adafruit learning system

Neopixel Jewel 10 Minute Necklace

Created by Erin St Blaine

Last updated on 2018-08-22 04:00:23 PM UTC

Guide Contents

Introduction

Show off your style and make any outfit shine with this simple neopixel jewel necklace. Customize the code and design your own light effects, and change it up as often as you like to match different outfits or environments.

This is a wonderful intro project for learning to make things with neopixels and microcontrollers. Get your kids involved -- they'll love showing off their glowing handmade necklace.

Materials

Find a pendant with a translucent or transparent center. The pendant should be at least 2 inches across. You can add your own diffusion material or find a pendant that works as-is.

I found this "black lace" pendant at Michael's craft store a few years ago and it's just perfect for this project.

If you don't have anything appropriate on-hand, you can always assemble the light module and take it with you to the craft or bead store. They will probably wonder what you're up to, but I imagine that if you're reading this guide, you're already used to that.

Remember to get a necklace cord as well if your pendant doesn't come with one.

> 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!

Electronics

1 x LiPoly Battery 100mAh
Small LiPoly battery
ADD TO CART

1 x Adafruit Gemma M0
Gemma M0 - Miniature Wearable Electronic Platform
ADD TO CART

1 x Neopixel Jewel
7 Shiny Neopixels
ADD TO CART

1 x Hook Up Wire
Solid Core Wire in 3 Colors
ADD TO CART

1 x Battery Charger
Maybe get two of these and keep one in the car.
ADD TO CART

Tools

- A good soldering iron (https://adafru.it/ide)
- Solder (https://adafru.it/dzp)
- Wire strippers (https://adafru.it/rf1)
- Flush Wire cutters (https://adafru.it/dxQ)
- Hot glue gun

If you're just starting out and ready to get serious, or if you're tired of using sub-par tools you inherited from your uncle, treat yourself to Lady Ada's Electronics Toolkit (https://adafru.it/fE3). It's got all the essentials for getting started with electronics.

A good tool is worth its weight in gold and will save you hours of frustration in the long run.

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!

Software Setup

If this is your first time using Arduino and Gemma, take a look at Adafruit Gemma M0 (https://adafru.it/BeC) or Introducing Gemma (https://adafru.it/e1V) to get a guided tour. You'll also want to be sure the Adafruit_NeoPixel library is installed in Arduino:

(`Sketch > Include Library > Manage Libraries...`)

In the demo video, we're using the Strandtest code that comes with the Neopixel library. Strandtest is great to start with, but it's a little bit "loud" for a necklace.

Here is some slightly quieter code that displays a gorgeous rainbow effect with three different modes for variations in speed and interest.

Go to your Tools menu and select Gemma from the list of boards. Plug your Gemma into your computer via the onboard USB port. Press the "reset" button on your Gemma and wait for the blinky red light, then click the upload button in Arduino.

```
#include <Adafruit_NeoPixel.h>

#define PIN 1

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(40, PIN);

uint8_t  mode   = 0, // Current animation effect
         offset = 0;
uint32_t color  = 0X00A4B3; // Starting color
uint32_t prevTime;

void setup() {
  pixels.begin();
  pixels.setBrightness(40); // 1/3 brightness
  prevTime = millis();
}

void loop() {
  uint8_t  i;
  uint32_t t;

  switch(mode) {

   case 0: //rainbow hold
    rainbowHold(20);
    delay(500);
    break;

   case 1: //rainbow cycle slow
    rainbowCycleslow(20);
    delay(50);
```

```
    delay(50);
    break;

   case 2: //rainbow cycle fast
    rainbowCycle(5);
    delay(50);
    break;
  }

  t = millis();
  if((t - prevTime) > 8000) {      // Every 8 seconds...
    mode++;                        // Next mode
    if(mode > 3) {                 // End of modes?
      mode = 0;                    // Start modes over
      color >>= 8;                 // Next color R->G->B
      if(!color) color = 0xB300A4; // Reset color
    }
    for(i=0; i<32; i++) pixels.setPixelColor(i, 0);
    prevTime = t;

  }


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

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

void rainbowCycle(uint8_t wait) {
  uint16_t r, j;

  for(j=0; j<256*6; j++) { // 6 cycles of all colors on wheel
    for(r=0; r< pixels.numPixels(); r++) {
      pixels.setPixelColor(r, Wheel(((r * 256 / pixels.numPixels()) + j) & 255));
    }
    pixels.show();
    delay(wait);
  }
}
void rainbowCycleslow(uint8_t wait) {
  uint16_t r, j;

  for(j=0; j<256*3; j++) { // 3 cycles of all colors on wheel
    for(r=0; r< pixels.numPixels(); r++) {
      pixels.setPixelColor(r, Wheel(((r * 256 / pixels.numPixels()) + j) & 255));
    }
    pixels.show();
    delay(wait);
  }
}
void rainbowHold(uint8_t wait) {
  uint16_t r, j;
```

```
  for(j=0; j<256*1; j++) { // 3 cycles of all colors on wheel
    for(r=0; r< pixels.numPixels(); r++) {
      pixels.setPixelColor(r, Wheel(((r * 256 / pixels.numPixels()) + j) & 255));
    }
    pixels.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 pixels.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos < 170) {
   WheelPos -= 85;
   return pixels.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return pixels.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Taking a look at the code, you'll see that there are a few

variables at the top you can change.

Adafruit_NeoPixel pixels = Adafruit_NeoPixel(40, PIN);

Change this number to make the rainbow appear more or less spread out. We really only have 7 pixels, but to slow the animation down I've told the Gemma we have 40.

pixels.setBrightness(40);

Set the brightness here. Max is 255, but that will blind people. 40 is just bright enough that people will notice both you AND your necklace.

rainbowHold(20);
delay(500);

Play with these numbers to speed up or slow down the animation, or to keep one animation running longer.

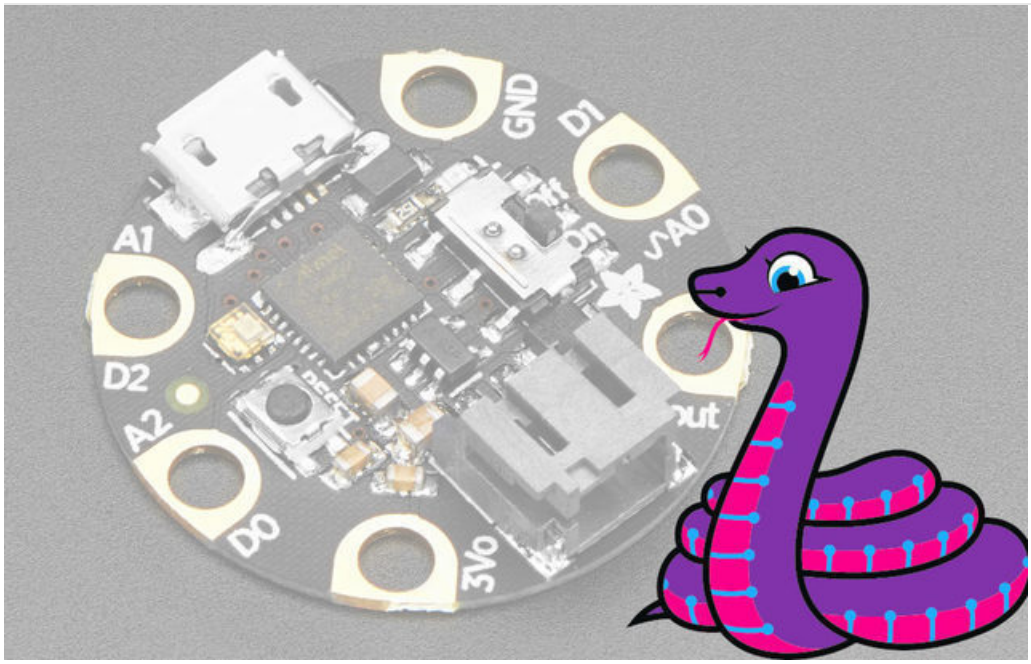
If you encounter trouble...

Any time you hit a roadblock with a neopixel project, we'll usually ask that you start with the "strandtest" example from our own Adafruit_NeoPixel library.

You'll find the strandtest example under **File→Sketchbook→Libraries→Adafruit_NeoPixel→strandtest**

If strandtest fails to run, this suggests a hardware issue...for example, connecting to the wrong Gemma pin. Make sure you've wired your neopixels to pin 1, and that they're wired to the "in" rather than the "out" pin on the Jewel.

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.

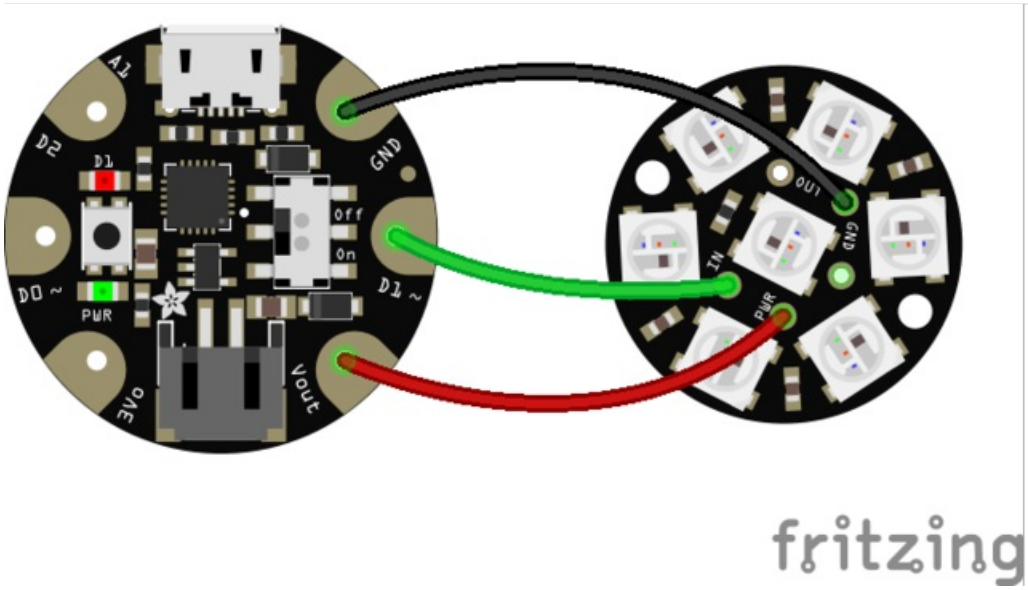
```
import time

import board
import neopixel

pixpin = board.D1
numpix = 7

pixels = neopixel.NeoPixel(pixpin, numpix, brightness=.3, auto_write=False)

rgb_colors = ([179, 0, 0],
              [0, 179, 0],
```

```
              [0, 0, 0])

rgb_idx = 0  # index counter - primary color we are on
color = (0, 164, 179)  # Starting color
mode = 0  # Current animation effect
offset = 0
prevtime = 0


def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if (pos < 0) or (pos > 255):
        return (0, 0, 0)
    if pos < 85:
        return (int(pos * 3), int(255 - (pos * 3)), 0)
    elif pos < 170:
        pos -= 85
        return (int(255 - pos * 3), 0, int(pos * 3))

    pos -= 170
    return (0, int(pos * 3), int(255 - pos * 3))


def rainbow_cycle(wait):
    for j in range(255 * 6):  # 6 cycles of all colors on wheel
        for r in range(len(pixels)):
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = wheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow(wait):
    for j in range(255):
        for index in range(len(pixels)):
            idx = int(index + j)
            pixels[index] = wheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow_cycle_slow(wait):
    for j in range(255 * 3):  # 3 cycles of all colors on wheel
        for r in range(len(pixels)):
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = wheel(idx & 255)
        pixels.write()
        time.sleep(wait)


def rainbow_hold(wait):
    for j in range(255 * 1):  # 3 cycles of all colors on wheel
        for r in range(len(pixels)):
            idx = int((r * 255 / len(pixels)) + j)
            pixels[r] = wheel(idx & 255)
    pixels.write()
    time.sleep(wait)
```

```
while True:

    if mode == 0:  # rainbow hold
        rainbow_hold(0.02)
        time.sleep(.5)

    elif mode == 1:  # rainbow cycle slow
        rainbow_cycle_slow(0.02)
        time.sleep(0.05)

    elif mode == 2:  # rainbow cycle fast
        rainbow_cycle(0.005)
        time.sleep(0.050)

    t = time.monotonic()

    if (t - prevtime) > 8:  # Every 8 seconds...
        mode += 1  # Next mode
        if mode > 2:  # End of modes?
            mode = 0  # Start modes over

        if rgb_idx > 2:  # reset R-->G-->B rotation
            rgb_idx = 0

        color = rgb_colors[rgb_idx]  # next color assignment
        rgb_idx += 1

        for i in range(numpix):
            pixels[i] = (0, 0, 0)

        prevtime = t
```

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).

Assembly



Assembly is very straightforward:

- Gemma **VBAT/VOUT** to Neopixel **PWR** +
- Gemma **GND** to Neopixel **GND**
- Gemma **D1** to Neopixel **IN**

Plug in your battery and glue everything into place, and you're done!

See the video at the beginning of this guide for a detailed time lapse of the whole assembly process.