NeoPixel Spats with Gemma and MakeCode

Created by Erin St Blaine

https://learn.adafruit.com/neopixel-spats

Last updated on 2024-03-08 03:19:26 PM EST

Table of Contents

Overview

Adorn your gams with a Gemma M0 and a Hotsy-Totsy strand of blinkers. The fellas will turn their heads as you strut by like a daisy in the Cat's Pajamas. You'll need a soldering iron, a sewing machine and a bit of pizazz. Dizzy them up with some drag-and-drop coding over at MakeCode (https://adafru.it/Dyi), and you'll soon be lousy with lookers.

Savvy?

The Goods

1 x NeoPixel Strand — https://www.adafruit.com/product/3630
2" Pitch NeoPixel Strand with 20 Pixels (10 for each leg)

2 x Gemma M0 — https://www.adafruit.com/product/3501
Gemma M0 MicroController (get one for each leg)

2 x On/Off Switch — https://www.adafruit.com/product/1092
Quick Click switch

2 x JST Extension Cable — https://www.adafruit.com/product/1131
For connecting the switch to the project

2 x Battery — https://www.adafruit.com/product/1578
500mAh LiPoly Battery

Keep your batteries full of juice

1 x Battery Charger https://www.adafruit.com/product/1304

Keep your batteries full of juice

Also Needed

- 1 yard of stretchy fabric (I used cotton rib knit in white)
- Sewing Machine (or a needle and thread and some patience)
- Soldering iron and accessories (http://adafru.it/136)
- Ritzy decorations -- buttons, gears, jewels or a bit of flim-flam
- E6000 glue, or other fabric / embellishing glue

Code with MakeCode

MakeCode is Microsoft's ritzy drag-and-drop code editor. It makes it easy to get up and running with the Gemma M0.

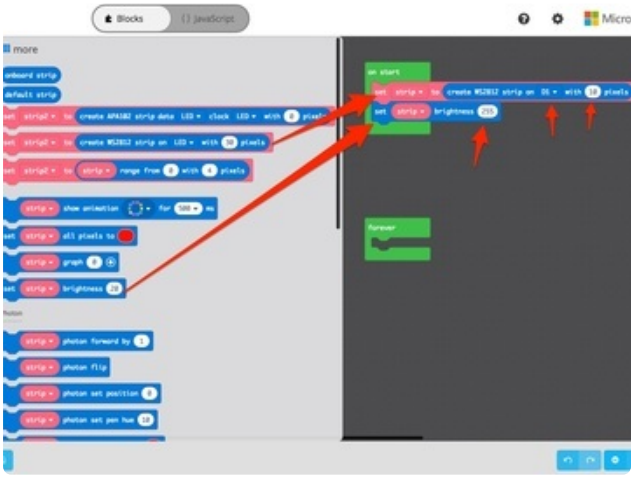
Find the Gemma M0 code editor at maker.makecode.com (https://adafru.it/DWO)

Here's my completed project, which runs a pink, blue and purple gradient along the LED strand. Click the Download link at the bottom, plug your Gemma M0 into your computer, click or double-click the reset button and drag the file onto the resulting **GEMMABOOT** drive.

Or, follow along below to create your own project from scratch.

Code It Yourself

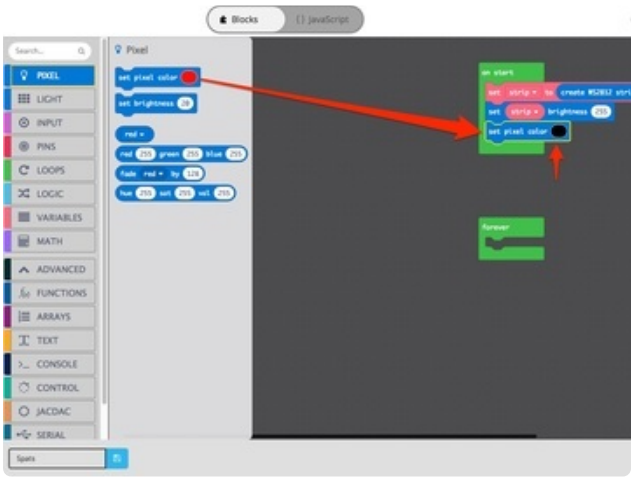
Head to https://maker.makecode.com (https://adafru.it/C9N) and select **New Project**. Choose the Gemma M0 board. Give your project a name. I called mine **Spats**.



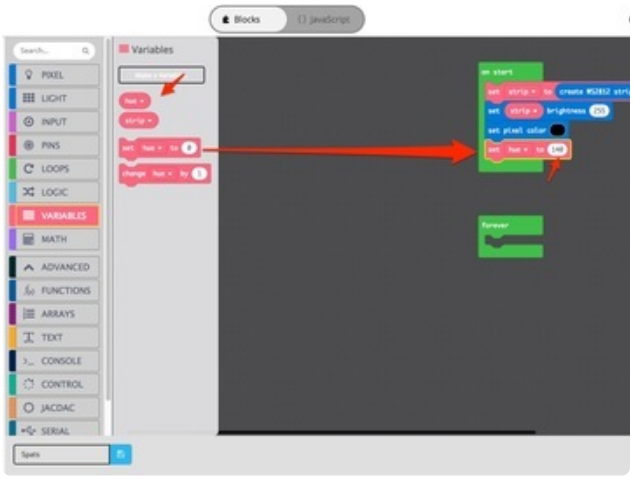
Click the **LIGHT** tab. Another tab titled **MORE...** will appear. Click on that one to find lots of options for attaching and controlling a NeoPixel strand soldered to the Gemma M0. Drag an instance of `set strip to create WS2812 strip on LED with 30 pixels` out into your workspace and place it inside the `on start` loop, which is usually there by default in a new project. (If it's not there, you can find it under the **LOOPS** tab). Change the pin to **D1** instead of **LED**, and enter **10** pixels, or however many you've planned to put in each spat.

Note: NeoPixels are also called WS2812 pixels, and DotStars are also called APA102 -- make sure you've got the right ones selected.

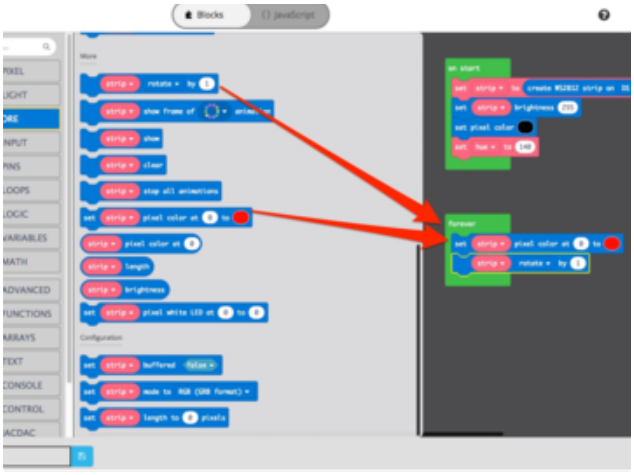
Next, drag an instance of `strip set brightness` into your `on start` loop. Set the brightness to whatever you'd like. I want them as bright as possible, so I chose `255`.



Above the **LIGHT** tab there's another tab called **PIXEL**. Anything in here controls the onboard NeoPixel on the face of the Gemma M0. I want this turned off on my spats -- no stray status lights for me, thank you -- so I dragged an instance of `set pixel color` into my `on start` loop and made it **black**. This will make sure the pixel stays off.



I want to be able to animate and change the colors of my NeoPixels over time. The way to do this is to assign the color a variable, and then change the variable this way or that, causing the lights to animate. Go to the **VARIABLES** tab and create a new variable called `hue`. Then, drag an instance of `set hue to 0` into your `on start` loop and pick a starting hue. I want a purple-pink-blue gradient, so I chose **140** as a starting hue which will give me a nice blue color.



Go back to the **LIGHT** > **MORE...** tab and drag an instance of `set strip pixel color at 0 to red` into your `forever` loop. Things in this loop will run forever, over and over. Then drag an instance of `strip rotate by 1` beneath it. This snippet will create the animation on the strip, as soon as we tell it what to do.

Hue, Saturation and Value

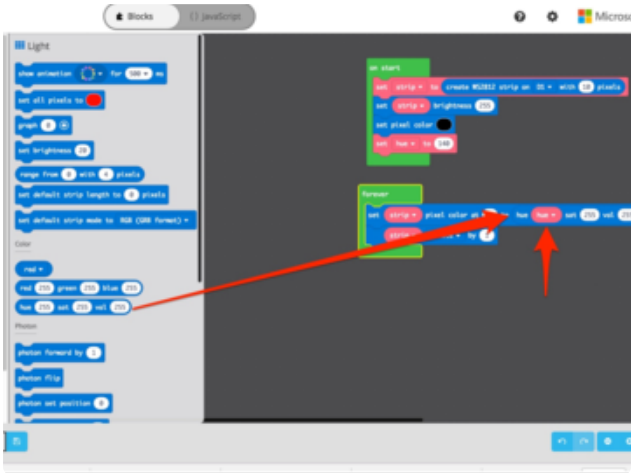
Hue, saturation and value are a means of defining a specific color using just numbers, on a scale of 0-255.

Hue refers to the color on the spectrum -- red, blue, purple, etc. Red is at 0, blue is at around 140, purple comes in at around 180, and the spectrum loops around back to red at 255.

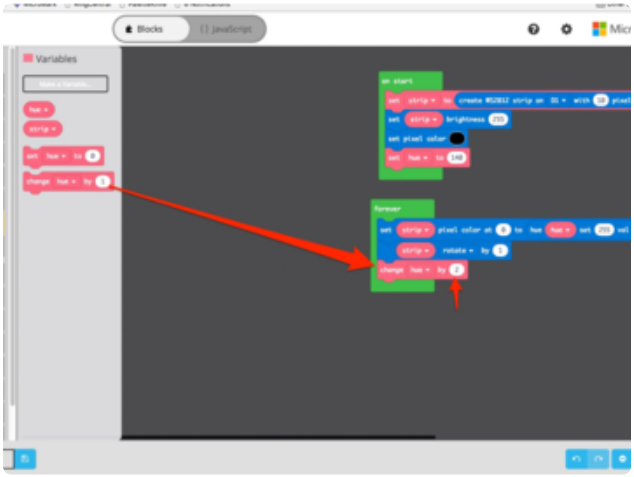
Saturation refers to how pastel or how vivid the colors are: 0 is a plain white, no matter what your hue is -- any completely unsaturated color just turns white. 255 is fully vivid and saturated with color.

Value, for our purposes, refers to the brightness or intensity of the color. The higher the value, the more red or blue is added.

You can read far more than you ever wanted to know about this on Wikipedia (https://adafru.it/CxV). What matters to us is that we can mess with the NeoPixel colors in all sorts of fun ways using hue, saturation and value.

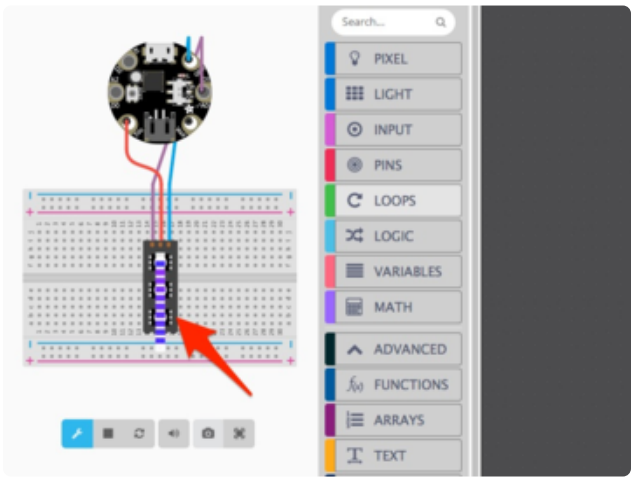


Under the **LIGHT** tab you'll find a block for `hue 255 saturation 255 value 255`. Drag this into your set `strip pixel color` block, replacing the `red`. Then, go into your **VARIABLES** tab and grab an instance of your new favorite variable, `hue`. Use this variable to replace the **255** in the `hue 255` block. Now we can mess with the pixel hue by changing the `hue` variable.

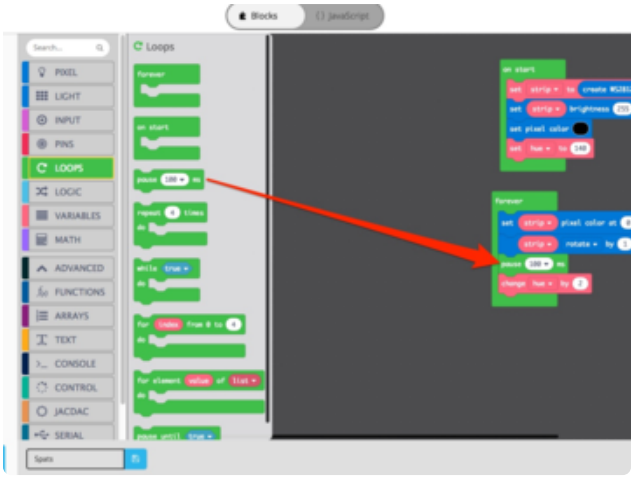


So, let's do exactly that. Grab an instance of `change hue by 1` from your **VARIABLES** tab and put it into your `forever` loop. I made mine a little more dramatic by changing the hue by **2**. Try some different values in here to see what you like.

Testing It Out



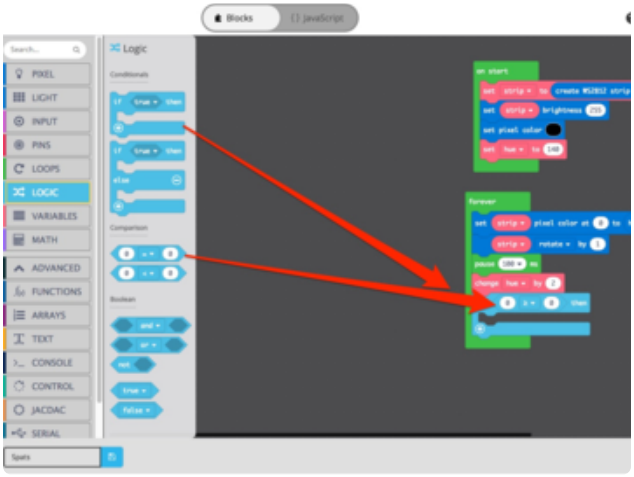
You may have noticed that over on the left, a little NeoPixel emulator strip has appeared beneath your Gemma M0. This emulator will run whatever your code is doing. This is a really easy way to preview your animations and decide if you like the effect, without having to download the code every time. My emulator is starting to look good, but it's running a bit faster than I'd like. Let's slow it down a bit.



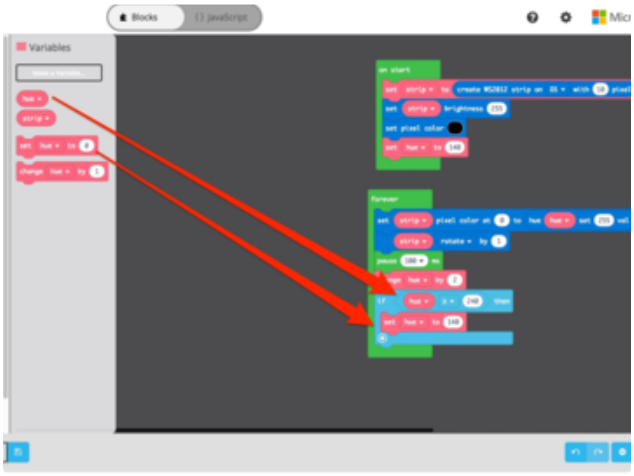
Under the **LOOPS** tab, find `pause 100 ms` and drag it into your `forever` loop. Check the emulator again. Do you like it better? Find a speed that works for you. You can also randomize the timing (`pick random` under the **MATH** tab) if you'd like.

Almost there! We've created a customizable rainbow animation on our strip. However, I don't want a full rainbow -- I just want a partial rainbow, or a gradient animation. We can limit our hue variable to only show hues in one part of the spectrum using an `if / then` statement.

We'll start with a hue of **140** (blue), then increase by **2** until we reach **240** (magenta), at which time we will place a limit and tell the code to circle back around to **140**.



Under the **LOGIC** tab, find an `if true then` block and drag it into your `forever` loop. Grab a `comparison` block to replace the `true`, then change the `=` to `>=`. Now we don't have to worry about hitting our limit number exactly.



From the **VARIABLES** tab, grab another `hue` variable and use it in the first comparison spot. Set the second comparison spot to **240**. Drag an instance of `set hue to 0` and place it inside the `if / then` block, and set the value to **140**.

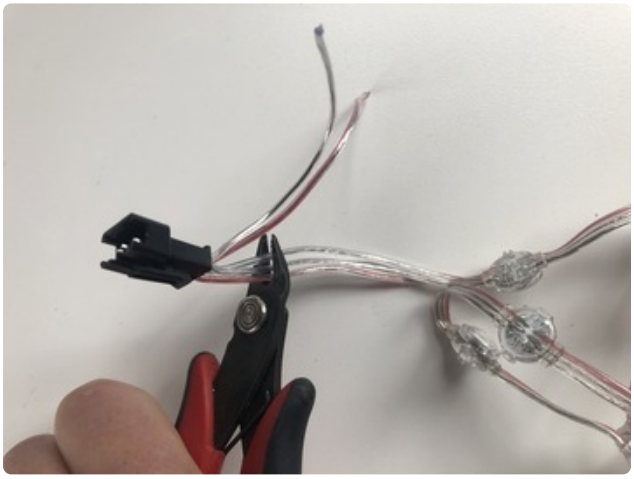
Now the hue number will count up from **140** (since you set it to **140** in the setup block) by **2**'s, until it reaches **240**, at which point it will pop back down to **140**. This effectively makes the NeoPixels change from blue (**140**) to magenta (**240**) and then back again. Success! We've got our gradient.

It's easy to change these values to get the color gradient you want.

Plug your Gemma M0 into your computer using a USB cable. Click the tiny **Reset** button in the middle of the board and the onboard pixel will turn green. When the lights turn green, a new drive will appear on your computer called **GEMMABOOT**.

In MakeCode, click the Download button to download your code to a file. Drag this file onto the **GEMMABOOT** drive. It may also ask you if you'd like to pair the board to your computer. Feel free to follow the directions to do this (you may need to update the bootloader), or you can skip it and simply drag the downloaded file onto **GEMMABOOT.** Pairing the board makes it easier if you want to download and test again and again, since it will load the code automatically without you needing to drag the file each time.

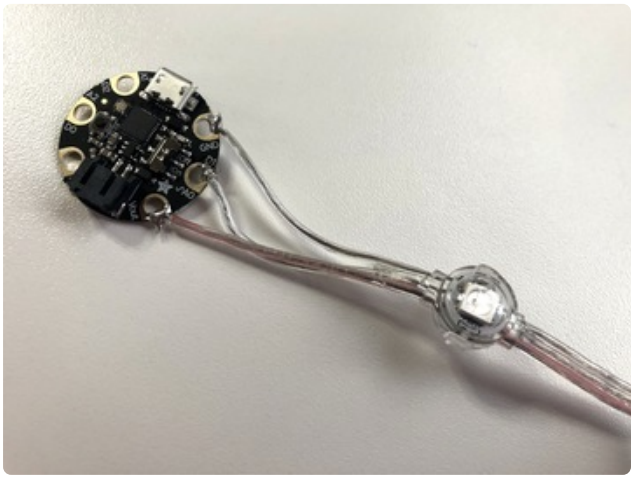
Electronics Assembly



Find the end of your LED strand with the **male** connector. This should be the **IN** end of the strand. Double check this is correct by looking carefully at the back of the LEDs. Sharp eyes can spot an arrow pointing in the direction of data flow.

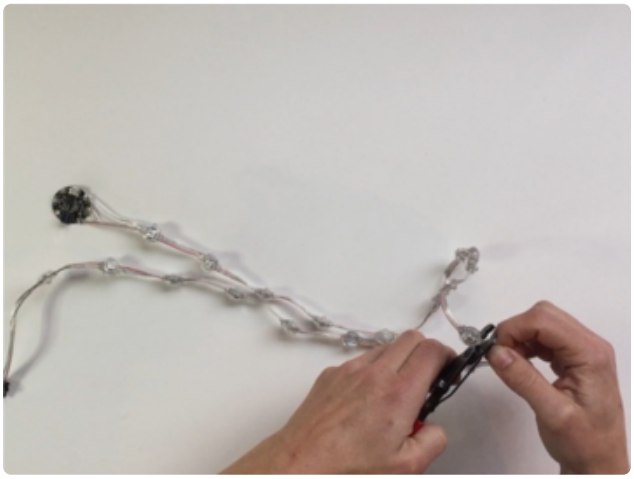
This is very hard to see. If you're not sure, start with the end that has a male connector - 90% of the time this is the **IN** end. If your lights don't come on down the road a bit, you may have a strip that's manufactured differently -- try the other end.

Cut the connector off with wire cutters and strip about 1/4" from each of the three wires.



Wrap the red wire around the Gemma's **VOUT** pin, the middle wire around **D1** and the remaining wire around **G**. Solder the wires in place. Plug a battery into your Gemma and see if the lights come on. If they do, swell. If not, check the on/off switch on the face of the Gemma and make sure it's switched to ON. (It's easy to miss!)

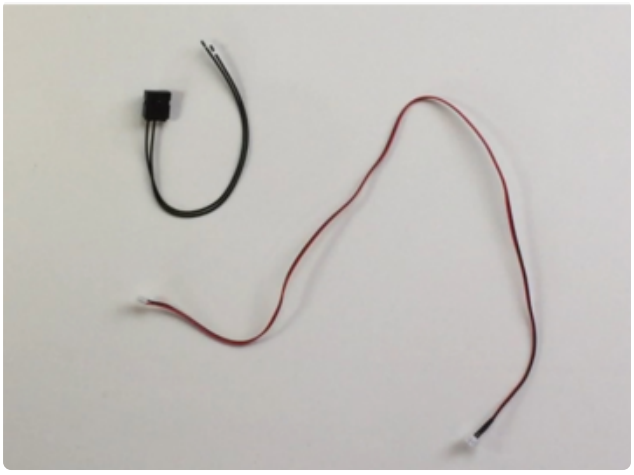
If your lights still don't come on, try re-uploading your code or try using a different battery. If it's still not working, try soldering to the other end of your LED strand, you may have misread the direction of data flow.



Find the middle of your LED strand and cut the wires close to the 10th LED. Add the second Gemma M0 to the wires you just cut in the same manner. Test this strip as well, and if all is working, finish by trimming the female connector off the tail end of this strip.

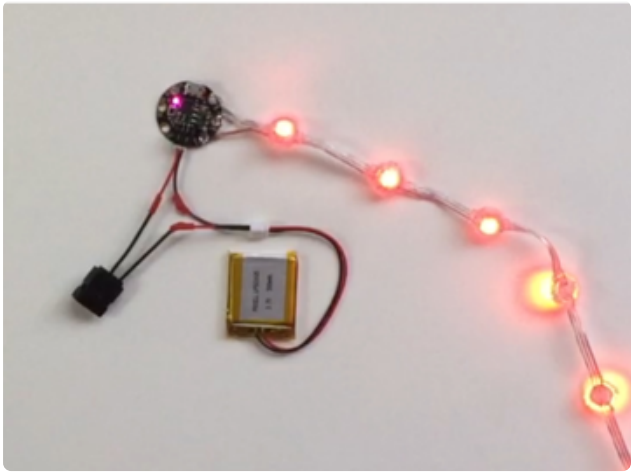
On/Off Switch

You can use that itsy bitsy onboard switch as your main on/off switch if you'd like. But it's tiny and a little tricky to finger while you're walking around the juice joint, and truth be told, it will slowly drain your battery dry if you leave it plugged directly in -- even while it's switched off. Adding a larger physical on/off switch will keep your battery and your gin-addled fingers copacetic.



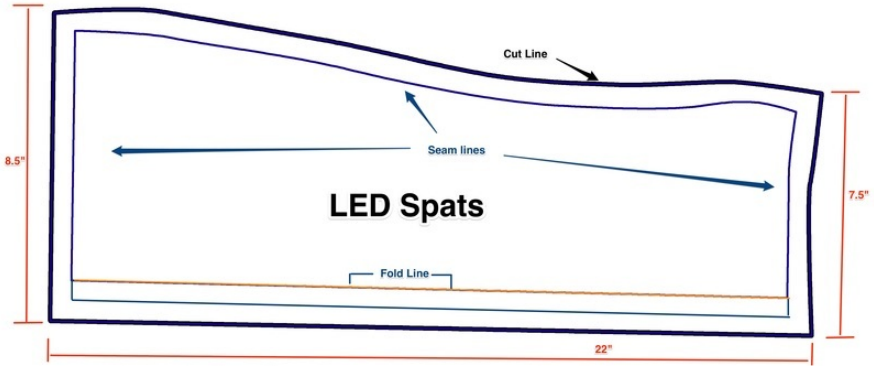
Find your tactile on/off switch and your battery cable. Clip the lead wires on the switch, and trim off both ends of the battery cable to about 2 inches.

Slip on some heat shrink. Solder the red wires from each JST connector to the leads from the switch. Solder the two remaining black wires back together again. You'll end up with a triangle shaped short JST connector line with a switch in the middle of the red wire, as shown.



Plug your battery into the switch adapter and the switch adapter into the Gemma M0. Click your switch and test to be sure the lights go on and off. (If not, check that onboard Gemma switch again, that cat can be a real hayburner).

Sew the Spats



Here's a downloadable and printable pattern. Print each of the 3 pages on letter sized paper, then tape them together to get the whole shebang aligned.

Adjust to fit your measurements until you've got a ringer. On me, floor-to-knees is 22" and halfway around my knee is 7.5". My spats come to just below the knee, but yours can be any size that makes you feel dolled up.

spats_pattern.zip

https://adafru.it/DTI

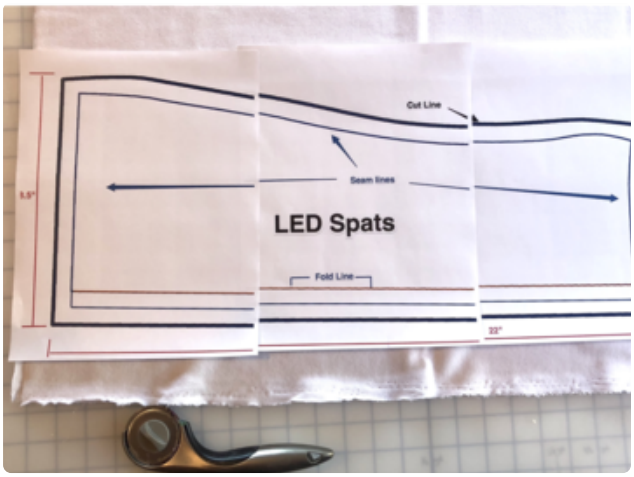
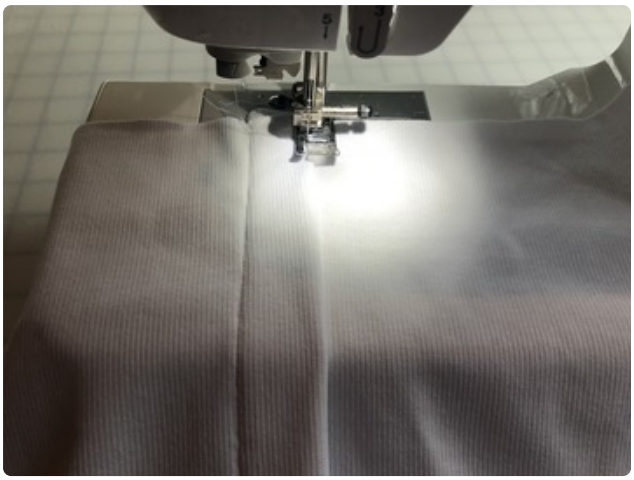


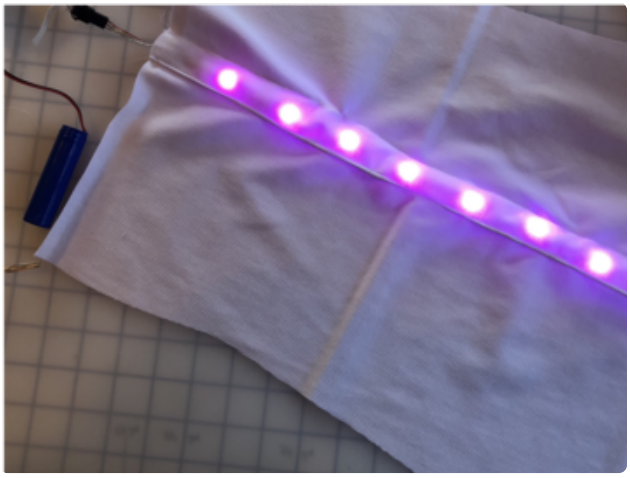
Figure out which way your fabric wants to stretch. Lay the pattern on the fabric so the stretch will end up going around your leg and the non-stretch direction spans the knee-to-floor length. Cut two of the pattern shape for each leg (so 4 copies total for 2 spats).

Stitch or serge along the straight cut line (the one next to the fold line in the pattern. I've allowed about 5/8" seam allowance, but since the fabric is stretchy this isn't too critical.

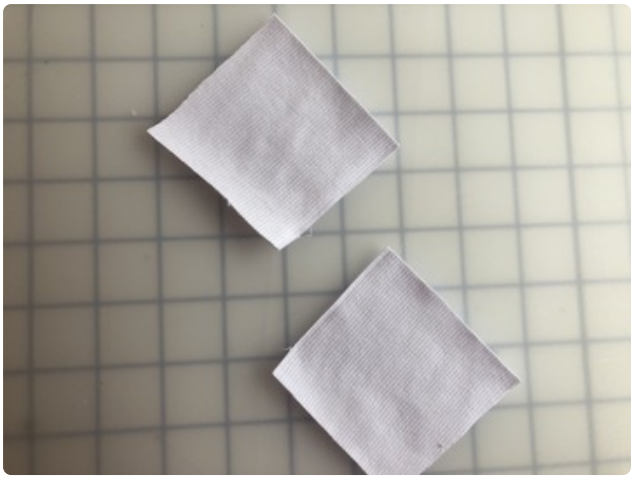


Take the seam you just made and lift it up off the table about 3/4 inch. Fold it over, making a doubled-up section that will act as the pocket for the LEDs. Pin this flap in place.

Stitch through all layers along the original seam line. Flip your spat over and stitch again along the outside edge of the fold you just made. This will secure the pocket on both sides, as well as giving you a nice tailored look.



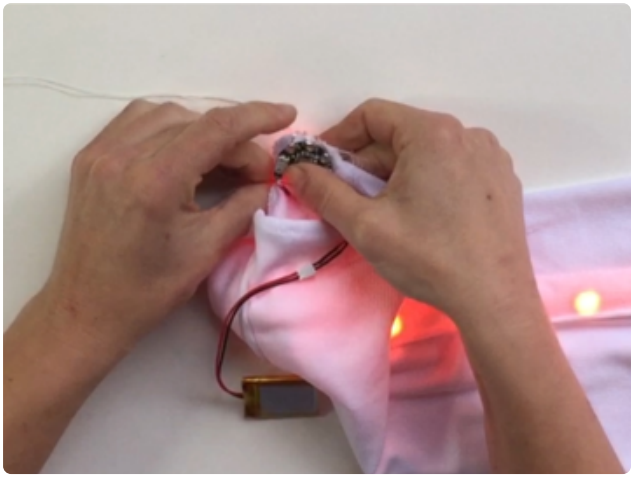
Test-fit your LEDs in the pocket by slipping them in from one side. They should fit snugly without slipping around too much. Pull them back out again for now.

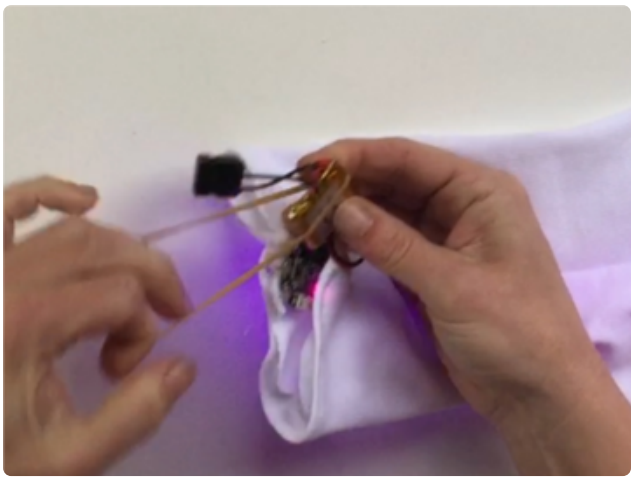


Cut a small square of fabric for each spat that's the right size to hold the battery and Gemma M0, so they won't rub against your leg. Stitch the pocket along the inside of the spat near or behind the LED pocket, leaving the top open.

Fold the spat in half with right sides together, and stitch or serge the opposite side. Try it on to be sure it fits. Once you're happy with the fit, make a narrow hem along the bottom. Make another hem along the top, leaving the battery pocket and LED pocket open so you can get the electronics in.

Be sure to use a zigzag or stretch stitch for the hems since this will pull when you put the spats on.





Slip the LED strip into their casing, being sure they're all facing outwards. Sew the Gemma inside the battery pocket using the unused pads. Wrap a rubber band around the battery and switch connector, trapping the battery leads inside. This will add some strain relief and keep your battery leads from breaking off, which can be a problem with these types of batteries. Slip the whole assemblage into the battery pocket.

Finally, glue some embellishments on the outside of the spats above each light. Buttons work great here, or metal gears from the scrapbook section at the craft store (if you're going for Steampunk). I settled on black leather Adafruit logo stars, cut on the laser cutter at my makerspace. Just use whatever irons your shoelaces.