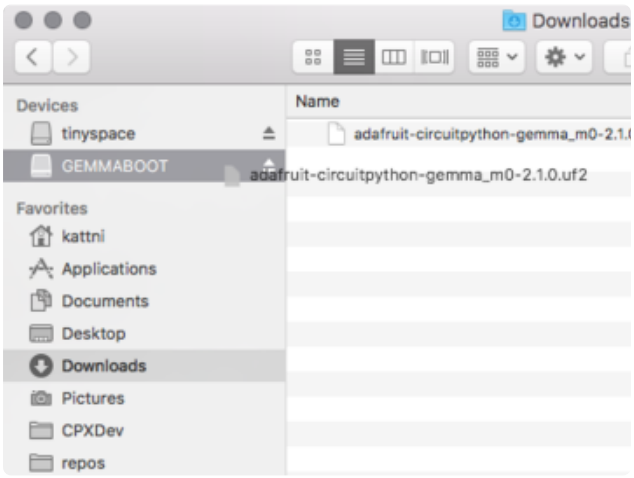
**A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.**

Double-click the small **Reset** button opposite the On/Off switch on your board. You will see the Dotstar RGB LED turn green. If it turns red, check the USB cable, try another USB port, etc. **Note:** The little LED next to the On/Off switch will be red - this is ok!

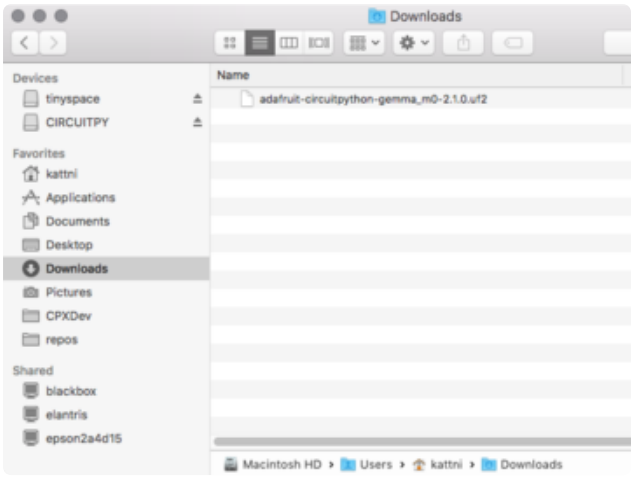
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!




You will see a new disk drive appear called **GEMMABOOT**.




Drag the **adafruit_circuitpython_etc.uf2** file to **GEMMABOOT**.




The red LED will flash. Then, the **GEMMABOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

That's it, you're done! :)

## Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython (https://adafru.it/Amd).

## Gemma Default Zip Install

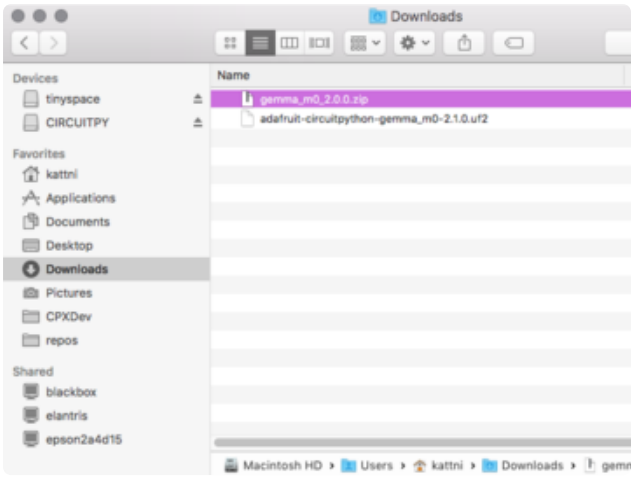
Gemma M0 is limited on space. As you begin working on projects, you may run out of space. Operating systems can create hidden files that take up space. To prevent these files from being added to your Gemma, we suggest installing the Gemma Default Zip.

**Download the Gemma Default Zip**

https://adafru.it/Aoj

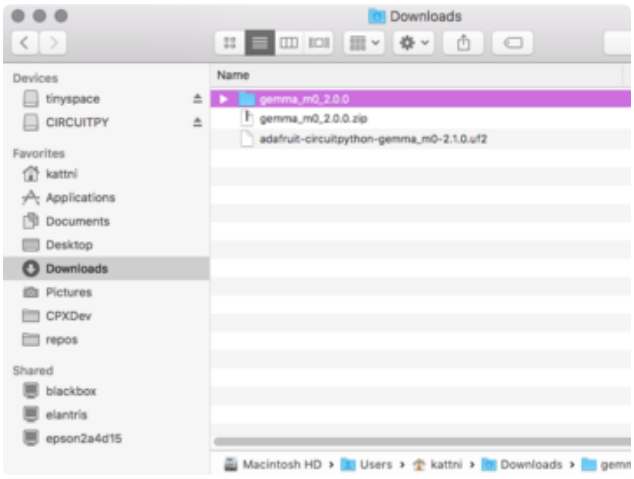

**Click the link above to download the default zip.**

Download and save it to your desktop, or wherever is handy!

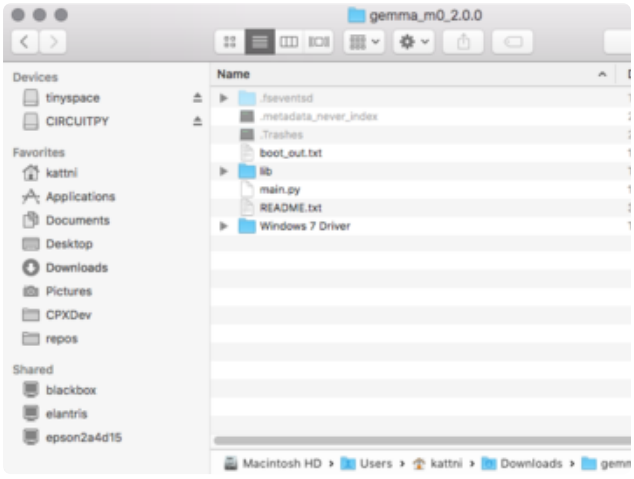



If you haven't already, **plug your Gemma into your computer using a known-good USB cable.**

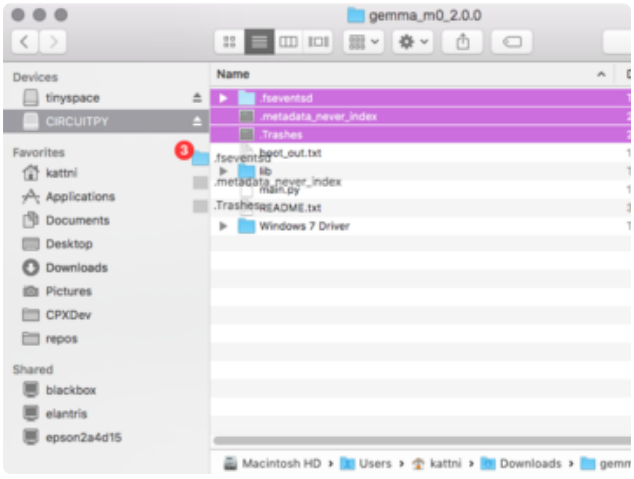
Make sure your **CIRCUITPY** drive appears.




Once downloaded, **double-click the file to extract the contents.**




Double click newly extracted folder to open it.



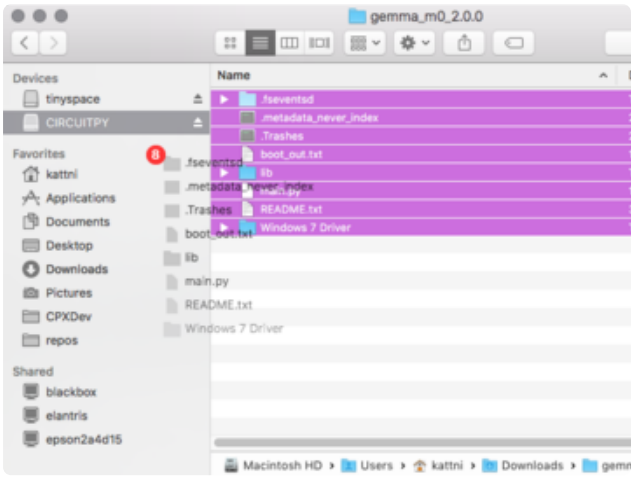

To load the files that will keep the system from adding hidden files to your drive, highlight these three files:

**.fseventsd**

**.metadata_never_index**

**.Trashes**

Drag them to your **CIRCUITPY** drive. If it asks to replace any, say yes!




If you'd like to **reset your Gemma to the same files it shipped with**, you can do that with these files. **If you changed main.py, and you want to keep your changes, back up main.py first.**

Highlight all the files in this folder. Drag them all to your CIRCUITPY drive.

If it asks to replace anything, say yes.

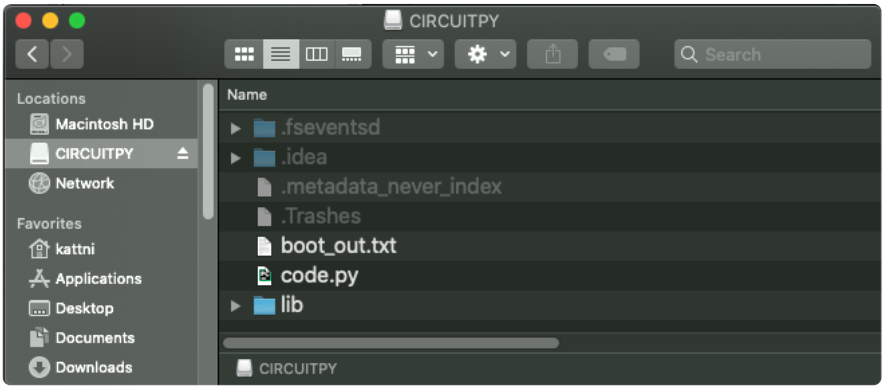
# CircuitPython Libraries

As CircuitPython development continues and there are new releases, Adafruit will stop supporting older releases. Visit https://circuitpython.org/downloads to download the latest version of CircuitPython for your board. You must download the CircuitPython Library Bundle that matches your version of CircuitPython. Please update CircuitPython and then visit https://circuitpython.org/libraries to download the latest Library Bundle.

Each CircuitPython program you run needs to have a lot of information to work. The reason CircuitPython is so simple to use is that most of that information is stored in other files and works in the background. These files are called libraries. Some of them are built into CircuitPython. Others are stored on your **CIRCUITPY** drive in a folder called **lib**. Part of what makes CircuitPython so great is its ability to store code

separately from the firmware itself. Storing code separately from the firmware makes it easier to update both the code you write and the libraries you depend.

Your board may ship with a **lib** folder already, it's in the base directory of the drive. If not, simply create the folder yourself. When you first install CircuitPython, an empty **lib** directory will be created for you.




CircuitPython libraries work in the same way as regular Python modules so the Python docs (https://adafru.it/rar) are an excellent reference for how it all should work. In Python terms, you can place our library files in the **lib** directory because it's part of the Python path by default.

One downside of this approach of separate libraries is that they are not built in. To use them, one needs to copy them to the **CIRCUITPY** drive before they can be used. Fortunately, there is a library bundle.

The bundle and the library releases on GitHub also feature optimized versions of the libraries with the **.mpy** file extension. These files take less space on the drive and have a smaller memory footprint as they are loaded.

Due to the regular updates and space constraints, Adafruit does not ship boards with the entire bundle. Therefore, you will need to load the libraries you need when you begin working with your board. You can find example code in the guides for your board that depends on external libraries.

Either way, as you start to explore CircuitPython, you'll want to know how to get libraries on board.

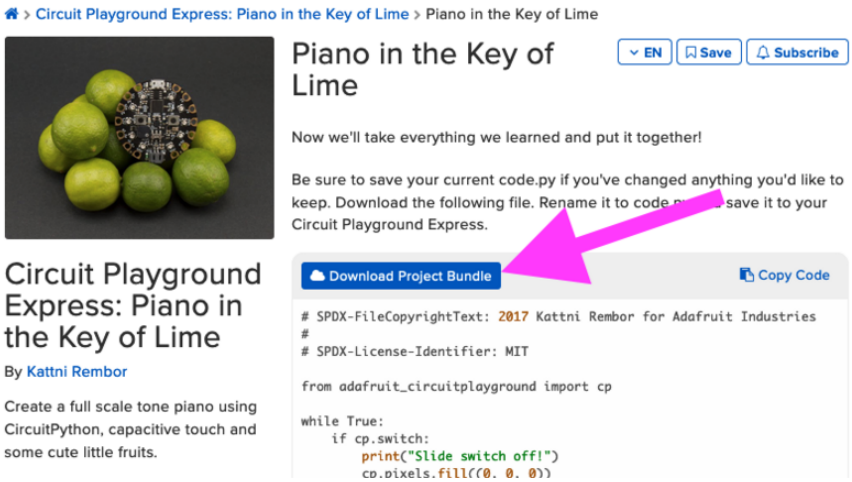
# The Adafruit Learn Guide Project Bundle

The quickest and easiest way to get going with a project from the Adafruit Learn System is by utilising the Project Bundle. Most guides now have a **Download Project**

**Bundle** button available at the top of the full code example embed. This button downloads all the necessary files, including images, etc., to get the guide project up and running. Simply click, open the resulting zip, copy over the right files, and you're good to go!

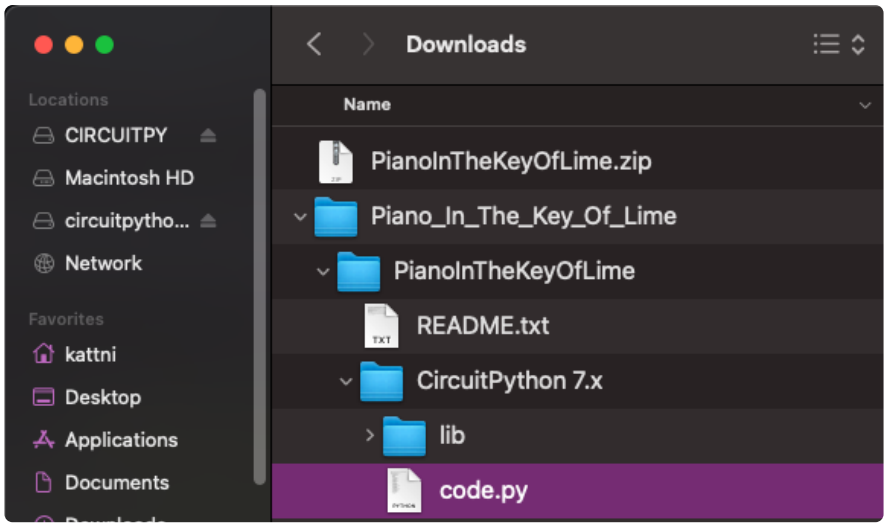
The first step is to find the Download Project Bundle button in the guide you're working on.

The Download Project Bundle button is only available on full demo code embedded from GitHub in a Learn guide. Code snippets will NOT have the button available.




When you copy the contents of the Project Bundle to your CIRCUITPY drive, it will replace all the existing content! If you don't want to lose anything, ensure you copy your current code to your computer before you copy over the new Project Bundle content!

The Download Project Bundle button downloads a zip file. This zip contains a series of directories, nested within which is the **code.py**, any applicable assets like images or audio, and the **lib/** folder containing all the necessary libraries. The following zip was downloaded from the Piano in the Key of Lime guide.




The Piano in the Key of Lime guide was chosen as an example. That guide is specific to Circuit Playground Express, and cannot be used on all boards. Do not expect to download that exact bundle and have it work on your non-CPX microcontroller.

When you open the zip, you'll find some nested directories. Navigate through them until you find what you need. You'll eventually find a directory for your CircuitPython version (in this case, 7.x). In the version directory, you'll find the file and directory you need: **code.py** and **lib/**. Once you find the content you need, you can copy it all over to your **CIRCUITPY** drive, replacing any files already on the drive with the files from the freshly downloaded zip.

In some cases, there will be other files such as audio or images in the same directory as code.py and lib/. Make sure you include all the files when you copy things over!

Once you copy over all the relevant files, the project should begin running! If you find that the project is not running as expected, make sure you've copied ALL of the project files onto your microcontroller board.

That's all there is to using the Project Bundle!

# The Adafruit CircuitPython Library Bundle

Adafruit provides CircuitPython libraries for much of the hardware they provide, including sensors, breakouts and more. To eliminate the need for searching for each library individually, the libraries are available together in the Adafruit CircuitPython Library Bundle. The bundle contains all the files needed to use each library.

## Downloading the Adafruit CircuitPython Library Bundle

You can download the latest Adafruit CircuitPython Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

**Match up the bundle version with the version of CircuitPython you are running.** For example, you would download the 6.x library bundle if you're running any version of CircuitPython 6, or the 7.x library bundle if you're running any version of CircuitPython 7, etc. If you mix libraries with major CircuitPython versions, you will get incompatible mpy errors due to changes in library interfaces possible during major version changes.

**Click to visit circuitpython.org for the latest Adafruit CircuitPython Library Bundle**

https://adafru.it/ENC

**Download the bundle version that matches your CircuitPython firmware version.** If you don't know the version, check the version info in **boot_out.txt** file on the **CIRCUITPY** drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

There's also a **py** bundle which contains the uncompressed python files, you probably don't want that unless you are doing advanced work on libraries.

# The CircuitPython Community Library Bundle

The CircuitPython Community Library Bundle is made up of libraries written and provided by members of the CircuitPython community. These libraries are often written when community members encountered hardware not supported in the Adafruit Bundle, or to support a personal project. The authors all chose to submit these libraries to the Community Bundle make them available to the community.

**These libraries are maintained by their authors and are not supported by Adafruit.** As you would with any library, if you run into problems, feel free to file an issue on the GitHub repo for the library. Bear in mind, though, that most of these libraries are supported by a single person and you should be patient about receiving a response.

Remember, these folks are not paid by Adafruit, and are volunteering their personal time when possible to provide support.

## Downloading the CircuitPython Community Library Bundle

You can download the latest CircuitPython Community Library Bundle release by clicking the button below. The libraries are being constantly updated and improved, so you'll always want to download the latest bundle.

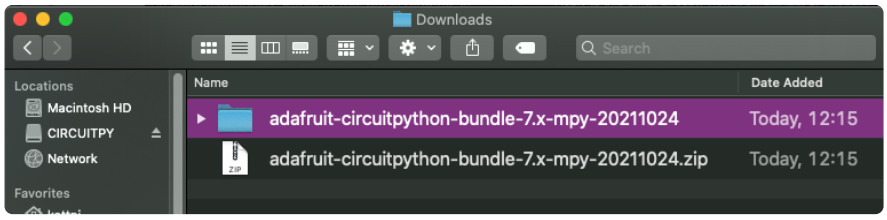
**Click for the latest CircuitPython Community Library Bundle release**

https://adafru.it/VCn

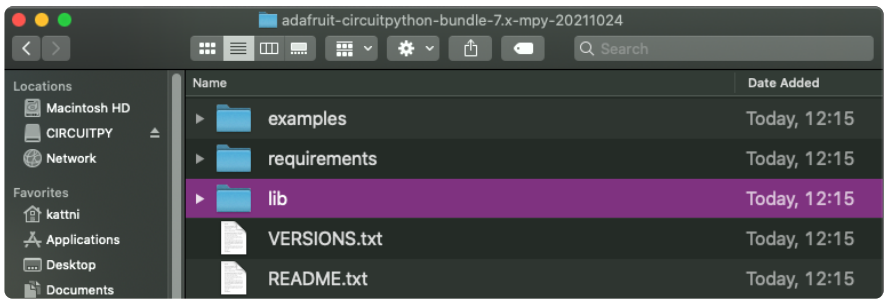
The link takes you to the latest release of the CircuitPython Community Library Bundle on GitHub. There are multiple versions of the bundle available. **Download the bundle version that matches your CircuitPython firmware version.** If you don't know the version, check the version info in **boot_out.txt** file on the **CIRCUITPY** drive, or the initial prompt in the CircuitPython REPL. For example, if you're running v7.0.0, download the 7.x library bundle.

# Understanding the Bundle

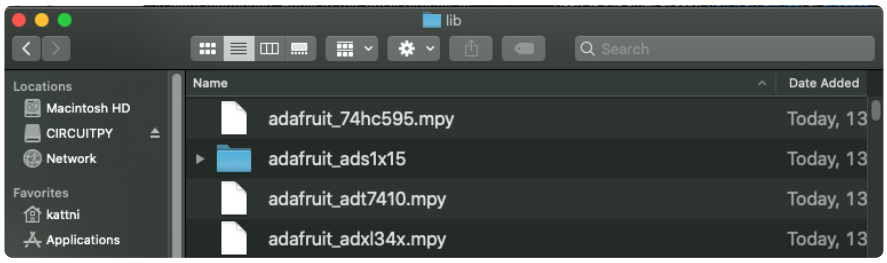
After downloading the zip, extract its contents. This is usually done by double clicking on the zip. On Mac OSX, it places the file in the same directory as the zip.




Open the bundle folder. Inside you'll find two information files, and two folders. One folder is the lib bundle, and the other folder is the examples bundle.



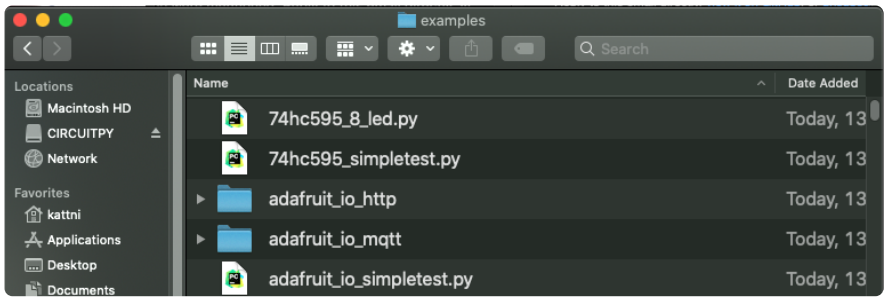
Now open the lib folder. When you open the folder, you'll see a large number of **.mpy** files, and folders.




## Example Files

All example files from each library are now included in the bundles in an **examples** directory (as seen above), as well as an examples-only bundle. These are included for two main reasons:

- Allow for quick testing of devices.
- Provide an example base of code, that is easily built upon for individualized purposes.




## Copying Libraries to Your Board

First open the **lib** folder on your **CIRCUITPY** drive. Then, open the **lib** folder you extracted from the downloaded zip. Inside you'll find a number of folders and **.mpy** files. Find the library you'd like to use, and copy it to the **lib** folder on **CIRCUITPY**.

If the library is a directory with multiple **.mpy** files in it, be sure to **copy the entire folder to CIRCUITPY/lib**.

This also applies to example files. Open the **examples** folder you extracted from the downloaded zip, and copy the applicable file to your **CIRCUITPY** drive. Then, rename it to **code.py** to run it.

If a library has multiple .mpy files contained in a folder, be sure to copy the entire folder to CIRCUITPY/lib.

# Understanding Which Libraries to Install

You now know how to load libraries on to your CircuitPython-compatible microcontroller board. You may now be wondering, how do you know which libraries you need to install? Unfortunately, it's not always straightforward. Fortunately, there is an obvious place to start, and a relatively simple way to figure out the rest. First up: the best place to start.

When you look at most CircuitPython examples, you'll see they begin with one or more `import` statements. These typically look like the following:

- `import library_or_module`

However, `import` statements can also sometimes look like the following:

- `from library_or_module import name`
- `from library_or_module.subpackage import name`
- `from library_or_module import name as local_name`

They can also have more complicated formats, such as including a `try` / `except` block, etc.

The important thing to know is that **an `import` statement will always include the name of the module or library that you're importing**.

Therefore, the best place to start is by reading through the `import` statements.

Here is an example import list for you to work with in this section. There is no setup or other code shown here, as the purpose of this section involves only the import list.

```
import time
import board
import neopixel
import adafruit_lis3dh
import usb_hid
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode
```

Keep in mind, not all imported items are libraries. Some of them are almost always built-in CircuitPython modules. How do you know the difference? Time to visit the REPL.

In the Interacting with the REPL section (https://adafru.it/Awz) on The REPL page (https://adafru.it/Awz) in this guide, the `help("modules")` command is discussed. This command provides a list of all of the built-in modules available in CircuitPython for your board. So, if you connect to the serial console on your board, and enter the REPL, you can run `help("modules")` to see what modules are available for your board. Then, as you read through the `import` statements, you can, for the purposes of figuring out which libraries to load, ignore the statement that import modules.

The following is the list of modules built into CircuitPython for the Feather RP2040. Your list may look similar or be anything down to a significant subset of this list for smaller boards.

```
>>> help("modules")
__main__          board             micropython       storage
_bleio            builtins          msgpack           struct
adafruit_bus_device                 busio             neopixel_write    supervisor
adafruit_pixelbuf collections       onewireio         synthio
aesio             countio           os                sys
alarm             digitalio         paralleldisplay   terminalio
analogio          displayio         pulseio           time
array             errno             pwmio             touchio
atexit            fontio            qrio              traceback
audiobusio        framebufferio     rainbowio         ulab
audiocore         gc                random            usb_cdc
audiomixer        getpass           re                usb_hid
audiomp3          imagecapture      rgbmatrix         usb_midi
audiopwmio        io                rotaryio          vectorio
binascii          json              rp2pio            watchdog
bitbangio         keypad            rtc
bitmaptools       math              sdcardio
bitops            microcontroller   sharpdisplay
```

Now that you know what you're looking for, it's time to read through the import statements. The first two, `time` and `board`, are on the modules list above, so they're built-in.

The next one, `neopixel`, is not on the module list. That means it's your first library! So, you would head over to the bundle zip you downloaded, and search for **neopixel**. There is a **neopixel.mpy** file in the bundle zip. Copy it over to the **lib** folder on your **CIRCUITPY** drive. The following one, `adafruit_lis3dh`, is also not on the module list. Follow the same process for **adafruit_lis3dh**, where you'll find **adafruit_lis3dh.mpy**, and copy that over.

The fifth one is `usb_hid`, and it is in the modules list, so it is built in. Often all of the built-in modules come first in the import list, but sometimes they don't! Don't assume

that everything after the first library is also a library, and verify each import with the modules list to be sure. Otherwise, you'll search the bundle and come up empty!

The final two imports are not as clear. Remember, when `import` statements are formatted like this, the first thing after the `from` is the library name. In this case, the library name is `adafruit_hid`. A search of the bundle will find an **adafruit_hid folder**. When a library is a folder, you must copy the **entire folder and its contents as it is in the bundle** to the **lib** folder on your **CIRCUITPY** drive. In this case, you would copy the entire **adafruit_hid** folder to your **CIRCUITPY/lib** folder.

Notice that there are two imports that begin with `adafruit_hid`. Sometimes you will need to import more than one thing from the same library. Regardless of how many times you import the same library, you only need to load the library by copying over the **adafruit_hid** folder once.

That is how you can use your example code to figure out what libraries to load on your CircuitPython-compatible board!

There are cases, however, where libraries require other libraries internally. The internally required library is called a dependency. In the event of library dependencies, the easiest way to figure out what other libraries are required is to connect to the serial console and follow along with the `ImportError` printed there. The following is a very simple example of an `ImportError`, but the concept is the same for any missing library.

## Example: `ImportError` Due to Missing Library

If you choose to load libraries as you need them, or you're starting fresh with an existing example, you may end up with code that tries to use a library you haven't yet loaded.  This section will demonstrate what happens when you try to utilise a library that you don't have loaded on your board, and cover the steps required to resolve the issue.

This demonstration will only return an error if you do not have the required library loaded into the **lib** folder on your **CIRCUITPY** drive.

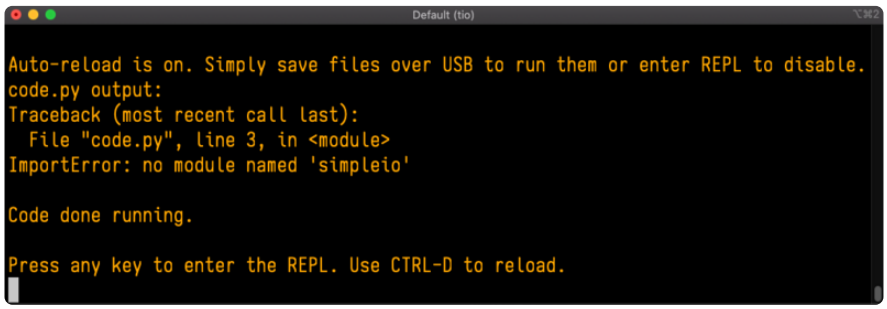
Let's use a modified version of the Blink example.

```
import board
import time
import simpleio
```

```
led = simpleio.DigitalOut(board.LED)

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

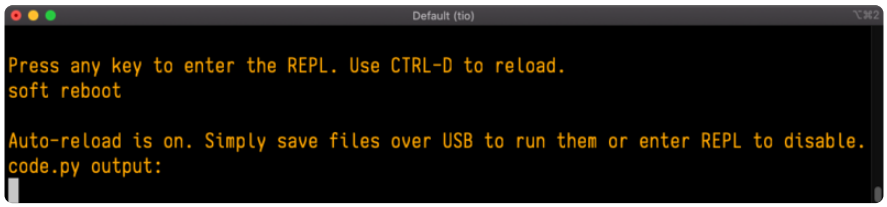
Save this file. Nothing happens to your board. Let's check the serial console to see what's going on.




You have an `ImportError`. It says there is `no module named 'simpleio'`. That's the one you just included in your code!

Click the link above to download the correct bundle. Extract the lib folder from the downloaded bundle file. Scroll down to find **simpleio.mpy**. This is the library file you're looking for! Follow the steps above to load an individual library file.

The LED starts blinking again! Let's check the serial console.




No errors! Excellent. You've successfully resolved an `ImportError`!

If you run into this error in the future, follow along with the steps above and choose the library that matches the one you're missing.

# Library Install on Non-Express Boards

If you have an M0 non-Express board such as Trinket M0, Gemma M0, QT Py M0, or one of the M0 Trinkeys, you'll want to follow the same steps in the example above to install libraries as you need them. Remember, you don't need to wait for an `ImportError` if you know what library you added to your code. Open the library

bundle you downloaded, find the library you need, and drag it to the **lib** folder on your **CIRCUITPY** drive.

You can still end up running out of space on your M0 non-Express board even if you only load libraries as you need them. There are a number of steps you can use to try to resolve this issue. You'll find suggestions on the Troubleshooting page (https://adafru.it/Den).

# Updating CircuitPython Libraries and Examples

Libraries and examples are updated from time to time, and it's important to update the files you have on your **CIRCUITPY** drive.

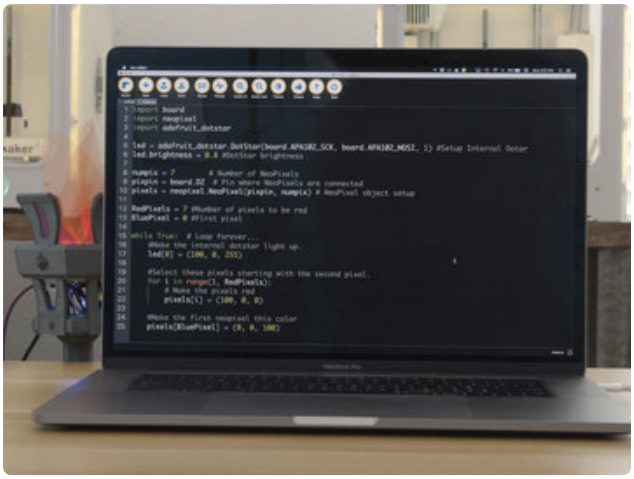
To update a single library or example, follow the same steps above. When you drag the library file to your lib folder, it will ask if you want to replace it. Say yes. That's it!

A new library bundle is released every time there's an update to a library. Updates include things like bug fixes and new features. It's important to check in every so often to see if the libraries you're using have been updated.

## CircUp CLI Tool

There is a command line interface (CLI) utility called CircUp (https://adafru.it/Tfi) that can be used to easily install and update libraries on your device. Follow the directions on the install page within the CircUp learn guide (https://adafru.it/-Ad). Once you've got it installed you run the command `circup update` in a terminal to interactively update all libraries on the connected CircuitPython device. See the usage page in the CircUp guide (https://adafru.it/-Ah) for a full list of functionality

# Code



## Coding

This project uses the neopixel library from the CircuitPython. For more information on NeoPixel animations, check out the LED animation guide (https://adafru.it/LZF) and the NeoPixel Uber Guide (https://adafru.it/CYZ).

## Download the Adafruit CircuitPython Library Bundle

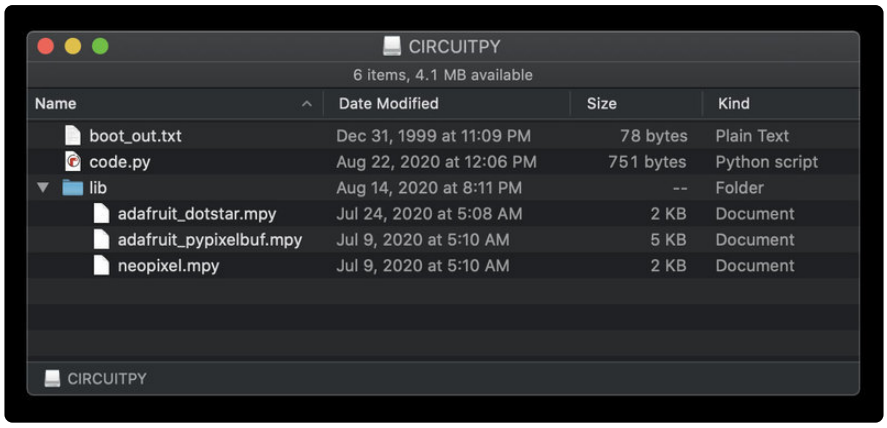
In order to run the code, we'll need to download a few libraries. Libraries contain code to help interface with hardware a lot easier for us.

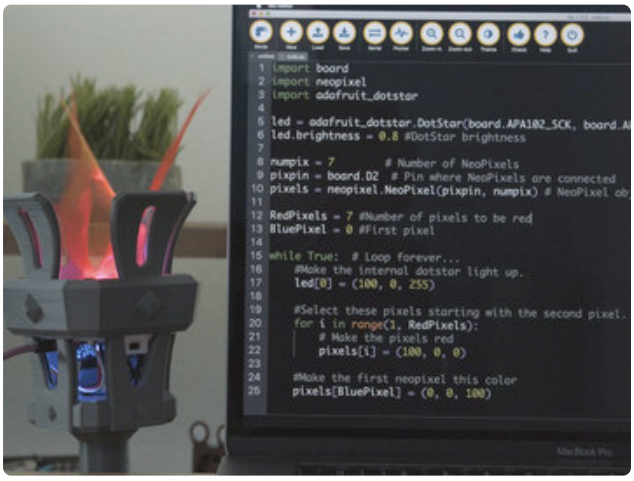
Use the GEMMA M0 page on Installing CircuitPython (https://adafru.it/zAl) to get the library that matches the major version of CircuitPython you are using noted above.

To run the code for this project, we need the three libraries in the Required Libraries list below. Unzip the library bundle and search for the libraries. Drag and drop the files into a folder named **lib** on the **CIRCUITPY** drive (which appears when your board is plugged into your computer via a known good USB cable) if the directory is not already on the GEMMA M0).

## Required Libraries

- **adafruit_dotstar.mpy**
- **adafruit_pypixelbuf.mpy**
- **neopixel.mpy**



## The Mu Python Editor

Mu is a simple Python editor that works with Adafruit CircuitPython hardware. It's written in Python and works on Windows, MacOS, Linux and Raspberry Pi. The serial console is built right in, so you get immediate feedback from your board's serial output! While you can use any text editor with your code, Mu makes it super simple. Instructions for Mu are available here (https://adafru.it/ANO).

## Installing or upgrading CircuitPython

You should ensure you have CircuitPython 5.0 or greater on your board. Plug your board in with a known good data + power cable (not the cheesy USB cable that comes with USB power packs, they are power only). You should see a new flash drive pop up.

If the drive is **CIRCUITPY**, then open the **boot_out.txt** file to ensure the version number is 5.0 or greater.

```
Adafruit CircuitPython 5.3.1 on 2020-07-13; Adafruit Gemma M0 with samd21e18
```

Click on the download link below to grab the project code directly from GitHub.

## Upload Code

**Ensure the file** is named **code.py** and drop it onto the **CIRCUITPY** drive main (root) directory that appears when your **GEMMA M0** is plugged into your computer via a

known good USB data cable. The code will run properly when all of the files have been uploaded including libraries.

```
# SPDX-FileCopyrightText: 2020 Noe Ruiz for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import board
import neopixel
import adafruit_dotstar

LED = adafruit_dotstar.DotStar(board.APA102_SCK, board.APA102_MOSI, 1) #Setup
Internal Dotar
LED.brightness = 0.8 #DotStar brightness

NUMPIX = 7        # Number of NeoPixels
PIXPIN = board.D2  # Pin where NeoPixels are connected
PIXELS = neopixel.NeoPixel(PIXPIN, NUMPIX) # NeoPixel object setup

RED = 7 #Number of pixels to be red
BLUE = 0 #First pixel

while True:  # Loop forever...
    #Make the internal dotstar light up.
    LED[0] = (100, 0, 255)

    #Select these pixels starting with the second pixel.
    for i in range(1, RED):
        # Make the pixels red
        PIXELS[i] = (100, 0, 0)

    #Make the first neopixel this color
    PIXELS[BLUE] = (0, 0, 100)
```

---

# Silk Flame


## Materials

The flame is cut from silk fabric. These dance scarves (https://adafru.it/N2c) are made from a lightweight material with a low thread count. This pack includes several colors, orange and red being the ones suited for this project.

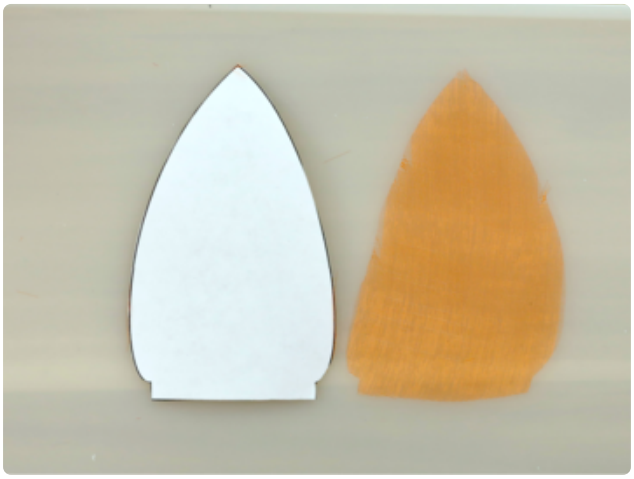
**flame_cutout.svg**

https://adafru.it/M-D

## Cutting Silk

Print out the template and cut out the shape of the flames. Attach the cutouts to the fabric by taping it to the surface. Use sharp scissors to cut out the shape from the fabric.

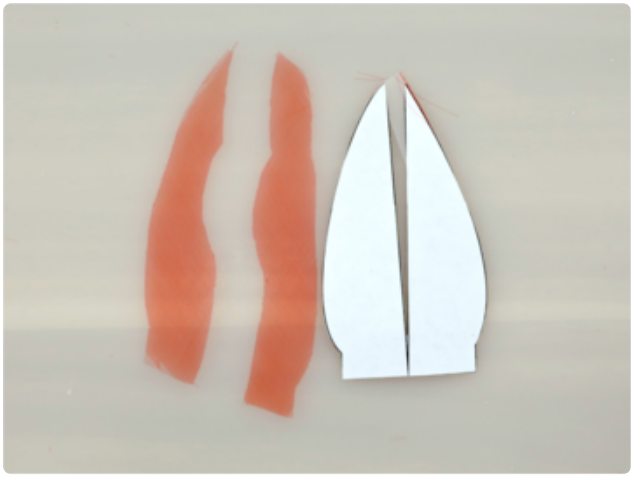
Avoid the using fabric that has creases or deformations. These can distort the movement of the flame. Use neat and clean material that with a flat and even surface.



## Flame Template

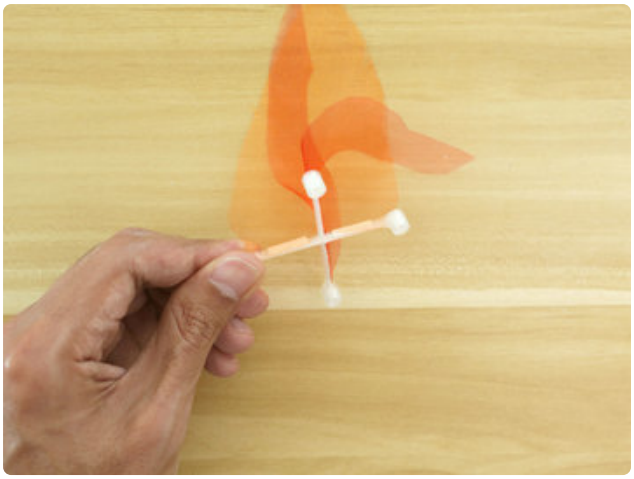
Carefully remove the template from the fabric and peel off any excess tape or loose threads.

Handle silk gently! It can be easily damaged, and can't be fixed. Avoid stretching, pinching or creasing the fabric.



## Additional Flames

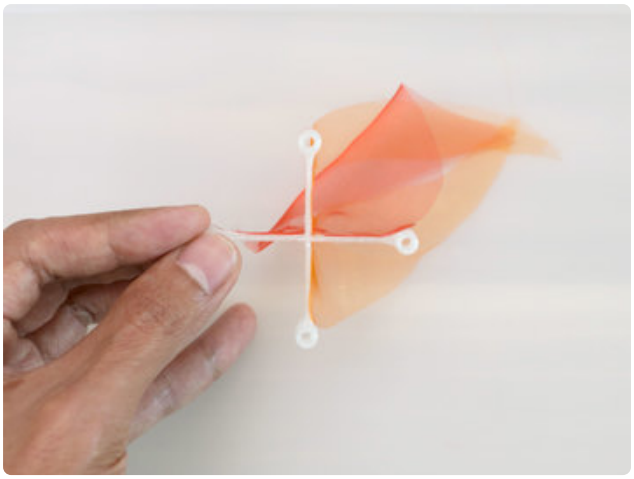
Repeat this process for the second flame.



## Secure Silk Fabric

The silk flames are attached to the crossbar using double-stick tape. Orient the flames with the crossbar so the standoffs are facing down. Reference the photo for correct placement.

Apply double stick tape to all four sides of the crossbar.



## Flame Crossbar

Attach the large single flame to the edge along the side of the crossbar. Attach the two thinner flames to the opposite side. Add pieces of double-stick tape to the crossbar first, then carefully place the fabric over.

# Wiring

## Slide Switch JST Adapter

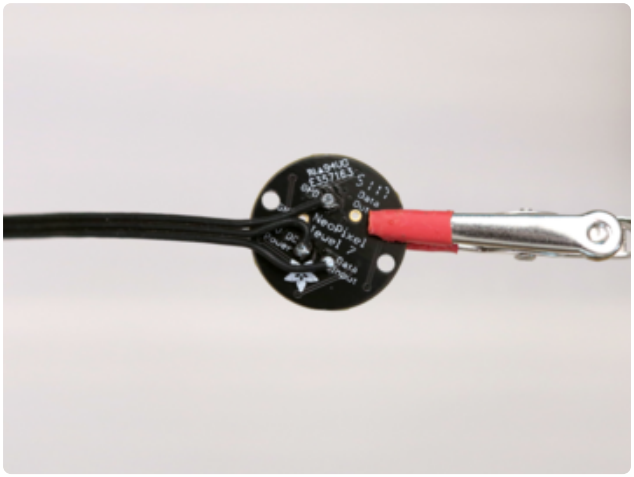
Use a 2-pin JST female and male cables to get a slide switch JST adapter. This allows the power to be cut from the battery. The switch is wired in-line with the voltage (red) wire.

## Wired Slide Switch

Solder the ground wires from both JST cables together. Solder the two voltage wires to the middle pin and either side pin on the slide switch. Use pieces of heat shrink tubing to insulate the exposed connections.

## NeoPixel Jewel JST Cable

A male 3-pin JST cable is soldered to the DIN, 5V and GND pins on the NeoPixel Jewel.

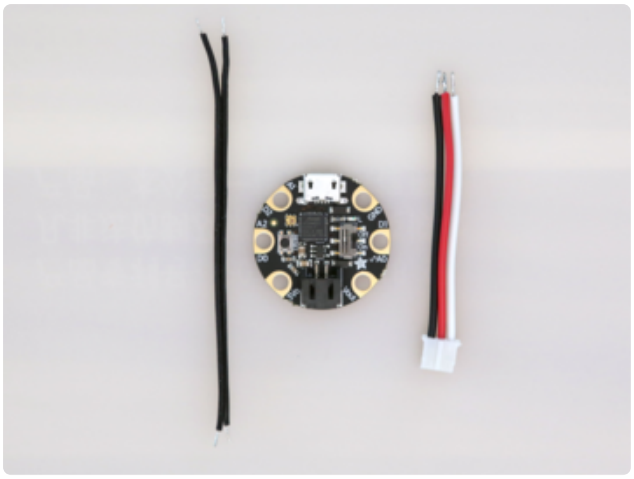


## NeoPixel Jewel JST Cable

Solder the red wire to the 5V pin, the black wire to GND and the white wire to the DIN pin on the back of the NeoPixel Jewel.

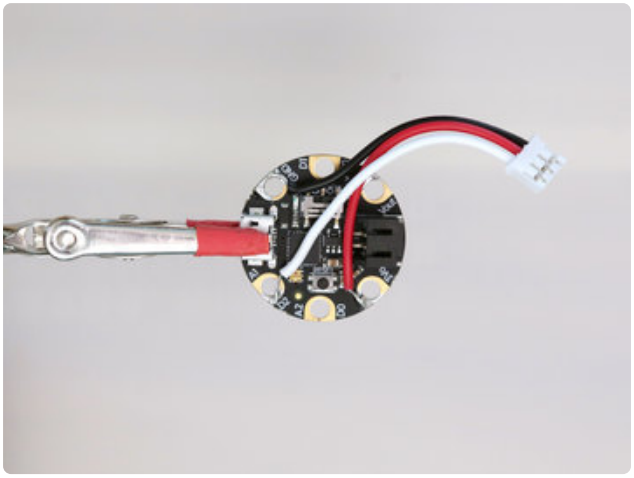
## Wired NeoPixel Jewel

Double check the solder joints are solid. Note, the cable in the photo is silicone stranded cover ribbon wire.



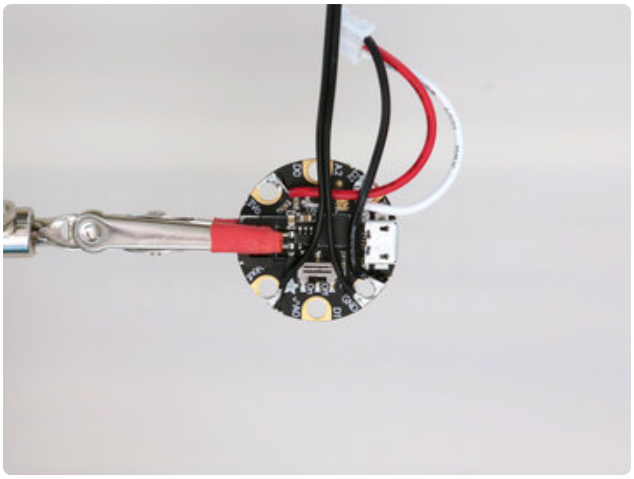
## Cables for GEMMA M0

A female 3-pin JST cable and 2-wire ribbon cable is soldered to the GEMMA M0.



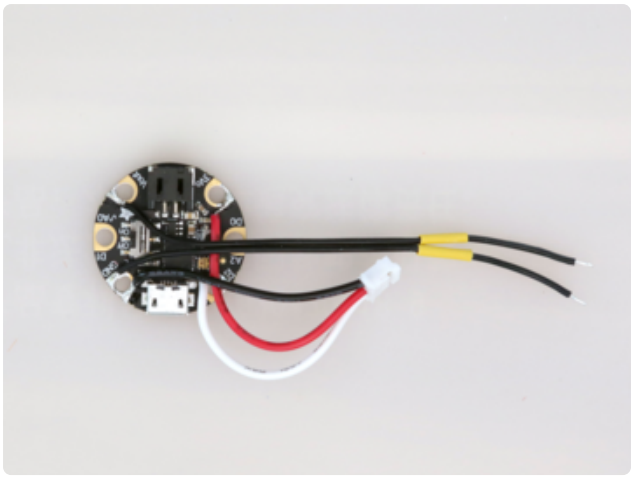
## Connect 3-pin JST to GEMMA

Solder the white wire to D2, black wire to GND and red wire to 3V.



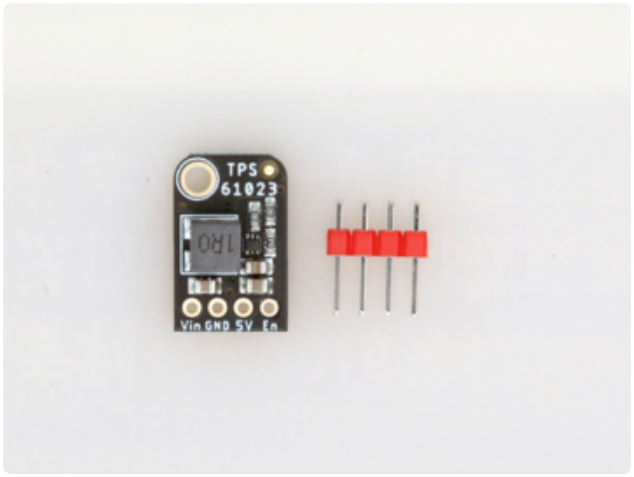
## Connect 2-wire Cable to GEMMA

Solder the 2-wire ribbon cable to the GND and VOUT pins on the GEMMA M0.



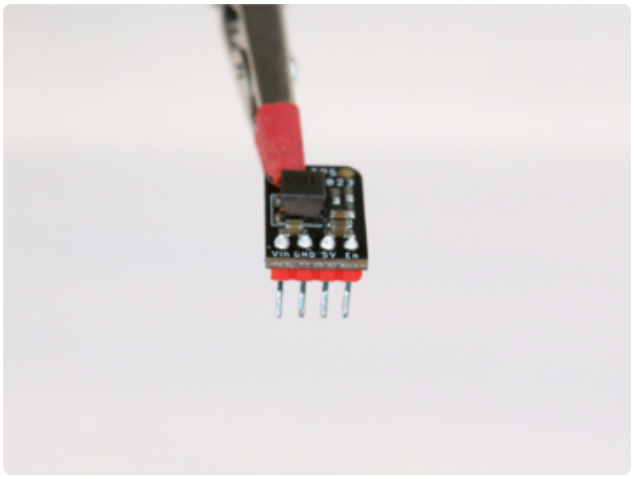
## Wired GEMMA M0

The ground pin on the GEMMA M0 shares common ground with the JST cable and 2-wire ribbon cable.




## 5V MiniBoost Headers

Install a 4-pin male headers to the 5V MiniBoost. This will make it easier to attach multiple wires to the pins.

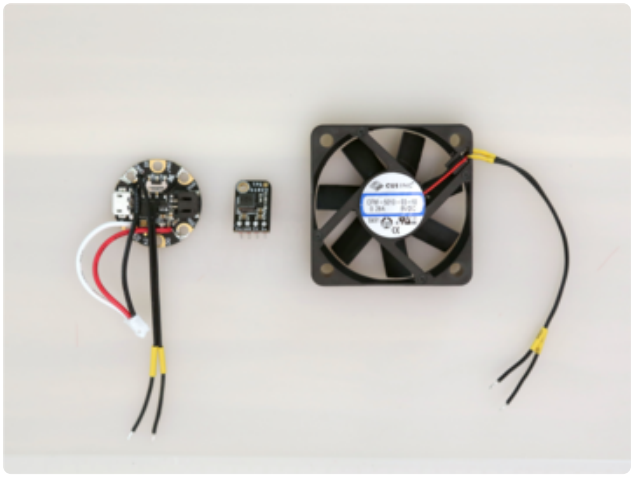


## Installed Headers

Tin the pins on the headers by adding a bit of solder. This will make it easier to attach wires to the pins.

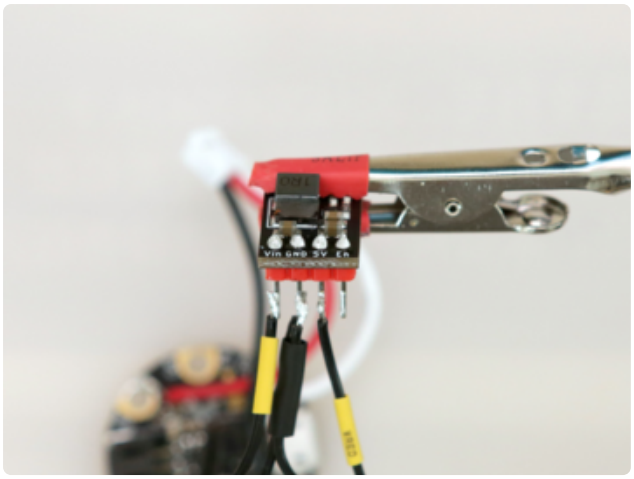
## 5V DC Fan

The fan comes with pre-soldered ground and voltage wires. Silicone cover stranded core wires can be substituted for stronger more flexible wires.



## Wiring Gemma and Fan to Boost

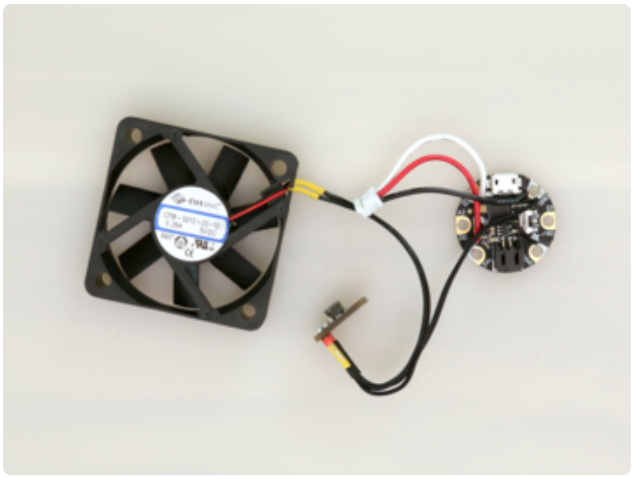
The GEMMA M0 and 5V fan are soldered to the pins on the 5V MiniBoost. The 5V MiniBoost provided the fan with proper 5V with 1A current.



## Wiring Boost

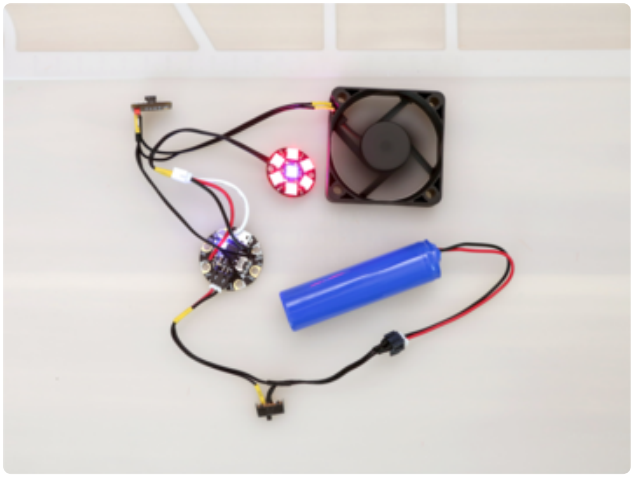
Solder the wires to the following pins:

VOUT from GEMMA to VIN on MiniBoost
GND from GEMMA to GND on MiniBoost
GND from fan to GND on MiniBoost
Voltage from fan to 5V on MiniBoost.



## Wired GEMMA and 5V Fan

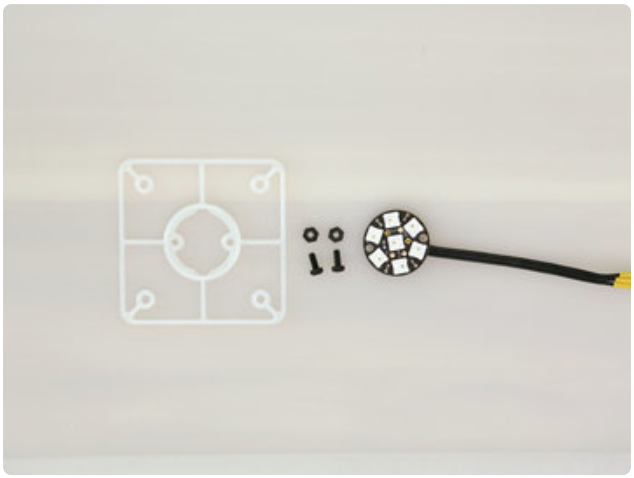
Double check the solder joints are solid.



## Circuit Testing

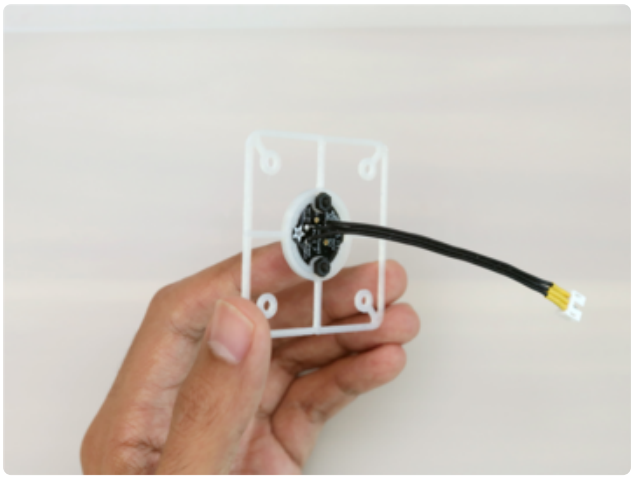
Connect the NeoPixel Jewel to the GEMMA M0. Connect the slide switch JST adapter to the GEMMA M0 and 2200mAh battery. Ensure the fan blades are not being obstructed before powering on. Use the switch to power on the circuit. Ensure the on/off switch on-board the GEMMA M0 is set to the ON position.

# Assembly



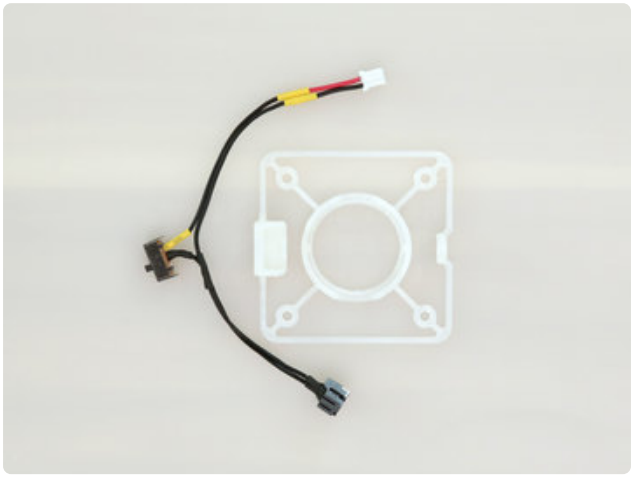
## Secure NeoPixel Jewel to Mount

The NeoPixel Jewel is secured to the jewel-mount.stl using M2.5 x 6mm long screws and hex nuts.



## Install Hardware for NeoPixel Jewel

Place the NeoPixel Jewel into the center of the jewel-mount and line up the mounting holes. Insert 2x M2.5 x 6mm screws. Secure PCB using 2x M2.5 hex nuts.

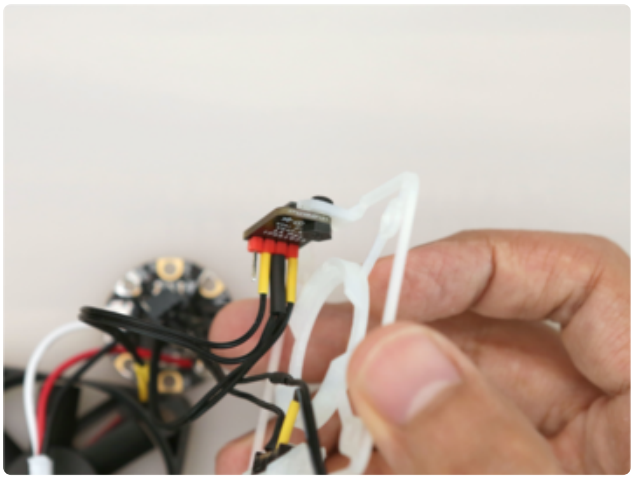


## Install Switch to GEMMA Mount

The slide switch JST adapter is press fitted into the built-in holder on the gemma-mount.

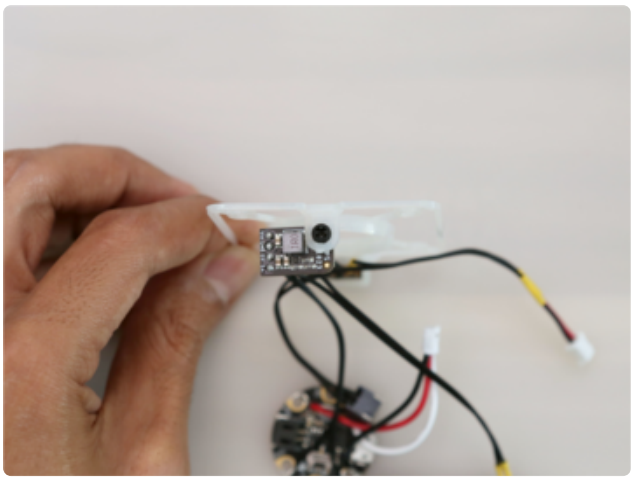
## Installed Switch

Insert the body of the slide into the built-in holder at an angle.



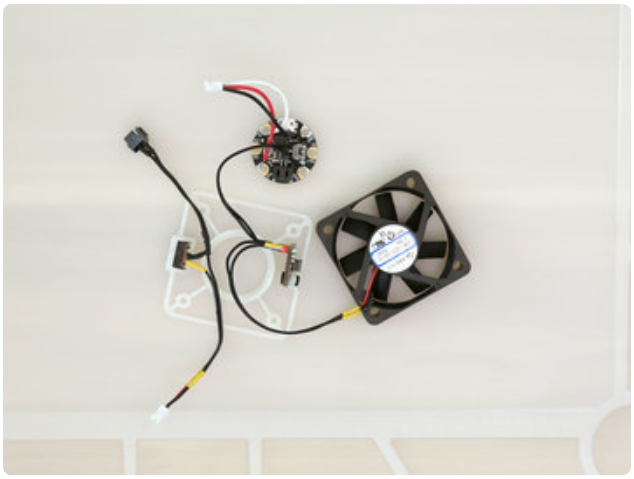
## Install 5V MiniBoost

The 5V MiniBoost is secured to the mounting tab on the side of the Gemma mount.



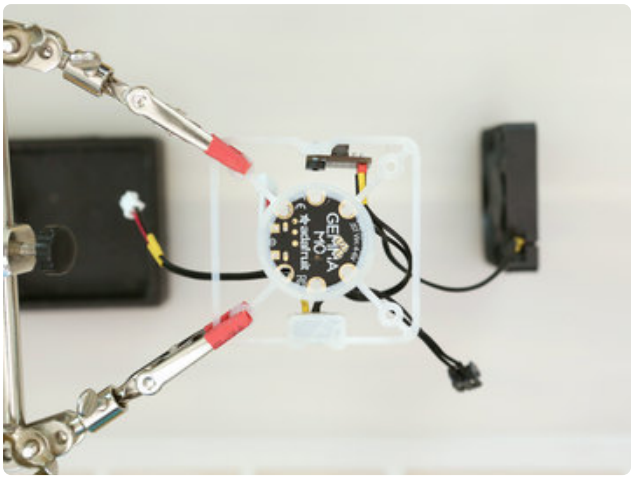
## Secure 5V MiniBoost

Use a single M2.5 x 6mm long machine screw and hex nut to secure the 5V MiniBoost to the Gemma mount.



## GEMMA Mount

The GEMMA M0 is press fitted into the center of the Gemma mount.

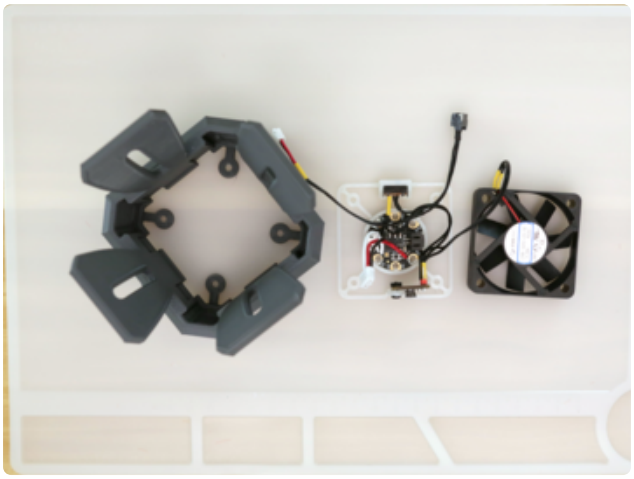


## Install GEMMA

The GEMMA M0 is inserted at an angle with one side of the PCB fitting under the nub. Carefully flex the gemma-mount to fit the other side of the PCB.

## Secured GEMMA

The GEMMA M0 should be secured to the mount with the wires and cables accessible.

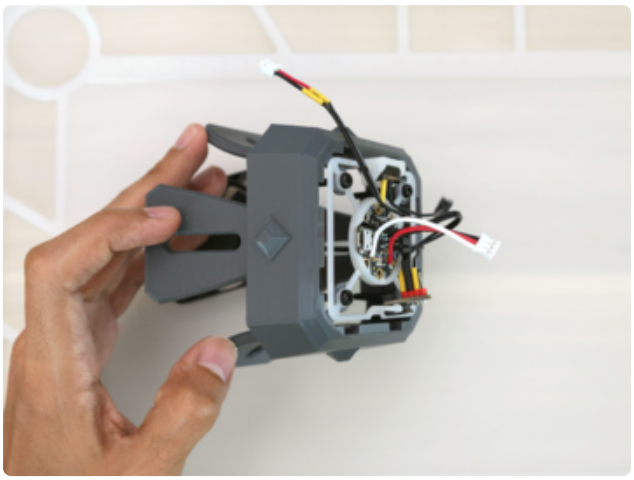


## Installing GEMMA and Fan

The fan and Gemma are secured to the tabs built-in to the top cage.

## Install GEMMA Mount

Flip the top cage so the tabs are facing up. Place the Gemma mount over the bottom and line up the four mounting holes.



## Install Hardware for GEMMA Mount

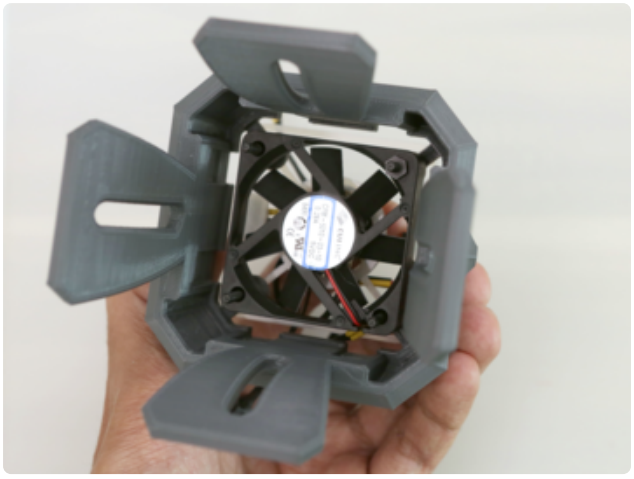
Insert 4x M3 x 30mm long screws through the four tabs. These screws are intentionally long in order to fit through the fan and various mounts.

## Secured GEMMA Mount

Insert and fasten four M3 hex nuts onto the threads of the screws. Finger tighten the hex nuts to secure the Gemma mount to the top cage.

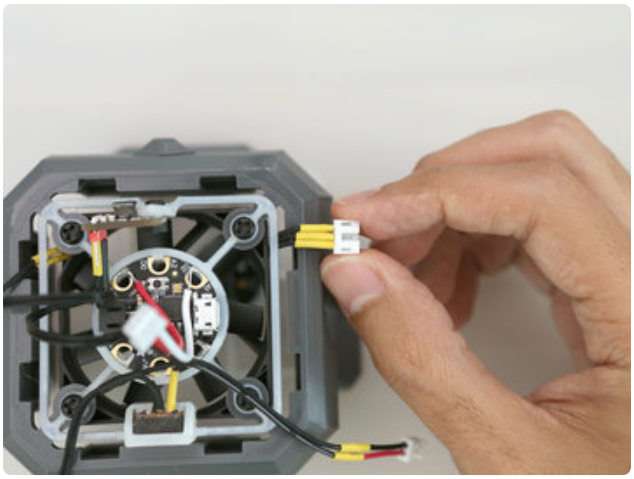
## Install 5VDC Fan

Place the fan over the top cage and fit the holes through the screws. Press the fan down into the cage so it rests on the hex nuts.



## Secure 5VDC Fan

Insert four more M3 hex nuts onto the threads of the screws to secure the fan to the top cage.

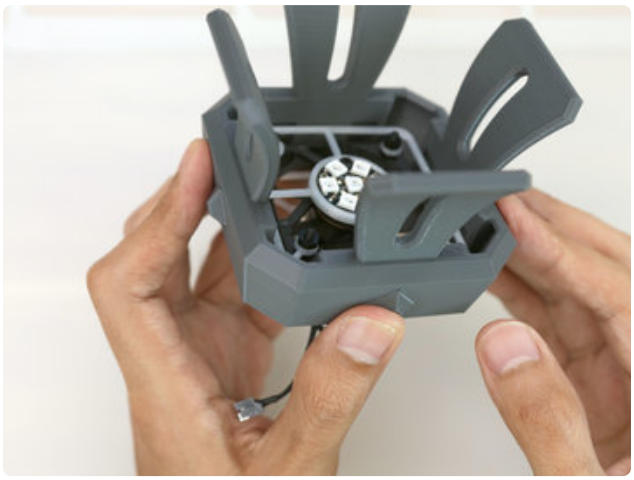


## Install JST for NeoPixel Jewel

Insert the JST connector from the NeoPixel Jewel through the top of the cage and through the opening on the bottom. Reference the photo for correct placement.

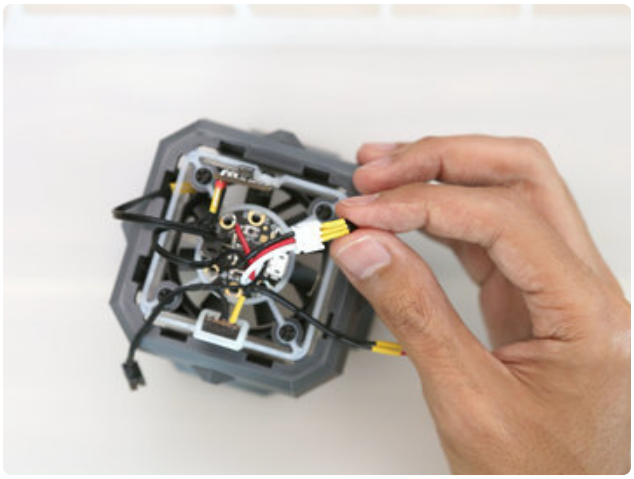
## Install NeoPixel Jewel Mount

Place the NeoPixel Jewel mount into the top cage with the tabs fitting through the screws.



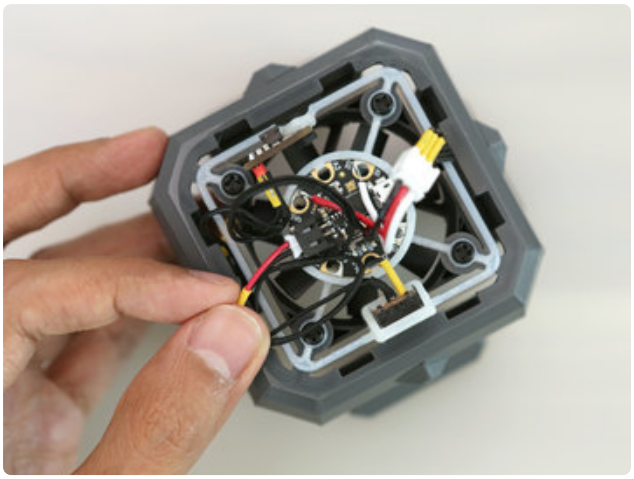
## Secure NeoPixel Jewel Mount

Install and insert four additional M3 hex nuts onto the threads on top of the NeoPixel Jewel mount.



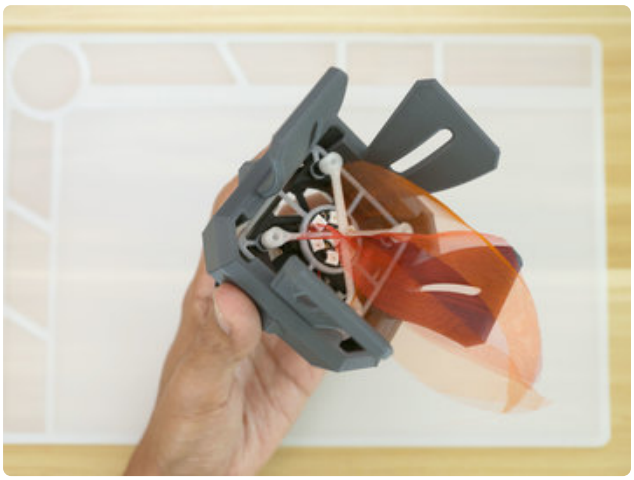
## Connect NeoPixel Jewel to GEMMA

Plug in the JST cable from the NeoPixel Jewel to the Gemma.



## Connect Switch to GEMMA

Ensure the 2-pin JST cable from the slide switch is plugged into the Gemma M0.

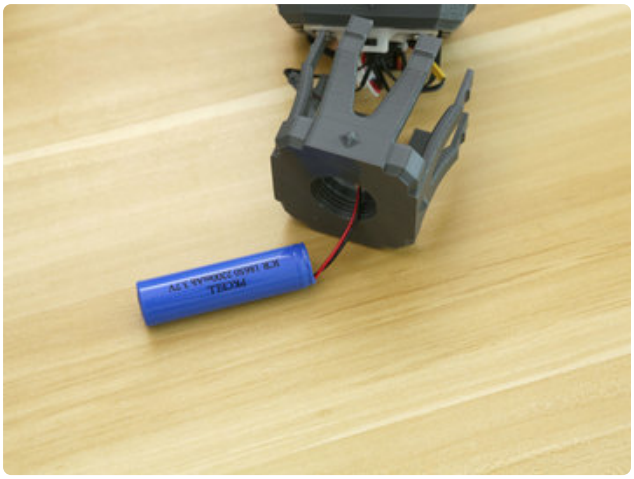


## Install Crossbar

Place the crossbar over the tips of the screws. Press down to fully set the standoffs into the threads of the screws.

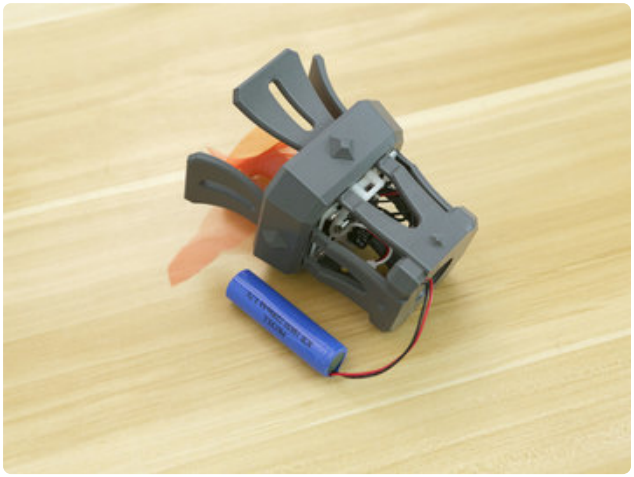
## Flame Test

Test out the placement of the crossbar and silk fabric. Connect the 2200mAh battery to the JST connector on the slide switch. Use the slide switch to power on the circuit.



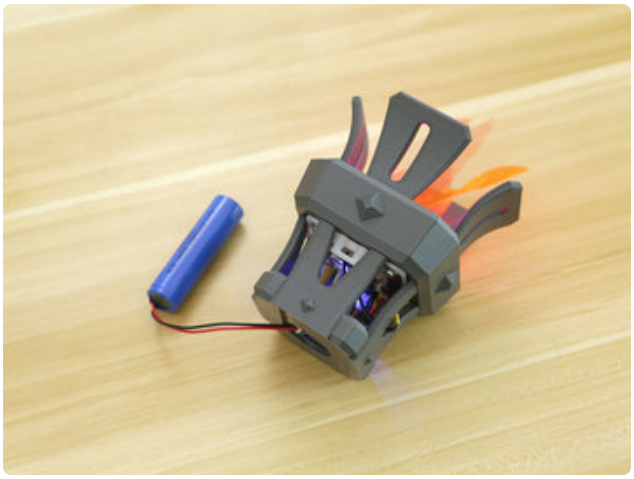
## Install Battery to Cage Bottom

Insert the 2200mAh battery through the cage bottom. Reference the photo for correct placement.



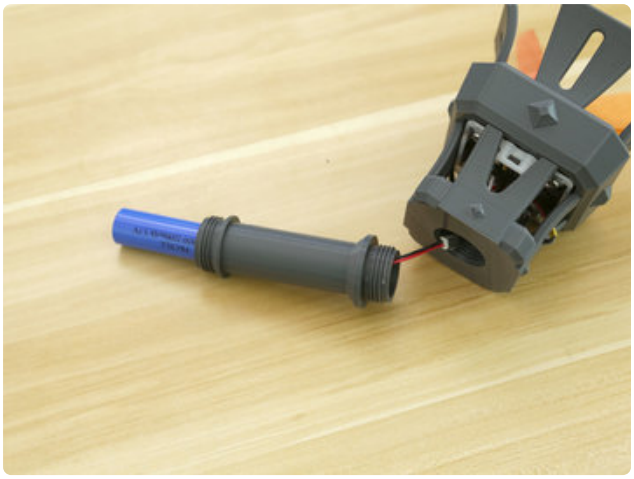
## Installed Cage Bottom

Double check all of the wires are fitted inside the cage before snap fitting together.



## Assembled Cage

Fully install the cage halves by firmly pressing them together. Test the circuit again to ensure everything is still working properly.



## Install Battery through Collar

Fit the 2200mAh battery through the collar. Begin to install the collar by screwing in into the bottom of the cage.

Threaded parts are righty tighty, lefty, loosey.

## Install Battery to Handle

Fit the 2200mAh battery through the handle. Begin to install the handle to the collar by screwing it in.

## Install Pommel to Handle

The pommel is screwed into the bottom of the handle.

## Final Build

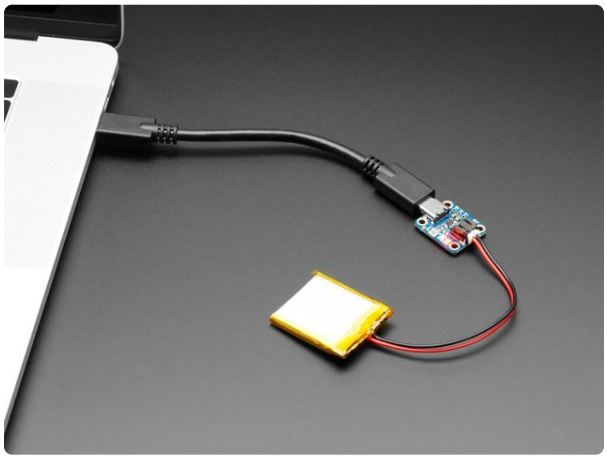
And there you have it! Be sure to tighten all of the parts before handling. Use the slide switch to power the circuit on and off.

## Fueling the flame

Getting the flame to move in a realistic way can be a bit tricky. We found the effect looks the best when it's in low lighting, while in motion or interacting with an additional air stream.

## Recharge Battery

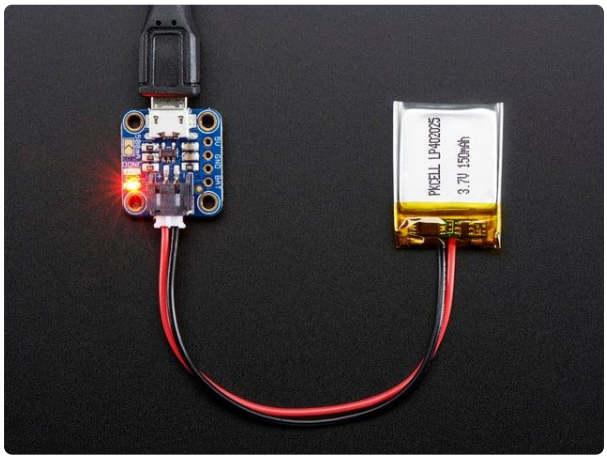
Remove the battery from the handle and disconnect from slide switch. Use a lipoly USB recharging breakout to recharge the 3.7v 2200mAh battery.



Adafruit Micro-Lipo Charger for LiPoly Batt with USB Type C Jack

Oh so handy, this little lipo charger is so small and easy to use you can keep it on your desk or mount it easily into any project! Simply plug it via any USB C cable into a USB port...

https://www.adafruit.com/product/4410




Adafruit Micro-Lipo Charger for LiPo/LiIon Batt w/MicroUSB Jack

Oh so handy, this little lipo charger is so small and easy to use you can keep it on your desk or mount it easily into any project! Simply plug it via any MicroUSB cable into a USB...

https://www.adafruit.com/product/1904