Ever-Burning Flame Painting

Created by Erin St Blaine

https://learn.adafruit.com/ever-burning-flame-painting

Last updated on 2024-06-03 02:18:43 PM EDT

Table of Contents

Introduction 3

- Additional Materials
- Tips & Tricks

Wiring Diagram 5

Code 6

Assembly 10

- Power Supply

Calibration 17

- Flickering Adjustment

Introduction

Add gorgeous illumination to your favorite work of art with neopixels and Circuit Playground. This guide includes arduino code for a flame effect, and you can easily modify or add your own code to illuminate paintings of landscapes, rainbows, oceans, muscle cars, or whatever suits your style.

A small rosette serves as a capacitive touch on/off toggle switch. Touch it gently and the painting will flare up. Touch it again and the flames slowly fade to nothingness.

1 x Circuit Playground Classic Circuit Playground Classic	https://www.adafruit.com/product/3000
1 x 144/m Neopixels Neopixels -- Skinny or Regular	https://www.adafruit.com/product/2969
3 x Silicone Stranded Wire 26 awg stranded wire in at least 3 colors	https://www.adafruit.com/product/1881
1 x Screw Terminal Female screw terminal	https://www.adafruit.com/product/368
1 x Power Supply 5v 2a power supply	https://www.adafruit.com/product/276
1 x Diode 1N4001 Diode	https://www.adafruit.com/product/755

Additional Materials

- Art Canvas that's at least 1" thick
- Bristol board as wide as the canvas
- Soldering Iron & Accessories
- Clear packing tape
- Hot glue
- Solid, uncoated metal jewelry finding or button

Tips & Tricks

I used a canvas from a local craft store that's around 1 1/4 inch thick on the sides, and painted with regular acrylic paint. The thickness of the canvas creates space behind

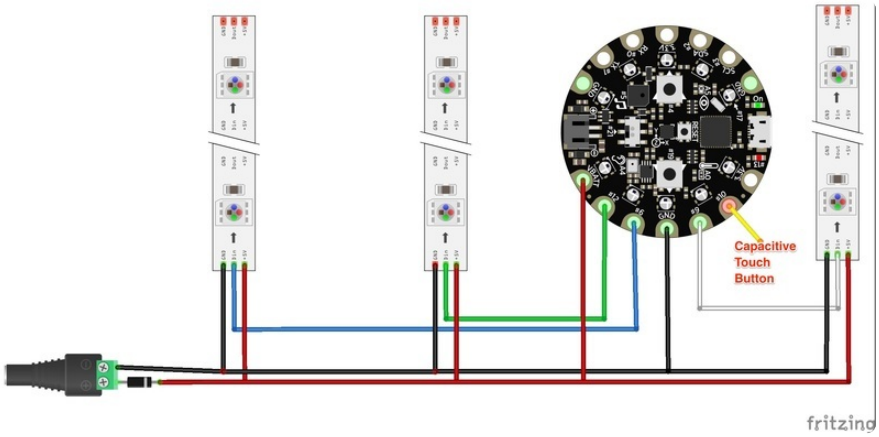
the painting that allows the light from the LEDs to diffuse beautifully. I'm using super high density 144/m neopixels because they give me a gorgeous, buttery smooth animation.

I created a painting that has both light and dark areas. The dark acrylic paint blocks the light and the lighter areas let the light through. You'll have best success if you match the paint colors to the neopixel colors -- they do get filtered through the paint color.

I used a small metal rosette with a sewable loop on the back for my capacitive touch on/off switch. I got this at a jewelry store, and it's just right for the purpose -- the sewable hole allows me to solder a wire firmly to the back of the button, and it's small enough and low-profile enough that it adds to the overall look of the artwork. You could also use artfully cut copper tape, a coin, or any other non-coated metal finding. Look in the scrapbooking section or the jewelry section at your local craft store if you don't have anything like this.

Also, remember that this painting will need to be plugged in. It's a good idea to think about how to get power to the painting's location without having to run ugly wires along your wall.

Wiring Diagram



Connections:

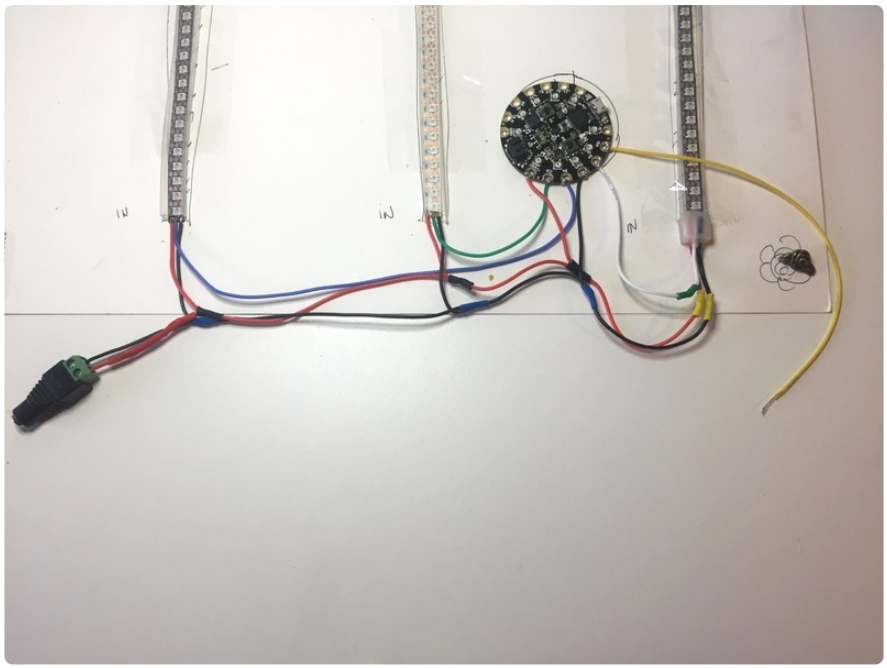
- Neopixel IN Strand 1 --> Circuit Playground 6
- Neopixel IN Strand 2 --> Circuit Playground 9
- Neopixel IN Strand 3 --> Circuit Playground 12
- Capacitive Touch Button --> Circuit Playground 10

Power will run from the power supply, through the screw terminal and the diode, then into the neopixel strips, and into the Circuit Playground's VBATT and GND pads.

We've added a diode near the power supply to protect our pixels from excessive voltage. A lot of inexpensive power supplies aren't super accurate: they're labeled as 5v and they give off 5v-ish, and it's sometimes closer to 5.5 or 6.

These neopixel strands can be pretty finicky, and they really don't like more than 5v. Adding a diode between the power supply and the LED strand will compensate by dropping the voltage just enough, so we get around 4.5-5.7 volts.

Alternatively, you can use a switching power supply (http://adafru.it/1448) with a 3 or 4.5 volt option, or one that's rated for 4.5 volts or lower. **Just be sure you don't switch the supply up to 6 or 12 volts,** or you could fry your controller and pixels!



Code

Before You Start

If this is your first foray into the world of arduino-based microcontrollers, you'll need to install some software first. Head over to the Circuit Playground Lesson 0 guide (https://adafru.it/tGC) for detailed installation and setup instructions.

You'll only need to do all this once, so be a little patient if it seems like a lot!

FastLED Library

You will also need to install the **FastLED** library in Arduino (`Sketch > Include Library > Manage Libraries...`)

One other note: if you're using **FastLED** with Circuit Playground, be sure to `#include` the Circuit Playground library FIRST and the **FastLED** library second, or you may run into problems.

Upload Code

Once you've got everything installed and your computer can talk to the Circuit Playground, it's time to upload the code.

Plug your Circuit Playground into your computer and select the Circuit Plaground under `Tools > Boards`. Then select the Circuit Playground as the Port.

Copy and paste this code into a new Arduino window and click "upload".

```
// SPDX-FileCopyrightText: 2018 Erin St Blaine for Adafruit Industries
//
// SPDX-License-Identifier: MIT

#include <Adafruit_CircuitPlayground.h>
#include <FastLED.h> // add FastLED library AFTER Circuit Playground library to
avoid issues

#define STRIP1_DATA_PIN 9  // define data pins for all 3 LED strips
#define STRIP2_DATA_PIN 12
#define STRIP3_DATA_PIN 6

#define COLOR_ORDER GRB

#define NUM_LEDS    80     // how many LEDs in each strip
#define NUM_LEDS_2   52
#define NUM_LEDS_3  69

#define CAP_THRESHOLD   50       //Change capacitive touch sensitivitiy here
#define FRAMES_PER_SECOND 35    // faster or slower burning fire

#define COOLING  55  // Less cooling = taller flames. Default 55, suggested range
20-100
#define SPARKING 50 //Higher chance = more roaring fire.  Default 120, suggested
range 50-200
#define BRIGHTNESS 125  // set global brightness here.  0-255
#define FADE 40  //How slowly the LEDs fade to off

CRGB leds[NUM_LEDS];       //separate LED arrays for all 3 strips
CRGB leds2[NUM_LEDS_2];
CRGB leds3[NUM_LEDS_3];

static byte heat[NUM_LEDS];    // separate heat arrays for all 3 strips
static byte heat2[NUM_LEDS_2];
static byte heat3[NUM_LEDS_3];

CRGBPalette16 currentPalette;
TBlendType    currentBlending;
CRGBPalette16 gPal;


//BUTTON SETUP STUFF
byte prevKeyState = HIGH;

//FIRST ACTIVE MODE
#define NUM_MODES 1     // actually 2 modes, mode 0 (off) and mode 1 (on)
int ledMode = 1;       // change to 0 to make the LEDs dark on startup

//READ CAP TOUCH BUTTON STATE
boolean capButton(uint8_t pad) {
  if (CircuitPlayground.readCap(pad) > CAP_THRESHOLD) {
    return true;
  } else {
    return false;
  }
}

//------------------------------------------------
void setup() {
  // Initialize serial.
```

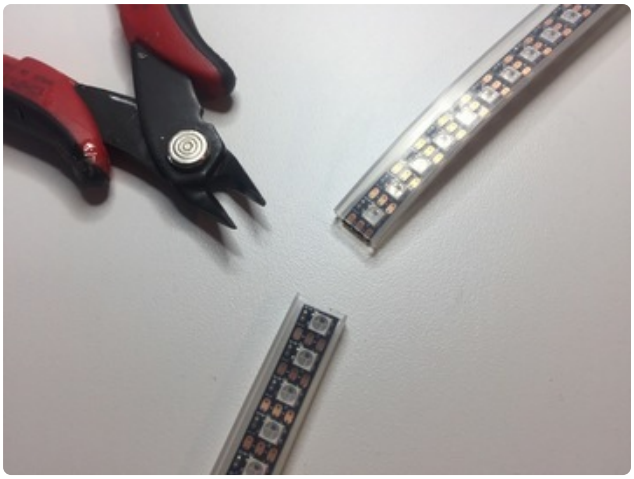
```
  Serial.begin(9600);

  // Initialize Circuit Playground library.
  CircuitPlayground.begin();

  // Add all 3 LED strips for FastLED library
  FastLED.addLeds<WS2812B, STRIP1_DATA_PIN, COLOR_ORDER>(leds,
NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.addLeds<WS2812B, STRIP2_DATA_PIN, COLOR_ORDER>(leds2,
NUM_LEDS_2).setCorrection( TypicalLEDStrip );
  FastLED.addLeds<WS2812B, STRIP3_DATA_PIN, COLOR_ORDER>(leds3,
NUM_LEDS_3).setCorrection( TypicalLEDStrip );

  //Set global brightness
  FastLED.setBrightness(BRIGHTNESS);
  currentBlending = LINEARBLEND;
  // Choose your color Palette
  gPal = HeatColors_p;
  //gpal = LavaColors_p;
  //gpal = RainbowColors_p;
  //gpal = CloudColors_p;
  //gpal = ForestColors_p;
  //gpal = PartyColors_p;
  //gpal = RainbowStripeColors_p;

}

//------------------------------------------------------
void loop() {
    switch (ledMode) {
       case 0: fire(); break;
       case 1: alloff(); break;
    }
      // READ THE BUTTON
        byte currKeyState = capButton(10);
        Serial.println (capButton(10));
        if ((prevKeyState == true) && (currKeyState == false)) {
            keyRelease();
        }

        prevKeyState = currKeyState;

}


//BUTTON CONTROL
void keyRelease() {
    Serial.println("short");

    ledMode++;
    if (ledMode > NUM_MODES){
    ledMode=0; }
}


 void fire()
{

  currentPalette = HeatColors_p;
  Fire2012WithPalette(); // run simulation frame, using palette colors
  Fire2012WithPalette2();
  Fire2012WithPalette3();
  FastLED.show(); // display this frame
  FastLED.delay(1000 / FRAMES_PER_SECOND);

}


void Fire2012WithPalette()
```

```
{
  random16_add_entropy( random());

    for( int i = 0; i < NUM_LEDS; i++) {
      heat[i] = qsub8( heat[i],  random8(0, ((COOLING * 10) / NUM_LEDS) + 2));
    }
    for( int k= NUM_LEDS - 3; k > 0; k--) {
      heat[k] = (heat[k - 1] + heat[k - 2] + heat[k - 2] ) / 3;
    }
    if( random8() < SPARKING ) {
      int y = random8(7);
      heat[y] = qadd8( heat[y], random8(160,255) );
    }
    for( int j = 0; j < NUM_LEDS; j++) {
      byte colorindex = scale8( heat[j], 240);
      leds[j] = ColorFromPalette( currentPalette, colorindex);

    }
}

void Fire2012WithPalette2()
{
  random16_add_entropy( random());
  static byte heat2[NUM_LEDS_2];
    for( int i = 0; i < NUM_LEDS_2; i++) {
      heat2[i] = qsub8( heat[i],  random8(0, ((COOLING * 10) / NUM_LEDS_2) + 2));
    }
    for( int k= NUM_LEDS_2 - 3; k > 0; k--) {
      heat2[k] = (heat2[k - 1] + heat2[k - 2] + heat2[k - 2] ) / 3;
    }
    if( random8() < SPARKING ) {
      int y = random8(7);
      heat2[y] = qadd8( heat2[y], random8(160,255) );
    }
    for( int j = 0; j < NUM_LEDS_2; j++) {
      byte colorindex = scale8( heat2[j], 240);
      leds2[j] = ColorFromPalette( currentPalette, colorindex);

    }
}
void Fire2012WithPalette3()
{
  random16_add_entropy( random());
    for( int i = 0; i < NUM_LEDS_3; i++) {
      heat3[i] = qsub8( heat3[i],  random8(0, ((COOLING * 10) / NUM_LEDS_3) + 2));
    }
    for( int k= NUM_LEDS_3 - 3; k > 0; k--) {
      heat3[k] = (heat3[k - 1] + heat3[k - 2] + heat3[k - 2] ) / 3;
    }
    if( random8() < SPARKING ) {
      int y = random8(7);
      heat3[y] = qadd8( heat3[y], random8(160,255) );
    }
    for( int j = 0; j < NUM_LEDS_3; j++) {
      byte colorindex = scale8( heat3[j], 240);
      leds3[j] = ColorFromPalette( currentPalette, colorindex);

    }
}


void alloff() {  // Fade all LEDs slowly to black
  for (int i = 0; i < NUM_LEDS; i++){
    leds[i].fadeToBlackBy( FADE );
    leds2[i].fadeToBlackBy( FADE );
    leds3[i].fadeToBlackBy( FADE );
  }
    for(int i = 0; i < NUM_LEDS; i++) {
  heat[i] = 0;
```

```
}
  for(int i = 0; i < NUM_LEDS; i++) {
  heat2[i] = 0;
}
  for(int i = 0; i < NUM_LEDS; i++) {
  heat3[i] = 0;
}

  FastLED.show();
  delay(20);
}
```

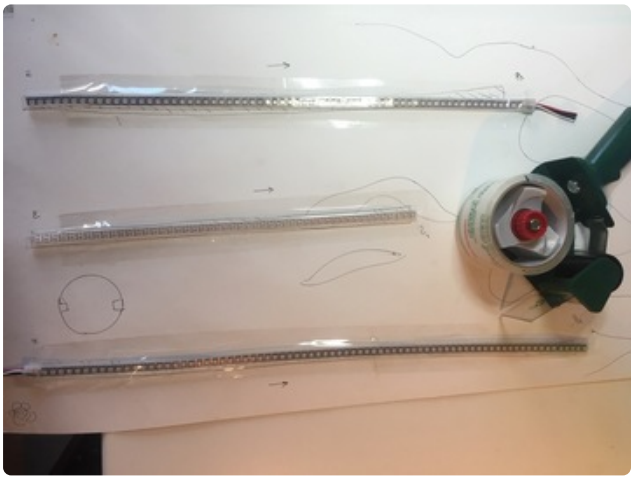
Assembly

Cut a piece of bristol board or thin cardboard so it's slightly smaller than your canvas. Sketch out where you plan to place your LEDs and your Circuit Playground. Mark the spot where your capacitive touch button will thread through the canvas.



Measure out the appropriate length of neopixel strip. Find the "in" end of the pixels (look for the arrow on the strip, it may be partially obscured). Carefully cut through the copper pads between LEDs. Cut close to the "out" side of the last neopixel in your strand, leaving more copper pad on the "in" edge of the next pixel in line.

These 144/m lights can be difficult to solder. The pads are really tiny and very close together. We don't need to solder anything on to the "out" ends of the strips for this project, so we can sacrifice that part of the pad and leave more copper pad on the "in" side of the pixel, making the soldering a bit easier.



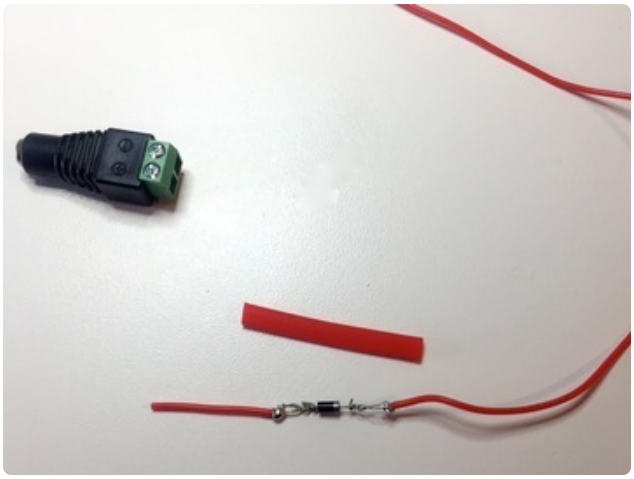
Lay out your strips on your bristol board, being sure the "in" ends are at the bottom. Secure the strips to the bristol board with clear packing tape, being sure the solder pads are accessible.

Power Supply

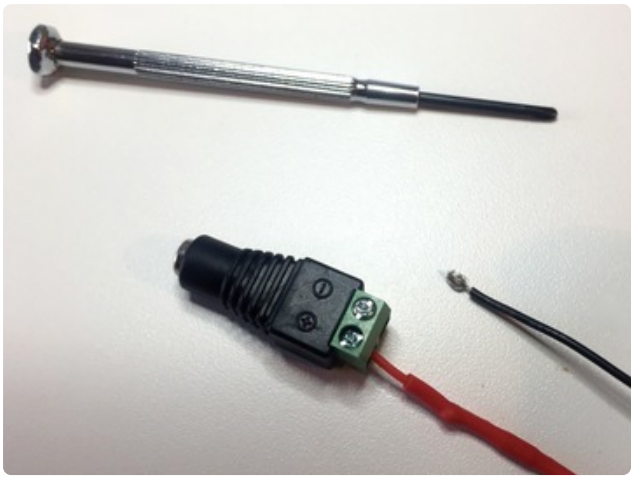
The 5v 2A power supply recommended for this project will supply just slightly too many volts for our neopixels -- they prefer around 3.5-4.5V for optimal performance. This is a nice, small, inexpensive power supply -- it's everything we want if we can just drop the voltage a smidge. An easy way to do this is to use a diode.

Diodes are meant to protect your circuit in the event of reverse polarity (like if you hook something up backwards). A side effect of adding this protection is that each diode you add will give about a 0.7V drop in your circuit. Perfect! That will bring our power supply to exactly the range we want.

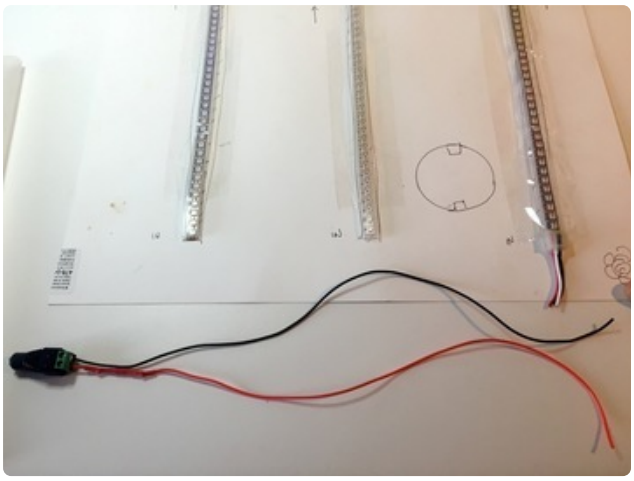
More about diodes and how they work here (https://adafru.it/Ck0)



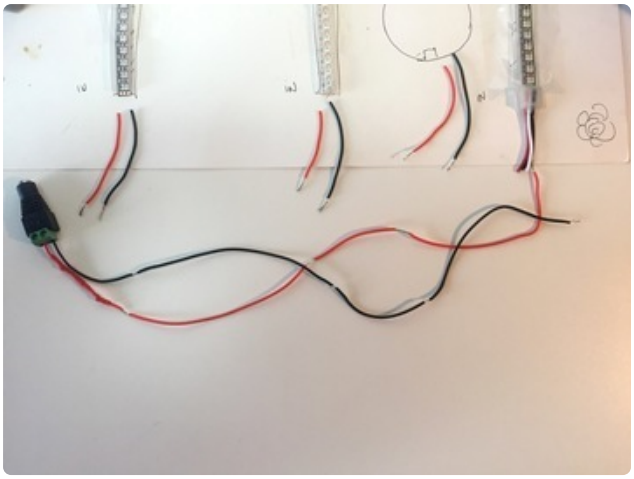
Twist the legs of your diodes into little loops. Look for the silver stripe on the diode. The stripe should be on the side facing AWAY from your power supply. Solder a very short wire to the non-striped end and a longer wire to the striped end. Cover with heat shrink.



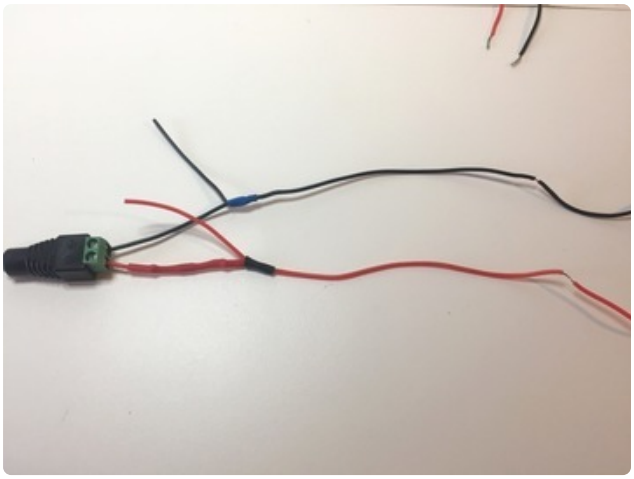
Place the short end of the red wire into the + pad on your screw terminal and screw it tight. Screw a long black wire into the - terminal. Tug on the wires to make sure they're really secure.

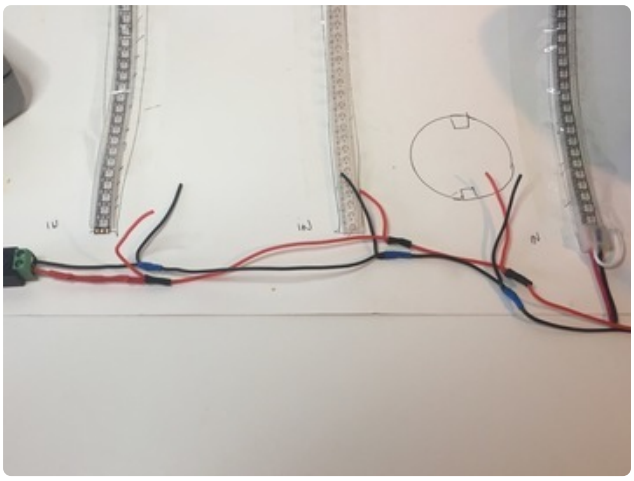


It's easiest to get the screw terminal to hold if you strip about 1/2" of shielding from the wire and then twist the bare wires into a little ball, which will fit snugly inside the terminal.

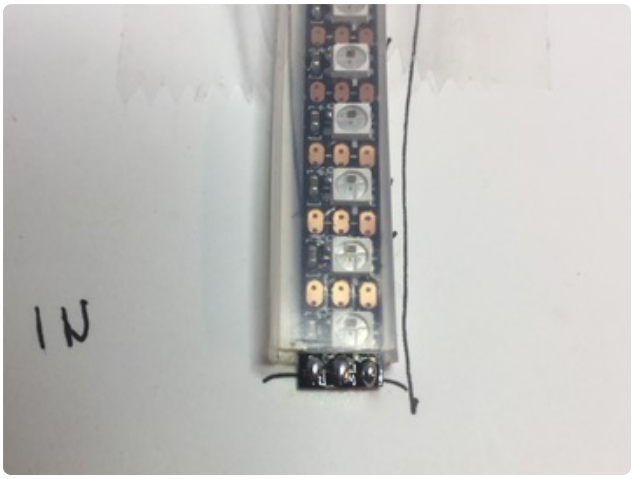


One of my favorite things about this stranded silicone wire is that it's easy to strip a little of the silicone shielding to expose a section of wire without cutting the wire. Just grab where you'd like to solder on a wire, then with a tight pinch and pull with your fingernails, expose a tiny bit of wire.

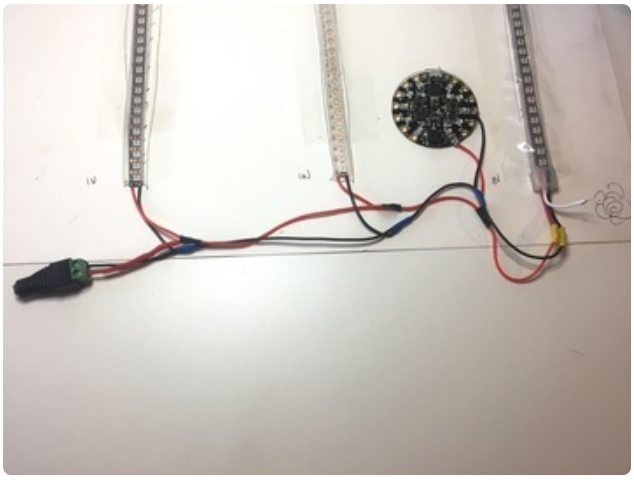




Cut several short red and black wires, long enough to reach your neopixel strips and your circuit playground. Splice the short wires to your long power wires coming from the screw terminal.

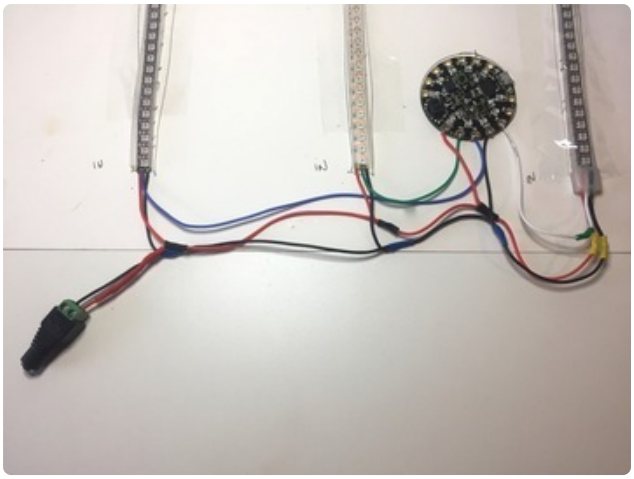


Tin the pads on your neopixel strips, checking again that you have the "in" ends of the strips down near the bottom.

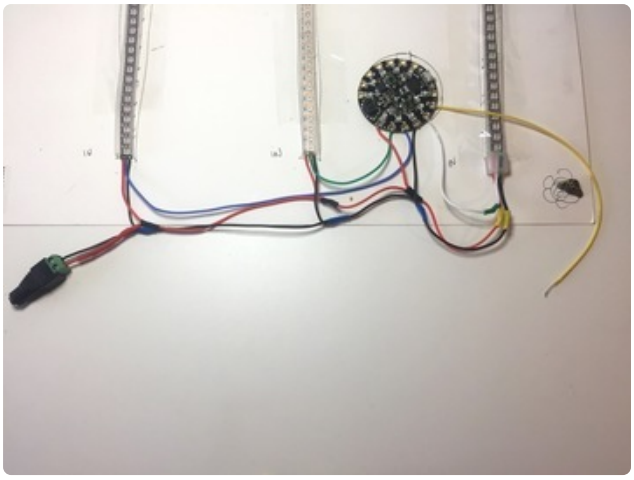


Solder a red power wire to VBAT on your Circuit Playground and a black wire to G.

Solder the short power wires to the neopixel strips: red wires to + and black wires to - or G. Be sure you get the right pads -- the labeling can be hard to read on these tiny LEDs.



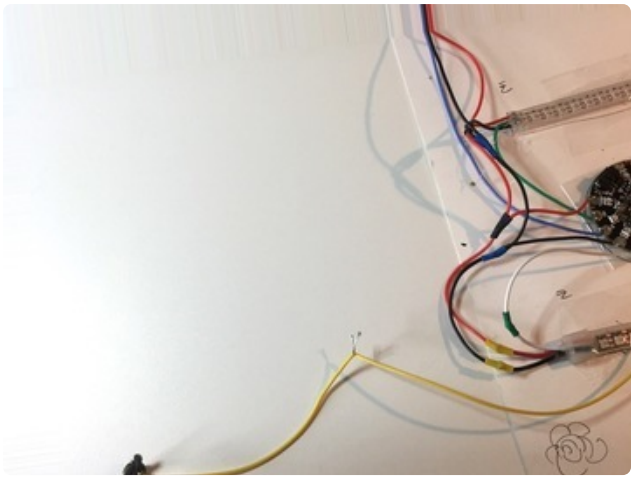
Solder a colored wire from pads 6, 9, and 12 to each "in" data pad on your neopixel strips.



Finally, add a longish colored wire to pin 10. This will go to your capacitive touch on/off switch.

Time for testing! Once all your connections are made, plug in the power supply and watch the flames flicker! Touch the bare end of the wire coming from your capacitive touch pad, and be sure the lights fade out and back on again.

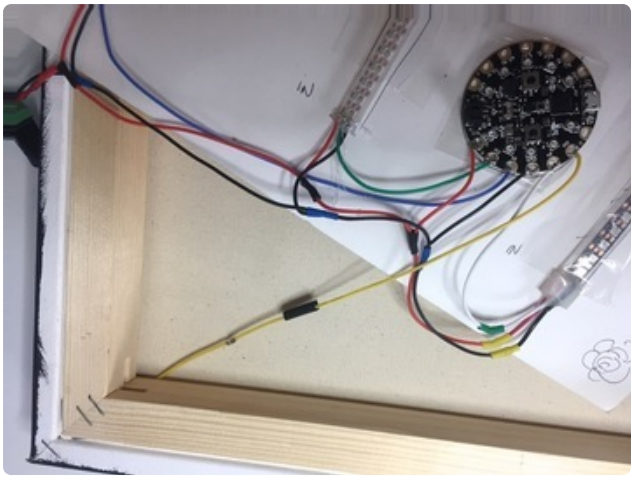
Solder a short wire firmly to the back of your button. Twist the wire together with your capacitive touch button wire temporarily and test the button to be sure it works.



Some metal jewelry findings have a coating on them that will make capacitive touch tricky. If yours doesn't work at first, try soaking it in 99% alcohol for a few minutes. This will clean off any oil or residue and make the connection work better.

If your button doesn't work at this point, head over to the calibration page and try calibrating the code to the button.

If it still doesn't work, you may need to find a different thing to use as a button. Try copper tape or anything made from a clean, solid, conductive metal.



Once your button is working reliably, mount it on the front of your painting with the wire poking carefully through the canvas. Solder the wire to the capacitive touch wire coming from the Circuit Playground.

Attach the bristol to the back of your canvas with staples, leaving the power cord accessible. You can also cut a small hole in the bristol board so you can access the Circuit Playground's USB port, in case you want to change the code later on.

Calibration

After everything is assembled, you may find that your capacitive touch button is too sensitive, or not sensitive enough. Luckily this is easy to adjust in the code.

Plug your Circuit Playground into your computer and be sure it's selected under Tools > Port > Adafruit Circuit Playground.

Open the Serial monitor and move your hands away from the buton. You should see numbers scrolling by. This is your baseline number, so write it down.

Now touch the button. This is your active number. Write this down too.

Go find this line in the code:

```
#define CAP_THRESHOLD   50       //Change capacitive touch sensitivitiy here
```

The `CAP_THRESHOLD` variable is what sets the sensitivity. Change the number to something near the median point between your baseline number and your active number. Upload the code again and test to see if it's behaving better.

Flickering Adjustment

I like a really dynamic fire, where the flames burn tall but have a lot of motion and action to them. You can adjust a couple of variables so the flames react the way you want.

Look for these variables:

```
#define FRAMES_PER_SECOND 35    // faster or slower burning fire

#define COOLING  55  // Less cooling = taller flames. Default 55, suggested range
20-100
#define SPARKING 50 //Higher chance = more roaring fire.  Default 120, suggested
range 50-200
#define BRIGHTNESS 125  // set global brightness here.  0-255
#define FADE 40  //How slowly the LEDs fade to off
```

`FRAMES_PER_SECOND` will make the fire appear to burn faster or slower.

`COOLING` and `SPARKING` will give you the most control over the height and flicker of the flames.

`BRIGHTNESS` will set the overall brightness of your project. Be sure to test it in daylight and in darkness to find the right level for your room.

`FADE` will control how quickly or slowly the LEDs will fade to darkness when you touch the button.