Mystical LED Halloween Hood

Created by Abigail Torres
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>5</td>
</tr>
<tr>
<td>NeoPixel GEMMA circuit</td>
<td>6</td>
</tr>
<tr>
<td>Arduino Code</td>
<td>9</td>
</tr>
<tr>
<td>• NeoPixel Überguide: Arduino Library Installation</td>
<td></td>
</tr>
<tr>
<td>CircuitPython Code</td>
<td>11</td>
</tr>
<tr>
<td>Sewing Pattern</td>
<td>13</td>
</tr>
<tr>
<td>3D Printed Battery Pocket</td>
<td>17</td>
</tr>
<tr>
<td>• TPE Flexible Filament</td>
<td></td>
</tr>
<tr>
<td>Final Assembly</td>
<td>18</td>
</tr>
<tr>
<td>Wear it!</td>
<td>19</td>
</tr>
</tbody>
</table>
Overview

Create glowing eyes for your costume! This project is ideal for any character with glowing eyes, like the Black Mage from Final Fantasy, Jawa from Star Wars, or Orko from He-Man. Two NeoPixel Jewels can appear any color or animating pattern, and they are driven by a GEMMA microcontroller powered by a 500mAh lipoly battery in a 3D printed pocket.

Before you begin, check out these prerequisite guides:

- Introducing GEMMA () or Introducing GEMMA M0 ()
- NeoPixel Uberguide ()
- Adafruit's Guide to Excellent Soldering ()
- Let's put LEDs in Things ()

For this project, you will need:

- Two NeoPixel Jewels ()
- Adafruit GEMMA () or GEMMA M0 () microcontroller
- Silicone coated stranded wire (30awg recommended) ()
- Soldering tools and supplies
- Sewing pins
- Tailor's chalk

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!
• 19awg galvanized steel wire
• Coat fabric for hood/cape
• Translucent fabric for face panel
• Sewing machine
• Scissors
• Needle () and thread
• SemiFlex 3D printing filament ()

NeoPixel GEMMA circuit

This diagram uses the original Gemma but you can also use the Gemma M0 with the exact same wiring!
Connections are as follows:

- GEMMA D1 to 1st NeoPixel Jewel IN
- 1st NeoPixel Jewel OUT to 2nd NeoPixel Jewel IN
- GEMMA Vout to 1st and 2nd NeoPixel Jewel PWR
- GEMMA GND to 2st and 2nd NeoPixel Jewel GND

Strip, tin, and solder wires to the NeoPixel Jewels according to the circuit diagram. It's easiest to insert the wires from the front of the PCB and solder on the back, where there aren't any other components to bump into.
Be sure to leave enough slack between jewels to space your eyes out in your costume! Extra slack is ok, you can always tack it down with a needle and thread.

Solder long wires (at least 18 inches) between GEMMA and the 1st NeoPixel jewel according to the circuit diagram.
You can optionally diffuse the LEDs by covering them in white paper.

---

**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!

If this is your first time using GEMMA, work through the Introducing GEMMA or Gemma M0 starter guide first; you need to customize some settings in the Arduino IDE. Once you have it up and running (test the ‘blink’ sketch), then follow the instructions on the following page for installing the NeoPixel library:
Plug in your circuit via USB and test that all LEDs are functioning properly with the NeoPixel example sketch 'strandtest.' Please refer to the Gemma starter guide and the NeoPixel Überguide if you haven't before.

Once you've verified your wiring is correct, load your desired color code or the sketch below that gently pulses the LEDs red.

```c
#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(14, PIN, NEO_GRB + NEO_KHZ800);

int alpha; // Current value of the pixels
int dir = 1; // Direction of the pixels... 1 = getting brighter, 0 = getting dimmer
int flip; // Randomly flip the direction every once in a while
int minAlpha = 25; // Min value of brightness
int maxAlpha = 100; // Max value of brightness
int alphaDelta = 5; // Delta of brightness between times through the loop

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

void loop() {
  flip = random(32);
  if(flip > 20) {
    dir = 1 - dir;
  }
  // Some example procedures showing how to display to the pixels:
  if (dir == 1) {
    alpha += alphaDelta;
  }
  if (dir == 0) {
    alpha -= alphaDelta;
  }
  if (alpha < minAlpha) {
    alpha = minAlpha;
    dir = 1;
  }
  if (alpha > maxAlpha) {
    alpha = maxAlpha;
    dir = 0;
  }
}
```

© Adafruit Industries
// Change the line below to alter the color of the lights
// The numbers represent the Red, Green, and Blue values
// of the lights, as a value between 0(off) and 1(max brightness)
//
// EX:
// colorSet(strip.Color(alpha, 0, alpha/2)); // Pink
//colorSet(strip.Color(0, 0, alpha)); // Blue
//colorSet(strip.Color(alpha, alpha/2, 0)); // Yellow
colorSet(strip.Color(alpha, 0, 0)); // Red

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

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 to 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.

```python
# SPDX-FileCopyrightText: 2017 Phillip Burgess for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import board
import neopixel

try:
    import urandom as random  # for v1.0 API support
except ImportError:
    import random

numpix = 17  # Number of NeoPixels
pixpin = board.D1  # Pin where NeoPixels are connected
strip = neopixel.NeoPixel(pixpin, numpix)

minAlpha = 0.1  # Minimum brightness
maxAlpha = 0.4  # Maximum brightness
alpha = (minAlpha + maxAlpha) / 2  # Start in middle
alphaDelta = 0.008  # Amount to change brightness each time through loop
alphaUp = True  # If True, brightness increasing, else decreasing

strip.fill([255, 0, 0])  # Fill red, or change to R,G,B of your liking

while True:  # Loop forever...
    if random.randint(1, 5) == 5:  # 1-in-5 random chance
        alphaUp = not alphaUp  # of reversing direction
    if alphaUp:  # Increasing brightness?
        alpha += alphaDelta  # Add some amount
    if alpha >= maxAlpha:  # At or above max?
        alpha = maxAlpha  # Limit to max
        alphaUp = False  # and switch direction
    else:  # Else decreasing brightness
        alpha -= alphaDelta  # Subtract some amount
    if alpha <= minAlpha:  # At or below min?
        alpha = minAlpha  # Limit to min
        alphaUp = True  # and switch direction

    strip.brightness = alpha  # Set brightness to 0.0 to 1.0
    strip.write()  # and issue data to LED strip
```

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://github.com/adafruit/neopixel).
Download and print our hood/cape pattern, then tile and tape the pieces together, then cut out the pattern pieces and arrange them as shown on a folded piece of thick coat fabric (fold is along top edge in photo)
Trace around the edge of the pattern with tailor's chalk, then also trace a sewm allowance outside that line 1/4" to 5/8" or more depending on your preference.
Cut out pattern pieces and flip them over. Unpin the paper pattern and flip it over too, using it to trace its shape onto the other side of the folded fabric (so each fabric piece has a full pattern perimeter traced). This makes it easier to sew in the right place!

Pin shoulder seams and darts together and stitch, removing pins as you go.

Pin the hood and cape pieces together along the neckline, starting with the center back crease. Machine stitch along the seam and remove pins as you go.
To finish the hood, fold with top edges aligned, right sides together, and stitch the seam.

Cut out a round piece of translucent fabric to use for the face panel, roughly the same size as the hood opening.
3D Printed Battery Pocket

Print a sew-on pocket for your lipoly battery! The pocket protects the battery from abuse and also makes it easy to remove the battery for charging. It's not strictly necessary, though, but bare lipoly batteries can be risky, so unless you protect it in some way, we recommend using a hard shell alkaline pack like the 3xAAA holder (https://adafruit.it/727).

TPE Flexible Filament

The battery pocket works best when printed in flexible material like Ninjaflex or Semiflex. This material requires a direct-drive extruder system and can be challenging to print. We recommend Semiflex because it handles overhangs better than Ninjaflex, and has a shell hardness (98A). Follow the print settings below for best results.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Printing speed</td>
<td>30mm/sec</td>
</tr>
<tr>
<td>Retraction</td>
<td>OFF</td>
</tr>
<tr>
<td>Raft / Support Material</td>
<td>OFF</td>
</tr>
<tr>
<td>Extruder Temperature</td>
<td>220-230c</td>
</tr>
</tbody>
</table>
The part should be centered on the print bed and ready to print "as-is". We recommend using CURA, or Simplify3D to slice the file.

Final Assembly

Fold over a 1/2" seam along the front edge of the hood and thread in a long piece of galvanized wire (we're using 19awg). Twist the ends and stitch them down to anchor the wire. This wire gives structure to the hood and helps support the weight of the face panel.

Try on the hood and decide where you want the face panel to be, then pin and sew it in place by hand with a few tack stitches, or run it through the sewing machine.
Once the face panel is in place, try the hood on again to find the placement of the eyes. Hand stitch them in place using the mounting holes on the PCB.

Stitch through unused holes on GEMMA to affix it to the inside of the cape's lapel. Stitch the battery holder near GEMMA and insert the 500mAh rechargable lipoly battery.

Wear it!

Enjoy your spooky new costume! We hope this serves as a jumping off point for your own project. Dress it up, dress it down, change the fabric and LED color-- the possibilities are numerous!

It's pretty easy to see out of the translucent face panel, and to remove the hood for eating/drinking.
If you need your circuit to be water-resistant, check out our video on ruggedizing your projects: