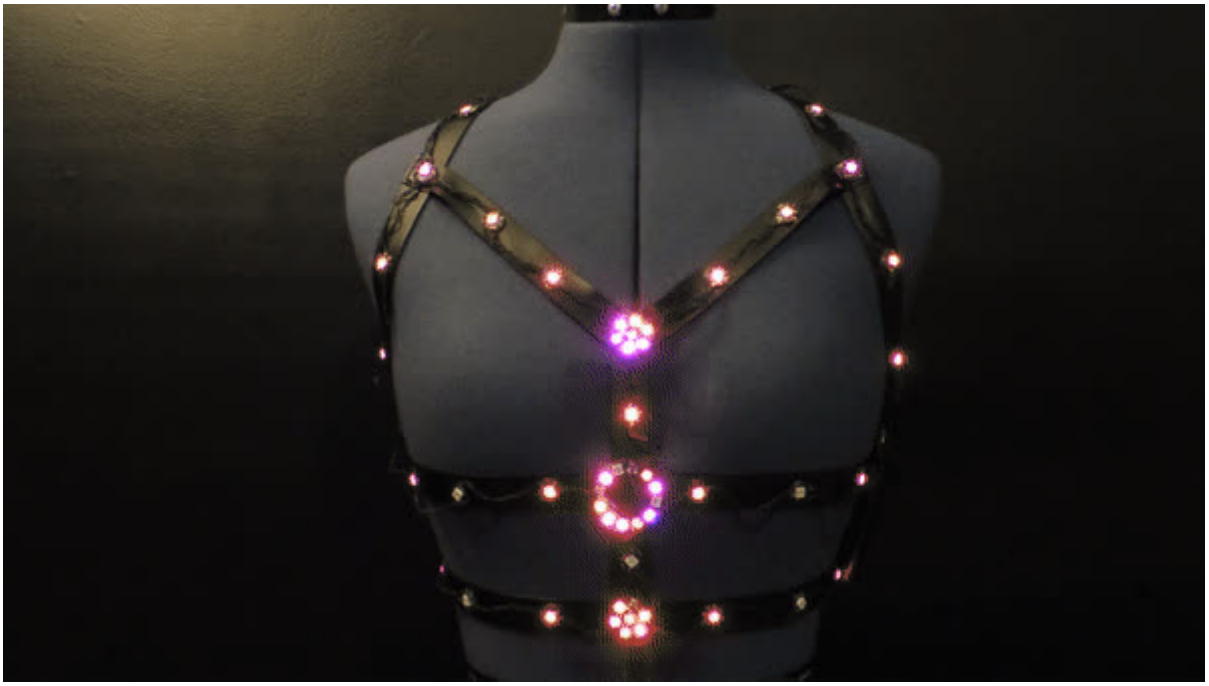
LED Harness Bra

Created by Amelia T



https://learn.adafruit.com/neopixel-led-harness-bra

Last updated on 2024-03-08 03:12:28 PM EST

Table of Contents

Overview 3

Materials & Tools 4

- Parts
- Tools
- Sewing and Other Supplies

Harness 9

Prep 10

- Halter Piece
- Upper Horizontal Piece
- Lower Horizontal Piece
- The "V"
- Vertical Piece

Sewing Tips 12

Construction 14

- Attach halter to horizontal pieces
- Attach vertical & V pieces

Code 16

- Arduino IDE
- Code

LEDs 26

- Cut & Strip Wires
- Make the Data Line
- Power & Ground
- Braid

Circuit Diagram 28

Data Line 29

- Sections
- Bring it all together

Power & Ground 32

- Wires
- Solder, Part 1
- Braid
- Solder, Part 2
- Bring it all Together

Final Assembly and Wear 35

- Attach to Harness
- Battery Pocket
- Wear It!

Overview

Construct a custom-fit harness bra with a unique array of NeoPixels and a Circuit Playground Express.

Caged clothing accessories can be worn over anything from a tank top to long sleeves. Harnesses come in all shapes and sizes and although they may accentuate certain body parts, they can be worn by anyone.

In the mood for a lot of soldering? This project requires quite a bit of soldering. Be patient with yourself throughout the build!

The guide uses a Circuit Playground Express, so there are unlimited possibilities for turning your harness into a unique wearable LED accessory.

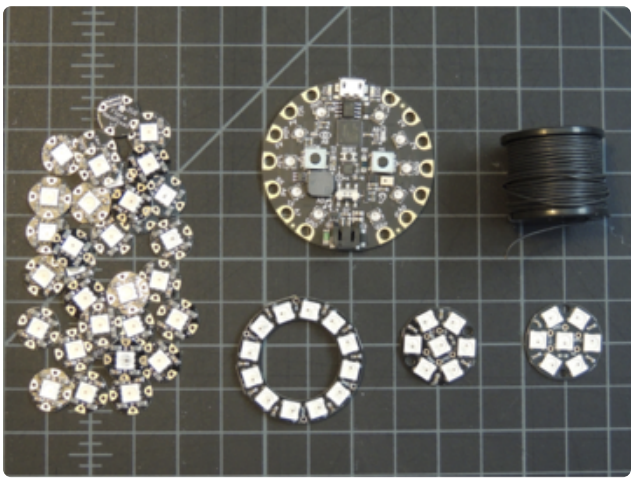
Make sure you're familiar with the following guides:

- Adafruit Neopixel Uberguide (https://adafru.it/dhw)
- Adafruit Circuit Playground Express (https://adafru.it/adafruit-cpx)

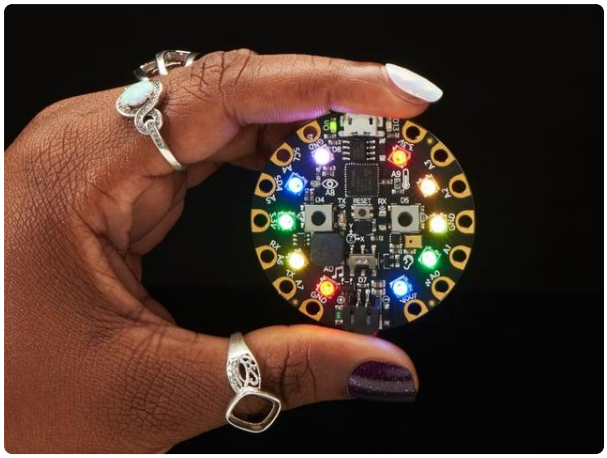
Materials & Tools

Parts

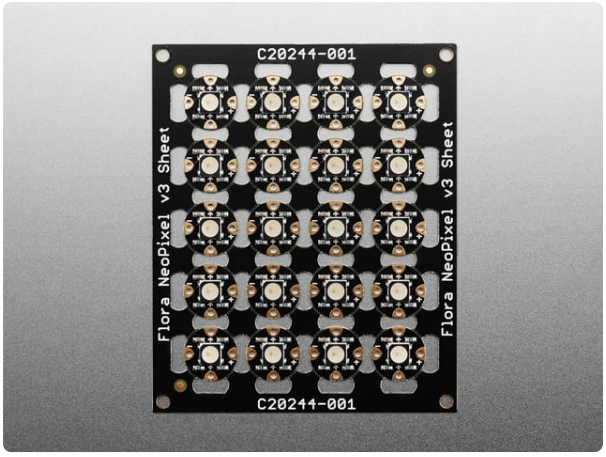
The following parts are the electronic parts for the project from Adafruit. Note the number of FLORA LEDs, you will need multiple, depending on your project.



1 x Adafruit Circuit Playground Express (http://adafru.it/3333)
32 x FLORA RGB NeoPixels (http://adafru.it/1260)
2 x NeoPixel Jewel (http://adafru.it/2226)
1 x NeoPixel Ring - 12 LEDs (http://adafru.it/1643)
Silicone Cover Stranded-Core Wire 30AWG Black (http://adafru.it/3164)
1 x Lithium Ion Polymer Battery (http://adafru.it/328) or AAA battery pack (http://adafru.it/727)
JST Extension Cable with an On/Off switch (http://adafru.it/3064) or JST-PH Extension Cable (http://adafru.it/1131)

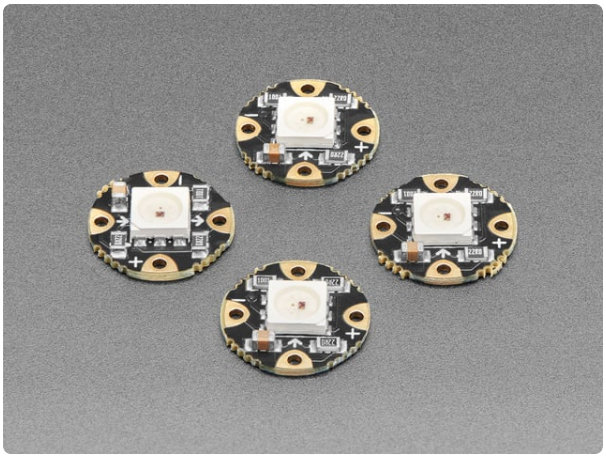


Circuit Playground Express
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...
https://www.adafruit.com/product/3333



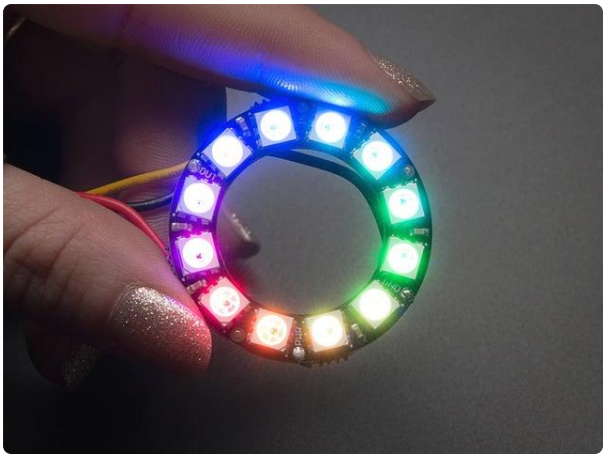
Flora RGB Smart NeoPixel version 3 - Sheet of 20

So, you want lots and lots of NeoPixels? And you want them for less? Not a problem! Here's a sheet of Flora NeoPixels fresh from the (reflow) oven. Cut them off as you need 'em...
https://www.adafruit.com/product/1559



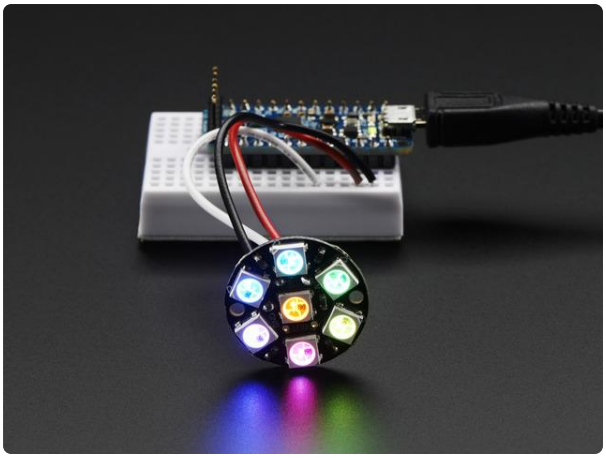
Flora RGB Smart NeoPixel version 2 - Pack of 4

What's a wearable project without LEDs? Our favorite part of the Flora platform is these tiny smart pixels. Designed specifically for wearables, these updated Flora NeoPixels have...
https://www.adafruit.com/product/1260



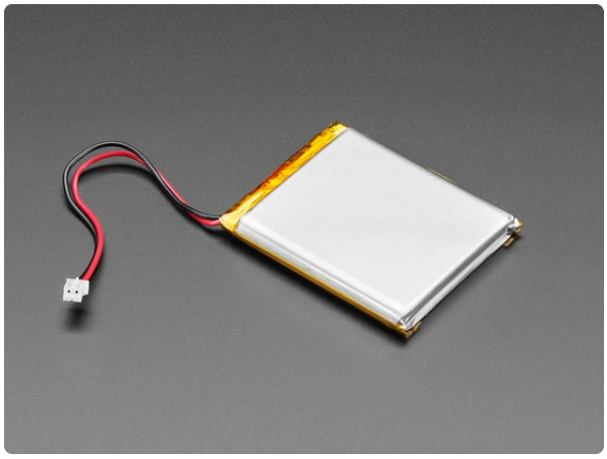
NeoPixel Ring - 12 x 5050 RGB LED with Integrated Drivers

Round and round and round they go! 12 ultra bright smart LED NeoPixels are arranged in a circle with 1.5" (37mm) outer diameter. The rings are 'chainable' - connect the...
https://www.adafruit.com/product/1643



NeoPixel Jewel - 7 x 5050 RGB LED with Integrated Drivers

Be the belle of the ball with the NeoPixel Jewel! We fit seven of our tiny 5050 (5mm x 5mm) smart RGB LEDs onto a beautiful, round PCB with mounting holes and a...
https://www.adafruit.com/product/2226

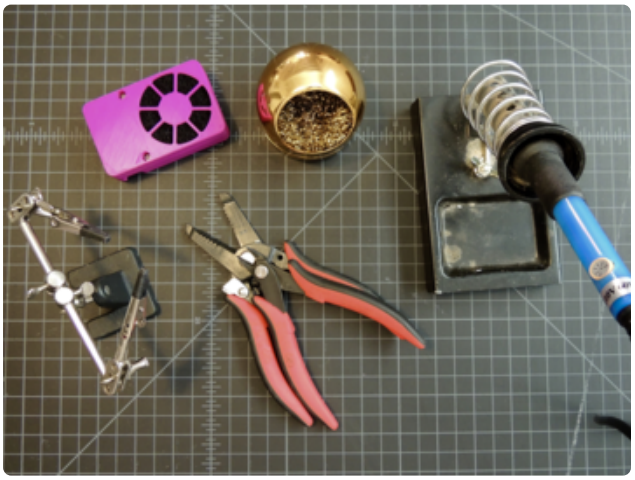


[Lithium Ion Polymer Battery - 3.7v 2500mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/328

[JST-PH Battery Extension Cable - 500mm]
By popular demand, we now have a handy extension cord for all of our JST PH-terminated battery packs (such as our LiIon/LiPoly and 3xAAA holders). One end has a JST-PH compatible...
https://www.adafruit.com/product/1131

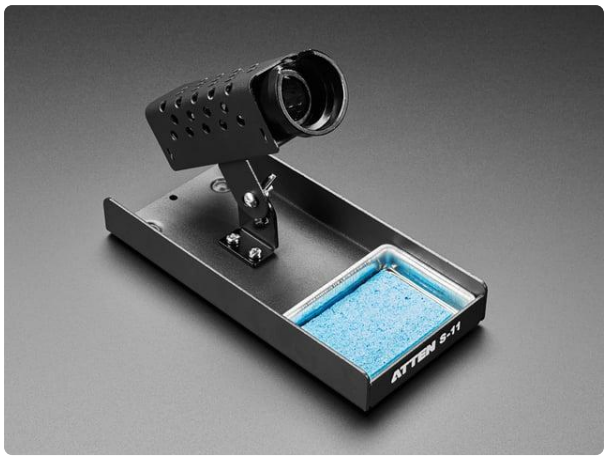
[JST 2-pin Extension Cable with On/Off Switch - JST PH2]
By popular request - we now have a way you can turn on-and-off Lithium Polymer batteries without unplugging them.This PH2 Female/Male JST 2-pin Extension...
https://www.adafruit.com/product/3064

Tools



Soldering iron - 120V US (http://adafru.it/3685), 220V UK (http://adafru.it/3759), 220V EU (http://adafru.it/3760)
Soldering iron stand (http://adafru.it/150)
Solder
Wire stripper (http://adafru.it/527)
Helping hands (http://adafru.it/291)
Solder Tip Cleaner (http://adafru.it/11721)
Ventilation: the USB Rechargeable Mini Solder Fume Extractor (https://adafru.it/CMX) from Phillip Burgess (https://adafru.it/iPc)

Hakko Professional Quality 20-30 AWG Wire Strippers
These are the finest wire strippers we have used, and if you have to do a lot of wiring, you will agree! They have soft rounded grips - very comfortable to use, and precision ground...
https://www.adafruit.com/product/527



Soldering iron stand
A real stand with sponge and solidly-built sheet metal holder prevents your iron from 'rolling away' or burning a hole in the table. If you're starting out and have a...
https://www.adafruit.com/product/150

Helping Third Hand Magnifier W/ Magnifying Glass Tool

The classic 'third hand tool,' as seen on every desk! We have one next to our Panavise jr, they complement each other well. This...

https://www.adafruit.com/product/291

Sewing and Other Supplies

Pleather Trim (https://adafru.it/D4U)

Scrap fabric for a battery holder

Black standard thread

Elastic – ¾” wide

Double sided tape

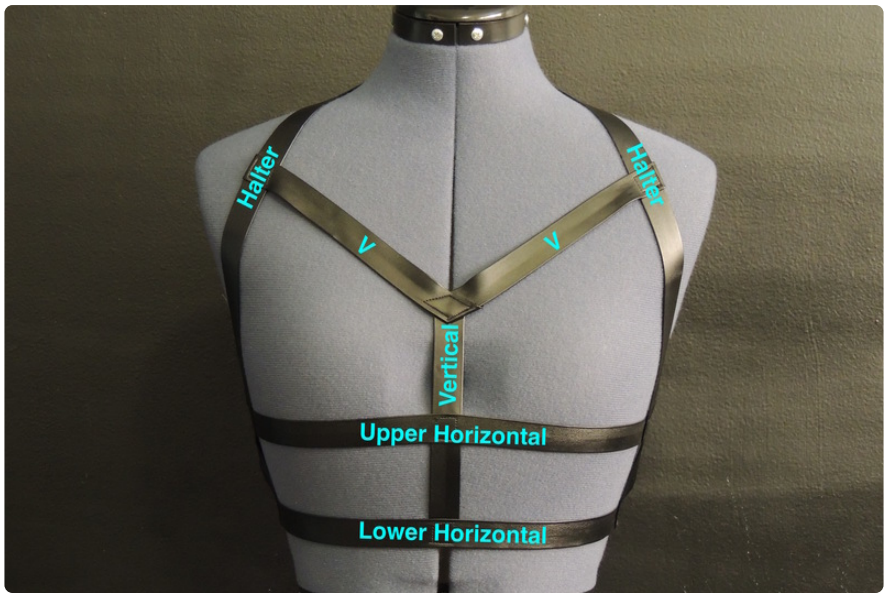
Temporary adhesive / blue tack

E6000 or hot glue

- Sewing tools and supplies – a sewing machine (https://adafru.it/D4V), scissors, straight pins, tape measure, fabric chalk, etc.
- Dress form (https://adafru.it/D4W) – Since a generic dress form may not match your proportions, you can always build one from scratch using one of these tutorials (https://adafru.it/D4X)

Harness

In this part of the guide, you'll measure and cut pieces of pleather to create a custom harness. This simple harness design is comprised of 6 pieces which I'll name for this guide: The halter, upper-horizontal, lower-horizontal, vertical, and a "V" (left & right) on top.



Prep

Halter Piece

Measure the distance from the left side of your waist, over your neck, and down to the right side of your waist. Label this piece and mark the halfway point.

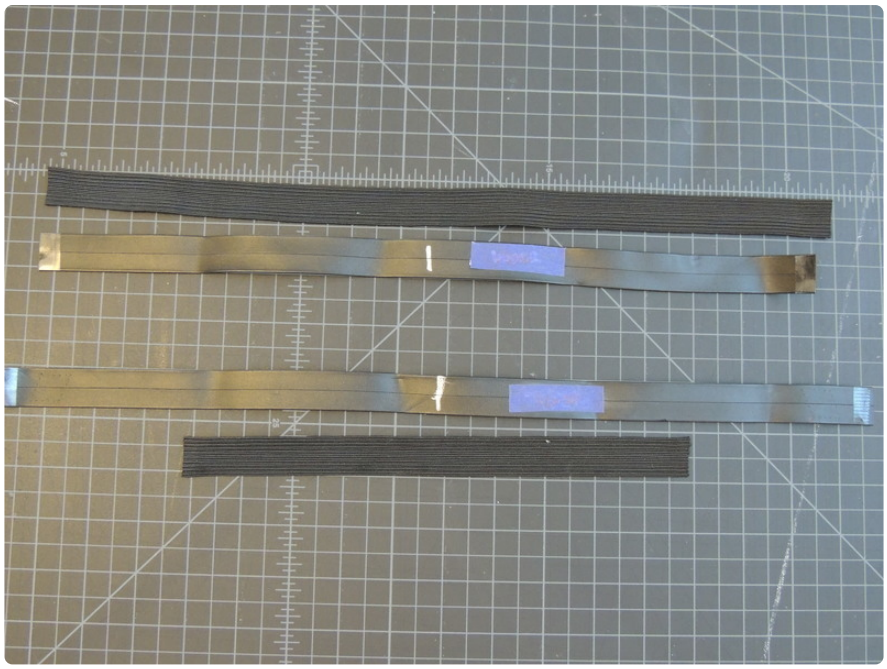
leds_wearables_HalterMeasure.gif

Upper Horizontal Piece

Measure around your body directly under your chest or wherever you'd like the upper horizontal piece to sit. Cut half of this length in pleather and the other half in elastic. Label and mark the halfway point on the piece of pleather.

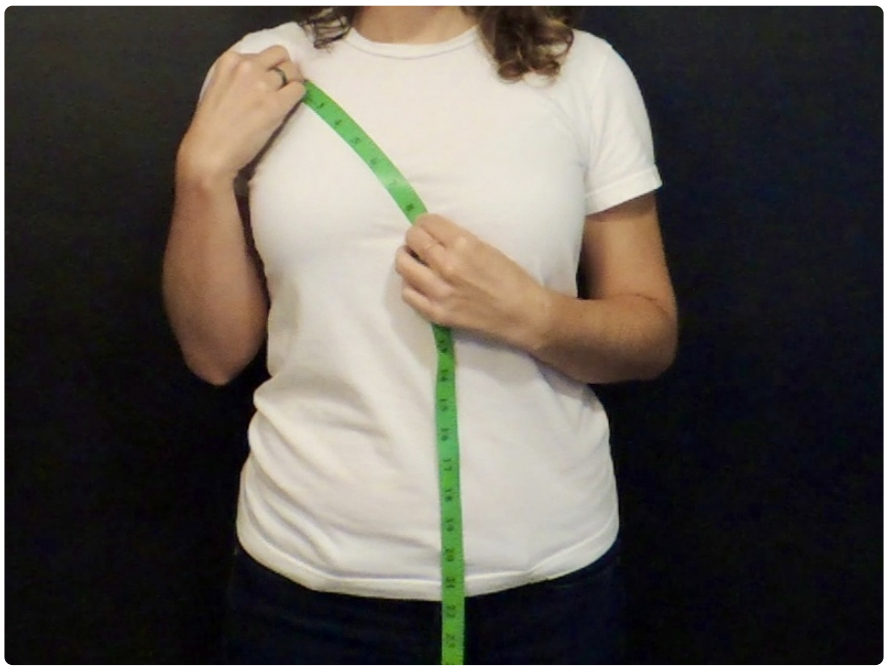
Lower Horizontal Piece

Measure your waist or wherever you'd like the lower horizontal piece to sit. Cut 2/3 of this length in pleather and 1/3 in elastic. Label and mark the halfway point on the piece of pleather.



The "V"

I measured between where the halter piece hit my clavicle to where I wanted the V to hit the center of my chest. Add 1” and cut two pieces.



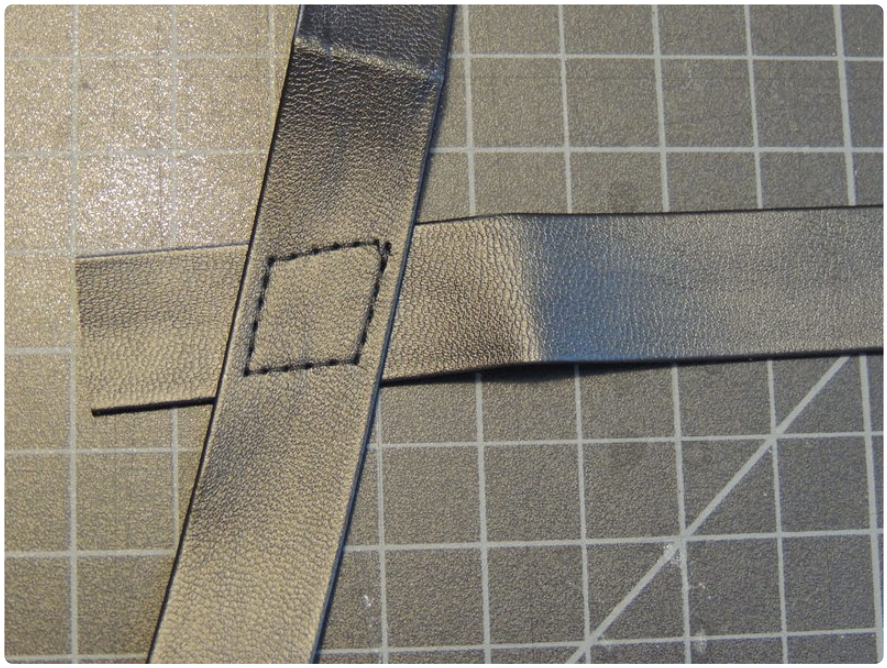
Vertical Piece

Measure from your waistline directly up to the V. Add 1” and cut.

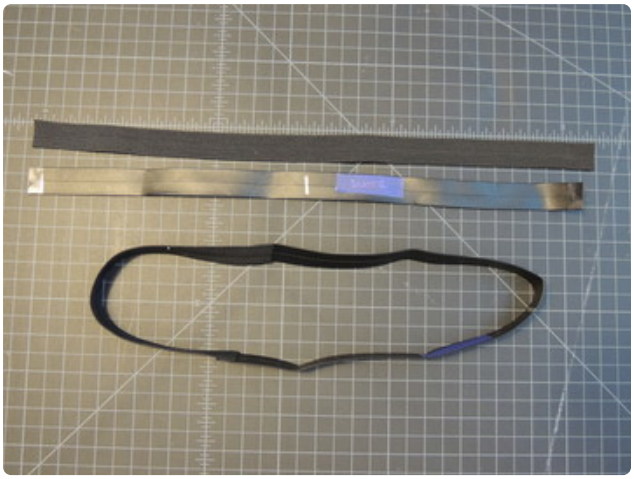
Sewing Tips

Here are some tips on using pleather:

- If you can't find pleather trim, cut strips ¾” or 1” wide out of a non-stretch pleather fabric.
- Test sew two pieces of pleather trim together to determine the best needle, stitch settings, and tension for your machine.
- Use pins or a small piece of double sided tape to hold pleather together.
- When sewing the pleather together, you may need to use something as a stabilizer. Tissue paper works great. (https://adafru.it/D4Y) Note: I did not need stabilizer for the pleather trim, but I did for raw pleather.



Construction



Start with the pleather and elastic for the upper horizontal. Overlap the two pieces by ½" & sew together.

Repeat for lower horizontal pleather and elastic.

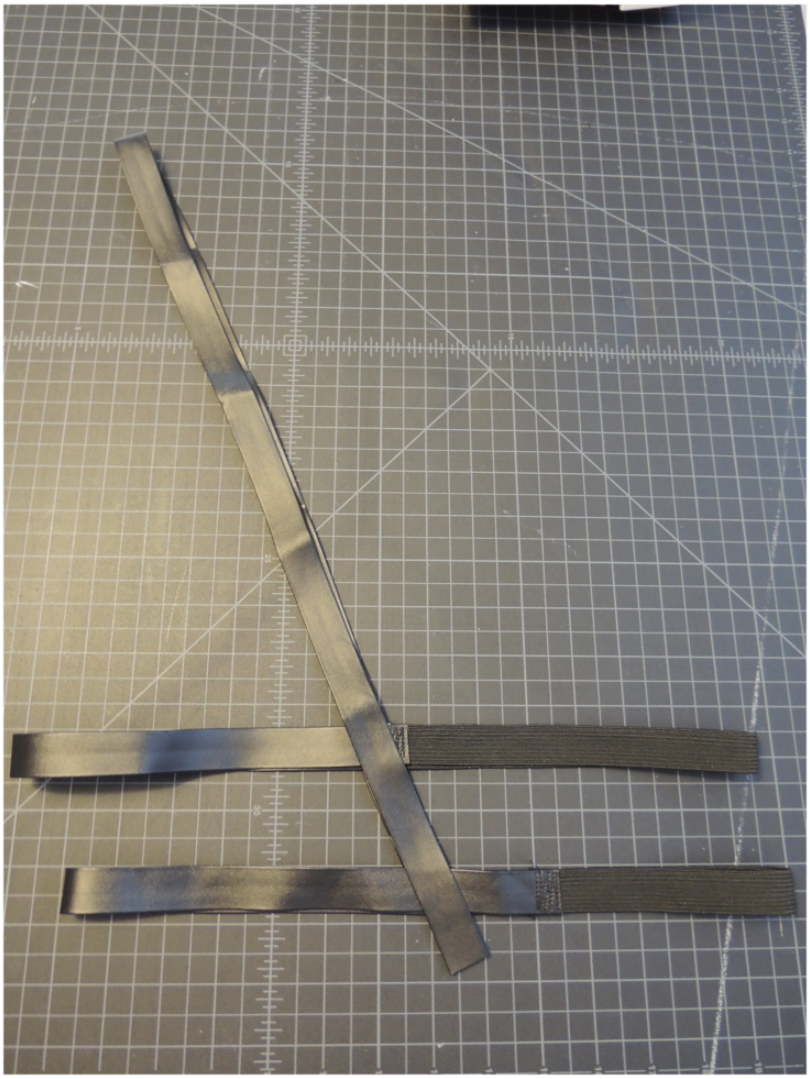
Attach halter to horizontal pieces

With the upper and lower horizontal pieces on your body or the dress form, center the harness halter piece over your neck.

Determine where the pieces should overlap. Attach with pins or a small piece of double sided tape between the strips of fabric.

Measure both sides off body to ensure things are still even and adjust accordingly.

Sew together. Try on and if there are any issues, pick the stitches out and try again.



Attach vertical & V pieces

Align the one end of the vertical piece with the bottom center of the lower horizontal piece. Align the upper horizontal piece with the vertical piece, so that the horizontal pieces are parallel. Sew together.

Align one piece of the V just below the right clavicle. Pin together or use a small piece of double sided tape between the strips of fabric. Measure to determine the right location for the left piece of the V. Pin together or use a small piece of double sided tape between the strips of fabric.

Try it on again and attach to the vertical piece.

Trim any excess fabric or threads.

Code

Arduino IDE

Open your Arduino IDE. If you are new to Arduino, check out this guide: Adafruit Arduino IDE Setup (https://adafru.it/CMS).

This sketch uses three libraries: the FastLED library, the Adafruit Circuit Playground microphone library, and the Adafruit Arduino Zero / Feather M0 I2S audio library. See FastLED.io (https://adafru.it/ebm) for more information, the FastLED github (https://adafru.it/ezj), or visit their community on Reddit (https://adafru.it/Fr-). You can learn more about Arduino libraries by reading this guide from Adafruit (https://adafru.it/dit).

Download the Circuit Playground library if you haven't already done so. See the Circuit Playground Express guide (https://adafru.it/BeF) to learn more about the board.

In this sketch, we'll utilize some of the supporting libraries used by the Circuit Playground.

Code

Download the sketch in its entirety or take the following steps to ensure you have the right files for running the sketch.

Open a new sketch in your Arduino IDE & save it as a new folder. Find the folder just created by your new sketch. Copy & paste the following files from your Circuit Playground utilities folder into the newly created folder:

- Adafruit_CPlay_Mic.cpp
- Adafruit_CPlay_Mic.h
- Adafruit_ZeroPDM.cpp
- Adafruit_ZeroPDM.h

Plug the Adafruit Circuit Playground Express into your computer with a USB cable, verify and upload the following code.

Once you have uploaded your code, test it out on your Adafruit Circuit Playground Express. Although it will think that there are more LEDs attached, click through the modes and test the switch to make sure it all works.

```
// SPDX-FileCopyrightText: 2018 Amelia Tetterton for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/*
HarnessBra by Amelia Tetterton as of 11/15/18

Summary:
  -Use the left button (D4) to cycle through some fun FastLED examples.
    -Create your own twinkles and adapt the BPM mode using FastLED palettes
    or create and name your own palettes!
  -Use the right button (D5) to activate the pretty vu meter from the Circuit
Playground
  pretty_meter example sketch, adapted for this strand.
  -Use the slide-switch (D7) to turn off the LEDs. This does not turn the board off.
*/
#include <FastLED.h>
#include <Adafruit_CircuitPlayground.h>

// Circuit Playground Setup-----------------------------------------------------
#define CP_PIN      8         //CPX neopixels live on pin 8, CP live on pin 17
#define NUM_CP      10        //number of neopixels on the CP

const int switchPin = 7;      // the pin for the slideswitch
const int leftButtonPin = 4;  // the pin for the Left Button
const int rightButtonPin = 5; // the pin for the Right Button

bool switchState;
bool turnedOn;
```

```
bool PrevleftButton = false;
bool PrevrightButton = false;

bool leftButton = false;
bool rightButton = false;

// Strip Setup-----------------------------------------------------------------
#define LED_PIN     A1      //led strand is soldered to pin A1
#define NUM_STRIP   37      //number of LEDs called in my strand

#define LED_TYPE    WS2812B
#define COLOR_ORDER GRB
#define BRIGHTNESS  32  // 255 is full brightness
#define NUM_LEDS NUM_STRIP + NUM_CP

CRGB leds[NUM_LEDS];

CRGBPalette16 currentPalette;
CRGBPalette16 gPalette;
TBlendType    currentBlending;
uint8_t gHue = 0; // rotating "base color" used by many of the patterns
uint8_t BeatsPerMinute = 62; //for the FastLED BPM modes
int ledMode = 0;

#define NUM_MODES 9  // change this number if you add or subtract modes

#define UPDATES_PER_SECOND 100
#define SATURATION  255 // 0-255, 0 is pure white, 255 is fully saturated color
#define SPEED       100 // How fast the colors move. Higher numbers = faster motion
#define STEPS       3   // How wide the bands of color are.
                        //1 = more like a gradient, 10 = more like stripes

// TWINKLE SETUP ---------------------------------------------------------------
#define STARTING_BRIGHTNESS 255
#define FADE_IN_SPEED       80
#define FADE_OUT_SPEED      60
#define DENSITY             255

// SOUND REACTIVE SETUP --------------------------------------------------------

// To keep the display 'lively,' an upper and lower range of volume
// levels are dynamically adjusted based on recent audio history, and
// the graph is fit into this range.

#define  FRAMES 8
uint16_t lvl[FRAMES],           // Audio level for the prior #FRAMES frames
         avgLo  = 6,            // Audio volume lower end of range
         avgHi  = 512,          // Audio volume upper end of range
         sum    = 256 * FRAMES; // Sum of lvl[] array
uint8_t  lvlIdx = 0;            // Counter into lvl[] array
int16_t  peak   = 0;            // Falling dot shows recent max
int8_t   peakV  = 0;            // Velocity of peak dot

// COLOR TABLES for pretty_meter animation -------------------------------------
const uint8_t PROGMEM
  reds[]   = { 0x9A, 0x75, 0x00, 0x00, 0x00, 0x65, 0x84, 0x9A, 0xAD, 0xAD },
  greens[] = { 0x00, 0x00, 0x00, 0x87, 0xB1, 0x9E, 0x87, 0x66, 0x00, 0x00 },
  blues[]  = { 0x95, 0xD5, 0xFF, 0xC3, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },
  gamma8[] = { // Gamma correction improves the appearance of midrange colors
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
    0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x02, 0x03, 0x03, 0x03, 0x03,
    0x03, 0x03, 0x04, 0x04, 0x04, 0x04, 0x05, 0x05, 0x05, 0x05, 0x05, 0x06,
    0x06, 0x06, 0x06, 0x07, 0x07, 0x07, 0x08, 0x08, 0x08, 0x09, 0x09, 0x09,
    0x0A, 0x0A, 0x0A, 0x0B, 0x0B, 0x0B, 0x0C, 0x0C, 0x0D, 0x0D, 0x0D, 0x0E,
    0x0E, 0x0F, 0x0F, 0x10, 0x10, 0x11, 0x11, 0x12, 0x12, 0x13, 0x13, 0x14,
    0x14, 0x15, 0x15, 0x16, 0x16, 0x17, 0x18, 0x18, 0x19, 0x19, 0x1A, 0x1B,
    0x1B, 0x1C, 0x1D, 0x1D, 0x1E, 0x1F, 0x1F, 0x20, 0x21, 0x22, 0x22, 0x23,
```

```
    0x24, 0x25, 0x26, 0x26, 0x27, 0x28, 0x29, 0x2A, 0x2A, 0x2B, 0x2C, 0x2D,
    0x2E, 0x2F, 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39,
    0x3A, 0x3B, 0x3C, 0x3D, 0x3E, 0x3F, 0x40, 0x41, 0x42, 0x44, 0x45, 0x46,
    0x47, 0x48, 0x49, 0x4B, 0x4C, 0x4D, 0x4E, 0x50, 0x51, 0x52, 0x54, 0x55,
    0x56, 0x58, 0x59, 0x5A, 0x5C, 0x5D, 0x5E, 0x60, 0x61, 0x63, 0x64, 0x66,
    0x67, 0x69, 0x6A, 0x6C, 0x6D, 0x6F, 0x70, 0x72, 0x73, 0x75, 0x77, 0x78,
    0x7A, 0x7C, 0x7D, 0x7F, 0x81, 0x82, 0x84, 0x86, 0x88, 0x89, 0x8B, 0x8D,
    0x8F, 0x91, 0x92, 0x94, 0x96, 0x98, 0x9A, 0x9C, 0x9E, 0xA0, 0xA2, 0xA4,
    0xA6, 0xA8, 0xAA, 0xAC, 0xAE, 0xB0, 0xB2, 0xB4, 0xB6, 0xB8, 0xBA, 0xBC,
    0xBF, 0xC1, 0xC3, 0xC5, 0xC7, 0xCA, 0xCC, 0xCE, 0xD1, 0xD3, 0xD5, 0xD7,
    0xDA, 0xDC, 0xDF, 0xE1, 0xE3, 0xE6, 0xE8, 0xEB, 0xED, 0xF0, 0xF2, 0xF5,
    0xF7, 0xFA, 0xFC, 0xFF };

//Setup----------------------------------------------------------------------

void setup() {
  Serial.begin(57600);
  delay(3000); // 3 second delay for recovery

  //Add the onboard CP LEDs & the string of LEDs to create a single array
  FastLED.addLeds<LED_TYPE, CP_PIN,COLOR_ORDER>(leds, 0,
NUM_CP).setCorrection(TypicalLEDStrip);
  FastLED.addLeds<LED_TYPE, LED_PIN,COLOR_ORDER>(leds, NUM_CP,
NUM_STRIP).setCorrection(TypicalLEDStrip);

  FastLED.setBrightness(BRIGHTNESS);
  pinMode(switchPin, INPUT_PULLUP); //set up the CP switch
  pinMode(leftButtonPin, INPUT_PULLDOWN);  //set up the CP left button
  pinMode(rightButtonPin, INPUT_PULLDOWN);  //set up the CP right button
  currentBlending = LINEARBLEND; //FastLED blending
  for(uint8_t i=0; i<FRAMES; i++) lvl[i] = 256; //setup from sound reactive
}

void loop() {

  PrevleftButton = leftButton;
  leftButton = digitalRead(leftButtonPin);

  PrevrightButton = rightButton;
  rightButton = digitalRead(rightButtonPin);

  switchState = digitalRead(switchPin);

  if (!switchState) {
    turnedOff();
  } else {
    turnedOn = true;

    if (!PrevleftButton && leftButton) {
      leftButtonPress();
    }
    if (!PrevrightButton && rightButton) {
      rightButtonPress();
    }

      static uint8_t startIndex = 0;
      startIndex = startIndex + 20; /* motion speed */

      switch (ledMode) {
        case 0:   rainbow();  break;
        case 1:   confetti(); break;
        case 2:   RainbowBPM(); break;
        case 3:   RainbowTwinkles(); break;
        case 4:   sinelon(); break;
        case 5:   juggle(); break;
        case 6:   PartyBPM(); break;
        case 7:   PartyTwinkles(); break;
        case 8:   LavaBPM(); break;
        case 9:   LavaTwinkles(); break;
```

```
        case 99:  soundreactive(); break;
      }
      FastLED.show();
      FastLED.delay(1000/SPEED);
      EVERY_N_MILLISECONDS(20) { gHue++; } // slowly cycle the "base color" through
the rainbow
  }
}

void clearpixels() //clears all pixels, but does not put the board to sleep
{
    for (int i = 0; i < NUM_LEDS; i++) leds[i] = CRGB::Black;
    FastLED.show();
}

void turnedOff()
{
   if (turnedOn){
   clearpixels();
    turnedOn = false;
  }
}

void leftButtonPress()
{
  Serial.println("Left Button Pressed");
  clearpixels();
  ledMode++;
    if (ledMode > NUM_MODES){
    ledMode=0;
  }
}

void rightButtonPress()
{
  Serial.println("Right Button Pressed");
  clearpixels();
  ledMode=99;
}

void FillLEDsFromPaletteColors( uint8_t colorIndex)
{
  for (int i = 0; i < NUM_LEDS; i++) leds[i] = ColorFromPalette( currentPalette,
colorIndex, BRIGHTNESS, currentBlending);
    colorIndex += STEPS;
}

// Twinkles!! ----------------------------------------------------------------
  // I've included twinkles for all of FastLEDs built-in palettes
  // Make your own palette (see ColorPalette example) to personalize twinkles

enum { GETTING_DARKER = 0, GETTING_BRIGHTER = 1 };

void OceanTwinkles()
{
  // Make each pixel brighter or darker, depending on
  // its 'direction' flag.
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);

  // Now consider adding a new random twinkle
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( OceanColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
```

```
}

void ForestTwinkles()
{
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( ForestColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void PartyTwinkles()
{
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( PartyColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void RainbowTwinkles()
{
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( RainbowColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void HeatTwinkles()
{
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( HeatColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void LavaTwinkles()
{
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( LavaColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void CloudTwinkles()
{
```

```
  brightenOrDarkenEachPixel( FADE_IN_SPEED, FADE_OUT_SPEED);
  if( random8() < DENSITY ) {
    int pos = random16(NUM_LEDS);
    if( !leds[pos]) {
      leds[pos] = ColorFromPalette( CloudColors_p, random8(), STARTING_BRIGHTNESS,
NOBLEND);
      setPixelDirection(pos, GETTING_BRIGHTER);
    }
  }
}

void brightenOrDarkenEachPixel( fract8 fadeUpAmount, fract8 fadeDownAmount)
{
 for( uint16_t i = 0; i < NUM_LEDS; i++) {
    if( getPixelDirection(i) == GETTING_DARKER) {
      // This pixel is getting darker
      leds[i] = makeDarker( leds[i], fadeDownAmount);
    } else {
      // This pixel is getting brighter
      leds[i] = makeBrighter( leds[i], fadeUpAmount);
      // now check to see if we've maxxed out the brightness
      if( leds[i].r == 255 || leds[i].g == 255 || leds[i].b == 255) {
        // if so, turn around and start getting darker
        setPixelDirection(i, GETTING_DARKER);
      }
    }
  }
}

CRGB makeBrighter( const CRGB& color, fract8 howMuchBrighter)
{
  CRGB incrementalColor = color;
  incrementalColor.nscale8( howMuchBrighter);
  return color + incrementalColor;
}

CRGB makeDarker( const CRGB& color, fract8 howMuchDarker)
{
  CRGB newcolor = color;
  newcolor.nscale8( 255 - howMuchDarker);
  return newcolor;
}

// For illustration purposes, there are two separate implementations
// provided here for the array of 'directionFlags':
// - a simple one, which uses one byte (8 bits) of RAM for each pixel, and
// - a compact one, which uses just one BIT of RAM for each pixel.

// Set this to 1 or 8 to select which implementation
// of directionFlags is used.  1=more compact, 8=simpler.
#define BITS_PER_DIRECTION_FLAG 1

#if BITS_PER_DIRECTION_FLAG == 8
// Simple implementation of the directionFlags array,
// which takes up one byte (eight bits) per pixel.
uint8_t directionFlags[NUM_LEDS];

bool getPixelDirection( uint16_t i) {
  return directionFlags[i];
}

void setPixelDirection( uint16_t i, bool dir) {
  directionFlags[i] = dir;
}
#endif

#if BITS_PER_DIRECTION_FLAG == 1
// Compact (but more complicated) implementation of
// the directionFlags array, using just one BIT of RAM
```

```
// per pixel.  This requires a bunch of bit wrangling,
// but conserves precious RAM.  The cost is a few
// cycles and about 100 bytes of flash program memory.
uint8_t  directionFlags[ (NUM_LEDS+7) / 8];

bool getPixelDirection( uint16_t i) {
  uint16_t index = i / 8;
  uint8_t  bitNum = i & 0x07;
  // using Arduino 'bitRead' function; expanded code below
  return bitRead( directionFlags[index], bitNum);
  // uint8_t  andMask = 1 << bitNum;
  // return (directionFlags[index] & andMask) != 0;
}

void setPixelDirection( uint16_t i, bool dir) {
  uint16_t index = i / 8;
  uint8_t  bitNum = i & 0x07;
  // using Arduino 'bitWrite' function; expanded code below
  bitWrite( directionFlags[index], bitNum, dir);
  //  uint8_t  orMask = 1 << bitNum;
  //  uint8_t andMask = 255 - orMask;
  //  uint8_t value = directionFlags[index] & andMask;
  //  if( dir ) {
  //    value += orMask;
  //  }
  //  directionFlags[index] = value;
}
#endif

// GLITTER --------------------------------------------------------------------

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

void rainbowWithGlitter()
{
  // built-in FastLED rainbow, plus some random sparkly glitter
  rainbow();
  addGlitter(80);
}

void addGlitter( fract8 chanceOfGlitter)
{
  if( random8() < chanceOfGlitter) {
    leds[ random16(NUM_LEDS) ] += CRGB::White;
  }
}

// FastLED Demo Reel ----------------------------------------------------------

void confetti()
{
  // random colored speckles that blink in and fade smoothly
  fadeToBlackBy( leds, NUM_LEDS, 10);
  int pos = random16(NUM_LEDS);
  leds[pos] += CHSV( gHue + random8(64), 200, 255);
}

void juggle() {
  // eight colored dots, weaving in and out of sync with each other
  fadeToBlackBy( leds, NUM_LEDS, 20);
  byte dothue = 0;
  for( int i = 0; i < 8; i++) {
    leds[beatsin16( i+7, 0, NUM_LEDS )] |= CHSV(dothue, 200, 255);
    dothue += 32;
  }
```

```
}

void sinelon()
{
  // A colored dot sweeping back and forth, with fading trails
  fadeToBlackBy( leds, NUM_LEDS, 20);
  int pos = beatsin16( 13, 0, NUM_LEDS-1 );
  leds[pos] += CHSV( gHue, 255, 192);
}

// Various BPM examples -----------------------------------------------------
  // I've included BPM examples for all of FastLEDs built-in palettes
  // Make your own palette (see ColorPalette example) to personalize the BPM example

void PartyBPM()
{
  // colored stripes pulsing at a defined Beats-Per-Minute (BPM)
  CRGBPalette16 palette = PartyColors_p;//can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}

void RainbowBPM()
{
  CRGBPalette16 palette = RainbowColors_p; //can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}

void CloudBPM()
{
  CRGBPalette16 palette = CloudColors_p; //can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}

void ForestBPM()
{
  CRGBPalette16 palette = ForestColors_p;//can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}

void HeatBPM()
{
  CRGBPalette16 palette = HeatColors_p; //can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}

void LavaBPM()
{
  CRGBPalette16 palette = LavaColors_p; //can adjust the palette here
  uint8_t beat = beatsin8( BeatsPerMinute, 64, 255);
  for( int i = 0; i < NUM_LEDS; i++) { //9948
    leds[i] = ColorFromPalette(palette, gHue+(i*2), beat-gHue+(i*10));
  }
}
```

```
void soundreactive() {
  uint8_t  i, r, g, b;
  uint16_t minLvl, maxLvl, a, scaled;
  int16_t  p;

  p           = CircuitPlayground.mic.soundPressureLevel(10); // 10 ms
  p           = map(p, 56, 140, 0, 350);    // Scale to 0-350 (may overflow)
  a           = constrain(p, 0, 350);       // Clip to 0-350 range
  sum        -= lvl[lvlIdx];
  lvl[lvlIdx] = a;
  sum        += a;                              // Sum of lvl[] array
  minLvl = maxLvl = lvl[0];                     // Calc min, max of lvl[]...
  for(i=1; i<FRAMES; i++) {
    if(lvl[i] < minLvl)      minLvl = lvl[i];
    else if(lvl[i] > maxLvl) maxLvl = lvl[i];
  }

  // Keep some minimum distance between min & max levels,
  // else the display gets "jumpy."
  if((maxLvl - minLvl) < 40) {
    maxLvl = (minLvl < (512-40)) ? minLvl + 40 : 512;
  }
  avgLo = (avgLo * 7 + minLvl + 2) / 8; // Dampen min/max levels
  avgHi = (maxLvl >= avgHi) ?           // (fake rolling averages)
    (avgHi *  3 + maxLvl +1) /  4 :     // Fast rise
    (avgHi * 31 + maxLvl + 8) / 32;     // Slow decay

  a = sum / FRAMES; // Average of lvl[] array
  if(a <= avgLo) {  // Below min?
    scaled = 0;     // Bargraph = zero
  } else {          // Else scale to fixed-point coordspace 0-256*NUM_LEDS
      scaled = (256L * (NUM_LEDS)) * (a - avgLo) / (avgHi - avgLo);
      if(scaled > 256 * (NUM_LEDS)) scaled = (256 * (NUM_LEDS));
    }
  if(scaled >= peak) {            // New peak
    peakV = (scaled - peak) / 4;  // Give it an upward nudge
    peak  = scaled;
  }

  uint8_t  whole  = scaled / 256,    // Full-brightness pixels (0-10)
           frac   = scaled & 255;    // Brightness of fractional pixel
  int      whole2 = peak / 256,      // Index of peak pixel
           frac2  = peak & 255;      // Between-pixels position of peak
  uint16_t a1, a2;                   // Scaling factors for color blending

  for(i=0; i<NUM_LEDS; i++) {              // For each NeoPixel...
    if(i <= whole) {                 // In currently-lit area?
      r = pgm_read_byte(&reds[i]),   // Look up pixel color
      g = pgm_read_byte(&greens[i]),
      b = pgm_read_byte(&blues[i]);
      if(i == whole) {               // Fraction pixel at top of range?
        a1 = (uint16_t)frac + 1;     // Fade toward black
        r  = (r * a1) >> 8;
        g  = (g * a1) >> 8;
        b  = (b * a1) >> 8;
      }
    } else {
      r = g = b = 0;                 // In unlit area
    }
    // Composite the peak pixel atop whatever else is happening...
    if(i == whole2) {                // Peak pixel?
      a1 = 256 - frac2;              // Existing pixel blend factor 1-256
      a2 = frac2 + 1;                // Peak pixel blend factor 1-256
      r  = ((r * a1) + (0x84 * a2)) >> 8; // Will
      g  = ((g * a1) + (0x87 * a2)) >> 8; // it
      b  = ((b * a1) + (0xC3 * a2)) >> 8; // blend?
    } else if(i == (whole2-1)) {     // Just below peak pixel
      a1 = frac2 + 1;                // Opposite blend ratios to above,
      a2 = 256 - frac2;              // but same idea
```

```
      r  = ((r * a1) + (0x84 * a2)) >> 8;
      g  = ((g * a1) + (0x87 * a2)) >> 8;
      b  = ((b * a1) + (0xC3 * a2)) >> 8;
    }
       leds[i].r = pgm_read_byte(&gamma8[r]);
       leds[i].g = pgm_read_byte(&gamma8[g]);
       leds[i].b = pgm_read_byte(&gamma8[b]);
   }
   FastLED.show();

  peak += peakV;
  if(peak <= 0) {
    peak  = 0;
    peakV = 0;
  } else if(peakV >= -128) {
    peakV -= 2;
  }

  if(++lvlIdx >= FRAMES) lvlIdx = 0;

}
```

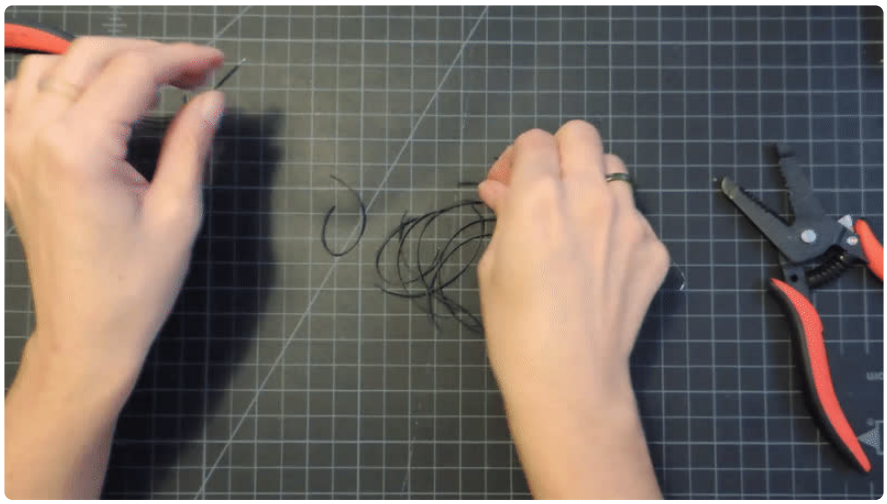
LEDs

There are a few ways to approach stringing LEDs. This part of the guide will go over my method of braiding the three wires between each NeoPixel. This page provides an overview of the method covered in greater detail over the next few pages.

If you aren't interested in braiding, Erin (https://adafru.it/CfR) has a great method for quickly stringing NeoPixels together, (https://adafru.it/D4Z) shown in her twinkling LED parasol.

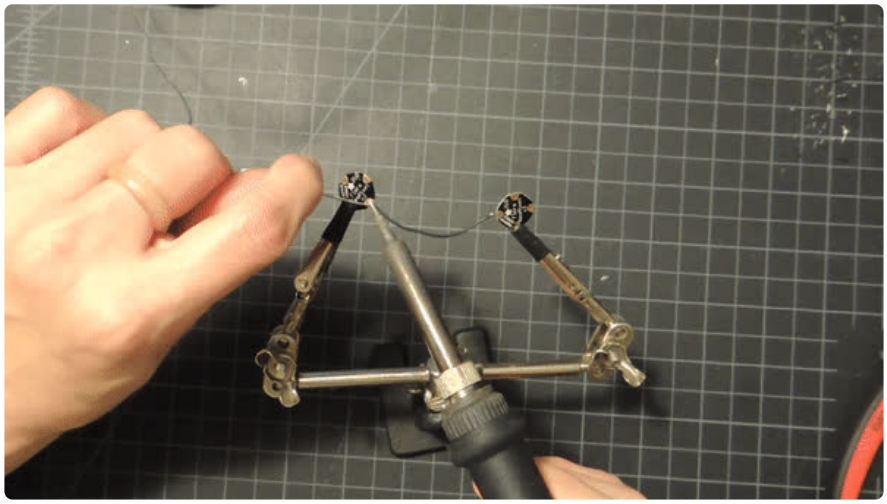
Cut & Strip Wires

When cutting wires, determine the approximate distance between the LEDs & add at least 1". Use a wire stripper tool or your fingernails to strip approximately ¼" off the ends.



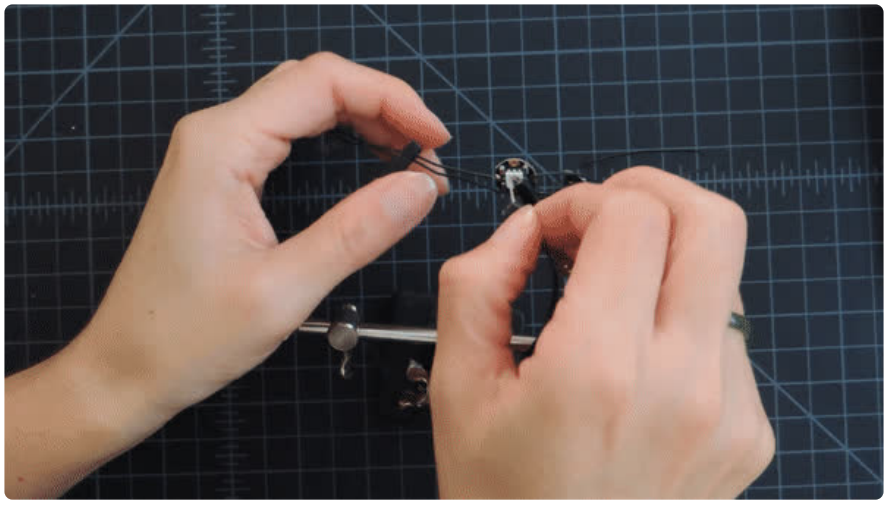
Make the Data Line

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



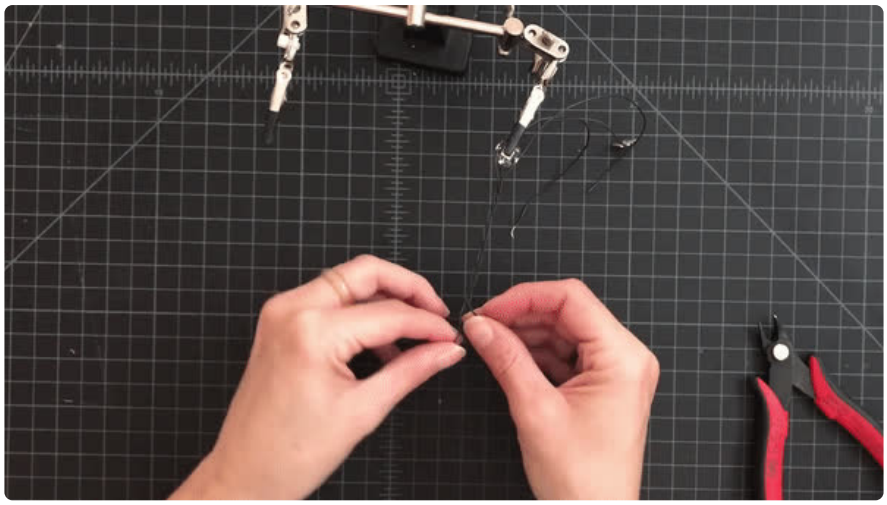
Power & Ground

Add power and ground wires. Use different color wires or mark wires to ensure that the lines don't cross. I attach power before ground wires and strip the ends of my power wires.



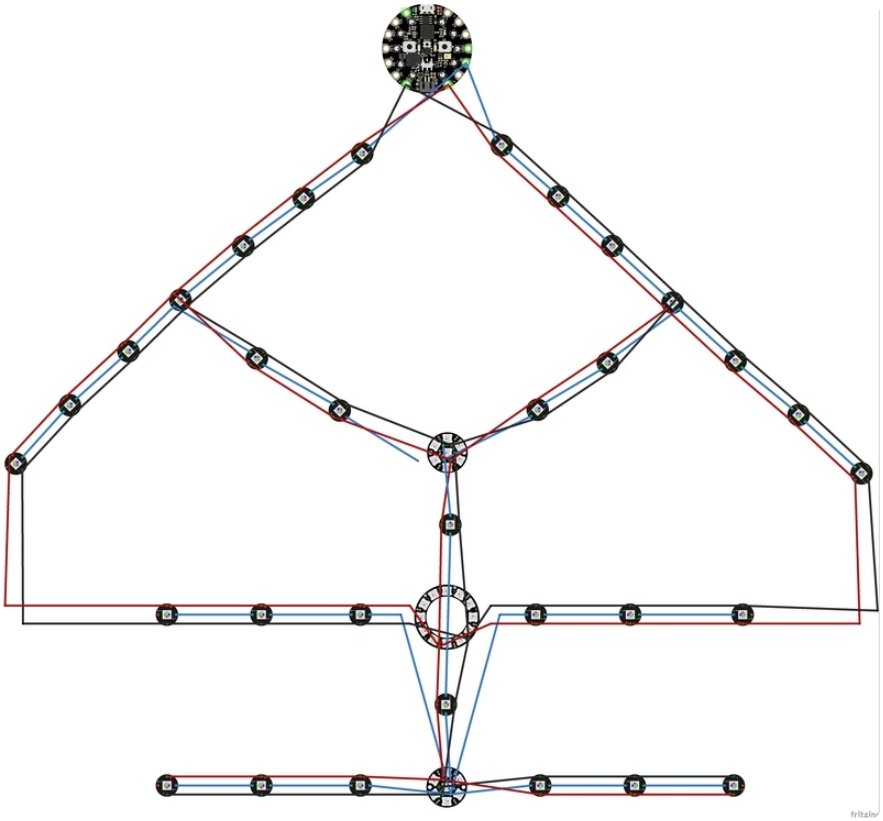
Braid

Braid and solder wires to NeoPixels, starting with the power wires, before moving onto the ground wires.



Circuit Diagram

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



Splitting the data line into multiple directions inhibits the ability to address each pixel individually.

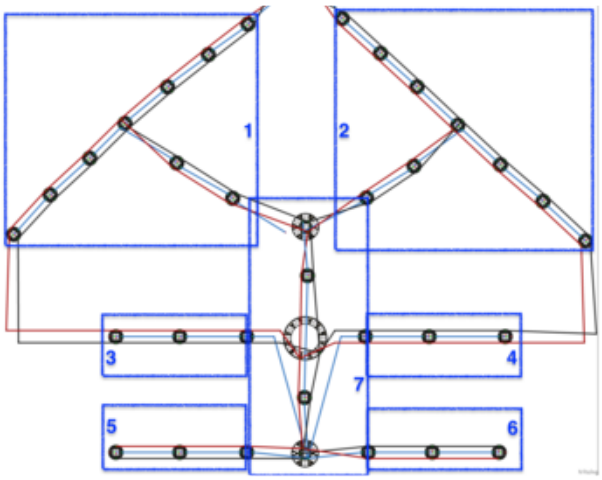
I've had success with this method with a small number of LEDs. For a larger project, you should review the voltage drop on the signals coming in to a data in. If the wires

are long or go in too many directions, the signal may not be strong enough to drive the data along the strand.

This approach allows us to create a dimensional array of LEDs without doing extensive mapping: the LEDs on the right and left as well as the pieces along the waist and under the bust will show mirrored animations.

The sections of pixels will be reviewed in detail on the next page.

Data Line



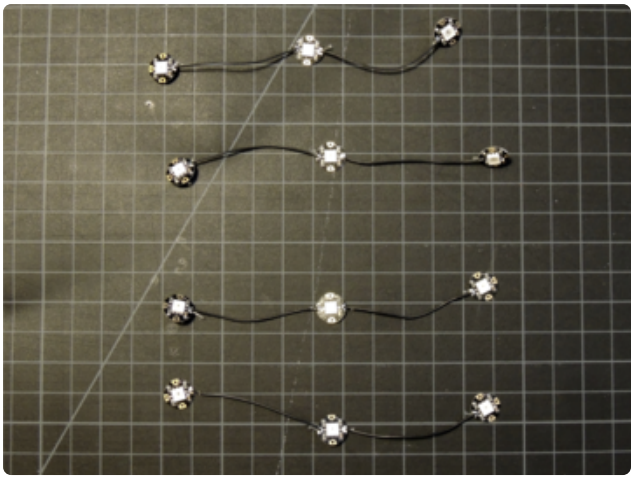
Start connecting the data line using the pieces of wire you've cut and stripped.

Work in the sections recommended to ensure that the data flows all in one direction.

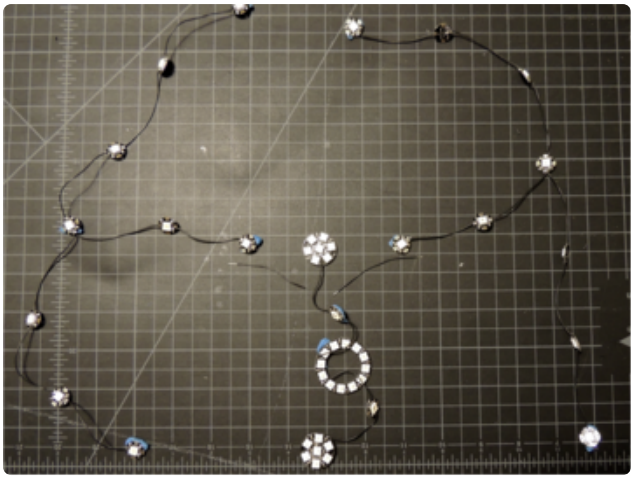
Sections

Sections 1 & 2 are identical and each contain 9 individual LEDs.

There will be two data lines out of the 4th LED on each side.

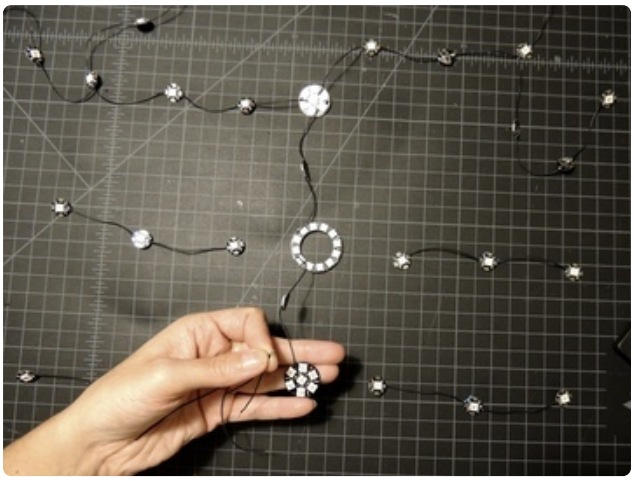


Sections 3-6 are identical and each contain 3 LEDs.



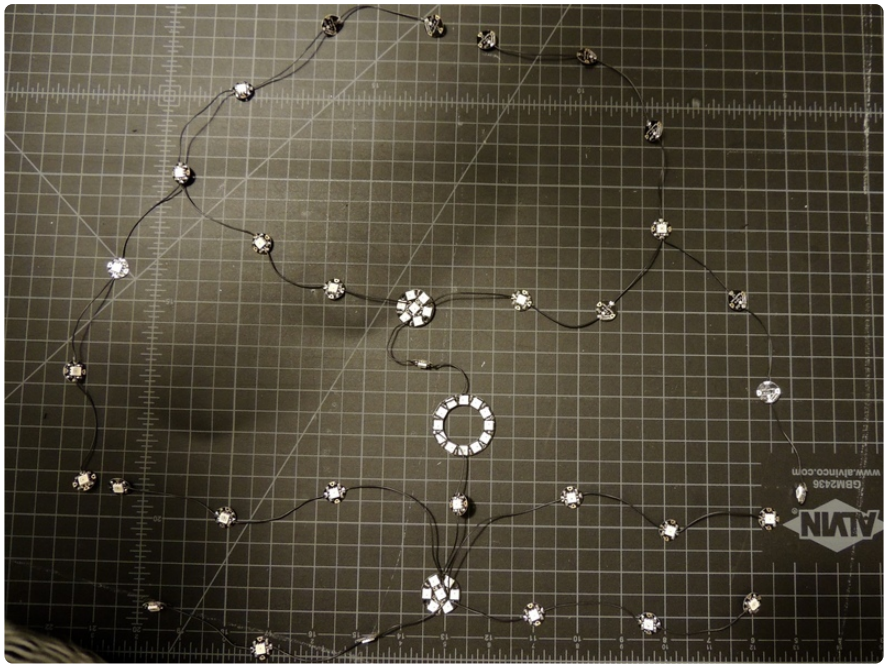
Section 7 brings together the NeoPixel ring, two NeoPixel Jewels & two individual NeoPixels.

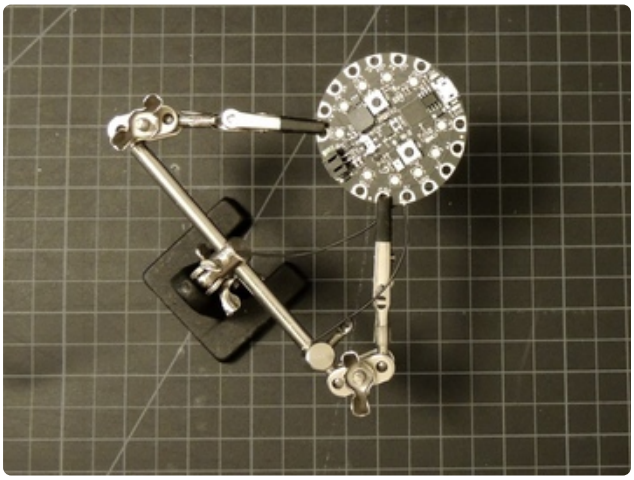
The data in on the top jewel is fed from one LED.



There are 4 data out wires on the bottom jewel, one pointing towards each of the "sections" 3-6.

Bring it all together

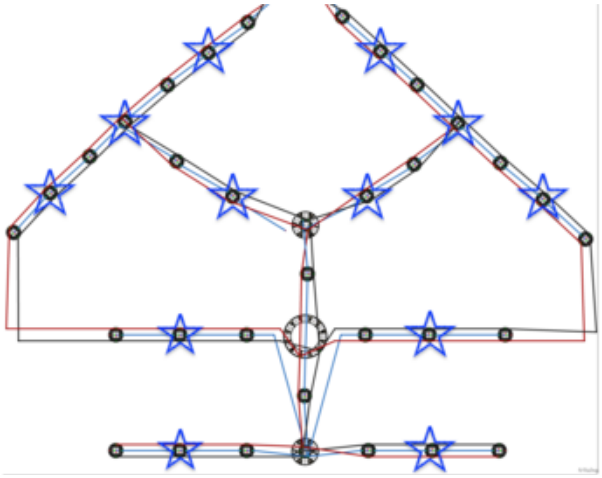




Attach two wires to the **A1** pin on the Circuit Playground Express (CPX) and connect it to the two NeoPixels at the top.

One of the wires will need to be long enough to travel under the CPX and to the first pixel on that side.

Power & Ground



Start with one of the LEDs identified by a star in the image.

Determine the approximate distance between the ground and power holes on this and the adjacent LEDs.

Wires

Cut wires at least 1” longer than the longest distance needed. You will need 4 wires for most of the LEDs, but 6 for LEDs where the data line is split.

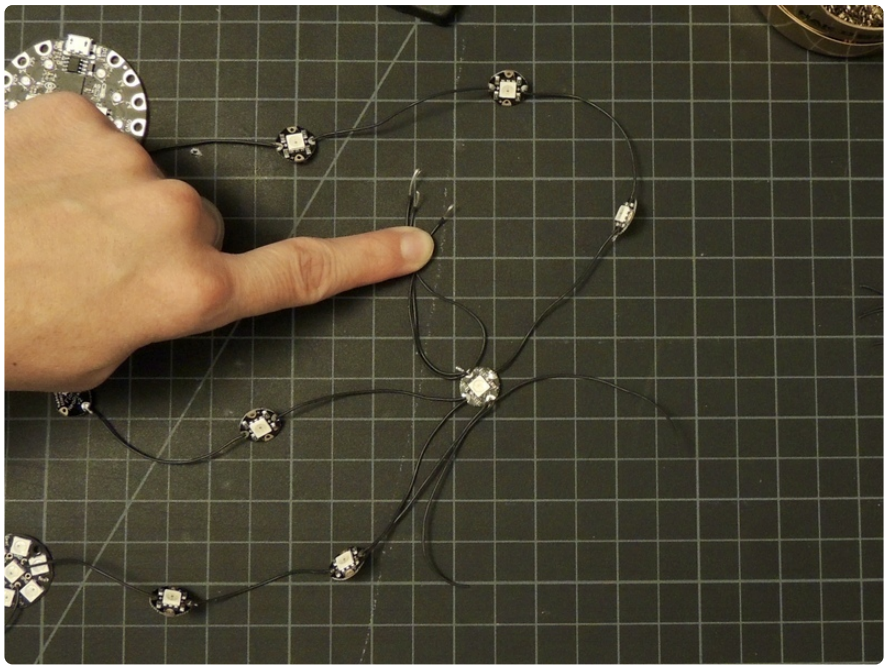
Strip approximately ¼” off one side of each wire. If you are using different color wires, strip all sides at once.

Twist the stripped ends of wire into a pair or triplet so that it can fit into a hole on the neopixels

Solder, Part 1

Attach one pair (or triplet) of wires into the power and the other pair into the ground holes of the LED.

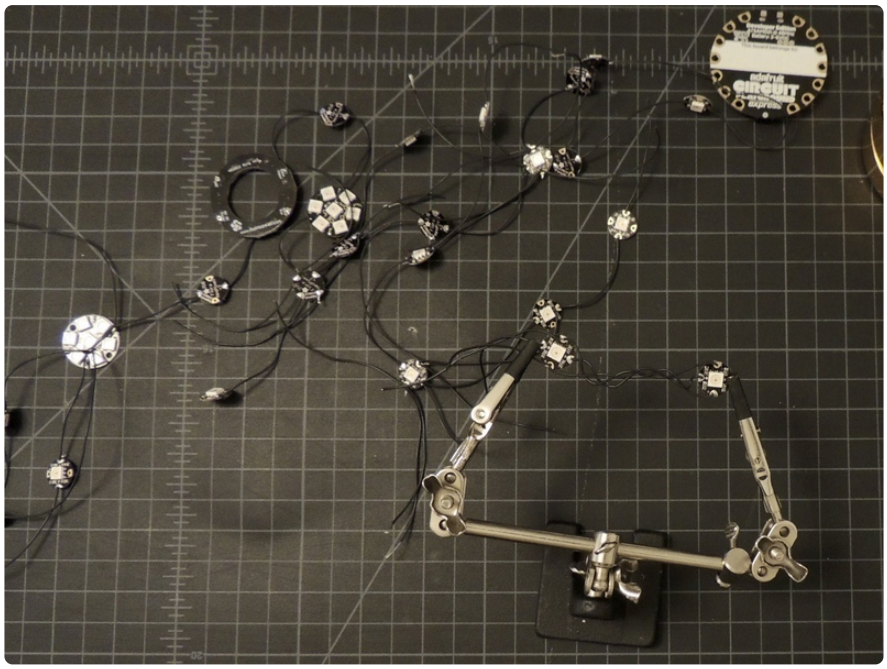
Strip about ¼” off the ends of the wires in the power hole of the LED. If you are using different color wires, strip all sides at once.



Braid

Braid wires from two LEDs towards an LED without any wires.

The 30awg wire will fit into the NeoPixel holes without being stripped. I found this was a great way to keep the braid on one side while braiding another set of wires or waited until another section was complete.

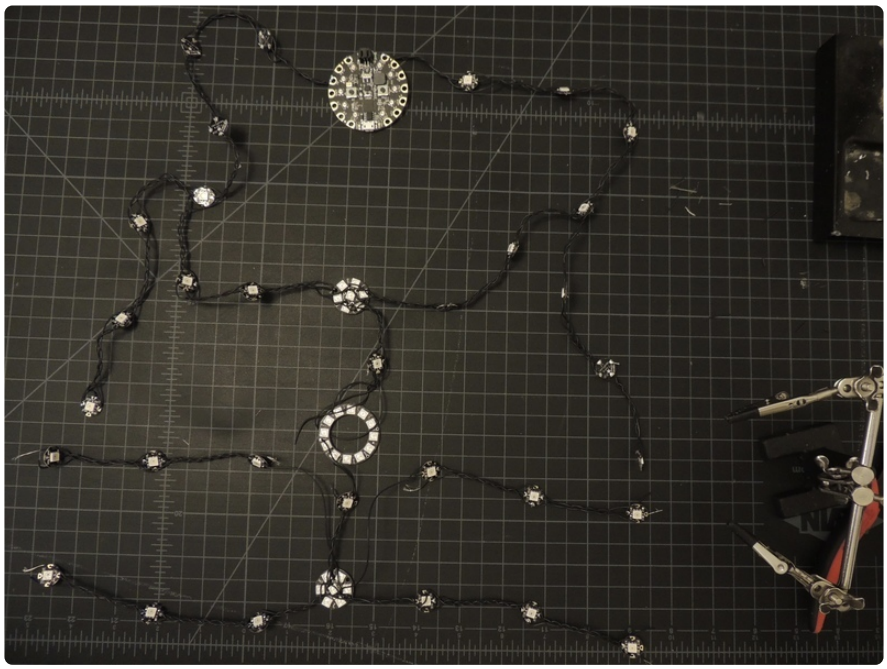


Solder, Part 2

After braiding, check to ensure that wiring is correct from one LED to the next.

Strip any additional sheathing from the power wires, twist together, feed through the power hole, and solder.

Once the power wires are attached, strip ground wires, twist together, feed through the ground hole, and solder.

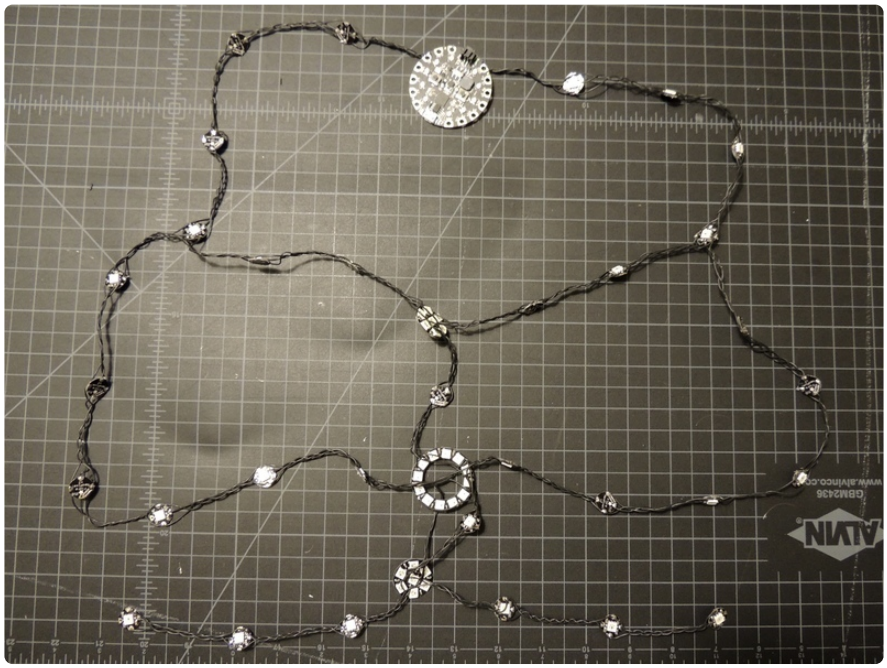


Bring it all Together

The power and ground lines of the NeoPixels in Sections 3 & 4 connect directly to the power line on the NeoPixel Ring in the center. These NeoPixels also connect up to the top along the outside.

Follow the diagram to complete the rest of the power and ground wiring.

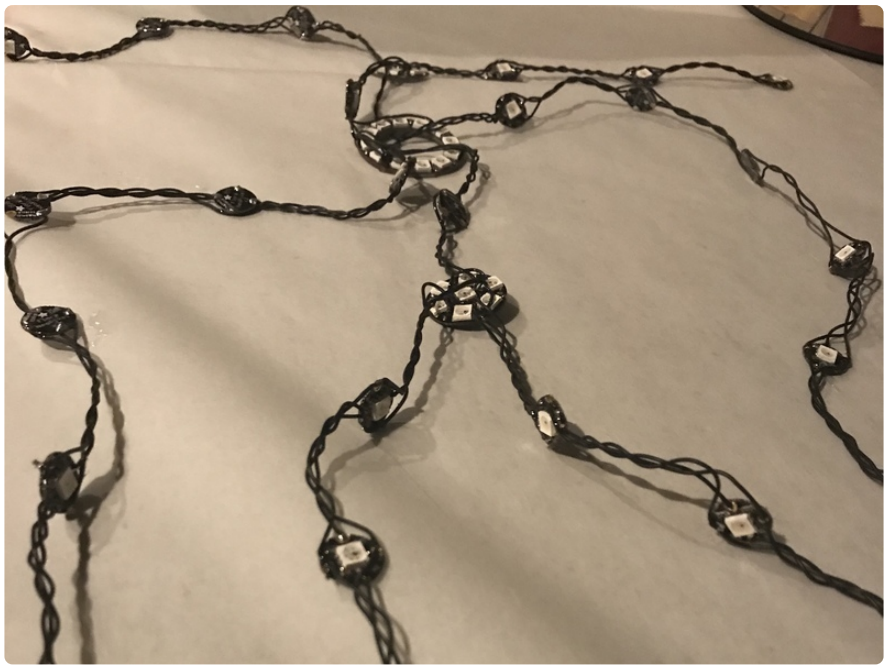
It may feel like a mess at some points. Remember to take breaks, double check your work, and say nice things to yourself!



Check that all of your wires are running to and from the correct locations, plug in your battery and test!

Final Assembly and Wear

In a well ventilated space, dab solder joints with e6000 or other adhesive of your choice.



Attach to Harness

Position the strand on the pleather harness. Dab e6000 adhesive onto the larger NeoPixel pieces and the LEDs that will sit on ends and corners and attach to the harness and clip into place. Use e6000 and thread to tack down the rest of the pixels.

Battery Pocket

Print a 3D Printed Battery Pocket (https://adafru.it/kSa)for your Lithium Ion Polymer battery or sew a battery pocket out of fabric and attach to the side of the harness.

Wear It!

Plug the battery into the extension cable and the CPX and turn it on.

Use the slide-switch to turn on and off the LEDs. The switch itself doesn't turn the board off, but allows you to go a little incognito with the lights.

Use the left button on the CPX to cycle through some FastLED examples. Customize your sketch by creating new modes and palettes!

Use the right button on the CPX to activate the pretty VU meter example sketch.

leds_wearables_xmastree.gif