Cosplay Glow Fur Raver Bandolier

Created by Erin St Blaine

https://learn.adafruit.com/cosplay-glow-fur-raver-bandolier

Last updated on 2023-08-29 03:41:22 PM EDT

Table of Contents

Introduction

Festival season is nearly upon us. When the sun goes down, professional party-goers light it up with NeoPixels.

But what to make? A top hat? Fur leggings? An LED harness? Versatility is key when you're on the playa, or camping with your tribe with limited tent space.

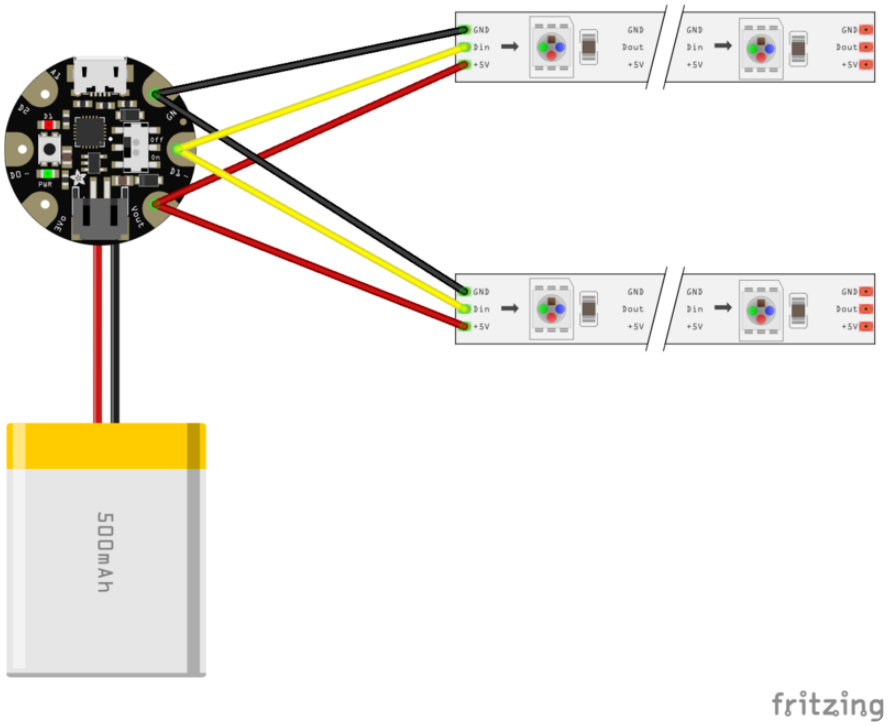
This bandolier can be worn many different ways, so you can sport a different LED costume accessory every night of the week. Make a few extras and loan them to your friends, or strap them all onto yourself and be the life of the party. Or use them to decorate your tent so you can find your way home at night.

This is a fairly easy beginner project. In addition to the parts listed to the right, you'll need:

- Soldering iron & solder
- Clear 1/2 inch Heat shrink tubing
- Hot glue gun
- Heat gun
- 2" - 3" wide Fun fur trim
- 1/4" wide sparkle trim
- Ribbons for tying
- Needle & thread
- Small piece of velcro

You can add your own code and animation modes or just copy and paste ours for delicious animated rainbow goodness.

Wiring Diagram



The Gemma will be placed in the middle of the bandolier, with one of the LED strips on either side. We're wiring both strips' data pins to the same Gemma pin (D1).

This will make both strips show the exact same animations at the same time. We'll get a symmetrical effect, with the animations running outwards in both directions from the Gemma.

Gemma Vout	NeoPixel +5V
Gemma D1	NeoPixel IN
Gemma GND	NeoPixel GND

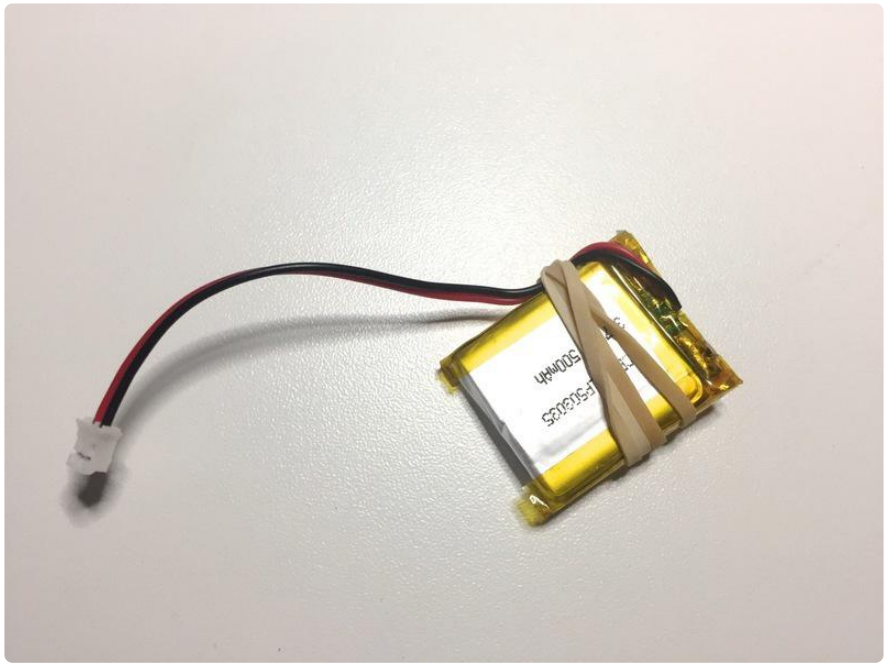
A Note about Power

This project assumes you'll have around 20-30 NeoPixels on each strip -- no more than 50-60 total in the project. With just a few pixels we can keep our wiring really simple and draw power through the battery port on the Gemma.

If you plan to drive a lot more pixels -- say, if you want to use the stunning 144/m NeoPixels () -- it's best to draw power directly from the battery instead of going through the Gemma.

Here's a tutorial () that shows how to wire up lots of pixels, connecting them directly to the power source.

Learn lots more about ways to power NeoPixels here. ()



Batteries

My favorite batteries for wearable projects are LiPoly batteries. They're small and light, and will run a moderate number of NeoPixels for hours. They come in various sizes so you can choose the right one to suit your project.

However, “with great power comes great responsibility.” If LiPoly batteries are punctured or mistreated, they could start a fire. The wires also love to come disconnected from their solder points at the top of the battery, usually as soon as you get to your cosplay event or costume party.

I always wrap a rubber band around the battery and wires to provide strain relief, and minimize the chance that the wires will pull out. A dab of E6000 glue at the wire-to-battery junction is another option.

And don't forget to get a charger ()!

If LiPoly batteries sound too troublesome, you can also power your project with AA or AAA alkaline cells. This battery case () will provide the right voltage, and has a nice on/off switch to boot. Using AAA batteries means you can always find fresh batteries if yours have run out, and they're safer to use for kids, or for extreme cosplay events.

Code

Ravers love rainbows. It's hard to beat a smooth animated bright rainbow pattern -- it goes with everything.

We've included two different ways to program your LEDs: Arduino or CircuitPython.

Which one should I choose?

There are pros and cons for each method. At the time of writing, CircuitPython is still a brand new thing, while Arduino is a bit more tried and true. Both will work to make pretty rainbows!

The Arduino () IDE currently has way more code samples and control options available. If you're excited about tweaking the code to get it just right, and building on code samples posted by other makers, you may want to stick with Arduino for now. It takes a few steps to get everything installed, but you only need to go through the process once.

CircuitPython () is exciting because it's easy to set up and easy to use. Many people already know how to code in Python, and it's just a short step from Python to CircuitPython. There are fewer steps -- not so much to install, and you don't need to compile and upload the code every time. Instead, the Gemma M0 will show up as a drive on your computer and you can just save files directly to it. But at the moment, there aren't yet a zillion open-source samples on the internet just waiting for you to download and try.

Arduino Code

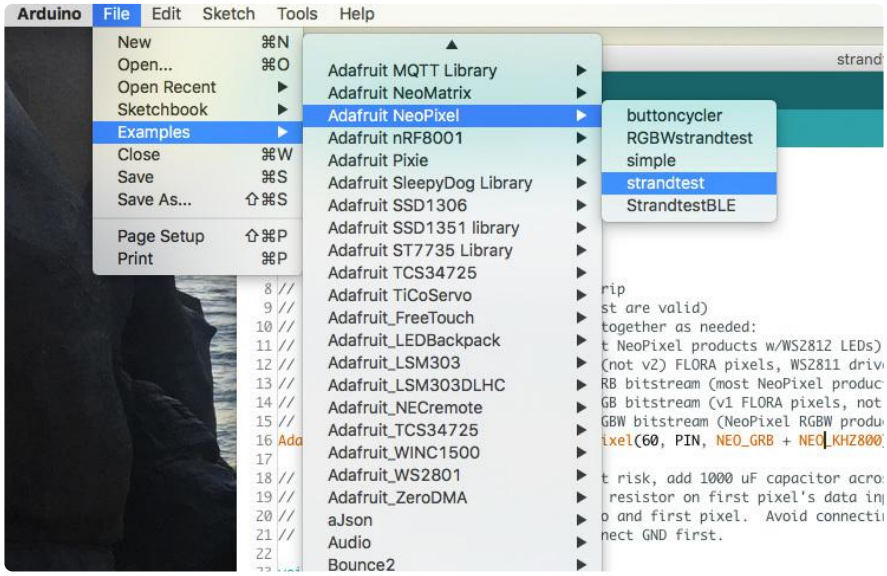
Adafruit's NeoPixel library comes with a pre-installed rainbow animation effect. You'll need to make just a few edits to the code.

If this is your first time using Arduino, you'll need to download and install the free software and the NeoPixel library.

Here's a guide that will walk you through installing Arduino and getting the Gemma board set up. ()

And..

Here's a detailed NeoPixel library installation guide. ()



Once you've got Arduino installed and the NeoPixel library installed, open Arduino and choose:

File > Examples > Adafruit NeoPixel > strandtest

This will open up a window with the code we want to modify.

First, find this line near the top of the strandtest code:

```
#define PIN 6
```

Since we soldered our LED strand to pin 1, we'll change the pin number from 6 to 1.

```
#define PIN 1
```

Next, look for this line:

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);
```

This is where we tell the Gemma how many LEDs we soldered onto pin 1. The default is 60. Change this to reflect the number of pixels you have in ONE strip.

Why count just one strip? We soldered both our strips to pin 1, so they will show the same code at the same time. So even if you have 40 NeoPixels total, we'll want to tell the microcontroller we have 20 since that's how many are in each strip.

```
Adafruit_NeoPixel strip = Adafruit_NeoPixel(20, PIN, NEO_GRB + NEO_KHZ800);
```

The strandtest code runs a series of animations, doing color wipes and chase animations along with the rainbow. For this project I want nothing but rainbows, so we'll go into the code and comment out all the other animations.

Find this section:

```
void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
//colorWipe(strip.Color(0, 0, 0, 255), 50); // White RGBW
  // Send a theater pixel chase in...
  theaterChase(strip.Color(127, 127, 127), 50); // White
  theaterChase(strip.Color(127, 0, 0), 50); // Red
  theaterChase(strip.Color(0, 0, 127), 50); // Blue

  rainbow(20);
  rainbowCycle(20);
  theaterChaseRainbow(50);
}
```

The `loop()` function is the part of the code that will run continuously on the Gemma. Right now it's set up to cycle through three `colorWipe()` animations (the fourth one is already commented out), then three `theaterChase()` animations, then we get to `rainbow()`

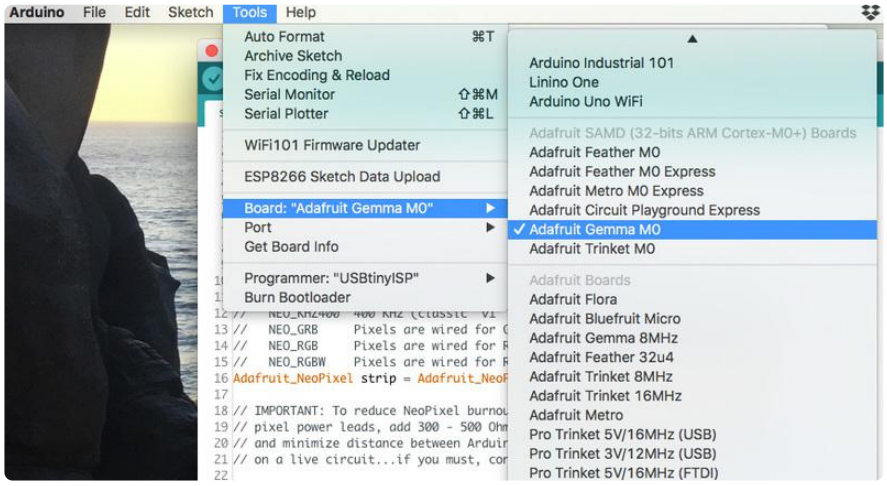
Add two slashes:

//

...to the beginning of each animation you do NOT want to show. Your code will look like this:

```
void loop() {
  // Some example procedures showing how to display to the pixels:
  //colorWipe(strip.Color(255, 0, 0), 50); // Red
  //colorWipe(strip.Color(0, 255, 0), 50); // Green
  //colorWipe(strip.Color(0, 0, 255), 50); // Blue
//colorWipe(strip.Color(0, 0, 0, 255), 50); // White RGBW
  // Send a theater pixel chase in...
  //theaterChase(strip.Color(127, 127, 127), 50); // White
  //theaterChase(strip.Color(127, 0, 0), 50); // Red
  //theaterChase(strip.Color(0, 0, 127), 50); // Blue

  rainbow(20);
  //rainbowCycle(20);
  //theaterChaseRainbow(50);
}
```

You can also mess with the speed of the rainbow animation. Try a larger or smaller number in place of the `(20)` to see what works for you.



Now, plug your Gemma into your computer via the USB port. Go to Tools > Board and select Adafruit Gemma M0.

Don't see it there? Be sure you followed all the instructions on this page. ()

Double click the reset button on the Gemma, and at the same time, click the upload button in Arduino. You'll see a second LED light up on the face of the Gemma, in red, and it will flicker a bit. That's how you know it's working!

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 ().

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 "main.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.

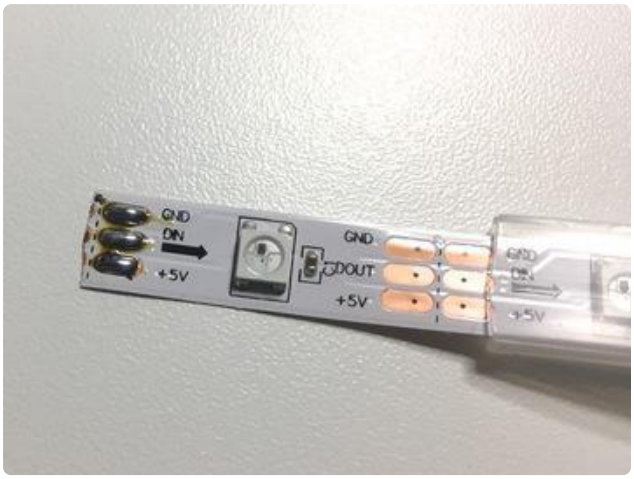
```
# SPDX-FileCopyrightText: 2017 Mikey Skylar for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
from rainbowio import colorwheel
import board
import neopixel

pixpin = board.D1
numpix = 20

pixels = neopixel.NeoPixel(pixpin, numpix, brightness=.3, auto_write=False)

rgb_colors = ([179, 0, 0],
              [0, 179, 0],
              [0, 0, 0])

rgb_idx = 0  # index counter - primary color we are on
color = (0, 164, 179)  # Starting color
mode = 0  # Current animation effect
offset = 0
prevtime = 0


def rainbow_cycle(wait):
    for j in range(255 * 6):  # 6 cycles of all colors on colorwheel
        for r in range(len(pixels)):
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = colorwheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow(wait):
    for j in range(255):
        for index in range(len(pixels)):
            idx = int(index + j)
            pixels[index] = colorwheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow_cycle_slow(wait):
    for j in range(255 * 3):  # 3 cycles of all colors on colorwheel
        for r in range(len(pixels)):
```

```
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = colorwheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow_hold(wait):
    for j in range(255 * 1):  # 3 cycles of all colors on colorwheel
        for r in range(len(pixels)):
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = colorwheel(idx & 255)
    pixels.write()
    time.sleep(wait)


while True:

    if mode == 0:  # rainbow hold
        rainbow_hold(0.02)
        time.sleep(.5)

    elif mode == 1:  # rainbow cycle slow
        rainbow_cycle_slow(0.02)
        time.sleep(0.05)

    elif mode == 2:  # rainbow cycle fast
        rainbow_cycle(0.005)
        time.sleep(0.050)

    t = time.monotonic()

    if (t - prevtime) > 8:  # Every 8 seconds...
        mode += 1  # Next mode
        if mode > 2:  # End of modes?
            mode = 0  # Start modes over

        if rgb_idx > 2:  # reset R-->G-->B rotation
            rgb_idx = 0

        color = rgb_colors[rgb_idx]  # next color assignment
        rgb_idx += 1

        for i in range(numpix):
            pixels[i] = (0, 0, 0)

        prevtime = t
```

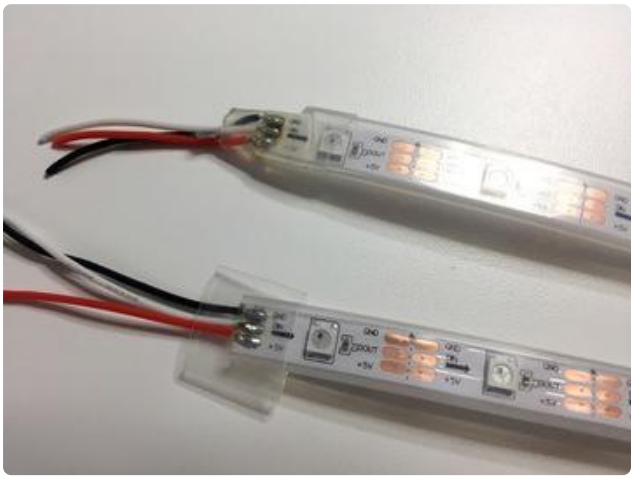
This code requires the neopixel.py library. A factory-fresh board will have this already installed. If you've just reloaded the board with CircuitPython, create the "lib" directory and then download neopixel.py from Github ().

LED Assembly

Look closely at your LED strips and find the "in" end (arrows pointing away). Carefully count out the number of pixels you want in your bandolier. Count them again! Then cut the strip and the silicone sleeve between the copper pads.

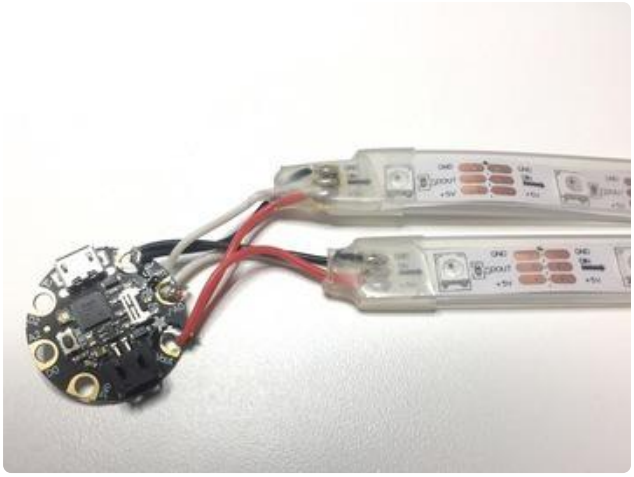


"Tin" the pads on the IN end of each strip by adding a blob of solder with your soldering iron.



Cut two red, two black, and two white wires to about 3 inches long. Strip a tiny bit of the shielding off one end of each wire, then "tin" the wires too, adding a little blob of solder on each one. Solder the wires to the NeoPixel strips: red to +5V, white to IN, and black to GND.

Be sure to check the labeling on your strips and match up the wires to the labels. Some batches of NeoPixels may have the pads laid out in a different order, so follow the pin labels rather than physical positions. If you wire them up incorrectly you could damage your strips or your Gemma.



Trim the wires so they're all about 2" long. Strip the loose ends. Twist the colored pairs together, then solder the red wires to Gemma's Vout pin, the white wires to D1 and the black wires to GND.

Plug your battery in and flip the Gemma's on/off switch to ON. If all your LEDs come on, it's time to secure everything to make it robust.

Slide a piece of 1/2" clear heat shrink over the wire connection. Squirt some hot glue inside the silicone sleeve, and while the glue is still wet, use a heat gun to shrink the heat shrink down. This will create a solid plastic stopper on the end of your strip so the wires will never pull out or get disconnected.

Repeat with the other strip, and be sure to seal the bottom ends with hot glue and heat shrink the same way.

Disconnect the battery when not in use. Gemma's on/off switch only turns off the microcontroller...but power is still connected between the battery and LED strips. NeoPixels draw a tiny bit of power even when they appear "off," and this will slowly drain a connected battery.

Troubleshooting

If your LEDs don't come on, or don't look right, here are a few things to try:

1. Check to be sure you soldered to the IN end of the NeoPixel strips. The strips won't work if you solder them to OUT.
2. Be sure the black wires are connected to GND on both Gemma and the LED strips, and the red wires are connected to +5V and Vout
3. Check to be sure your battery is charged
4. Try re-uploading the code to the Gemma

Bandolier Assembly

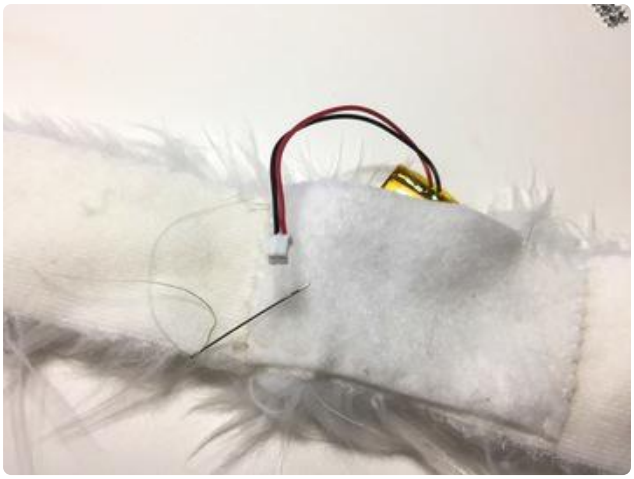
Cut two pieces of white fun fur about 2.5 inches wide and slightly longer than your LED strips. Cut two pieces of 1/4" trim to the same length, and 4 pieces of ribbon to about 2 1/2 feet -- just long enough to tie the bandoliers onto your costume.

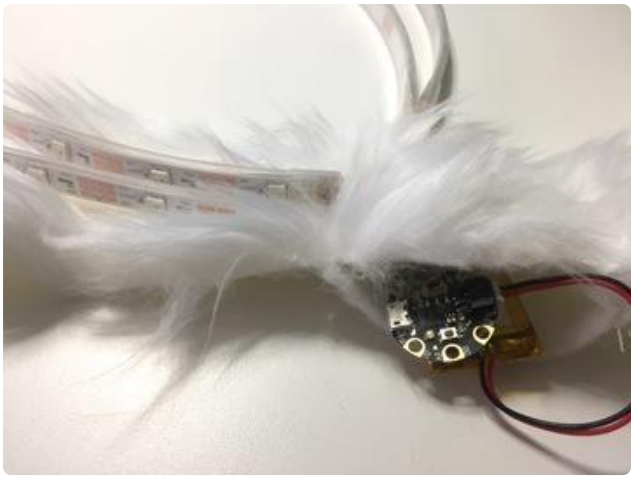
Find the top end of the fur strips -- the end you want to start petting from. Line them up with the furry sides facing inwards.

Sew along the left and right edges for about 1/2 inch: enough to hold the two strips together, leaving a hole in the middle large enough for the Gemma to fit through.

Stitch two ribbons securely to the free ends of both fur strips. You can use a sewing machine or hand-stitch, but it helps to use heavy duty thread for strength.

Cut a small piece of scrap fabric the same width as your fur strips and wide enough to hold the Gemma and your battery. Sew one of a small piece of velcro along the top edge as shown. Sew the other side of the velcro to the back side of your fur, next to the center seam.





Line up the velcro, then hand-sew the other three sides of the fabric to the back of the fun fur. Be sure your battery fits nicely inside, and be sure the velcro keeps it in place. Slide your Gemma into the pocket through the hole you left in the fur's center seam, with the LEDs emerging on the furry side of the fabric.

Place your LED strip down on the furry side of the fur strip, face down. Make sure the wires going to the Gemma are well hidden. Place your 1/4" trim over the back of the strip. Hand-sew the strip and trim in place, catching the silicone sleeve but not piercing the actual LED strip with your needle.

Wear It