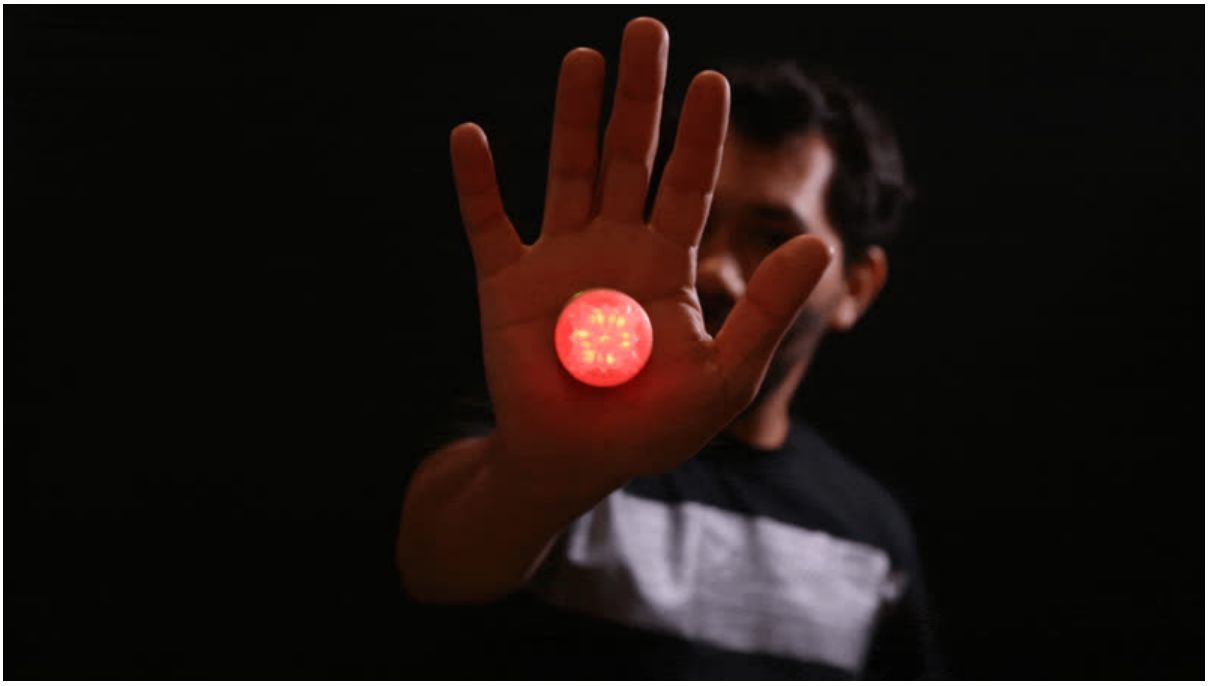
Logan’s Run Hand Jewel LED

Created by Ruiz Brothers



https://learn.adafruit.com/led-in-you-hand-logans-run-life-clock

Last updated on 2024-06-03 02:05:54 PM EDT

Table of Contents

Overview

Glowing Element

In this guide, you'll learn how to build a bright glowing light that can be used for cosplay elements, decor and wearables. The circuit and components are fully contained in a 3D printed cylindrical enclosure. An integrated switch allows you to turn the circuit off without having to take it apart. The circuit is powered by an Adafruit GEMMA M0 or v2 and a small 150mAh lithium polymer battery. The NeoPixel Jewel contains 7 individual LEDs that can be programmed to display different colors and animations. With all of the components fitted inside the enclosure, this makes for a very tight package that can be used in all sorts of projects.

Inspired by Logan's Run

This project was inspired by the "Life Clock" in 1976's epic Sci-Fi film, Logan's Run. In the story, each person is implanted at birth with a "life-clock" crystal in the palm of their hand that changes color as they get older and begins blinking as they approach their "Last Day". This DIY version is different than the original prop seen in the film, and not at all screen accurare, but it does resemble the iconic object.

Project Ideas

I can see this being used in cosplay, imagine energy beams and fire balls. This could be used as a lighting element for photography. The NeoPixel Jewel can output some pretty bright light, so this can give a great source of illumination. The circuit and enclosure are moderately small so this allows you to put it in different things. It

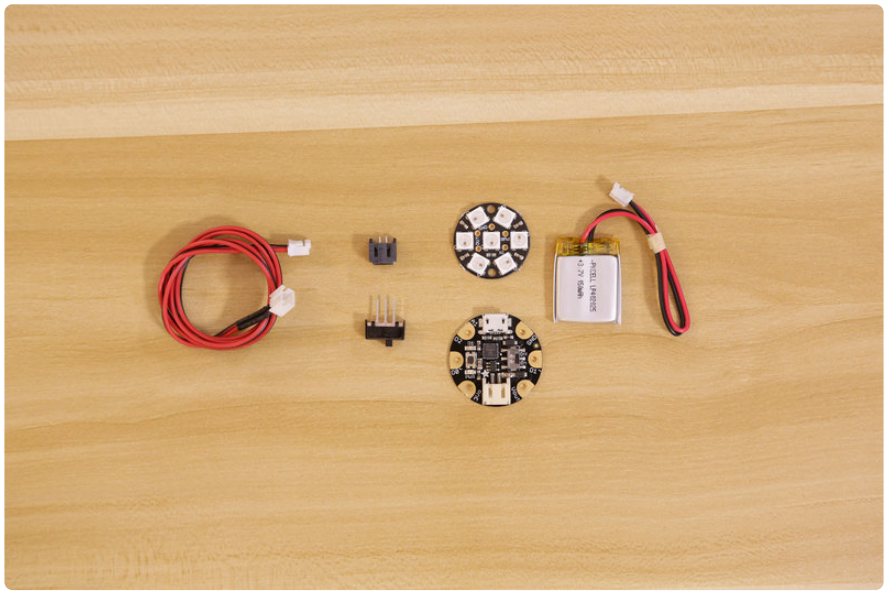
doesn't have to be stuck on your palm. Get creative and remix this project to make it your own.

This guide was written for the Gemma v2 board, but can be done with either the v2 or Gemma M0. We recommend the Gemma M0 as it is easier to use and is more compatible with modern computers!

Prerequisite Guides

This guide assumes you have basic soldering skills and know how to upload code an arduino. For beginners hobbyists looking to get started with the basics, we recommend walking through the following tutorials and get familiar with the components.

- Collin's Lab: Soldering (https://adafru.it/rBf) – How To Solder
- Gemma M0 guide (https://adafru.it/zxE)
- Classic Introducing GEMMA guide (https://adafru.it/e1V) (https://adafru.it/dgH) – How To Setup GEMMA
- Slide Switch JST Adapter (https://adafru.it/vej) – How To DIY Switches



Parts

You'll need the following parts to build this project.

- Gemma M0 (http://adafru.it/3501) or GEMMA v2 wearable microcontroller (http://adafru.it/1222)

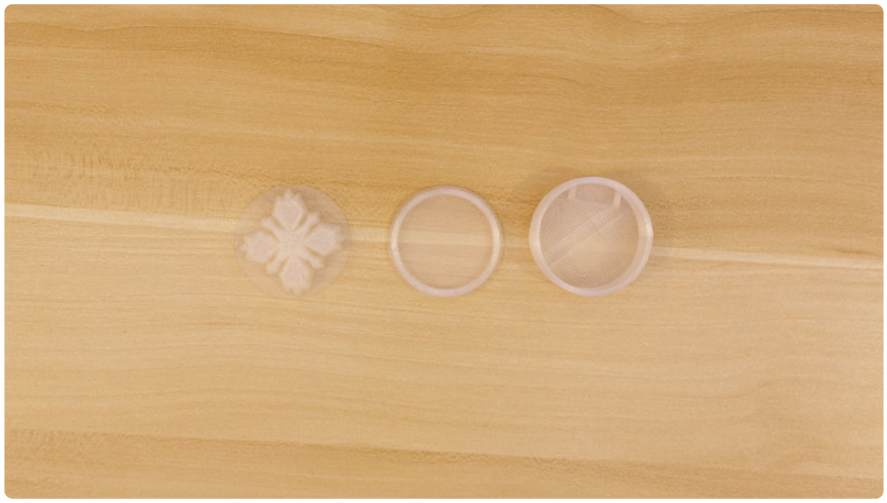
- Adafruit 7-NeoPixel Jewel (http://adafru.it/3047)
- 150mAh 3.7v Lithium Ion Polymer Battery (http://adafru.it/1317)
- JST-PH 2-pin SMT Right Angle Connector (http://adafru.it/1769)
- JST-PH Battery Extension Cable (http://adafru.it/1131)
- Slide Switch (http://adafru.it/805)

Tools & Supplies

You'll also need the following tools and supplies to complete this project.

- Soldering Iron & Solder (https://adafru.it/drl)
- Tweezers Rhino SW-11 (http://adafru.it/3096)
- Wire Strippers (http://adafru.it/527)
- Diagonal Flush Snips (http://adafru.it/152)
- Helping Third Hands (http://adafru.it/291)
- Flat Pliers (http://adafru.it/1368)

3D Printing



Download and 3D Print

You can 3D print the parts using Translucent filament on almost any desktop 3D printer. The 3D printed parts can be downloaded with the link below. If you don't have a 3D printer, the files are free to download so can send them to a 3D printing service.

The design is modeled in Autodesk Fusion 360 and designed to print in PLA filament. The parts were 3D printed using the BCN3D Sigma and Ultimaker 3.

Download files from Thingiverse

https://adafru.it/vjD

Download files from Youmagine

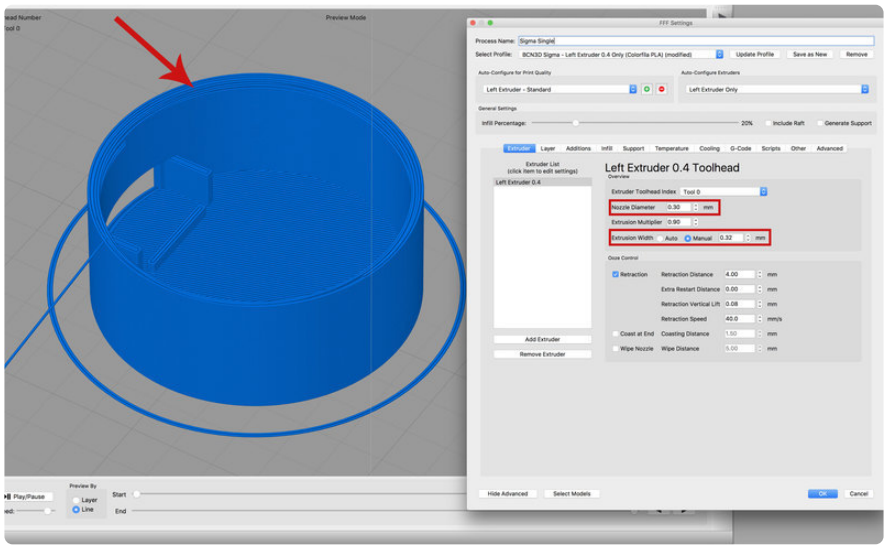
https://adafru.it/vjE

Edit Design

https://adafru.it/19DP

Edit Design

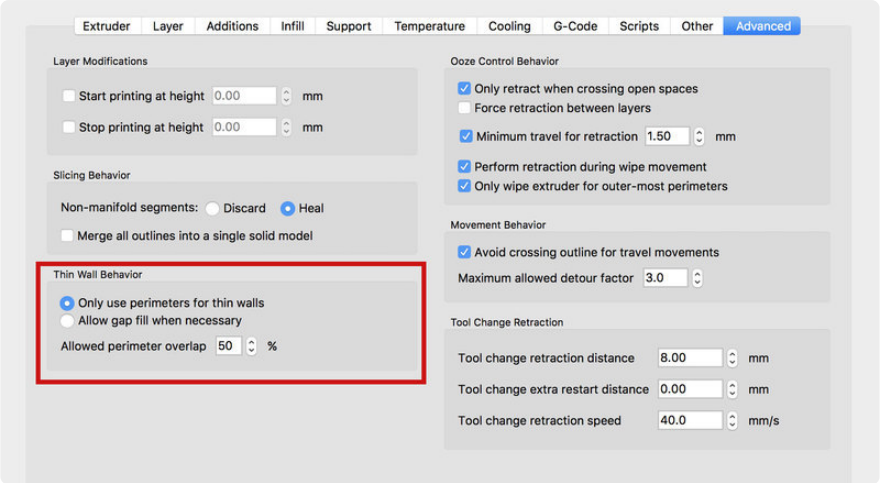
You can easily update the size or add features by editing the Fusion360 designs. The sketches are all listed in the timeline, so it's easy to adjust the size to each component.



Slice Settings

Depending on your 3D printer, you may need to adjust the slice settings. We tested the enclosure on a BCN3D Sigma. The parts are oriented to print "as is".

- Nozzle: 0.3mm
- Extrusion Multiplier: 1.0
- Extrusion Width: 0.32mm
- Layer Height: 0.2mm
- Nozzle Temperature: 225c
- Print Speed: 60mm/s



Wall Strength

We'll want the walls of the parts to be really strong by making sure gaps between them are solid (no zig zags inside the gaps). We used a .4mm nozzle but had to adjust the nozzle diameter to **.3mm** with an extrusion width of **.32mm** to make sure no "zig zag" paths are inside the wall gaps. To enable parimeters between walls you'll need to set the overlap to **50%**.

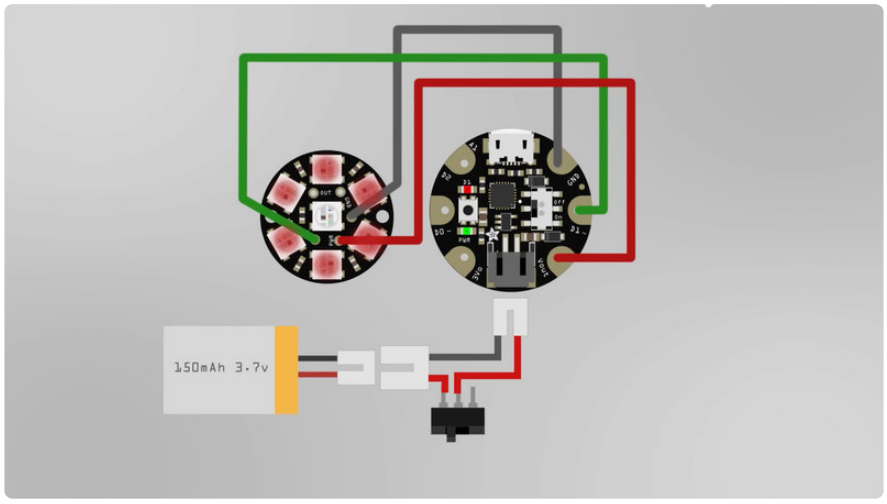
Make sure to always preview tools paths by stepping through each layer/line before printing to ensure proper tool paths are generated.

Glue Gem Top

The **jewel-top-thread.stl** is printed separately to avoid using supports on the threads. We can attach the **jewel-top-thread** part to the **jewel-top.stl** by using a couple drops of super glue.

Make sure to align the two parts so the edges are flush.

Circuit Diagram



This diagram uses the original Gemma but you can also use the Gemma M0 with the exact same wiring!

Wired Connections

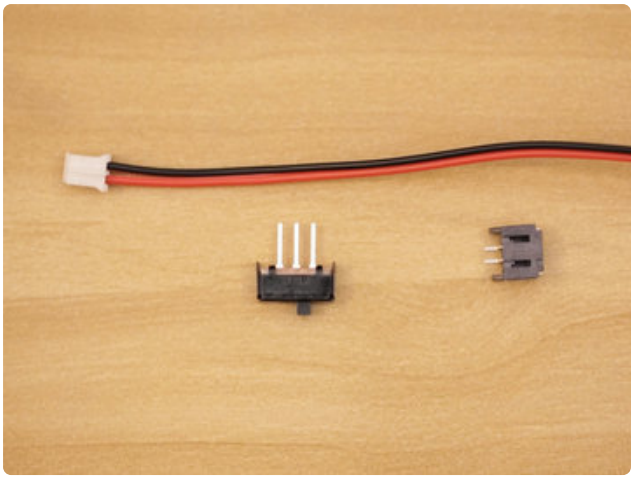
The circuit diagram above shows how the components will be wired together. This won't be 100% exact in the actual circuit but it's a very close approximation.

- GND on the Trinket to GND on the Jewel
- D1 on the Trinket to DIN on the Jewel
- Vout on the Trinket to PWR on the Jewel

Battery Power

The circuit will be powered by a 3.7V 150mAh Lithium ion battery via JST connection. The battery plugs into the JST slide switch adapter.

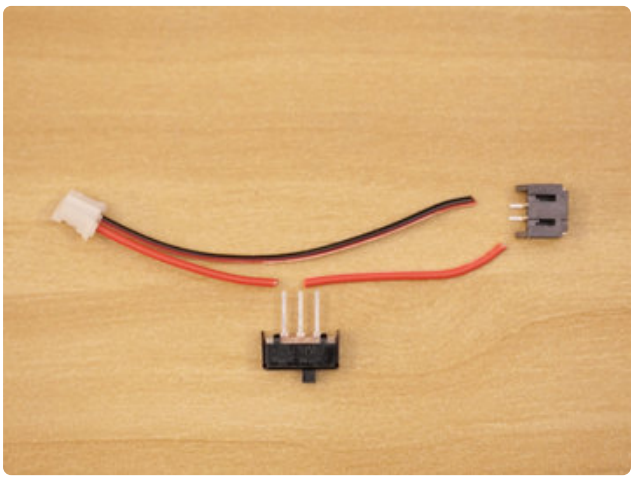
Assemble



On/Off Switch

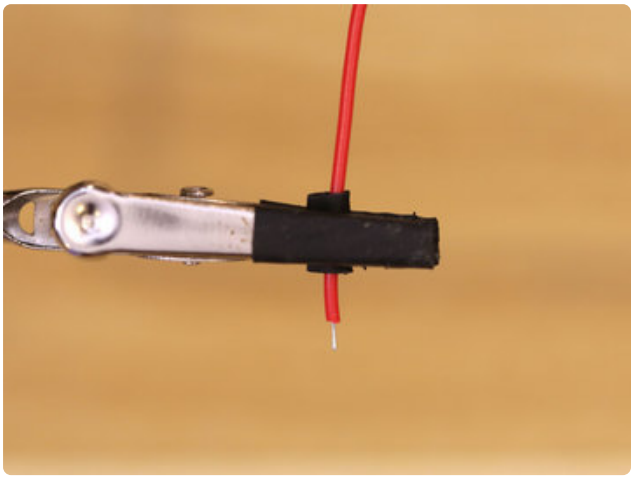
To start this project, we'll work on making the on/off switch first. Gather up the JST extension cable, JST right-angle female connector and the slide switch. We'll use this parts to build a on/off switch. This will allow us to easily cut the power from the circuit. Be sure to check out our DIY On/Off JST Slide Adapter (https://adafru.it/vjb) guide to get a full step-by-step tutorial. This guide summarizes it since it's only a portion of the build.

Measure & Cut JST Extension Cable

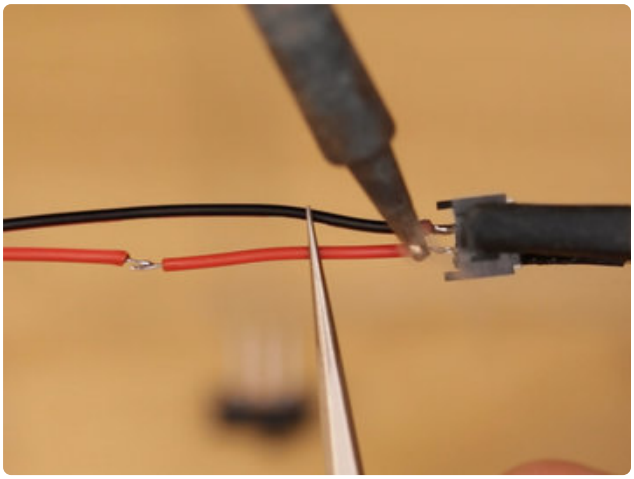


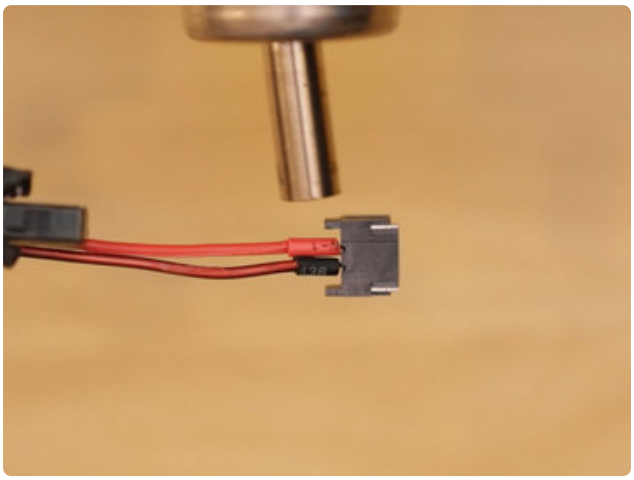
Arrange the JST extension cable, slide switch and female JST connector on your work surface. We'll need to cut the JST extension cable so the wires are a certain length. Grab hold of the end with the male JST connector and measure the cable so it's 62mm in length. Then, cut the red and black wires. We'll only need the end with the male JST connector, so we can store the other end for another project. Next, separate the red and black wires from the JST cable by pulling them apart. Then, cut the red wire in half so we have a separate connection. Now we can layout our components like shown the photo. We'll need to connect the two red wires to the slide switch, and the black and red wires to the female JST connector.

Prep Wires for Switch



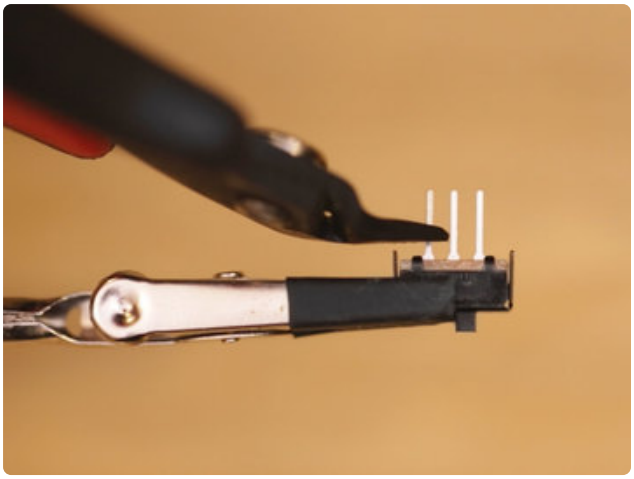
We'll need to strip and tin the tips of each wire in order to more easily attach them to the other components. Use wire strippers to remove a bit of insulation from each wire. Then, tin the exposed strands of wire by applying a small amount of solder to them. This will also prevent the strands of wire from fraying. You'll need to do this for each wire (thats a total of four tips). I suggest using helping third hands to assist you while soldering. The helping third hands can keep wires sturdy while you tin the wires.

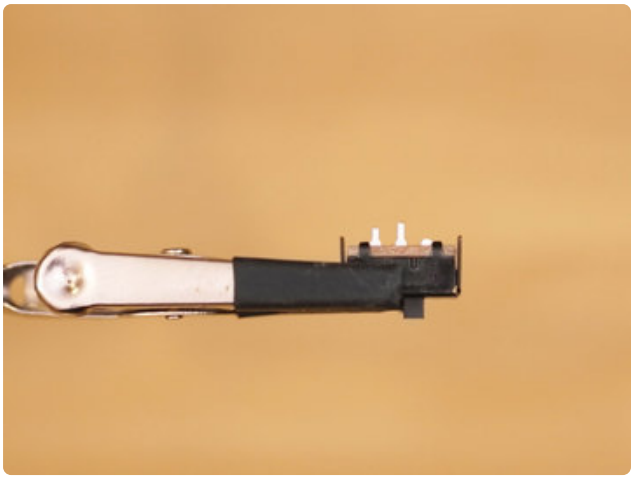




Attach Wires to JST Connector

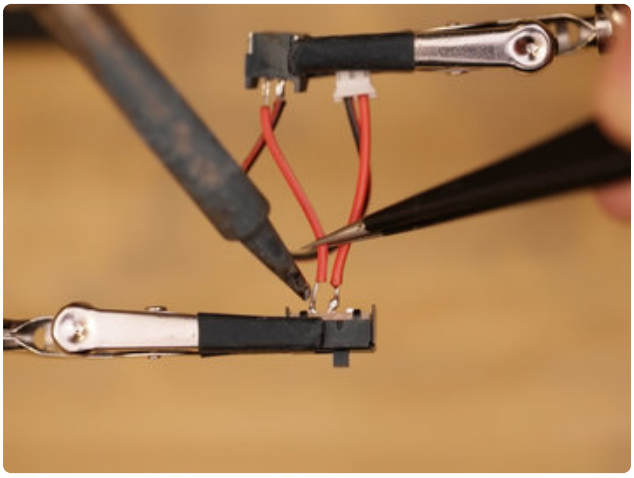
Now that we have our wires prepped, we can work on connecting wires to the female JST right-angle connector. Start by secureing the female JST connector to the helping third hands. Then, tin the two electrodes by applying a small bit of solder. Before you solder the wires to the slide switch, I suggest using some pieces of heat shrink tubing to insulate the exposed electrodes. Now we can solder the red wire (the one we cut from the male JST cable) to the positive electrode on the JST connector. Note: The positive electrode has a visible notch on the top surface of the female JST connector. Then, solder the black (negative/ground) wire from the male JST cable to the negative (ground) electrode on the female JST connector.





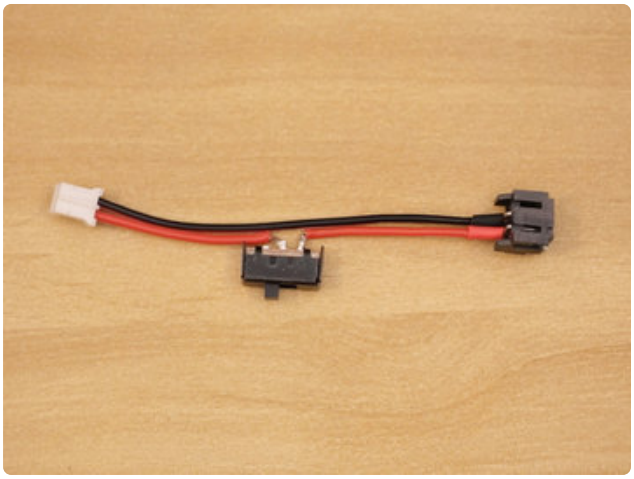
Prep Slide Switch

We'll need to prep the slide switch so that we can attach wires to the terminals. Start by removing one of the terminals from the switch using wire cutters – Either the far left or right, but NOT the middle! Then, trim the remaining two terminals short. Now we can tin them by applying a small amount of solder to them. This will make it easier to attach wires to the electrodes.



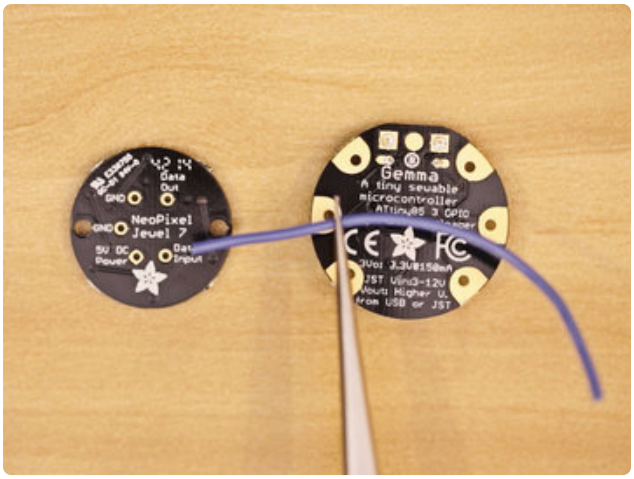
Connect Wires to Slide Switch

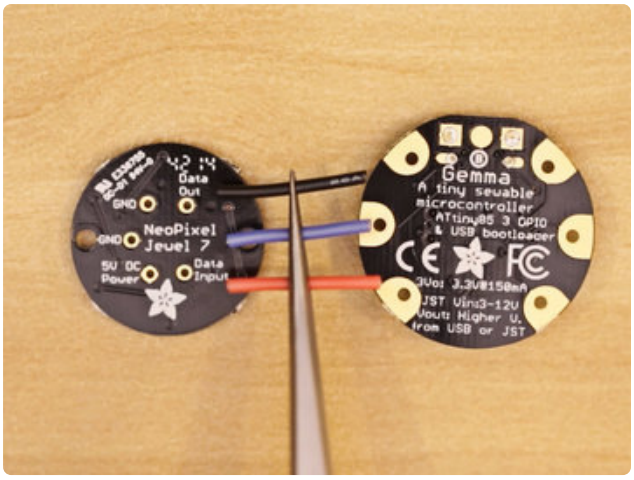
With our slide switch prepped, we can now connect wires to the electrodes on the slide switch. We'll need to solder the positive connections from the male and female JST connectors to the two electrodes on the slide switch – These red wires are the positive(red) wires from the male & female JST connectors. Connect them to the slide switch by soldering the tips of the wires to the electrodes.

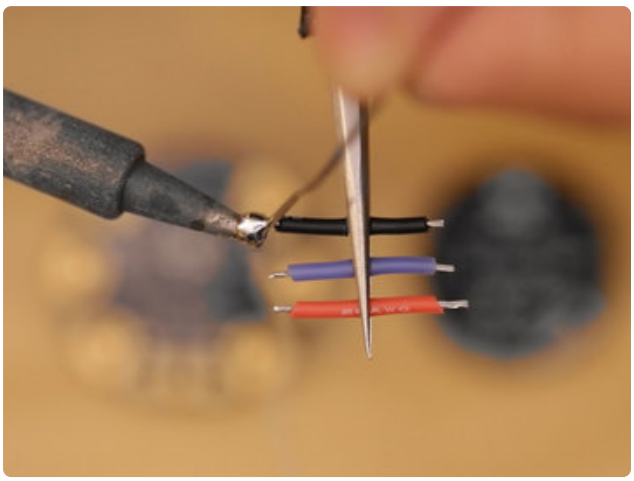


Slide Switch JST Adapter

And now we have a slide switch JST adapter that we can use to power on/off our circuit. We'll connect our battery to the female JST connector. The male JST connector from the slide switch will connect to the female JST connector on-board the Adafruit GEMMA. For now, we can set it a side and work on connecting the NeoPixel Jewel to the Adafruit GEMMA.

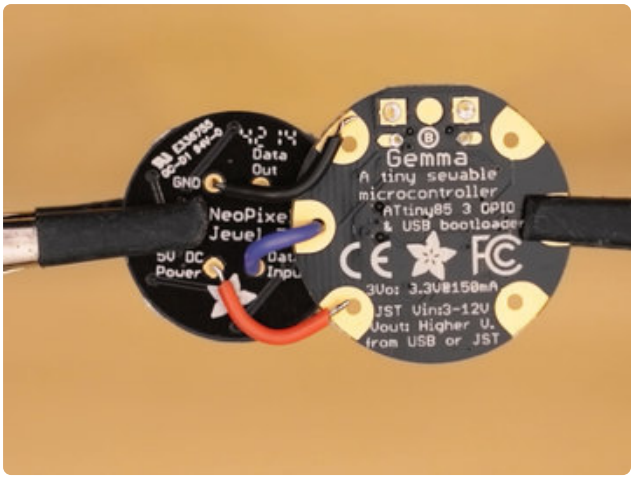


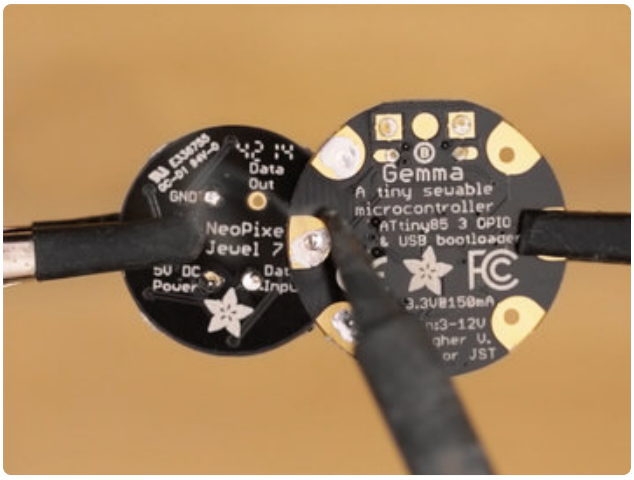


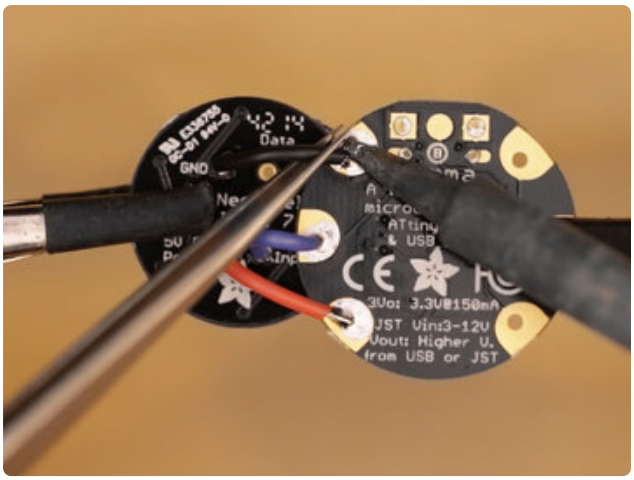


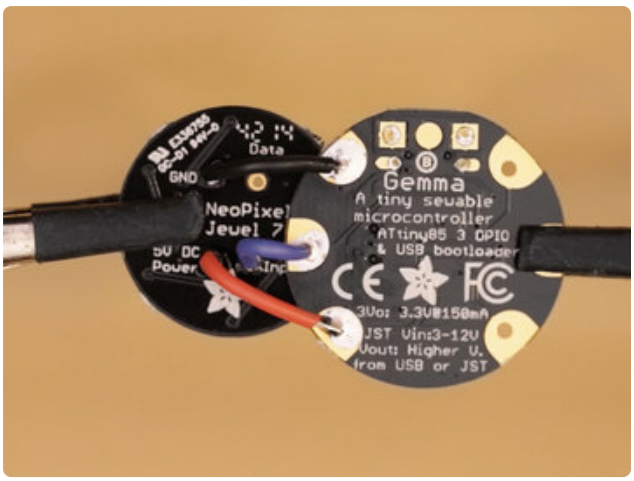
Prep Wires for NeoPixel Jewel

Now we can work on connecting the NeoPixel Jewel to the Adafruit GEMMA. We'll need three pieces of wire to connect these two components together. First, we'll need to measure out how long the three wires need to be – We found **24mm long** wires to be suffice for this project. Use wire cutters to cut three pieces of wire, 24mm in length. We used 25AWG silicone coated wires (https://adafru.it/vjc) because they're great for flexible connections. You can use different colored wires to help tell the connections apart, but it's not a requirement. Then, use wire strippers to remove a bit of insulation from the tips of each wire. After that, we can tin them by add a small amount of solder to the tips of each wire.









Connect NeoPixel Jewel to GEMMA

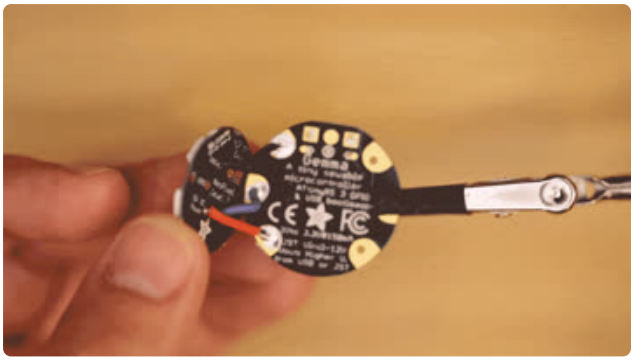
Insert Wires Into GPIO

Secure the two components to third helping hands with the back of the PCB facing up and bring the two arms close to each other like shown in the photo. This will make it easy to solder wires to each board. Insert the tips of each wire into the corresponding pins. This will make it easier to solder the wire in place.

GND from NeoPixel to **GND** on GEMMA
Data In from NeoPixel to **D1** on GEMMA
5V from NeoPixel to **VOUT** on GEMMA

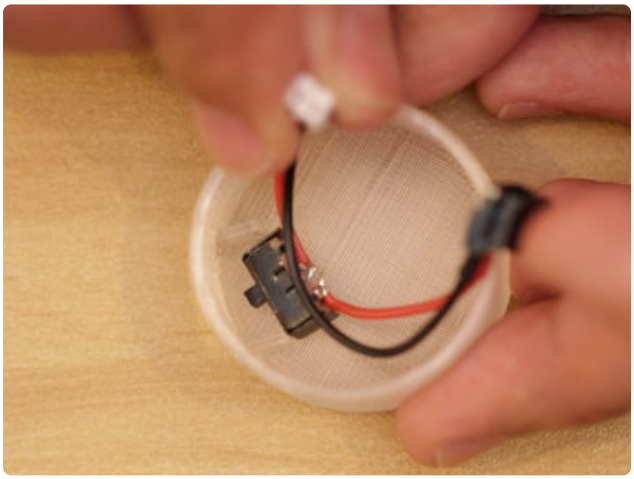
Solder Wires to Gemma and Jewel

Apply a small amount of solder to each pad on the Gemma. Try not to apply too much solder the large pads on the Gemma. Tin the pads on the NeoPixel Jewel and then solder the wires for power, data and ground. And now we have the NeoPixel Jewel connected to the Adafruit GEMMA, yay!



Circuit Sandwich

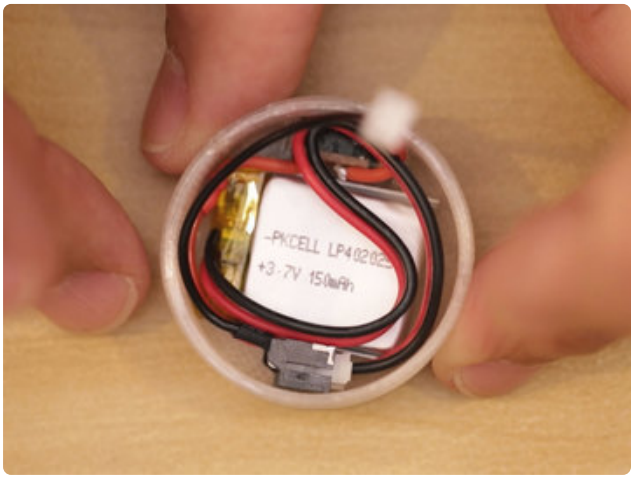
Fold the NeoPixel Jewel over the Gemma so the flat sides of the boards are back to back. The LEDs should be facing outward like in the picture. If the wires are too long or stick out, we will need to shorten the wires to make it compacted inside the enclosure. Now we should be able to lightly compress the two boards together. Make sure the wires are small enough to stay hidden between the two boards.



Install Slide Switch

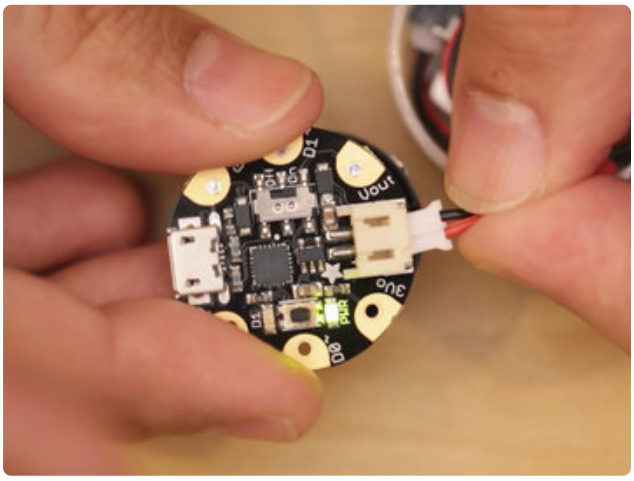
Next, we'll work on fitting the slide switch into the 3D printed enclosure. Grab the two parts and start working the slide switch into the enclosure. The body of the slide switch will be housed into three walls located inside the enclosure. Press the body down in between the walls until its flush with the surface. The actuator from the slide will protrode from the side of the enclosure.

Install Battery



Now we can connect the battery to the slide switch. Connect the male JST connector from the battery to the female JST connector thats attached to the slide switch. Next, carefully place the battery into the enclosure. You'll need to fit the cable from the battery into the enclosure, so we need to arrange the wiring so it's neatly packaged into the case. We found it best if you coil the wiring and tuck it inbetween the walls of the enclosure so it stays in place. The male JST connector from the slide switch should be accessible. Next, we'll connect the Adafruit GEMMA to the slide switch.

Connect Slide to Gemma



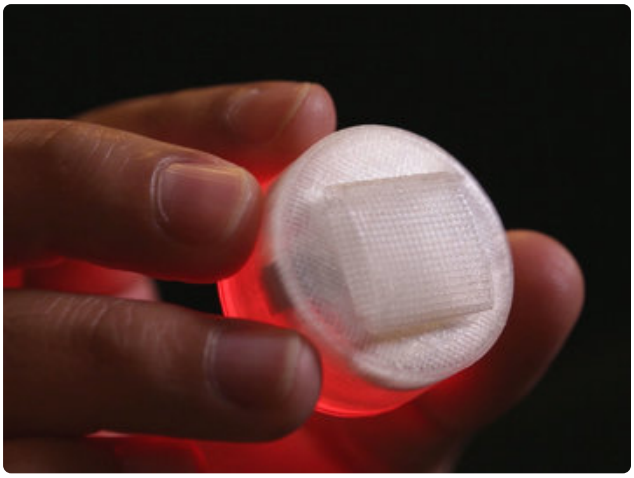
Now we can connect the male JST connector attached to the switch to the female JST connector on-board the Adaruit GEMMA by plugging the male JST connector to the female JST connector. Make sure the switch on-board the Adafruit GEMMA is set to the ON position. Next, we'll work on install the NeoPixel Jewel and GEMMA assembly into the 3D printed enclosure.

Install NeoPixel and GEMMA

Now we'll need to insert the NeoPixel and GEMMA into the 3D printed case. Lay the assembly on top of the battery with the NeoPixel LEDs facing up. You'll need keep all of the wiring contained within the enclosure. You may need to press the component together and hold them in place to install the cover. Next, we'll screw the top cover onto the enclosure.

Twist top

Grab the 3D printed top piece and place it over the enclosure. The cover and enclosure feature a custom thread that coils around edges of the parts. With the cover ontop of the case, slowly press them together and turn the cover clockwise until the threading meet and start screwing together. Keep turning until the two pieces are fully secured together. And thats it! Now we have our self-contained assembly, ready to wear, place or install onto our project. Sweeet!

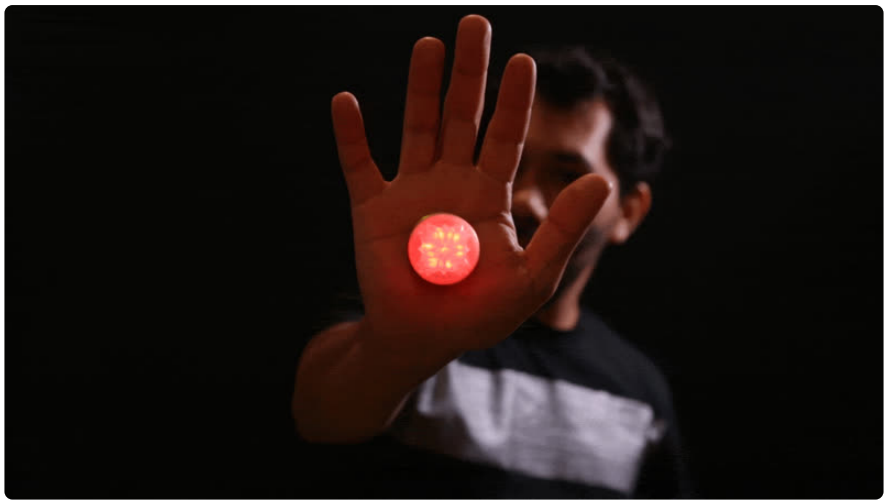


Usage

Attaching to Human Skin

We used3M Nextcare Transpore Flexible Clear Tape (a medical grade tape) to attach the enclosure to the palm of our hands. **Tegaderm Tape** is similar and specifically designed to adhere to human skin.

If you plan to attach the enclosure to another surface, you'll need to pick the right adhesive for the job – This depends on the materials. Cyanoacrylates (super glue) or E6000 work well for most applications. Just make sure you still have access to the power switch when securing the enclosure to another surface.



Arduino Code

The Arduino code presented below works equally well on all versions of GEMMA: v2 and M0. But if you have an M0 board, consider using the CircuitPython code on the next page of this guide, no Arduino IDE required!

Getting Code Onto GEMMA

In this portion of the guide, we'll get code uploaded to the **Adafruit GEMMA** micro-controller. If you don't write / understand code, don't to worry! You don't need to be a programmer to be able to upload prewritten code :-)

We'll walk you through the whole process.

First, visit the Adafruit GEMMA tutorial page by clicking the button below. Follow the instructions to download & setup the Arduino IDE and install drivers.

Getting Started with GEMMA

https://adafru.it/dgH

Install Adafruit NeoPixel Library

Next, we need to add support for NeoPixels.

Visit the Adafruit NeoPixel tutorial to install the NeoPixel library!

Install NeoPixel Library

Uploading Code to Board

Now that we have the Adafruit boards & NeoPixel library installed, we can get our code ready to upload onto the board. Select all of the code listed below in the black box and copy it to your clip board. Then, in Arduino IDE, paste it in the sketch window (making sure to overwrite anything currently there). Next, goto the **Tools** menu > **Board** and select **Adafruit GEMMA 8MHz**. Now you can click on the "**check mark**" icon to verify the code. If it's all good, we can continue to upload the code to the board.

Connect USB Data Cable to GEMMA

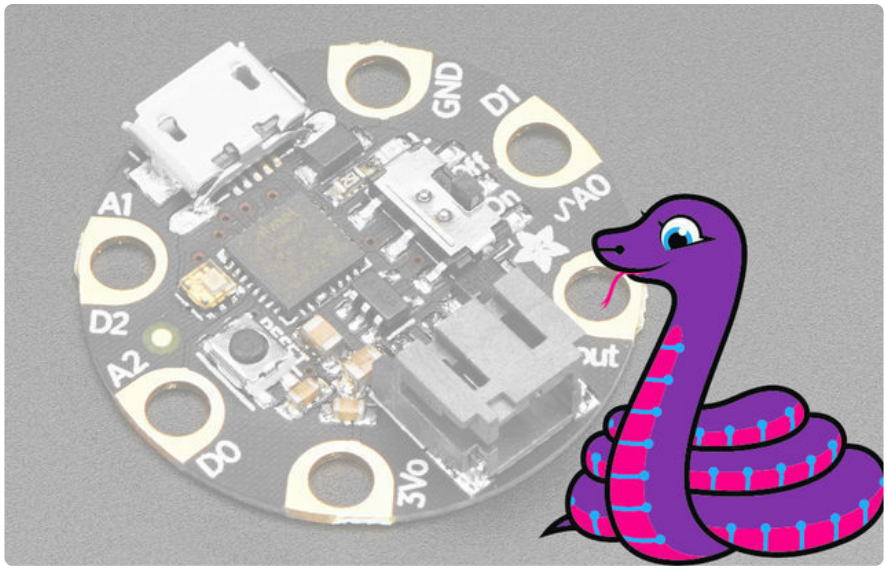
Be sure to use a micro USB cable that can transfer data - A USB cable that ONLY charges devices will simply not work. Plug it into the microUSB port on the Adafruit GEMMA board and the USB port on your computer (try to avoid connecting to a USB hub). As soon as you plug it in, you'll see a red LED blink on the Adaruit GEMMA - This let's you know the board is ready to except code. While the LED is blinking, click on the Upload button (It's a right arrow icon, next to the chekc mark). The Arduino IDE will notify you if the upload is successful and completed.

```
// SPDX-FileCopyrightText: 2017 Mikey Sklar for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#ifdef __AVR__
  #include <avr/power.h>
#endif

#include <Adafruit_NeoPixel.h>

#define PIXEL_PIN    1  // Pin connected to neo pixels
#define PIXEL_COUNT  7  // Count of neo pixels

Adafruit_NeoPixel strip = Adafruit_NeoPixel(PIXEL_COUNT, PIXEL_PIN, NEO_GRB +
NEO_KHZ800);

void setup() {

  strip.begin();
  strip.show();

}

// Linear interpolation of y value given min/max x, min/max y, and x position.
float lerp(float x, float x0, float x1, float y0, float y1)
{
  // Clamp x within x0 and x1 bounds.
  x = x > x1 ? x1 : x;
  x = x < x0 ? x0 : x;
  // Calculate linear interpolation of y value.
  return y0 + (y1-y0)*((x-x0)/(x1-x0));
```

```
}

// Set all pixels to the specified color.
void fill_pixels(Adafruit_NeoPixel& pixels, uint32_t color)
{
  for (int i=0; i < pixels.numPixels(); ++i) {
    pixels.setPixelColor(i, color);
  }
  strip.show();
}

// Get the color of a pixel within a smooth gradient of two colors.
uint32_t color_gradient(uint8_t start_r, // Starting R,G,B color
                        uint8_t start_g,
                        uint8_t start_b,
                        uint8_t end_r,   // Ending R,G,B color
                        uint8_t end_g,
                        uint8_t end_b,
                        float pos)       // Position along gradient, should be a
value 0 to 1.0
{
  // Interpolate R,G,B values and return them as a color.
  uint8_t red   = (uint8_t) lerp(pos, 0.0, 1.0, start_r, end_r);
  uint8_t green = (uint8_t) lerp(pos, 0.0, 1.0, start_g, end_g);
  uint8_t blue  = (uint8_t) lerp(pos, 0.0, 1.0, start_b, end_b);
  return Adafruit_NeoPixel::Color(red, green, blue);
}

void animate_gradient_fill(Adafruit_NeoPixel& pixels, // NeoPixel strip/loop/etc.
                           uint8_t start_r,           // Starting R,G,B color
                           uint8_t start_g,
                           uint8_t start_b,
                           uint8_t end_r,             // Ending R,G,B color
                           uint8_t end_g,
                           uint8_t end_b,
                           int duration_ms)           // Total duration of
animation, in milliseconds
{
  unsigned long start = millis();
  // Display start color.
  fill_pixels(pixels, Adafruit_NeoPixel::Color(start_r, start_g, start_b));
  // Main animation loop.
  unsigned long delta = millis() - start;
  while (delta < duration_ms) {
    // Calculate how far along we are in the duration as a position 0...1.0
    float pos = (float)delta / (float)duration_ms;
    // Get the gradient color and fill all the pixels with it.
    uint32_t color = color_gradient(start_r, start_g, start_b, end_r, end_g, end_b,
pos);
    fill_pixels(pixels, color);
    // Update delta and repeat.
    delta = millis() - start;
  }
  // Display end color.
  fill_pixels(pixels, Adafruit_NeoPixel::Color(end_r, end_g, end_b));
}
void loop() {

  // Run It:

  // Nice fade from dim red to full red for 1/2 of a second:
  animate_gradient_fill(strip,
                        10, 0, 0,
                        255, 0, 0,
                        500);
  // Then fade from full red to dim red for 1/2 a second.
  animate_gradient_fill(strip,
                        255, 0, 0,
                        10, 0, 0,
```

```
                    500);
//delay(1000); Use this delay if using multiple color fades
}
```

CircuitPython Code



GEMMA M0 boards can run **CircuitPython** — a different approach to programming compared to Arduino sketches. In fact, **CircuitPython comes factory pre-loaded on GEMMA M0**. If you've overwritten it with an Arduino sketch, or just want to learn the basics of setting up and using CircuitPython, this is explained in the **Adafruit GEMMA M0 guide** (https://adafru.it/z1B).

These directions are specific to the "M0" GEMMA board. The original GEMMA with an 8-bit AVR microcontroller doesn't run CircuitPython...for those boards, use the Arduino sketch on the "Arduino code" page of this guide.

Below is CircuitPython code that works similarly (though not exactly the same) as the Arduino sketch shown on a prior page. To use this, plug the GEMMA M0 into USB...it should show up on your computer as a small **flash drive**...then edit the file "**code.py**" with your text editor of choice. Select and copy the code below and paste it into that file, **entirely replacing its contents** (don't mix it in with lingering bits of old code). When you save the file, the code should **start running almost immediately** (if not, see notes at the bottom of this page).

If GEMMA M0 doesn't show up as a drive, follow the GEMMA M0 guide link above to prepare the board for CircuitPython.

```
# SPDX-FileCopyrightText: 2017 Mikey Sklar for Adafruit Industries
#
# SPDX-License-Identifier: MIT
```

```
import time

import board
import neopixel

pixpin = board.D1
numpix = 7
wait = .5  # 1/2 second color fade duration

# defaults to RGB|GRB Neopixels
strip = neopixel.NeoPixel(pixpin, numpix, brightness=1, auto_write=False)


# uncomment the following 3 line for RGBW Neopixels
# strip = neopixel.NeoPixel(
# pixpin, numpix, bpp=4, brightness=.3, auto_write=True
# )

# Linear interpolation of y value given min/max x, min/max y, and x position.


def lerp(x, x0, x1, y0, y1):
    # Clamp x within x0 and x1 bounds.
    if x > x1:
        x = x1

    if x < x0:
        x = x0

    # Calculate linear interpolation of y value.
    return y0 + (y1 - y0) * ((x - x0) / (x1 - x0))


# Set all pixels to the specified color.


def fill_pixels(r, g, b):
    for i in range(0, numpix):
        strip[i] = (r, g, b)
        strip.write()


# Get the color of a pixel within a smooth gradient of two colors.
# Starting R,G,B color
# Ending R,G,B color
# Position along gradient, should be a value 0 to 1.0


def color_gradient(start_r, start_g, start_b, end_r, end_g, end_b, pos):
    # Interpolate R,G,B values and return them as a color.
    red = lerp(pos, 0.0, 1.0, start_r, end_r)
    green = lerp(pos, 0.0, 1.0, start_g, end_g)
    blue = lerp(pos, 0.0, 1.0, start_b, end_b)

    return (red, green, blue)


# Starting R,G,B color
# Ending R,G,B color
# Total duration of animation, in milliseconds


def animate_gradient_fill(start_r, start_g, start_b, end_r, end_g, end_b,
                          duration_ms):
    start = time.monotonic()

    # Display start color.
    fill_pixels(start_r, start_g, start_b)
```

```
    # Main animation loop.
    delta = time.monotonic() - start

    while delta < duration_ms:
        # Calculate how far along we are in the duration as a position 0...1.0
        pos = delta / duration_ms
        # Get the gradient color and fill all the pixels with it.
        color = color_gradient(start_r, start_g, start_b,
                               end_r, end_g, end_b, pos)
        fill_pixels(int(color[0]), int(color[1]), int(color[2]))
        # Update delta and repeat.
        delta = time.monotonic() - start

    # Display end color.
    fill_pixels(end_r, end_g, end_b)


while True:
    # Run It:

    # Nice fade from dim red to full red for 1/2 of a second:
    animate_gradient_fill(10, 0, 0, 255, 0, 0, wait)

    # Then fade from full red to dim red for 1/2 a second.
    animate_gradient_fill(255, 0, 0, 10, 0, 0, wait)

    # time.sleep(1) # Use this delay if using multiple color fades
```

This code requires the **neopixel.py** library. A factory-fresh board will have this already installed. If you've just reloaded the board with CircuitPython, create the "lib" directory and then download neopixel.py from Github (https://adafru.it/yew).

Download neopixel.py from GitHub

https://adafru.it/yew