



Cyberpunk Spikes

Created by Becky Stern



<https://learn.adafruit.com/cyberpunk-spikes>

Last updated on 2024-06-03 01:28:37 PM EDT

Table of Contents

Overview	3
Download and 3D Print	5
Prepare NeoPixel Strip	7
Assemble Circuit	10
Arduino Code	13
CircuitPython Code	16
Wear it!	18

Overview



This guide was written for the Gemma board, but can be done with either the v1, v2 or Gemma M0. We recommend the Gemma M0 as it is easier to use and is more compatible with modern computers!

Before you get started, follow the [Gemma M0 guide \(https://adafru.it/zxE\)](https://adafru.it/zxE) or the [Classic Introducing GEMMA guide \(https://adafru.it/e1V\)](https://adafru.it/e1V)

Make your own flexible, spiky, glowing accessory using NeoPixel strip diffused by NinjaFlex flexible 3D printing filament! Magnets let you attach the spikes to anything in your wardrobe. The soft flexible enclosure holds GEMMA, the tiny microcontroller that animates the LEDs, and a rechargeable lipoly battery.

For this project you will need:

- [Gemma M0 \(http://adafru.it/3501\)](http://adafru.it/3501) or [GEMMA v2 wearable microcontroller \(http://adafru.it/1222\)](http://adafru.it/1138) (the [Trinket M0 \(http://adafru.it/3500\)](http://adafru.it/3500) would also work)
- [60 LED per meter NeoPixel strip \(1 meter\) \(http://adafru.it/1138\)](http://adafru.it/1138)
- [NinjaFlex Snow White flexible 3D printing filament \(http://adafru.it/1691\)](http://adafru.it/1691)
- [slide switch \(http://adafru.it/805\)](http://adafru.it/805)
- [JST extension \(http://adafru.it/1131\)](http://adafru.it/1131)
- [500mAh lipoly battery \(http://adafru.it/1578\)](http://adafru.it/1578)
- [six rare earth magnets \(http://adafru.it/9\)](http://adafru.it/9)
- safety pins or needle and thread
- Permatex 66B silicone adhesive

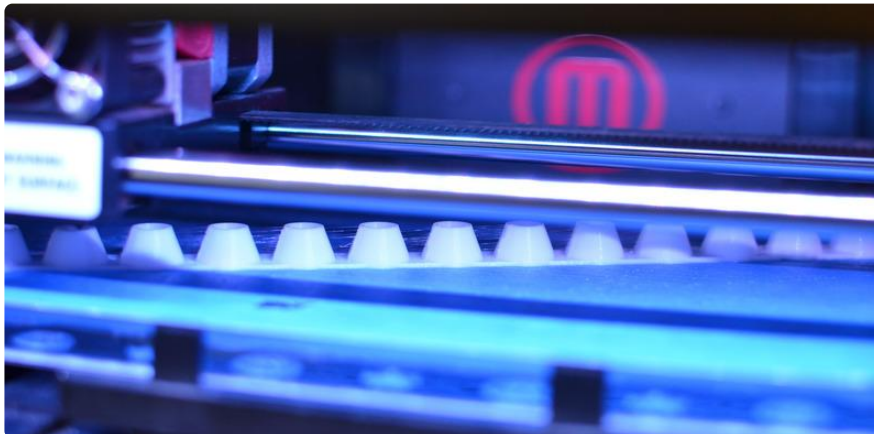


It's easy to wear your spikes around your collar, over your shoulder, in your hair, on the strap of a bag, and even in your hair. How will you wear it? We'd love to see your versions on our [weekly show and tell on Google+](https://adafru.it/showtell) (<https://adafru.it/showtell>)!



Portraits by Andrew Tingle.

Download and 3D Print



We designed two styles of spike strip-- one with regular round spikes and one crystal-inspired statement piece. Download whichever spikes you like and print in NinjaFlex filament at 225 degrees with a non-heated build plate. For more tips on working with NinjaFlex, check out our [NinjaFlex guide \(https://adafru.it/daS\)](https://adafru.it/daS).

[Download 3D spike files on Thingiverse](https://adafru.it/daT)

<https://adafru.it/daT>

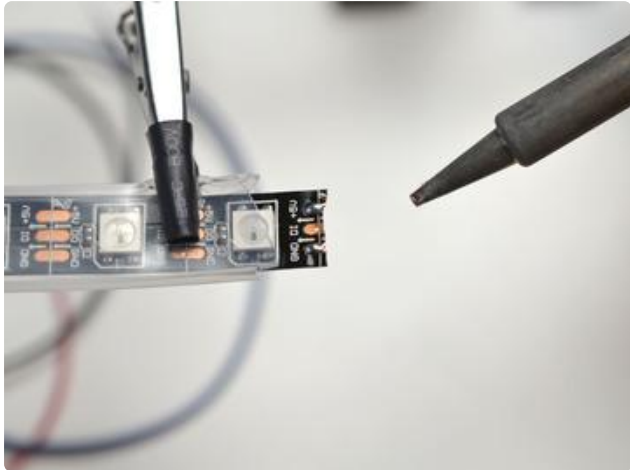


Also print the two pieces of the flexible enclosure for the GEMMA and battery. Since it's printed in NinjaFlex, the enclosure is soft and flexible, yet firm enough to protect your components. The enclosure shape includes tabs for pinning or sewing to your garment.

**Download enclosure files on
Thingiverse**

<https://adafru.it/daU>

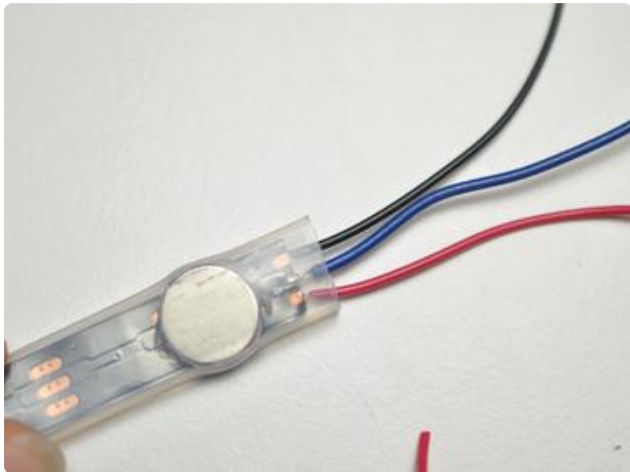
Prepare NeoPixel Strip



Prepare the input end of your NeoPixel strip by tinning the pads with solder. The strip won't work if you solder wires to the wrong end, so be sure the arrows on the PCB point away from the end you're wiring.



Solder three stranded wires, about eight inches long each, to the tinned pads of the NeoPixel strip. To help avoid the solder joints from being too cramped, solder the center pad's wire on the reverse side of the PCB (two on top, one on bottom), as shown.



Wrap three rare earth magnets in tape to prevent short circuits, and slide them into the NeoPixel strip sheathing, on the underside of the PCB. Our spike strip is 16 pixels long, and we used three magnets evenly spaced (one at each end and one in the center).



Prepare a protected work surface in an area with good ventilation.



Use Permatex 66B silicone adhesive to affix the 3D printed spikes to the NeoPixel strip. Apply both to the strip's silicone sheathing and the NinjaFlex strip of spikes, using a toothpick to spread it around if necessary.

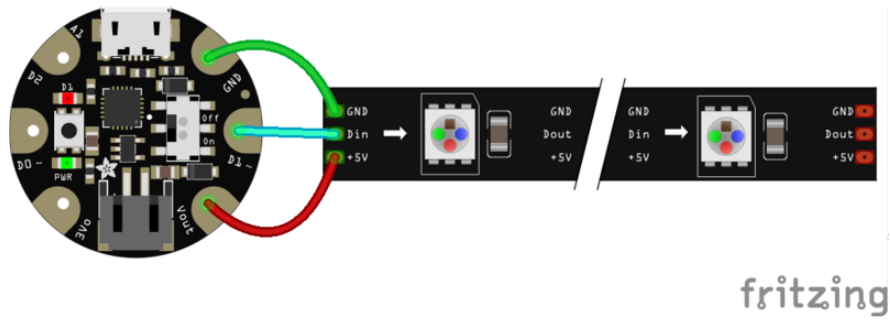
Squish a bit of silicone adhesive into the ends of the NeoPixel strip sheathing to provide water resistance and strain relief.



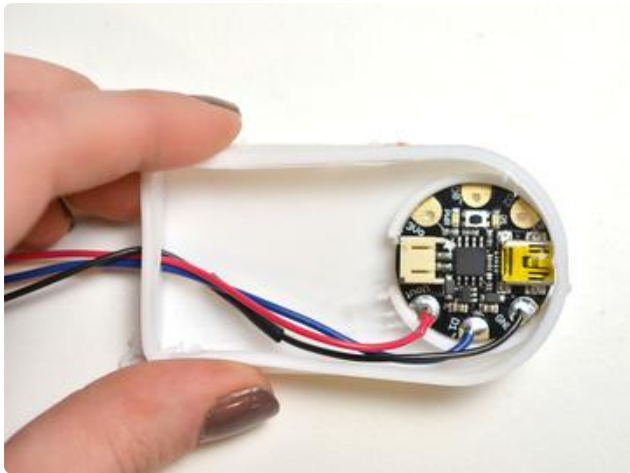
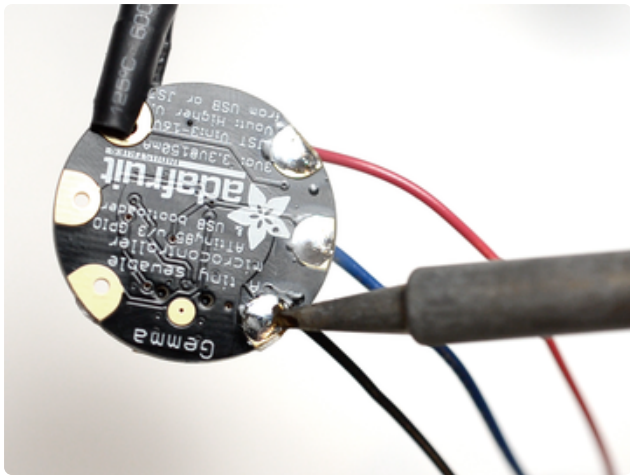
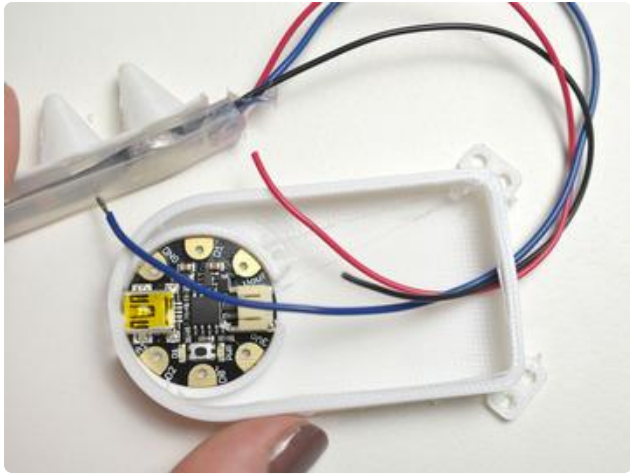
Allow adhesive to dry overnight.



Assemble Circuit



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



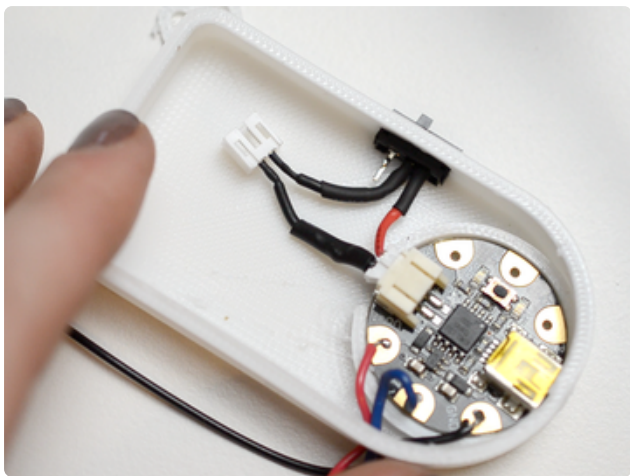
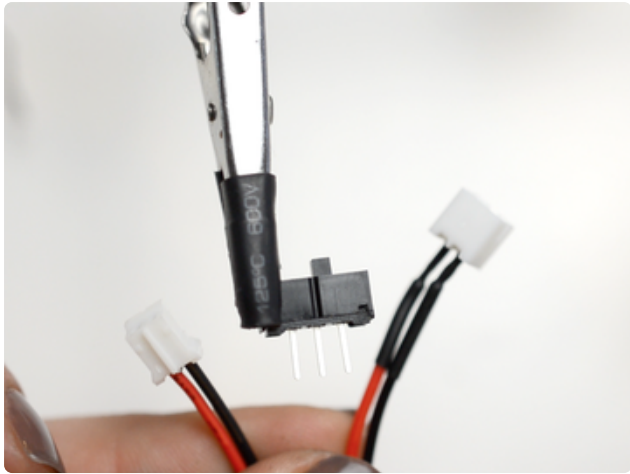
Route your NeoPixel strip's wires through the hole at the top of the enclosure, and solder them up to GEMMA:

NeoPixel GND -> GEMMA GND

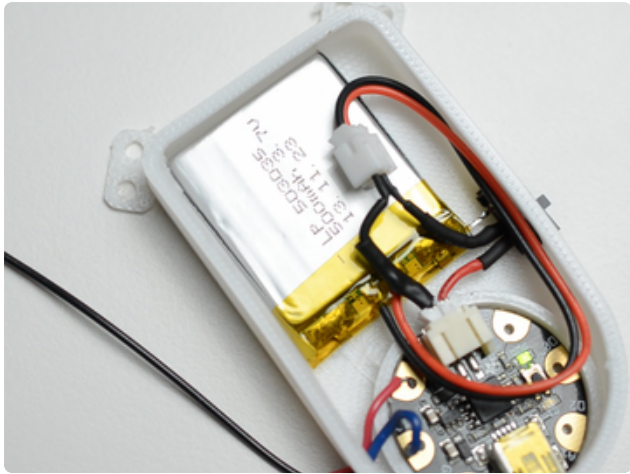
NeoPixel + -> GEMMA Vout

NeoPixel signal -> GEMMA D1

Seat GEMMA into the round outline inside the enclosure, with the USB port facing its opening at the bottom end of the enclosure.



Use a JST extension and slide switch to make this tiny adapter. The slide switch fits into the opening in the enclosure and now you can easily power up your circuit while still making it easy to disconnect the battery for recharging.



Fit everything neatly in the enclosure and press on the lid.

Arduino Code

The Arduino code presented below works equally well on all versions of GEMMA: v1, 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!

[Click to Download the NeoPixel Library](https://adafru.it/cDj)

<https://adafru.it/cDj>

First test your NeoPixel strip using the strandtest sketch after having downloaded the NeoPixel library for your GEMMA-modded Arduino. Did that sound like gibberish to you? Read the [Introducing GEMMA](https://adafru.it/cHH) (<https://adafru.it/cHH>) and [NeoPixel guides](https://adafru.it/cEz) (<https://adafru.it/cEz>).

Installing Arduino libraries is a frequent stumbling block. If this is your first time, or simply needing a refresher, please read the [All About Arduino Libraries](https://adafru.it/aYM) (<https://adafru.it/aYM>) tutorial. (<https://adafru.it/zAX>) If the library is correctly installed (and the Arduino IDE is restarted), you should be able to navigate through the “File” rollover menus as follows:

File→Sketchbook→Libraries→Adafruit_NeoPixel→strandtest

Connect up your NeoPixels in a solderless breadboard and use alligator clips to attach to GEMMA, referring to the circuit diagram if necessary.

You’ll need to change a few lines in the code regarding the data pin (1), type of pixels (RGB vs GRB), and number of pixels (5). The resulting (and slightly simplified) code is below:

```

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

#include <Adafruit_NeoPixel.h>

#define PIN 1

// Parameter 1 = number of pixels in strip
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(16, PIN, NEO_GRB + NEO_KHZ800);

// Here is where you can put in your favorite colors that will appear!
// just add new {nnn, nnn, nnn}, lines. They will be picked out randomly
//           R   G   B
uint8_t myColors[][3] = {{232, 100, 255}, // purple
                        {200, 200, 20},  // yellow
                        {30, 200, 200},   // blue
                        };

// don't edit the line below
#define FAVCOLORS sizeof(myColors) / 3

void setup() {
  strip.begin();
  strip.setBrightness(40);
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  flashRandom(5, 8); // first number is 'wait' delay, shorter num == shorter twinkle
  flashRandom(5, 5); // second number is how many neopixels to simultaneously light
  up
  flashRandom(5, 11);
  colorWipe(strip.Color(232, 100, 255), 50); // Red
  colorWipe(strip.Color(200, 200, 20), 50); // Green
  colorWipe(strip.Color(30, 200, 200), 50); // Blue
  rainbowCycle(20);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {

```

```

uint16_t i, j;

for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
  for(i=0; i< strip.numPixels(); i++) {
    strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
  }
  strip.show();
  delay(wait);
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos < 85) {
    return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
    WheelPos -= 85;
    return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
    WheelPos -= 170;
    return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}

void flashRandom(int wait, uint8_t howmany) {

  for(uint16_t i=0; i<howmany; i++) {
    // pick a random favorite color!
    int c = random(FAV_COLORS);
    int red = myColors[c][0];
    int green = myColors[c][1];
    int blue = myColors[c][2];

    // get a random pixel from the list
    int j = random(strip.numPixels());

    // now we will 'fade' it in 5 steps
    for (int x=0; x < 5; x++) {
      int r = red * (x+1); r /= 5;
      int g = green * (x+1); g /= 5;
      int b = blue * (x+1); b /= 5;

      strip.setPixelColor(j, strip.Color(r, g, b));
      strip.show();
      delay(wait);
    }
    // & fade out in 5 steps
    for (int x=5; x >= 0; x--) {
      int r = red * x; r /= 5;
      int g = green * x; g /= 5;
      int b = blue * x; b /= 5;

      strip.setPixelColor(j, strip.Color(r, g, b));
      strip.show();
      delay(wait);
    }
  }
  // LEDs will be off when done (they are faded to 0)
}

```

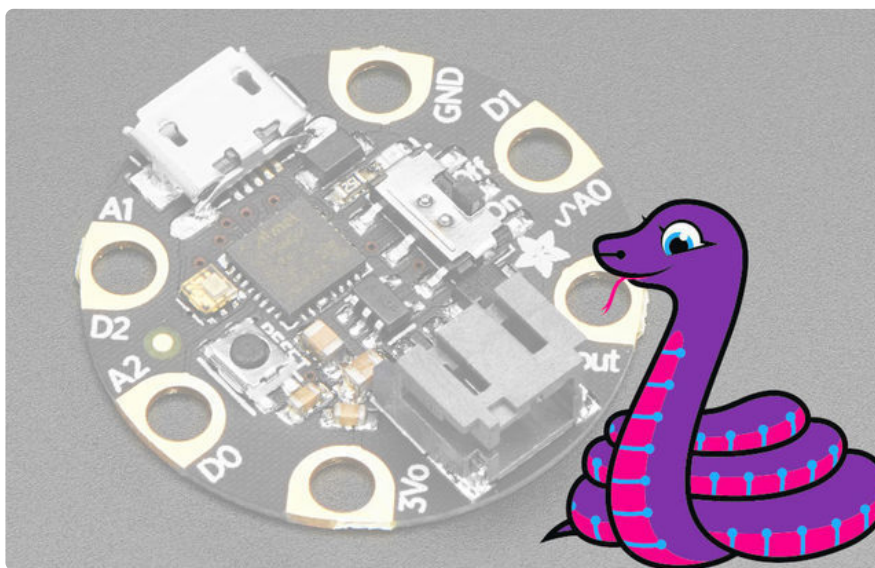
From the **Tools→Board** menu, select the device you are using:

- **Adafruit Gemma M0**
- **Adafruit Gemma 8 MHz**

Connect the USB cable between the computer and your device. The original Gemma (8 MHz) need the reset button pressed on the board, then click the upload button (right arrow icon) in the Arduino IDE. You do not need to press the reset on the newer Gemma M0.

When the battery is connected, you should get a light show from the LEDs. All your pixels working? Great! You can take apart this prototype and get ready to put the pixels in the collar. Refer to the [NeoPixel Uberguide \(https://adafru.it/dhw\)](https://adafru.it/dhw) for more info.

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 \(https://adafru.it/z1B\)](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.

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

import time
from rainbowio import colorwheel
import board
import neopixel
from digitalio import DigitalInOut, Direction

try:
    import urandom as random
except ImportError:
    import random

pixpin = board.D1
numpix = 16

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT

strip = neopixel.NeoPixel(pixpin, numpix, brightness=.2, auto_write=True)

colors = [
    [232, 100, 255], # Purple
    [200, 200, 20], # Yellow
    [30, 200, 200], # Blue
]

# Fill the dots one after the other with a color

def colorWipe(color, wait):
    for j in range(len(strip)):
        strip[j] = (color)
        time.sleep(wait)

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

# Slightly different, this makes the rainbow equally distributed throughout

def rainbow_cycle(wait):
    for j in range(255 * 5):
        for i in range(len(strip)):
            idx = int((i * 256 / len(strip)) + j)
            strip[i] = colorwheel(idx & 255)
            time.sleep(wait)

def flash_random(wait, howmany):
```

```

for _ in range(howmany):

    c = random.randint(0, len(colors) - 1) # Choose random color index
    j = random.randint(0, numpix - 1) # Choose random pixel
    strip[j] = colors[c] # Set pixel to color

    for i in range(1, 5):
        strip.brightness = i / 5.0 # Ramp up brightness
        time.sleep(wait)

    for i in range(5, 0, -1):
        strip.brightness = i / 5.0 # Ramp down brightness
        strip[j] = [0, 0, 0] # Set pixel to 'off'
        time.sleep(wait)

while True:
    # first number is 'wait' delay, shorter num == shorter twinkle
    flash_random(.01, 8)
    # second number is how many neopixels to simultaneously light up
    flash_random(.01, 5)
    flash_random(.01, 11)

    colorWipe((232, 100, 255), .1)
    colorWipe((200, 200, 20), .1)
    colorWipe((30, 200, 200), .1)

    rainbow_cycle(0.05)

```

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 \(https://adafru.it/yew\)](https://adafru.it/yew).

Download neopixel.py from GitHub

<https://adafru.it/yew>

Wear it!



You can use the included holes to stitch or pin the enclosure to your underarm or wherever you'd like to put it. For more permanent use, stitch a pocket for this enclosure on the inside of your garment and route the wires to the inside.



Use a fluffy bun-maker hair accessory and tuck the enclosure under it to wear these spikes around your head!



Epaulets, two styles



Around the collar



Cyber dragon, anyone?

Since you sealed up the strip with adhesive, this accessory is fairly water-resistant. Turn it off and remove the battery if you get stuck in a torrential downpour!