Bunny Ears with MakeCode

Created by Erin St Blaine

https://learn.adafruit.com/bunny-ears-with-makecode

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

Table of Contents

Introduction

Everyone needs a friend who is all ears. Get egg-cited and make a set of motion-reactive light up bunny ears with Circuit Playground Express and MakeCode.

These are the perfect accessory to draw attention away from your bad hare day. Be the fast and the furriest; the most unique bunny in the park. (How do you catch a unique bunny? Unique up on it)

Wear them to IHOP or take them on your Bunnymoon, or just use them to cover your receding hare line. You'll be so sparkly that everyone will want to hop down YOUR bunny trail.

This project is a great one to do with kids. There's a bit of soldering involved, a bit of hand sewing, and you'll have fun dragging and dropping code snippet blocks with MakeCode, to make the ears react to your bunny hops. Egg-spress yourself with Circuit Playground!

1 x Circuit Playground Express — https://www.adafruit.com/product/3333
Circuit Playground Express Microcontroller

1 x Neopixels — https://www.adafruit.com/product/1138
60/m White Neopixel Strip

1 x LiPoly Battery — https://www.adafruit.com/product/1578
500mAh LiPoly battery

1 x Battery Charger — https://www.adafruit.com/product/1304
USB LiPoly Battery Charger

1 x On/Off Switch — https://www.adafruit.com/product/3064
Battery extension cable and on/off switch

3 x Stranded Wire — https://www.adafruit.com/product/1970
Red, Black and White stranded wire

Tools & Other Materials

- Bunny ears (https://adafru.it/C9W)
- Hair clips / barettes
- Soldering iron & accessories
- Clear 1/4" heat shrink (https://adafru.it/C9X)

- Hot glue gun
- Heat gun
- Needle & thread

Programming with MakeCode

The Circuit Playground Express can be programmed a number of ways: it will run Arduino code, CircuitPython, or you can program it with MakeCode.

Microsoft MakeCode for Adafruit is a web-based code editor for physical computing. It provides a block editor, similar to Scratch or Code.org, and also a JavaScript editor for more advanced users.

This means you can drag and drop light animations and functionality using the Circuit Playground Express' onboard sensors without ever writing a single line of code. Just snap the blocks together and watch your lights dance.

Check out our MakeCode intro guide here.

https://adafru.it/AEp

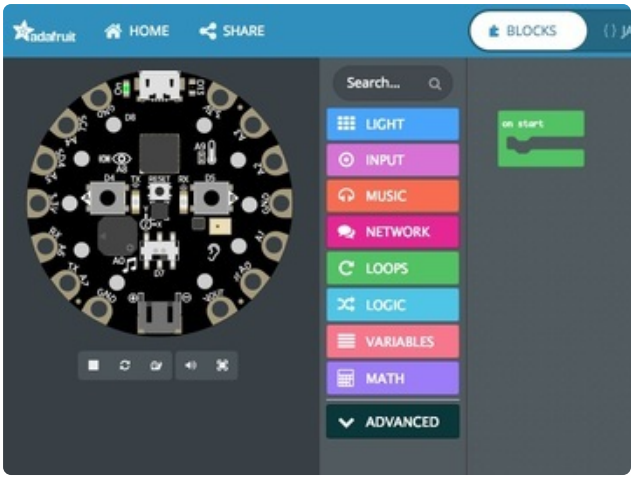
For this project, I'm using the tilt and shake functions on the Circuit Playground Express to trigger animations. Tilt your head left and a "photon" animation runs from left to right. Tilt your head right and the same animation runs from right to left. If you

shake your head (or hop up and down like a bunny) then the twinkle animation is triggered.

Check out this guide (https://adafru.it/CjC) for more info on working with neopixel strands and building your own animations.

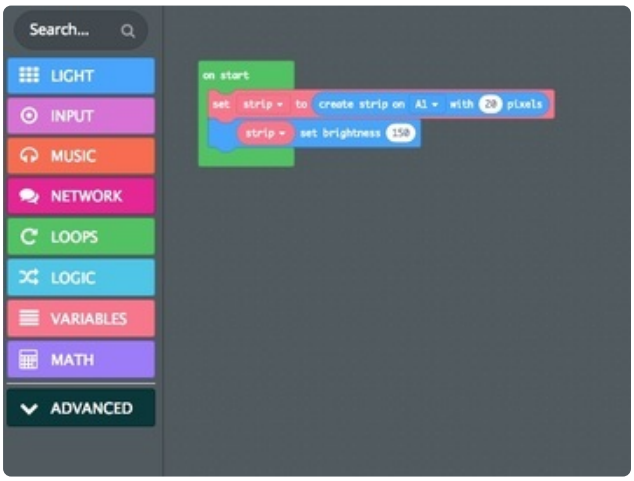
Go to https://makecode.adafruit.com/ (https://adafru.it/wmd)and select "New Project".

Set Up the Light Strand



We soldered our light strand to pin A1. Tell the code by adding an `on start` loop (Loops > `on start`). This block will run ONCE when the board powers up, so it's the perfect place to define our setup.

Drag the `on start` block onto the workspace. We don't need the "forever" block so you can delete it.

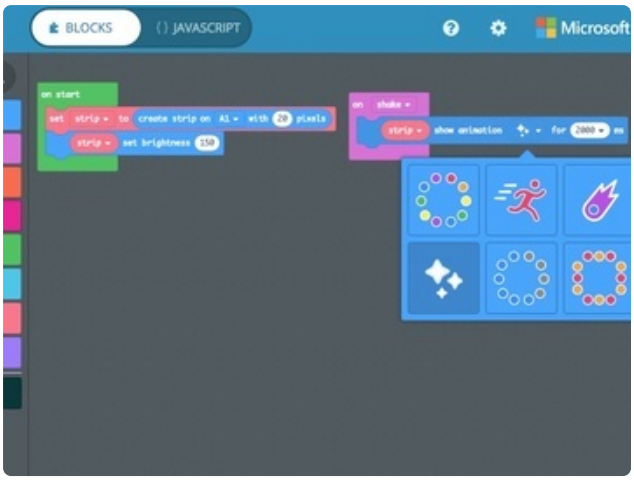


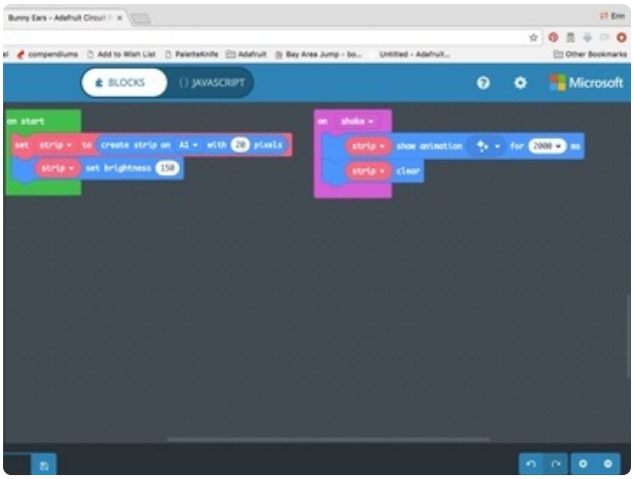
Next go to "Variables" add a new variable called `(strip)`. Then, drag `set (item) to 0` into the `on start` block.

Change the default `(item)` variable to the one you just made: `(strip)`.

Go to the "Neopixel" bin (you may need to click the "Light" bin to make it appear). Drag `create strip on (A1) with (24) pixels` into the block you just made, replacing the `(0)`. Change the `(24)` to reflect the number of pixels you have in ONE ear (I have 20).

Add "Shake" Animation





Go to "input" and grab the `on (shake)` block.

Go back to the "Neopixel" bin and choose `(strip) show animation for (500) ms`

You can choose from six different pre-programmed animations. I chose the Twinkle animation. You can also specify here how long you'd like the animation to run when you shake / hop. I liked 2 seconds -- experiment to find what works for you.

To make the lights turn off at the end of the specified time, place a Neopixel > `(strip) clear` block after the animation block.

Download Code

Let's test to see if it's working.

1. Plug your Circuit Playground Express into your computer with a USB cable.
2. Click the reset button.
3. Green lights will appear on the Circuit Playground's face and it will appear in your list of devices, called **CPLAYBOOT**.

If you don't see this, try double-clicking the reset button instead of single-clicking.

Click the pink **Download** button on your MakeCode screen and the code you just made will download to your computer. Drag it onto the **CPLAYBOOT** device.

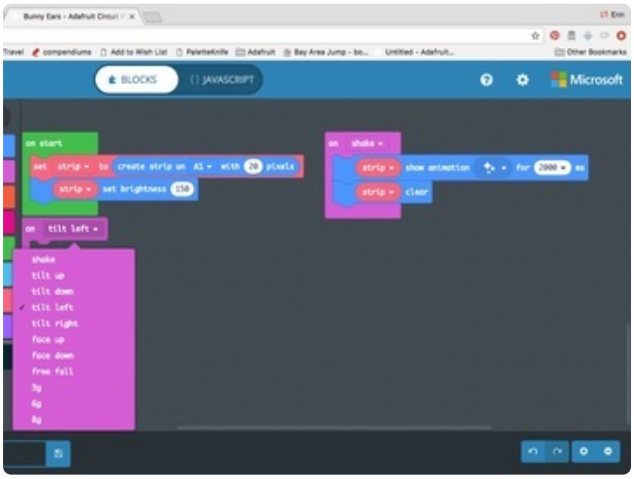
Shake your Circuit Playground and see the neopixel strands twinkle!

Add Tilt Animations

Let's make the lights react when you tilt your head to the side. We'll use the "Photon" option for this. It's a little more complicated to use than the pre-canned animations, but you have a lot more control as well.

Photon moves a light or series of lights along the light strand from one end to the other. You can control a lot of factors like color, speed and tail length.

More about using the Photon animation in this guide (https://adafru.it/CjJ).



Drag another instance of `on (shake)` from the "Inputs" bin. Change `(shake)` to `(tilt left)`

To start the Photon animation, drag an instance of `(strip) photon pen (down)` from the "Neopixels" bin.

NOTE: there are also instances of the photon functions in the "Light" bin. These don't have a variable attached, which means these animations will run on the face of the Circuit Playground and not on your Neopixel strip. Be sure you've got the right one!

From the "Loops" bin, drag a `repeat (4) times` block inside the `on (tilt left)` block. Change the `(4)` to the number of neopixels in ONE strip (I have 20).

This will make the photon animation repeat for each neopixel you have, creating the apparent motion.

Add Neopixel > `(strip) photon forward by (1)` block inside the loop block.
Drag a `pause (500) ms` block below it.

These blocks will control your animation speed. The first block advances the light by one pixel and the second controls how quickly this happens. I changed mine to `pause (10) ms` because I want the animation to run really fast.

Get another instance of `(strip) photon (pen down)`. The quickest way to do this is to click on the block and copy it to your clipboard, and then paste it into your workspace (ctrl-c, ctrl-v). Change `(pen down)` to `(eraser)`. From here you can drag it below your loop function within the `on (tilt left)` block.

Copy and paste the loop function too, and drag that below the new `(eraser)` function you just made.

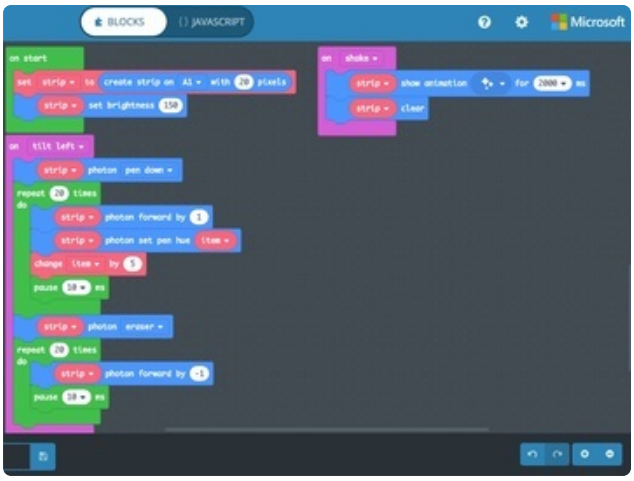
Change `(strip) photon forward by (1)` to `(strip) photon forward by (-1)`. This will make the animation disappear (eraser) in the reverse direction as it was originally drawn.

Shifting Color Hues

Each hue on the neopixel strip is assigned a value: red is 0, yellow is around 60, and the colors cycle back around to red at 255.

We can use a variable to make the colors change periodically. If we start with red (hue 0) and add 5, we get an orangey-red. Add 5 more, and then 5 more, and the color will cycle through to orange and yellow and so on, all the way around the rainbow. MakeCode is smart enough to automatically wrap back to 0 when it reaches 255.

Let's use the `(item)` variable, which is already set up for us.

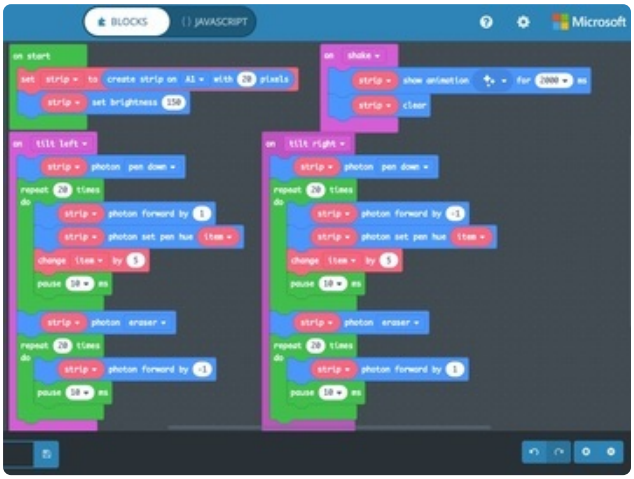


Drag Neopixel > `(strip) photon set pen hue (0)` into your loop as shown in the photo. Drag Variables > `(item)` to replace the `(0)`.

Drag Variables > `change (item) by (1)` below that. Set the `(1)` to `(5)`. This will change each LED's hue by 5 steps each time a pixel is colored. A value of 5 makes a nice gradient, but play with it to get a value you like.

Download the code and drag it to **CPLAYBOOT** to test your animations. You can download and test as many times as you want until you get it just right.

Once `(tilt left)` is working the way you want, we can copy/paste most of the blocks to add a `(tilt right)` animation.

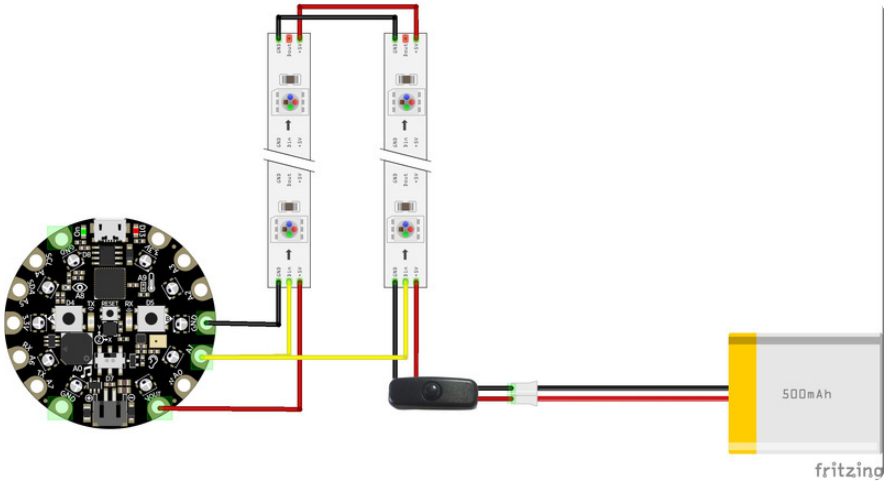


Click the on `(tilt left)` block and copy/paste another instance of it onto your worktop. Change on `(tilt left)` to `(tilt right)`.

Change the FIRST instance of `(strip) photon forward by (1)` to `(-1)` and the SECOND instance to `(1)`. This will make the animation run from right to left, instead of left to right.

Here's the completed project that you can play with directly.

Wiring Diagram



- Circuit Playground VOUT --> Neopixel 5v (to one strip)
- Circuit Playground G --> Neopixel G (to the same strip)
- Circuit Playground A1 --> Neopixel IN (**2 wires**: one to each strip)

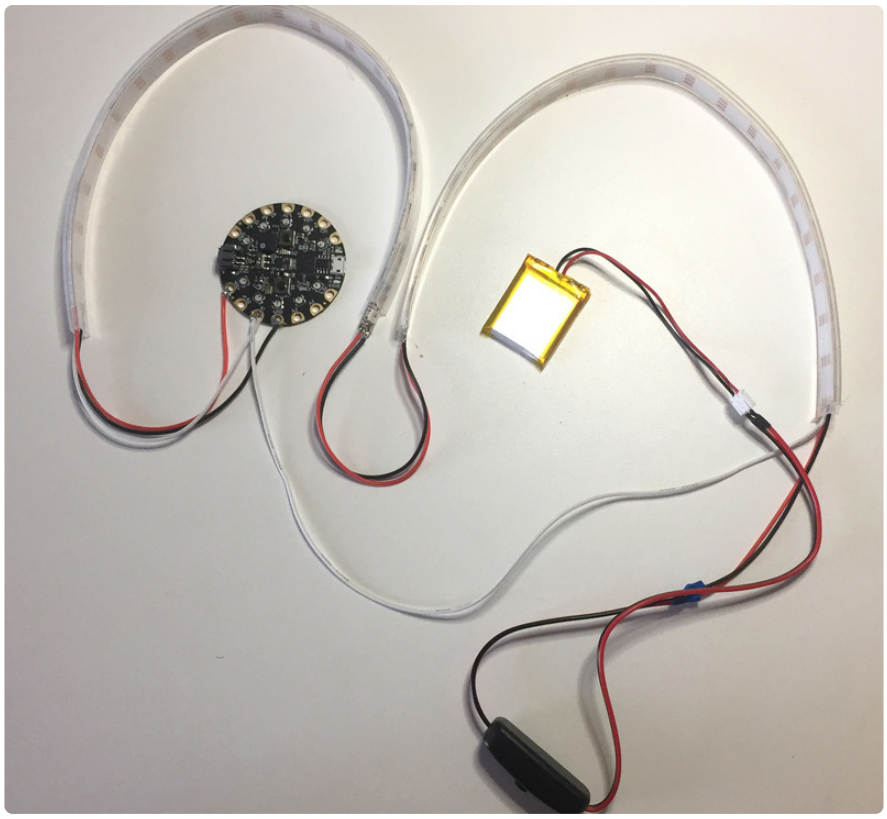
Power

Your battery will plug in to one end of the JST extension cable / switch. The leads from the other end of the switch will be soldered to the IN end of your second strip of neopixels (the one **NOT** used by the Circuit Playground).

The two neopixel strips will also have a power and ground wire running from the OUT of one to the OUT of the other.

Power will flow from the battery through the switch, and into the second Neopixel strand. It will continue out of that strip and into the first strip, then from there into the Circuit Playground. This will both protect your Circuit Playground from LEDs drawing too much voltage accidentally, and also balance the weight and bulk of the battery and the Circuit Playground between the two bunny ears.

Remember: Pixel data can only flow **one way,** from the IN end of the strip to the OUT end. Power can flow **either direction,** so it's fine to hook up your power supply to the opposite end of the strip from your microcontroller.

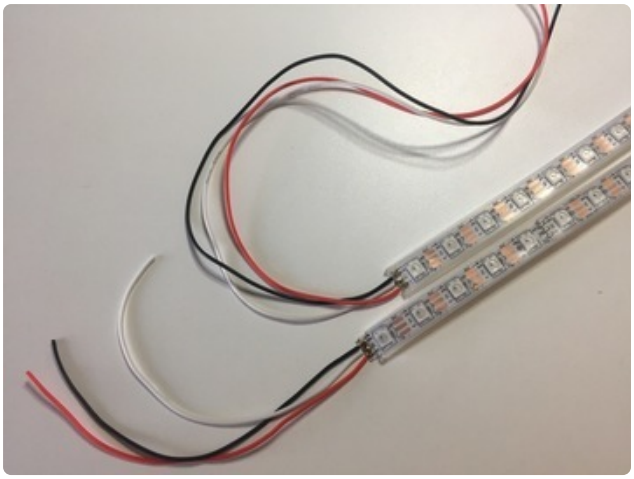


Assembly

Wire the Neopixels

Measure the distance around the edges your bunny ears and cut two neopixel strips to fit snugly inside. My strips ended up with 20 neopixels each.

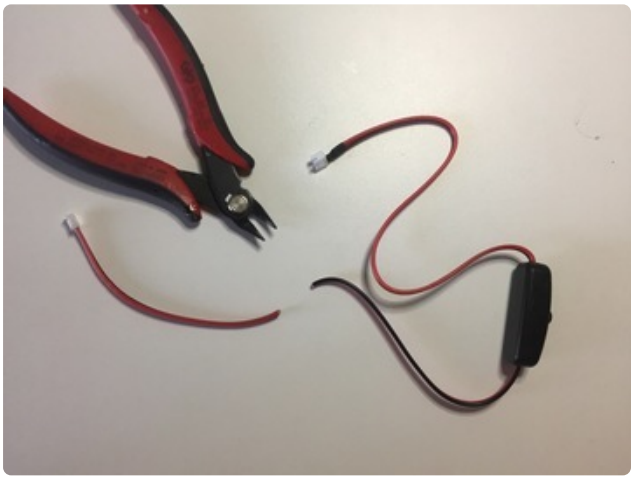
Cut carefully through the center of the copper pads, being sure to leave enough pad on each side for soldering.



Find the "in" end of each cut strip (the arrows should be facing AWAY from that end). Solder 3 color coded wires to each strip: red to 5v, black to G, and white to IN.

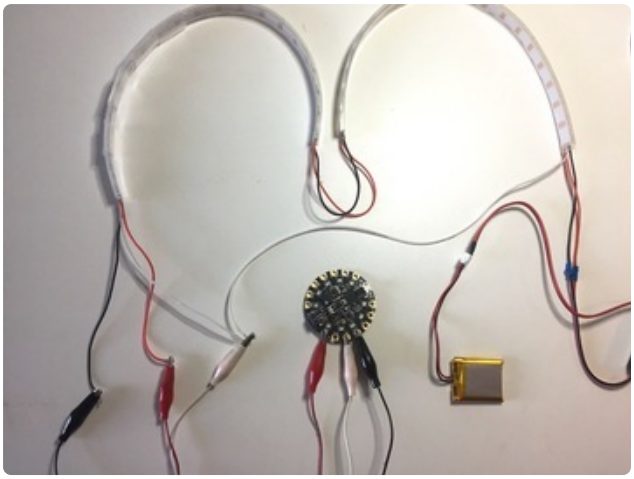
The wires should be about 5" long for one strip, and about 10" long for the other. We can trim them down later, so when guesstimating lengths, always err on the side of "too long".

We'll call the one with shorter wires strip #1, and the one with longer wires strip #2.

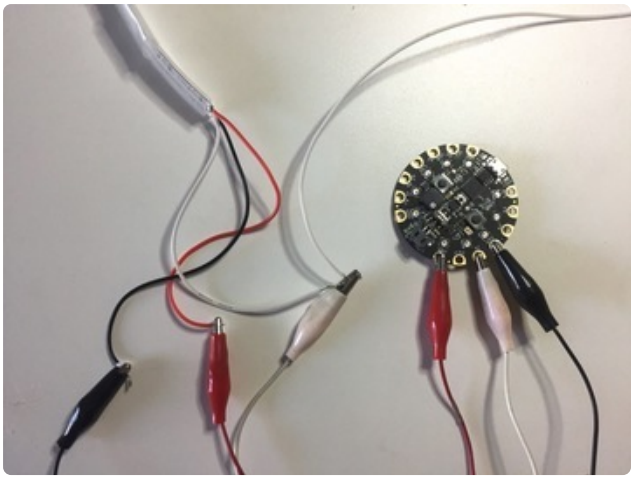


Look closely at your on/off switch. Figure out which end the battery wants to plug into: this is the male end and we'll leave that one alone. Find the other end (the female end) and snip the connector off with wire cutters.

Splice the long red and black wire attached to strip #2 to the red and black wire coming from the switch.



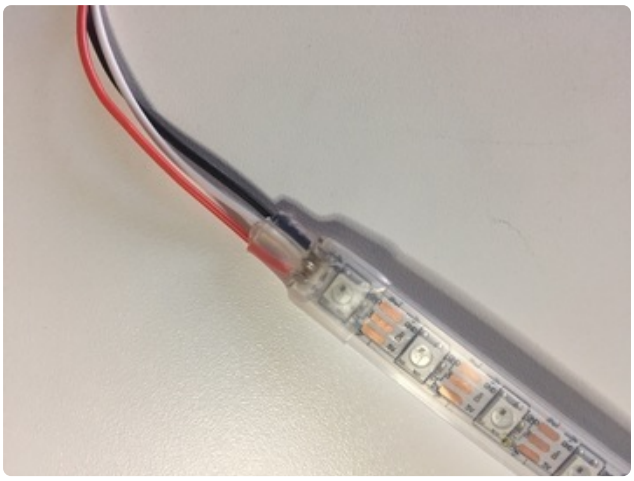
Solder a 4" red and black wire from the OUT end of strip #1 to the OUT end of strip #2.



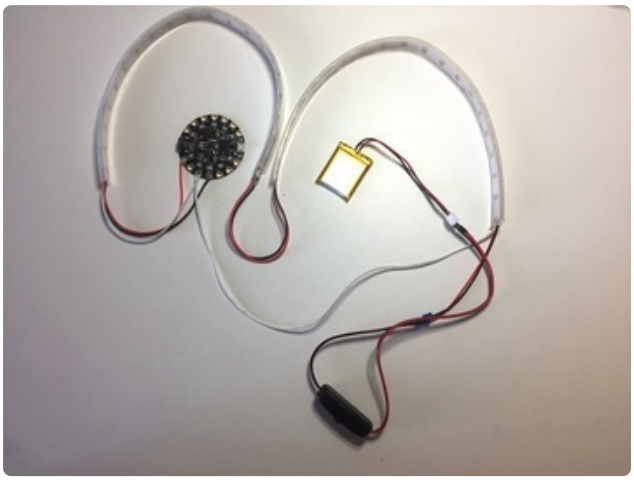
Test to be sure your connections are good by hooking up some alligator clips to the remaining free wires. Attach **BOTH** white wires to A1. Then attach the free red wire coming from strip #1 to VOUT and the free black wire to G on the Circuit Playground.

If all your connections are good, the strips will light up when you plug in your battery and tilt the Circuit Playground. Let's secure the ends of the strips so they don't ever pull out or break while you're wearing the ears.

Secure Strip Ends

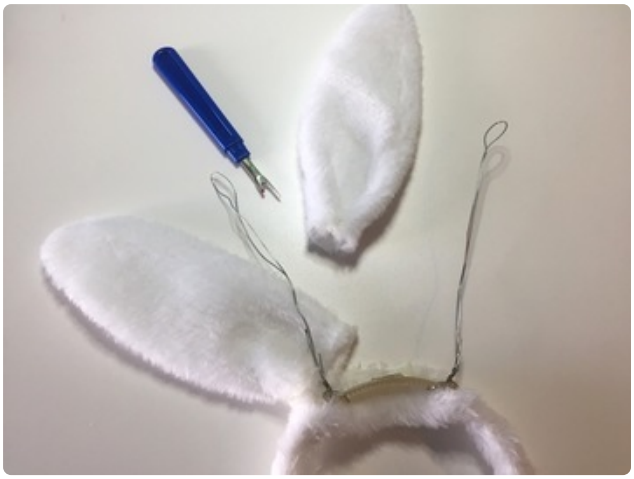


Cut four pieces of 1/4" clear heat shrink and place them over the end of each strip. Before you shrink them down, squirt some hot glue onto the strand to cover the solder connections. Then while the glue is still wet, use a heat gun to shrink the heat shrink down. This will create an indestructible and waterproof seal for your LED strips.



Take off your alligator clips and solder those four wires to the Circuit Playground's pads.

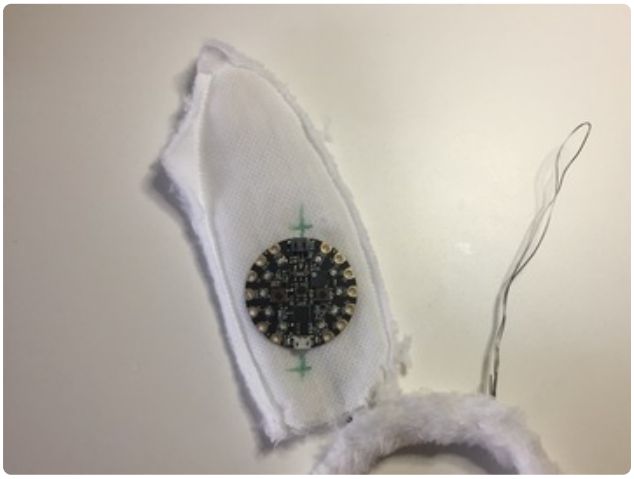
Install in the Ears



Use a thread ripper to carefully remove both ears from the headband.

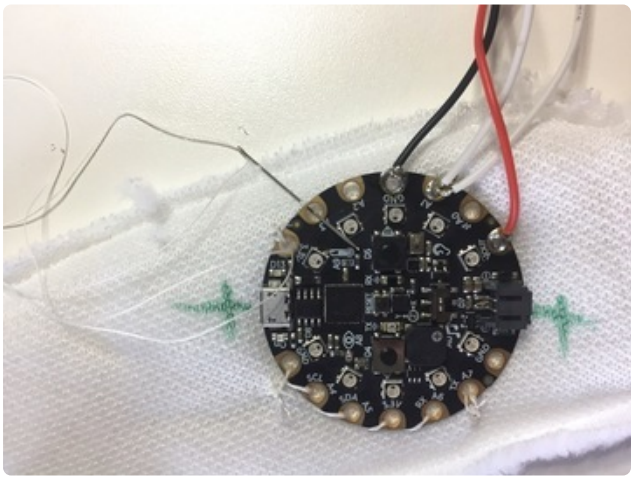
Fold the neopixel strip gently in half with the lights facing **outwards**. Slip the strips up inside the ears to be sure they fit.

Orient the ears so the animation runs from left to right when you tilt the board to the right, and vice versa. Think about how it's going to sit on your head and be sure it makes sense to you. It's easy to get the ears on backwards, so take a minute and be sure you've got things laid out right.

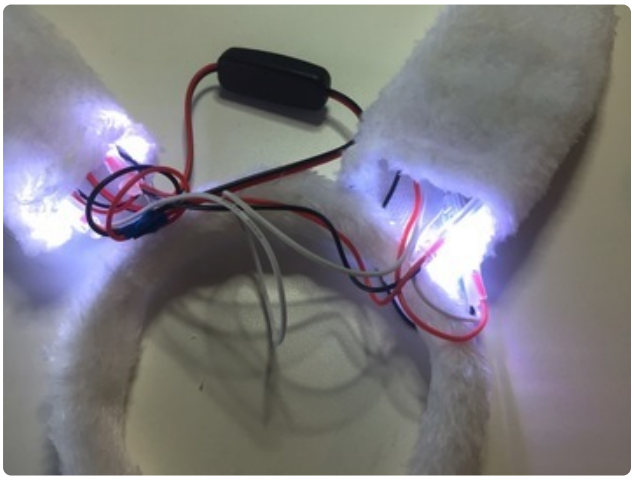


Once you're SURE you've got the layout right, remove the neopixel strand from one ear and turn the ear inside out.

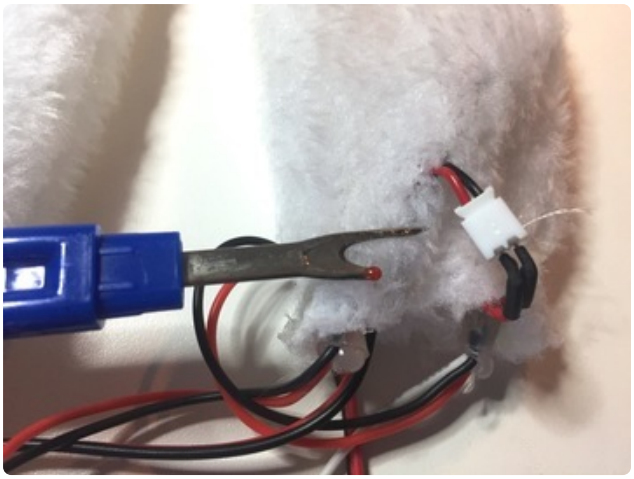
The Circuit Playground needs to be oriented carefully for the tilt sensors to work right. Mark the top and bottom with respect to the ground and sky while the ears are on. Don't just align it with the ear's top and bottom since the ear will be on your head at an angle.



Sew the Circuit Playground onto the **back** panel of the ear through the unused pads. You can also use your thread ripper to cut a small slit in the back of the ear to make the USB port accessible, in case you want to change or adjust the code later on.



Turn the ear right-side out again and replace the neopixel strip. Stuff the battery inside the opposite ear. Test your orientation again! The Circuit Playground should be on the **BACK** of one ear and the USB port should be perpendicular to the ground. Be sure the animation on both ears is running in the correct direction.



We'll need to be able to get to the battery for charging. Cut a small hole near the bottom of the battery-containing ear and slip the JST connector through before plugging it into the switch. This way it will be on the outside, so you can easily hook it up to the charger.

Place the switch on the back of the headband between the ears. Stuff the wires inside the ears and sew them to the headband, being sure you can still get to the JST connector.

Decorate your ears with flowers, ribbons, glitter, jewels, or the crushed skulls of your enemies.

The ears are a little heavier now than when you bought them. Use some hair clips or barrettes to help secure them to your head, so they don't tip off when you tilt to activate the animations.