Zelda Echoes Of Wisdom Tri Rod

Created by Ruiz Brothers

https://learn.adafruit.com/zelda-tri-rod

Last updated on 2024-10-09 01:46:07 PM EDT

Table of Contents

Overview

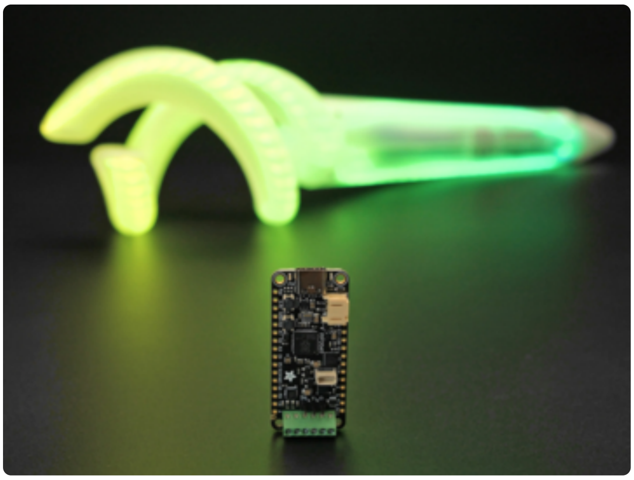
Build the Tri Rod from Zelda: Wisdom of Echoes!

This project features motion activated LED animations and sound effects powered by CircuitPython and the Prop-Maker Feather RP2040.

Motion Activated

When the accelerometer detects a swinging motion, it triggers the LEDs to animate a random sound effect.

The accelerometer detects when the prop is vertical, triggering an idle state where the LEDs fades slowly.



PropMaker RP2040

This dev-board is our favorite go-to that's designed for making prop projects. It's got the RP2040 chip with a built-in audio amplifier and accelerometer!

A 2000mAh lipo battery will provide several hours of use and can be recharged over USB.

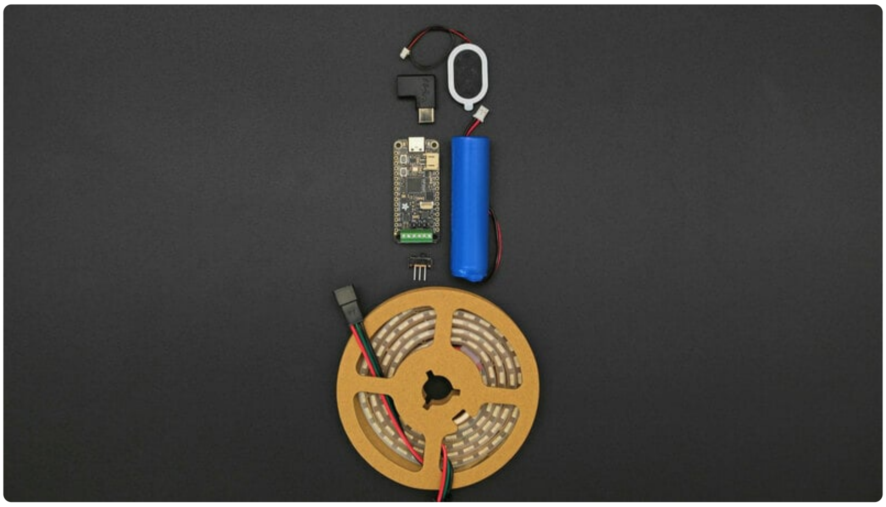
CircuitPython

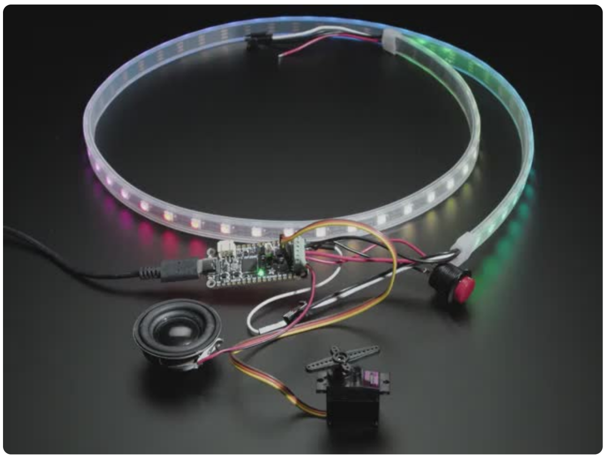
CircuitPython has made our Prop-Maker projects easier than ever thanks to our new demo code and libraries.

Development boards with CircuitPython are great for props because they load just like a USB flash drive so you can easily edit code and drag and drop audio files.

We think this could be modified to work great with similar props like a wizard's staff or a sci-fi space helmet.

Parts

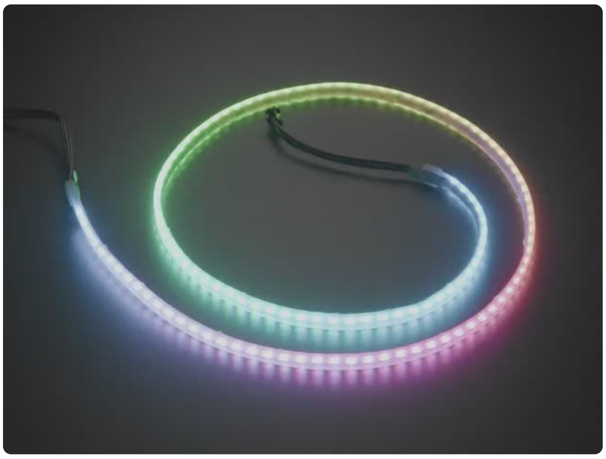




Adafruit RP2040 Prop-Maker Feather with I2S Audio Amplifier

The Adafruit Feather series gives you lots of options for a small, portable, rechargeable microcontroller board. By picking a feather and stacking on a FeatherWing you can create...

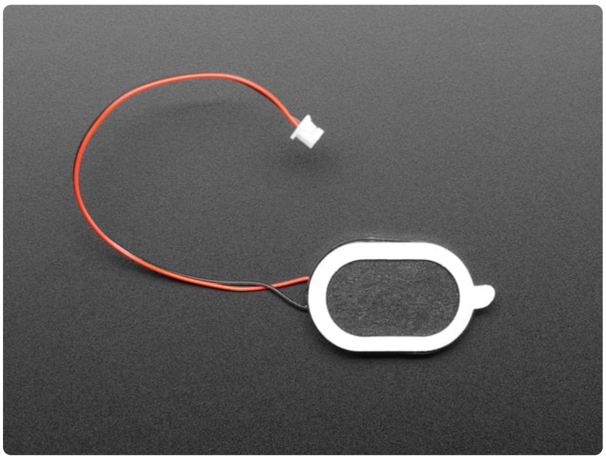
https://www.adafruit.com/product/5768



Adafruit Double-Sided Emitter NeoPixel LED Strip - 120 LEDs/m

"Prepare for trouble.. and make it double!" Fancy new dual-sided NeoPixel strips are a great alternative for folks who have loved and...

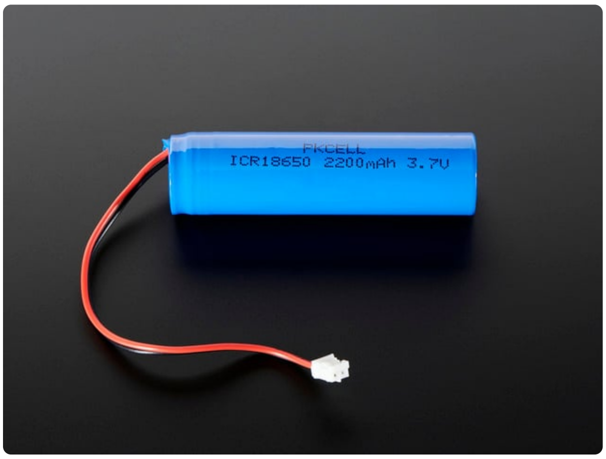
https://www.adafruit.com/product/5985



Mini Oval Speaker - 8 Ohm 1 Watt

Hear the good news! This wee speaker is a great addition to any audio project where you need 8 ohm impedance and 1W or less of power. We particularly like...

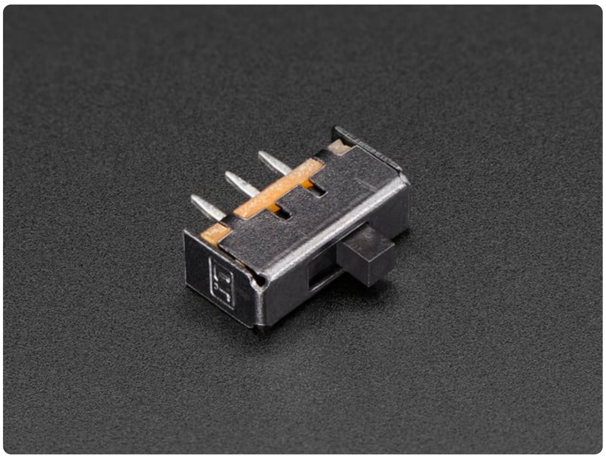
https://www.adafruit.com/product/3923



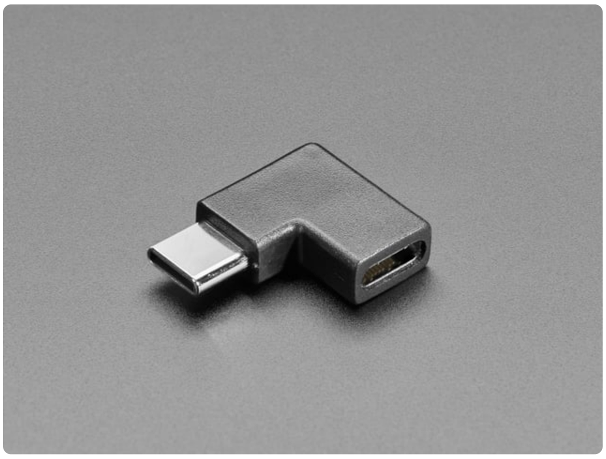
Lithium Ion Cylindrical Battery - 3.7v 2200mAh

Need a big battery for your project? This lithium-ion battery contains a 2200mAh and a protection circuit that provides over-voltage, under-voltage, and over-current protection. Yet,...

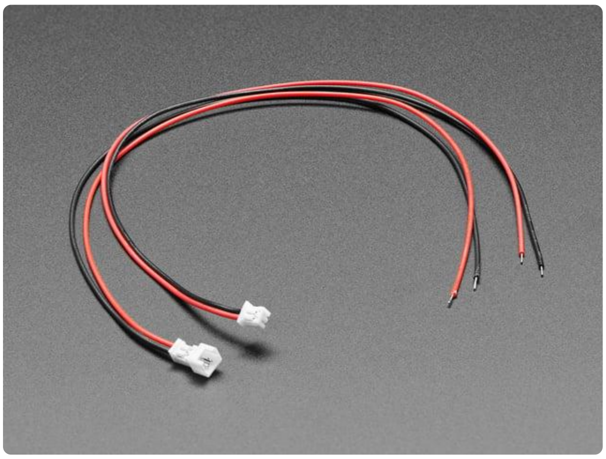
https://www.adafruit.com/product/1781



Breadboard-friendly SPDT Slide Switch
These nice switches are perfect for use with breadboard and perfboard projects. They have 0.1" spacing and snap in nicely into a solderless breadboard. They're easy to switch...
https://www.adafruit.com/product/805



Right Angle USB Type C Adapter - USB 3.1 Gen 4 Compatible
As technology changes and adapts, so does Adafruit, and speaking of adapting, this right angle adapter is USB C...
https://www.adafruit.com/product/4432



1.25mm Pitch 2-pin Cable Matching Pair - 40cm long
When 0.1" is too big, and JST PH's too chunky, these ultra-slim 1.25mm connectors are a reliable alternative. These are only 1.25mm pitch, but have a nice clicky...
https://www.adafruit.com/product/4720

Silicone Cover Stranded-Core Ribbon Cable - 10 Wire 1 Meter Long
For those who are fans of our silicone-covered wires, but are always looking to up their wiring game. We now have Silicone Cover Ribbon cables! These may look...
https://www.adafruit.com/product/3890

PLA Transparent White Filament

1 x PLA Transparent White Filament https://amzn.to/3ZSDzCB
PLA Transparent White Filament

2 x M2.5x6mm Screws https://amzn.to/3ZvDMvg
M2.5x6mm Screws

2 x M2x6mm Screws https://amzn.to/47AsYhn
M2x6mm Screws

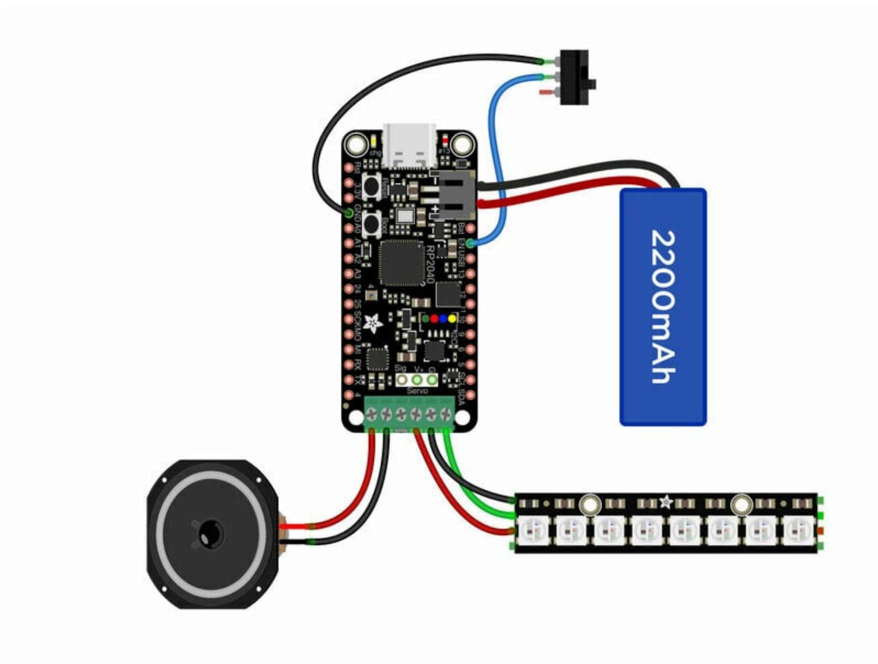
8 x M2x8mm Screws https://amzn.to/3K0Ag36
M2x8mm Screws

Circuit Diagram

The diagram below provides a general visual reference for wiring of the components once you get to the **Assembly** page. This diagram was created using the software package Fritzing (https://adafru.it/oEP).

Adafruit Library for Fritzing

Adafruit uses the Adafruit's Fritzing parts library to create circuit diagrams for projects. You can download the library or just grab individual parts. Get the library and parts from GitHub - Adafruit Fritzing Parts (https://adafru.it/AYZ).



Wired Connections

- The speaker is connected to the pins on the screw block terminal
- The NeoPixel Strip is connected to the pins on the screw block terminal
    - The Strip is has small text labels for each connection
    - Green is Signal (center of the strip)
    - Red is 5v
    - Black is GND
    - Read the [Basic Connections](https://adafru.it/ukF) (https://adafru.it/ukF) in the NeoPixel UberGuide
- The slide switch is connected to the **EN** and **GND** pins on the Feather
- The Lipo battery plugs into the Battery port

Code

Once you've finished setting up your Feather RP2040 Propmaker with CircuitPython, you can access the code and necessary libraries by downloading the Project Bundle.

To do this, click on the **Download Project Bundle** button in the window below. It will download to your computer as a zipped folder.

```
# SPDX-FileCopyrightText: 2023 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import random
import board
import audiocore
import audiobusio
import audiomixer
from digitalio import DigitalInOut, Direction
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff
import neopixel
import adafruit_lis3dh

# CUSTOMIZE SENSITIVITY HERE: smaller numbers = more sensitive to motion
SWING_1_THRESHOLD = 130
SWING_2_THRESHOLD = 170

pixel_time = 10
Y_THRESHOLD = -9.0
VERTICAL_TIME = 1.0  # Time in seconds the sensor needs to be vertical to trigger

fx_1 = audiocore.WaveFile(open("/sounds/swing1.wav", "rb"))
fx_2 = audiocore.WaveFile(open("/sounds/swing1b.wav", "rb"))
fx_3 = audiocore.WaveFile(open("/sounds/swing2.wav", "rb"))
fx_4 = audiocore.WaveFile(open("/sounds/swing3.wav", "rb"))

tracks = [fx_1, fx_2, fx_3, fx_4]
audio = audiobusio.I2SOut(board.I2S_BIT_CLOCK, board.I2S_WORD_SELECT,
board.I2S_DATA)
mixer = audiomixer.Mixer(voice_count=3, sample_rate=22050, channel_count=1,
                         bits_per_sample=16, samples_signed=True)
audio.play(mixer)

# external neopixels
num_pixels = 60
pixels = neopixel.NeoPixel(board.EXTERNAL_NEOPIXELS, num_pixels, auto_write=False)
pixels.brightness = 0.2

# Gradient colors (start and end)
pale_green = (50, 255, 50)
yellow = (255, 255, 0)
# Pulse parameters
pulse_speed = 0.01  # Speed of the pulse (lower is slower)
brightness_step = 5  # Amount by which to step the brightness

# onboard LIS3DH
i2c = board.I2C()
int1 = DigitalInOut(board.ACCELEROMETER_INTERRUPT)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c, int1=int1)
# Accelerometer Range (can be 2_G, 4_G, 8_G, 16_G)
lis3dh.range = adafruit_lis3dh.RANGE_2_G

# enable external power pin
# provides power to the external components
external_power = DigitalInOut(board.EXTERNAL_POWER)
external_power.direction = Direction.OUTPUT
external_power.value = True

mode = 0
brightness = 0
increasing = True
pixel_clock = ticks_ms()
# Variables to track time
vertical_start_time = None

while True:
```

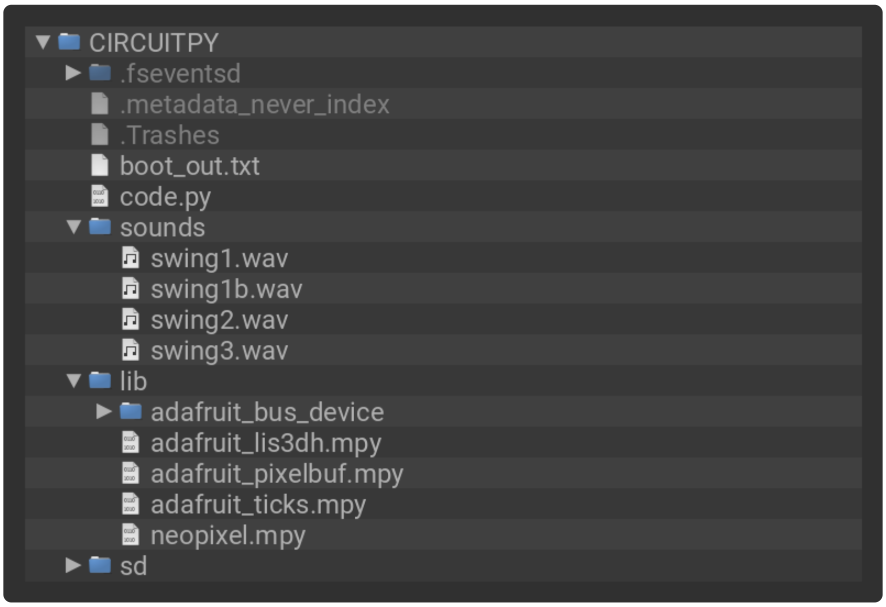
```
    # default
    if mode == 0:
        if ticks_diff(ticks_ms(), pixel_clock) >= pixel_time:
            for i in range(num_pixels):
                # Calculate gradient between pale_green and yellow
                ratio = i / (num_pixels - 1)
                r = int((yellow[0] - pale_green[0]) * ratio + pale_green[0])
                g = int((yellow[1] - pale_green[1]) * ratio + pale_green[1])
                b = int((yellow[2] - pale_green[2]) * ratio + pale_green[2])

                # Adjust brightness by scaling the RGB values
                scaled_r = int(r * (brightness / 255))
                scaled_g = int(g * (brightness / 255))
                scaled_b = int(b * (brightness / 255))

                pixels[i] = scaled_r, scaled_g, scaled_b
            pixels.show()
            # Adjust brightness up and down
            if increasing:
                brightness += brightness_step
                if brightness >= 255:
                    brightness = 255
                    increasing = False
            else:
                brightness -= brightness_step
                if brightness <= 0:
                    brightness = 0
                    increasing = True
            pixel_clock = ticks_add(pixel_clock, pixel_time)
        x, y, z = lis3dh.acceleration
        #print(x, y, z)
        if abs(y) > Y_THRESHOLD and abs(x) < 2.0 and abs(z) < 2.0:
            if vertical_start_time is None:
                vertical_start_time = time.monotonic()  # Start timing
            elif time.monotonic() - vertical_start_time >= VERTICAL_TIME:
                print("vertical")
                mode = "vertical"
        else:
            vertical_start_time = None  # Reset if the sensor is not vertical
        accel_total = x * x + z * z
        if accel_total >= SWING_2_THRESHOLD:
            print("swing 2")
            mode = "swing 2"
        elif accel_total >= SWING_1_THRESHOLD:
            print("swing 1")
            mode = "swing 1"
    elif mode == "swing 1":
        mixer.voice[0].play(tracks[random.randint(0, 1)])
        while mixer.voice[0].playing:
            pixels.fill(pale_green)
            pixels.show()
        mode = 0
    # clash or move
    elif mode == "swing 2":
        mixer.voice[1].play(tracks[2])
        while mixer.voice[1].playing:
            pixels.fill(yellow)
            pixels.show()
        mode = 0
    elif mode == "vertical":
        mixer.voice[2].play(tracks[3])
        while mixer.voice[2].playing:
            pixels.fill(yellow)
            pixels.show()
        vertical_start_time = None
        mode = 0
```

Upload the Code, Sound Effects and Libraries to the Feather RP2040 Propmaker

After downloading the Project Bundle, plug your Feather RP2040 Propmaker into the computer's USB port with a known good USB data+power cable. You should see a new flash drive appear in the computer's File Explorer or Finder (depending on your operating system) called **CIRCUITPY**. Unzip the folder and copy the following items to the Feather RP2040 Propmaker's **CIRCUITPY** drive.

- **lib** folder
- **code.py**
- **sounds** folder

Your Feather RP2040 Propmaker **CIRCUITPY** drive should look like this after copying the **lib** folder, **sounds** folder and the **code.py** file.



How the CircuitPython Code Works

At the top of the code are a few customizable variables for the prop. `SWING_1_THRESHOLD` and `SWING_2_THRESHOLD` affect the swing sensitivities for the two swing detections. `pixel_time` is the delay in milliseconds for the NeoPixel fade animation. `Y_THRESHOLD` is the Y axis value from the accelerometer to detect if the prop is vertical. Finally, `VERTICAL_TIME` is the number of seconds the prop needs to be vertical to trigger the vertical mode in the loop.

```
# CUSTOMIZE SENSITIVITY HERE: smaller numbers = more sensitive to motion
SWING_1_THRESHOLD = 130
SWING_2_THRESHOLD = 170

pixel_time = 10
```

```
Y_THRESHOLD = -9.0
VERTICAL_TIME = 1.0  # Time in seconds the sensor needs to be vertical to trigger
```

Audio

Four sound effects are loaded from the **sounds** folder. These sound effects are added to the `tracks` array. The onboard I2S amplifier is used for audio output and a `Mixer` is used to play and control the sound effects.

```
fx_1 = audiocore.WaveFile(open("/sounds/swing1.wav", "rb"))
fx_2 = audiocore.WaveFile(open("/sounds/swing1b.wav", "rb"))
fx_3 = audiocore.WaveFile(open("/sounds/swing2.wav", "rb"))
fx_4 = audiocore.WaveFile(open("/sounds/swing3.wav", "rb"))

tracks = [fx_1, fx_2, fx_3, fx_4]
audio = audiobusio.I2SOut(board.I2S_BIT_CLOCK, board.I2S_WORD_SELECT,
board.I2S_DATA)
mixer = audiomixer.Mixer(voice_count=3, sample_rate=22050, channel_count=1,
                         bits_per_sample=16, samples_signed=True)
audio.play(mixer)
```

NeoPixels

60 NeoPixels are used in the prop. Variables for the gradient fade effect are setup after the NeoPixels are instantiated.

```
# external neopixels
num_pixels = 60
pixels = neopixel.NeoPixel(board.EXTERNAL_NEOPIXELS, num_pixels, auto_write=False)
pixels.brightness = 0.2

# Gradient colors (start and end)
pale_green = (50, 255, 50)
yellow = (255, 255, 0)
# Pulse parameters
pulse_speed = 0.01  # Speed of the pulse (lower is slower)
brightness_step = 5  # Amount by which to step the brightness
```

I2C and External Power

The onboard LIS3DH accelerometer is instantiated over I2C. The external power pin is setup as an output and is set to `True` so that all of the external components (NeoPixels and I2S amp) receive power.

```
# onboard LIS3DH
i2c = board.I2C()
int1 = DigitalInOut(board.ACCELEROMETER_INTERRUPT)
lis3dh = adafruit_lis3dh.LIS3DH_I2C(i2c, int1=int1)
# Accelerometer Range (can be 2_G, 4_G, 8_G, 16_G)
lis3dh.range = adafruit_lis3dh.RANGE_2_G

# enable external power pin
# provides power to the external components
external_power = DigitalInOut(board.EXTERNAL_POWER)
external_power.direction = Direction.OUTPUT
external_power.value = True
```

Preflight Variables

Right before the loop, a few more variables are created. `mode` is used to track the mode of the prop. `brightness`, `increasing` and `pixel_clock` are used for the NeoPixel effect. `vertical_start_time` is used to track how long the prop is held vertically.

```
mode = 0
brightness = 0
increasing = True
pixel_clock = ticks_ms()
# Variables to track time
vertical_start_time = None
```

The Loop

In the loop, mode `0` acts as the default mode. The NeoPixels show a gradient that blends from light green to yellow. This gradient pulses in brightness by adjusting the color values. `ticks` are used to run this animation without a blocking delay.

```
if mode == 0:
        if ticks_diff(ticks_ms(), pixel_clock) &gt;= pixel_time:
            for i in range(num_pixels):
                # Calculate gradient between pale_green and yellow
                ratio = i / (num_pixels - 1)
                r = int((yellow[0] - pale_green[0]) * ratio + pale_green[0])
                g = int((yellow[1] - pale_green[1]) * ratio + pale_green[1])
                b = int((yellow[2] - pale_green[2]) * ratio + pale_green[2])

                # Adjust brightness by scaling the RGB values
                scaled_r = int(r * (brightness / 255))
                scaled_g = int(g * (brightness / 255))
                scaled_b = int(b * (brightness / 255))

                pixels[i] = scaled_r, scaled_g, scaled_b
            pixels.show()
            # Adjust brightness up and down
            if increasing:
                brightness += brightness_step
                if brightness &gt;= 255:
                    brightness = 255
                    increasing = False
            else:
                brightness -= brightness_step
                if brightness &lt;= 0:
                    brightness = 0
                    increasing = True
            pixel_clock = ticks_add(pixel_clock, pixel_time)
```

The accelerometer is read in this mode as well. If the code detects that the prop is being held vertically for longer than the `VERTICAL_TIME` variable, then mode is set to `vertical`. The force of the prop swing is also checked. If it is greater than either of the swing threshold values, then the modes are set to either `swing 2` or `swing 1`.

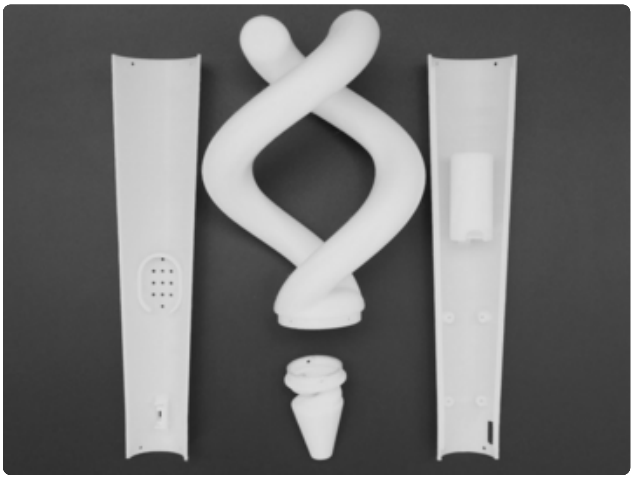
```
x, y, z = lis3dh.acceleration
#print(x, y, z)
if abs(y) &gt; Y_THRESHOLD and abs(x) &lt; 2.0 and abs(z) &lt; 2.0:
```

```
    if vertical_start_time is None:
        vertical_start_time = time.monotonic()  # Start timing
    elif time.monotonic() - vertical_start_time &gt;= VERTICAL_TIME:
        print("vertical")
        mode = "vertical"
else:
    vertical_start_time = None  # Reset if the sensor is not vertical
accel_total = x * x + z * z
if accel_total &gt;= SWING_2_THRESHOLD:
    print("swing 2")
    mode = "swing 2"
elif accel_total &gt;= SWING_1_THRESHOLD:
    print("swing 1")
    mode = "swing 1"
```

The three other modes trigger different effects for the prop. `swing 1` and `swing 2` modes play a specific sound effect and light up the prop either green or yellow. The `vertical` mode plays a longer sound effect and lights up the pixels yellow.

```
elif mode == "swing 1":
    mixer.voice[0].play(tracks[random.randint(0, 1)])
    while mixer.voice[0].playing:
        pixels.fill(pale_green)
        pixels.show()
    mode = 0
elif mode == "swing 2":
    mixer.voice[1].play(tracks[2])
    while mixer.voice[1].playing:
        pixels.fill(yellow)
        pixels.show()
    mode = 0
elif mode == "vertical":
    mixer.voice[2].play(tracks[3])
    while mixer.voice[2].playing:
        pixels.fill(yellow)
        pixels.show()
    vertical_start_time = None
    mode = 0
```

3D Printing



3D Printed Parts

STL files for 3D printing will need to be oriented for print using either FDM or SLS machines.

Parts were built with Translucent Clear PLA filament.

Original design source files may be downloaded using the links below.

Use Clear PLA Transparent White Filament (https://adafru.it/1a8m) to get the translucent effect

Edit Design

https://adafru.it/1a8j

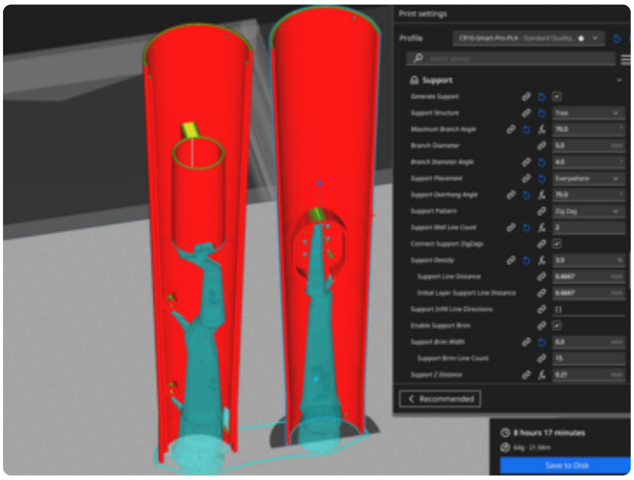
Download STEP file

https://adafru.it/1a8k

Download STLs

https://adafru.it/1a8n

Download STLs from Thingiverse



Slice with Settings for PLA material

The parts were sliced using CURA using the slice settings below.

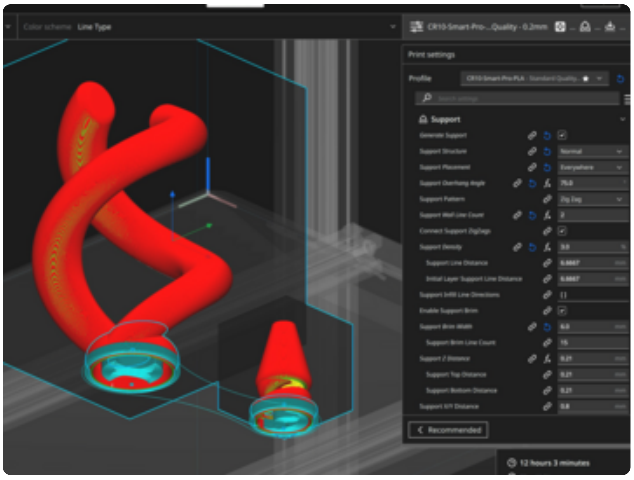
PLA filament 200c extruder

0.2 layer height

10% gyroid infill

60mm/s print speed

60c heated bed



Supports

For the handle: Tree Supports

For the Coil and End: Normal Supports

Placement: Everywhere

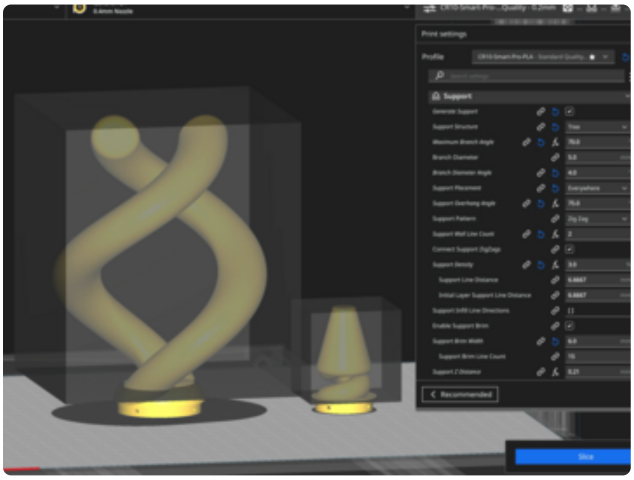
Support Overhang Angle: 70

Support Destiny: 10%

Enable Support Interface Off

Support Z Distance: .21

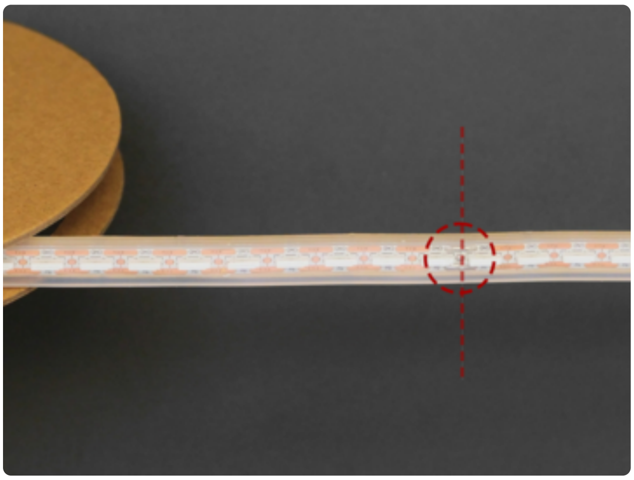
Brim: 6



Support Blockers

Use support blockers for the tops of the Coil and end parts. Allow the supports to attach the bottom overhangs to both parts.

Assembly

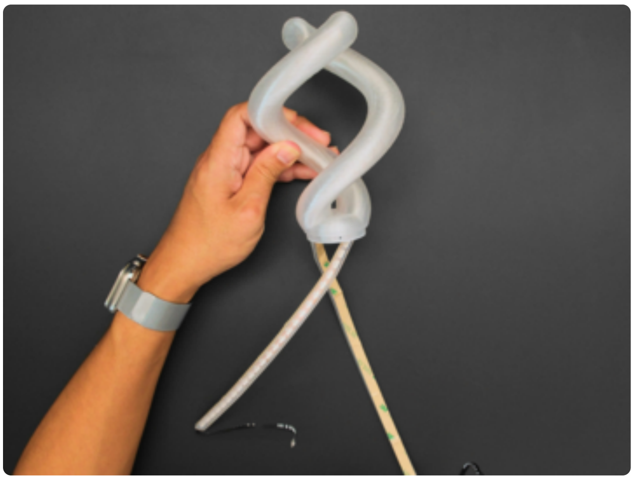


Prep LED Strip

Cut the LED strip to 60 pixels long, half of the 120 pixel strip. Measure and cut wires to solder to the pads on the two strips. Leave the silicone sheathing over the pads to protect the connections.

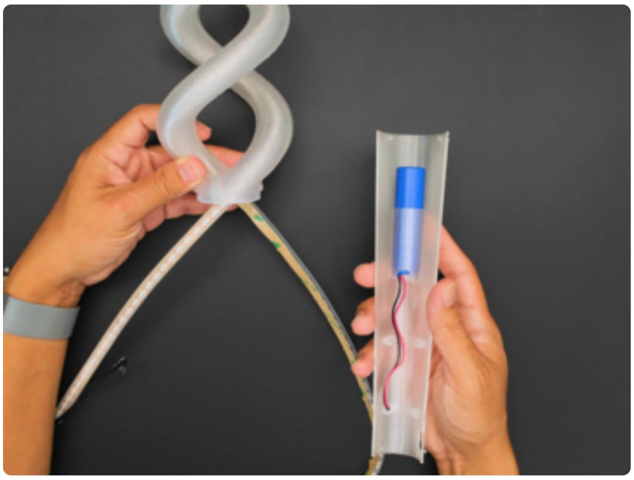
Pull the silicone sheathing back over the pads after wires are soldered.

The wires can be marked with oil based white marker (https://adafru.it/1a8l) to distinguish connections.



Fit Strip in coil

Thread the two LED strips into the Coil part.

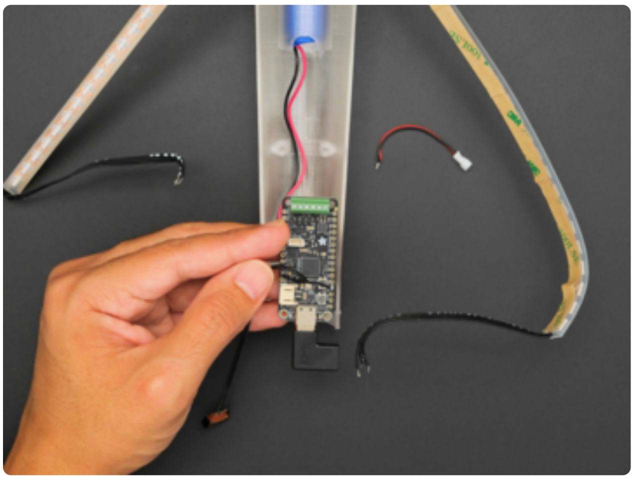


Place Lipo battery

Place the Lipo Battery with the wires first into the battery holder on the handle.

Attach Handle

Use M2x8mm screws to connect the Coil part.

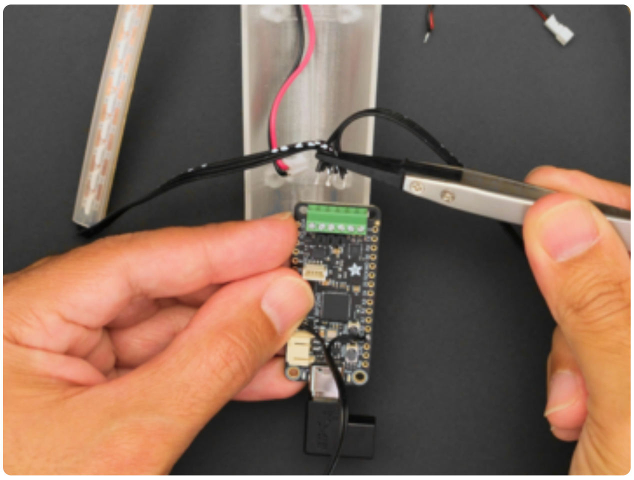


Solder Slide Switch

Measure wires for the slide switch to reach were the Feather board mounts to other the handle, about 120mm.

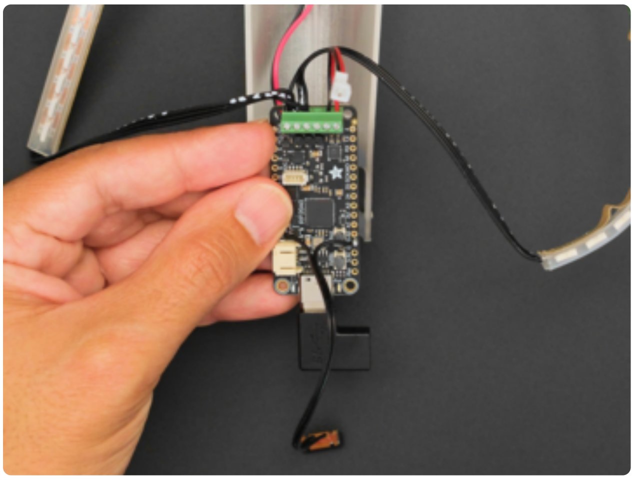
Attach Right Angle USB C Dongle

Plug in the right angle USB C dongle facing the cut out on the handle.



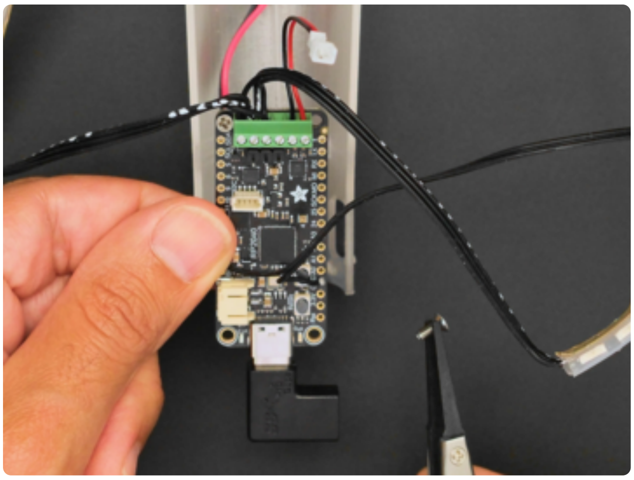
Terminal Wires

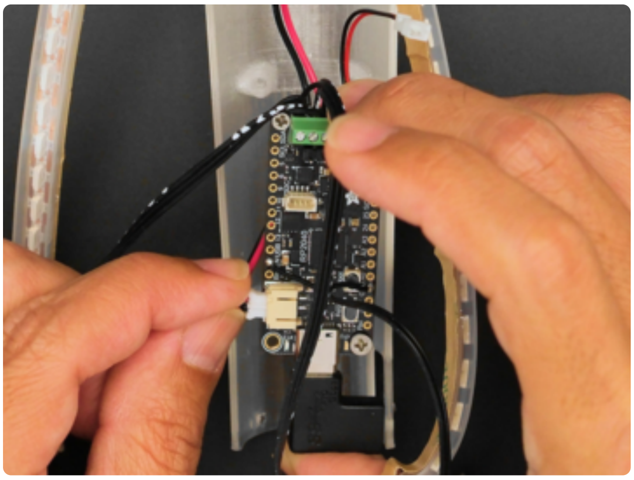
Match and pair the two LED wire connections together to fit into the terminal block.



Speaker Socket Wire

Measure and cut a 60mm long Pico Blade Socket wire to easily connect the Oval Speaker, into the terminal block.



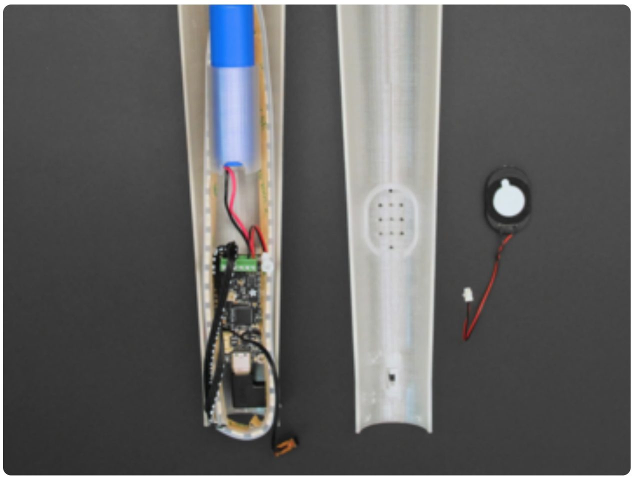


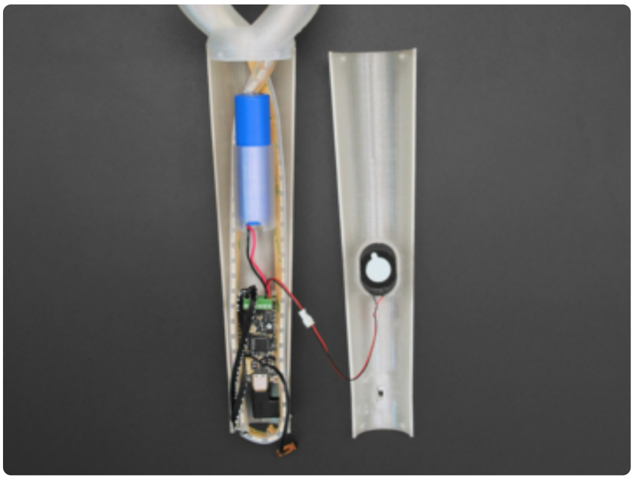
Place board screws

Add screws to mount the Feather board to the standoff on the handle. Place the Lipo Battery wire under the board before fastening to the standoffs.

Plug Lipo Battery

Plug in the Lipo Battery to the Feather before fastening the screws to the standoffs on the handle.





Mount Feather

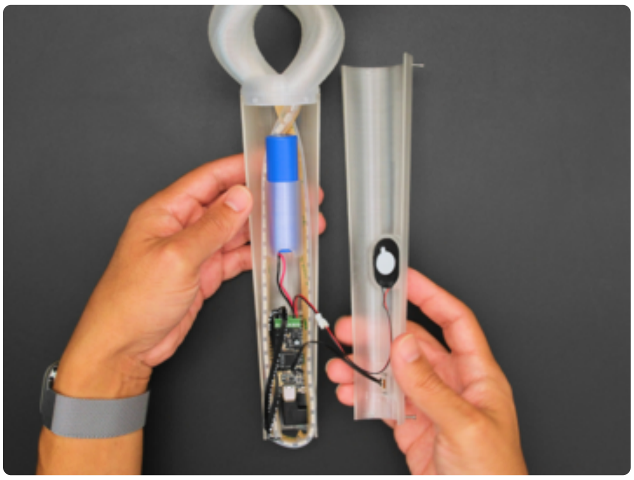
Fasten screws to the handle.

Arrange LED strip

Tuck the LED strips under the Lipo Battery holder on the handle.

Press Fit the Speaker

Align the speaker wire to the cutout on the cavity and press fit into the handle.

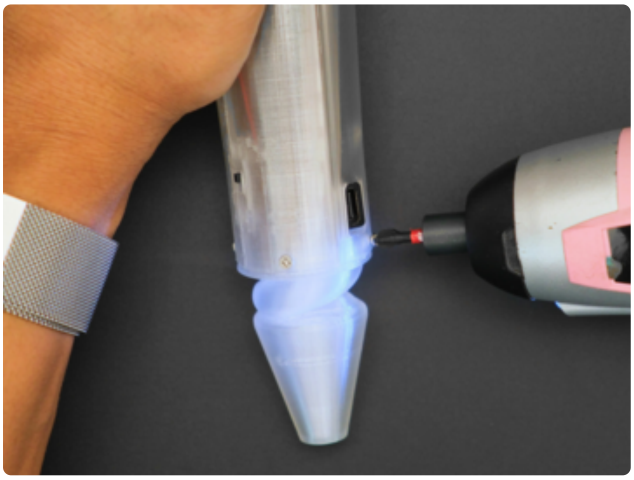


Press Fit the Slide Switch

Angle the slide switch and press fit into the cutout on the handle.

Align handle

Place the handle over the Coil part and fasten with four M2x8mm long screws.



Attach Bottom

Align the Bottom part to the handle and fasten with four more M2x8mm screws.

Complete