# Table of Contents

**Overview**  
- Reading Up

**Materials & Tools**  
- 1 meter LED Strip with 144/m  
- Gemma M0  
- Momentary switch button  
- Silicone wire  
- Power Supply  
- Other Materials  
- Tools

**Prep**  

**Circuit Diagram**  

**Create Strips**  

**Test**  

**Code**  

**Gemma**  

**Final Assembly**  

©Adafruit Industries
Overview

Put an Adafruit Gemma M0 to work as you attach 120 Adafruit NeoPixels to the lapels of a vintage band jacket.

In a nutshell, this project starts by soldering small sections of NeoPixels into two longer strips and arrange them into a matrix formation. Add a clicky button to a Gemma M0 to showcase some beautiful animations and pick a brightness setting. Connect the two strips of NeoPixels to the Gemma M0, utilizing Mark Kriegsman's special XY mapping code to tell the Gemma the location of each pixel. Finally, protect the strips from the elements, and attach it to a costume jacket. Whether you’re headed to TTITD (that thing in the desert aka burning man) or leading a marching band, this wearable is sure to make you the star of any event.

This is a job for a board that packs a punch without much of a footprint, a perfect fit for the bite-sized Adafruit Gemma M0. This sketch only uses about 10% of the space on the Gemma M0, so there’s loads more space to add additional animations or power additional pixels.

There are about 75 solder joints in this project, so it’s great if you enjoy soldering! Remember to take breaks and triple check your work.

Reading Up

Check out these guides before you get started:

- Adafruit Neopixel Uberguide
- Adafruit Gemma M0
- Washing Wearables – Although we’re insulating our pixels and solder joints in glue, you should remove all of the electronics when washing or bathing your jacket. However, you may want to give your Gemma a spray of canned air or a quick wipe down now and again.

Materials & Tools

1 meter LED Strip with 144/m

I used the regular sized pixels with a black back. The Mini Skinny NeoPixel strips could also be used for this project!
Adafruit NeoPixel Digital RGB LED Strip
144 LED - 1m Black
We crammed ALL THE NEOPIXELS into this strip! An unbelievable 144 individually-controllable LED pixels on a flexible PCB. It's completely out of control and ready for you to... https://www.adafruit.com/product/1506

Adafruit Mini Skinny NeoPixel Digital RGB LED Strip - 144 LED/m
So thin. So mini. So teeeeeny-tiny. It's the 'skinny' version of our classic NeoPixel strips! These NeoPixel strips have 144 digitally-addressable pixel Mini LEDs... https://www.adafruit.com/product/2970

Gemma M0
The Gemma v1 or v2 will NOT work for this project!

Adafruit GEMMA M0 - Miniature wearable electronic platform
The Adafruit Gemma M0 is a super small microcontroller board, with just enough built-in to create many simple projects. It may look small and cute: round, about the... https://www.adafruit.com/product/3501

Momentary switch button
Any size will do. These are my favorite and fit perfectly on the back of the Gemma!
Silicone switch buttons

Tactile Switch Buttons (12mm square, 6mm tall) x 10 pack
Medium-sized clicky momentary switches are standard input "buttons" on electronic projects. These work best in a PCB but [https://www.adafruit.com/product/1119](https://www.adafruit.com/product/1119)

Silicone wire

I used the 30awg wire between the each strip of pixels and the 26awg wire between the pixels and Gemma. Using the larger size will also work. I went with all black to blend in with the jacket I was using, but this would also look fabulous with any color!

If you don't feel comfortable with this technique and/or want to add more color, I'd encourage you to pick three or more colors for this project. Typically, a ground wire is black or white, a power wire is red, and a data wire will be any other color.

Silicone Cover Stranded-Core Wire - 50ft 30AWG Black
Silicone-sheathing wire is super-flexible and soft, and its also strong! Able to handle up to 200°C and up to 600V, it will do when PVC covered wire wimps out. We like this wire...
[https://www.adafruit.com/product/3164](https://www.adafruit.com/product/3164)

Silicone Cover Stranded-Core Wire - 25ft 26AWG - Black
Silicone-sheathing wire is super-flexible and soft, and its also strong! Able to handle up to 200°C and up to 600V, it will do when PVC covered wire wimps out. We like this wire...

©Adafruit Industries
Power Supply

A portable USB battery pack or LiPo will work. I'll be using a USB battery pack. If you select a Lithium Ion Polymer Battery, be sure to read up on [Li-Ion & LiPoly Batteries](https://www.adafruit.com/product/1566) first!

A quick note about batteries: it's hard to tell exactly how long a battery will last since it depends largely on the brightness and animation selected. I like to use a 20mAh per LED calculation as a starting point, so I wouldn't want to use anything less than 2400mAh unless I was only planning on using the wearable for a short period of time.

**USB Battery Pack for Raspberry Pi** -
**10000mAh - 2 x 5V outputs**
A large-sized rechargeable battery pack for your Raspberry Pi (or Arduino, or [https://www.adafruit.com/product/1566](https://www.adafruit.com/product/1566))

**USB Battery Pack for Raspberry Pi** -
**5000mAh - 5V @ 2.1A**
Discontinued - You can grab [USB Battery Pack for Raspberry Pi - 10000mAh - 2 x 5V outputs](https://www.adafruit.com/product/1565) instead! A...

**Other Materials**

- Awesome Vintage Costume Jacket
- Glue: E6000 or hot glue
- Velcro
- Optional: chiffon or other mesh fabric
Tools

- Scissors, straight pins, etc.
- Optional: Dress form
- Soldering Station
  - Soldering iron, Soldering iron stand, Solder
  - Wire stripper, Flush Cutters
  - Helping hands, Alligator Clips
  - Hakko Brass Sponge Solder Tip Cleaner (http://adafruit.it/1172)
  - Ventilation, I recommend the USB Rechargeable Mini Solder Fume Extractor guide from Phillip Burgess or the Desktop Fume Extractor from the Ruiz Brothers

Prep

Plan out the project and prepare materials for soldering!

Cut the ends of the pixels and remove them from their silicone case

Measure out some strips of paper down to the size of some various strips of pixels
Pin them to the jacket to see what would work best.

I decided on 8 strips for each side, ranging from 4-11 pixels in length.

Since the copper pads on 144 pixels/meter strips are on the tiny side, it's easiest to cut out and lose a pixel between each strip.

The Larson Scanner Shades guide () illustrates this in the wiring step as well.
Cut pixel strips to desired lengths.

Keep an eye on the half way mark on a meter of pixels. Since the distance between pixels isn’t equal, the strips can be desoldered or cut around.

Cut silicone pixel covers slightly larger by measuring up a pixel length. For example, I cut a silicone cover to match the 10 pixel strip and used it for a strip of 9 pixels.

---

**Circuit Diagram**

Review the circuit diagram. This illustration is meant for referencing wired connections. The length of pixels, wire, position, and size of components do not match the actual project.

The circuit diagram shows that all three open data pins on the Gemma M0 board will be in use:

- The pixels located on the LEFT (the wearer's right) connect to Ground, Power, & D1
- The pixels located on the RIGHT (the wearer's left) connect to Ground, Power, & D0
- The button connects to Ground & D2

Note that the top couple of rows of pixels aren't all pointing in the same direction, but the data line always flow in the correct direction. If the pixels weren't going to be
mapped out in the code, this might result in some weird animations. Mapping the order and orientation of the pixels out allows the strips to point to the left or right.

Create Strips

Begin with the 8 strips of pixels that will go on one side of the jacket. Pick a side to start with since they cannot be switched once started. Start with the data line and finish with the ground and power lines, braiding wires, if desired.

Refer to the circuit diagram as needed.

Start by connecting the data line between LEDs, ensuring that the arrows all point in the same direction!

Focus on two strips to start.
As the strips are on the jacket, measure the distance between the data out of the upper strip to the data in of the lower strip & cut a piece of thin wire ~1/2" (1.3 cm) longer.

Please be careful using a hot soldering iron to prevent burns. Unplug when not in use.

Complete the data line using pads and wires that have been tinned. Solder the data in to the data out using the wire. Solder the wire to the data in pad on the back of the strips.

When soldering to small copper pads, I find that its easiest to strip about 1/4" of silicone sheathing off the wire, tin the wire, and then trim down the exposed wire to a more appropriate size. The copper pads on the 144 led/meter strips are super tiny and too long of a tinned wire will happily attach itself to whatever surface mounted part is nearby.
Cut 14 wires long enough for the distance between the ground and power lines of each pixel. On half (7) of the wires, strip and tin both sides as the power wires. On the other half (7), strip and tin just one side as the ground wires.

If using wires of different colors, this method of stripping is not necessary.

Start one pixel section up from the bottom.

Twist and tin the tips of two power wires to a copper pad marked power.

On the same pixel section, twist and tin two ground wires to a copper pad marked ground.
Braid the wires down to the end pixel. Braiding is optional, but really helps with wire management!

Solder the power wire to the copper pad marked power. The power wires will have stripped ends and should be soldered first.

Strip and solder the ground wire to the copper pad marked ground. The ground wires will need to be tinned before soldering.
Continue to solder sections and braided together wires until one side is complete. Take a break!

Repeat for the opposite side. Take another break!

Test

Pick out where the Gemma will sit on the jacket. I picked the location of where a breast pocket might be.

Using the thicker (26 awg) wires, cut 3 pieces to connect the pixels on the left to the Gemma and another 3 pieces to connect the pixels on the right to the Gemma.

Solder the larger and longer wires to first pixels, labeling wires if they are all the same color.
Braid the wires!

Using alligator clips, test the strand using the NeoPixel Strandtest () code or other test code.

Note that if some strips have been flipped, a rainbow might appear a bit off. As long as the strip works, set it aside and test the second strip!

If you run into issues, turn things off, trace your data line, check your microcontroller, soldering & replace strips, as necessary.

---

**Code**

This step includes a review of pixel mapping, combining both strips into an array, and setting up new functions.
If you are new to Arduino, check out this guide: Adafruit Arduino IDE Setup. Check out this guide for additional information about Arduino IDE Setup for the Gemma M0.

This sketch uses the FastLED library. See FastLED.io for more information, the FastLED github, or visit their community on Reddit. You can learn more about Arduino libraries by reading this guide from Adafruit.

Download the sketch & open it on your Arduino IDE. The complete file is needed to run the sketch.

Connect the Gemma M0 and select it as the board in the Arduino IDE. Choose the connected Gemma from the port dropdown and upload the code.

The tabs of the sketch (DiscoBandCamp.ino, XYmap.h, util.h, effects.h, & buttons.h) are all reviewed below.

Download the Disco Band Camp Jacket for Arduino

DiscoBandCamp.ino

On the first tab, review the pins where the strips are connected:

```cpp
// Pins on Adafruit Gemma M0
#define LEFT_PIN    1     // Visual Left (LEDs on the wearers right) connected to D1
#define NUM_LEFT    60    // number of LEDs connected on the Left
#define RIGHT_PIN   0     // Visual Right (LEDs on the wearers left) connected to D0
#define NUM_RIGHT   60    // number of LEDs connected on the Right
```

The strips are combined into an array so that they act as a single unit:

```cpp
// Runs one time at the start of the program (power up or reset)
void setup() {

// Add the onboard Strip on the Right and Left to create a single array
FastLED.addLeds<CHIPSET, LEFT_PIN, COLOR_ORDER&>(leds, 0, NUM_LEFT);
FastLED.addLeds<CHIPSET, RIGHT_PIN, COLOR_ORDER&>(leds, NUM_LEFT, NUM_RIGHT);

// set global brightness value
FastLED.setBrightness( scale8(currentBrightness, MAXBRIGHTNESS) );

// configure input buttons
pinMode(MODEBUTTON, INPUT_PULLUP);
}
```
Mark Kriegsman, co-author of the FastLED library, created a great XY mapping code which allows us to program our project as if it were a (24x8) matrix. The Macetech RGB LED shades and Charity Stolarz Bright Top use this approach; both served as a great model for this project.

Start by mapping out pixels in a spreadsheet based on their location. I left a couple spaces between the strip on the left and the one on the right because there is an opening between the sections.

Fill in open holes on the spreadsheet with numbers above 119. I’ve highlighted the additions.

Add a comma at the end of every cell.

I used the concatenate function in my spreadsheet to add this quickly.

The mapping above creates a matrix with a height of 8 pixels and a width of 24 pixels, which can be entered into the sketch:

```cpp
// Parameters for width and height
const uint8_t kMatrixWidth = 24;
const uint8_t kMatrixHeight = 8;
const uint8_t kBorderWidth = 0;
#define NUM_LEDS (kMatrixWidth * kMatrixHeight)
CRGB leds[NUM_LEDS];
```

Next, copy and paste the entire spreadsheet into the table, titled JacketTable:

```cpp
const uint8_t JacketTable[] = {
10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 145,
153,60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
};
```
buttons.h

This part of the sketch has button setup information. The RGB Shades sketch utilized two buttons but the Gemma only has space for one. The pin the button is connected to is named here:

```c
#define NUMBUTTONS 1
#define MODEBUTTON 2  //define the pin the button is connected to
```

Next, review how the buttons will work. When there's a short button press, a new mode will be selected, and when there's a longer button press, the brightness will increase (or wrap down to a lower brightness).

```c
void doButtons() {
  switch (buttonStatus(0)) {
    case BTNRELEASED: // short button press
      cycleMillis = currentMillis;
      if (++currentEffect >= numEffects) currentEffect = 0; // loop to start of effect list
      effectInit = false; // trigger effect initialization when new effect is selected
      break;
    case BTNLONGPRESS: // long button press
      currentBrightness += 51; // increase the brightness (wraps to lowest)
      FastLED.setBrightness(scale8(currentBrightness, MAXBRIGHTNESS));
      break;
  }
}
```

effects.h

This tab contains a selection of effects from the FastLED library, Macetech's RGB Shades, and Charity Stolarz's Bright Top. This is a review of the easiest modes to personalize.
Create a single color mode with whatever you'd like using any of the six ways to set an RGB LEDs color with FastLED:

```cpp
void SolidRed() //for startup, good for saving battery
{
    fill_solid(leds, NUM_LEDS, CRGB::Red);
}
```

There are two confetti modes in the sketch. One selects a random built-in FastLED palette (from utils.h) and the other will always use the palette I've created.

Create your own confetti modes using the built in Palettes (see utils.h) or create your own. Use with the fadeAll function (see .ino) to allow old pixels to decay.

Update the palette below to see what different colors look like. If creating a new confetti function, generate a new palette, new confetti name, and update the ColorFromPalette selection.

```cpp
//Palette for myConfetti
const TProgmemPalette16 MyColors_p PROGMEM =
{
    CRGB::Crimson,
    CRGB::Maroon,
    CRGB::Red,
    CRGB::OrangeRed,
/
    CRGB::Crimson,
    CRGB::Maroon,
    CRGB::Red,
    CRGB::OrangeRed,
/
    CRGB::Crimson,
    CRGB::Maroon,
    CRGB::Red,
    CRGB::OrangeRed,
/
    CRGB::Crimson,
    CRGB::Maroon,
    CRGB::Red,
    CRGB::OrangeRed,
};

void myConfetti() {
    // startup tasks
    if (effectInit == false) {
        effectInit = true;
        effectDelay = 15;
    }
/
    // scatter random colored pixels at several random coordinates
    for (byte i = 0; i < 4; i++) {
        leds[XY(random16(kMatrixWidth), random16(kMatrixHeight))] =
            ColorFromPalette(MyColors_p, random16(255), 255); //CHSV(random16(255), 255, 255);
            random16_add_entropy(1);
    }
}
```
Incorporate some other example code

I really like the NoisePlusPalette FastLED example sketch, so I pulled it in here. The main functions are in utils.h:

```c
void NoisePlusPalette() {
    fillnoise8();
    mapNoiseToLEDsUsingPalette();
}
```

I really like the swirly example sketch used in Charity's Bright Top, so I pulled it in here, too:

```c
void swirly() {
    // startup tasks
    if (effectInit == false) {
        effectInit = true;
        effectDelay = 15;
    }

    // Apply some blurring to whatever's already on the matrix
    // Note that we never actually clear the matrix, we just constantly
    // blur it repeatedly. Since the blurring is 'lossy', there's
    // an automatic trend toward black -- by design.
    uint8_t t blurAmount = beatsin8(2,10,255);
    blur2d( leds, kMatrixWidth, kMatrixHeight, blurAmount);

    // Use two out-of-sync sine waves
    uint8_t t i = beatsin8(27, kBorderWidth, kMatrixHeight-kBorderWidth);
    uint8_t t j = beatsin8(41, kBorderWidth, kMatrixWidth-kBorderWidth);
    // Also calculate some reflections
    uint8_t ni = (kMatrixWidth-1)-i;
    uint8_t nj = (kMatrixWidth-1)-j;

    // The color of each point shifts over time, each at a different speed.
    uint16_t t ms = millis();
    leds[XY(i, j)] += CHSV( ms / 11, 200, 255);
    leds[XY(j, i)] += CHSV( ms / 13, 200, 255);
    leds[XY(ni,nj)] += CHSV( ms / 17, 200, 255);
    leds[XY(nj,ni)] += CHSV( ms / 29, 200, 255);
    leds[XY(i,nj)] += CHSV( ms / 37, 200, 255);
    leds[XY(ni, j)] += CHSV( ms / 41, 200, 255);

    FastLED.show();
}
```

This page contains bits and pieces that are helpful, including pulling from a random built-in FastLED palette:

```c
// Pick a random palette from a list
void selectRandomPalette() {
```
switch(random8(8)) {
    case 0:
    currentPalette = CloudColors_p;
    break;

    case 1:
    currentPalette = LavaColors_p;
    break;

    case 2:
    currentPalette = OceanColors_p;
    break;

    case 4:
    currentPalette = ForestColors_p;
    break;

    case 5:
    currentPalette = RainbowColors_p;
    break;

    case 6:
    currentPalette = PartyColors_p;
    break;

    case 7:
    currentPalette = HeatColors_p;
    break;
}

---

**Gemma**

Remove two of the opposite legs from the button.
Attach the button to the back of the Gemma and solder.

Beginning with the power and ground wires this time, twist them together, attach to the Gemma, and solder.

The power wires connect to Vout.

The ground wires connect to GND. Note that one of the legs for the button is also on the Gemma's GND pad.

Test the whole thing! Give the button both short and long presses to see the modes.
Final Assembly

Place pixels in silicone covers if they aren't already back in.

Glue to secure pixels and wires, snipping any excess silicone off.
Insulate joints on Gemma and pin LEDs to jacket
Optional: wrap pixels in a sheer fabric to match the jacket.

I used a dark sheer chiffon-like fabric, cut into 2" wide strips. I attached the fabric to the pixel covers using hot glue.
Attach pixels and Gemma to jacket.

I glued velcro to the covered pixels and jacket and used pieces of double sided velcro to hold wiring in place.