NeoPixel Punk Collar

Created by Becky Stern

https://learn.adafruit.com/neopixel-punk-collar

Last updated on 2021-11-15 06:12:41 PM EST

Table of Contents

Overview

Get your cybergoth on with five color-changing NeoPixels studded onto a leather collar. The tiny GEMMA microcontroller can display endless animations on this fun funky accessory that's easy to make with a little soldering!

GEMMA jewelry! The bitty board fits perfectly in the center of a NeoPixel ring for flashy hoop earrings or a charming pendant. Read on to build your own!

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

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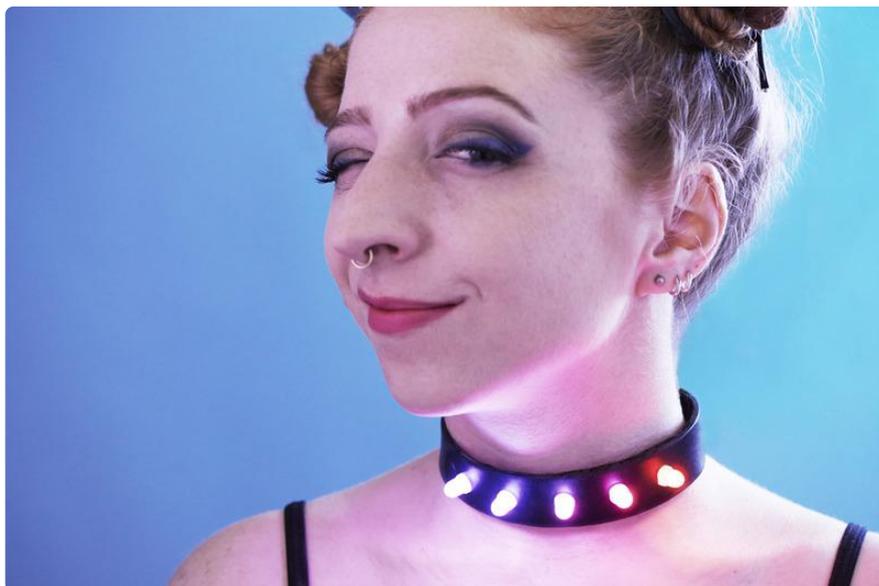

Portrait by Andrew Tingle

You will need:

- Gemma M0 (https://adafru.it/ytb) or Trinket M0 (https://adafru.it/zya)- you can also use a classic Gemma (https://adafru.it/iif)
- through-hole NeoPixels (https://adafru.it/dpE)
- leather collar or cuff
- 150mAh rechargeable lipoly (https://adafru.it/dpF) battery and charger (https://adafru.it/dpG)
- solderless breadboard and alligator clips for prototyping

- soldering iron and solder
- solid core or stranded wire (20 to 26 gauge)
- helping third hand tool
- pliers
- wire strippers
- flush diagonal cutters
- awl or other pointy tool to pierce leather
- sharp utility knife
- cutting mat
- ruler
- marking pen/pencil
- gaffer tape or other material to protect battery
- velcro tape to secure battery to collar

Circuit Diagram

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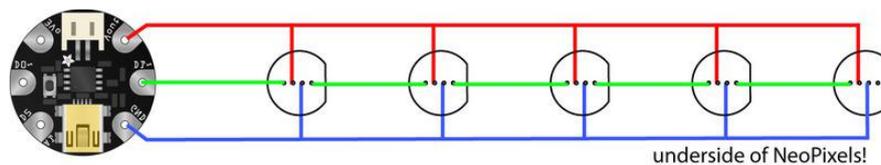
> This diagram uses the original Gemma but you can also use the Gemma M0 with the exact same wiring!

The GEMMA and battery live on the outside of the collar, and the NeoPixels pierce through the collar to be wired on the inside (hence why GEMMA is mirrored in the diagram). Connect up all pixels power pins to GEMMA's Vout, ground to GND, and the first data input to GEMMA D1. The data out from each pixel is wired to the data in on the next.

You could easily add five more pixels for a total of ten for a more intensely studded look!

Prototype Circuit
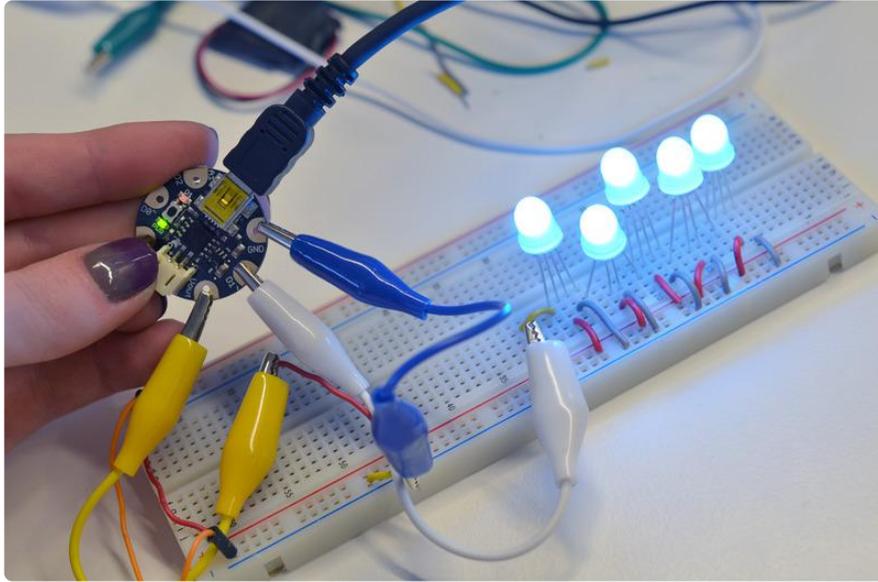
> **This step is not optional! Avoid heartbreak later by testing your circuit now.**

If this is your first time using GEMMA, work through the Gemma M0 guide (https://adafru.it/zxE) or the Classic Introducing GEMMA guide (https://adafru.it/e1V).

We recommend using the Gemma M0 (https://adafru.it/ytb) and following the CircuitPython Code (https://adafru.it/zyb) page for downloading the NeoPixel Library and NeoPixel_Punk_Collar using the flash drive feature.

The Arduino IDE can be used with the GEMMA M0 or older versions of the GEMMA. See the Arduino Code page (https://adafru.it/zyc) for this project for details. Once you have it up and running (test the 'blink' sketch), then download and install the NeoPixel library.

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

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) tutorial. (https://adafru.it/zyd)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:

```
#include <Adafruit_NeoPixel.h>

#define PIN 1

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino 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(5, PIN, NEO_RGB + NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, avoid connecting
// on a live circuit... if you must, connect GND first. Minimize
// distance between Arduino and first pixel.

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

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

  rainbow(20);
  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);
    }
  }
```

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

- Adafruit Gemma M0
- Adafruit Trinket M0
- Adafruit Gemma 8 MHz
- Adafruit Trinket 8 MHz

Connect the USB cable between the computer and your device. The original Gemma (8 MHz) and Trinket (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 or Trinket 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) for more info.

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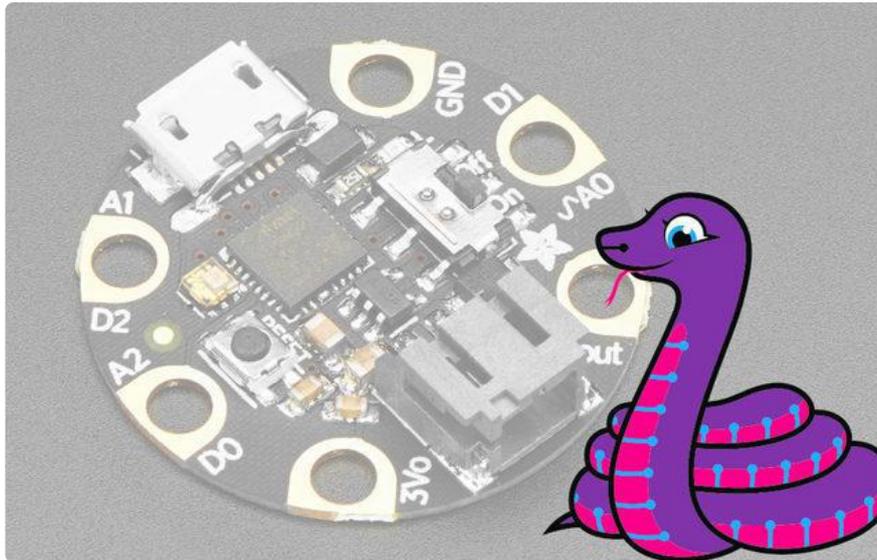
GEMMA M0 boards can run CircuitPython — a different approach to programming compared to Arduino sketches. In fact, CircuitPython comes factory pre-loaded on GEMMA M0. If you've overwritten it with an Arduino sketch, or just want to learn the basics of setting up and using CircuitPython, this is explained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

> 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.

Download NeoPixel_Punk_Collar.py

https://adafru.it/zye

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

pixpin = board.D1
numpix = 5

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

strip = neopixel.NeoPixel(pixpin, numpix, brightness=1, auto_write=True)


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


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


while True:
    colorWipe((255, 0, 0), .1)  # red and delay
    colorWipe((0, 255, 0), .1)  # green and delay
    colorWipe((0, 0, 255), .1)  # blue and delay

    rainbow(0.05)
    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).

Download neopixel.py from Github

https://adafru.it/yew

Build Collar

Try on your collar and determine where it will buckle. Take it off and re-buckle, to get an idea of where the center front is. Mark this spot with a pen or pencil.

Lay the collar flat and mark two more spots at one-inch increments on either side of the center mark.

Use a ruler to mark the top-bottom center line on the collar as well.

Use a utility knife to cut small horizontal slits in the collar at the marks. They should be just long enough for the legs of the NeoPixels to pass through, but not so long as they show on the front.

Pierce your NeoPixels through the collar with the flat sides all facing the same way (away from where the GEMMA will go).
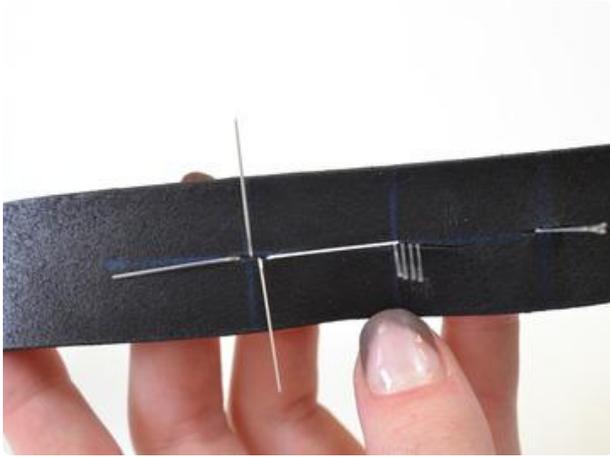
On the inside of the collar, bend the legs as shown-- power up, ground down, data to either side! Repeat with all pixels.

Bend the power and ground leads towards each other with a pair of pliers, and set the collar up, curved, in a set of helping hands. Trim the data leads to just overlap, but not get in the way of other legs.

Solder overlapping wire leads while the collar is curved, using a pair of pliers to hold them close where necessary.

Use small pieces of wire to make a continuous power bus, and do the same with ground.


Use an awl to mark where wires will pass through the collar from GEMMA to the pixels.

Pierce these marked spots with the awl. Prepare three pieces of wire by stripping one end, then thread them through the holes in the leather.

Pass the stripped ends of wire through GEMMAs pads marked Vout, D1, and GND, referring to the circuit diagram, and solder in place. Use flush snips to trim off excess wire.
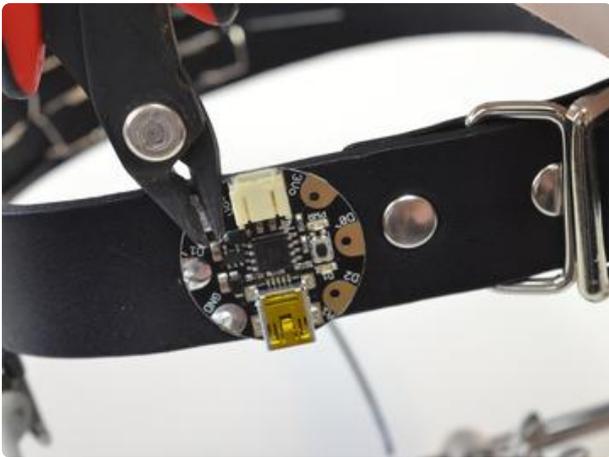
On the inside of the collar, trim and strip the other ends of the wire, then connect them up to the power bus, data in, and ground bus respectively (refer to circuit diagram). Wrap the wires around the pixel leads so they stay in place for soldering. Solder these connections and trim off any excess.

Plug your collar in with a USB cable and see if it's working! It should still have the program on it from testing earlier, and all five pixels should glow and animate. If they don't, use a multimeter to check your connections, and double check none of your pixels are in backwards!

Once you know your circuit is solid, you can protect your delicate neck skin by gluing on a piece of fuzzy fabric (we used microsuede, a durable fuzzy microfiber) with E6000 craft adhesive. Painter's tape can come in handy for clamping over the edge. Allow to dry for 24 hours.
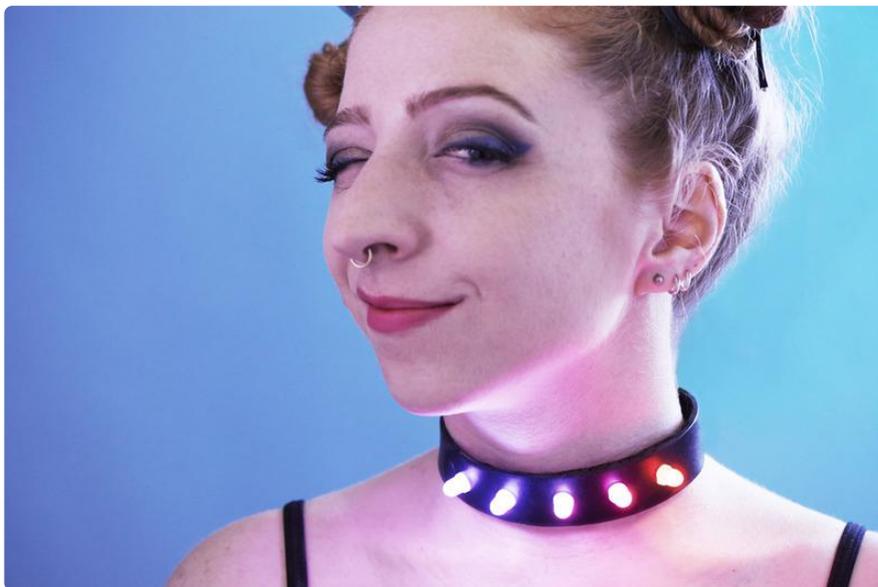
Wrap your battery in gaffer tape to protect it and provide strain relief to its wires.

Use a piece of Velcro tape to affix the battery to the collar right next to the GEMMA. We arranged GEMMA and the battery so the collar slack covers/protects them. The battery wire wraps around the collar once, then plugs into GEMMA. This is how you turn the collar off/on.

Wear it!

Here's a modified version of PhilB's larson scanner code, which chases back and forth across the collar. Use it as a jumping-off point, then code up your own crazy animations!

While the collar is pretty durable, use caution in heavy rainstorms or really sweaty dance parties-- remove and power down the collar if the circuit is going to get wet. Store your collar in the round, and don't shove it in your bag or it might get twisted or crushed, which could break the circuit.

```
#include &lt;Adafruit_NeoPixel.h&gt;

#define N_LEDS 5
#define PIN     1

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, PIN, NEO_RGB + NEO_KHZ800);

void setup() {
  strip.begin();
}

int pos = 0, dir = 1; // Position, direction of "eye"

void loop() {
  int j;

  // Draw 5 pixels centered on pos.  setPixelColor() will clip any
  // pixels off the ends of the strip, we don't need to watch for that.
  strip.setPixelColor(pos - 1, 0x100000); // Dark red
  strip.setPixelColor(pos    , 0xFF3000); // Center pixel is brightest
  strip.setPixelColor(pos + 1, 0x100000); // Dark red

  strip.show();
  delay(70);

  // Rather than being sneaky and erasing just the tail pixel,
  // it's easier to erase it all and draw a new one next time.
  for(j=-2; j&lt;= 2; j++) strip.setPixelColor(pos+j, 0);

  // Bounce off ends of strip
  pos += dir;
  if(pos &lt; 0) {
    pos = 1;
    dir = -dir;
  } else if(pos &gt;= strip.numPixels()) {
    pos = strip.numPixels() - 2;
    dir = -dir;
  }
}
```