adafruit learning system

Adafruit LED Sequins

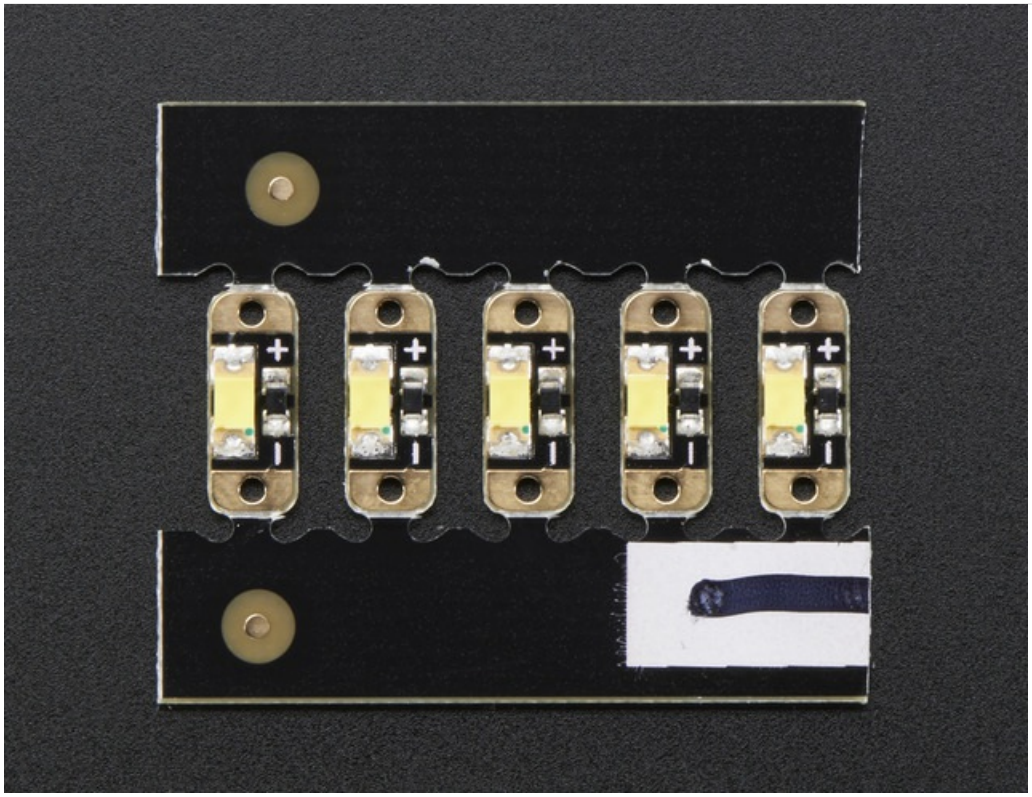
Created by Becky Stern

Last updated on 2018-08-22 03:40:45 PM UTC

Guide Contents

Guide Contents 2
Overview 3
Sewing with conductive thread 6
Circuit Diagram 10
GEMMA sequin hat 11
Arduino Code 15
CircuitPython Code 17
Downloads 19

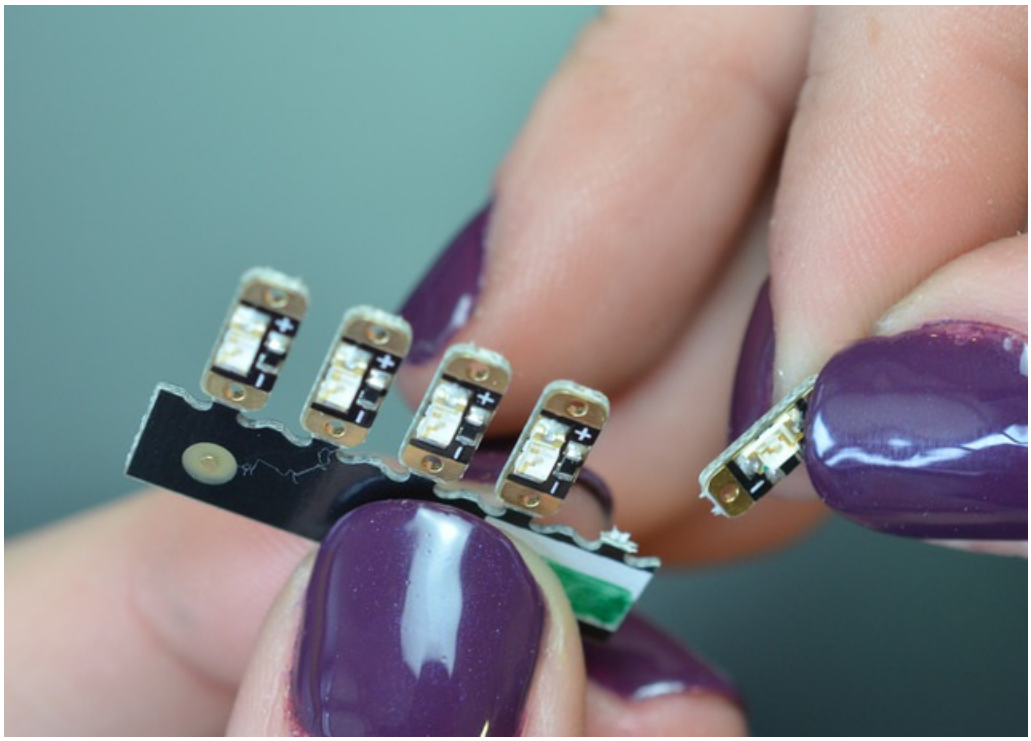
Overview



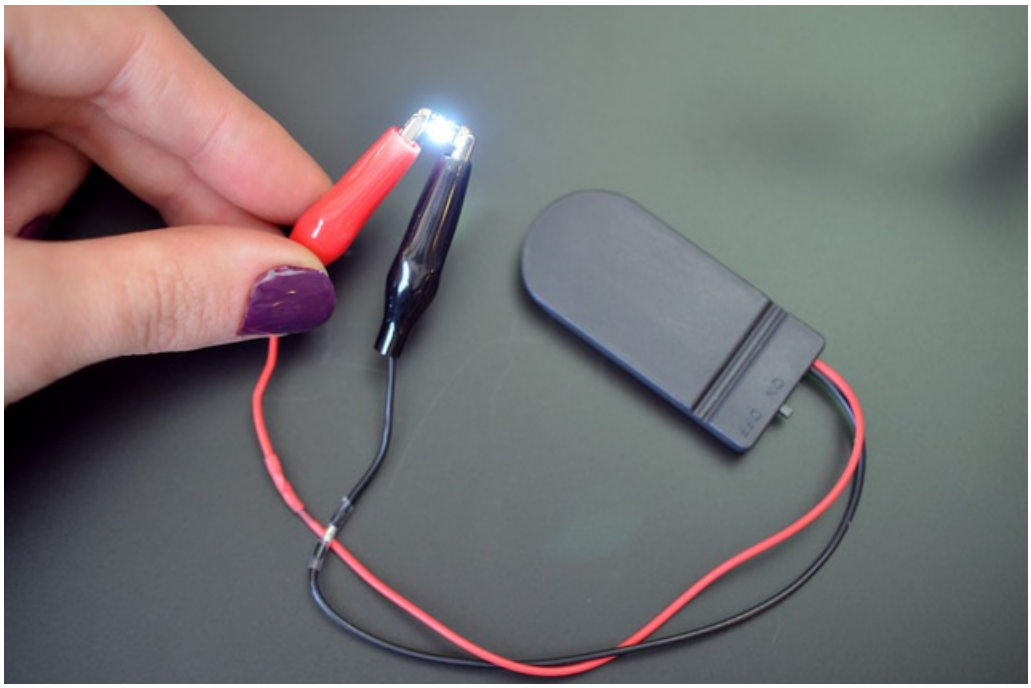
Sew a little sparkle into your wearable project with an Adafruit LED Sequin. These are the kid-sister to our popular Flora NeoPixel-- they only show a single color and they aren't addressable, but they are our smallest sewable LEDs ever and very easy to use!

Before you get started, follow the Introducing GEMMA guide (https://adafru.it/cHH) or Introducing Gemma M0 guide (https://adafru.it/yeq)

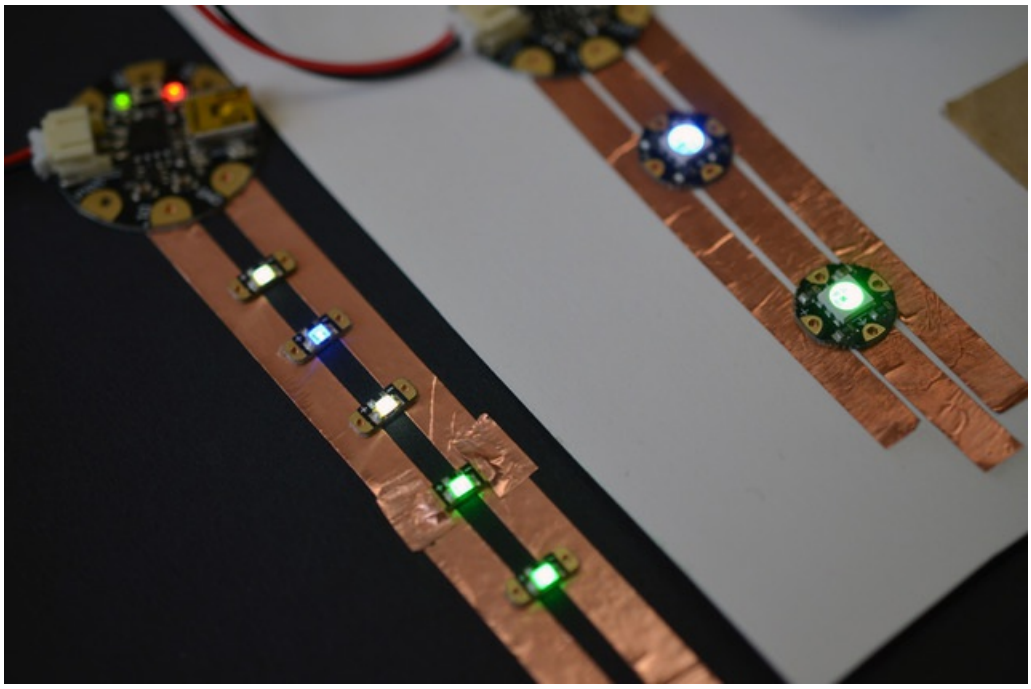
> This guide was written for the 'original' Gemma board, but can be done with either the original or M0 Gemma. We recommend the Gemma M0 as it is easier to use and is more compatible with modern computers!



They come in packs of five with breakaway tabs-- you can separate them one at a time as you build your project.



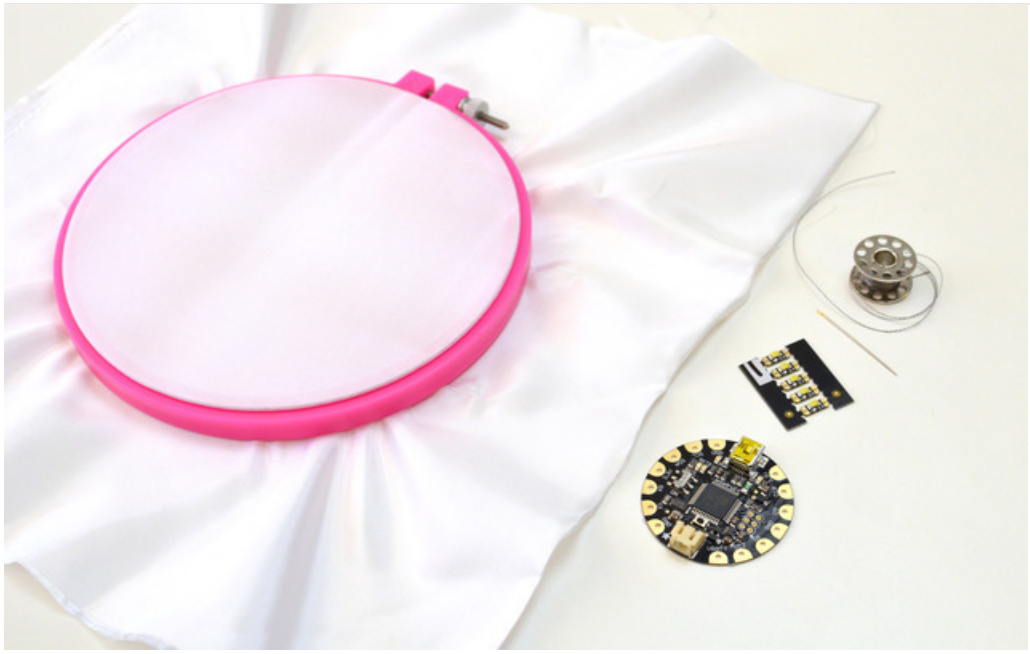
Simply connect 3 to 6VDC to the + pin and ground to the - pin, and the LED on the board will light up. In the photo above we've soldered alligator clips to a 2x2032 coincell battery holder (~6V). You could also use a single CR2032 sewable coincell holder-- you do not need a microcontroller to drive these sequins, unless you want them to blink or fade.



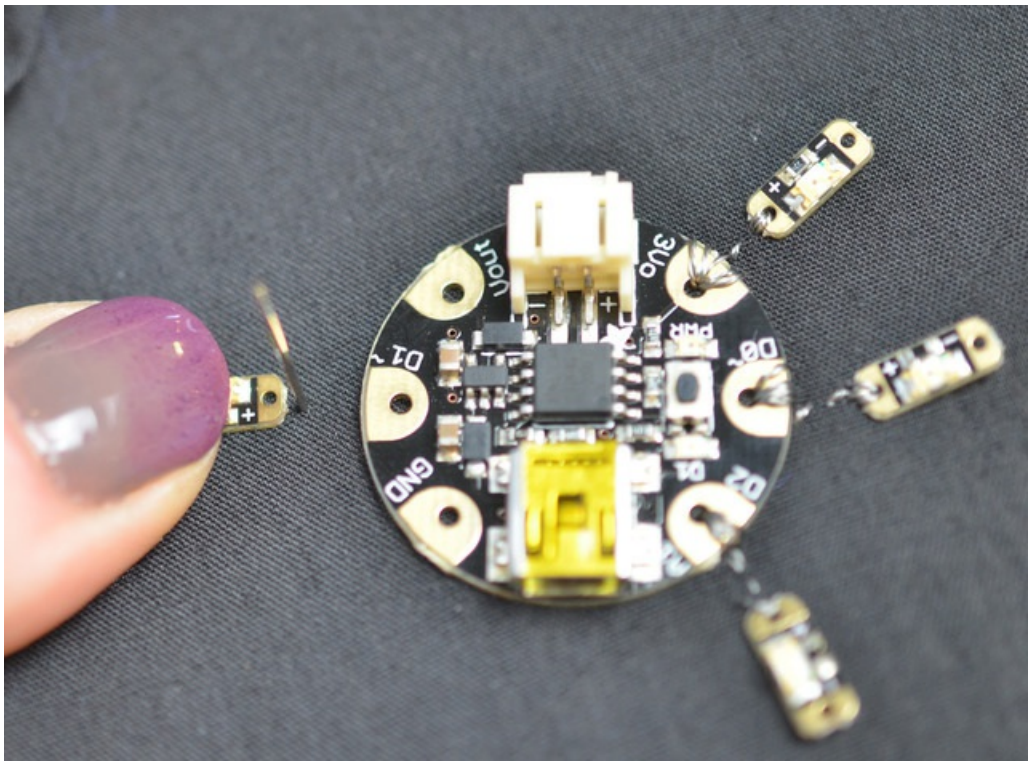
When powered from 3.3V they draw about 5mA so you can put up to 4 or 5 in parallel on a single microcontroller pin. We currently have these sequins in warm white, emerald green, ruby red, and royal blue.

You can make the LEDs fade and twinkle by using the PWM (a.k.a. analogWrite) functionality of your Gemma or Flora, or just connect directly to a digital I/O pin of a microcontroller to turn on and off (digitalWrite).

Sewing with conductive thread



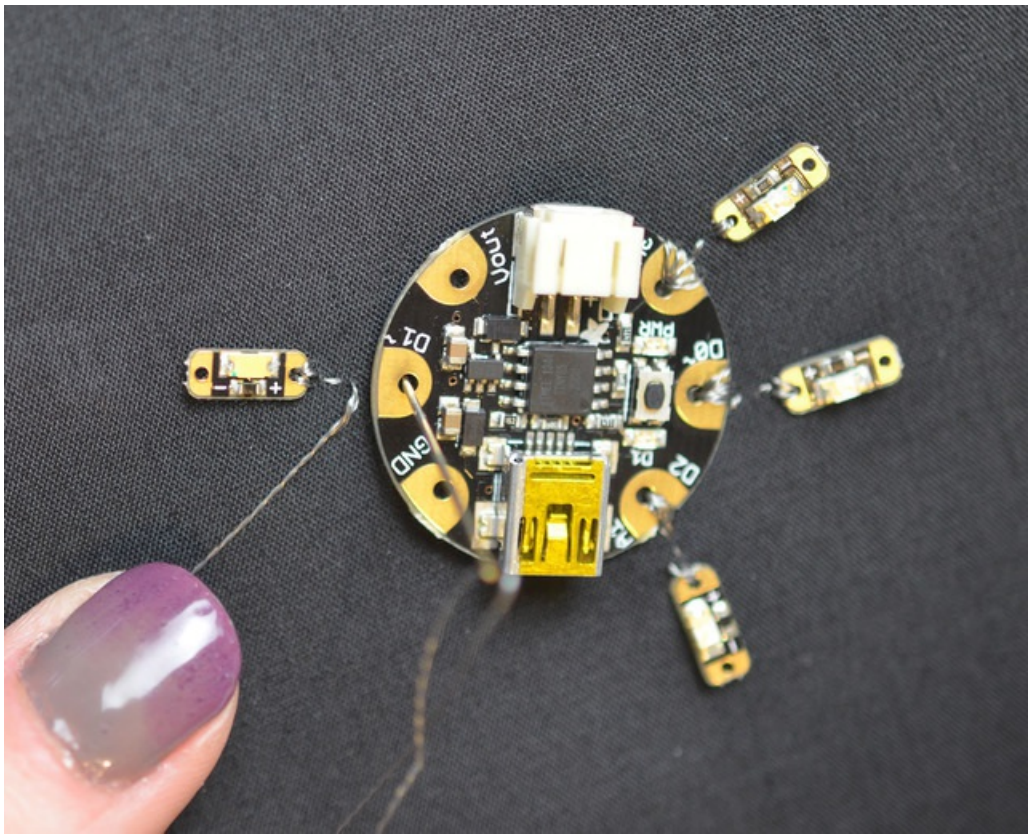
Grab a needle and test it's small enough to pass through the holes in the sequins. Thread up your needle with conductive thread, our 2-ply and 3-ply both are great for this purpose. Check out our conductive thread guide (https://adafru.it/aVx) to learn more about this stainless steel fiber.



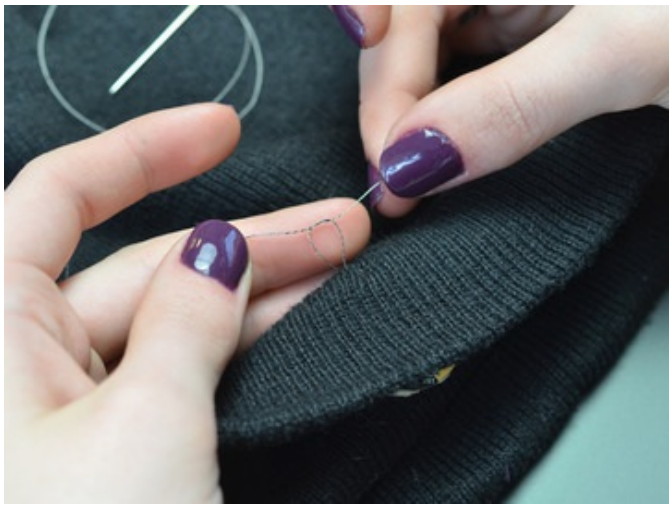
We find it easiest to load up your fabric in an embroidery hoop to keep it taught during stitching, especially for beginners.

Begin by piercing the needle through the fabric from back to front, near the + on the pixel leaving a six-inch thread tail at the back. Then affix the pixel to the fabric by piercing the needle down through the hole marked + and through to the back of the fabric. Repeat to make a few stitches around the pixel's + connection.

Avoid using pin D1 when using the Gemma M0. Instead use three LEDs in parallel on using pins D0 and D2.



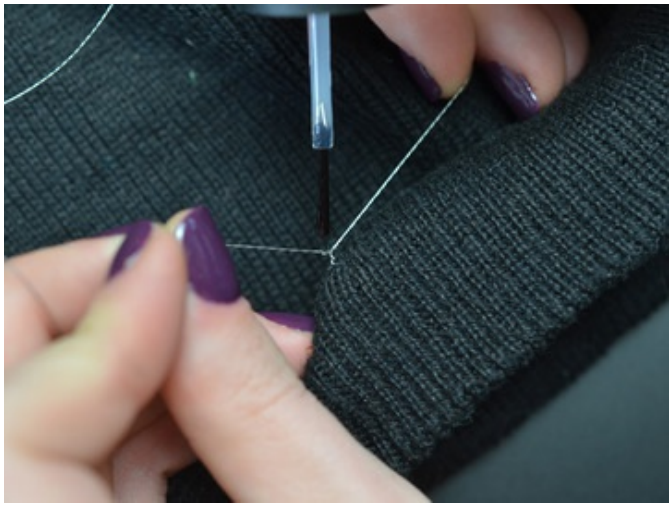
Stitch over to a digital or analog output on your microcontroller, and repeat the stitching process around the circuit board's pad.



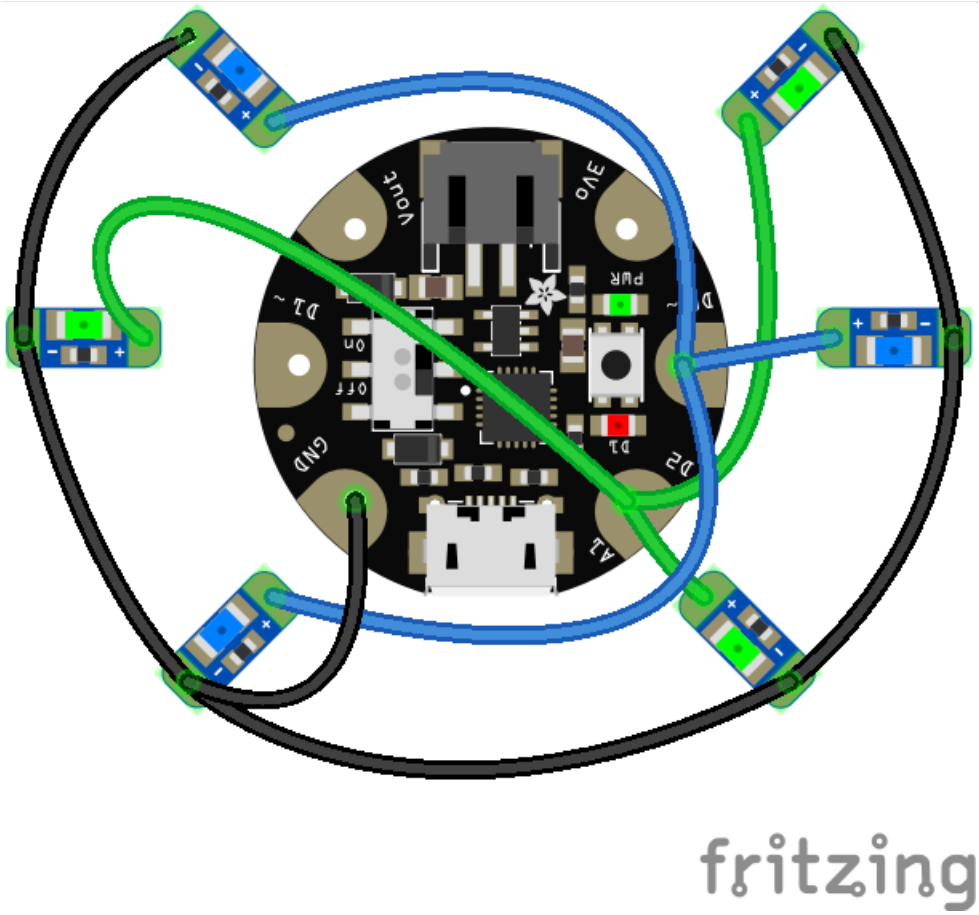
Stitch back over to your thread tail at the back and tie the two threads in a double knot.

Seal the knot from springing loose with some clear nail polish or other adhesive.

Snip the thread tails short.



Circuit Diagram



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

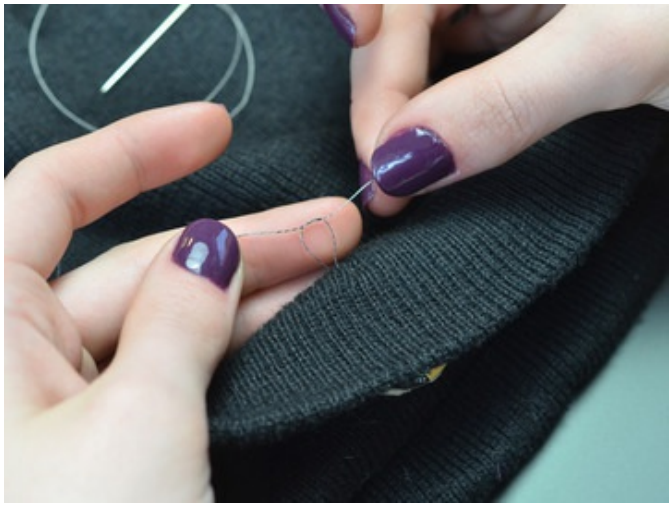
Three of the sequins connect to D0 and the other three connect to D2. The sequins are connected in parallel to these pins.

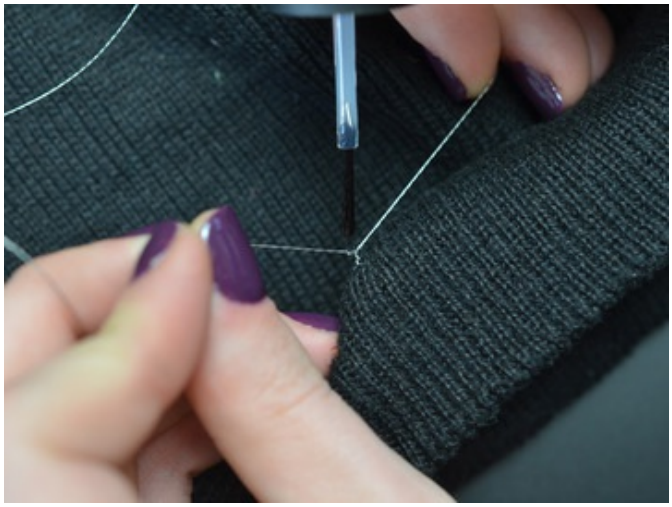
The original version of this project used D1 also, but since the Gemma M0 does not have PWM on this pin, we've tweaked it a bit to only use D0 and D2!

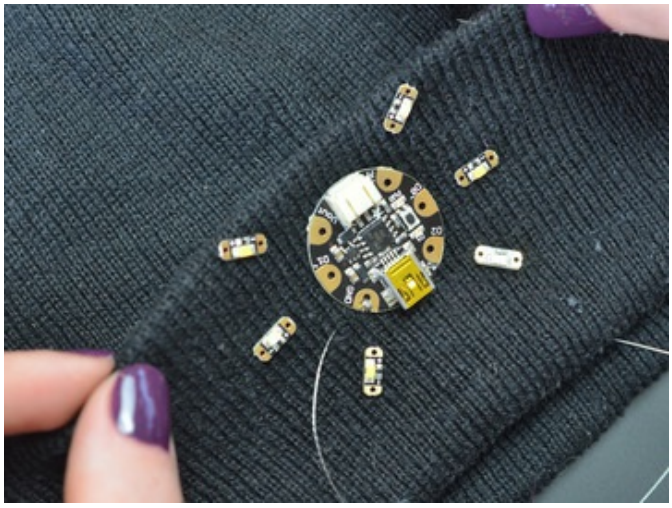
GEMMA sequin hat

LED sequins are great for clothing and accessories! Here's a simple hat project to build with GEMMA.

You can start with the pixel + or -. For this hat we chose to start with the shared ground line that hooks up all the sequins. Stitch around the GND pad on GEMMA at the edge of a knit cap and knot/seal at the back.



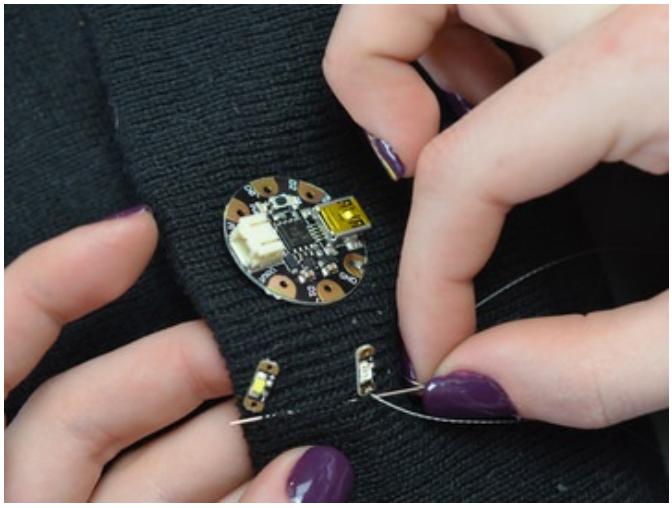


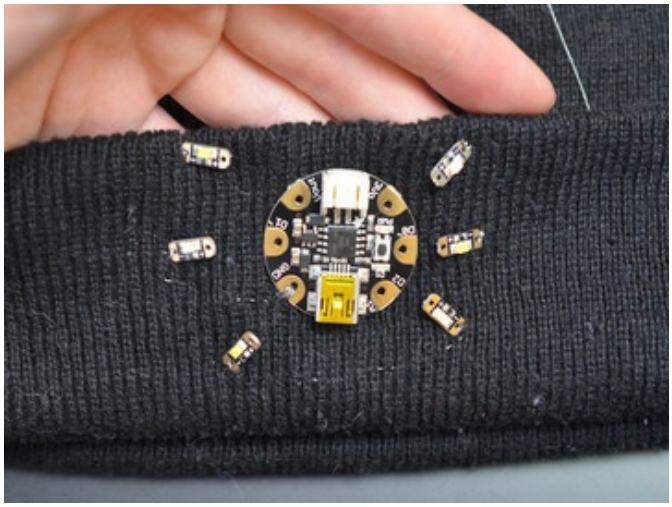


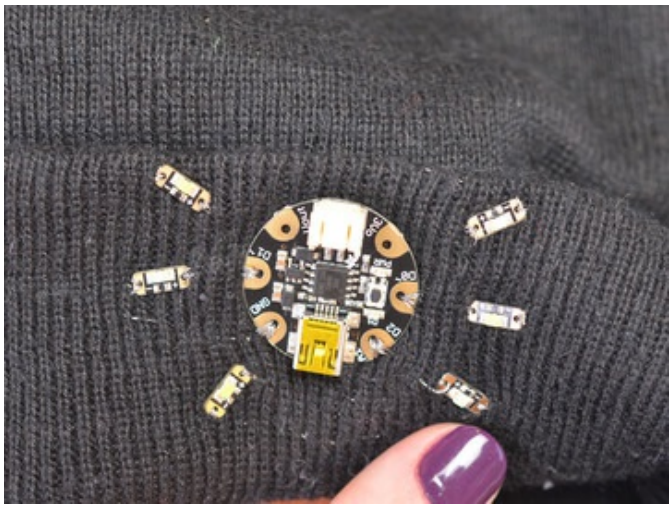
Lay out your own design or use our circuit diagram above, with the sequins' + sides facing the microcontroller.

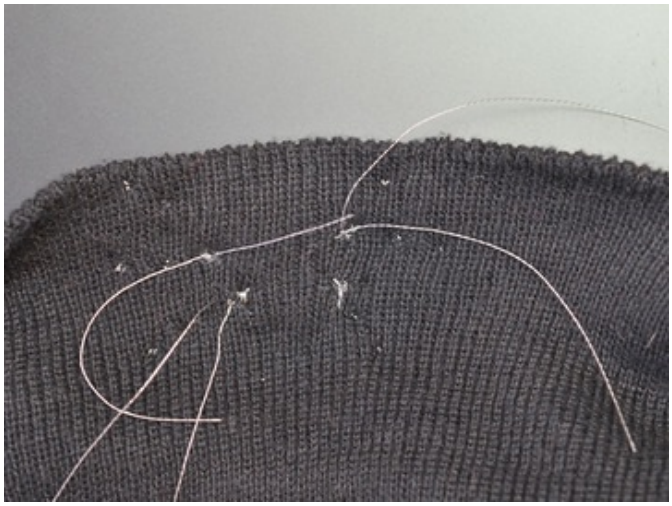
Continue stitching the ground line all the way around the perimeter of your sequin design, stitching to every sequin's - pad as you go.

Knot to your original knot, seal the knot, and trim the ends once dry.









Next hook up the + connections from each of GEMMA's outputs to three pixels. We stitched from one sequin to the GEMMA, then over to another sequin and on to the third sequin before going back to the first, making a sort of square that results in the thread tails being in the same place.

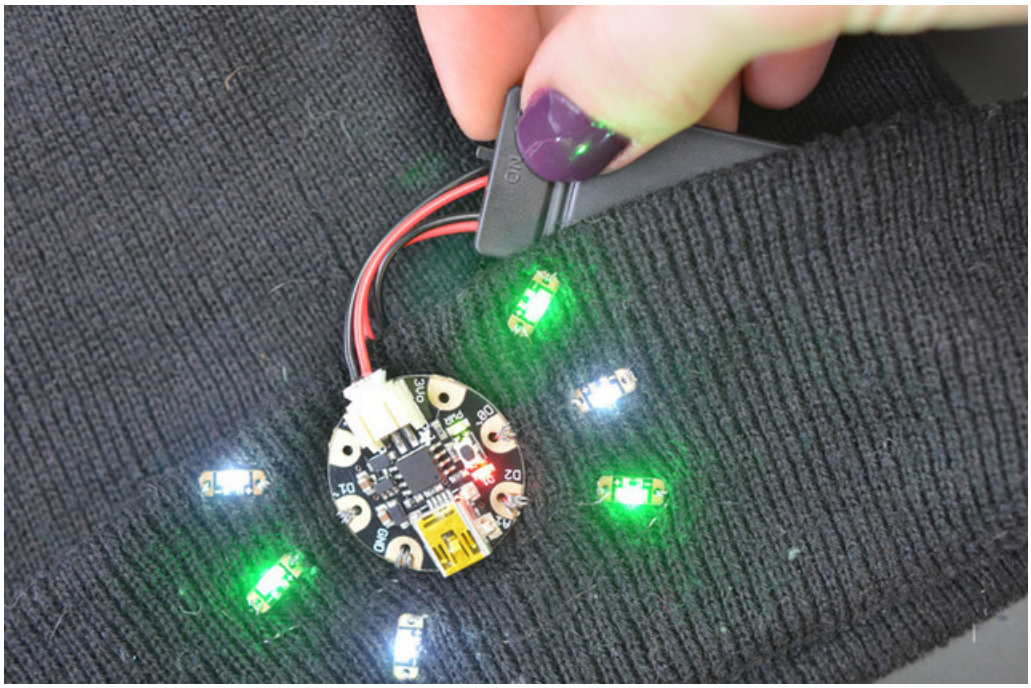
Knot these thread tails, seal and trim short.

Arduino Code

Plug in GEMMA over USB and load the following code into your Adafruit Arduino IDE. If you haven't before, check out our Introducing GEMMA guide (https://adafru.it/cHH) to get started with the software.

```
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by
int counter = 0;       // counter to keep track of cycles

// the setup routine runs once when you press reset:
void setup()  {
  // declare pins to be an outputs:
  pinMode(0, OUTPUT);
  pinMode(2, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of the analog-connected LEDs:
  analogWrite(0, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount;
    counter++;
  }
  // wait for 15 milliseconds to see the dimming effect
  delay(15);

// turns on the other LEDs every four times through the fade by
// checking the modulo of the counter.
// the modulo function gives you the remainder of
// the division of two numbers:
  if (counter % 4 == 0) {
    digitalWrite(2, HIGH);
  } else {
   digitalWrite(2, LOW);
  }
}
```

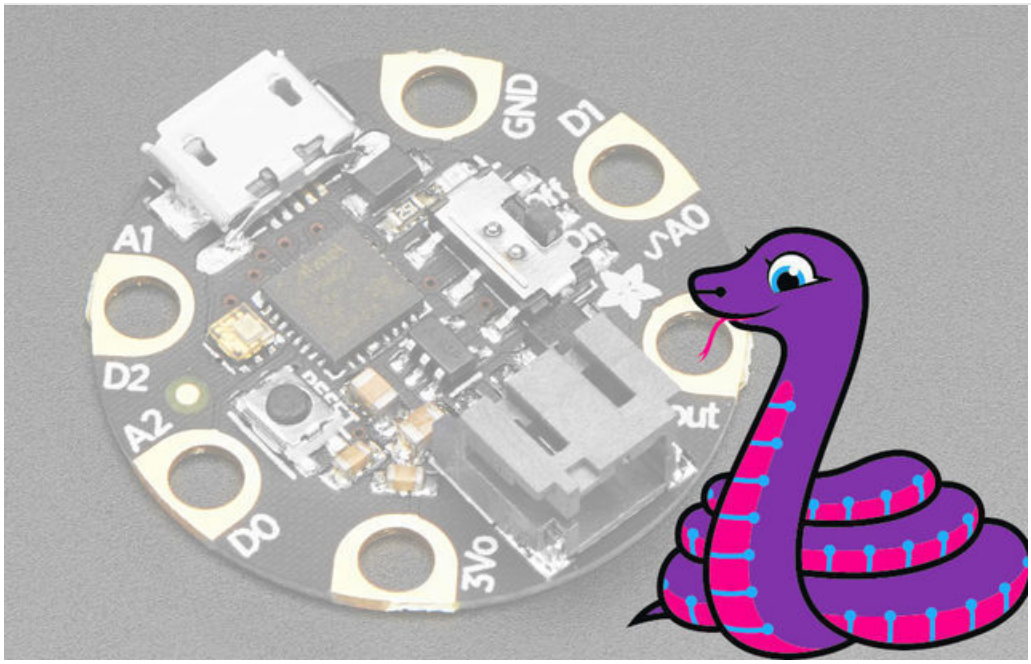
This sketch is a mash-up of two very basic Arduino examples: blink and fade. You can modify it to display the patterns of light you like best or code up your own sketch starting with the examples provided from inside Arduino.



Once your program is doing what you like, unplug the USB cable and plug in a battery pack like our 2x2032 holder with on/off switch.

Take the batteries out if you get stuck in a rainstorm and for washing. Enjoy your new light-up hat!

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 "**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.

```
import time

import board
import pulseio
from digitalio import DigitalInOut, Direction

# PWM (fading) LEDs are connected on D0 (PWM not avail on D1)
pwm_leds = board.D0
pwm = pulseio.PWMOut(pwm_leds, frequency=1000, duty_cycle=0)

# digital LEDs connected on D2
digital_leds = DigitalInOut(board.D2)
digital_leds.direction = Direction.OUTPUT
brightness = 0  # how bright the LED is
fade_amount = 1285  # 2% steping of 2^16
counter = 0  # counter to keep track of cycles

while True:

    # And send to LED as PWM level
    pwm.duty_cycle = brightness

    # change the brightness for next time through the loop:
    brightness = brightness + fade_amount

    print(brightness)

    # reverse the direction of the fading at the ends of the fade:
    if brightness <= 0:
        fade_amount = -fade_amount
        counter += 1
    elif brightness >= 65535:
        fade_amount = -fade_amount
        counter += 1

    # wait for 15 ms to see the dimming effect
    time.sleep(.015)

    # turns on the other LEDs every four times through the fade by
    # checking the modulo of the counter.
    # the modulo function gives you the remainder of
    # the division of two numbers:
    if counter % 4 == 0:
        digital_leds.value = True
    else:
        digital_leds.value = False
```

Downloads

- Datasheet for blue 1206 LED (https://adafru.it/qle)
- Datasheet for warm white LED (https://adafru.it/qlf)
- Datasheet for red 1206 LED (https://adafru.it/qlA)
- Datasheet for green LED (https://adafru.it/rla)
- Datasheet for pink LED (https://adafru.it/rlc)
- EagleCAD PCB files on GitHub (https://adafru.it/rlb)
- Fritzing object in Adafruit Fritzing library (https://adafru.it/aP3)