Wonder Woman Cosplay Bracers

Created by Erin St Blaine

https://learn.adafruit.com/wonder-woman-cosplay-bracers

Last updated on 2024-06-03 02:06:29 PM EDT

Table of Contents

Introduction

> Stop a bullet cold,
> Make the Axis fold,
> Change their minds,
> and change the world.
>
> Wonder Woman, Wonder Woman.
> Now the world is ready for you,
> and the wonders you can do.

Build your own motion and sound-reactive bracers to block every blow that comes your way. Neopixels shine and sparkle through side-glow acrylic. The deliciously easy-to-use sensors aboard the Circuit Playground create the magic of interactivity.

Electronics

2 x Circuit Playground Development Board	https://www.adafruit.com/product/3000
1 x Mini Skinny Neopixels 60/m	https://www.adafruit.com/product/2964
1 x Neopixel Pack of 5 Individual Pixels	https://www.adafruit.com/product/1612
2 x Lithium Ion Polymer Battery 3.7v and 500mAh	https://www.adafruit.com/product/1578
2 x JST Extension Cable with Switch On/Off switch and battery cable	https://www.adafruit.com/product/3064
3 x 26 awg Silicone Stranded Wire Get 3-4 different colors	https://www.adafruit.com/product/1881

Other Materials

There are a thousand and one ways to build a bracer, so go nuts with your own build idea -- the only thing that's non-negotiable for this design is the acrylic.

- Approx. 2mm thick acrylic (I found mine in the $1 bin at Tap Plastics)

- Black Gaffer's tape or Duct tape
- EVA craft foam (from the craft store)
- Black Worbla (https://adafru.it/wA1)
- Aluminum foil tape (https://adafru.it/wA2)
- Gold acrylic paint
- Snaps
- Spray Plasti-dip (https://adafru.it/wA3)
- Acrylic glue

Tools

I kept the build simple and didn't use anything fancy like 3d printers or laser cutters or vacuum chambers, though all those tools would have contributed their own charm to this project.

- Heat gun
- Rotary cutting & polishing tool
- Soldering iron & accessories
- Oven & baking sheet
- Aluminum foil
- Sharp Scissors

Design & Planning

Wonder Woman is an iconic figure who's been drawn and portrayed so many different ways throughout the last 7 decades, which gives us so much material to draw from. I

was definitely inspired by the lightning-enabled bracers from the 2017 Wonder Woman Movie Origin trailer (https://adafru.it/wA4). Booyah.

I love doing cosplay projects largely because of the melding of tech and magic in the finished piece. So even though it strays a bit from the Hollywood version, I wanted to display the Circuit Playground proudly on the outside of the bracers. The added bonus here is that I can now use all the neopixels on the face of the Circuit Playground as part of the design.

Considerations & Features

- Must appear to have crazy orange lightning inside
- Flashy bullet-blocking glitter light and sound effects
- Gold and silver finish, or it's just not Wonder Woman
- Interactive - I want to actually block imaginary bullets

Interactivity

I decided to use two different interactive sensors: sound, because a loud CRACK means someone fired on me, and motion, so the bracers notice when I make a ninja-worthy blocking move.

I considered adding an impact or vibration sensor, but the design relies on acrylic and acrylic is notoriously NOT impact-resistant. In fact, it likes to shatter with too much of an impact, so let's just remove the temptation.

Melee Mode

Cosplayers spend a lot of time standing around waiting for admirers to figure out their camera phones. For photo shoots and dance parties, I decided these bracers also need a "melee" mode, where they're constantly on. This way they look great in photos without continuous blocking or having to keep shouting "BANG".

Comfort & Ease of Use

I need to be able to get these on and off all by myself with one hand. I also need to be able to quickly remove and switch out the battery without disassembling the whole bracer.

Craft foam is a great choice for the layer that goes against my skin. It's pliable and pretty tough if it's sealed, plus it's cheap and easy to come by. And I can get in and out of snaps one-handed, so I definitely prefer that to a button or tie-on method, even though it may not be as "authentic" looking.

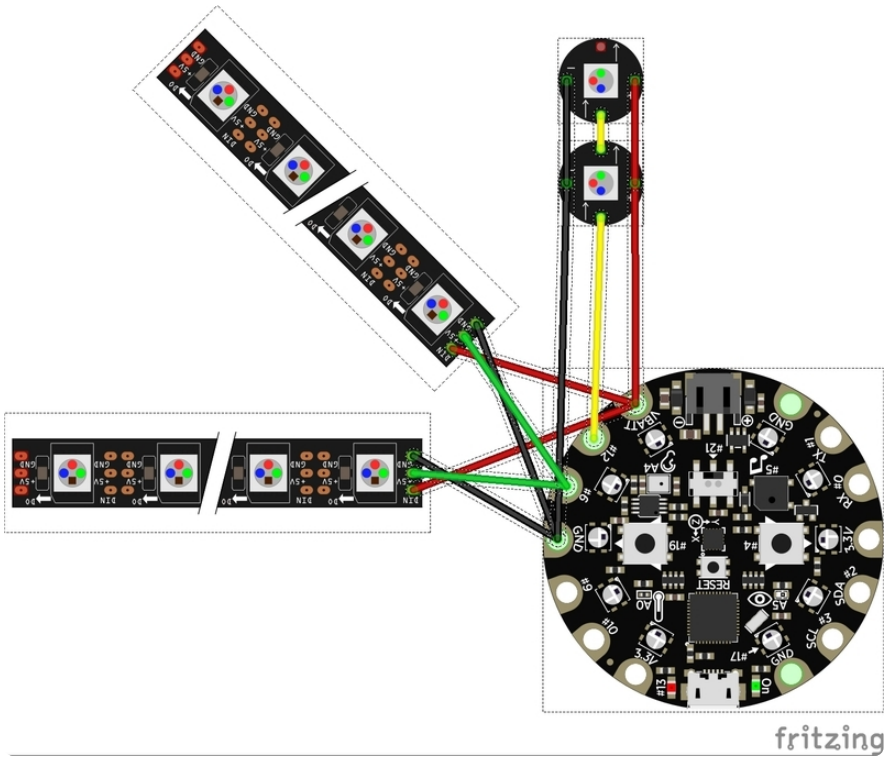
Details

This was my first time using black Worbla for details. This stuff is amazing. It's a thermoplastic sculpting material that's made more-or-less EXACTLY for this type of project. Hit it with a heat gun for a few seconds and it gets soft and melty. Sculpt it into the shape you want with a few finger swipes and when it cools off you have perfect detail with a good amount of strength. You can reheat it again and again until you get it right, and then it takes paint beautifully.

It's pretty thin so I wouldn't use it for structural elements, but our acrylic and foam will take care of that, and the Worbla does an amazing job with all the rest.

One caveat is that the Worbla will always melt at high temperatures.. so don't leave the bracers in your car in the summertime or you may come back to find that all your details have melted away.

Wiring Diagram



The wiring for this project is fairly simple, since all the sensors are already built into the Circuit Playground. All we need to do is add more lights.

Two strips of skinny neopixels are both soldered to pin 6 -- they'll be mirroring each other on each side of the bracer to light up the acrylic. Be sure to solder to the "in" end of the neopixel strip. Connect IN to pin 6, + to VBATT and - to GND.

We'll use the same VBATT and GND pin on the Circuit Playground to connect two additional individual neopixels, using pin 12 for data. These will be placed randomly near the wrist to add even more bullet-blocking glitter.

Notice that you'll be twisting 3 wires together to go into G and 3 wires to go into VBATT. Pin 6 has just one wire and pin 12 has two.

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.

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 (by Phil Burgess) into a new Arduino window and click "upload".

```
// "Amazon bracelets" sketch for Circuit Playground and NeoPixels.
// Press the RIGHT button on Circuit Playground for AUTO mode -- LEDs
// are then ON most of the time, best for photographing one's get-up.
// Press the LEFT button for DEFLECT mode -- LEDs are off until a loud
// sound or fast movement are detected.  Pew pew!  Deflect bullets!
// Triumph not with fists or firepower, but with love!

#include &lt;Adafruit_CircuitPlayground.h&gt;
#include &lt;FastLED.h&gt;

#define CP_PIN           17 // Circuit Playground's NeoPixels are on pin 17
#define CP_LEN           10 // Number of NeoPixels on Circuit Playground
#define STRIP_PIN         6 // NeoPixel strip is connected to this pin
#define STRIP_LEN        13 // Length of NeoPixel strip
#define SINGLES_PIN      12 // Single NeoPixels are connected to this pin
#define SINGLES_LEN       2 // Number of NeoPixel singles in chain
#define COLOR_ORDER     GRB
#define FPS              80 // Animation frames-per-second
#define TRIGGER_TIME 250000 // Debounce time after sound/motion trigger
```

```
(microseconds)

// SOUND REACTIVE SETUP --------------
// Higher number = less sensitive (louder sound required to trigger)
#define SOUND_THRESHOLD 200 // Range 0 to 341


// MOTION REACTIVE SETUP --------------
// Higher number = less sensitive (faster acceleration required to trigger)
#define G_THRESHOLD 3.0 // Range 1.0 to 8.0 G's

int deflectionping = 1; // Bullet deflection "ping" sound.  Change to 0 to turn
this off

// LED animation patterns cycle among several states.  There are two main modes
// (selected with the two buttons on Circuit Playground), each with some sub-states.
// In the 'auto' mode, the animation happens on its own, not in response to sound
// or motion.  In this mode it alternates between two states -- an 'idle' state
// with a solid color, and a periodic random flicker to add interest.  Best for
// photography, as it's all lit up most of the time.  In the 'deflect' mode,
// animation happens in response to sound and motion -- certain stimuli trigger a
// series of states for a 'deflected bullet' animation.  These goes from idle
// (awaiting sound/motion), flicker (the initial animation upon a 'bullet hit'),
// solid (brief state after hit) and back to idle (during which any lit LEDs will
// slowly fade), unless another bullet is deflected.
// An 'enumeration' is just the thing for this -- it's a C feature that creates a
// list of sequential integer constants by name, and ensures any variables of this
// type are not assigned outside the list.
enum {
  AUTO_IDLE,           // Auto mode (no sound/motion reaction) idle appearance
  AUTO_FLICKER,        // Auto mode, periodic random flicker
  DEFLECT_IDLE,        // Deflect mode (sound/motion reactive) idle appearance
  DEFLECT_FLICKER,     // Deflect mode, short flicker upon deflecting a bullet
  DEFLECT_SOLID,       // Deflect mode, short solid-color appearance
} mode = DEFLECT_IDLE; // Start out in DEFLECT_IDLE mode

// Color for solid() mode
#define HUE         24 // Orangey fire-ish color
#define SATURATION 255
#define BRIGHTNESS 255

CRGB cp[CP_LEN],               // Separate arrays for Circuit Playground pixels,
     strip[STRIP_LEN],         // strip and singles, so brightness can be
     singles[SINGLES_LEN];     // controlled separately if needed

uint32_t lastFrameTime   = 0L, // Used for frame-to-frame interval timing
         lastTriggerTime = 0L; // Used to 'debounce' sound &amp;amp; motion inputs
uint16_t frameCounter;         // Counter for animation timing

void setup() {
  CircuitPlayground.begin();
  FastLED.addLeds&lt;WS2812B, CP_PIN     , COLOR_ORDER&gt;(cp     ,
CP_LEN     ).setCorrection(TypicalLEDStrip);
  FastLED.addLeds&lt;WS2812B, STRIP_PIN  , COLOR_ORDER&gt;(strip  ,
STRIP_LEN  ).setCorrection(TypicalLEDStrip);
  FastLED.addLeds&lt;WS2812B, SINGLES_PIN, COLOR_ORDER&gt;(cp,
SINGLES_LEN).setCorrection(TypicalLEDStrip);  //singles are set up with the cp
array so they act like the Circuit Playground onboard LEDs
  set_max_power_in_volts_and_milliamps(5, 500); // FastLED 2.1 Power management set
at 5V, 500mA
  fill_solid(cp     , CP_LEN     , CRGB::Black);
  fill_solid(strip  , STRIP_LEN  , CRGB::Black);
}

void loop() {
  uint32_t currentTime;

  // Frame-to-frame timing is kept semi-consistent-ish by checking the current
  // time against the prior-frame time.  Rather than burning off this time with
```

```
  // a delay() call, this is a perfect opportunity to test for any button,
  // sound or motion inputs...repeatedly, if need be, until the elapsed frame
  // interval is passed.  A do/while loop is used so these tests are always
  // performed at least once per frame (else inputs might be ignored if there's
  // heavy animation going on).
  do {
    currentTime = micros();

    // Check first for button presses...
    if(CircuitPlayground.leftButton()) {         // Left button pressed?
      mode            = DEFLECT_IDLE;           //   Yes, switch to motion/sound
reactive mode
      lastTriggerTime = currentTime;            //   Note time of activation
      while(CircuitPlayground.leftButton());    //   Wait for button release
    } else if(CircuitPlayground.rightButton()) { // Right button pressed?
      mode            = AUTO_IDLE;              //   Yup, switch to 'auto' mode
(not reactive)
      frameCounter    = FPS;                    //   AUTO_IDLE animation for 1
second
      lastTriggerTime = currentTime;            //   Note time of activation
      while(CircuitPlayground.rightButton());   //   Wait for button release

    // Else no button presses.  If we're in a 'deflect' mode,
    // and if sufficient time has passed since last trigger...
    } else if((mode &gt;= DEFLECT_IDLE) &amp;&amp;
              ((currentTime - lastTriggerTime) &gt;= TRIGGER_TIME)) {
      // ..then check for sound and motion...

      int value = CircuitPlayground.soundSensor() - (1023 / 3); // Audio amplitude
      if(abs(value) &gt;= SOUND_THRESHOLD) {    // Loud noise?
          ping();
          mode            = DEFLECT_FLICKER; //   Bullet deflected!  Pew pew!
          frameCounter    = FPS * 3 / 2;     //   Flicker for 1.5 sec
          lastTriggerTime = currentTime;     //   Turn off detection for a moment
      } else {                               // No sound, check accelerometer...
        sensors_event_t event;
        CircuitPlayground.lis.getEvent(&amp;event);
        // Compare magnitude of accelerometer reading against G_THRESHOLD.
        // Costly sqrt() is avoided by ^2 the threshold value for comparison.
        if(((event.acceleration.x * event.acceleration.x) +
            (event.acceleration.y * event.acceleration.y) +
            (event.acceleration.z * event.acceleration.z)) &gt;=
           ((9.8 * G_THRESHOLD) * (9.8 * G_THRESHOLD))) {
          ping();
          mode            = DEFLECT_FLICKER; // Bullet deflected!  Pew pew!
          frameCounter    = FPS * 3 / 2;     // Flicker for 1.5 sec
          lastTriggerTime = currentTime;     // Turn off detection for a moment
        }
      }
    }
  // Repeat until frame interval has elapsed:
  } while((currentTime - lastFrameTime) &lt; (1000000L / FPS));
  lastFrameTime = currentTime; // Save time for next frame

  // Done with button/sound/motion sensing.  LEDs are updated immediately after
  // interval test, for more uniform timing.  So this actually displays the colors
  // calculated on the *prior* loop() pass...
  FastLED.show();
  frameCounter--; // Count down to zero (cycles modes)
  // Then we render one frame of animation for the *next* pass...

  switch(mode) {

    // The first two modes -- AUTO_IDLE and AUTO_FLICKER -- are used when
    // sound/motion reactivity has been turned off.  The code will alternate
    // between these two states until reactivity is switched on (left button).
    case AUTO_IDLE:
      solid();                    // Show solid color
      if(!frameCounter) {         // for desired number of frames, then...
```

```
        mode = AUTO_FLICKER;      //   Switch to AUTO_FLICKER mode and
        frameCounter = FPS * 2;   //   set it to flicker for 2 seconds
      }
      break;
    case AUTO_FLICKER:
      flicker();                  // Show occasional random flicker
      if(!frameCounter) {         // for desired number of frames, then...
        mode         = AUTO_IDLE; //   Switch back to AUTO_IDLE mode
        frameCounter = random(3 * FPS, 10 * FPS); // for 3 to 10 sec.
      }
      break;

    // The remaining modes are used when 'deflecting bullets.'
    case DEFLECT_IDLE:                      // Awaiting action
      fade();                               // Dim LEDs slightly
      break;
    case DEFLECT_FLICKER:                   // Initial reaction to bullet hit
      flicker();                            // Random-ish LED flickering
      if(!frameCounter) {                   // FLICKER time elapsed?
        mode         = DEFLECT_SOLID;       //   Switch to DEFLECT_SOLID mode
        frameCounter = FPS / 2;             //   for next 1/2 sec
      }
      break;
    case DEFLECT_SOLID:                     // Short 'solid' period follows flicker
      solid();                              // Solid color
      if(!frameCounter) {                   // SOLID time elapsed?
        mode         = DEFLECT_IDLE;        //   Switch back to IDLE mode
      }
      break;
  }
}

void fade() {
  fadeToBlackBy(cp     , CP_LEN     , 85); // Fade Circuit Playground LEDs by 1/3
  fadeToBlackBy(strip  , STRIP_LEN  , 8);  // Fade other LEDs slightly
}

void flicker() {

  fadeToBlackBy(cp, CP_LEN, 85); // Fade Circuit Playground LEDs by 1/3
  if(!random(8)) {               // Then, if random 1/8 chance...
    // Set one random pixel on Circuit Playground to white
    cp[random(0, CP_LEN)] = CRGB::White;
  }
  if(!random(8)) { // A separate 1/8 chance...
    // Set all other LEDs to a random brightness
    CRGB c = CHSV(HUE, SATURATION, random(32, BRIGHTNESS));
    fill_solid(strip  , STRIP_LEN  , c);
  } else {
    // Fade others slightly
    fadeToBlackBy(strip  , STRIP_LEN  , 8);
  }
}

void solid() {
  fadeToBlackBy(cp, CP_LEN, 85); // Fade Circuit Playground LEDs by 1/3
  CRGB c = CHSV(HUE, SATURATION, BRIGHTNESS);
  fill_solid(strip  , STRIP_LEN  , c); // Set other LEDs to solid color
}

void ping() {
  if (deflectionping==1) {
    CircuitPlayground.playTone(random(2000,2200), 30);
  }
}
```

Press the RIGHT button on the Circuit Playground for AUTO mode -- lights are on all the time, great for photo shoots and peace talks. You'll see random flashes every few seconds on the Circuit Playground's onboard LEDs, with no effort on your part.

Press the LEFT button to enter DEFLECT mode. Make some noise and/or shake your Circuit Playground around a bit. The motion and sound will trigger a bullet-blocking animation. The Circuit Playground will make a "ping" sound and the lights will flash randomly, simulating the spark of a bullet strike.

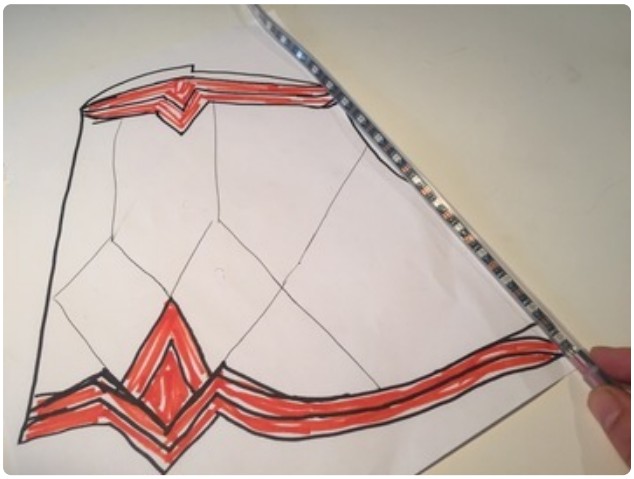
Note: If this "ping" sound drives you crazy, it's easy to turn it off in the code. Look for this line:

```
int deflectionping = 1; // Bullet deflection "ping" sound.  Change to 0 to turn
this off
```

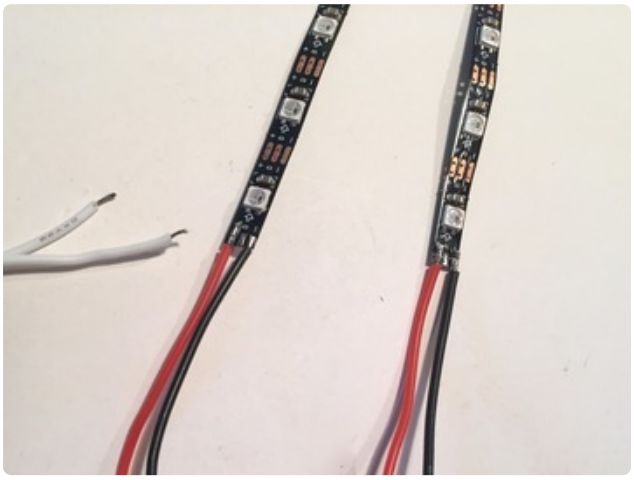
Change the 1 to a 0 to turn off the sound.

Neopixel Assembly

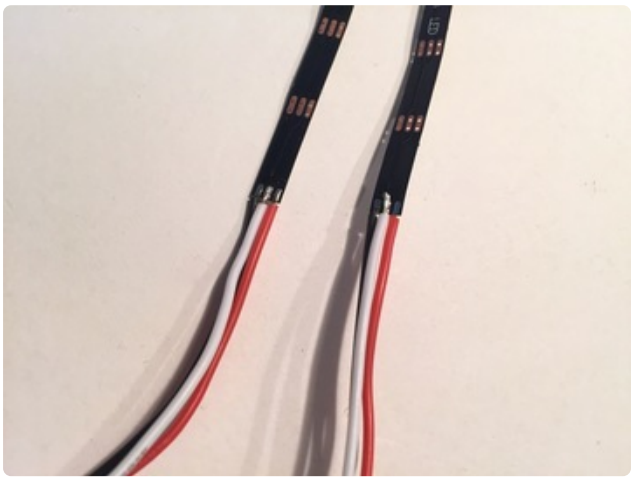
Neopixel Strips



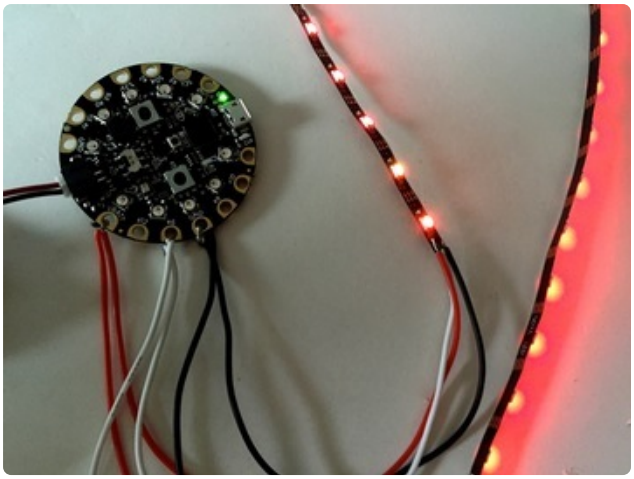
Measure out how many neopixels you'll want on each side against your pattern. Carefully cut through the solder pads between pixels and slip the silicone sleeve off.



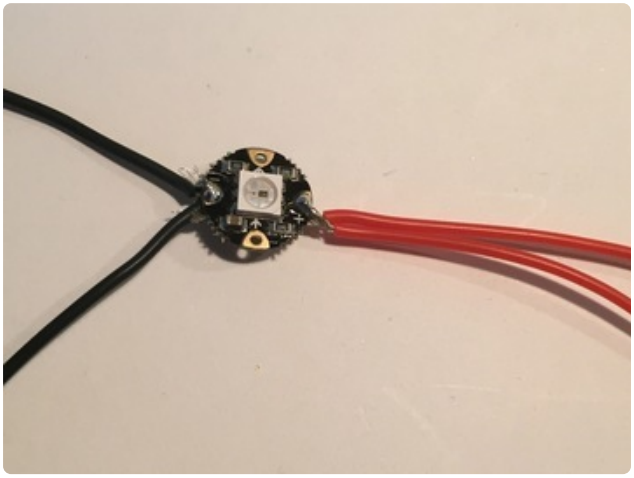
Tin the pads on both sides of the pixels at the "**in**" end. Cut two 6-8" red, white, and black wires. Solder the red and black to the front of the neopixel strip to + and -, and solder the white wire to the back of the neopixel strip (double checking again that you're on the **"in"** end).



Temporarily twist the matching wires from each strip together and twist them into the Circuit Playground's VBAT, 12 and G pins. Turn on the Circuit Playground and be sure all the lights come on and work. This is a great time to find any bad or broken pixels!

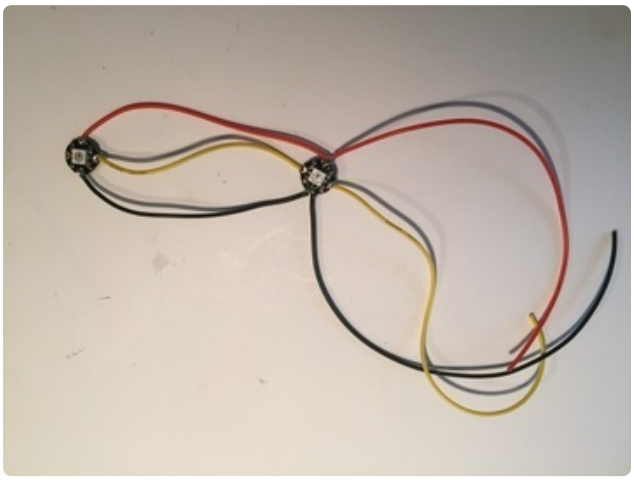


Single Pixels



Cut two more 6" red, black and black wires, and 2 6" yellow wires

Solder both red and black wires into the + and - pads of the first pixel. Solder one yellow wire to both the in and out pads.



Solder the second pixel "in" to the wire coming "out" of the first pixel, and solder the red and black wires to this pixel's + and - pads.

Temporarily twist-test these with your Circuit Playground as well to be sure they're working.

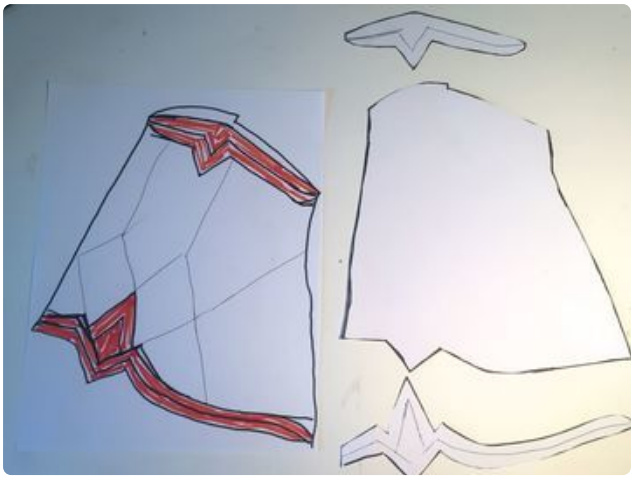
We will wait a bit to solder the lights to the Circuit Playground; first, let's prep our acrylic and get our layout right so we can be sure we have the wires the right length.

Pattern & Acrylic

Making the Pattern

I made a custom pattern by wrapping my arm in saran wrap, then duct tape. I drew my details onto the tape while it was on my arm so everything would line up correctly, then carefully cut it off along the opening on the underside of my arm.

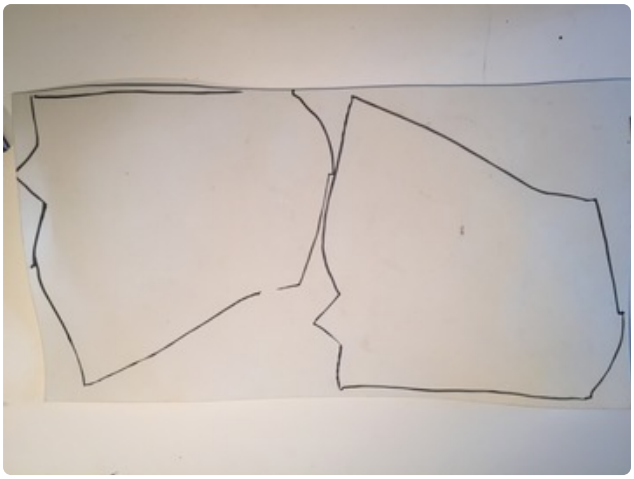
It helps to have a friend assist you with this part.



Copy the pattern onto a piece of paper and trace it out a few times to finalize your details.

Use the pattern and some aluminum foil to make a mockup of your forearm*. Get this as close as possible to the size and shape of your actual arm.

***Feel free to make a tin foil hat at the same time, nobody is looking.**

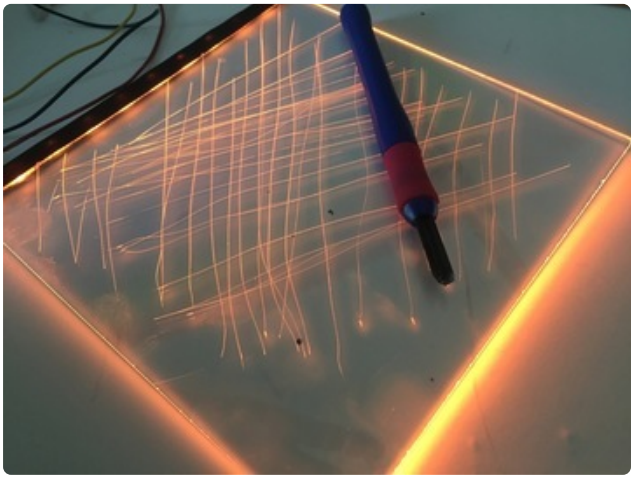


Trace the pattern onto your craft foam. Be sure you flip the pattern so you have a left and a right.

Spray the outside of the foam with Plasti-dip to seal the foam and give it a nice leather-like texture. I found it more comfortable to leave the inside of the foam in its natural state.

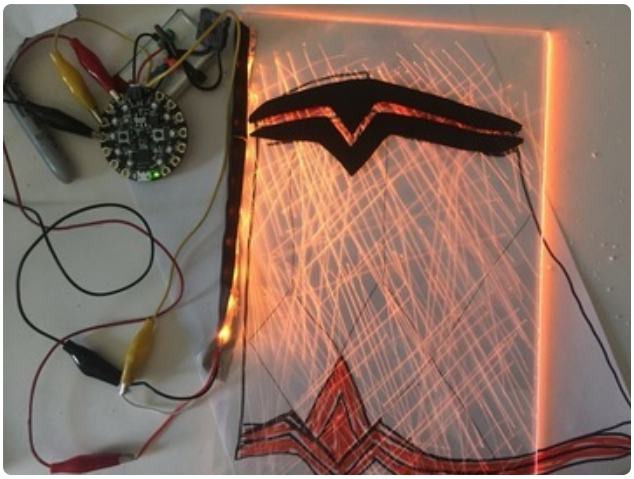
Set the foam aside for now while it dries completely.

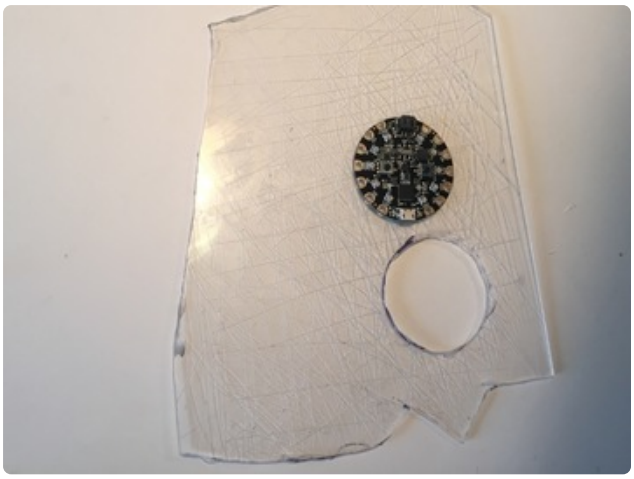
Acrylic



Peel the plastic off your acrylic and admire its perfection. Tape one of your neopixel strands along one edge. Then grab your favorite sharp object (https://adafru.it/vDK) and make some scratches.

Really gouge the heck out of it. I mean really. Take out all your aggression. Scratch it some more in the other direction. Remember, the cracks are where the light gets in.

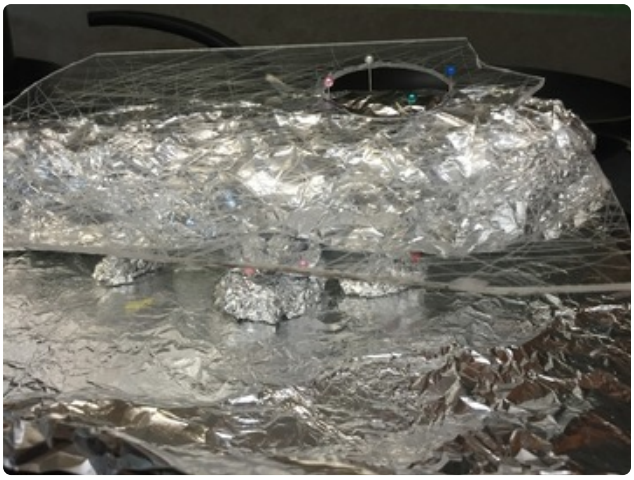




Use a piece of acrylic that covers MOST of your pattern but not all. The acrylic will be unyielding once it's shaped, so plan for it to cover just the top half of your arm rather than wrapping all the way around, or you won't be able to get your bracer on and off.

Place your Circuit Playground into the design and trace around it. With the scratches on the back side, cut out your shape with a rotary tool (or a laser cutter would work great here too). Remember to wear your safety glasses! This stuff gets everywhere. Polish up the edges so they're as smooth and shiny as possible.

Go find your aluminum foil arm. Pin some little aluminum foil feet to the bottom side to hold it up off the ground a bit and give more room for the acrylic to move and melt.



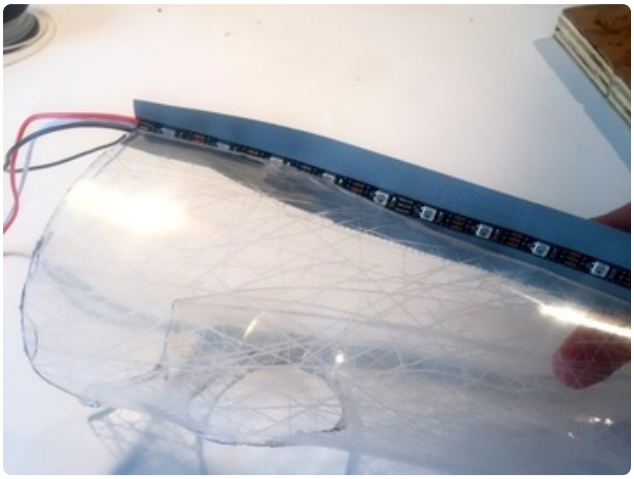
Line up your acrylic with the center line of your aluminum foil arm and balance it on a foil lined baking sheet in the oven. Sewing pins can help hold it in place. Bake for about 10-15 minutes at 300 degrees, until the acrylic is soft and pliable.

CAREFULLY shape the warm plastic with oven mitts until it fits your arm. Be patient - you may need to reheat it a couple of times and / or reform your aluminum foil arm to be closer to the shape of your actual arm.

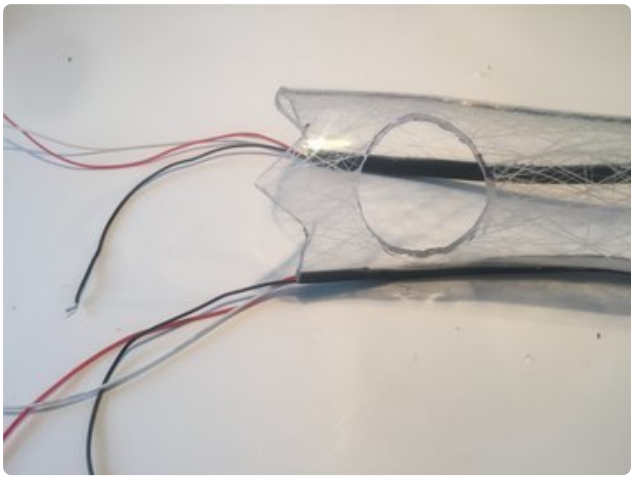
Slightly too big is better than too small here. You'll have a layer between your arm and the acrylic so leave a little space.

Kevlar mechanic heat-resistant sleeves (https://adafru.it/vDL) can help a lot in this stage. The plastic is hot! Be careful! Once it cools it will hold its shape very well.

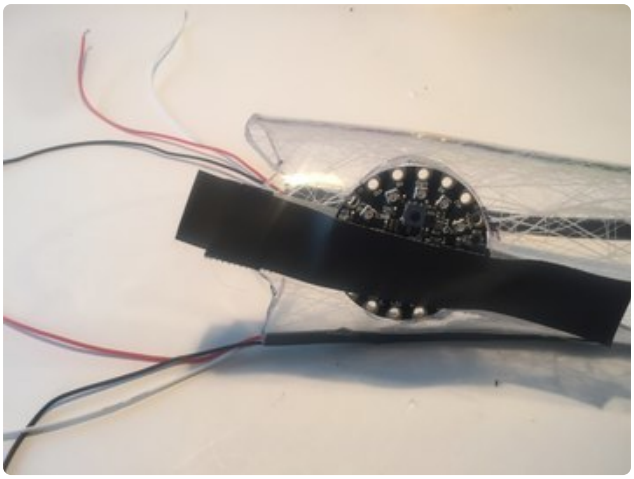
Bracer Assembly

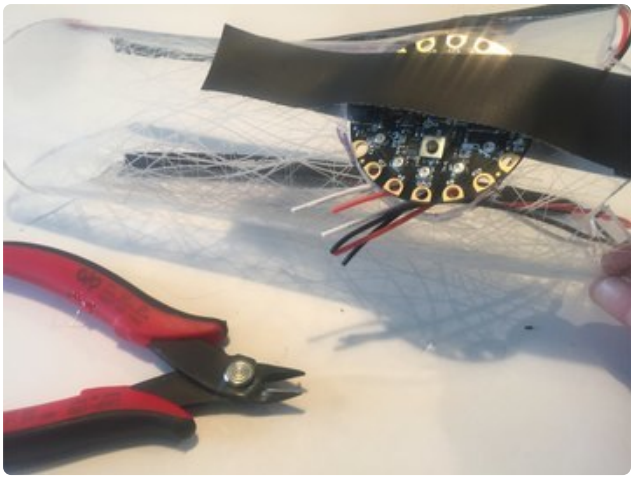


Cut a piece of gaffer's tape to the length of your pixels and about 3/4" wide. Center your pixels on the tape.

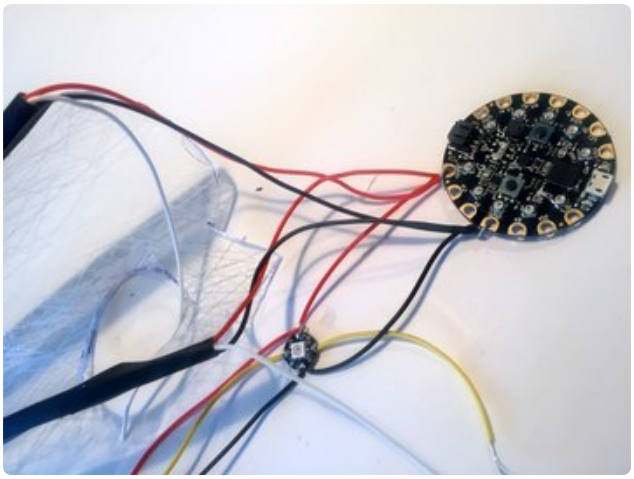


Tape the pixels securely to both sides of your acrylic with the wires coming out the bottom end (the end toward your elbow).

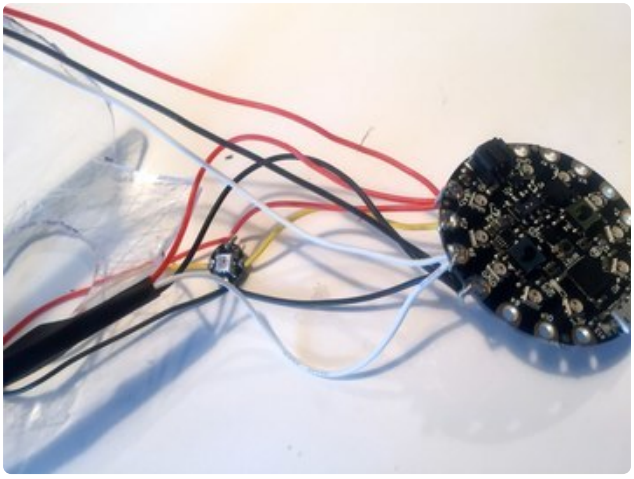




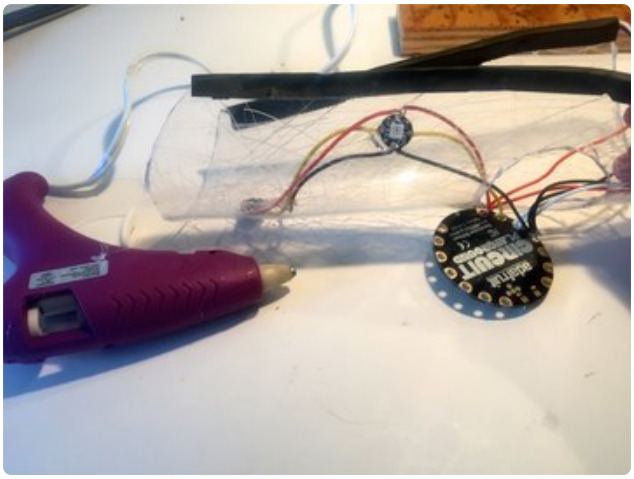
Temporarily tape the Circuit Playground in place with the USB port pointing at your elbow and the battery port pointing at your wrist. Feed the wires along the inside of the bracer, flush with the plastic, until they comfortably reach the VBAT, 12, 6 and G pads. When you find the appropriate wire length, trim the wires.



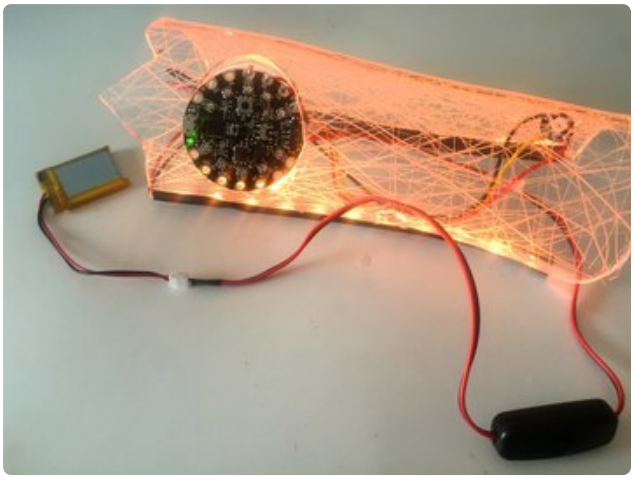
Grab your loose neopixels and measure their wire lengths the same way. Then, twist all 3 red wires together and solder into VBAT. Do the same with the black wires into G.



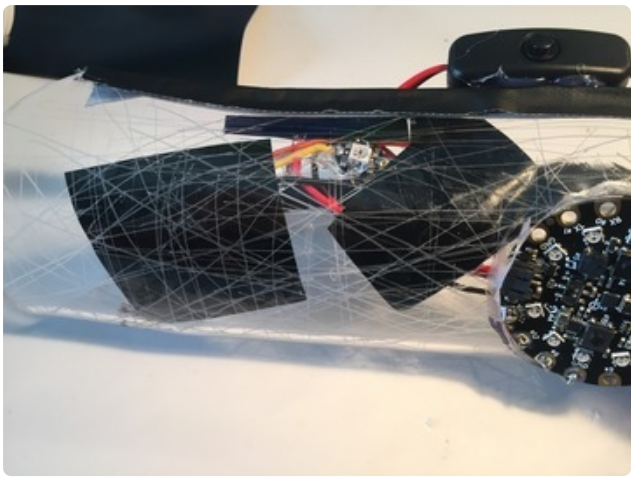
Connect the data wires from the neopixel strips to pin 6, and the data wire from the individual pixels to pin 12.



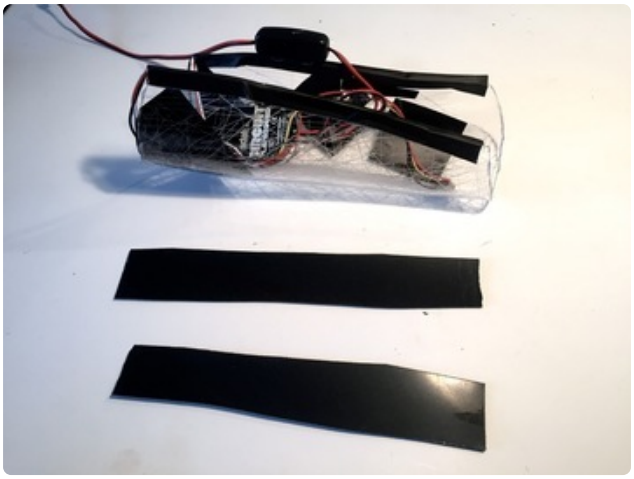
Glue the individual pixels face-out against the acrylic in an artistically random arrangement. These pixels will provide the "sparks" from the imaginary bullets you're blocking.



Plug your battery in and shake the bracer around a little, making sure everything works.

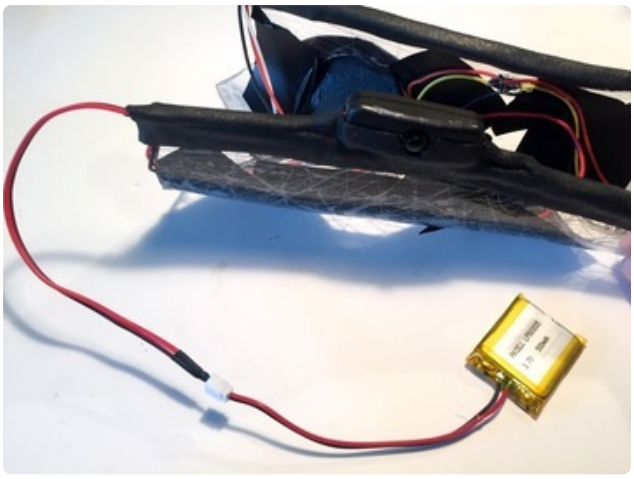


If you don't want your wires showing, add some black tape or contact paper over them on the inside of the acrylic.

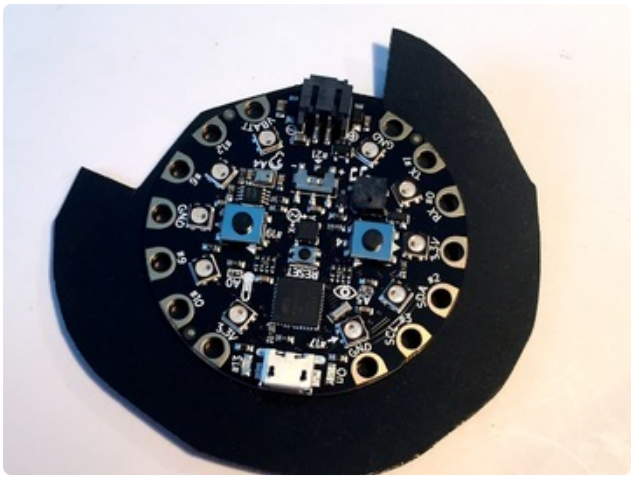


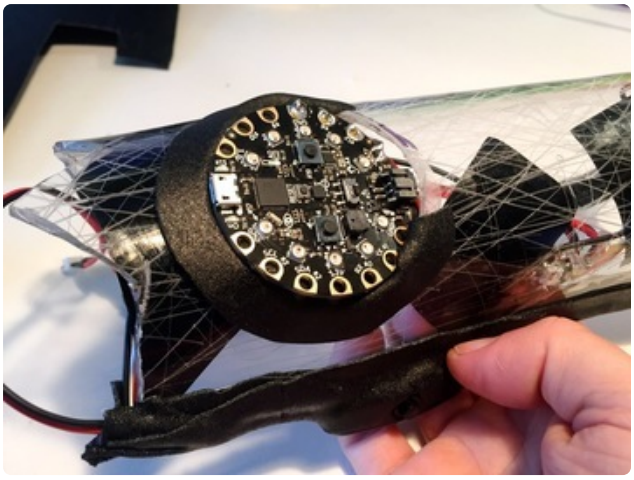
Cut two strips of Worbla to the length of your bracer and about 1" wide. Cut a slit in one of the strips to accommodate your on/off switch.

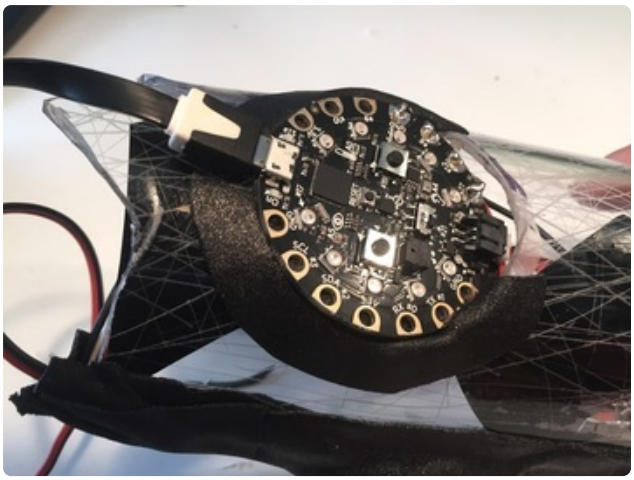
Hot-glue your switch to the **outside** edge of your acrylic and mold the Worbla over it to hold it in place. Hide the battery wire inside the Worbla edging.



Place the other strip of Worbla over the neopixels on the opposite edge.







Cut a circle of Worbla to create a mount / cradle for the Circuit Playground. Be sure to leave space for the wires and the battery cable to pass through.

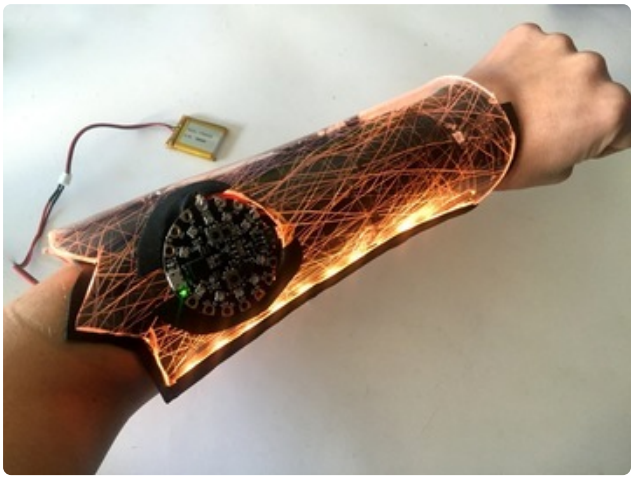
With the Circuit Playground tilted slightly so the battery cable can pass on the **inside** of the gauntlets, and the USB port is accessible from the **outside**, gently melt the Worbla in place around the Circuit Playground.

Decoration & Finishing

Time to make them look Wonder-riffic!

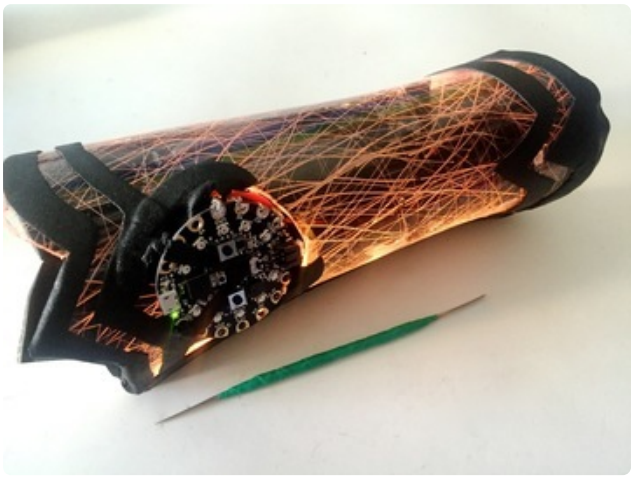
Foam Lining

Place the foam pattern pieces inside the cuff and secure them with glue. Hot glue works surprisingly well here -- the plasti-dip foam will tear before the glue will yield. Hot glue sure loves foam and acrylic.



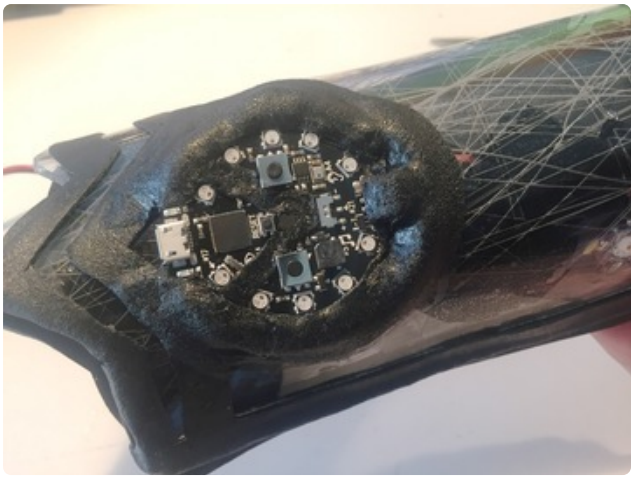
My other favorite thing about hot glue is it's "undo" feature: once it's set, you can use 99% alcohol to release the hold without damaging your foam or acrylic one bit. So if you don't get it lined up perfectly the first time you can try again.

Worbla Details



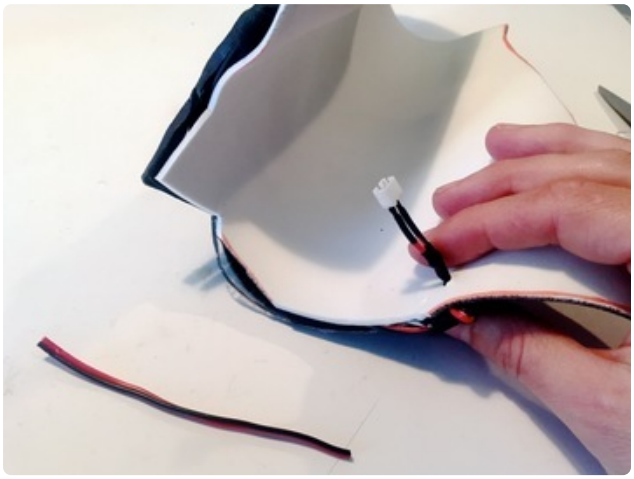
Trace your pattern details onto your sheet of worbla with a sharpie. Pay attention to whether you're doing the **right or the left** and be sure your pattern's facing the correct way. Cut them out.

Use a heat gun to gently melt the worbla and stick it in place on your acrylic. Smooth it down with a sculpting tool. No need to glue, it will stick in place really well once it hardens.



Use some more worbla to decorate the setting for the Circuit Playground. Be sure to leave the USB port accessible (plug a cable in to be sure) and don't cover up the lights or the buttons.

Battery Cable



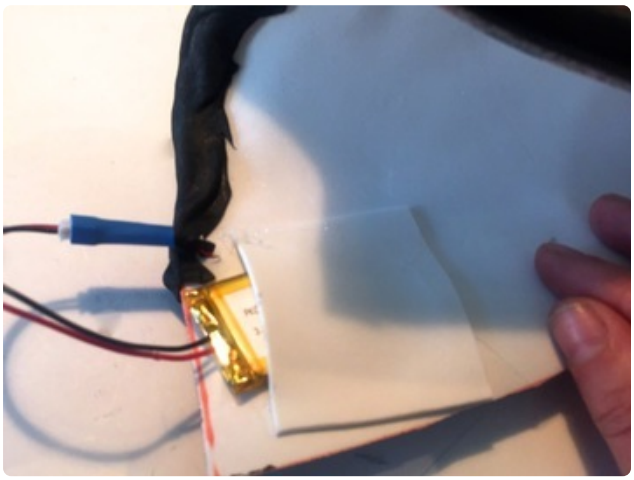
Cut a slit in the foam and feed your battery cable through. Mine was really long so I trimmed the wire and re-soldered the connector to minimize the chance of pulling or tangling.

Robust-ify the connector by covering it in 1/2" heat shrink. Optionally put a little hot glue inside the heat shrink to be sure it's as bulletproof as you are, but be sure not to get any glue inside the connector.

Edges

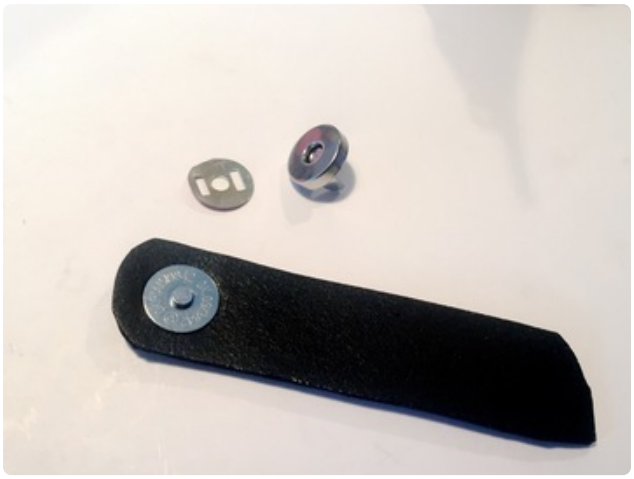
Mold a strip of Worbla along the top and bottom edges, covering both the foam and the acrylic. It will get stiff, so only do this along the acrylic, and leave the foam edges free to bend around your arm.

Battery Pocket



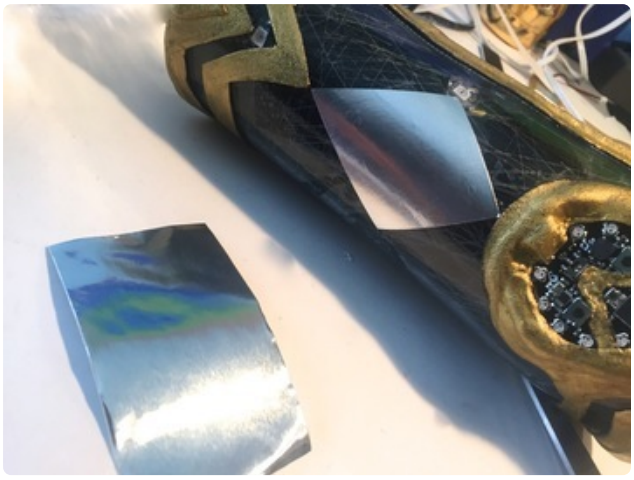
Glue a piece of foam to the inside near the battery cable to form a battery pocket.

Straps



Grab your plasti-dipped straps and affix the male side of your snaps to one end. Attach the female snaps to the bracer's other edge and then glue the straps in place so they're the appropriate size.

Paint and Finish



Time to add the silver and gold! I used acrylic metallic paint directly on the Worbla. (My aluminum foil arm came in "handy" again for this part)

Cut stickers out of aluminum tape and carefully smooth them on to give a metallic finish. These will block the light entirely so be sure you don't cover the individual neopixels.

Use & Calibration

Press the RIGHT button on Circuit Playground for AUTO mode -- LEDs are then ON most of the time, best for photographing one's get-up.

Press the LEFT button for DEFLECT mode -- LEDs are off until a loud sound or fast movement are detected. Pew pew! Deflect bullets! Triumph not with fists or firepower, but with love!

Calibrating for Sensitivity

Once your bracers are finished, put them on try out your Amazon high-blocks. Test for the ideal motion and sound needed to trigger the animation purposefully.

Look for these lines in the code:

```
// SOUND REACTIVE SETUP --------------
// Higher number = less sensitive (louder sound required to trigger)
#define SOUND_THRESHOLD 200 // Range 0 to 341


// MOTION REACTIVE SETUP --------------
// Higher number = less sensitive (faster acceleration required to trigger)
#define G_THRESHOLD 3.0 // Range 1.0 to 8.0 G's
```

Play with adjusting the numbers up and down until the bracers react the way you want. Convention halls and events have a very high noise threshold. Be sure to try

them out in various different environments so they only trigger when you want them to.

It helps to have a henchman minion assistant who can make sounds for you at a bit of a distance, since the microphone is pretty sensitive to proximity. This means that any tribal Amazon screams you make will be quite likely to set them off, but I count that as a win.