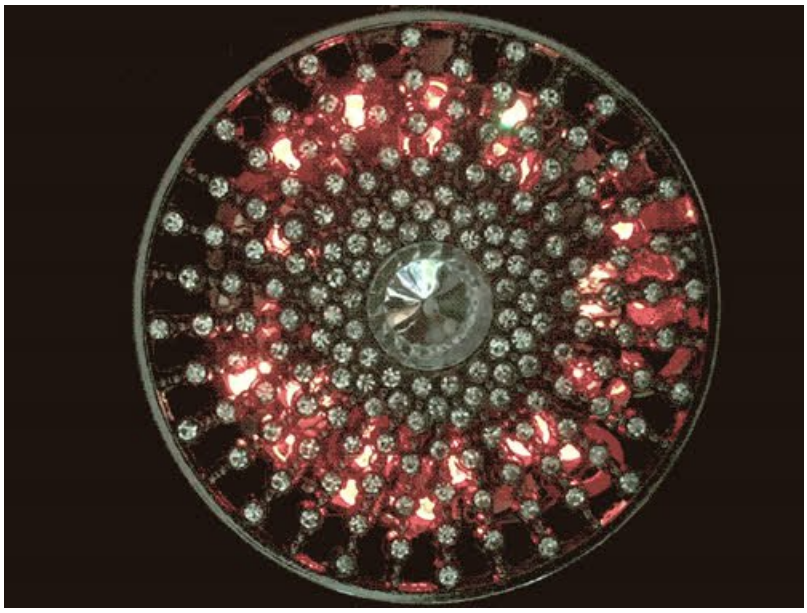
adafruit learning system

Gemma Color Touch Pendant Necklace

Created by Erin St Blaine



Last updated on 2018-10-11 10:11:41 PM UTC

Guide Contents

Guide Contents 2
Overview 3
- Parts 3
- Materials 4

Wiring Diagram 5
- Wiring Connections 5

Software 7
- 1. Arduino IDE 7
- 2. Board Support 7
- 3. Libraries 7
- Upload the Code 7

Assembly 10
Calibration 15
- Capacitive Touch Sensitivity 15
- Color Calibration 16

Overview

Make a stunning light up necklace that goes with anything in your closet. It twinkles softly, showing a beautiful color gradient animation. Touch the face of the pendant to make the necklace change color. It's easy to customize the code to include as many different color combinations as you'd like.

Gemma M0's onboard capacitive touch pad combined with a solid metal pendant face make this magic happen. We've added two different sized NeoPixel rings for 28 beautiful pixels of light.

Parts

1 x Gemma M0

Gemma M0 microcontroller

ADD TO CART

1 x NeoPixel Ring (12 pixels)

12 NeoPixel Ring

ADD TO CART

1 x NeoPixel Ring (16 pixels)

16 NeoPixel Ring

ADD TO CART

1 x Battery

150 mAh Lipoly Battery

ADD TO CART

1 x Charger

USB Battery Charger

ADD TO CART

3 x Stranded Wire

Wire in 3 Colors

ADD TO CART

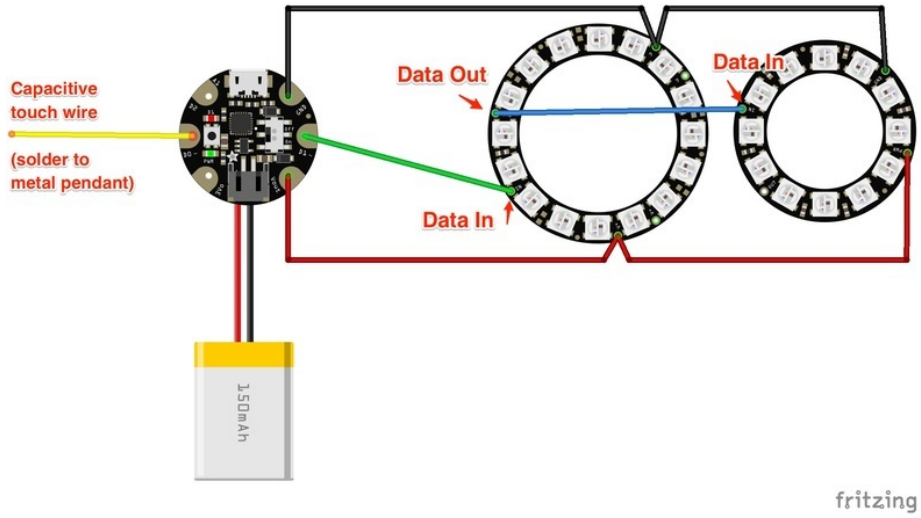
Materials

In addition to the products above, you'll need:

- A solid, conductive, non-coated metal pendant, around 2" or larger in diameter, with cutouts that allow the light through. I used this one from hildie & jo (https://adafru.it/CKD)
- Soldering iron & solder
- E6000 glue or epoxy
- A necklace cord or chain

A multimeter (https://adafru.it/CKE) can also be handy to be sure your pendant conducts electricity. Bring it to the craft store with you and test all the pendants you can find.

Wiring Diagram



We'll stack the two NeoPixel rings with the smaller ring on top, and nestle the battery inside the larger ring. The gemma will sit on the back of the assembly, so the on/off switch, USB port and battery port are accessible.

Wiring Connections

The Assembly page will show step-by-step wiring. Here is a table of the connections that will be made:

Gemma Gnd	Neopixel 16 G	Neopixel 12 G
Gemma Vout	Neopixel 16 5v	Neopixel 12 5v
Gemma D1	Neopixel 16 IN	
	Neopixel 16 OUT	Neopixel 12 IN
Gemma D0	back of metal pendant	

For the capacitive touch pad to work when it's connected to your pendant, you'll need to be sure to use a pendant that conducts electricity. Use your multimeter to test and be sure your pendant is conductive.

If your pendant doesn't conduct electricity you can still make this work. Instead of touching the face of the necklace to change colors, you'll just need to flip it over and touch the gemma's capacitive touch pad directly.

Software

It's a great idea to get your software all set up and loaded onto your board right away, to make testing your connections easier later on.

To get the code running you'll need:

1. Arduino IDE (1.8 or newer preferred)
2. Adafruit Board support for Gemma M0
3. Arduino libraries: FastLED, Adafruit_FreeTouch, Adafruit_NeoPixel

1. Arduino IDE

If you're not using a recent version of the Arduino IDE (1.8.5 or newer), this would be a good time to upgrade (https://adafru.it/fvm). If this is your first time using Arduino, head over to this guide to get it installed (https://adafru.it/jDQ). It's free and fairly simple to get set up.

2. Board Support

You'll need to tell the Arduino IDE which board you're using. This takes just a few minutes of setup, and you'll only need to do it once.

Here is a step-by-step tutorial for setting up Gemma M0 (https://adafru.it/Cib)

3. Libraries

All three libraries can be installed using the Arduino Library Manager — use **Sketch → Include Library → Manage Libraries...** and search for all or part of the library's name, then click "Install."

Look for:

- `FastLED`
- `Adafruit_FreeTouch`
- `Adafruit_NeoPixel`

`Adafruit_NeoPixel` isn't absolutely required for this project, but it's handy to have installed in case you have problems with FastLED. Troubleshooting the basics is a little easier with `Adafruit_NeoPixel`.

Upload the Code

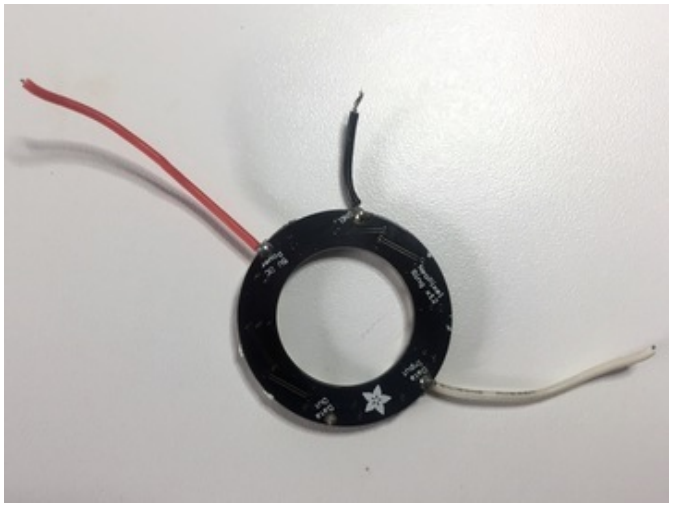
Plug your microcontroller into your computer with a USB cable. In the Arduino IDE, go to **Tools -> Boards** and select the name of the board. Then go to **Tools -> Port** and select the board there too. (If it's not showing up there, be sure your microcontroller is plugged into your computer via USB).

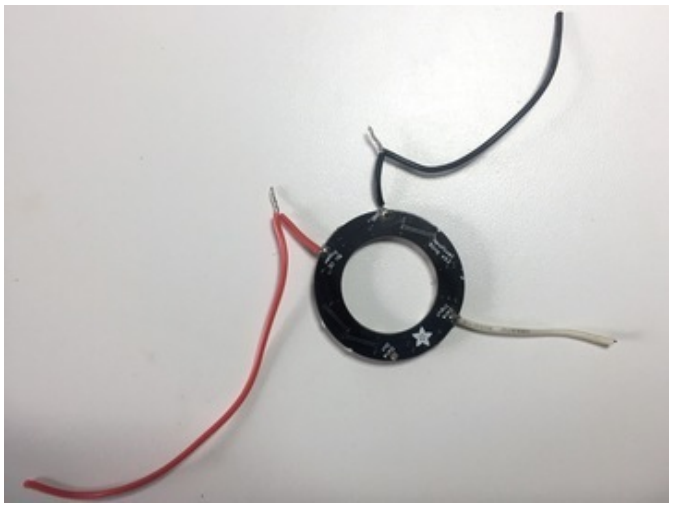
```
#include "Adafruit_FreeTouch.h"
#include "FastLED.h"

#define CAPTOUCH_PIN A1
#define NEOPIXEL_PIN 1
#define LED_PIN 0
#define NUM_LEDS    12
```

```
#define LED_TYPE    WS2812
#define COLOR_ORDER GRB
CRGB leds[NUM_LEDS];

int BRIGHTNESS=150;
int touch = 500;    // Change this variable to something between your capacitive touch serial readouts fo

long oldState = 0;
int gHue=0;

Adafruit_FreeTouch qt_1 = Adafruit_FreeTouch(CAPTOUCH_PIN, OVERSAMPLE_4, RESISTOR_50K, FREQ_MODE_NONE);
//Adafruit_FreeTouch qt_2 = Adafruit_FreeTouch(A2, OVERSAMPLE_4, RESISTOR_50K, FREQ_MODE_NONE);



void setup() {
  Serial.begin(115200);

  if (! qt_1.begin())
    Serial.println("Failed to begin qt on pin A1");

   pinMode(LED_PIN, OUTPUT);  //initialize the LED pin

   FastLED.addLeds<WS2812, NEOPIXEL_PIN, COLOR_ORDER>(leds, NUM_LEDS);  // Set up neopixels with FastLED
   FastLED.setBrightness(BRIGHTNESS);
   FastLED.setMaxPowerInVoltsAndMilliamps(3,350);  //Constrain FastLED's power usage
}

void loop() {

  Serial.print(qt_1.measure());
  Serial.write(' ');
  checkpress();
  delay(20);
}

void checkpress() {

// Get current button state.

    long newState =  qt_1.measure();
    Serial.println(qt_1.measure());
   if (newState > touch && oldState < touch) {
    // Short delay to debounce button.
    delay(20);
    // Check if button is still low after debounce.
    long newState =  qt_1.measure(); }


  if (newState > touch ) {
     dark();
     digitalWrite(LED_PIN, HIGH);
     delay(20);
    }

    else {
      rainbow();
      digitalWrite(LED_PIN, LOW);
      delay(20);
```

```
    }



  // Set the last button state to the old state.
  oldState = newState;

  // do some periodic updates
  EVERY_N_MILLISECONDS( 20 ) { gHue++; } // slowly cycle the "base color" through the rainbow
}

void rainbow()
{
  // FastLED's built-in rainbow generator
  fill_rainbow( leds, NUM_LEDS, gHue, 7);
  FastLED.show();
  delay(20);

}

void dark()
{
  for(int i = 0; i < NUM_LEDS; i++) {
  leds[i] = CRGB::Black;
  FastLED.show();
  delay(20);
}
}
```

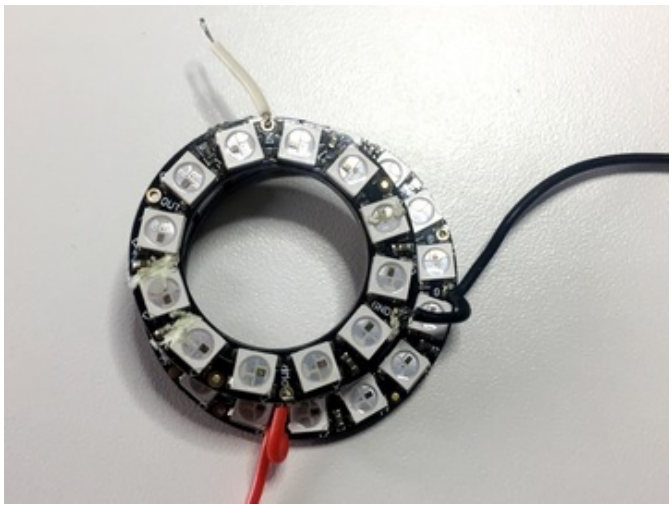
Assembly

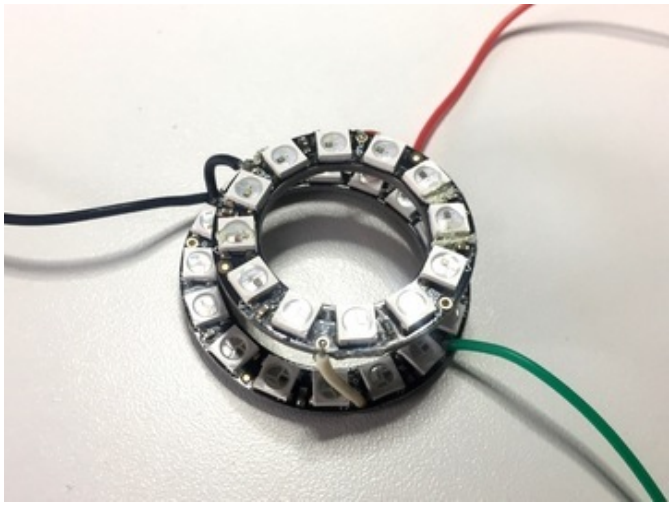




Start with your smaller 12 pixel ring. Solder a red wire to 5v, a black wire to G, and a white wire to Data IN. It's easiest to insert the wires from the front and solder on the back of the ring.

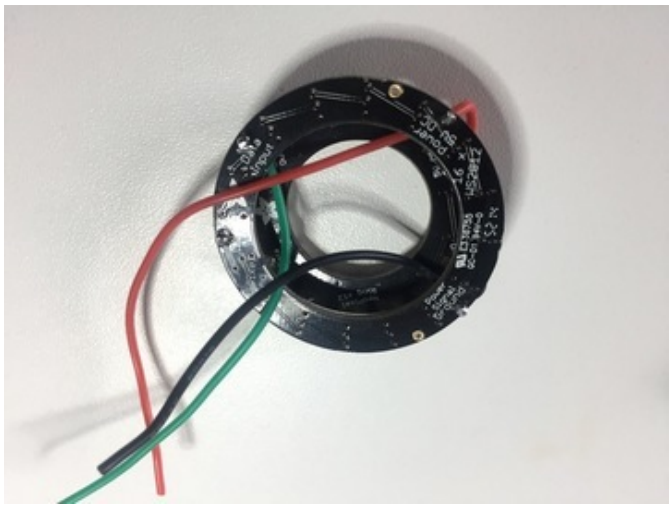
Trim the wires to about 1/2" and strip the ends. Twist another red wire and another black wire together with the wires you soldered.

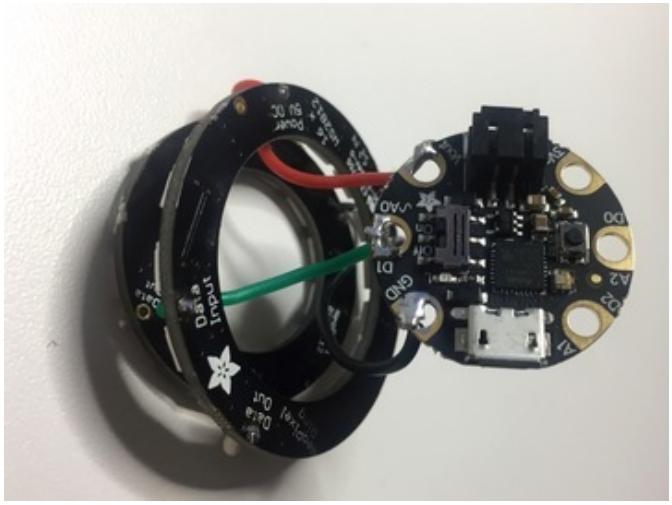




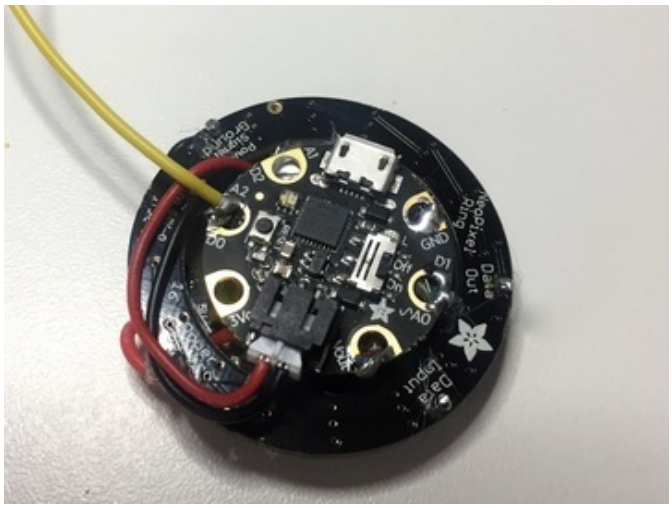
Place the 12 pixel ring on top of the 16 pixel ring. Solder BOTH the twisted red and black wires into the respective 5v and G holes on the second ring.

Solder the white wire coming from the smaller ring into the Data OUT pin. Then, add another wire to the 16 pixel ring's Data IN pin.





Pull the three loose wires through so they come out the back of the larger ring. Trim the wires and solder them to the Gemma: red to Vout, black to GND, and your data wire to D1.



Solder a 2" wire to gemma's pin D0. This wire will connect Gemma's capacitive touch pad to your metal pendant front to activate the capacitive touch features.

Plug your battery into the Gemma and flip the switch to "on". Be sure both rings are lighting up. Test the capacitive touch function by stripping the other end of the wire you soldered to D0, and touch the bare wires to make the lights change color.

If they're not changing, or if they appear to be changing lightning-fast, you may need to calibrate your code. Check the calibration section later in this guide.

Set the battery inside the large ring and nestle it against the back of the smaller ring. Center the rings so the LEDs show as much as possible.

Place the Gemma on the back of the pile, with the switch facing up. Be sure the battery wire can be pulled out easily and replaced -- you'll need to unplug this for charging. Once it's all aligned, glue all the components together with E6000 glue or epoxy.

Glue the whole assembly to the back of your pendant.

Thread the capacitive touch wire UNDER the battery wires if needed. Trim the wire and strip and tin the end. Solder it securely to the back of your metal pendant. Add a blob of glue on top of the solder to keep it secure.

Add a necklace cord. Put your pendant on, above your bare skin (not over a shirt or sweater) and touch to change color modes.

You may need to do some calibration in the code to get it reacting correctly to your touch.

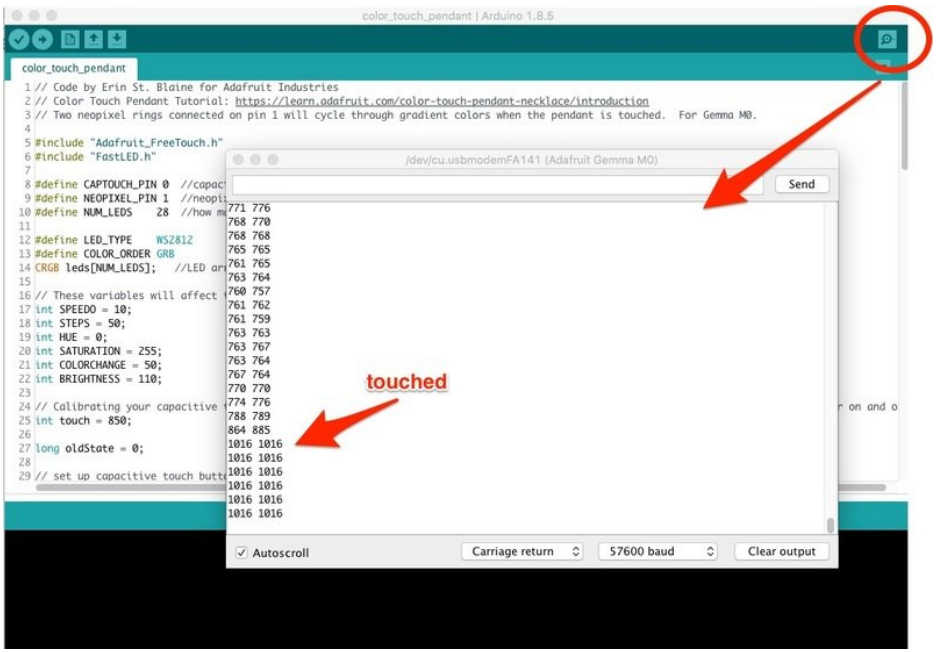
Calibration

Capacitive Touch Sensitivity

Capacitive touch pads can be finicky, especially for wearables and battery powered projects. Capacitive touch functionality is happiest when there's a strong ground connection, like you get with a wall plug. For small battery powered projects there's no real ground. But that's okay! Our body's electromagnetic field can work as our ground connection.

When this necklace is being worn against bare skin, the electronics have contact with the body, and through the body, the ground. You may find the necklace won't change colors if you touch it when you aren't wearing it, or if it's being worn over a sweater or other clothing. It needs that skin contact to be able to read your touch.

This can make calibration a little bit tricky, but with some trial and error you can get it to respond correctly.



With your necklace fully assembled, plug it in to your computer's USB port and switch it on. Open the code in your Arduino window, then open the serial monitor. You should see numbers scrolling by. (If you don't see them, try re-uploading the code)

Touch the pendant's face firmly with the pads of your fingers. You should see a change in the numbers scrolling by when the necklace reads your touch. When you let go, the values will revert.

Now look at your code and find this line near the top:

```
// Calibrating your capacitive touch sensitivity: Change this variable to something between your capaciti
int touch = 850;
```

Change the number to a number roughly halfway between your "touched" and "not touched" numbers. Since I have 700-ish as "not touched" and 1000-ish as "touched", I chose 850.

Re-upload the code. Now the color should change when you touch the necklace.

You're not quite done yet! Unplug the necklace and put it on, so that the back of the necklace is touching your skin (not over your shirt). See if it still works the same way. If so, you're golden! If not, plug your necklace back in and try adjusting this value until it works the way you like.

Color Calibration

There are several other variables you can change that will allow you to customize the way your necklace acts.

BRIGHTNESS: This will adjust the brightness -- use a value between 0 and 255.

SATURATION: Make your colors more pastel (lower number) or more rainbow-bright (255 max).

COLORCHANGE: Use this variable to define how much the color value changes each time you touch the necklace.

The code starts with a hue of 0, also known as "red". FastLED breaks down hue values on a range of 0-255, with red at 0, yellow around 60, and blue around 180, until it cycles back to red at 255. Using a value of 50 means that each time you touch the necklace, the hue will jump by 50. This gives me 6 different modes: red-orange (0), yellow-green (50), green-blue (100), blue-purple (150), purple-pink (200), and red-purple (250).

Using a value of 30 would give me 8 different color modes, since the hue would jump by 30 each time. Play with this until you get a satisfying array of colors.