FLORAbrella

Created by Leslie Birch

https://learn.adafruit.com/florabrella

Last updated on 2024-03-08 01:55:32 PM EST

Table of Contents

Overview 3

• New for 2019

Tools and Supplies 4

Make with FLORA 5

Wiring Diagram 6

Solder the FLORA circuit 7

Code 10

Test the NeoPixel Strip 14

• Configure the FLORA
• Test the NeoPixel Strip

Make with Circuit Playground 16

Wiring Diagram 17

Code with MakeCode 18

• Variables
• Inputs
• Rainbow Mode
• Raindrop Mode
• Color Sensing Mode

Prepare NeoPixel Strip 30

• Prepare the NeoPixel Strips
• Prepare the Wires
• Solder the Wires
•
•
• Now that you have things soldered, it would be good to try another test with the strips. Remember that the USB is not powerful enough to light up the strips, so once you load the program, plug the battery in and then turn it on. You should get a vibrant display. If not, use your multimeter to troubleshoot the circuit.

Insert the Circuit 40

• Prepare the Umbrella
• Insert the Circuit
• Secure the NeoPixel Strips

Finishing Touches 45

Use it! 48

Overview

You'll be a rainbow in any storm with the FLORAbrella. With its NeoPixel LED strips and color sensor, you'll be able to match your clothing, or display rainbow and rain patterns. Get ready to have an entourage at the next parade!

For this project, you'll need to create the circuit, mount it in the dome of the umbrella, and then finish it off by creating clear vinyl hammocks for the battery and electronics.

New for 2019

This guide shows you how to make the project in two versions, original and 'easier'

The original uses the FLORA controller and color sensor, programmed in Arduino which takes some time to assemble and code.

The recommended, newer, easier version uses the Circuit Playground Express (http://adafru.it/3333). It already has an onboard color sensor and embedded buttons, so you can get the same effects with a bit less work! Check the pages under Circuit Playground for instructions on using the Circuit Playground Express. Coding is easier also as you use MakeCode so you drag and drop the code to load, no software on your computer is required.

This is an intermediate-level project that requires some precise soldering on the LED strips, so just stay away from caffeine and keep your cat away. Also, make sure you read and practice the following guides:

- Getting Started with FLORA (https://adafru.it/aSZ) OR Getting Started with Circuit Playground Express (https://adafru.it/adafruit-cpx)
- Adafruit Color Sensors (https://adafru.it/cb8)
- Adafruit NeoPixel Überguide (https://adafru.it/cEz)

Okay, let's get the supplies!

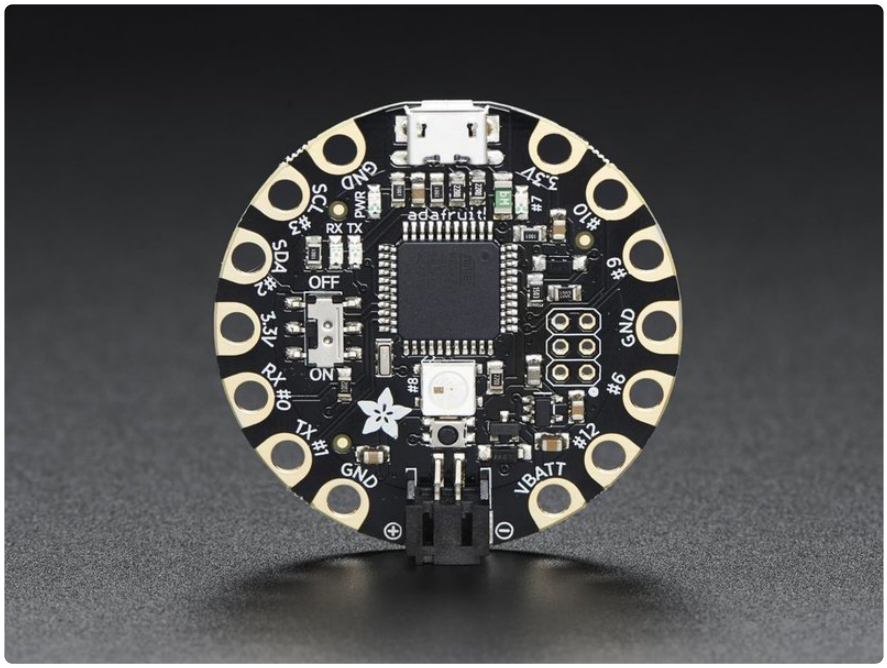
Tools and Supplies

For this project you will need:

- FLORA wearable microcontroller (http://adafru.it/659) OR Circuit Playground Express (http://adafru.it/3333)
- USB cable - A/MicroB - 3ft (http://adafru.it/592)
- 5 meters of NeoPixel digital RGB LED strip 30 LED white (http://adafru.it/1376)
- FLORA color sensor (http://adafru.it/1356) (not needed if you're using the Circuit Playground)
- LiPoly rechargeable battery (3.75V 2500mAh) (http://adafru.it/328)
- micro Lipo battery charger (http://adafru.it/1304)
- tactile switch (http://adafru.it/1119)(not needed if you're using the Circuit Playground)
- stranded wire in three colors
- sharp wire strippers (http://adafru.it/527)
- sharp flush snips (http://adafru.it/1368)
- needle nose pliers (http://adafru.it/146)
- soldering iron and solder (https://adafru.it/c6b)
- multimeter (for troubleshooting) (https://adafru.it/aOy)
- third hand tool (http://adafru.it/291)
- alligator clips (http://adafru.it/1008) (for prototyping)
- small scissors (manicure style great)
- clear dome umbrella (we used one from Totes)
- 1/4 yd. clear vinyl

- adhesive velcro
- one pack of 4 in. clear cable ties (smallest size)
- Permatex 66B silicone adhesive

Make with FLORA



The following pages describe making the project with the original FLORA microcontroller board and a separate color sensor.

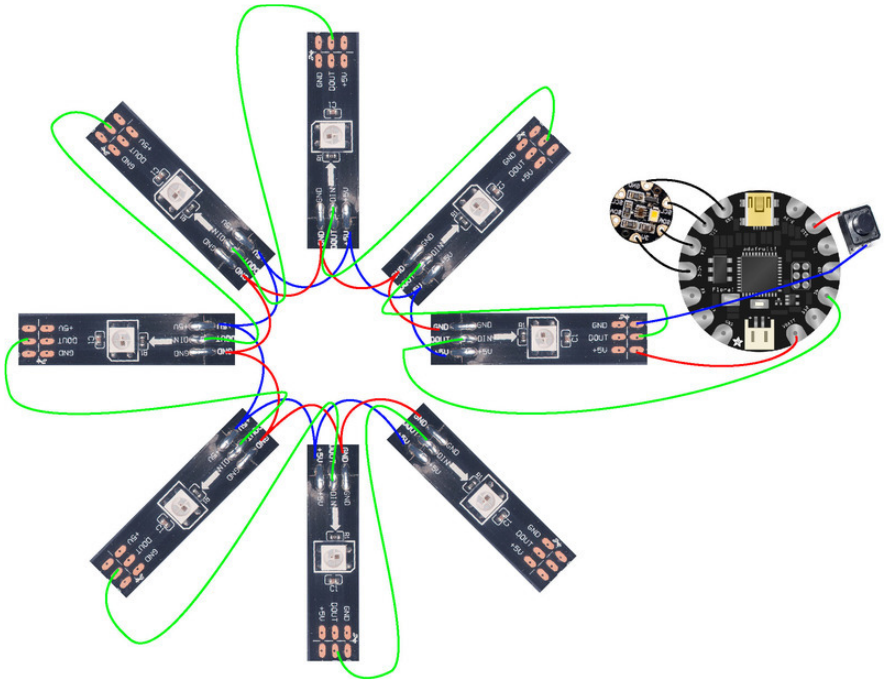
Many FLORAbrellas are in the wild in the years this tutorial first went live.

Starting in 2019, Adafruit recommends using the newer, more user friendly Circuit Playground Express controller which already includes a color sensor and buttons. The Circuit Playground Express requires less wiring, is easy to code, and is more prolific. It is likely the FLORA will reach end of life much sooner than the Circuit Playground Express.

The FLORA is a great board and it ushered in a new era in wearable electronics. Adafruit has incorporated lessons learned in using FLORA in creating new boards like Circuit Playground Express.

You are free to continue on the following pages in using FLORA.

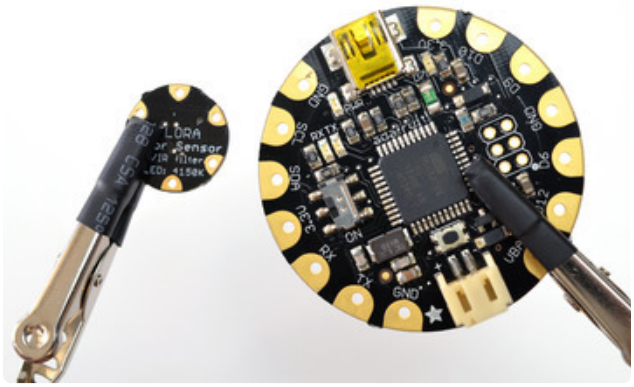
Wiring Diagram

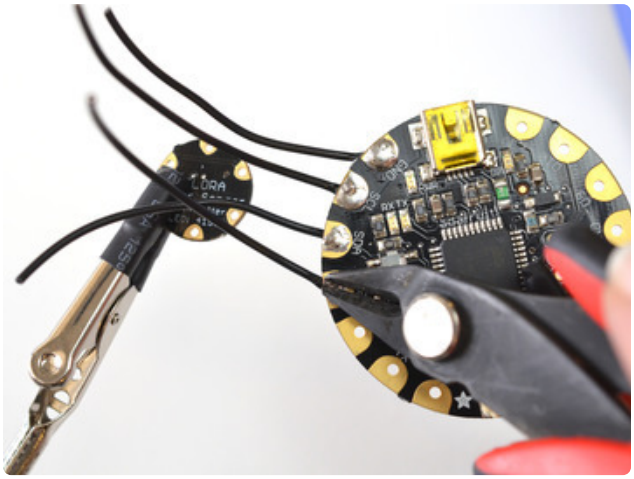


NeoPixels and color sensor face outside of umbrella, FLORA main board faces interior of umbrella. Other connections as follows:

FLORA **GND**-> NeoPixels **GND** (all)
FLORA **VBATT**-> NeoPixels **5V** (all)
FLORA **D6**-> NeoPixels **DIN** (data in)
FLORA **GND** -> color sensor **GND**
FLORA **SCL** -> color sensor **SCL**
FLORA **SDA** -> color sensor **SDA**
FLORA **3V** -> color sensor **3V**
FLORA **D10** -> pushbutton to **GND**

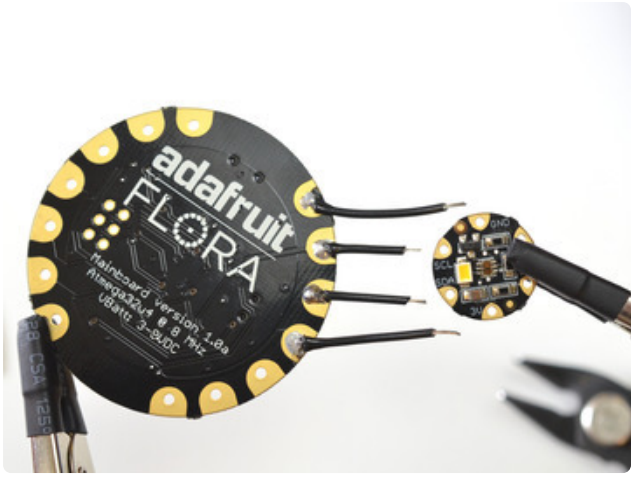
Solder the FLORA circuit

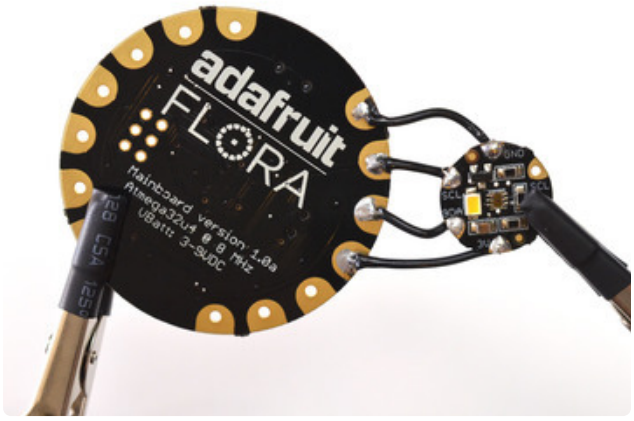




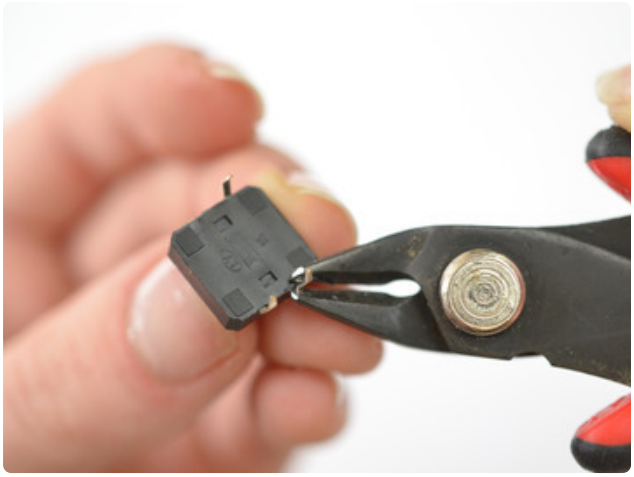
Arrange FLORA and the color sensor in your third hand tool as shown, facing opposite each other but still with **3v**, **SCL**, **SDA**, and **GND** pads aligned.

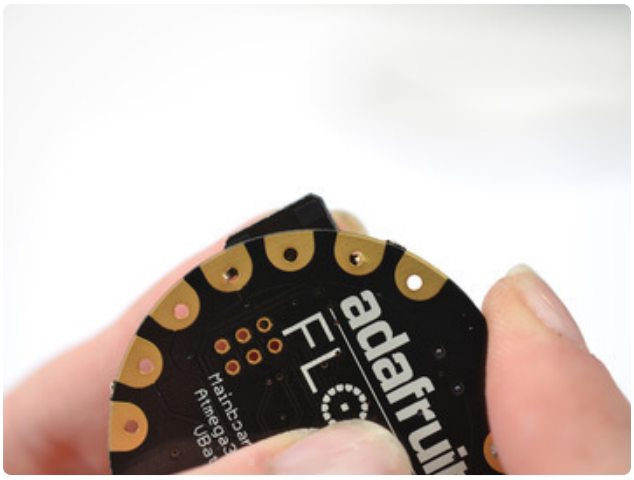
Solder four short wires to FLORA and clip off any excess.

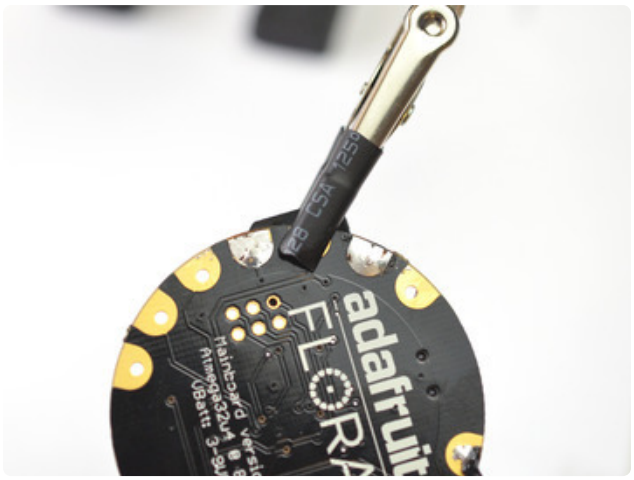




Trim and strip these wires to line up to the color sensor, then solder all four and clip off any excess wire.





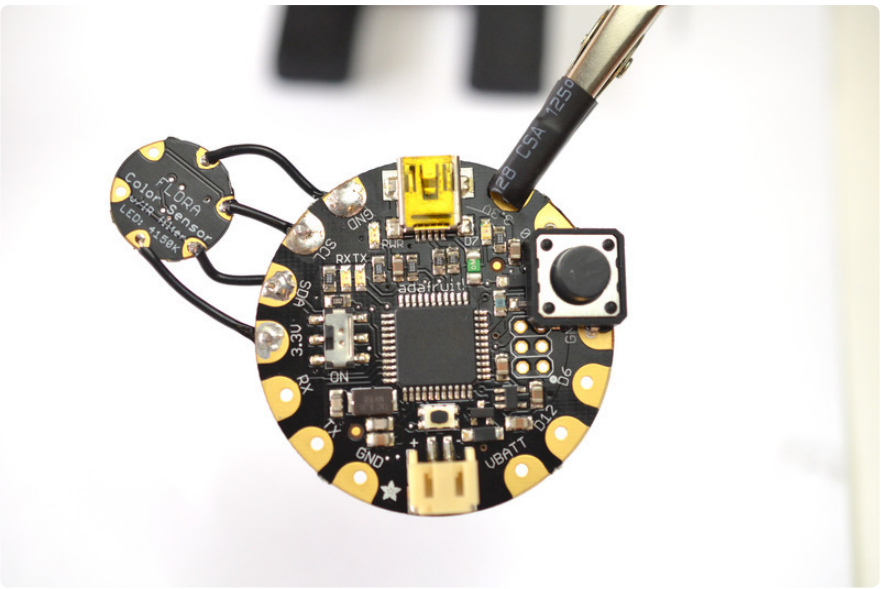


Trim two diagonal pins from a tactile switch, and then poke the remaining two legs through **D10** and **GND** on FLORA.

Solder in place.

The resulting circuit will live in the edge of the umbrella, facing the color sensor to the outside and the FLORA's power switch and the mode-selecting button you soldered facing in towards you as you use it.

Now that your color sensor is attached, you can test it out by hooking up your NeoPixels and running the sample code included with the color sensor (https://adafru.it/dj8) library, the Chameleon Scarf code (https://adafru.it/dj9), or the code included later in this guide.



Code

The FLORA version of this project uses free software called the Arduino IDE. This is available for PC and Mac and involves some download and possible configuration to set up the environment needed to program the FLORA itself.

If you are new to Arduino, you may want to review the following guides:

- Getting Started with FLORA (https://adafru.it/dgt)
- Adafruit Arduino IDE setup guide (https://adafru.it/jDQ)
- All about Arduino libraries (https://adafru.it/dit)

Load the following sketch in the Adafruit Arduino IDE (https://adafru.it/aSZ). Requires the color sensor library (https://adafru.it/cb8) and NeoPixel library (https://adafru.it/dhw).

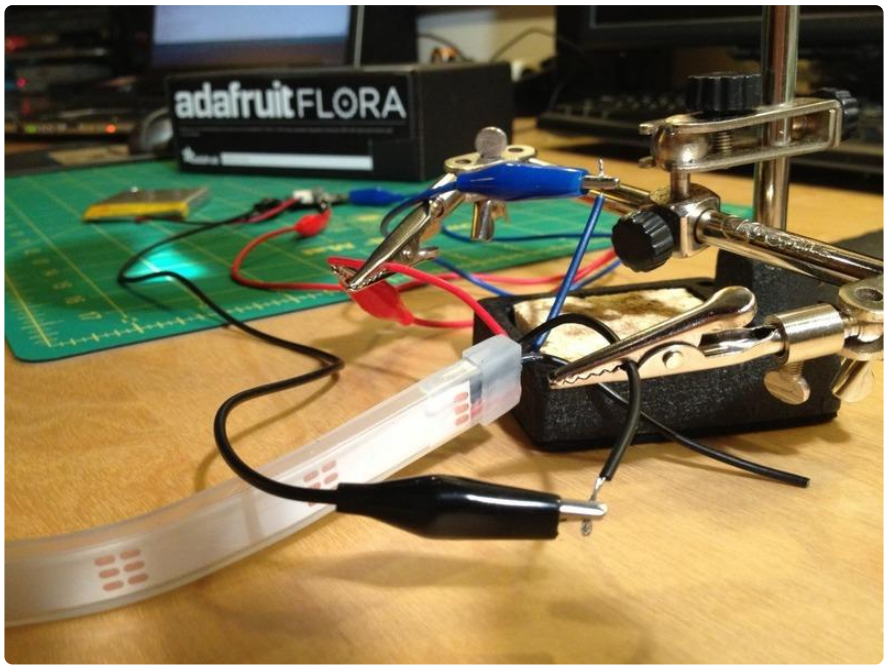
```
#include &lt;Wire.h&gt;
#include "Adafruit_TCS34725.h"
#include &lt;Adafruit_NeoPixel.h&gt;

#define PIN 6
#define TPIXEL 160 //The total amount of pixel's/led's in your connected strip/
stick (Default is 60)

int switchPin = 10; // switch is connected to pin 10
int val; // variable for reading the pin status
int val2;
int buttonState; // variable to hold the button state
int lightMode = 0; // how many times the button has been pressed

Adafruit_NeoPixel strip = Adafruit_NeoPixel(TPIXEL, PIN, NEO_GRB + NEO_KHZ800);
// our RGB -&gt; eye-recognized gamma color
byte gammatable[256];
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_50MS,
TCS34725_GAIN_4X);
```

```
void setup() {
Serial.begin(9600); // Set up serial communication at 9600bps
pinMode(switchPin, INPUT_PULLUP); // Set the switch pin as input
pinMode(PIN, OUTPUT);
strip.setBrightness(80); //adjust brightness here
buttonState = digitalRead(switchPin); // read the initial state
strip.begin();
strip.show(); // Initialize all pixels to 'off'
if (tcs.begin()) {
Serial.println("Found sensor");
} else {
Serial.println("No TCS34725 found ... check your connections");
while (1); // halt!
}
// thanks PhilB for this gamma table!
// it helps convert RGB colors to what humans see
for (int i=0; i&lt;256; i++) {
float x = i;
x /= 255;
x = pow(x, 2.5);
x *= 255;
gammatable[i] = x;
//Serial.println(gammatable[i]);
}
for (int i=0; i&lt;3; i++){ //this sequence flashes the first pixel three times as
a countdown to the color reading.
strip.setPixelColor (0, strip.Color(188, 188, 188)); //white, but dimmer-- 255 for
all three values makes it blinding!
strip.show();
delay(1000);
strip.setPixelColor (0, strip.Color(0, 0, 0));
strip.show();
delay(500);
}
uint16_t clear, red, green, blue;

tcs.setInterrupt(false); // turn on LED

delay(60); // takes 50ms to read
tcs.getRawData(&amp;red, &amp;green, &amp;blue, &amp;clear);
tcs.setInterrupt(true); // turn off LED
Serial.print("C:\t"); Serial.print(clear);
Serial.print("\tR:\t"); Serial.print(red);
Serial.print("\tG:\t"); Serial.print(green);
Serial.print("\tB:\t"); Serial.print(blue);
// Figure out some basic hex code for visualization
uint32_t sum = red;
sum += green;
sum += blue;
sum = clear;
float r, g, b;
r = red; r /= sum;
g = green; g /= sum;
b = blue; b /= sum;
r *= 256; g *= 256; b *= 256;
Serial.print("\t");
Serial.print((int)r, HEX); Serial.print((int)g, HEX); Serial.print((int)b, HEX);
Serial.println();

Serial.print((int)r ); Serial.print(" "); Serial.print((int)g);Serial.print(" ");
Serial.println((int)b );
 colorWipe(strip.Color(gammatable[(int)r], gammatable[(int)g], gammatable[(int)b]),
0);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i&lt;strip.numPixels(); i++) {
```

```
    strip.setPixelColor(i, c);
    strip.show();
    delay(20);
  }
}

void loop(){
  //rain();
  val = digitalRead(switchPin); // read input value and store it in val
  delay (20);
  val2 = digitalRead(switchPin);
  if (val == val2) {
    if (val != buttonState &amp;&amp; val==LOW) { // the button state has changed!
        if (lightMode == 0) {
          lightMode = 1;
        }
      else if (lightMode == 1) {
          lightMode = 2;
        }
      else if (lightMode == 2){
          lightMode = 3;
          //delay (20);
        }
      else if (lightMode == 3) {
          lightMode = 0;
        }
      }
    }

buttonState = val; // save the new state in our variable
if (lightMode == 0) {
strip.show();
}
if (lightMode == 1) {
rainbow(10);
delay(20);
}
if (lightMode == 2) {
rain();
delay(20);
}
if (lightMode == 3) {
rainbowCycle(10);
delay(20);
  }
}
// Rain Program

void rain() {
// Create an array of 20 raindrops
  const int count = 20;
  int pos[count];
// Set each rain drop at the starting gate.
// Signify by a position of -1
for( int i=0; i &lt; count; i++) {
pos[i]=-1;
}
// Main loop. Keep looping until we've done
// enough "frames."
boolean done=false;
int counter = 0;
while(!done) {
// Start by turning all LEDs off:
for(int i=0; i&lt;strip.numPixels(); i++)
strip.setPixelColor(i, 0);

// Loop for each rain drop
for( int i=0; i &lt; count; i++) {
// If the drop is out of the starting gate,
```

```
// turn on the LED for it.
if( pos[i] &gt;= 0 ) {
strip.setPixelColor(pos[i], strip.Color(0,0,127));
// Move the drop down one row
pos[i] -= 7;
// If we've fallen off the strip, but us back at the starting gate.
if( pos[i] &lt; 0 )
pos[i]=-1;
}
// If this drop is at the starting gate, randomly
// see if we should start it falling.
if ( pos[i] == -1 &amp;&amp; random(40) == 0 &amp;&amp; counter &lt; 380 ) {
// Pick one of the 6 starting spots to begin falling
pos[i] = 143-random(6);
}
strip.show();
delay(2);
}

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

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

// Rainbow Cycle Program - Equally distributed
void rainbowCycle(uint8_t wait) {
uint16_t i, j;

for(j=0; j&lt;256*5; j++) { // 5 cycles of all colors on wheel
for(i=0; i&lt; strip.numPixels(); i++) {
strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) &amp; 255));
}
strip.show();
delay(wait);
}
}
// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
if(WheelPos &lt; 85) {
return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
} else if(WheelPos &lt; 170) {
WheelPos -= 85;
return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
} else {
WheelPos -= 170;
return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
}
}
```

Test the NeoPixel Strip



Let's start by lighting up 5 meters of NeoPixel strip, just to be sure it's working properly (and it's just an awesome experience). To do this, you will need your alligator clips, the third hand, your NeoPixel strip, FLORA and battery.

This is just a friendly reminder that you need to review the FLORA (https://adafru.it/aSZ) and NeoPixel guides (https://adafru.it/cEz) to perform this test. You don't want to damage anything, right?

Configure the FLORA

Connect the FLORA to your computer using the USB cable. Follow the instructions on the Flora tutorial (https://adafru.it/aSZ) and don't forget to install the NeoPixel library. Then, load the following modified version of the NeoPixel example **"strandtest"**. Once it is loaded, you can turn off the power and unplug the USB cable.

```
#include &lt;Adafruit_NeoPixel.h&gt;

#define PIN 6

// Parameter 1 = number of pixels in strip
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(160, PIN, NEO_GRB + NEO_KHZ800);

void setup() {
  strip.begin();
```

```
  strip.setBrightness(30); //adjust brightness here
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:
  colorWipe(strip.Color(255, 0, 0), 50); // Red
  colorWipe(strip.Color(0, 255, 0), 50); // Green
  colorWipe(strip.Color(0, 0, 255), 50); // Blue
  rainbow(20);
  rainbowCycle(20);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i&lt;strip.numPixels(); i++) {
      strip.setPixelColor(i, c);
      strip.show();
      delay(wait);
  }
}

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

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

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j&lt;256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i&lt; strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) &amp; 255));
    }
    strip.show();
    delay(wait);
  }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
  if(WheelPos &lt; 85) {
   return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
  } else if(WheelPos &lt; 170) {
   WheelPos -= 85;
   return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  } else {
   WheelPos -= 170;
   return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
}
```

Test the NeoPixel Strip

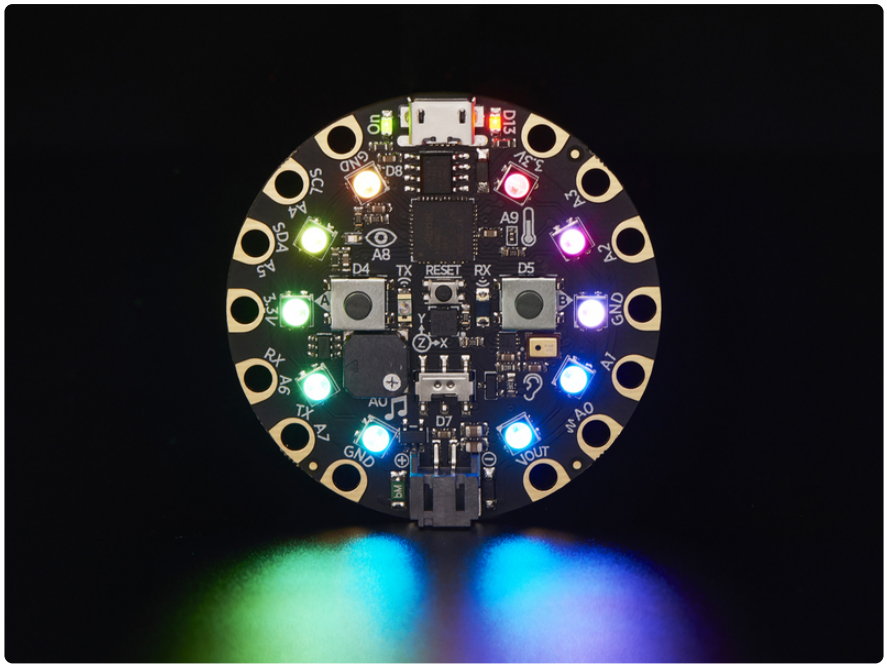
Unwind your NeoPixel strip on a table or other flat surface. You may want to tape it here and there with masking tape just to keep it from moving. Since you have a 5

meter quantity of strip, you will see test leads on one end. Hook up the LED strip to the FLORA micro-controller with alligator clips in this fashion (the third hand may be helpful here to keep the wires separated).

- Black alligator clip - ground (GND) pin on strip --> ground (GND) pin on FLORA
- Red alligator clip - 5V pin on strip --> VBATT pin on FLORA
- Blue alligator clip - DIN pin on strip --> D6 on FLORA

Make sure you double check the sketch to be sure the quantity of NeoPixels (160) and the pin number for the FLORA are set correctly (usually 6 or 12). Set the brightness low to start out (30). Also, check that the arrows on the NeoPixel strip are pointing away from the FLORA. Finally, take one last look at the alligator clips to be sure they are not touching each other. Now, plug the battery into the FLORA and switch on the power. You should see a fun LED show with different patterns. If not, don't worry, just double check your connections and double check the sketch you loaded into the FLORA. Got it? If so, it's time to move on to the circuit.
(https://adafru.it/dj3)

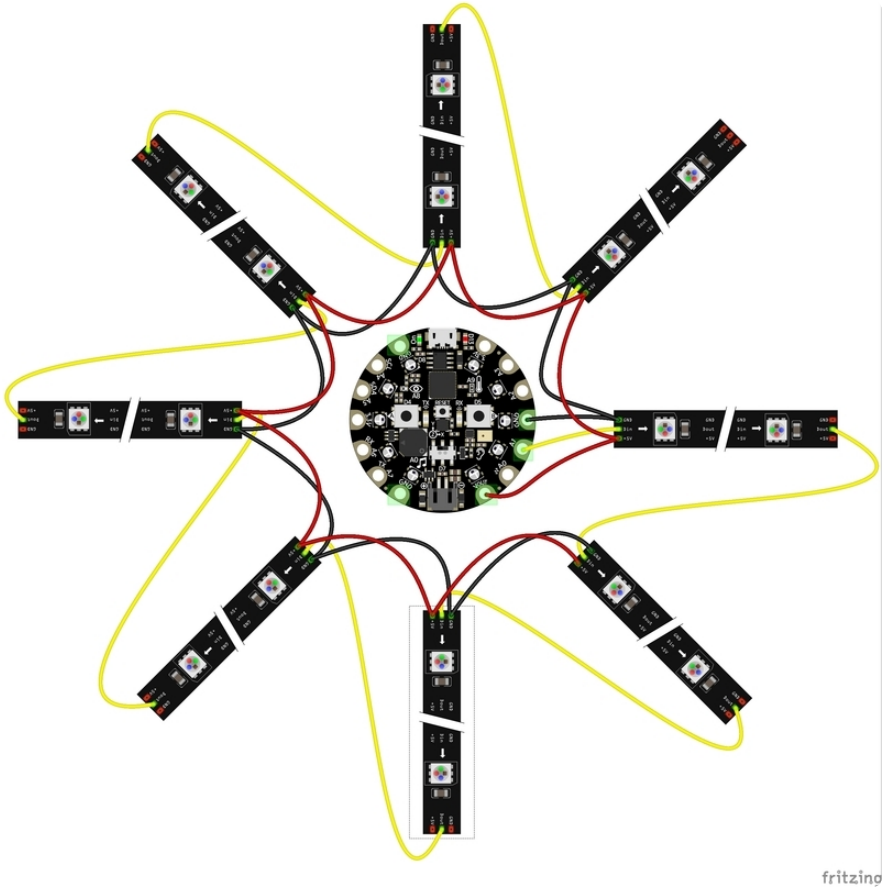
Make with Circuit Playground



The Flora is a fabulous board, but the newer Circuit Playground Express makes this kind of project much easier. It already has an onboard color sensor and two onboard buttons, so you can get the same functionality (and more) with a lot less wiring and soldering.

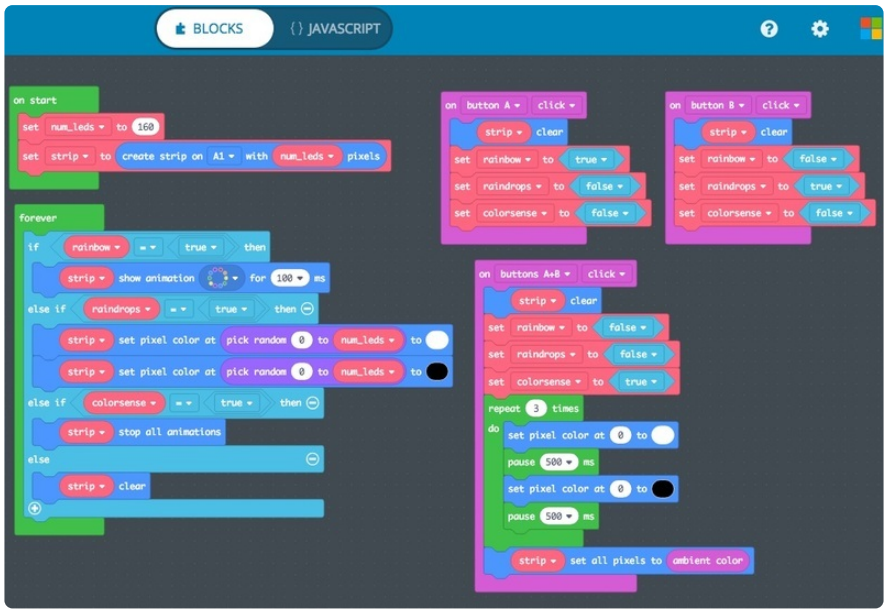
As a bonus, the Circuit Playground Express can be coded with the Microsoft MakeCode editor. This is a drag-and-drop code editor that makes it fun to play around with pieces of code, and build stuff. It feels a bit like building with Lego, instead of making things from scratch. It's hard to get wrong, and there are endless possibilities for drag-and-drop experimentation.

Wiring Diagram



Wiring with a Circuit Playground Express is a bit simpler. The LED strip wiring will stay the same, but wire the data line to **A1.** The Circuit Playground already has onboard buttons and a color sensor, so we don't need to add those components at all.

Code with MakeCode



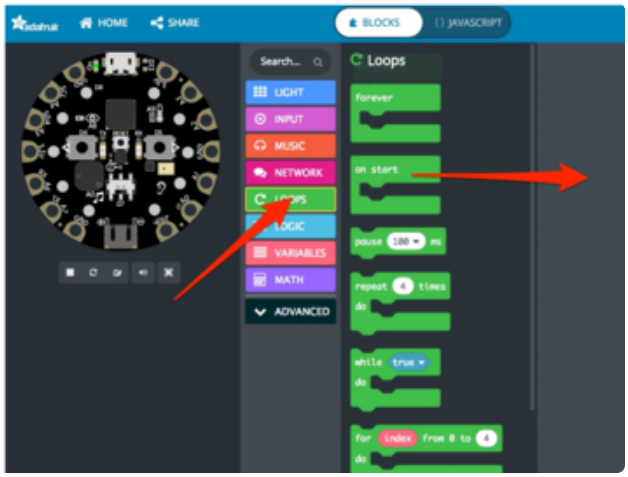
Microsoft's MakeCode editor is a great way to get your Circuit Playground Express to do things without having to muck about in actual code. It's a fun drag-and-drop editor that's easy to use and sneakily teaches you how to write code while you're making pretty things that light up.

The MakeCode project linked below works in a similar way to the Flora-and-Arduino project but uses the Circuit Playground's onboard buttons and sensors so you don't have to do as much wiring. You can download the completed code and get up and running right away, or you can follow along and create your own code, adding your own customizations as you go.

Complete FloraBrella MakeCode

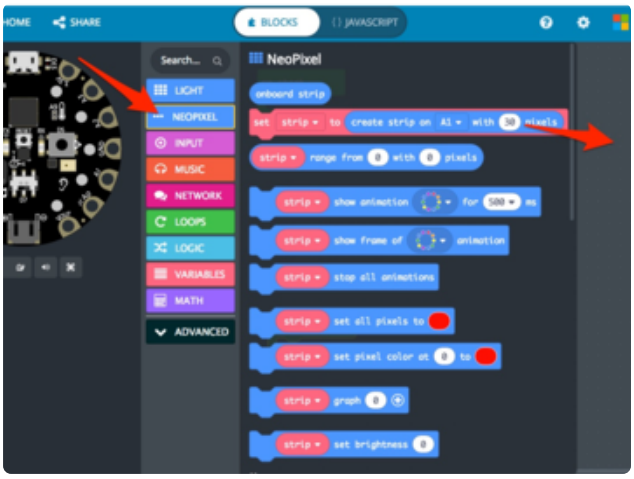
https://adafru.it/DGj

Our umbrella has three different color modes: a color-sensing mode, a raindrops animation, and a rainbow animation. This will make our MakeCode project a bit complex, so if it's your first time using MakeCode, head over to the MakeCode Intro Guide (https://adafru.it/AEp) and play around with a couple of the beginner projects to get a feel for how it all works. Once you're ready to go, head over to www.makecode.com, (https://adafru.it/DGk) roll up your sleeves and let's dive in.



Click the Circuit Playground Express and then **New Project**. You'll see your workspace, with a Circuit Playground on the left, some tabs in the middle and your workspace on the right. Click the **LOOPS** tab, and find `on_start`. Drag it into your workspace.

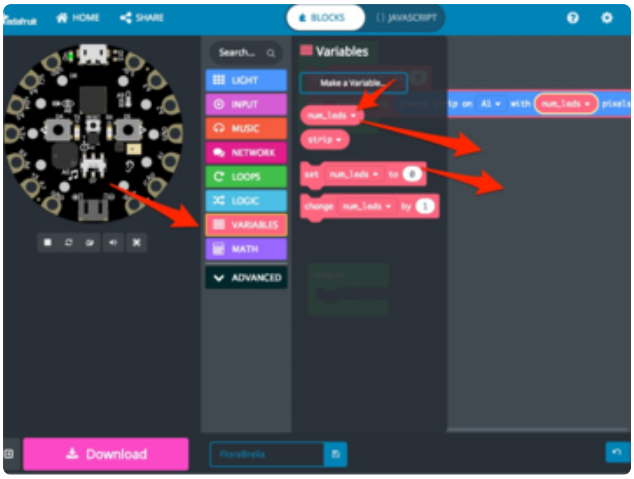
Whatever you put into the `on_start` loop will run once, when you first boot up your Circuit Playground Express. Whatever you put in the `forever` loop will run over and over again, forever.

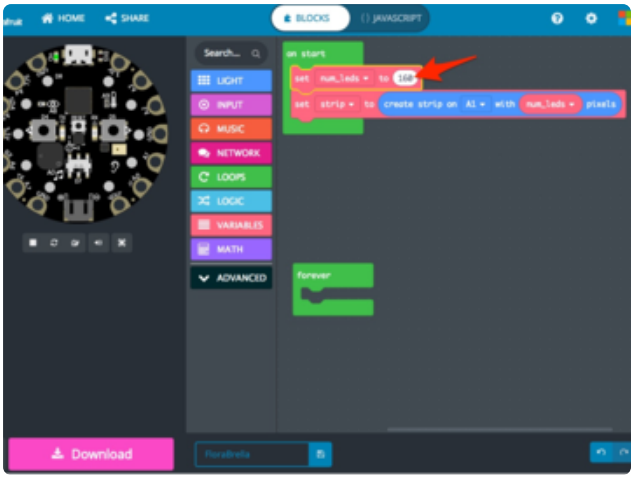


Let's tell the Circuit Playground Express that we attached some lights to pin **A1**. Click the **LIGHT** tab and another sub-tab will appear labeled **NEOPIXEL**. Anything in the **LIGHT** tab refers to the lights on the face of the Circuit Playground Express itself. Anything in the **NEOPIXEL** tab will refer to lights that you add. Find `set strip to create strip on A1 with 30 pixels` and drag it onto your workspace inside the `on_start` loop.

Variables

A variable is a placeholder or container for a number. We use them so we can change numbers willy-nilly, on the fly, or in multiple places at once. We'll set up several variables now so we can use them later on in our code. The first one we'll call `num_leds` and it will refer to the number of LEDs in our strip. This way, if you ever decide to add more LEDs or take some away, you'll only need to change this number in one place in your code, and the rest of the code will read the variable and know what you mean.





Click the **VARIABLES** tab and create a new variable called `num_leds`. Drag an instance of set `num_leds` to **0** into your `on_start` loop at the top. Then go back to the **VARIABLES** tab and grab an instance of `num_leds`, and use it to replace the number in the `create strip` block. Set `num_leds` to **160** (or however many pixels you've got on your umbrella).

This seems a bit overly complicated when you're first starting out, but getting comfortable using variables in this way will make a lot more complex code possible down the road. And it will eventually make your life easier, I promise!

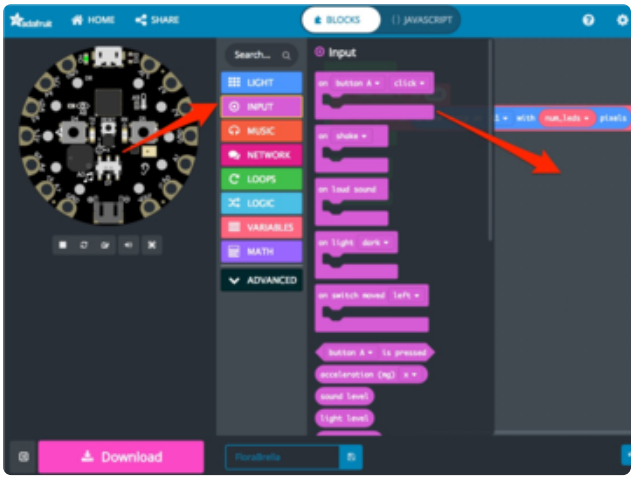
Our umbrella code has three different color modes, so let's go ahead and make a variable for each color mode now so we can refer to them easily in the future.

Click the **VARIABLES** tab and create a variable called `rainbow`, one called `raindrops`, and one called `colorsense`. Don't do anything else with them yet.

Inputs

The Circuit Playground Express has a whole lot of ways to trigger effects. The easiest to use for this project are the onboard buttons. We've got `button A` and `button B`, and another trigger for `buttons A+B` pressed at the same time. We also have three modes, so it looks like this will work out perfectly.

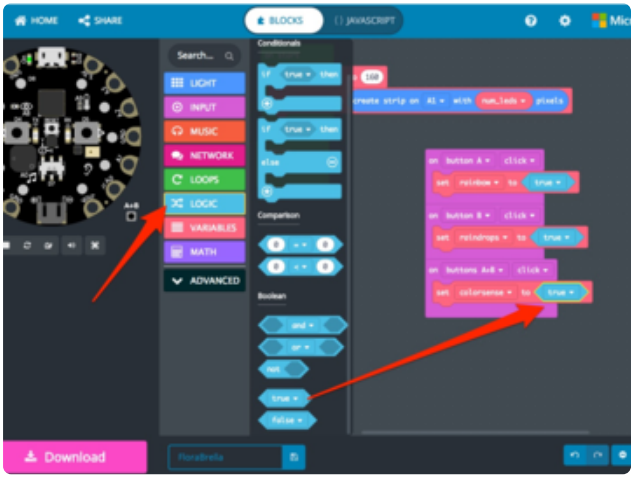
I've decided to assign **button A** to rainbow mode, **button B** to raindrop mode, and the **double-button press** to color sense mode. Select your own inputs to customize your code to work the way you want.



Head to the **INPUT** tab and drag an instance of `on button A click` into your workspace. Drag a second and third instance out as well, and change the dropdown to read `button B` and `buttons A+B`. Whatever you put inside each loop will be triggered by the action you chose.

Click the **VARIABLES** tab. Drag an instance of `set rainbow to 0` into the `button A` input loop. (You may need to find it in the dropdown menu). Drag another instance into the `button B` loop and set it to `raindrops`. Set `buttons A+B` to `colorsense`.

We want to be able to turn the modes on and off. We'll use a `true/false` modifier to tell the Circuit Playground Express which mode we want at what time.



Click the **LOGIC** tab and scroll down until you find `true`. Replace each of the `0` fields in the three modes with `true`.

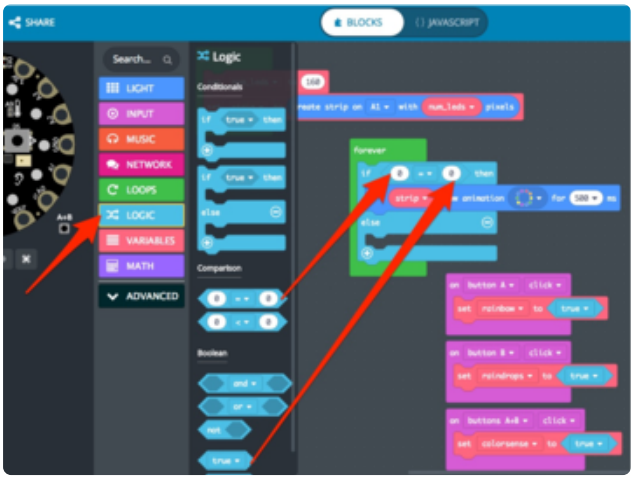
Next we'll create the modes themselves. We'll run them inside the `forever` loop so they run continuously until we press a different button.

Click the **LOGIC** tab again and find the `if / then / else` block. Drag it inside your `forever` loop. Now we can tell the Circuit Playground Express what to do when each different mode we set up is "true".

Rainbow Mode

MakeCode already has a few onboard animations built in, and one of them is a Rainbow animation. Perfect! We can use this as one of our modes.

Drag an instance of `strip show animation (rainbow)` from the **NEOPIXEL** tab into your `forever` loop, inside the `if/then` statement. Then, head back to the **LOGIC** tab. Find the Comparison block `0=0` and drag it into your `if/then` statement, replacing the `true` that was there as the default. Then grab another instance of `true` from the **LOGIC** tab and replace the second `0`.



It can be a little tricky to drag these blocks into each other. Look for the highlighted outline when you're hovering with your mouse, to know you're dragging to the right spot.

Lastly, head back to the **VARIABLES** tab and grab an instance of the `rainbow` variable we made earlier. Drag it into the first comparison spot, replacing the `0`. Now the code basically says, "If rainbow = true, then play the rainbow animation" -- and remember, pressing **button A** makes `rainbow = true`. We're coding like pros! Let's download it and see if it works.

Give your code a name (I called mine FloraBrella) and save it. Click the Download button and a file will download to your computer. Plug your Circuit Playground Express into your USB port and click the **reset** button. You should see all the lights on the face of your Circuit Playground Express turn green, and a drive will appear on your computer called **CPLAYBOOT**. If you see a drive called **CIRCUITPY**, press the **reset** button a second time to get to **CPLAYBOOT**.

Drag the downloaded file onto this drive to program the Circuit Playground Express.

Now, press **button A**. Did a rainbow appear? It's like MAGIC! Yay.

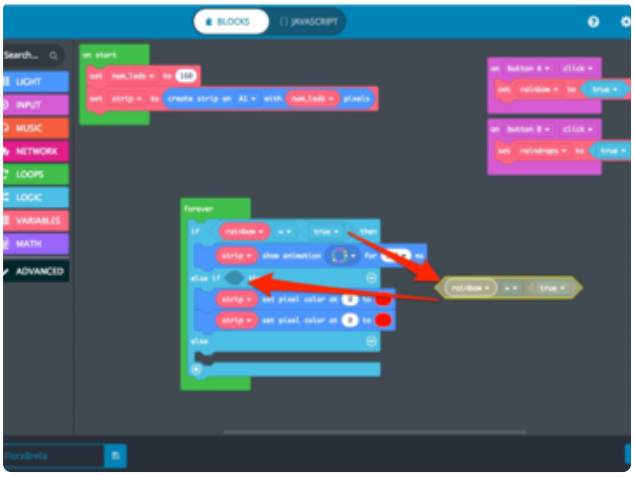
If you're having trouble with the download process, head over to the Circuit Playground Express (https://adafru.it/AQW) guide for some more detailed instructions and troubleshooting ideas.

Once you've got it working, let's add some more modes.

Raindrop Mode

MakeCode has a Twinkle animation that you could use as a raindrop mode. Just repeat the steps above but select the Twinkle animation instead of the Rainbow animation. It's pretty, but I don't think it looks all that much like raindrops. So I created a simple mode that randomly adds and removes white lights and looks a bit more like rain spatter to me.

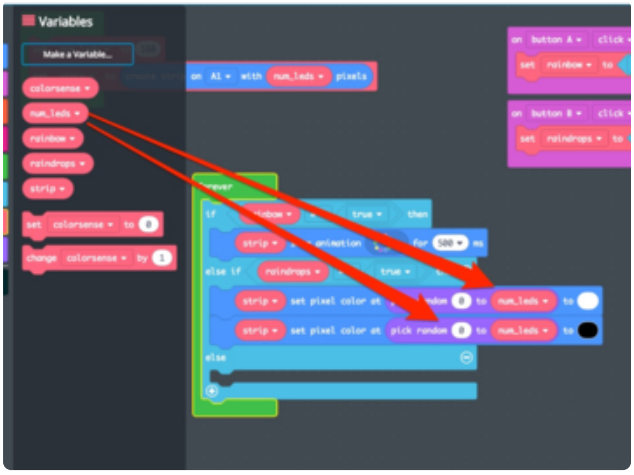
First click the `+` button at the bottom of your `if/then` statement to add another slot. Then drag two instances of `strip set pixel color at 0` to red inside the slot you created.



Select the conditional statement from the `rainbow` mode slot and copy it to your clipboard. Paste an instance of it directly to your workspace. Now you can drag this new instance into the new slot you're working and change the variable to `raindrop`. Anything that happens in this slot will happen if `raindrop = true` (when you press **button B**).

We have two instances of `strip set pixel color` so we can turn the pixels both **on** (white, or whatever color you'd like your raindrops to be) and **off** (black). Select the colors you want here.

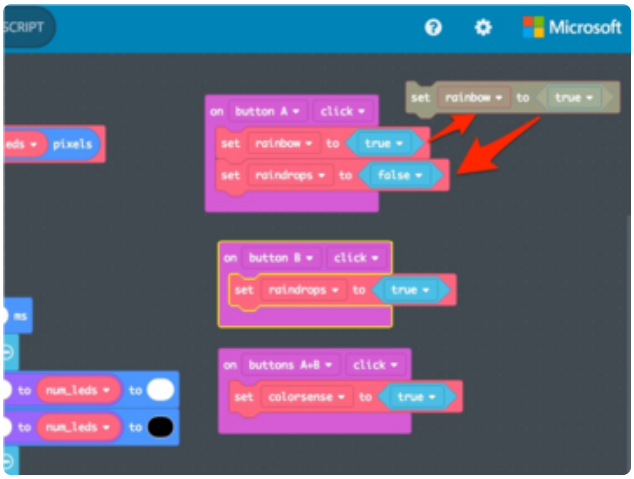
I want my raindrops to spatter and disappear randomly, so we can add randomness from our **MATH** tab to get this effect. Replace the `0` in both of these blocks with `pick random 0 to 10` from the **MATH** tab.

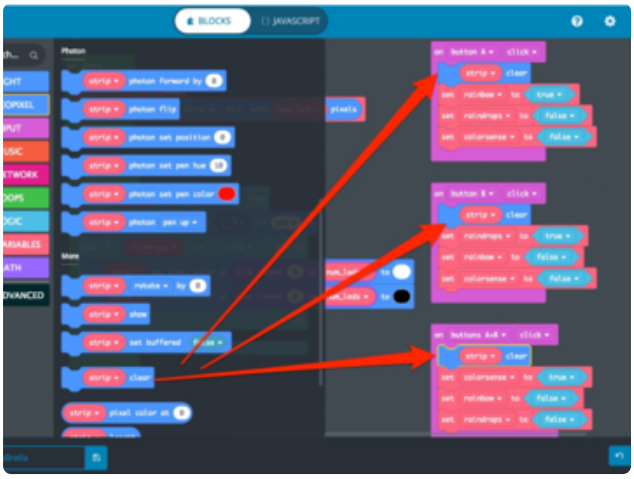


Head back to the **VARIABLES** tab and drag an instance of `num_leds` to replace the `10` in each of these blocks. Now we'll have a random number chosen from our entire LED strip instead of just the first 10 pixels. (See? I told you variables were useful!)

Let's download and test our code to see if we like it! Download to the Circuit Playground and press **button B**. Pretty raindrops! Press **button A**. A rainbow! This is fun. Press **B** again and... nothing happens. What's going on here?

We've told our animations to trigger when `rainbow = true` or `raindrops = true`. We need a way to turn the animations back **off** (`false`) when we're not using them, so they don't try to all run at the same time. One way to do this is to make each input / button press turn the other modes off while turning its favorite mode on. This will make it so we can toggle between modes.





Find your input blocks. You can use the copy/paste method here -- copy an instance of `set rainbow to true` and place it inside the loop, changing it so it reads `set raindrops to false`. Make another one for the `colorsense` mode. Repeat in the other input blocks, so that each block sets its own mode to `true` and the other two modes to `false`.

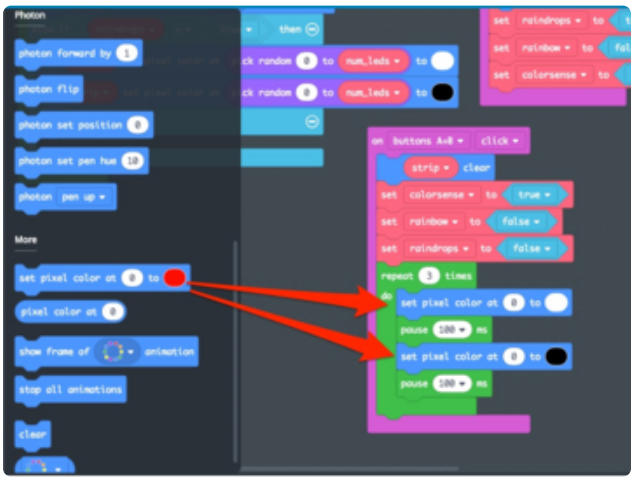
To be sure the old mode goes away, let's also add an instance of `strip clear` to each input as well. This will turn all the lights off for just a moment.

Download again and try it now. Success! We can toggle between the two modes. Let's go ahead and add our third mode -- color sensing.

Color Sensing Mode

The Circuit Playground Express has an onboard color sensor we can use to capture a color and make the umbrella match. We'll make the onboard NeoPixel next to the sensor blink 3 times to indicate the color sensing is about to happen, then tell the NeoPixel strip to show that color.

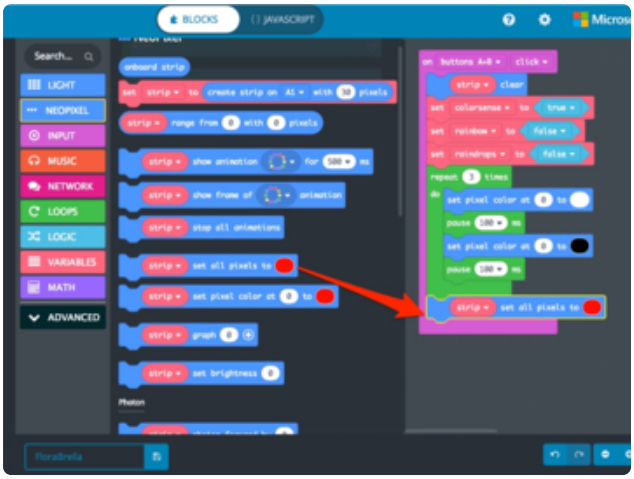
We'll do the sensing in the input (`on click`) loop since we only want it to run once (not forever, over and over).



First we'll make the NeoPixel light #0 on the Circuit Playground Express' face blink 3 times. Grab a `repeat 4 times` loop from the **LOOPS** tab and put it into your `on buttons A+B click` block at the end. Change the `4` to `3`.

Now drag 2 instances of `set pixel color to red` from the **LIGHT** tab (not the **NEOPIXELS** tab this time. Remember, we want to affect the onboard lights, not the strip). Change first the `red` to `white` (or whatever color you'd like the blinking to be) and the second `red` to `black` (for off).

Grab 2 instances of `pause 100ms` from the **LOOPS** tab and place them in between. Change `100ms` to `500ms`, to make them blink on and off every half-second.



Now we'll sense the color and set our NeoPixel strip to match. Drag an instance of **NEOPIXEL** > `strip set pixel color` into your input loop outside the `repeat` loop. (If you put it inside, it would sense three times).

Finally, go to **INPUT** and drag out an `ambient color` variable, and use it to replace the `red` color.

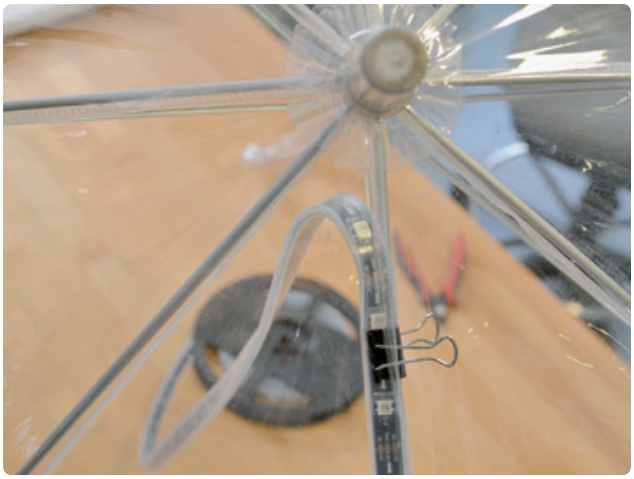
Back in our **FOREVER** loop, we need to repeat what we did before and tell the Circuit Playground to watch for when `colorsense` is `true`. You can use the copy/paste method again to make your **FOREVER** loop look like the example. I also added a `strip stop all animations` block inside this loop, just in case the animations get wonky.

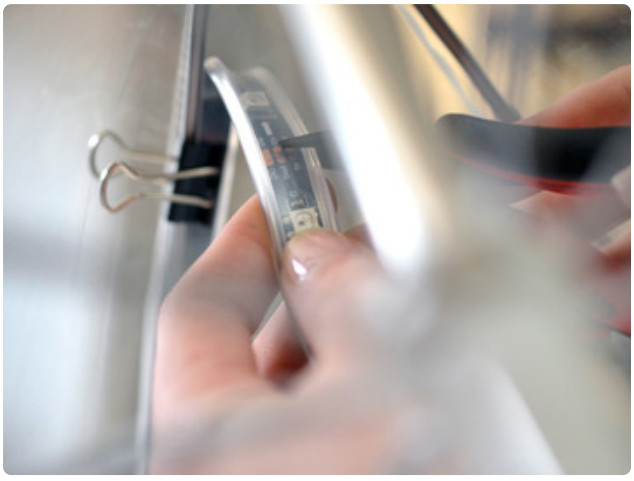
In the last `else` block, I placed a `strip clear`. In this case, the umbrella will be dark unless a button is pressed. If you want an animation that happens when no button has been pressed, you can place it here.

Download your code and play around with it. See if it does what you want. If it doesn't quite meet your specifications, mess around and change stuff! You can always come back to the original project if you break it, but breaking code is the best way to learn to fix it, in my experience.

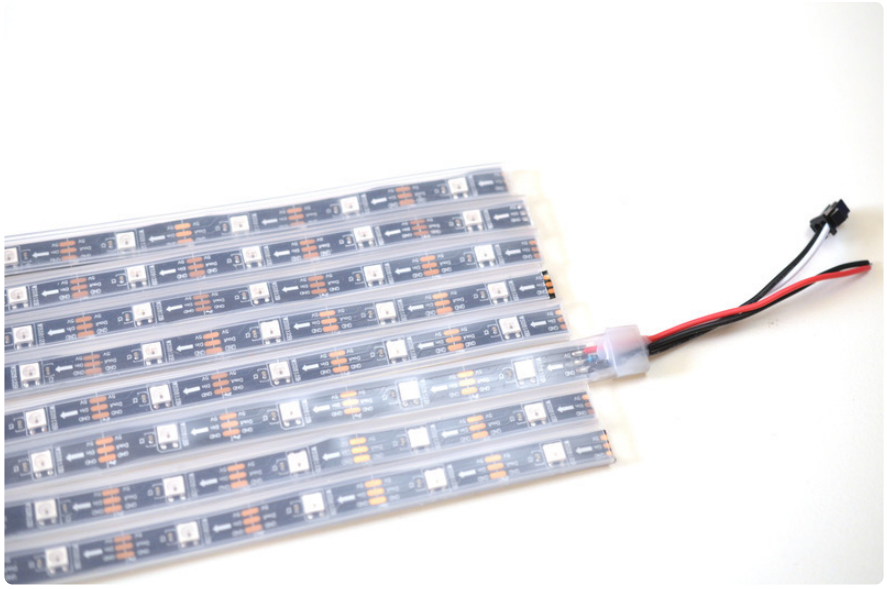
What animations will your umbrella show?

Prepare NeoPixel Strip

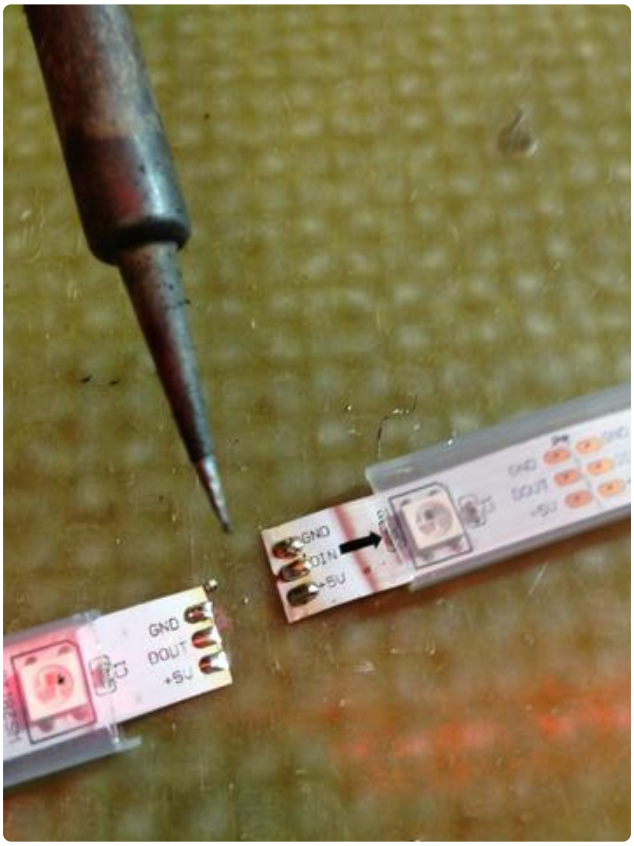




Unwind a section of NeoPixel strip from your spool and clip it into one rib of your umbrella. Decide how long you want each strip, and clip with diagonal cutters at the mark where the copper pads connect.



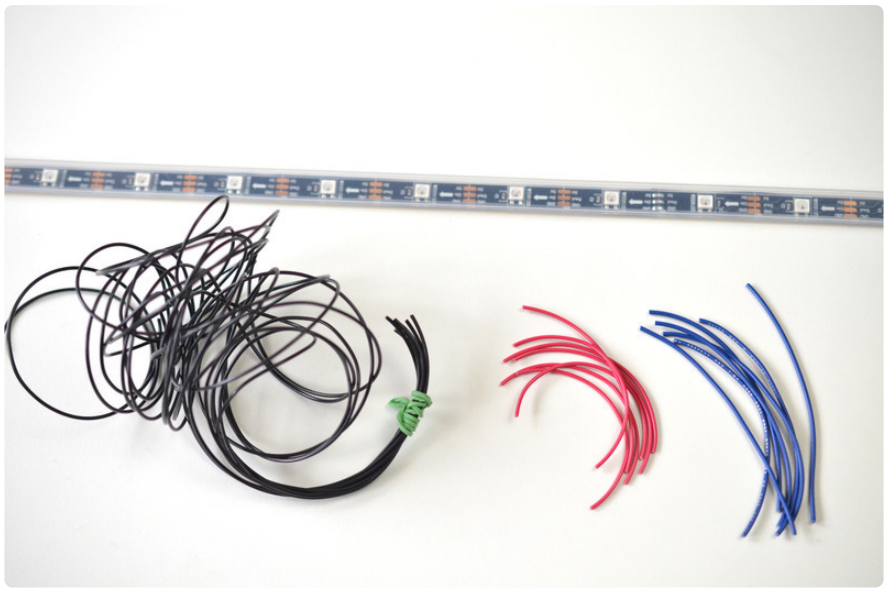
Cut eight strips of the same length.



Prepare the NeoPixel Strips

Grab your utility knife or scissors. Starting at the end of the NeoPixel Strip without the test wires, cut the strip into eight segments, each having 20 individual NeoPixels. You will see cut lines marked between each double set of copper pads, which makes it easy.

At some point you will run across a factory solder point where you need to make a cut. Get your soldering iron to a nice hot temperature and de-solder the joint, using copper braid. Have patience, because these joints are even stronger than you think. You will need to pull gently as you operate the soldering iron in order to be successful.



Prepare the Wires

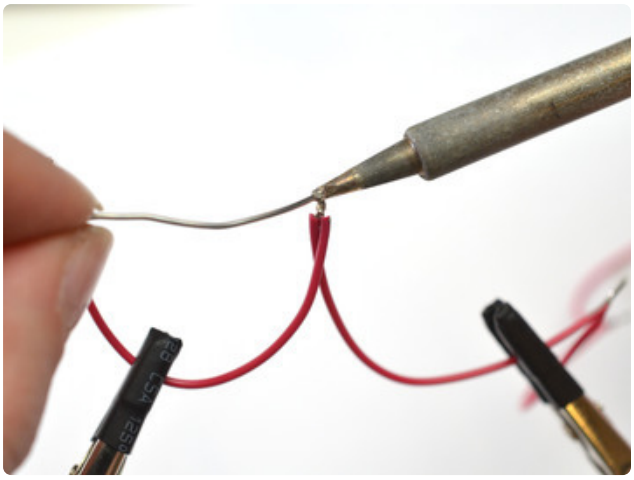
You will need quite a few wires for this project, so it's best to cut them now so you will be ready to solder. There are three different sizes of wire. Long wires will run down the entire length of the NeoPixel strips. Short wires which will be soldered in clusters of two on the strips in the center of the umbrella.

Now that you've double-checked the wires, go ahead and grab your snips and cut.

- 7 long wires - 29" each (choose white or black to match your NeoPixel strip)
- 6-7 short wires - 3 1/2" each (ground)
- 6-7 short wires - 3 1/2" each (power)

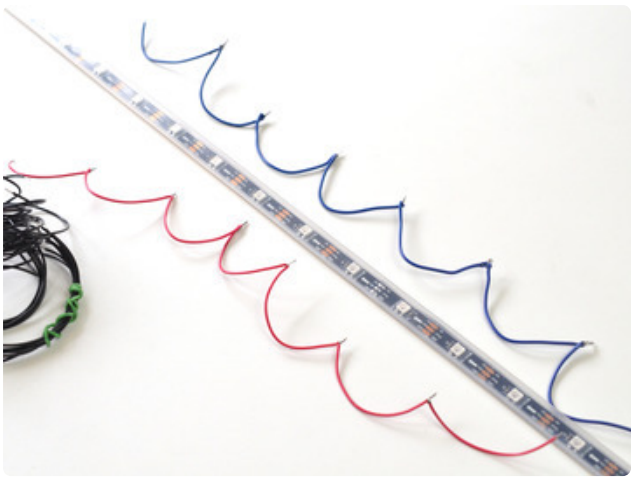
Now would be a great time to prep both ends of the wires using a wire stripper (1/4").

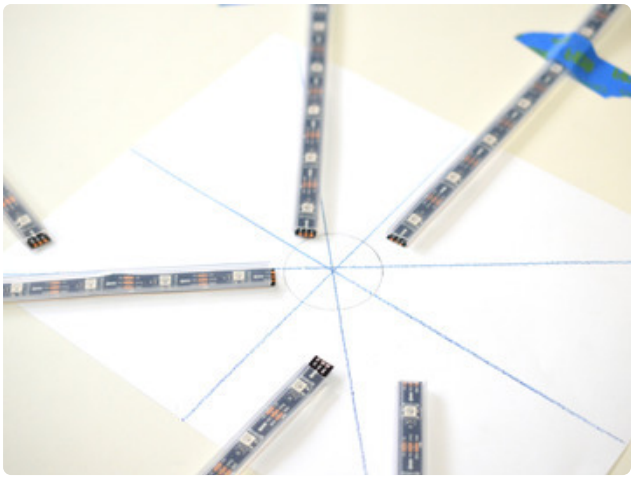
These wire lengths were calculated based on the Totes Dome Umbrella. If you are using a different brand, or you are not sure if it is the identical Totes model, you should double check the lengths by measuring with a piece of string or measuring tape.

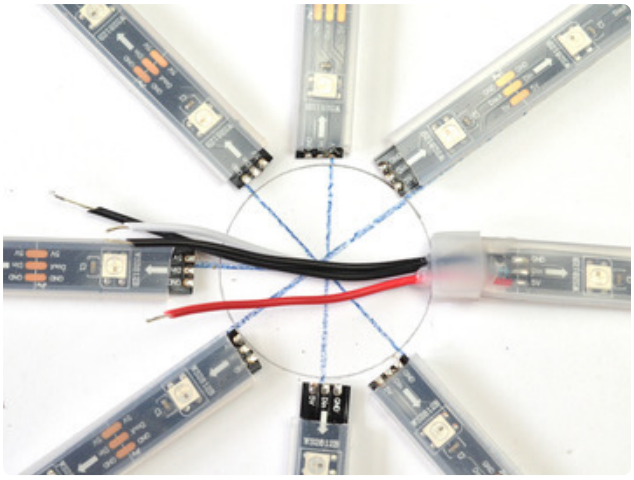


Solder the Wires

Take the short pieces of blue (**ground**) wire and gently twist the ends together as shown in the photo, then solder these connections. Repeat with the red (**power**) wires.

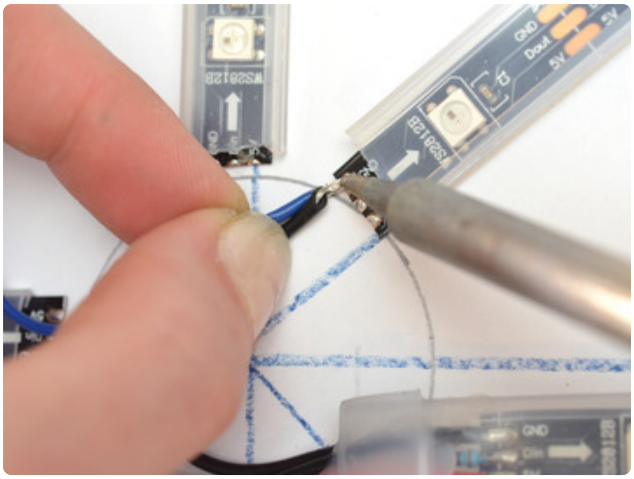


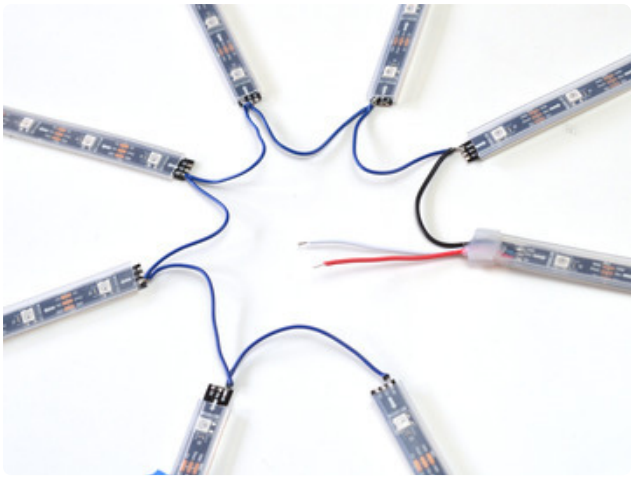




The strips are easiest to solder if you place them in a star-burst design with NeoPixels facing up on a large table. The directional arrows on all of the strips should start at the center and point outwards. Now, using a needle-nose pliers, carefully pull just a tiny bit of the LED strip out of its protective covering. That way you will be able to solder without melting the clear plastic.

Use the wired and sealed input end of one strip in your starburst and cut/strip the wires as shown.

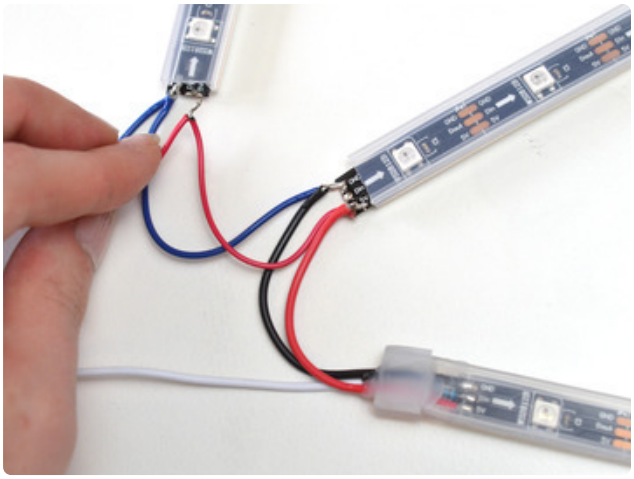


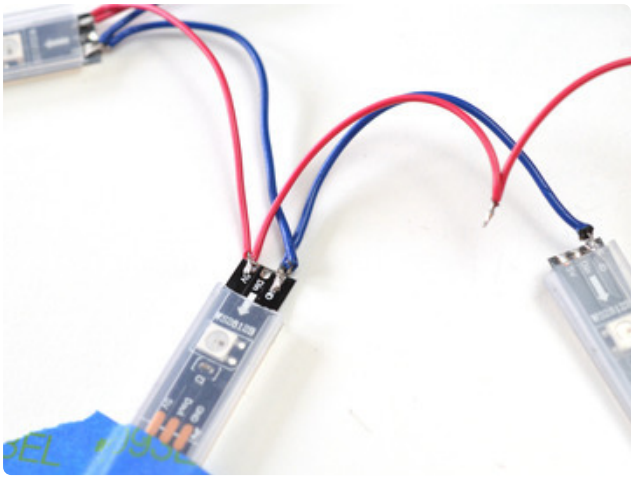


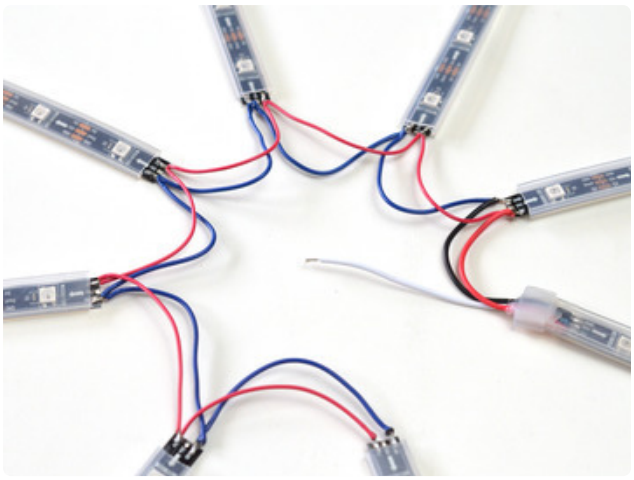
Start with your blue wire chain and one of the black ground wires attached to your NeoPixel strip, if you've got one.

Solder the ground connections on each strip by tinning the pad with solder, then heating both the wire and pad to flow the solder together.

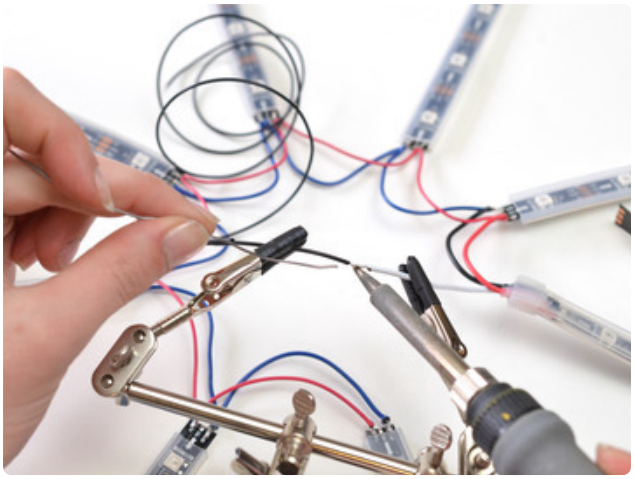
Repeat for all ground connections as shown.

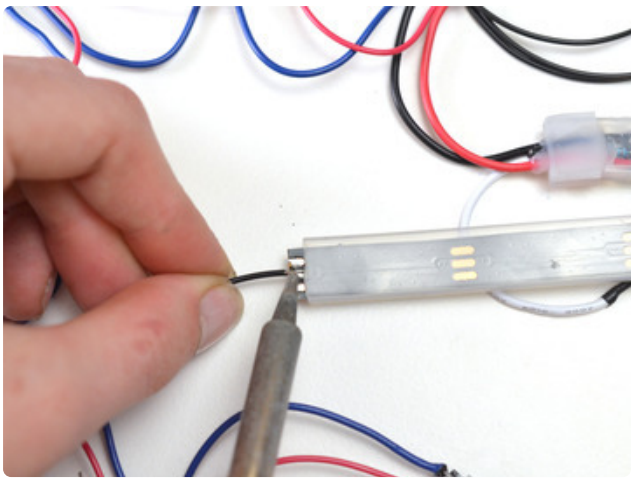


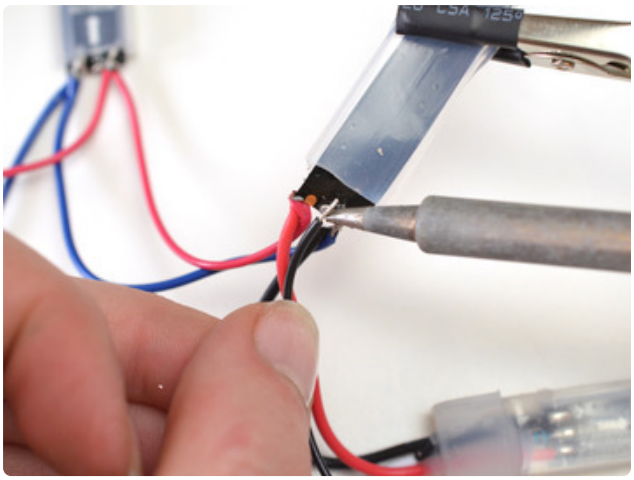




Repeat the whole process for the power connections on each strip, daisy chaining them together with the wires you prepared earlier.



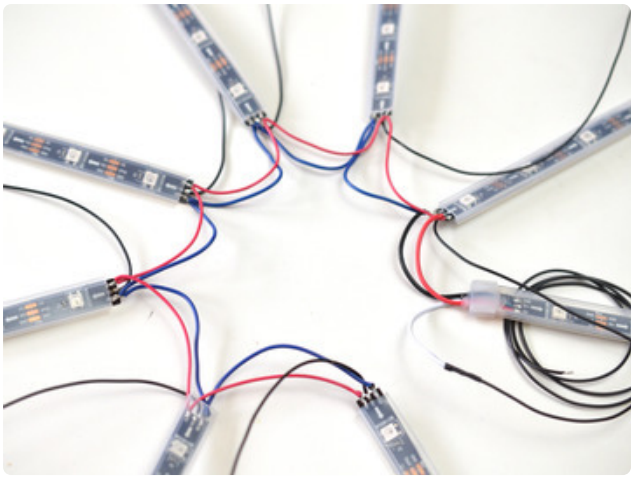


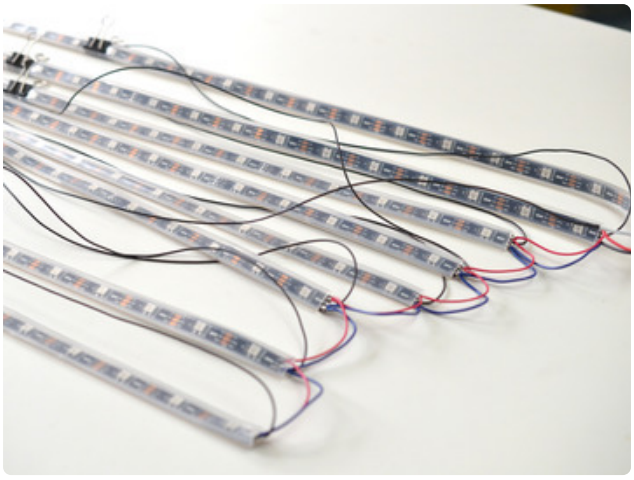


Solder one of your long wires to the first strip's input wire, and finish off with some heat shrink tubing. This wire will extend to the FLORA.

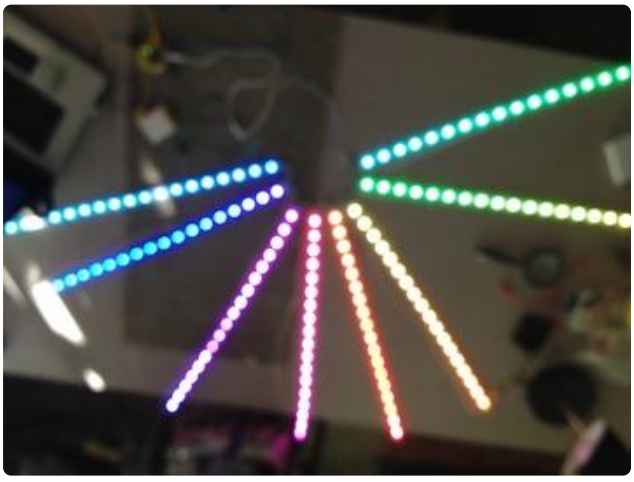
Now we will do a reverse move on the clear weatherproof covers on the NeoPixel strips. Carefully use your pliers to pull the strips back under the cover. You will need to over pull it so the pads at the bottom of the strips become exposed. Then, solder the other end of each of the white wires onto the DO pads. Once this task is complete, you can again pull the weatherproof covers back into their correct place.

Connect the other end of this long wire to the data input on its neighboring strip, using the underside of the flex PCB so the wire doesn't bump shoulders with the power or ground.





When you're finished, your wire connections should appear as shown. Double check that your solder joints are secure before arranging the strips in parallel to prep for umbrella-fying.



Now that you have things soldered, it would be good to try another test with the strips. Remember that the USB is not powerful enough to light up the strips, so once you load the program, plug the battery in and then turn it on. You should get a vibrant display. If not, use your multimeter to troubleshoot the circuit.

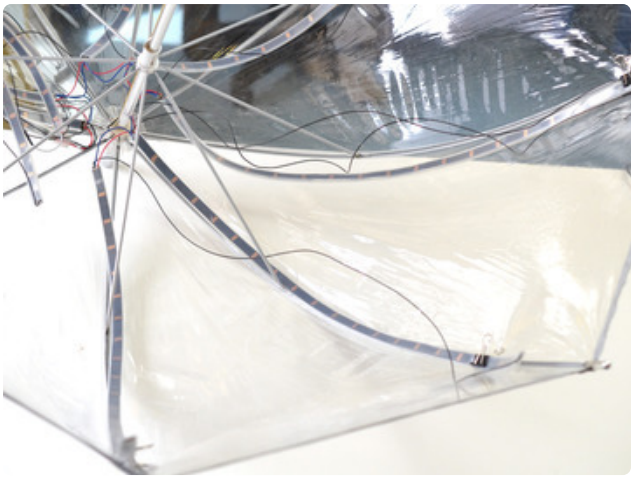
Now it's time to prepare the umbrella.

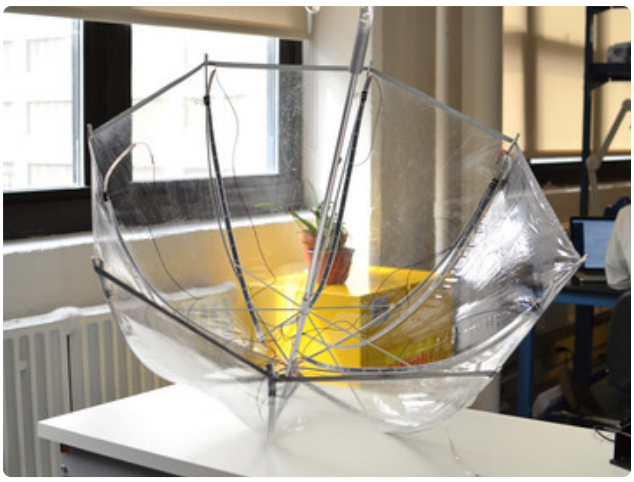
Insert the Circuit

Prepare the Umbrella

You will notice that the umbrella comes coated with a powder type substance. This was done intentionally to keep the vinyl from sticking to itself when it opens. So, do yourself a favor and leave it there. :) If you have already washed it off, you may want to add a little baby powder.

Cut the threads holding the vinyl to the umbrella, if there is any, using small scissors.

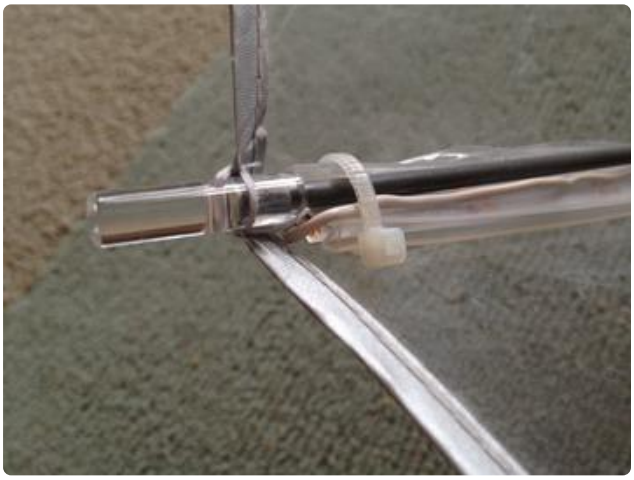




Insert the Circuit

Place your umbrella on the floor and gently move the circuit into its dome, making sure the front of the NeoPixel strips face the floor.

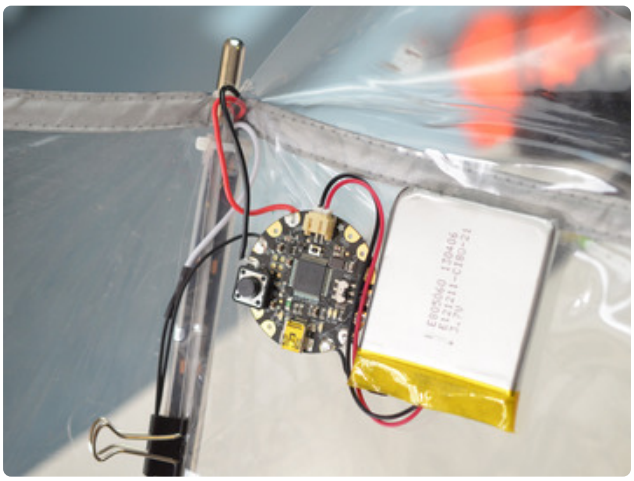
Now, take one LED strip and move it into place under one of the arms of the umbrella. Repeat with the rest of the strips. Note that this is a particularly fiddly task because the strips will want to shift. You may want to use masking tape or binder clips to secure as you work your way around. If you used solid-core wire, you may find you have to bend the wires slightly at the hub to ensure that they fit correctly.



Secure the NeoPixel Strips

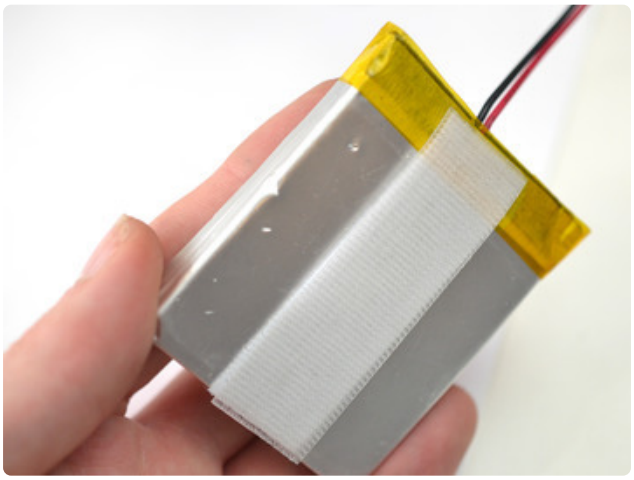
Next, we will make the strips secure by adding cable ties. Align the strip end to the end of the umbrella's rib and use just a few cable ties to preliminarily secure the strip. Be sure the ties don't cover any of the LEDs on the NeoPixel strip. Repeat this process on each rib. Leave each end of NeoPixel strip free from the umbrella, since we'll glue them later, once we know the circuit works.

Keep these strip ends away from the metal ribs, letting them hand freely for now, so they don't short out.



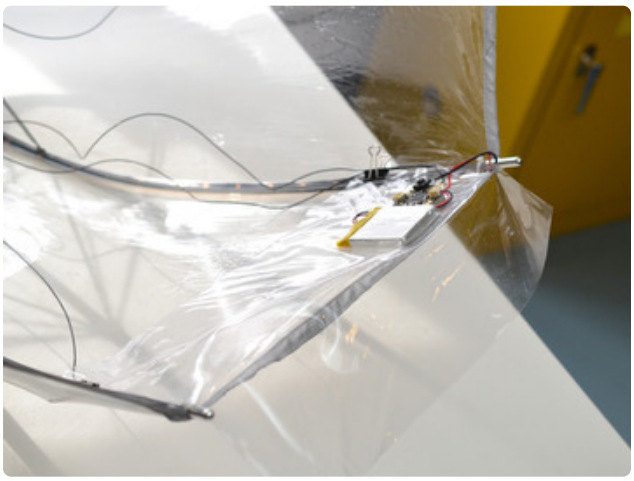
One of your strips is bound to have a connector end-- this is a great place to connect up FLORA or Circuit Playground Express.

FLORA: Solder the **red** wire to **VBATT**, the **black** wire to **GND**, and the **white** signal wire to **D6**.

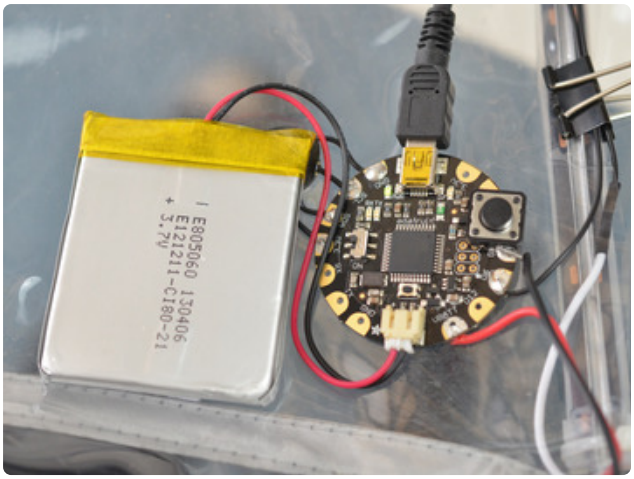


For Circuit Playground Express: the **red** wire to **VOUT**, the **black** wire to **GND**, the **white or yellow** signal wire to **A1**.

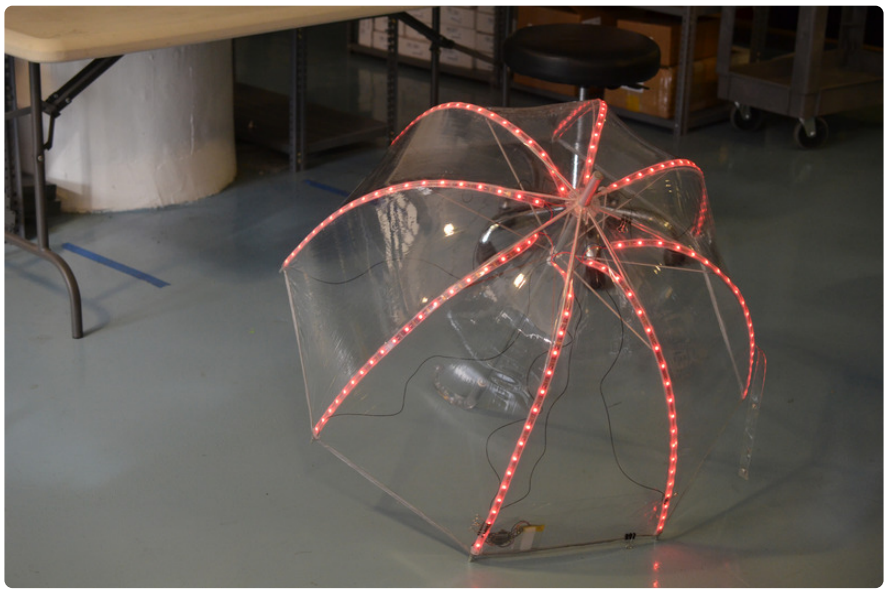
On a spare piece of vinyl you've binder-clipped into the edge of the umbrella, use velcro tape to secure the battery and double sided tape to secure the FLORA or Circuit Playground Express.



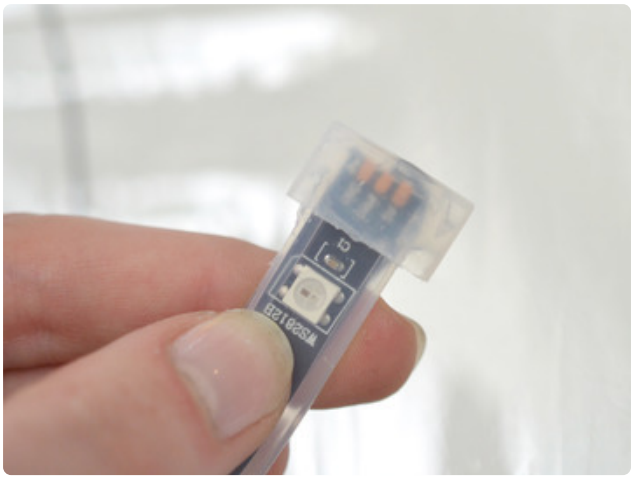
Test that your NeoPixels are all working! You should still have the custom strandtest sketch on your FLORA or the Circuit Playground Express MakeCode running from previous steps, but double check your number of pixels and pixel data pin in case some/all of your pixels aren't lighting up. Remember that a standard USB port can't power this many pixels, so you should be testing with USB disconnected and battery power on.

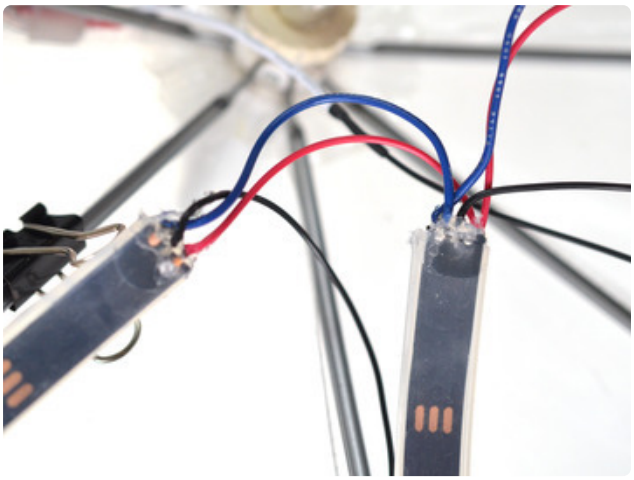


Everything's on? Great! Now onto tidy everything up.



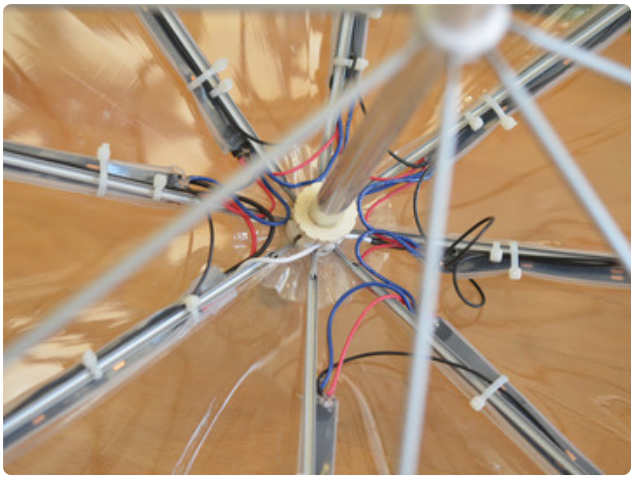
Finishing Touches

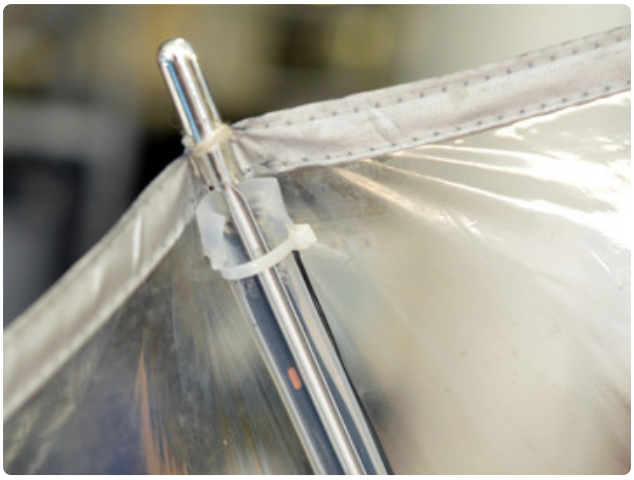




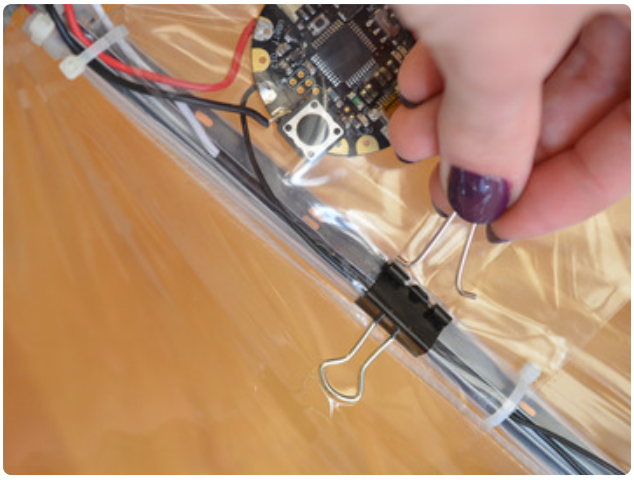
Now that you know your circuit works, it's time to seal up the ends of the NeoPixel strip. Not only does this help with weatherproofing your circuit, but it provides strain relief for the delicate soldered wires inside your morphing umbrella.

Use some Permatex 66B silicone adhesive to stick end caps on the strip ends and also wedge some inside the dome-side ends too. Wear gloves, have proper ventilation, and let dry for 24 hours.





After the glue is completely dry, use zipties to secure the strip ends inside the umbrella ribs.



Fold the extra vinyl hammock over the FLORA or Circuit Playground Express circuit and binder-clip in place. Squeeze the metal tabs to remove them, leaving the clip with a lower profile.

Use it!

Okay, time to have some fun! Make sure your battery is hooked up and turn the FLORA or Circuit Playground Express on. The first NeoPixel in the first strip will blink three times to warn you that the board is getting ready to read a color. Hold up the outside of the umbrella that has the color sensor right next to your fabric/item of choice. The color sensor will flash and then the color will be loaded into all the NeoPixels.

Hit the push button once on your FLORAbrella for a fun rainbow pattern. Hit the button again and you will have the rain pattern. You'll have to switch the power off and on again to go back to the color sensor mode.

Have fun and send us pics of your finished FLORAbrella!