NeoPixel Cyber Falls Wig

Created by Phillip Burgess

https://learn.adafruit.com/neopixel-cyber-falls

Last updated on 2024-06-03 01:24:38 PM EDT

Table of Contents

Overview

“Cyber falls” are a hair fashion accessory you might see in some rave and Burning Man circles. Also great for Halloween! This project steps it up with LED animation using NeoPixel strips.

HEADS UP: Before You Proceed...

Despite its playful appearance, this is a somewhat challenging project. It requires competent soldering and quite a variety of tools, most with the potential for injury. Read through first and decide if it's really for you. We have plenty of other wearable projects that are less daunting! Young makers should read through with a parent to help decide.

This guide was written for the Trinket Mini board, but can also be done with the Trinket M0. We recommend the Trinket M0 as it is easier to use and is more compatible with modern computers!

Tools used:

- Soldering iron and associated paraphernalia
- Heat gun (a hair dryer will not work, nor will a lighter)
- Hot glue gun and glue sticks
- Sewing machine and accoutrements
- Scissors, needlenose pliers, wire cutters
- Foam wig head

Materials used:

- NeoPixel LED strips - 30 LED strips use less power, 60 LED strips are glowier! (https://adafru.it/cSc)
- Trinket microcontroller (http://adafru.it/1501) and "Small Tin" Perma Proto board (http://adafru.it/1214)
- 3 x AA battery case w/switch (http://adafru.it/771) (lasts longer!) OR 3XAAA battery case w/switch (slightly smaller) (http://adafru.it/727)
- Stranded (not solid) copper wire, 24 to 20 gauge, three different colors
- Clear heat-shrink tube — 1/2" diameter before shrinking
- Tubular crinoline ribbon
- "Do rag" head scarf
- Elastic or fabric ribbon
- Small cable ties
- Craft foam (optional)

Not all of these parts are available from Adafruit...some detective work will be required. Read through the whole guide before committing to anything...maybe you'll

come up with substitutions for a different design or different assembly techniques. This is not a complete step-by-step how-to so much as an idea showcase.

Can I use a Gemma instead of Trinket?

Yes,

Gemma has only three output pins for controlling NeoPixels. Our example code uses all five output pins on the Trinket, so you would need to adapt the number of LED dreads and how many connect to each pin.

Making LED Dreads

The exact number and distribution of NeoPixels in this project is all up to you. My goal was to re-use some half-meter NeoPixel segments remaining from early Firewalker LED Sneakers (https://adafru.it/cQN) prototypes. Aiming for a “mad scientist” look — wild, mid-length hair — the half-meter segments were each cut in half, yielding ten strips of 15 pixels each (1/4 meter each, 2.5 meters total). Though “flexible,” these strips don’t really like repeated bending, so keeping them relatively short (above shoulder length) keeps them out of harm’s way — less likely to be bumped or flexed. Longer strips are probably okay for infrequent use but may give out at some point.

You can go longer or shorter, more or fewer depending on the look you’re after and your budget. Strips can be different lengths if desired. Both the 30 LEDs/meter and 60 LEDs/meter strip (or a mix) can be used. The former is more affordable but more sparse...looks fine for a random “sparkle” effect, while the 60 LEDs/meter looks better for smooth motion effects along each dread. Don’t use the 144 LEDs/meter high-density strip though...that type doesn’t have the silicone sleeve, which we rely on for extra durability and “body” to the hair.

The hair tubes can be ordered from a number of sources online — try Etsy, Amazon or eBay, or a bit of Googling will turn up other sources. Try searching for “cyberlox,” “tubular crinoline” or “tubular crin.” The stuff comes in an amazing variety of colors and patterns. All types will pass at least some light, but lighter and non-metallic colors are best. For a “mad science” headpiece we used a mix of metallic and non-metallic white. The stripey ones are fun too...sort of a Tim Burton aesthetic.

Making a full head of hair requires a lot of ribbon! Our headpiece used about 20 meters and was still too sparse — the thin spots needed to be filled out with strips of craft foam. 30 meters of ribbon would have been about right for this design.

As this project goes together, parts and connections become progressively more inaccessible. Between that and the need for impeccable soldering, it’s strongly recommended that you work methodically in a mass-production fashion, testing components at every stage. Test the full NeoPixel strip before cutting. Perform soldering on all the sub-strips, then test the lot, setting aside items needing repair and further testing. Seal all the working strips inside tubes, then test again...and so forth, throughout the build process. Going through the whole process for each strip separately would take much longer!

If you have extra NeoPixel strip, you may want to make a couple spare dreads in case some don’t work, or for repair or replacement later.

Test and Prepare NeoPixel Strip

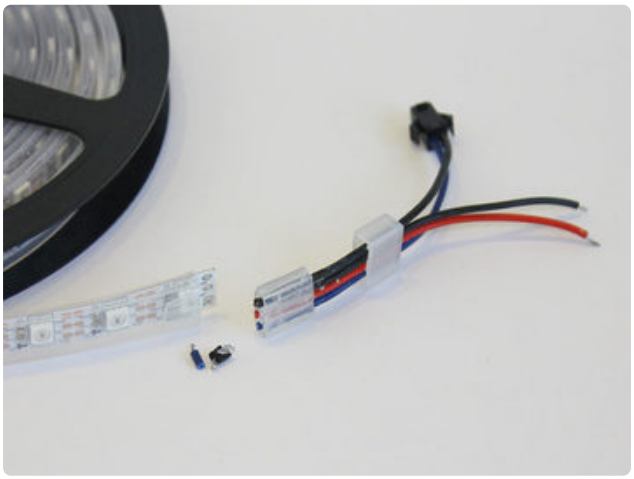
Before starting any of the LED work, test your full supply of NeoPixel strip, whether a full reel or in shorter segments. Run the strandtest example included with the Adafruit_NeoPixel library (https://adafru.it/aZU) for Arduino, confirm that all pixels are working.

For long strands and whole reels, testing is best done with an Arduino Uno or similar microcontroller; the Trinket only has enough RAM for about 100 NeoPixels.

A 5V DC supply (http://adafru.it/276) powers the strand, with a 1000 µF capacitor to prevent surges to the pixels. Both ground and data are connected to the Arduino.

What are my options if some pixels don't work?

- Start a thread in the Adafruit Forums describing the problem, ideally with a photo that clearly shows your wiring and power setup...we'll take a look and troubleshoot any common problems. If the strip is defective, we can then arrange for an exchange.
- If you're in a serious rush and can't wait for a replacement, and if it's just one or two bad pixels out of the whole reel: depending how you plan to divide up the strip for this project, you be able to cut around the bad pixels, salvaging the good segments between them.



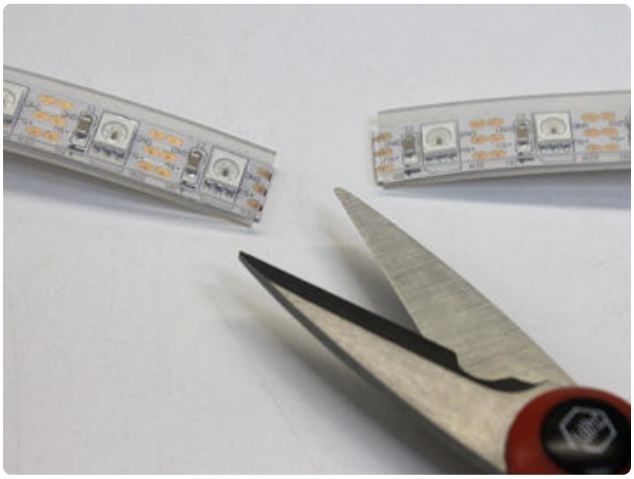
If starting with a full NeoPixel reel, cut away the end connectors and carefully desolder the wires from the flex PCB. We'll make new and different end plugs later — they need to be stronger than the factory type.

The 2-pin JST connectors (if your strip includes them) may come in handy later. Set them aside, or add them to your parts bin for other things.

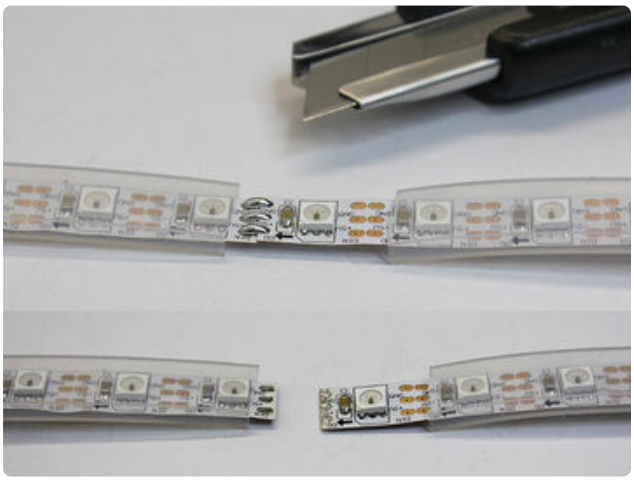
Cut NeoPixel Strip into Segments

You should already have a plan at this point for how you intend to divide and distribute the LEDs around the headpiece. Ours was simple: ten equal-length segments of 15 NeoPixels each, but yours might be more complex with different lengths. Sketch it out in advance or try a mockup with a foam wig head and strips of paper.

There are electrical and software factors, too. The Trinket board only has enough RAM to manage about 100 pixels, and has a maximum of 5 output pins (our design pairs up the NeoPixel strips, each pair always shows the same pattern of lights). If you need more pixels or fully independent control of many strips, you might need to step up to an Arduino Micro (http://adafru.it/1315) or a Teensy (http://adafru.it/199) board.



At most points along the strip, you can just cut through the covering and the flex PCB with utility scissors. Do not use your nice fabric shears for this!



LED flex strips are manufactured in half-meter segments, soldered together to produce a full reel. If one of your planned cut points coincides with a seam, cut through the outer covering with a razor blade or hobby knife, then separate the connections with a soldering iron and a desoldering pump or wick.

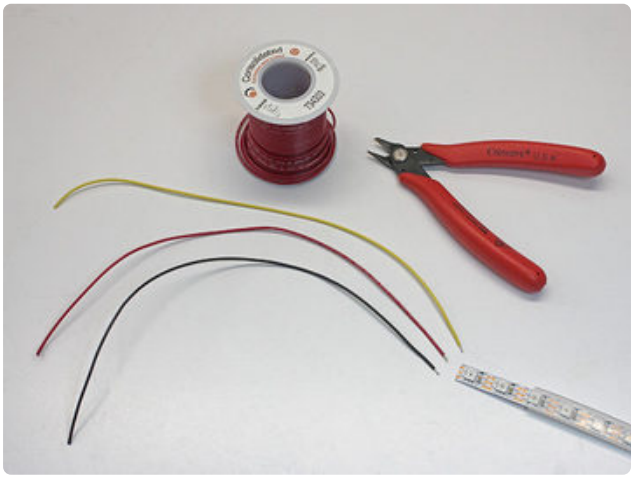
Alternative: if you're making different-length strips, see if you can reorder the cuts to skip over these joints.

Cut all of your LED sections before continuing. It's much less tedious to do each of these steps in bulk than to fully work through each dread onesy-twosey.

Solder Wires to Strips

This project, more so than others, requires competent soldering skills and a quality iron in good working shape (you'll see why shortly). An old hand-me-down iron with a crusty tip isn't going to cut it, nor will weak "cold" solder joints.

Use stranded wire for this project! Solid-core wire does not take well to repeated bending, which wearable projects tend to see a lot of. 24 gauge is ideal, but a little thicker (22 or 20 gauge) can work too.

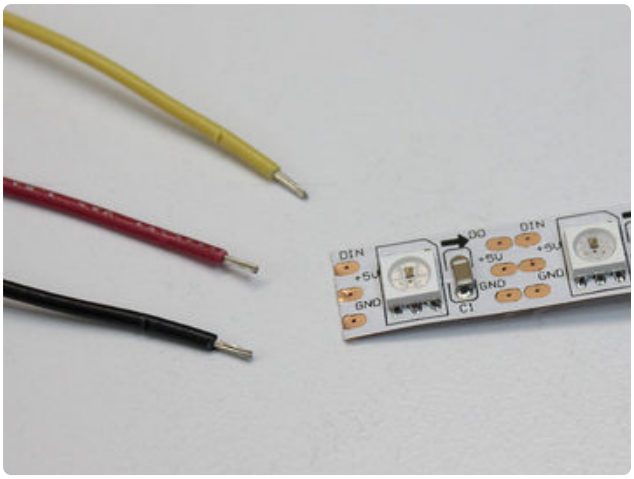


Cut wires for all of your strips: three wires per strip (use different colors), about 10 inches long.

The color-coding will save your sanity later when hooking everything together. It gets quite "hairy!"

Most of these will be trimmed shorter later...a little wasteful, but much less frustrating than trying to plan every wire's length (and finding you having to splice extra wire later).

As with the prior steps, it's most efficient to do this in bulk; cut the wires for all the LED strips now.



Strip just a little insulation from one end of every wire. About 3 mm or 1/8 inch.

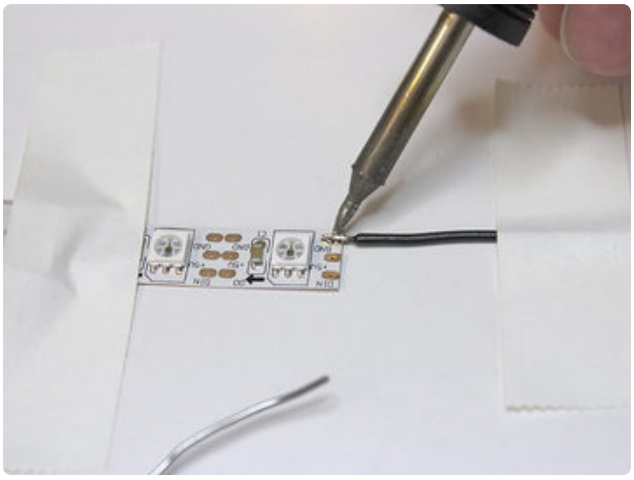
Again, batch process your work. Strip all the wires!

In a moment, we'll start soldering wires to strips. Slide the covering down a little to expose the strip inside.

See those little arrows? Those indicate the direction that data moves. You want to make sure you're connecting to the **input** end of the strip, labeled "DIN".

Connect wires to the input ("DIN") end of the strip. It won't work otherwise.

Line up the first wire with the strip, either with a "helping hands" tool, or just tape both down to a working surface, then solder the connection between them.



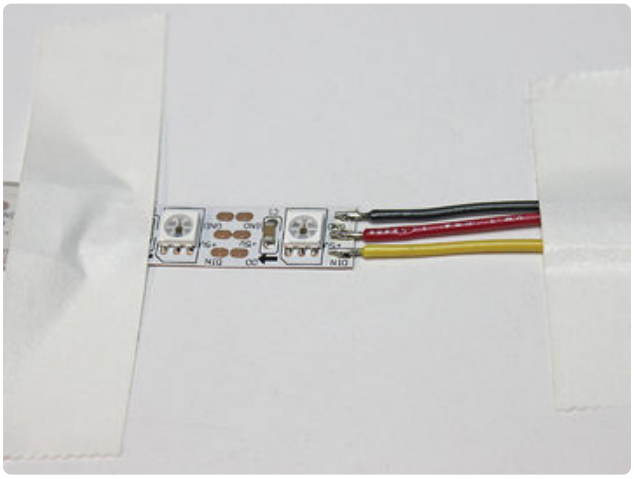
Tin the tip of the iron and touch it to the wire and copper pad on the strip, allowing both to heat up for a moment before adding more solder.

Properly done, solder will flow onto the pad and saturate between the wire strands. If the solder beads up on the surface, that's a cold solder joint and will soon break.

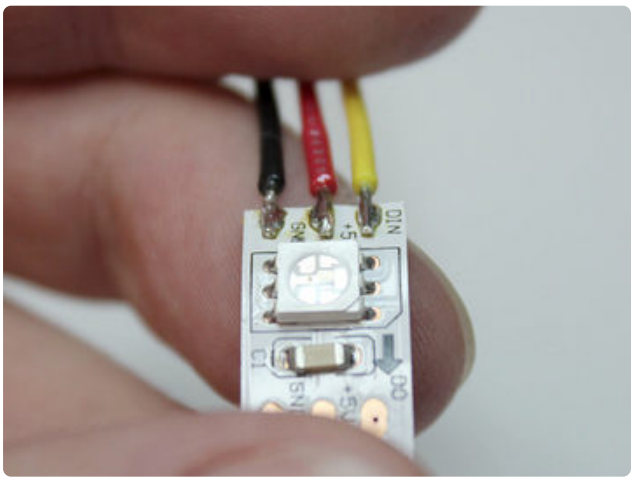
Do not heat a blob of solder on the tip and then move it to the joint. That's just a recipe for failure.

Some people find it easier instead to "tin" the wire with solder, and also the pad, then bring the two together and re-melt the solder. This is not ideal — the flux has already burned away at that point and the solder may be sticky on the iron tip — but

sometimes it’s good enough.



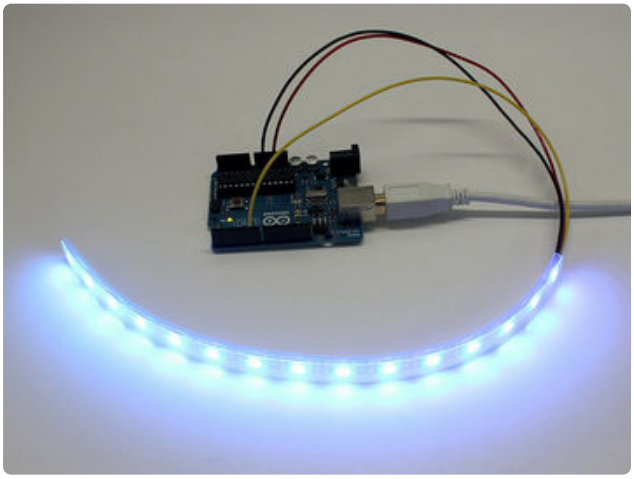
Repeat with the other two wires.



Look closely (click to embiggen). Notice how the solder has soaked well into the stranded wire and spreads out nicely from there to the solder pads — it’s not a blob resting on the surface.

Do the same for all your strips now. Let’s get all the tricky soldering done and out of the way. When finished, strip about 3/8" of insulation from the other end of all the wires.

After all of your strips are wired up, test each one using the strandtest sketch again.



When briefly testing short strips like this, it's usually okay to power them from the Arduino's 5V pin. The initial demand for current may cause the Arduino to "brown out" and stop; if the LEDs do nothing at first, try pressing the Reset button.

For longer strips, use the same power supply and capacitor as before.

If a strip doesn't light up, make sure you soldered to the "DIN" (input) end, and check for solder bridges or other connection problems. Set aside any problem strips. When you're done testing the rest, go back and fix the soldering on these.

Seal Ends of Strips

Soldering iron can be switched off for now.

Plug in your hot glue gun and get your heat gun ready.

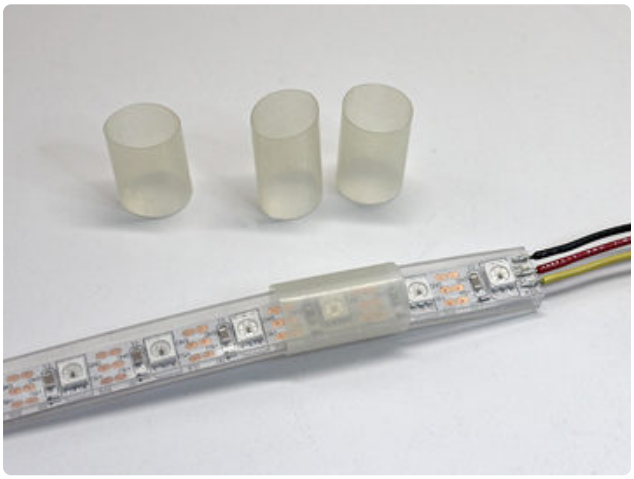
Both of these tools use a lot of current. They should be plugged into an outlet with no other major appliances on the same circuit. If there's a coffee pot or a microwave oven on the other side of the wall, this can trip the circuit breaker if they're all running at the same time. Run both tools for a moment to check that everything's okay.

And now we do something unspeakable!

If there were a Rule Number One for wearable electronics, it might be "never use current-carrying wires as a load-bearing element." Guess what we're about to do...

Our goal is to make sure the solder connections don't get strained and crack clean off. Yet they're right at the base of the "hair," where it attaches to the head. The very point of most flexing.

To accomplish this, the connections at the end of the strip need to be potted — encased in a solid material to help resist shock and vibration. This is why the solder connections must be flawless, because they will be inaccessible from this point forward, permanently encased in a blob of plastic.



While your glue gun warms up, cut some pieces of clear heat-shrink tube, about 3/4 to 1 inch long. You'll need **two for each strip.**

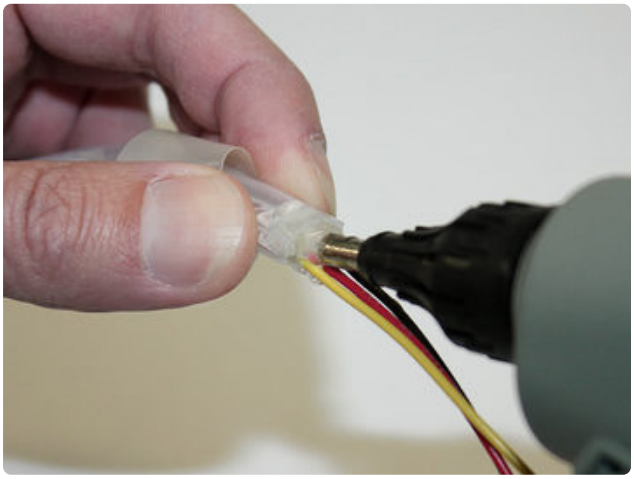
The tubing used here is about 1/2 inch in diameter before heating.

Slide a piece of tubing over the strip, about an inch from the wired end.

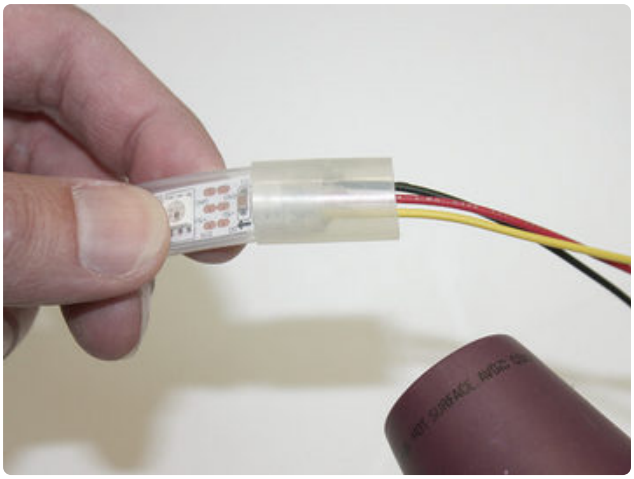
Danger, Will Robinson!

The tip of a hot-melt glue gun is nearly 400 degrees Fahrenheit, and the hot air gun is similar. This is more than enough to inflict **third degree burns**.

The following photo sequence shows these tools operating less than an inch from bare fingers. Do not interpret this as "harmless." **These tools WILL bite**.

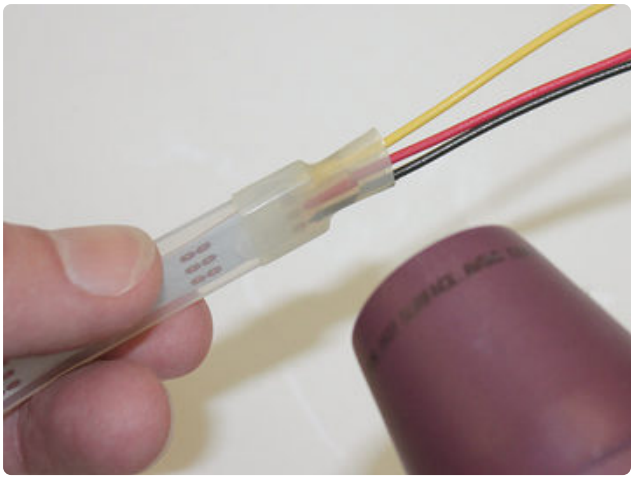


Pinch the LED rubber sleeve and inject a dollop of hot glue. Do this on both the back side and front side of the strip. It's okay if a bit oozes out the end.

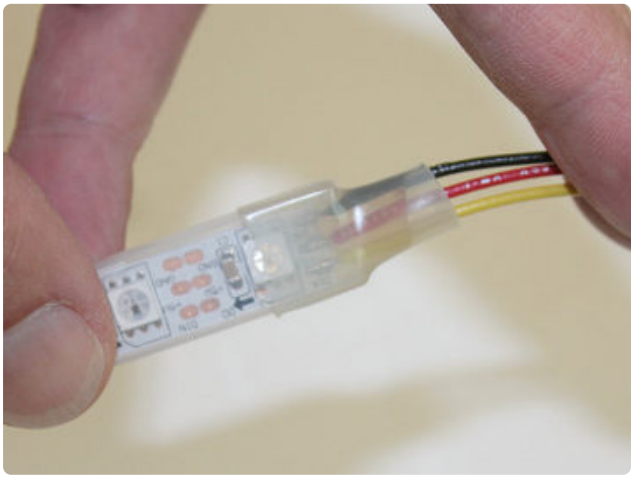


Quickly, while the glue is still hot, slide the heat shrink down so half its length overhangs the end of the strip. Then begin shrinking the tube with the heat gun.

The heat gun really is necessary; a lighter won't cut it. We're using a big tube and we're trying to keep the glue molten at the same time.

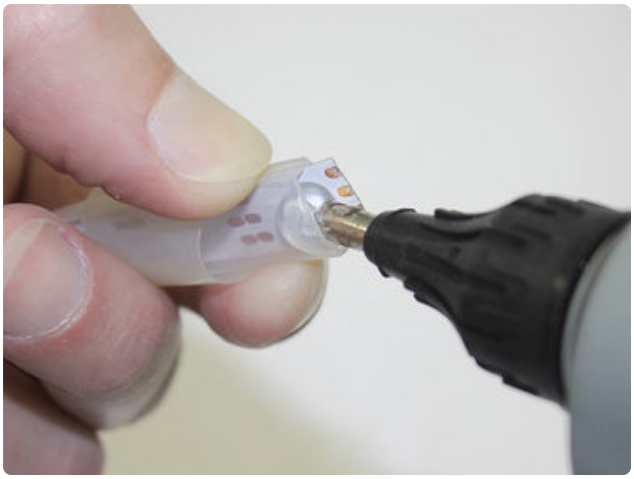


Turn it over a few times as you work, making sure the tubing is fully shrunk with no wrinkly spots. The still-liquid glue will ooze toward the end as you do this...that's okay.



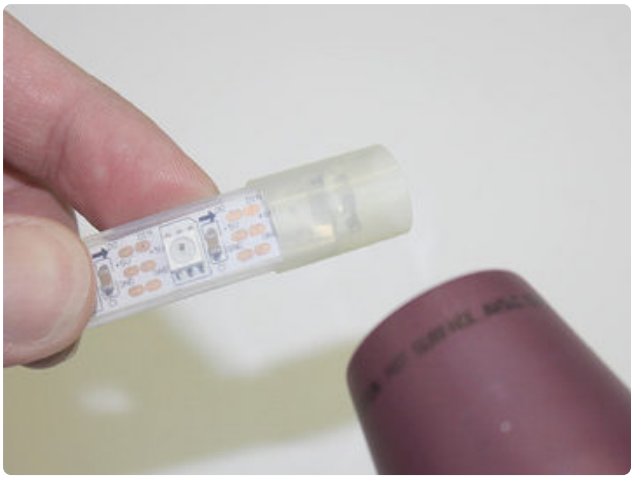
Turn the heat gun off and hold the strip and wires for about a minute as the glue cools and solidifies. If needed, you can work the shape a bit with pliers during this time.

Once cooled: the ends of the strip, the wires and the solder connections are firmly encased in a solid blob of plastic and should be impervious to just about anything.



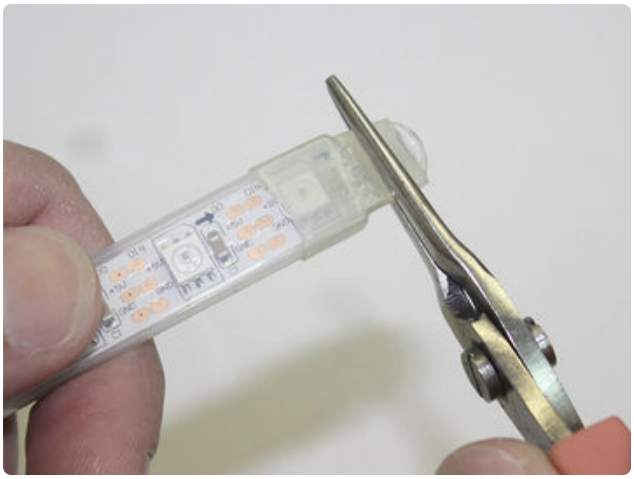
Now the process is repeated at the other end of the strip, without the wires.

Slide a bit of heat-shrink tube to the ready position and squirt some hot glue into the LED tube, both front and back.

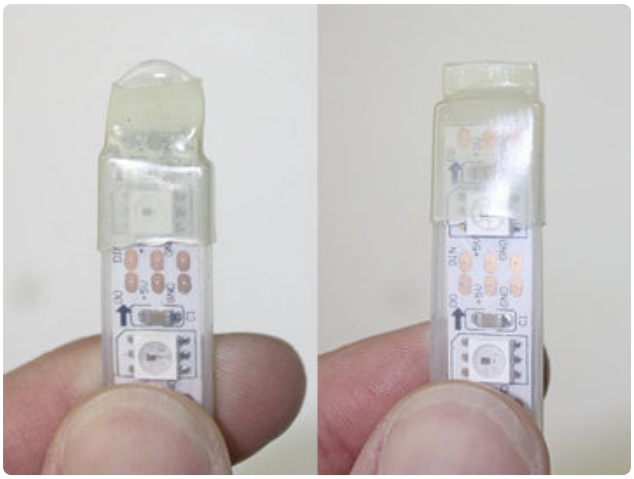


Slide the heat-shrink halfway off the end and start applying heat.

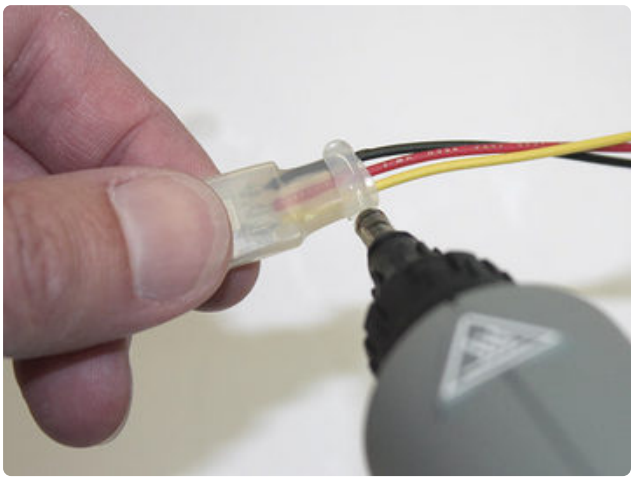
As the tube compresses, the glue will spread toward the tip, which might flare out a bit. That's fine.



Turn off the heat gun. While the glue and tubing are still hot, pinch it off tightly with needlenose pliers or a hemostat. Hold it there for about a minute until the glue solidifies.

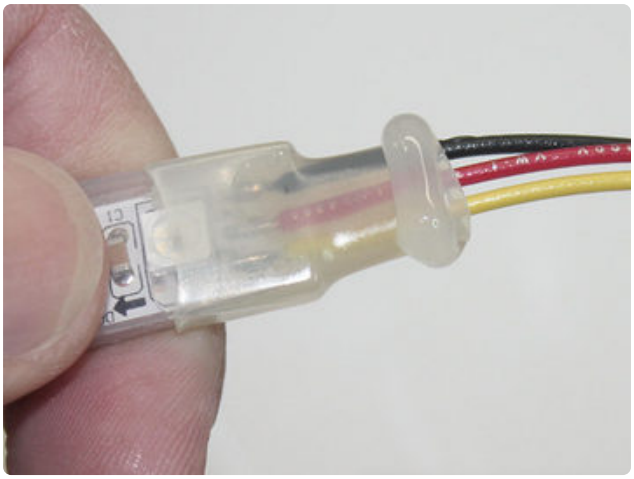


Once cooled, the flared end (and possibly a blob of glue) can be trimmed off for a clean, flat closure.



Going back to the end with the wires now...

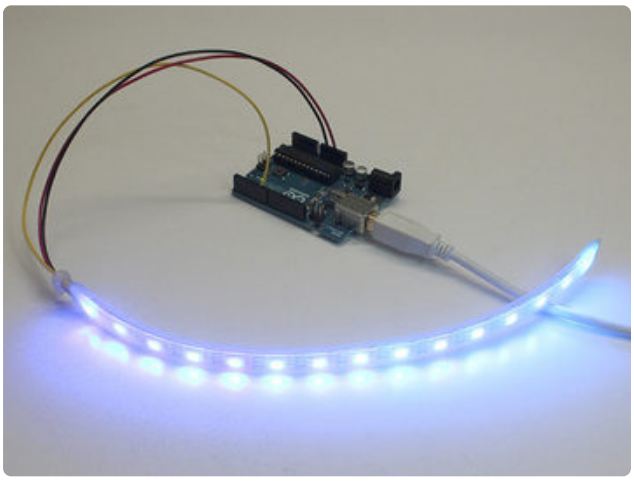
Add a bead of hot glue all the way around the end of the heat-shrink tubing.



Hold it for about a minute as it cools off. Don't set this down before then — it'll stick to things.

The goal is to create a sort of flange at the end that will later be used to secure this strip to another part.

Repeat the above steps for all of your LED strips.



After **all** of your LED strips are sealed, flanged and **fully cooled**, turn off your tools and run all the LED strips through the strandtest procedure again.

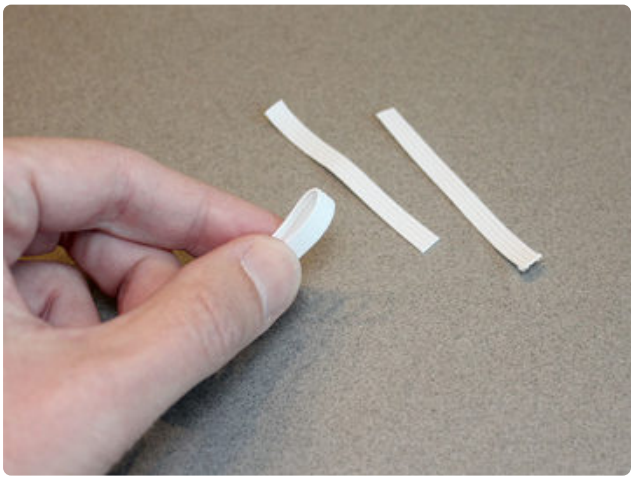
If you encounter a dud — if a solder connection broke or the strip was damaged from all the heating — your options are limited. Sometimes it's easiest just to make another, if you have extra LED strip available. Otherwise you can try sacrificing the first LED...cut off the wires and trim the first LED, then strip and re-solder the wires to the input of the next LED and re-test. If it works, seal it up and test again. It'll be one pixel shorter than before, but most likely nobody will notice the difference.

Whew! Take a break. Make sure your glue gun and soldering iron are turned off.

Assembling the Headpiece

Place a “do rag” on a foam wig head. If you plan to wear the finished piece with goggles (https://adafru.it/cGs), add those to the head too and mark above the strap with chalk, all the way around. This way you won’t install dreads in the area where a strap needs to be.

Aside from looking cool in their own right, goggles can help hide the hairline in front.



Cut a whole bunch of pieces of cloth ribbon or elastic, each about 2 inches long. Fold these in half into little loops.

You’ll need a bunch of these...not just for the “active” dreads (those with LEDs installed), but extras for “passive” dreads as well.

The full wig gets quite “hairy” as it goes together, so we’ll demonstrate with just a few ribbons to help keep things clear.

Figure an arrangement for your “active” LED dreads first, then some extra “passive” dreads around them. The way these all hang, most will be anchored up toward the crown of the head, you probably won’t need any down by the goggles strap. Pin things in place temporarily as you work out these positions.

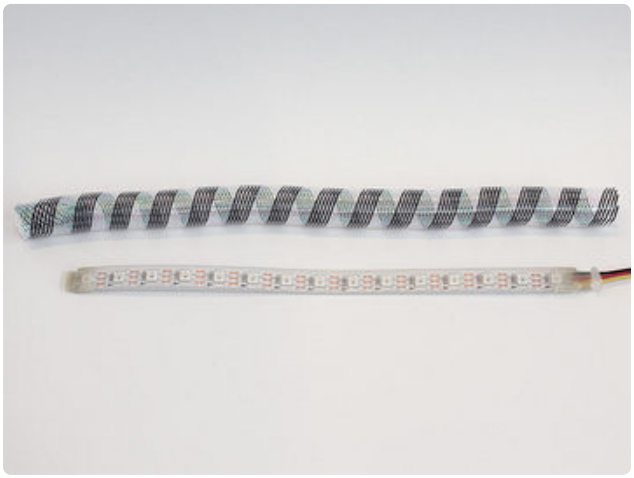
Once you’ve settled on an arrangement, the loops can be secured with basting glue or a couple of hand stitches.

After the basting glue has dried (or immediately, if you hand-stitched them), sew the loops securely to the do-rag. This is easiest with a sewing machine, but hand-sewing can work if you're patient.

Because the loops will be load bearing (supporting the weight of the LED strips), you want to make sure they're extra secure. Sew back and forth over each loop at least three times.

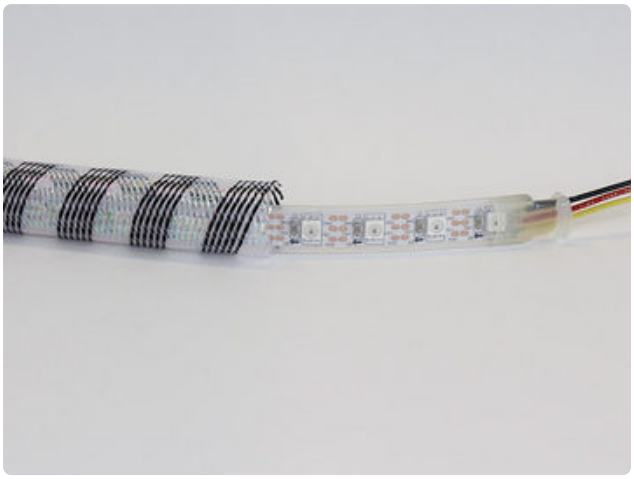
Finish all the sewing before proceeding with the next step. Just like the soldering, it's most efficient to work in batches rather than continually switching modes.

Mount "Active" Dreads

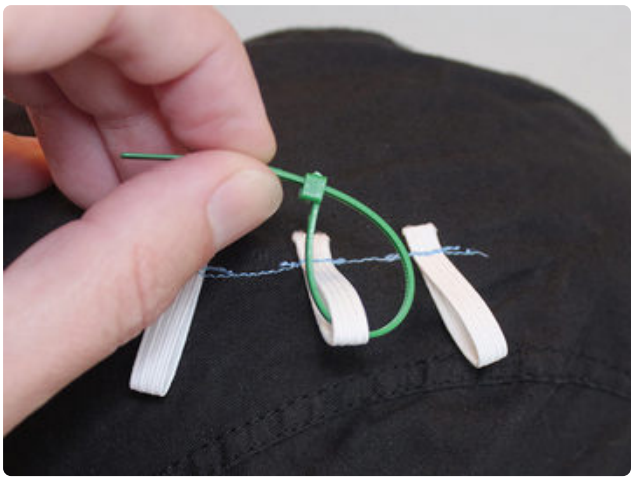


Cut a piece of tubular crinoline ribbon at least 2-3 inches longer than each of your LED strips.

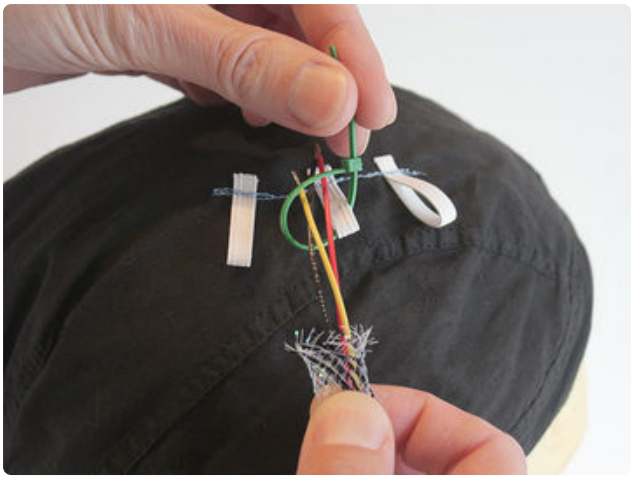
To keep the LED strips from poking out the bottoms of the dreads as they move around, pinch the end shut using a small cable tie, then clip away the excess “zipper” part of the tie. You can then push this up into the tube to hide the cable tie knot.



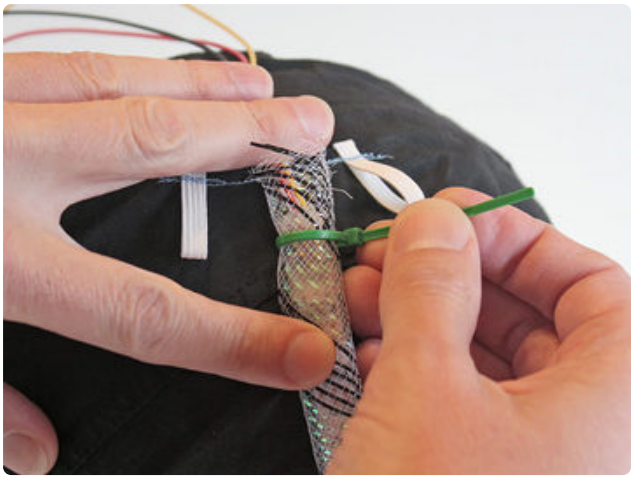
Slide all of the LED strips into the tubes.



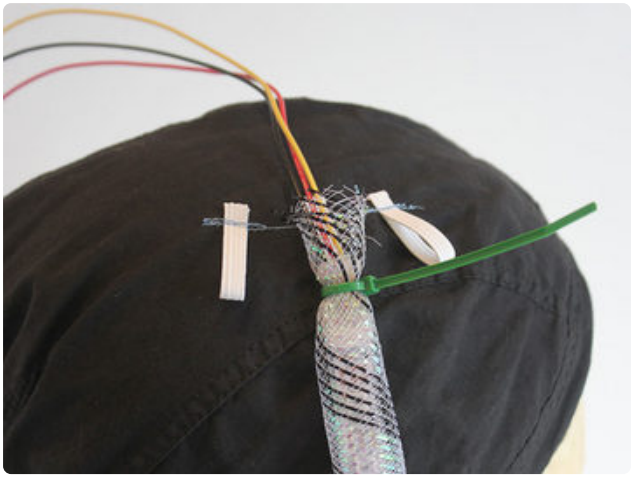
Put a small cable tie through one of the elastic loops, then start to close it on itself. Don’t cinch it down yet...keep it open, just the first click or two.



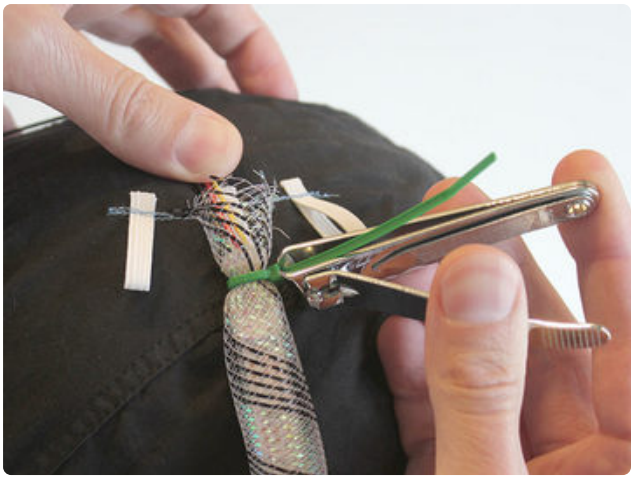
Bring the wires for one LED strand up through this cable tie ring.



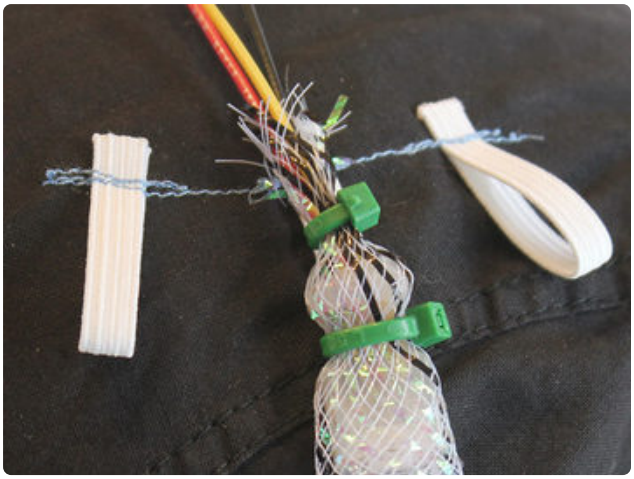
Feed the crinoline tube and base of the LED strip through the cable tie, then start to cinch it down. The cable tie should close down on the groove you created earlier — the narrow section between the LED strip and the hot glue flange.



Cinch it down tightly. There should be no wiggle between the dread and cable tie.



Then clip away the cable tie “zipper.” Wire cutters are okay for this, but I’ve found that nail clippers give a perfect, scratch-free edge.



Optionally, a second cable tie can be added just above the flange, to keep fraying under control.

Repeat these steps for all of your “active” dreads first. Once that’s done, you can position the microcontroller and start routing wires.

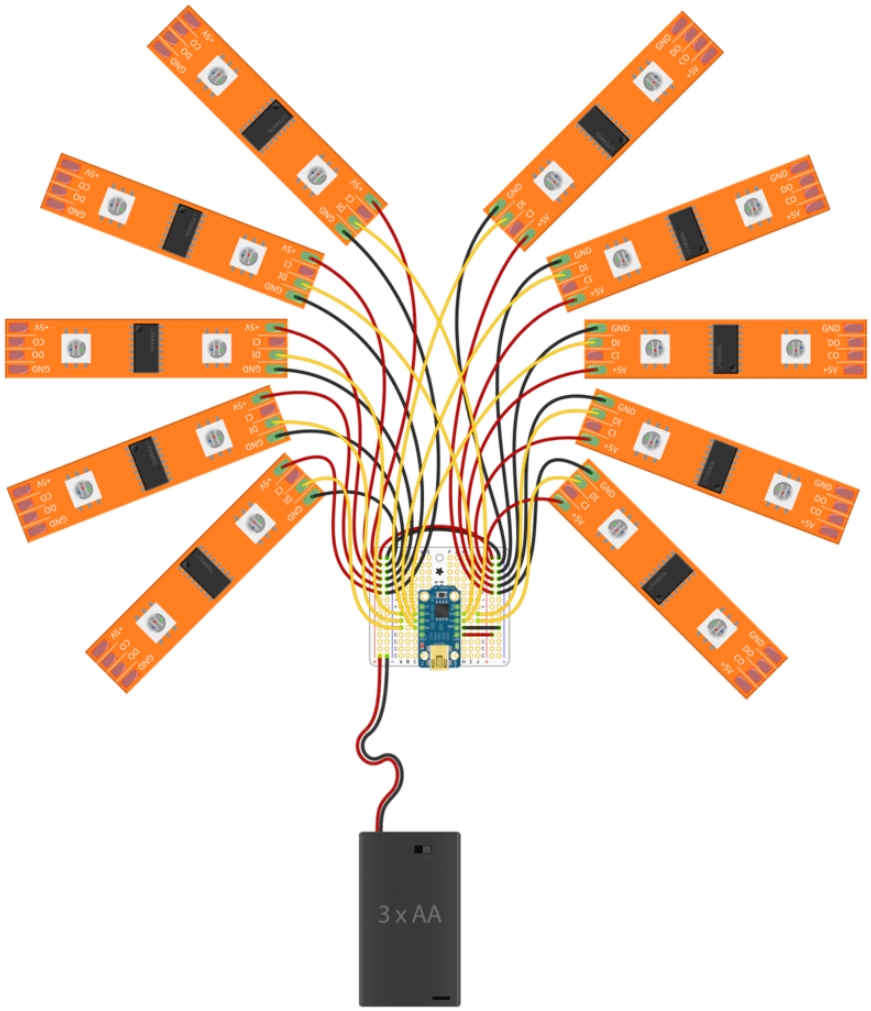
Install Trinket and Route Wires

Because Trinket is so tiny, we can actually mount it right on the wig...no need to run all these wires to a pocket (we’ll do that for just the battery).

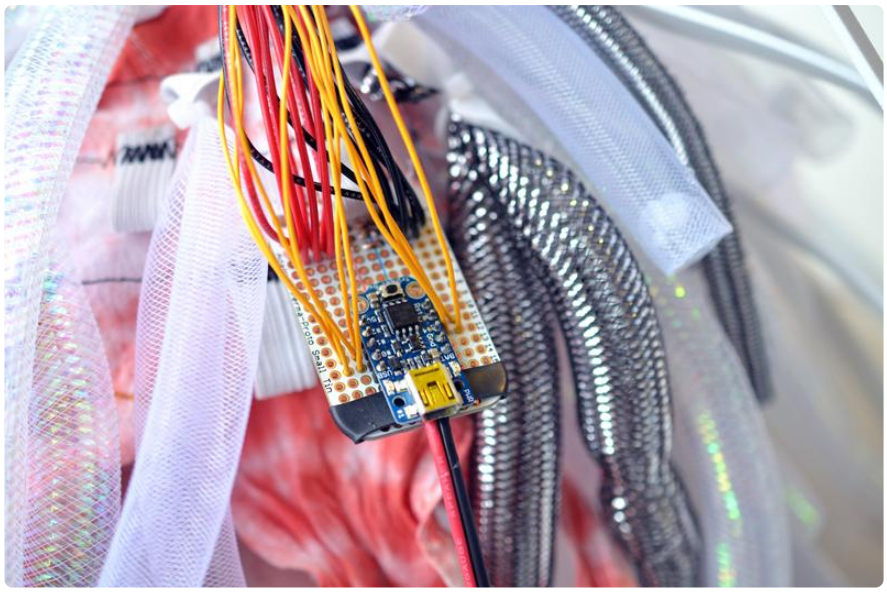
In order to distribute power and data to all of these strips, a small Perma Proto board was used. If you’d like to give everything a dry run first, you can hook things up temporarily on a breadboard, then transfer everything over to the Perma Proto once you’re satisfied.

Our wig used 10 LED strips. These were installed such that then “fan out” from a ring near the top of the head, looking like some sort of luminous deep-sea squid landed there. Viewed from the top (with the Trinket board at the back), it looks a bit like this:

This guide was written for the Trinket Mini board. The pins used are the same for the Trinket M0. We recommend the Trinket M0 as it is easier to use and is more compatible with modern computers!



(Diagram shows LPD8806 LED strips rather than NeoPixels, but the idea is the same.)



Position the Trinket/proto board at the back of the head where it will be less visible, then sew a few loops of thread through one of the board's mounting holes. This is just a temporary hold...you want to be able to flip the board over and solder underneath... we'll sew it down more securely later.

+ and – from the battery needs to reach all the strips and the Trinket board (the Bat+ and Gnd pins). Depending on the number of LED strips and the proto board configuration, you may need to bridge a few tracks together. Clip each wire to a more manageable length as you're routing them, then strip and solder them to the board (clipping any excess wire from underneath). It gets quite chaotic, hence the recommendation for color-coding the wires. Like all the prior steps, work methodically...for example, route all the ground wires, then all the +V wires, then signals.

The Trinket board only has five output pins, but our wig has ten LED strips. What we've done is connect two strips to each pin: the rear-most strip on the left and front-most strip on the right are both connected to Pin #0. Second rear-most left strip and second front-most right strip are connected to Pin #1, and so forth. You should be able to connect up to 3 or 4 strips per pin if necessary, but any more than that may become unreliable as the Trinket can only push so much current per pin.

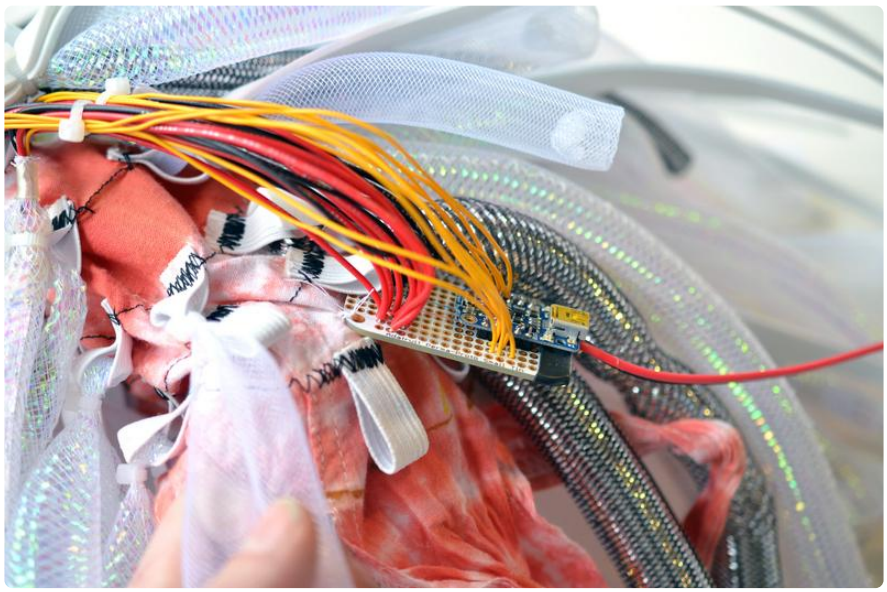
Some users have reported trouble with uploading code to the board when NeoPixels are attached to Pins #3 and/or #4. This is one good reason for testing with a breadboard first. If you can upload code with no problem, then that's great, you can install it permanently this way. If you encounter trouble, there are a couple of options:

- Add female socket headers to the proto board, so the Trinket can be removed for programming and then reinstalled. Or...

- Add a 2-pin JST connector to Pins #3 and #4, with the opposite plug connected to the corresponding LED strips. Unplug this connector before uploading code, then reconnect afterward.

The battery pack is relatively heavy and would pull the wig down, so we added about 3 feet of wire and hide the pack in a pocket. This is another good spot for a 2-pin JST connector or a DC power jack and plug set.

Test all the electronics using the code on the next page (or write your own) before wrapping up this stage. Once you're satisfied, the proto board can be stitched down more securely. If you'll be wearing this somewhere warm or might just be dancing and perspiring a lot, first cover the underside of the board with a couple layers of packing tape or glue down some craft foam to help prevent electrical shorts. Perspiration — being mostly salt water — is both conductive and corrosive!



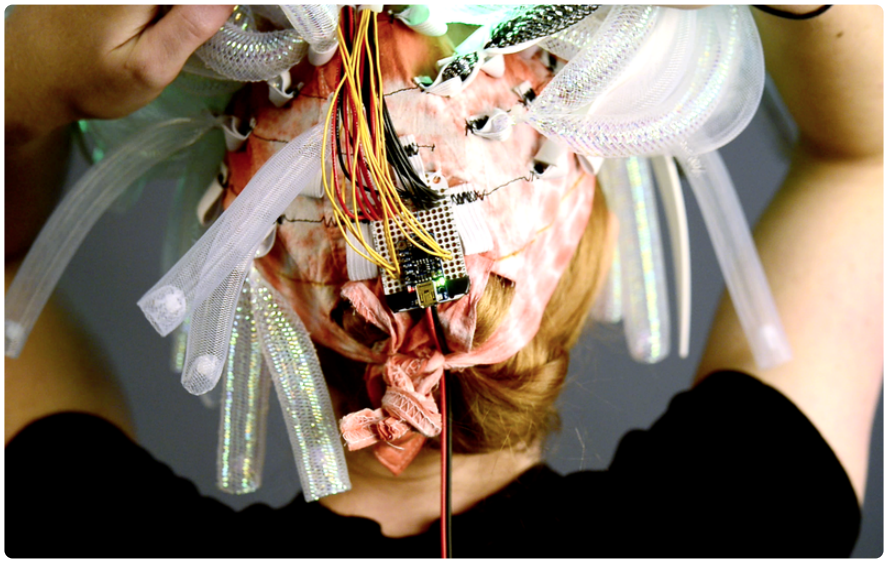
Mount "Passive" Dreads

For passive dreads, fraying at the ends can be controlled by simply pushing the tips inside-out on themselves, no need for cable ties.

You can make these twice as long, then pinch and fold them in the middle to do two dreads at once. Here we're doing it with just one tube...but these pinch down so small, you can do it with a whole handful all at once!

Continue filling out the empty spots in your wig until the desired level of poofyness is reached.

If you're running short of crinoline tubes, or if you just want to add some variety, you can mix it up with long strips of craft foam, colorful bits of ribbon, or lengths of flat ribbon cable for a technological theme.



Arduino Code

The Arduino code presented below works well on Trinket Mini boards. But if you have a Trinket M0 board you must use the CircuitPython code on the next page of this guide, no Arduino IDE required!

The software for this project is a slightly modified version of the Firewalker Sneakers (https://adafru.it/cDk) code...but running continuously rather than in response to a pressure sensor.

One of the major changes is the use of a timer interrupt, so the animation speed is consistent and smooth regardless of the current activity level. Setting up the timers involves accessing hardware registers directly, and as a result this code will work only on the Trinket. Changing this around for other boards requires some pretty deep programming skill.

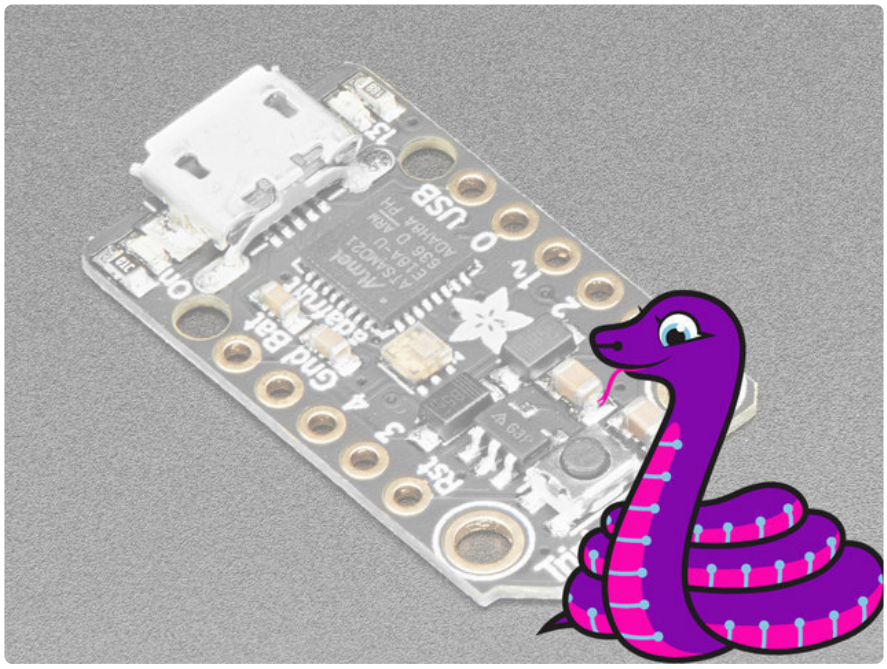
It also requires the NeoPixel library (https://adafru.it/aZU). If you've used the strandtest code to test the LEDs, then this is already installed. If you're a long-time NeoPixel user, make sure you've got the latest version installed — this code makes use of newer features.

```
// SPDX-FileCopyrightText: 2018 Mikey Sklar for Adafruit Industries
//
// SPDX-License-Identifier: MIT

// 'Cyber falls' sketch, adapted from code for Firewalker sneakers.
// Creates a fiery rain-like effect on multiple NeoPixel strips.
// Requires Adafruit Trinket and NeoPixel strips.  Strip length is
// inherently limited by Trinket RAM and processing power; this is
// written for five 15-pixel strands, which are paired up per pin
// for ten 15-pixel strips total.

#include <Adafruit_NeoPixel.h>
#include <avr/power.h>

const uint8_t gamma[] PROGMEM = { // Gamma correction table for LED brightness
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
    0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  1,  1,  1,  1,
    1,  1,  1,  1,  1,  1,  1,  1,  1,  2,  2,  2,  2,  2,  2,  2,
    2,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4,  4,  5,  5,  5,
    5,  6,  6,  6,  6,  7,  7,  7,  7,  8,  8,  8,  9,  9,  9, 10,
   10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16,
   17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 24, 25,
   25, 26, 27, 27, 28, 29, 29, 30, 31, 32, 32, 33, 34, 35, 35, 36,
   37, 38, 39, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50,
   51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68,
   69, 70, 72, 73, 74, 75, 77, 78, 79, 81, 82, 83, 85, 86, 87, 89,
   90, 92, 93, 95, 96, 98, 99,101,102,104,105,107,109,110,112,114,
  115,117,119,120,122,124,126,127,129,131,133,135,137,138,140,142,
  144,146,148,150,152,154,156,158,160,162,164,167,169,171,173,175,
  177,180,182,184,186,189,191,193,196,198,200,203,205,208,210,213,
  215,218,220,223,225,228,231,233,236,239,241,244,247,249,252,255 };

#define STRIPLEN 15 // Length of LED strips
#define MAXDROPS  5 // Max concurrent 'raindrops'
#define N_STRIPS  5 // Connect strips to pins 0 to (N_STRIPS-1)

Adafruit_NeoPixel strip = Adafruit_NeoPixel(STRIPLEN, 0);

// Everything's declared volatile because state changes inside
// an interrupt routine.
volatile struct {
  uint8_t  strip;
  int16_t  pos;
  uint8_t  speed;
  uint16_t brightness;
} drop[MAXDROPS];
volatile uint8_t
  nDrops    = 0,   // Number of 'active' raindrops
  prevStrip = 255; // Last selected strip
volatile uint16_t
  countdown = 0;   // Time to next raindrop
```

```
void setup() {

  // Set up Timer/Counter1 for ~30 Hz interrupt
#if F_CPU == 16000000L
  clock_prescale_set(clock_div_1);
  TCCR1  = _BV(PWM1A) |_BV(CS13) | _BV(CS12);             // 1:2048 prescale
#else
  TCCR1  = _BV(PWM1A) |_BV(CS13) | _BV(CS11) | _BV(CS10); // 1:1024 prescale
#endif

  // Turn strips off ASAP (must follow clock_prescale_set)
  strip.begin();
  for(uint8_t s=0; s<N_STRIPS; s++) {
    strip.setPin(s);
    strip.show();
  }

  // Finish timer/counter setup
  OCR1C  = 255;        // ~30 Hz
  GTCCR  = 0;          // No PWM out
  TIMSK |= _BV(TOIE1); // Enable overflow interrupt
}

void loop() { } // Not used -- everything's in interrupt below

// A timer interrupt is used so that everything runs at regular
// intervals, regardless of current amount of motion.
ISR(TIMER1_OVF_vect) {

  uint16_t mag[STRIPLEN];
  uint8_t  s, d, p, r, g, b;
  int      mx1, m, level;

  if(countdown == 0) {         // Time to launch new drop?
    if(nDrops < MAXDROPS-1) {  // Is there space for one in the list?
      do {
        s = random(N_STRIPS);
      } while(s == prevStrip); // Don't repeat prior strip
      drop[nDrops].strip      = prevStrip = s;
      drop[nDrops].pos        = -32; // Start off top of strip
      drop[nDrops].speed      = 1 + random(3);
      drop[nDrops].brightness = 250 + random(520);
      nDrops++;
      countdown = 9 + random(28); // Time to launch next one
    }
  } else countdown--;


  for(s=0; s<N_STRIPS; s++) { // For each strip...
    memset(mag, 0, sizeof(mag)); // Clear magnitude table

    // Render active drops for this strip into magnitude table
    for(d=0; d<nDrops; d++) {
      if(drop[d].strip == s) {
        for(p=0; p<STRIPLEN; p++) { // For each pixel...
          mx1 = (p << 2) - drop[d].pos; // Position of LED along wave
          if((mx1 <= 0) || (mx1 >= 32)) continue; // Out of range
          if(mx1 > 24) { // Rising edge of wave; ramp up fast (2 px)
            m = ((long)drop[d].brightness * (long)(32 - mx1)) >> 4;
          } else { // Falling edge of wave; fade slow (6 px)
            m = ((long)drop[d].brightness * (long)mx1) / 24;
          }
          mag[p] += m; // Accumulate result in magnitude buffer
        }
      }
    }

    // Remap magnitude table to RGB for strand
```

```
    for(p=0; p<STRIPLEN; p++) {      // For each pixel in strip
      level = mag[p];                // Pixel magnitude (brightness)
      if(level < 255) {              // 0-254 = black to green-1
        r = b = 0;
        g = pgm_read_byte(&gamma[level]);
      } else if(level < 510) {       // 255-509 = green to yellow-1
        r = pgm_read_byte(&gamma[level - 255]);
        g = 255;
        b = 0;
      } else if(level < 765) {       // 510-764 = yellow to white-1
        r = g = 255;
        b = pgm_read_byte(&gamma[level - 510]);
      } else {                       // 765+ = white
        r = g = b = 255;
      }
      strip.setPixelColor(p, r, g, b);
    }

    strip.setPin(s); // Select output pin
    strip.show();    // Strips don't need to refresh in sync
  }

  // Move 'active' drops
  for(d=0; d<nDrops; d++) {
    drop[d].pos += drop[d].speed;
    if(drop[d].pos >= (STRIPLEN * 4)) { // Off end?
      // Remove drop from list (move last one to this place)
      memcpy((void *)&drop[d], (void *)&drop[nDrops-1], sizeof(drop[0]));
      nDrops--;
    }
  }
}
```

CircuitPython Code



Trinket M0 boards can run **CircuitPython** — a different approach to programming compared to Arduino sketches. In fact, **CircuitPython comes factory pre-loaded on Trinket 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** (https://adafru.it/19oC)Trinket M0 guide **(https://adafru.it/19oC)**.

These directions are specific to the "M0" boards. The original Trinket Mini 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 Trinket 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 Trinket M0 doesn't show up as a drive, follow the Trinket M0 guide link above to prepare the board for CircuitPython.

```
# SPDX-FileCopyrightText: 2018 Mikey Sklar for Adafruit Industries
#
# SPDX-License-Identifier: MIT

# 'Cyber falls' sketch
# Creates a fiery rain-like effect on multiple NeoPixel strips.
# Requires Adafruit Trinket and NeoPixel strips.  Strip length is
# inherently limited by Trinket RAM and processing power; this is
# written for five 15-pixel strands, which are paired up per pin
# for ten 15-pixel strips total.

import time
import board
import neopixel
import adafruit_fancyled.adafruit_fancyled as fancy

num_leds = 15       # number of LEDs per strip
saturation = 255    # 0-255, 0 is pure white, 255 is fully saturated color
blend = True        # color blending between palette indices
brightness = 0.5    # half brightness the range is 0.0 - 1.0
concurrent = 3      # number of LEDs on at a time
on_time = 0.04      # 0.04 seconds == 40 milliseconds

# NeoPixel objects using all five Trinket M0 GPIO pins 0-4
drop0 = neopixel.NeoPixel(board.D0, num_leds)
drop1 = neopixel.NeoPixel(board.D1, num_leds)
drop2 = neopixel.NeoPixel(board.D2, num_leds)
drop3 = neopixel.NeoPixel(board.D3, num_leds)
drop4 = neopixel.NeoPixel(board.D4, num_leds)

# list of neopixel strips
drop_list = [drop0, drop1, drop2, drop3, drop4]

def led_drops(strip):

    # FancyLED allows for mixing colors with palettes
    palette = [fancy.CRGB(200, 255, 200),       # lighter (more white) green
               fancy.CRGB(0, 255, 0)]           # full green
```

```
    for i in range(num_leds):
        # FancyLED can handle the gamma adjustment, brightness and RGB settings
        color = fancy.palette_lookup(palette, i / num_leds)
        color = fancy.gamma_adjust(color, brightness=brightness)
        strip[i] = color.pack()

        # turn off the LEDs as we go for raindrop effect
        if  i >= concurrent:
            strip[i - concurrent] = (0,0,0)

        if i >= num_leds - 1:
            for j in range(concurrent,-1,-1):
                strip[i-j] = (0,0,0)
                time.sleep(on_time)

        time.sleep(on_time)

while True:

    # loop through each neopixel strip in our list
    for drops in drop_list:
        led_drops(drops)
```

Installing Libraries:

This code requires two libraries be installed:

- **neopixel**
- **adafruit_fancyled**

A factory-fresh board will have the neopixel library already installed. If you've just reloaded the board with CircuitPython, create the "lib" directory and then copy in the neopixel.mpy and adafruit_fancyled folder from the latest release of the Adafruit_CircuitPython_Bundle (https://adafru.it/CeF).

The FancyLED library (https://adafru.it/CeG) being using in this CircuitPython example is not the same as the FastLED (https://adafru.it/ezj)used for Arduino. FancyLED has a subset of FastLED features and some different syntax. The FancyLED tutorial provides an excellent overview (https://adafru.it/AKx).

The file system layout on your gemma M0 should look like this:

```
$ pwd
/Volumes/CIRCUITPY
$ find .
.
./boot_out.txt
./.fseventsd
./.fseventsd/fseventsd-uuid
./lib
./lib/neopixel.mpy
./lib/adafruit_fancyled
./lib/adafruit_fancyled/adafruit_fancyled.mpy
```

```
./lib/adafruit_fancyled/fastled_helpers.mpy
./main.py
```