# Table of Contents

**Introduction**  
- Base Material

**Code**  
- Wiring Diagram

**3d Printing**  
- Slice Settings

**Circuit Layout**  
- Layout
- Mark the Pixels
- Find Wire Lengths

**Neopixel Strands**  
- Testing

**Assembly**  
- Finishing
Introduction

Finally, a skirt that does everything.

This skirt uses the amazing Circuit Playground board and stranded neopixels to create a versatile garment with loads of features.

Sound / music reactive mode
Motion sensing twinkle mode
Rainbow color palette modes

I made a petticoat or under-slip that can be worn underneath a variety of long skirts or dresses. It would be simple to convert this project to pants, chaps, a vest, or any layered costume look.

The 3d printed "belt buckle" houses the Circuit Playground and adds another 10 reactive lights and button controls to the ensemble. It can work as a standalone light buckle without the skirt as well.

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

I wanted to cover my whole skirt with just a few pixels, so I took some extra time and wired the pixels up individually. This way I could leave 10-12 inches between pixels and get a more spread-out and random twinkle effect.

If you don't like soldering much, you can also use our pre-soldered LED dot strands. They come in 2" or 4" spacing, so your pixel density will be a bit higher, but they're much easier to build and can be more reliable than hand-soldered pixels.
This guide uses 3 colors of silicone stranded wire to connect the pixels. You can instead use our silicone stranded ribbon cable. It's a little trickier to keep track of your wires since they're all black, but you end up with a more streamlined project with fewer wires to pull and catch on.

3 x **Silicone Stranded Wire**
3 Colors of Silicone Stranded Wire  
https://www.adafruit.com/product/2513

1 x **Ribbon Cable Wire**
Silicone Stranded Ribbon Cable Wire  
https://www.adafruit.com/product/3889

---

### Flora RGB Smart NeoPixel version 2 - 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

### Adafruit NeoPixel LED Dots Strand - 20 LED 4" Pitch
Attaching NeoPixel strips to your costume can be a struggle as the flexible PCBs can crack when bent too much. So how to add little dots of color? Use these stranded NeoPixel dots!

https://www.adafruit.com/product/3631

### Adafruit NeoPixel LED Dots Strand - 20 LEDs at 2" Pitch
Attaching NeoPixel strips to your costume can be a struggle as the flexible PCBs can crack when bent too much. So how to add little dots of color? Use these stranded NeoPixel dots!

https://www.adafruit.com/product/3630
For a lighter weight power source that lasts a little longer, you can use a Lithium polymer battery. These are great for adult use but can catch fire if they're damaged or if they get wet, so if you're making this for a kid, stick with the AAA battery pack.

<table>
<thead>
<tr>
<th>Item</th>
<th>Quantity</th>
<th>Description</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA Batteries</td>
<td>1 x</td>
<td></td>
<td><a href="https://www.adafruit.com/product/727">https://www.adafruit.com/product/727</a></td>
</tr>
<tr>
<td>3 x AAA Battery Holder</td>
<td></td>
<td>with On/Off Switch and 2-Pin JST</td>
<td></td>
</tr>
<tr>
<td>Lithium Polymer Battery</td>
<td>1 x</td>
<td></td>
<td><a href="https://www.adafruit.com/product/258">https://www.adafruit.com/product/258</a></td>
</tr>
<tr>
<td>Battery Charger</td>
<td>1 x</td>
<td></td>
<td><a href="https://www.adafruit.com/product/1304">https://www.adafruit.com/product/1304</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Battery Charger for Lithium Polymer Battery</td>
<td></td>
</tr>
</tbody>
</table>

**Base Material**

- [Long, loose slip](#), or
- 2-3 Yards of white [Tricot Fabric](#) (to make your own)
- Fabric scraps to sew a battery pocket
- 3d printed case with buttons - print your own or [order from Shapeways](#)
**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 [Arduino page of the Circuit Playground Express guide](#) 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 Adafruit Circuit Playground Express under **Tools > Boards**. Then select the Circuit Playground Express as the Port.

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

```cpp
#include <Adafruit_Circuit Playground.h>
#include <FastLED.h>

#define LED_PIN A1  // led strand is soldered to pin 6
#define CP_PIN 8   // circuit playground's neopixels live on pin 8
#define NUM_LEDS 11 // number of LEDs in my strand
#define NUM_CP 10   // number of neopixels on the circuit playground
#define COLOR_ORDER GRB

uint8_t brightness = 255; // led strand brightness control
uint8_t cpbrightness = 40; // circuit playground brightness control

int STEPS = 25; // makes the rainbow colors more or less spread out
int NUM_MODES = 5; // change this number if you add or subtract modes

CRGB leds[NUM_LEDS]; // I've set up different arrays for the neopixel strand and the circuit playground
```

©Adafruit Industries
CRGB cp[NUM_CP];    // so that we can control the brightness separately

CRGBPalette16 currentPalette;
TBlendType currentBlending;

int ledMode = 0;       //Initial mode
bool leftButtonPressed;
bool rightButtonPressed;

// SOUND REACTIVE SETUP --------------
#define MIC_PIN         A4                                    // Analog port for microphone
#define DC_OFFSET  0                                          // DC offset in mic signal - if unsure, leave 0
#define NOISE     200                                         // Noise/hum/ interference in mic signal and increased value until it went quiet
#define SAMPLES   60                                          // Length of buffer for dynamic level adjustment
#define TOP (NUM_LEDS + 2)                                    // Allow dot to go slightly off scale
#define PEAK_FALL 10                                          // Rate of peak falling dot
 byte
  peak      = 0,                                              // Used for falling dot
 dot
  dotCount  = 0,                                              // Frame counter for delaying dot-falling speed
 volCount  = 0;                                              // Frame counter for storing past volume data
 int
  vol[SAMPLES],                                              // Collection of prior volume samples
 lvl       = 10,                                             // Current audio level, change this number to adjust sensitivity
 minLvlAvg = 0,                                              // For dynamic adjustment of graph low & high
 maxLvlAvg = 512;

// MOTION CONTROL SETUP----------
#define MOVE_THRESHOLD 10 // movement sensitivity. lower number = less twinklitude

float X, Y, Z;
// Here is where you can put in your favorite colors that will appear!
// just add new {nnn, nnn, nnn}, lines. They will be picked out randomly
//
// R   G   B
//
// Change colors by inputting different R, G, B values on these lines
//
// don't edit the line below
#define FAVCOLORS sizeof(myFavoriteColors) / 3

void setup() {
    Serial.begin(57600);
    CircuitPlayground.begin();
    FastLED.addLeds<WS2812B, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection( TypicalLEDStrip );
    FastLED.addLeds<WS2812B, CP_PIN, COLOR_ORDER>(cp,
```cpp
10).setCorrection( TypicalLEDStrip );
currentBlending = LINEARBLEND;
set_max_power_in_volts_and_milliamps(5, 500);               // FastLED 2.1 Power
management set at 5V, 500mA
}

void loop() {

  leftButtonPressed = CircuitPlayground.leftButton();
  rightButtonPressed = CircuitPlayground.rightButton();

  if (leftButtonPressed) { // left button cycles through modes
    clearpixels();
    ledMode = ledMode + 1;
    delay(300);
    if (ledMode > NUM_MODES){
      ledMode = 0;
    }
  }
  if (rightButtonPressed) { // right button turns all leds off
    ledMode = 99;
  }

  switch (ledMode) {
    case 0: currentPalette = RainbowColors_p; rainbow(); break;
    case 1: motion(); break;
    case 2: soundreactive(); break;
    case 3: currentPalette = OceanColors_p; rainbow(); break;
    case 4: currentPalette = LavaColors_p; rainbow(); break;
    case 5: currentPalette = RainbowStripeColors_p; rainbow(); break;
    case 99: clearpixels(); break;
  }
}

void clearpixels()
{
  CircuitPlayground.clearPixels();
  for (int i = 0; i < NUM_LEDS; i++) leds[i] = CRGB::Black;
  for (int i = 0; i < NUM_CP; i++) cp[i] = CRGB::Black;
  FastLED.show();
}

void rainbow()
{
  static uint8_t startIndex = 0;
  startIndex = startIndex + 1; /* motion speed */
  FillLEDsFromPaletteColors( startIndex);
  FastLED.show();
  FastLED.delay(10);
}

// this bit is in every palette mode, needs to be in there just once
void FillLEDsFromPaletteColors( uint8_t colorIndex)
{
  for (int i = 0; i < NUM_LEDS; i++) leds[i] = ColorFromPalette( currentPalette,
    colorIndex, brightness, currentBlending);
  for (int i = 0; i < NUM_CP; i++) cp[i] = ColorFromPalette( currentPalette,
    colorIndex, cpbrightness, currentBlending);
  colorIndex += STEPS;
}
```
```c
void soundreactive() {
  uint8_t  i;
  uint16_t minLvl, maxLvl;
  int      n, height;

  n = analogRead(MIC_PIN);                                    // Raw reading from mic
  n = abs(n - 512 - DC_OFFSET);                               // Center on zero
  n = (n <= NOISE) ? 0 : (n - NOISE);                         // Remove noise/hum
  lvl = ((lvl * 7) + n) >> 3;                                 // "Dampened" reading
  if (height < 0L)       height = 0;                          // Clip output
  else if (height > TOP) height = TOP;                        // Keep 'peak' dot at top

  // Calculate bar height based on dynamic min/max levels (fixed point):
  height = TOP * (lvl - minLvlAvg) / (long)(maxLvlAvg - minLvlAvg);
  if (height < 0L)       height = 0;                          // Clip output
  else if (height > TOP) height = TOP;                        // Keep 'peak' dot at top

  // Color pixels based on rainbow gradient -- led strand
  for (i=0; i<NUM_LEDS; i++) {
    if (i >= height)   leds[i].setRGB( 0, 0,0);
    else leds[i] = CHSV(map(i,0,NUM_LEDS-1,0,255), 255, brightness);  //constrain colors here by changing HSV values
  }

  // Draw peak dot -- led strand
  if (peak > 0 && peak <= NUM_LEDS-1) leds[peak] = CHSV(map(peak, 0,NUM_LEDS-1,0,255), 255, brightness);

  // Color pixels based on rainbow gradient -- circuit playground
  for (i=0; i<NUM_CP; i++) {
    if (i >= height)   cp[i].setRGB( 0, 0,0);
    else cp[i] = CHSV(map(i,0,NUM_CP-1,0,255), 255, cpbrightness);  //constrain colors here by changing HSV values
  }

  // Draw peak dot -- circuit playground
  if (peak > 0 && peak <= NUM_CP-1) cp[peak] = CHSV(map(peak, 0,NUM_LEDS-1,0,255), 255, cpbrightness);

  // Every few frames, make the peak pixel drop by 1:
  if (++dotCount >= PEAK_FALL) {                            // fall rate
    if(peak > 0) peak--;
    dotCount = 0;
  }

  vol[volCount] = n;                                          // Save sample for dynamic leveling
  if (++volCount >= SAMPLES) volCount = 0;                    // Advance/rollover sample counter

  // Get volume range of prior frames
  minLvl = maxLvl = vol[0];
  for (i=1; i<SAMPLES; i++) {
    if (vol[i] < minLvl)     minLvl = vol[i];
    else if (vol[i] > maxLvl) maxLvl = vol[i];
  }
  // minLvl and maxLvl indicate the volume range over prior frames, used for vertically scaling the output graph (so it looks interesting regardless of volume level). If they're too close together though (e.g. at very low volume levels) the graph becomes super coarse and 'jumpy'...so keep some minimum distance between them (this also lets the graph go to zero when no sound is playing):
```
if((maxLvl - minLvl) < TOP) maxLvl = minLvl + TOP;
minLvlAvg = (minLvlAvg * 63 + minLvl) >> 6;  // Dampen min/max levels
maxLvlAvg = (maxLvlAvg * 63 + maxLvl) >> 6;  // (fake rolling average)

show_at_max_brightness_for_power();  // Power managed
FastLED.display
Serial.println(LEDs.getFPS());
} // fastbracelet()

void motion() {
  X = CircuitPlayground.motionX();
  Y = CircuitPlayground.motionY();
  Z = CircuitPlayground.motionZ();

  // Get the magnitude (length) of the 3 axis vector
double storedVector = X*X;
  storedVector += Y*Y;
  storedVector += Z*Z;
  storedVector = sqrt(storedVector);
  Serial.print("Len: "); Serial.println(storedVector);

  // wait a bit
delay(100);

  // get new data!
  X = CircuitPlayground.motionX();
  Y = CircuitPlayground.motionY();
  Z = CircuitPlayground.motionZ();
  double newVector = X*X;
  newVector += Y*Y;
  newVector += Z*Z;
  newVector = sqrt(newVector);
  Serial.print("New Len: "); Serial.println(newVector);

  // are we moving
  if (abs(10*newVector - 10*storedVector) > MOVE_THRESHOLD) {
    Serial.println("Twinkle!");
    flashRandom(5, 1);  // first number is 'wait' delay, shorter num == shorter
    twinkle
    flashRandom(5, 3);  // second number is how many neopixels to simultaneously
    light up
    flashRandom(5, 2);
  }
}

void flashRandom(int wait, uint8_t howmany) {
  for(uint16_t i=0; i<howmany; i++) {
    // pick a random favorite color!
    int c = random(FAVCOLORS);
    int red = myFavoriteColors[c][0];
    int green = myFavoriteColors[c][1];
    int blue = myFavoriteColors[c][2];

    // get a random pixel from the list
    int j = random(NUM_LEDS);
    //Serial.print("Lighting up "); Serial.println(j);

    // now we will 'fade' it in 5 steps
    for (int x=0; x < 5; x++) {
      int r = red * (x+1); r /= 5;
      int g = green * (x+1); g /= 5;
      int b = blue * (x+1); b /= 5;
    }
  }
}
leds[j].r = r;
leds[j].g = g;
leds[j].b = b;
FastLED.show();
CircuitPlayground.setPixelColor(j, r, g, b);
delay(wait);
} // & fade out in 5 steps
for (int x=5; x >= 0; x--) {
    int r = red * x; r /= 5;
    int g = green * x; g /= 5;
    int b = blue * x; b /= 5;

    leds[j].r = r;
    leds[j].g = g;
    leds[j].b = b;
    FastLED.show();
    CircuitPlayground.setPixelColor(j, r, g, b);
    delay(wait);
}
// LEDs will be off when done (they are faded to 0)

If all goes well, the lights on the Circuit Playground will come on. Press the right side button and the lights will go off. Press the left side button to toggle between modes:

1. Rainbow Colors
2. Motion Reactive
3. Sound Reactive
4. Ocean Colors
5. Lava Colors
6. Rainbow Stripe Colors

Press the button once to enter Motion Reactive mode. Jiggle the Circuit Playground around to watch the lights dance.

Press the button again to enter Sound Reactive mode. Make some noise!

Add your own modes or customize the modes that are already there.
This diagram shows how the wiring will lay out in your project. Don't solder everything together yet! You'll want to put the case in place and feed the wires through before soldering.

The wiring for this project is simple and robust.

- Circuit Playground Pin 6 --> Neopixel IN
- Circuit Playground VBATT --> Neopixel +
- Circuit Playground G --> Neopixel G

Then, string the neopixels in strands with OUT connecting to IN of the next pixel. Connect all the + pads together, and all the - pads together.

Finally, connect the last + and - in each neopixel strand back to the very first neopixel in that strand. This will distribute your power evenly and will make your project fail gracefully: if any one black or red wire breaks along the strand, all the neopixels will still light up, since they're getting power from both ends.
3d Printing

This is a remix of the 3d Printed Circuit Playground Yo Yo project by the Ruiz Brothers.

There are 5 different pieces to print: the base, the top, the buttons, the cover, and the belt clip.

If you don't have access to a 3d printer, you can order the whole case ready-to-go from Shapeways.

Download from Thingiverse

Order from Shapeways

Slice Settings

Below are some recommended slice settings. These parts were printed on an Up! Mini.

- .15mm layer height
- 220C extruder / PrintInZ Skin
- 20% infill
- 4 top/bottom layers
- 2 shells / parameters
- 40mm/s printing speed
Circuit Layout

One of the Holy Grails of LED costume design is the quest for the Stretchy Circuit.

Clothing needs to move and stretch with the body. Wires and electronics do not stretch, as a rule.

For this project, we're going to use LOTS OF WIRE. We'll zig zag the wire so we can stretch the fabric without breaking it. Since we're making an under-layer, we can get away with using an abundance of wire. The main costume layer should hide the wire nicely while just letting the light shine through.

Denier Tricot is a wonderful fabric for slips and under-clothing. It's made for lingerie production so it feels nice next to the skin. It has a 2 way stretch, meaning it'll hug your body and stretch left-to-right, but won't stretch in length, even if you sew lots of heavy wires and batteries to it.

I made my own since I had some Tricot on-hand; you can do the same or order one pre-made from Amazon or your favorite lingerie supplier. (It might be fun to use a tulle petticoat or hoop skirt as well!)

Layout

Now it's time to figure out how many lights you want to add and where they will be placed.

My skirt will have two LED strands with 10 neopixels apiece, mirrored on the left and right sides. Since I'm mirroring, I only need to map one strand, then I can copy it on the other side.
NOTE: I finished my whole project using this diagram, then later realized I may want to wear this under a dress as well as under a skirt. If I were to do it again, I'd add an extra 12-18 inches of wire between the Circuit Playground and the first neopixel in the strand. That way I could wear the Circuit Playground module as a belt buckle or as a brooch, clipped to my neckline.

Mark the Pixels

Lay the skirt out on the floor and mark where you'd like each LED to be. Draw a line connecting all the pixels, with arrows showing the flow of data. Connect + and – from the first pixel to the last pixel, making a rough circle (do not connect the data wire back to the first, only the power wires).

Find Wire Lengths

Measure the distance between each planned neopixel spot. Multiply each measurement by 1.5 to allow for stretch and movement in the wires. Note down all the wire lengths. Number each measurement, so you don't lose your place later on.
Neopixel Strands

First Pixel

1. Cut a red, black, and white wire to the length of your first measurement. One end of these wires will connect to your Circuit Playground, and the other will run to your first neopixel.
2. Cut a red and black wire the length of your last measurement. This will complete your circle connecting the first pixel to the last pixel in your strand.
3. Solder the white wire to the "in" pad on your neopixel. (Don't solder the red and black yet)
4. Cut the red, black and white wires for your second measurement.
5. Solder this second white wire to the "out" pad on your neopixel.
6. Connect all three of the red wires together into the + pad on your first neopixel.
7. Connect all three of the black wires to the - pad on your neopixel.
Additional Pixels

1. Solder the white wire coming from the first neopixel to the "in" pad on your neopixel.
2. Cut 3 wires the length of measurement #3
3. Solder the white wire to the "out" pad on your neopixel.
4. Twist your newly cut red wire together with the last wire in the strand. Solder them together into the + pad on your neopixel.
5. Repeat with the black wire.

This is tricky to explain but simple once you get the concept. It was easiest for me to cut one length of 3 wires at a time, to avoid giant confusing piles of wire.
Testing

As you go along, you'll want to test and be sure the pixels are working. This is easy to do with the Circuit Playground. Strip a little wire from the top end of your strands and temporarily connect them to the Circuit Playground. You can use alligator clips or just thread the wire through the hole and wrap it around itself a few times. Don't solder yet -- we need to get the 3d printed case in place first -- but connecting the wires temporarily right now will allow you to be sure all your soldering connections are good and your neopixels are working.
Attach Strands

Lay out your neopixel strand on your skirt, placing each pixel on the markings you made earlier. Glue the pixels in place using fabric glue, and let dry.

Pin the wires to the fabric in a zig zag between the pixels.

Hand stitch or machine stitch (using a zig zag tack) the wires wherever they stick up. A looping wire will catch on things and break, so wrestle them all neatly into place.
Unwind the wires from the circuit playground and feed them through the hole on the lower right side of the 3d printed case back. Solder them in place.

Feed the female end of your battery extension cable switch into the hole on the lower left. Plug this cable into the JST connector on the Circuit Playground.

Nestle the circuit playground inside the case, with all wires feeding neatly underneath, and press down so the USB port lines up with its hole and the posts slot.
neatly into the Circuit Playground's holes. Be sure the wires are feeding neatly and then secure the Circuit Playground to the posts with a dab of hot glue.

Place the lid inside the top and tape it in place. Screw the top onto the base and then adjust the lid until the holes line up perfectly with the buttons. Carefully remove the lid and top, and glue them together.

Turn the whole assembly upside down and place the buttons into the holes in the lid, face-down. Screw or press the base into place. Make sure the buttons work, then glue the lid down.
Using a strong acrylic glue, affix the belt clip to the back of the base with the USB port facing down.

**Finishing**

Find a convenient yet out-of-the-way spot for the switch and secure it to the skirt with glue and thread tacks.

Use some scrap fabric to machine or hand stitch a pocket for your battery.

Try it on under various skirts! I like to wear it with the Circuit Playground clipped onto the outer skirt like a belt buckle, with the lights and wires twinkling beneath.
Another idea: If you plan to wear this with full dresses in addition to skirts, you could add an extra 12-18 inches of wire between the circuit playground case and the top of the skirt. Then you could wear the Circuit Playground as a brooch along your neckline instead of a weirdly glowing circle under the dress at your waist. It's also possible to control the brightness of the Circuit Playground independently of the LED strands in the code, so play with brightness levels until you find a good balance.