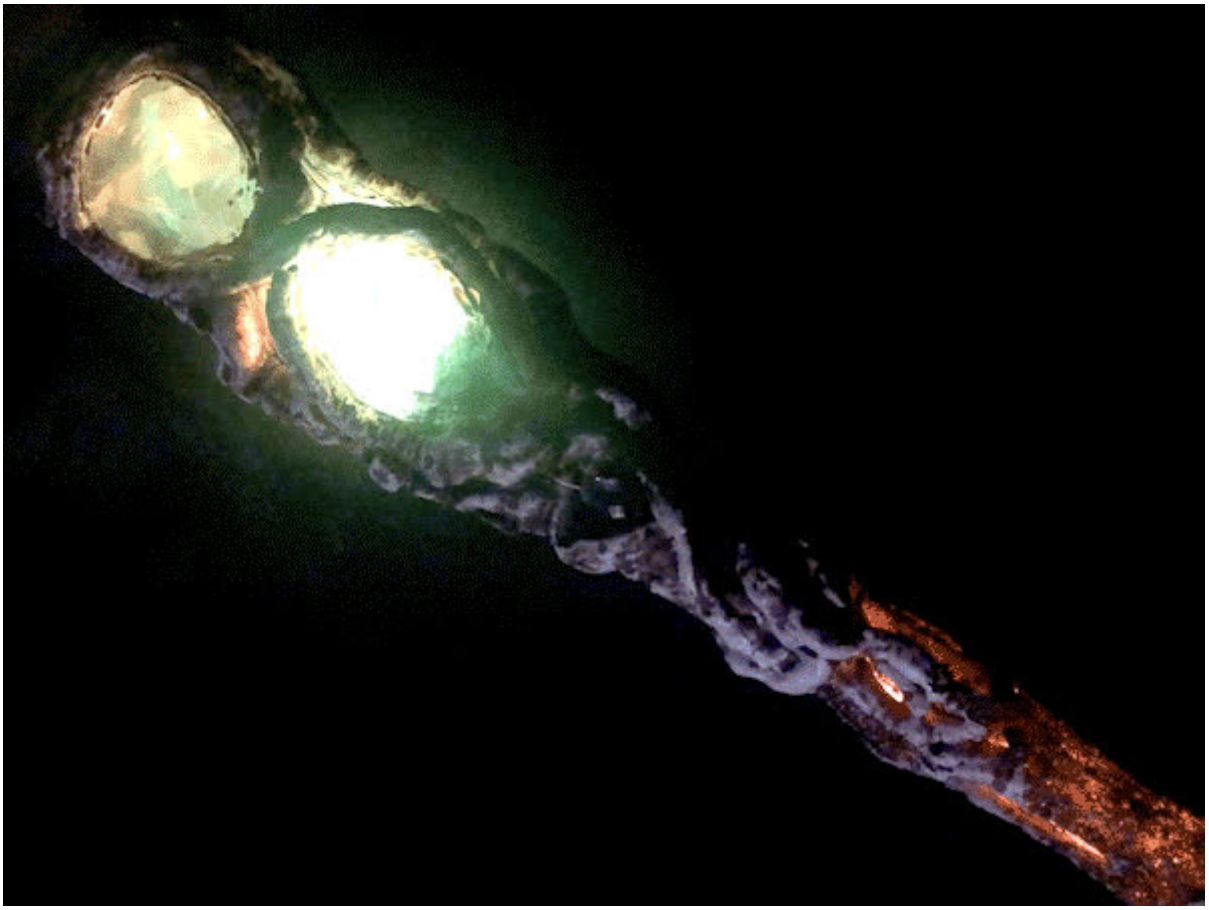
Burning Fire Wizard Staff

Created by Erin St Blaine



https://learn.adafruit.com/burning-fire-wizard-staff

Last updated on 2024-06-03 03:04:44 PM EDT

Table of Contents

Overview

> "I am a servant of the Secret Fire, wielder of the flame of Anor. You cannot pass. The dark fire will not avail you, flame of Udûn. Go back to the Shadow! You cannot pass."
> — J.R.R. Tolkien

Complete your Wizard or Druid cosplay with a glowing, ever-burning fire staff. This prop is lightweight and sturdy, and made from inexpensive materials. You don't need much in the way of sculpting ability to create something organic and magical, that looks as though it was cut from the Yggdrasil with an enchanted axe and then imbued with fires from the Mines of Moria before serving the White Witch Jadis for a century or two and then falling into your hands at the end of a rollicking adventure.

You'll need some soldering ability, a little creativity, and a willingness to get a bit messy. You can upload our CircuitPython code as-is, or it's easy to change the colors and sound effects to complement your own Wizard or Barbarian character. This would also make a wonderful staff for a monster or Fairy Queen with a few stylistic variations.

The cost for this project the way I did it comes in at around $125 including all electronics and craft materials. Nice!

Adafruit Products Needed

1 x Feather M4 Express — https://www.adafruit.com/product/3857
Adafruit Feather M4 Express Microcontroller

---

1 x PropMaker FeatherWing — https://www.adafruit.com/product/4145
PropMaker Wing for the Feather M4

---

2 x NeoPixel Strip — https://www.adafruit.com/product/1460
2 meters of 30/m NeoPixel Strip

---

1 x NeoPixel Ring — https://www.adafruit.com/product/1643
12 Pixel NeoPixel Ring

---

1 x Mini Speaker — https://www.adafruit.com/product/3923
Mini Oval Speaker

---

1 x On/Off Switch — https://www.adafruit.com/product/1092
Tactile On/Off Switch with Leads

---

1 x 3-Pin JST Connector — https://www.adafruit.com/product/4336
Connector for NeoPixels

---

1 x Wire — https://www.adafruit.com/product/3892
Silicone Stranded Wire Ribbon Cable

2200mAh Lithium Ion Battery

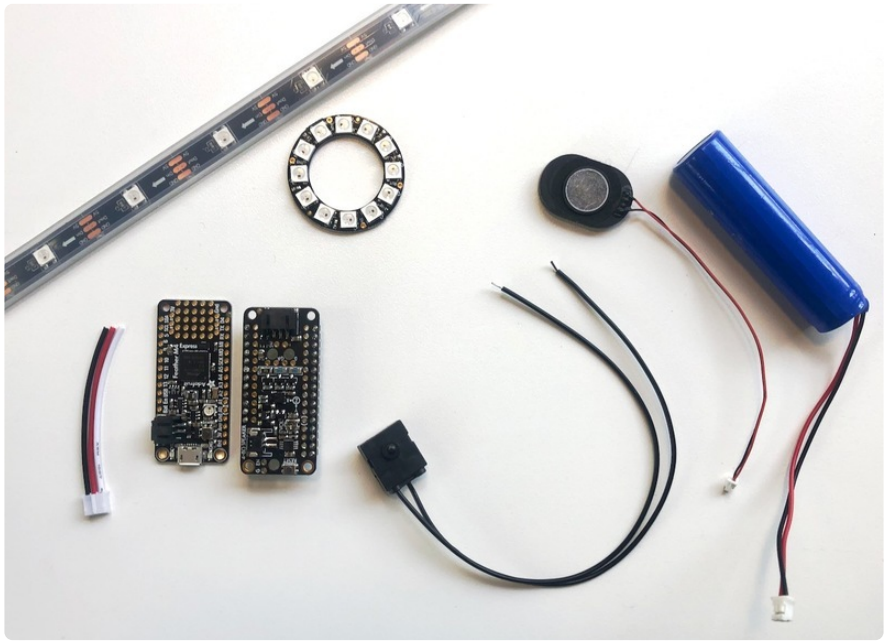
1 x Battery — https://www.adafruit.com/product/1781

2200mAh Lithium Ion Battery

---

1 x USB Cable — https://www.adafruit.com/product/4148

For charging & uploading code



Additional Materials

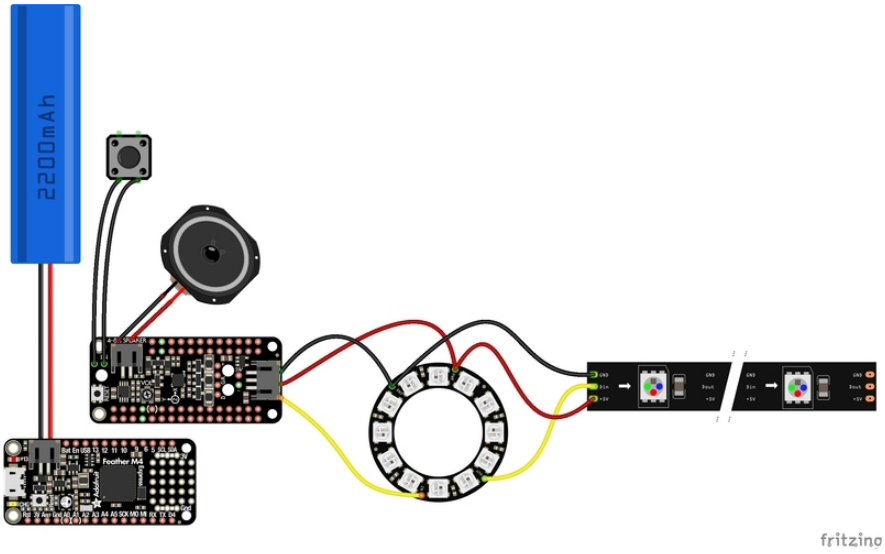
- Clear 1" polycarbonate tube - around 1.5-2m long and at least 7/8" internal diameter, with end cap (I love the ones from FlowToys (https://adafru.it/JCS))
- Plastic water bottle for the top of the staff (I used one with a pretty shape from the grocery store, made by Eternal Water (https://adafru.it/JF2))
- Great Stuff Expanding Foam (https://adafru.it/MD3)
- Latex paint in black, brown and light grey
- Clear iridescent cellophane wrap scraps
- Jewel or jewelry finding to decorate the on/off switch

Tools Needed

- Soldering iron & accessories
- Latex gloves
- Plastic drop cloth sheeting
- Small keyhole saw or serrated knife
- Paintbrush & paint mixing cups
- Utility / X-acto knife
- Hot glue gun & glue sticks

- 99% alcohol (https://adafru.it/MD4) for paint clean-up

Wiring Diagram



We'll use the headers that come soldered to the assembled PropMaker Wing to mount it directly on top of the Feather M4 Express, to make the smallest possible package to hide inside the staff.

The on/off switch will be soldered into the **GND** and **ENABLE** pins near the reset button on the PropMaker Wing.

The speaker, battery, and NeoPixel ring will plug into their respective ports on the PropMaker Wing using JST connectors.

The NeoPixel ring will be soldered to the 3-pin JST connector as follows:

- Red wire --> **PWR**
- Black wire --> **GND**
- White wire --> **IN**

We'll also connect the NeoPixel ring to the NeoPixel strip as follows:

- **PWR --> 5V+**
- **GND --> GND**
- **OUT --> IN**

Software

Install CircuitPython

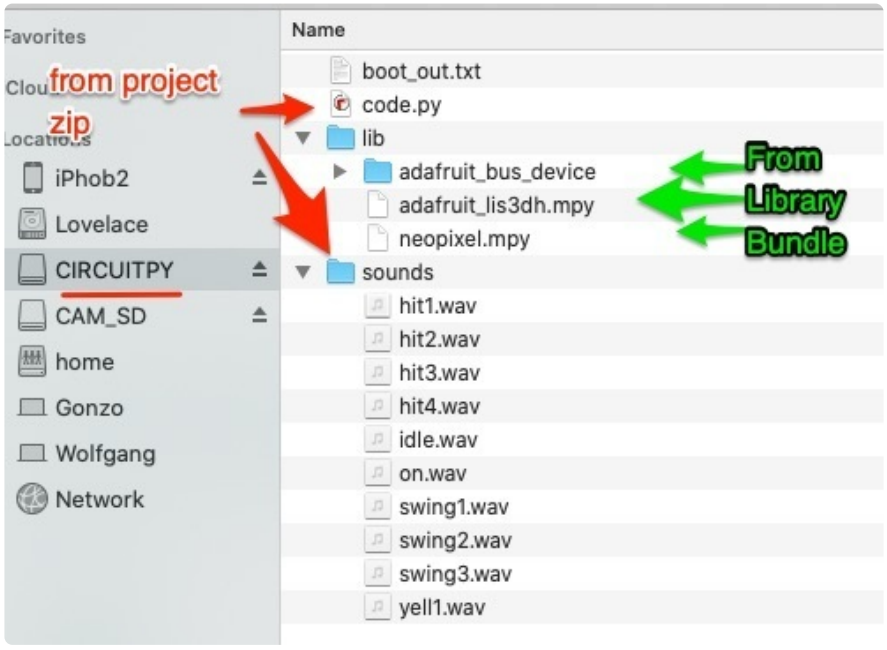
The Adafruit Feather M4 ships with CircuitPython, but let's go ahead and update it to the latest version. It's super easy with the **circuitpython.org** website. Follow the directions on the Feather M4 Express guide (https://adafru.it/CVs).

Adafruit Circuit Python Libraries

Download the CircuitPython library bundle per the Feather M4 guide instructions here (https://adafru.it/HDo). Unzip the files into a folder on your computer. Create a new folder on the **CIRCUITPY** drive and name it **lib**. The following libraries are required to run the code properly.

- **adafruit_bus_device** (directory)
- **adafruit_lis3dh.mpy**
- **neopixel.mpy**

Find all these files in the library bundle and copy them into the **lib** file you just made on your **CIRCUITPY** drive.



Upload Files

Click the link below to **download the project zip** – This contains the code and audio files. Upload the **code.py** file to the **CIRCUITPY** drive root (main) folder.

Create a new folder at the top level of the **CIRCUITPY** drive and name it **sounds**. Upload the audio files to that folder. The code will run properly when all of the files have been uploaded.

Check out the image above to see what your **CIRCUITPY** drive should look like when all the files are in place.

```
# SPDX-FileCopyrightText: 2019 Kattni Rembor Adafruit Industries
# SPDX-FileCopyrightText: 2019 Erin St Blaine for Adafruit Industries
# SPDX-FileCopyrightText: 2019 Limor Fried for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Prop-Maker based Burning Wizard Staff
Adafruit invests time and resources providing this open source code.
Please support Adafruit and open source hardware by purchasing
products from Adafruit!
Written by Kattni Rembor, Erin St Blaine & Limor Fried for Adafruit Industries
Copyright (c) 2020 Adafruit Industries
Licensed under the MIT license.
All text above must be included in any redistribution.
"""

import time
import random
import digitalio
import audioio
import audiocore
import board
import neopixel
import adafruit_lis3dh

# CHANGE TO MATCH YOUR RING AND STRIP SETUP
NUM_RING = 12   #12 pixel ring
NUM_STRIP = 44  # 44 pixels in my NeoPixel strip
NUM_PIXELS = NUM_STRIP + NUM_RING  #total number of pixels

NEOPIXEL_PIN = board.D5  # PropMaker Wing uses D5 for NeoPixel plug
POWER_PIN = board.D10


# CUSTOMISE COLORS HERE:
COLOR = (200, 30, 0)      # Default idle is orange
ALT_COLOR = (0, 200, 200)  # hit color is teal
SWING_COLOR = (200, 200, 200) #swing animation color is white
TOP_COLOR = (100, 100, 0)  #top color is yellow-green
YELL_COLOR = (200, 0, 200)  #yell color is purple

# CUSTOMISE IDLE PULSE SPEED HERE: 0 is fast, above 0 slows down
IDLE_PULSE_SPEED = 0  # Default is 0 seconds
SWING_BLAST_SPEED = 0.007

# CUSTOMISE BRIGHTNESS HERE: must be a number between 0 and 1
IDLE_PULSE_BRIGHTNESS_MIN = 0.2  # Default minimum idle pulse brightness
IDLE_PULSE_BRIGHTNESS_MAX = 1  # Default maximum idle pulse brightness

# CUSTOMISE SENSITIVITY HERE: smaller numbers = more sensitive to motion
HIT_THRESHOLD = 1150
SWING_THRESHOLD = 800
YELL_THRESHOLD = 700

# Set to the length in seconds of the "on.wav" and "yell1.wav" files
POWER_ON_SOUND_DURATION = 3.0
YELL_SOUND_DURATION = 1.0


enable = digitalio.DigitalInOut(POWER_PIN)
enable.direction = digitalio.Direction.OUTPUT
enable.value = False

# Set up NeoPixels
strip = neopixel.NeoPixel(NEOPIXEL_PIN, NUM_PIXELS, brightness=1, auto_write=False)
strip.fill(0)  # NeoPixels off ASAP on startup
strip.show()
```

```
audio = audioio.AudioOut(board.A0)  # Speaker
wave_file = None

# Set up accelerometer on I2C bus, 4G range:
i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
accel = adafruit_lis3dh.LIS3DH_I2C(i2c)
accel.range = adafruit_lis3dh.RANGE_4_G

COLOR_IDLE = COLOR # 'idle' color is the default for the staff handle
COLOR_HIT = ALT_COLOR  # "hit" color is ALT_COLOR set above
COLOR_SWING = SWING_COLOR  # "swing" color is SWING_COLOR set above
COLOR_TOP = TOP_COLOR  #"top" color is idle color for the ring


def play_wav(name, loop=False):
    """
    Play a WAV file in the 'sounds' directory.
    :param name: partial file name string, complete name will be built around
                 this, e.g. passing 'foo' will play file 'sounds/foo.wav'.
    :param loop: if True, sound will repeat indefinitely (until interrupted
                 by another sound).
    """
    global wave_file  # pylint: disable=global-statement
    print("playing", name)
    if wave_file:
        wave_file.close()
    try:
        wave_file = open('sounds/' + name + '.wav', 'rb')
        wave = audiocore.WaveFile(wave_file)
        audio.play(wave, loop=loop)
    except OSError:
        pass # we'll just skip playing then


def power(sound, duration, reverse):
    """
    Animate NeoPixels with accompanying sound effect for power on.
    @param sound: sound name (similar format to play_wav() above)
    @param duration: estimated duration of sound, in seconds (>0.0)
    @param reverse: Reverses animation. If True, begins animation at end of strip.
    """
    if reverse:
        prev = NUM_PIXELS
    else:
        prev = 0
    start_time = time.monotonic()  # Save audio start time
    play_wav(sound)
    while True:
        elapsed = time.monotonic() - start_time  # Time spent playing sound
        if elapsed > duration:                   # Past sound duration?
            break                                # Stop animating
        total_animation_time = elapsed / duration            # Animation time, 0.0
to 1.0
        if reverse:
            total_animation_time = 1.0 - total_animation_time            # 1.0 to
0.0 if reverse
        threshold = int(NUM_PIXELS * total_animation_time + 0.5)
        num = threshold - prev # Number of pixels to light on this pass
        if num != 0:
            if reverse:
                strip[threshold:prev] = [ALT_COLOR] * -num
            else:
                strip[prev:threshold] = [ALT_COLOR] * num
            strip.show()
            prev = threshold
```

```
def mix(color_1, color_2, weight_2):
    """
    Blend between two colors with a given ratio.
    :param color_1:  first color, as an (r,g,b) tuple
    :param color_2:  second color, as an (r,g,b) tuple
    :param weight_2: Blend weight (ratio) of second color, 0.0 to 1.0
    :return (r,g,b) tuple, blended color
    """
    if weight_2 < 0.0:
        weight_2 = 0.0
    elif weight_2 > 1.0:
        weight_2 = 1.0
    weight_1 = 1.0 - weight_2
    return (int(color_1[0] * weight_1 + color_2[0] * weight_2),
            int(color_1[1] * weight_1 + color_2[1] * weight_2),
            int(color_1[2] * weight_1 + color_2[2] * weight_2))

# List of swing wav files without the .wav in the name for use with play_wav()
swing_sounds = [
    'swing1',
    'swing2',
    'swing3',
]

# List of hit wav files without the .wav in the name for use with play_wav()
hit_sounds = [
    'hit1',
    'hit2',
    'hit3',
    'hit4',
]

# List of yell wav files without the .wav in the name for use with play_wav()
yell_sounds = [
    'yell1',
]


mode = 0  # Initial mode = OFF

# Setup idle pulse
idle_brightness = IDLE_PULSE_BRIGHTNESS_MIN  # current brightness of idle pulse
idle_increment = 0.01  # Initial idle pulse direction

# Main loop
while True:

    if mode == 0:  # If currently off...
        enable.value = True
        power('on', POWER_ON_SOUND_DURATION, True)  # Power up!
        play_wav('idle', loop=True)  # Play idle sound now
        mode = 1  # Idle mode
        time.sleep(1.0) #pause before moving on

        # Setup for idle pulse
        idle_brightness = IDLE_PULSE_BRIGHTNESS_MIN
        idle_increment = 0.01
        # lights the ring in COLOR_TOP color:
        strip[0:NUM_RING] = [([int(c*idle_brightness) for c in COLOR_TOP])] *
NUM_RING
        # lights the strip in COLOR_IDLE color:
        strip[NUM_RING:NUM_PIXELS] = [([int(c*idle_brightness) for c in
COLOR_IDLE])] * NUM_STRIP
        strip.show()

    elif mode >= 1:  # If not OFF mode...
        x, y, z = accel.acceleration  # Read accelerometer
        accel_total = x * x + z * z #x axis used for hit and for swing
```

```
        accel_yell = y * y + z * z  #y axis used for yell
        # Square root isn't needed, since we're
        # comparing thresholds...use squared values instead.)
        if accel_total > HIT_THRESHOLD:  # Large acceleration on x axis = HIT
            TRIGGER_TIME = time.monotonic()  # Save initial time of hit
            play_wav(random.choice(hit_sounds))  # Start playing 'hit' sound
            COLOR_ACTIVE = COLOR_HIT  # Set color to fade from
            mode = 3  # HIT mode
        elif mode == 1 and accel_total > SWING_THRESHOLD:  # Mild acceleration on x
axis = SWING
            TRIGGER_TIME = time.monotonic()  # Save initial time of swing
            play_wav(random.choice(swing_sounds))  # Randomly choose from available
swing sounds
            # make a larson scanner
            strip_backup = strip[0:-1]
            for p in range(-1, len(strip)):
                for i in range(p-1, p+2): # shoot a 'ray' of 3 pixels
                    if 0 <= i < len(strip):
                        strip[i] = COLOR_SWING
                strip.show()
                time.sleep(SWING_BLAST_SPEED)
                if 0 <= (p-1) < len(strip):
                    strip[p-1] = strip_backup[p-1]  # restore previous color at the
tail
                strip.show()
            while audio.playing:
                pass # wait till we're done
            mode = 2  # we'll go back to idle mode
        elif mode == 1 and accel_yell > YELL_THRESHOLD:  # Motion on Y axis = YELL
            TRIGGER_TIME = time.monotonic()  # Save initial time of swing
            # run a color down the staff, opposite of power-up
            previous = 0
            audio_start_time = time.monotonic()  # Save audio start time
            play_wav(random.choice(yell_sounds))  # Randomly choose from available
yell sounds
            sound_duration = YELL_SOUND_DURATION
            while True:
                time_elapsed = time.monotonic() - audio_start_time  # Time spent
playing sound
                if time_elapsed > sound_duration:  # Past sound duration?
                    break  # Stop animating
                animation_time = time_elapsed / sound_duration  # Animation time,
0.0 to 1.0
                pixel_threshold = int(NUM_PIXELS * animation_time + 0.5)
                num_pixels = pixel_threshold - previous  # Number of pixels to
light on this pass
                if num_pixels != 0:
                    # light pixels in YELL_COLOR:
                    strip[previous:pixel_threshold] = [YELL_COLOR] * num_pixels
                    strip.show()
                    previous = pixel_threshold
            while audio.playing:
                pass # wait till we're done
            mode = 4  # we'll go back to idle mode
        elif mode == 1:
            # Idle pulse
            idle_brightness += idle_increment  # Pulse up
            if idle_brightness > IDLE_PULSE_BRIGHTNESS_MAX or \
               idle_brightness < IDLE_PULSE_BRIGHTNESS_MIN:  # Then...
                idle_increment *= -1  # Pulse direction flip
            # light the ring:
            strip[0:NUM_RING] = [([int(c*idle_brightness) for c in COLOR_TOP])] *
NUM_RING
            # light the strip:
            strip[NUM_RING:NUM_PIXELS] = [([int(c*idle_brightness) for c in
                                            COLOR_IDLE])] * NUM_STRIP
            strip.show()
            time.sleep(IDLE_PULSE_SPEED)  # Idle pulse speed set above
        elif mode > 1:  # If in SWING or HIT or YELL mode...
```

```
            if audio.playing:  # And sound currently playing...
                blend = time.monotonic() - TRIGGER_TIME  # Time since triggered
                if mode == 2:  # If SWING,
                    blend = abs(0.5 - blend) * 3.0  # ramp up, down
                strip.fill(mix(COLOR_ACTIVE, COLOR, blend))  # Fade from hit/swing
to base color
                strip.show()
            else:  # No sound now, but still SWING or HIT modes
                play_wav('idle', loop=True)  # Resume idle sound
                mode = 1  # Return to idle mode
```

Customizing Your Code

The best way to edit and upload your code is with the Mu Editor, a simple Python editor that works with Adafruit CircuitPython hardware. It's written in Python and works on Windows, MacOS, Linux and Raspberry Pi. The serial console is built right in so you get immediate feedback from your board's serial output. Instructions for installing Mu is here (https://adafru.it/JF4).

Audio Files

Adafruit CircuitPython supports **16-bit, Mono, 22.050kHz .wav** audio format. See this guide (https://adafru.it/BvU) to help format any audio files you might want to use in this project besides the files provided.

In the main loop, the swing and hit modes randomly choose from a list of sounds. For example, swing1.wav, swing2.wav, swing3, etc. This makes the motion effects feel much more varied and less repetitive.

The yell mode plays **yell1.wav**. If you want more yelling sounds, you can add more files -- just call them **yell2.wav**, **yell3.wav** etc and you can integrate them into the code.

- Power on – **on.wav**
- Idle loop – **idle.wav**
- Swing 1 – **swing1.wav**
- Swing 2 – **swing2.wav**
- Swing 3 – **swing3.wav**
- Hit 1 – **hit1.wav**
- Hit 2 – **hit2.wav**
- Hit 3 – **hit3.wav**
- Hit 4 – **hit4.wav**
- Yell – **yell1.wav**

sounds.zip

https://adafru.it/JF5

Open the code in the Mu editor (or another text editor) and look near the top. You'll see a lot of variables that you can change to customize the color palettes and sensitivity of the motion of your staff.

First, change the value of `NUM_STRIP` to match the actual number of pixels in your NeoPixel strip. If you're using a different sized ring, you can change that number here as well.

```
# CHANGE TO MATCH YOUR RING AND STRIP SETUP
NUM_RING = 12   #12 pixel ring
NUM_STRIP = 44  # 44 pixels in my NeoPixel strip
NUM_PIXELS = NUM_STRIP + NUM_RING  #total number of pixels
```

Colors

The default color for the handle of the staff at idle is orange. The format here is R, G, B -- so we've made orange by mixing 200 Red, 50 Green, and 0 Blue values. You can play with the numbers here to mix your own colors, but be sure to keep the values under around 200.

More about mixing colors in CircuitPython can be found here. (https://adafru.it/DOd)

```
# CUSTOMISE COLORS HERE:
COLOR = (200, 50, 0)      # Default idle is orange
ALT_COLOR = (0, 200, 200)  # hit color is teal
SWING_COLOR = (200, 200, 200) #swing animation color is white
TOP_COLOR = (100, 100, 0)  #top color is yellow-green
YELL_COLOR = (200, 0, 200)  #yell color is purple
```

Idle Pulse Animation Speed

Next, you can customize the speed and brightness of the idle pulse animation and the swing blast speed.

```
# CUSTOMISE IDLE PULSE SPEED HERE: 0 is fast, above 0 slows down
IDLE_PULSE_SPEED = 0  # Default is 0 seconds
SWING_BLAST_SPEED = 0.007

# CUSTOMISE BRIGHTNESS HERE: must be a number between 0 and 1
IDLE_PULSE_BRIGHTNESS_MIN = 0.2  # Default minimum idle pulse brightness
IDLE_PULSE_BRIGHTNESS_MAX = 1  # Default maximum idle pulse brightness
```

Sensitivity

This section allows you to adjust the motion sensitivity of the staff. I have it set up so that the swing and yell thresholds trigger pretty easily, and I have to really move the staff hard to get the hit mode to trigger.

The swing mode triggers when the staff is moved along the X axis, and the yell mode triggers when the staff is moved along the Y axis. The hit mode is set up on the X axis also, but requires a firmer movement.

This means that to activate the swing animation, I move the staff left to right, and to activate the yell animation I move it front to back.

These numbers will really affect the feel and control of your staff, so don't be afraid to play around with them until it all feels just right.

```
# CUSTOMISE SENSITIVITY HERE: smaller numbers = more sensitive to motion
HIT_THRESHOLD = 1150
SWING_THRESHOLD = 800
YELL_THRESHOLD = 700
```

Wav File Updates

If you change the .wav files for power-up or for the yell mode, you can adjust the animation times here.

The code will automatically adjust the animation lengths for the hit and swing modes to match the length of your .wav files.

```
# Set to the length in seconds of the "on.wav" and "yell1.wav" files
POWER_ON_SOUND_DURATION = 3.0
YELL_SOUND_DURATION = 1.0
```

If you add or remove .wav files, you'll also need to update the list further down in the code (starting at line 146):

```
# List of swing wav files without the .wav in the name for use with play_wav()
swing_sounds = [
    'swing1',
    'swing2',
    'swing3',
]

# List of hit wav files without the .wav in the name for use with play_wav()
hit_sounds = [
    'hit1',
    'hit2',
```

```
    'hit3',
    'hit4',
]

# List of yell wav files without the .wav in the name for use with play_wav()
yell_sounds = [
    'yell1',
]
```

Adding your Own Wav Files

I found that even following the very specific guidelines for .wav file setup, some files would crash the Feather when triggered. If you're having this problem, head to this guide (https://adafru.it/BvU) and re-crunch your files using Audacity.

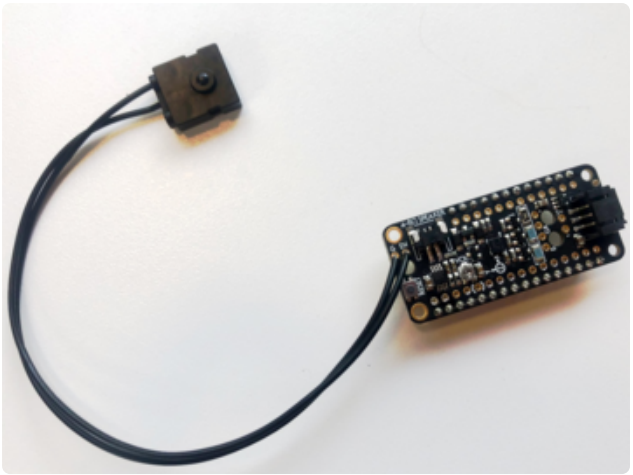
This is a teeny tiny speaker, so sound files with more high-end frequencies will sound a lot better than deep, booming sounds. Sound files that have a lot of (or even any) low frequencies do not sound good when played back over small speakers. Too much low end will cause a small speaker to distort. To optimize your sound files for small speaker playback use a High Pass Filter (HPF) in your preferred audio editing software. Rolling off low frequencies below 250 Hz is a good starting point.

Audacity is great and simple audio editing app, and best of all, it's free!

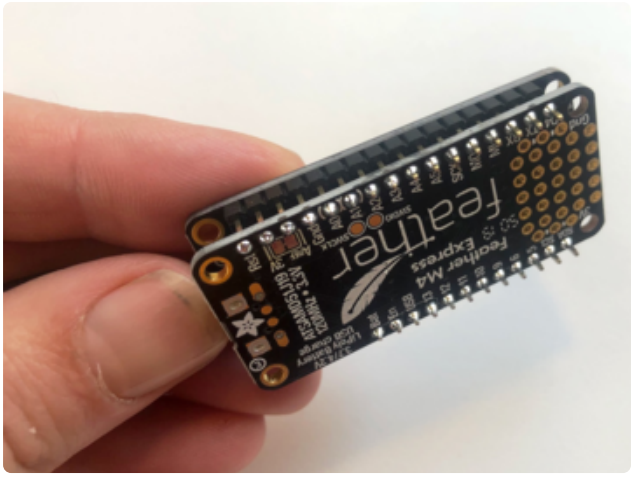
If your character doesn't resonate with the sounds included with this project, but you don't want to create your own, try using the files from the Zelda Master Sword (https://adafru.it/JF6) project as an alternative. It's amazing how the personality of the staff changes with different sound effects.

Electronics Assembly

Feather & PropMaker Wiring



First we'll solder the on/off switch onto the PropMaker Wing. Strip the ends of the wires and solder to the **G** and **ENABLE** pins near the reset button on the PropMaker Wing.



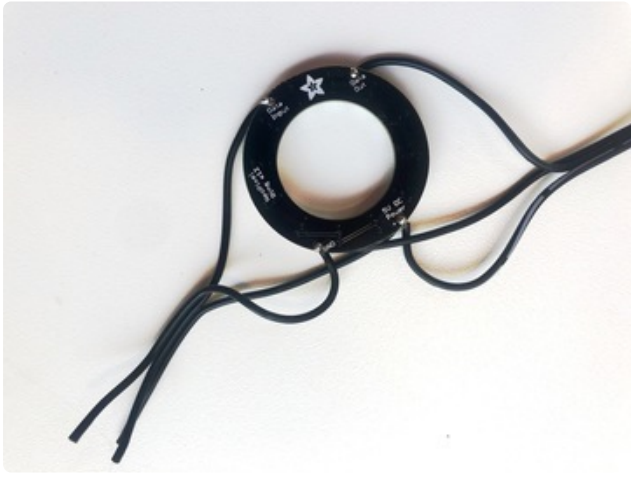
Next we'll attach the Feather M4 to the PropMaker Wing. Insert all the header pins into the holes on the Feather M4 and solder them in place one by one. Yes, this is scary. It helps to get the ones on the corners soldered in place so the two boards stay put, then you can take your time with all the ones in the middle.

NeoPixel Wiring

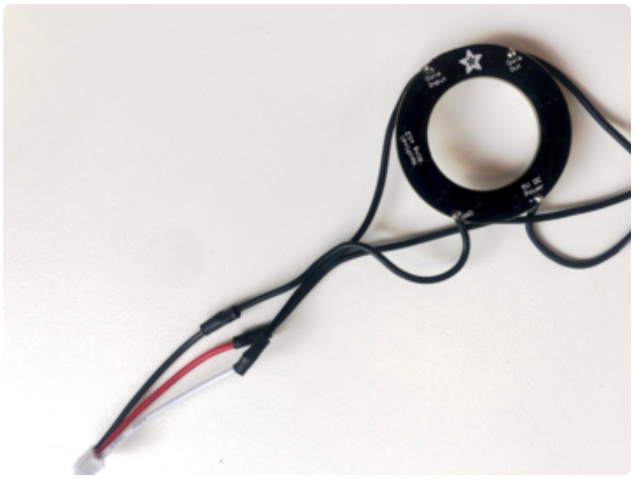
Next, we'll wire up the NeoPixel ring. I'm using this 4-strand silicone wire ribbon cable for this part, since it's so much easier to hide wires that are all stuck together and colored black. You've got to pay closer attention to which wire is which, so double-check before you solder!

First, find the striped wire. This will always be your power wire (traditionally colored red). The wire next to it will be our data wire (for **IN** and **OUT** pins) and the third wire will be for **G**. Take the fourth wire and pull it off entirely, since we don't need it for this project.

Cut two lengths of this 3-strand ribbon cable, each about 4-5" long.



Strip a little shielding off all three wires on one end of each section. Solder the two striped wires into **PWR**, getting both wires carefully into the hole. Then, solder the two middle wires into **IN** and **OUT** respectively. Twist the two remaining wires together and solder them both into **G**.

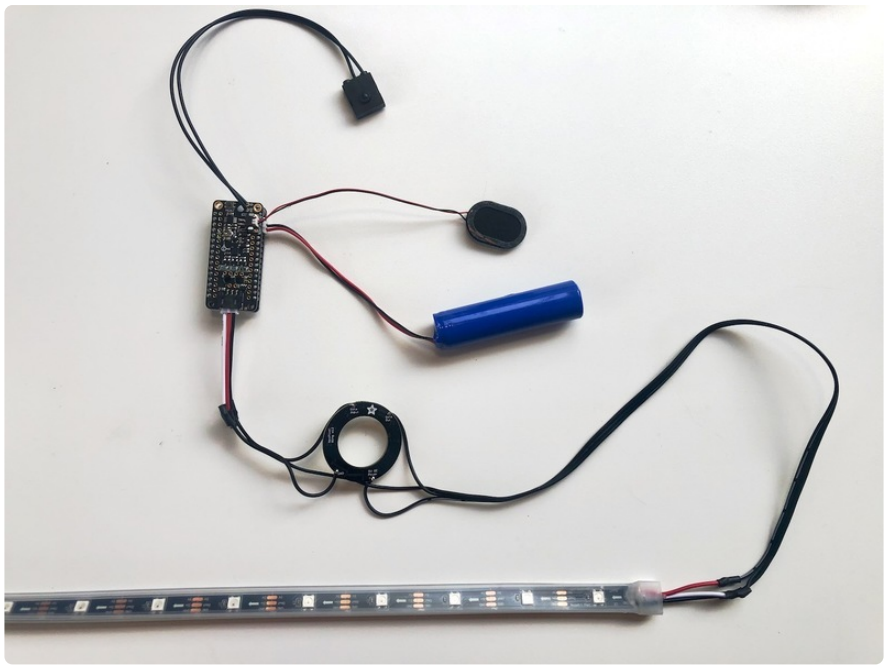


Trim the ribbon cable connected to the **IN** side to about 2-3 inches, and trim your 3-pin JST connector to about the same length. Splice red to **PWR** (striped wire), white to **IN** (middle wire), and black to **GND** (remaining wire).

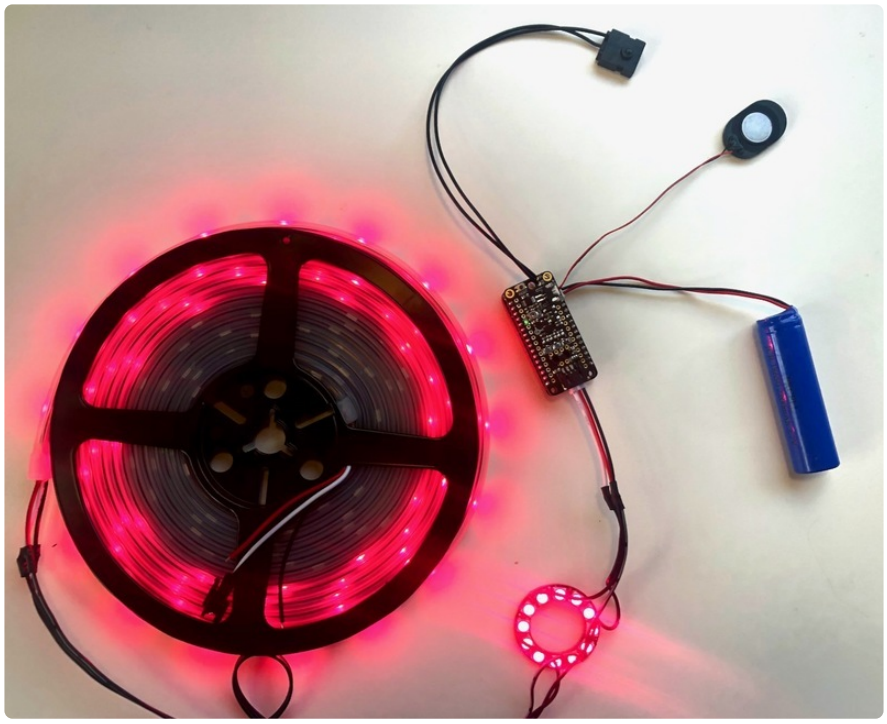
Find the **IN** end of your NeoPixel strip. This is usually coiled on the outside of the reel, with a female JST connector, but check the direction arrows printed on the strip to be double-sure you have the correct end. It won't work if you solder to the **OUT** end.

Trim off the JST connector and any additional wires - you'll need just one red, one black, and one white wire.

Splice the three remaining wires coming from the NeoPixel ring to the wires on the strip in the same order as with the connector: Splice red to **PWR** (striped wire), white to **OUT** (middle wire), and black to **GND** (remaining wire).



Plug your speaker into the mini JST connector on the PropMaker Wing, and the NeoPixel setup into the 3-pin JST connector at the other end. Plug your battery into the 2-pin connector on the Feather M4 and click your on/off switch. If you've uploaded your code already, the lights should come on and sound will start playing. Shake the PropMaker assembly around to test the different effects.



Troubleshooting

If your lights don't come on, here are a few things to try:

- If you're getting sound but no lights, check your NeoPixel wiring. Did you solder from the board to the **IN** pin on the NeoPixel ring, then from **OUT** to **IN** on the NeoPixel strip? If you get it backwards the lights won't work.
- Is your battery charged up? This board has onboard charging, so try plugging a USB cable into the board to see if that fixes things.
- If the lights on your board come on and blink, but you don't get any lights on the NeoPixels or sounds, you may be missing a software library file or you may have uploaded your code incorrectly. Head back to the software page and make sure you didn't miss any steps.
- If you get light effects but no sounds, make sure you copied the **sounds** folder onto the board with the .wav files inside. Also try plugging and unplugging your speaker to make sure it's seated correctly.
- I had trouble with some .wav files crashing

Make the Staff

The handle of the staff is made from 1" polycarbonate tubing. Make sure the interior diameter of your tubing is at least 7/8", so the battery will fit inside.

A note about polycarbonate vs. acrylic:

Acrylic tubes are cheaper and a bit easier to find, and they seem basically the same, but acrylic plastic is a more delicate material and is much more likely to shatter. This might not be an issue if you're a careful wizard who never lets anyone else play with her staff, but I'd really advise spending a few extra dollars to get polycarbonate. Magic tools should be sturdy.

I get my polycarbonate tubes from FlowToys (https://adafru.it/JCS). You can get a wonderful end cap (https://adafru.it/JCT) for the bottom from these guys as well. Capping the bottom end will extend the life of your staff and keep mud and debris out and away from your LEDs.

I am using a size 12F, which is about 5' long. Once I add the water bottle to the top of the staff, the whole thing becomes almost 6' long, just slightly taller than me.

Remove any labels from your water bottle using Goo Gone or another solvent. Cut off the top of the bottle just below the threads, so that it fits snugly over your polycarbonate tube. Secure it tightly with gaffer's tape or duct tape. You can add some tape around the top of the tube as well to get a nice tight fit.

Put some plastic sheeting down on your worktable to protect it from expanding foam and paint splatters. Expanding foam sticks to just about everything and does NOT come off, so you'll want to wear gloves and an apron or coveralls as well, or you'll ruin your favorite yoga pants.

I'm using Great Stuff (https://adafru.it/IPB) brand expanding foam, the Gaps & Cracks variety, since it expands a bit less than the Big Gap Filler variety so I can get finer details. Mind, it's still tough to get fine details with this stuff. It goes where it will, so it's great for messy organic shapes, but if you're looking to make an Elven High Court type staff, this might not be the best material for symmetry and smooth lines.

I sketched out a rough design on the water bottle with a sharpie before adding the foam. I want the water bottle to be covered enough that it doesn't look like a water bottle anymore, but have large "windows" to let the light shine through. Also, the beads of foam need enough contact with each other that they'll support the bottle without breaking off.

We'll need to cut through one of these windows later on to get our electronics inside, so keep that in mind when deciding on your design.

I also placed the end cap on the bottom of the staff, with a piece of saran wrap between the cap and the staff. This way the foam will flare out at the base of the staff, leaving room for the end cap inside, but not fusing to the end cap itself.

(I'm not sure if this was necessary -- I have no idea if the foam will stick to the silicone cap or not -- but I didn't want to take the chance, because this foam seems to stick to everything!)

Shake the can and attach the straw to the top. It's not a bad idea to practice on a piece of scrap wood first, to get the hang of how the foam comes out of the can, so you can get smoother lines. Once you've got the hang of it, start covering your staff with foam in long vertical lines.

Misting the foam with water will speed up the curing time from 45 minutes to around 20 minutes, so having a spray bottle handy is a good idea.

I found it works best to cover the top half of the tube, let the foam set up, then flip the staff and do the other side.

In between sections, close and cover the tip of the expanding foam straw with tape to keep it from oozing out or drying up.

You'll want to cover the tube with enough foam that it makes a cohesive casing. The foam is very strong when it's stuck to itself, but loose, open strands will be weaker and a bit more likely to break off. At the same time, it's fine to leave a few gaps here and there for more light to shine through.

Once you're happy with your basic shape and things are all covered, let the foam cure for at least 24 hours.

Once your foam is fully cured, grab your keyhole saw or bread knife and get carving.

If you like the cartoony look of the foam you can skip this step, but I really like the carved-down look. It looks like a tree branch to me, and has a lot of interesting organic texture to it.

Carve lengthwise along the staff, just taking the top off the foam and roughing out your shape. You can always add more foam if you take off too much.

Painting & Finishing

I'm using latex house paint in matte black, chocolate brown, and light grey for the highlights.

Our first step is to paint in all the nooks and crannies in the foam.

Mix the paint about 50/50 with water to thin it. This will carry the paint way down into the crevices and darken the deep places, and will keep the polycarbonate underneath relatively paint-free so the light will still shine through. Use a paintbrush to work it in.

I alternated brown and black paint for this part, to give more depth of color to my staff.

(For my previous prototype I used just black and no brown, and the staff came out looking more like wood that had been in the fire for a while, all ashen and burnt, which is also a very cool effect.)

Don't worry too much about getting paint on the water bottle. The thinned latex paint doesn't care to stick to the plastic too much, and once it dries, we'll clean it up a bit using 99% alcohol.

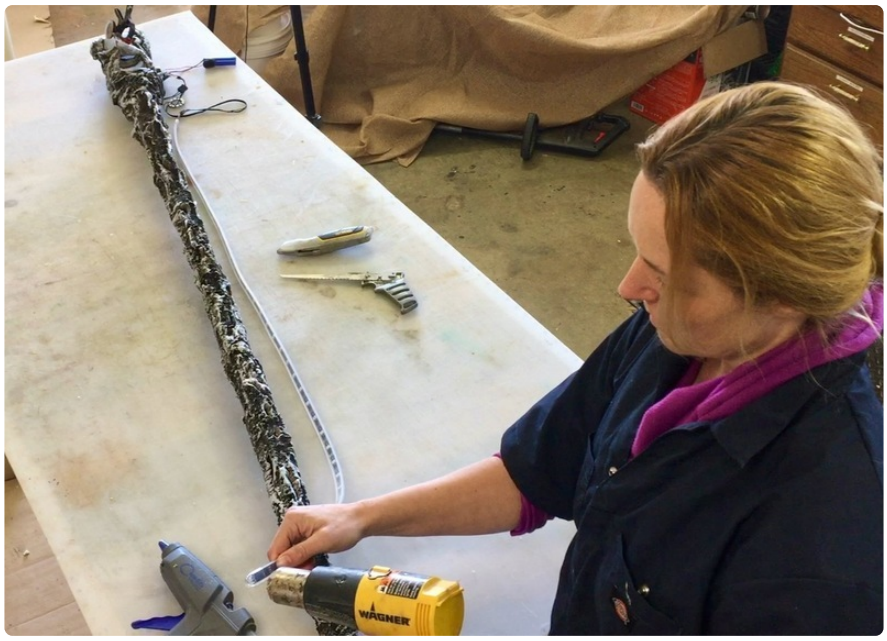
You can stop here, or create more of an oak-tree look by adding a light grey highlight.

Let the base coat dry, then use your paintbrush and a light grey latex paint (not thinned) to brush very lightly across the top of your staff. The paint will hit just the peaks and highlight all the lovely texture created by the foam.

Adding the Electronics

Use a utility knife to carefully cut a flap in your water bottle so you can access the inside where the bottle attaches to the tube. Make the cuts beneath the foam to hide them. Don't cut the window out entirely - just cut enough that you can peel back a flap of plastic and reach the inside.

Be careful cutting, using a glue gun, and using a heat gun.



Lay out your LED strip next to your staff, with the top of the lights next to where the bottle attaches. Measure and cut the lights carefully across the copper pads at the bottom of your staff. Seal up the end using hot glue and clear heat shrink.

Count how many LEDs you have in your strip, so you can use that number in your code.

Feed the LED strip down inside the polycarbonate tube from the top until all the lights are inside the staff. Then, slip the battery into the top of the tube above the lights. It should fit fairly snugly -- if it's really loose, add some tape around it so it doesn't shake around.

Find a good location for the speaker. It'll sound best if it's up inside the water bottle, but it's nice if it's hidden from view behind some expanding foam. Hot glue it into place.

Cut a small hole in the side of the bottle for the switch to come through. Thread the switch and wires through the hole, but leave it dangling loose for now -- we'll secure it down a bit later.

I used some black plastic tape to secure the NeoPixel ring to the back of the FeatherWing assembly. I wanted a tight, small package but also needed to avoid any potential short circuiting due to the parts bumping against each other. Adding the tape in between was a good solution.

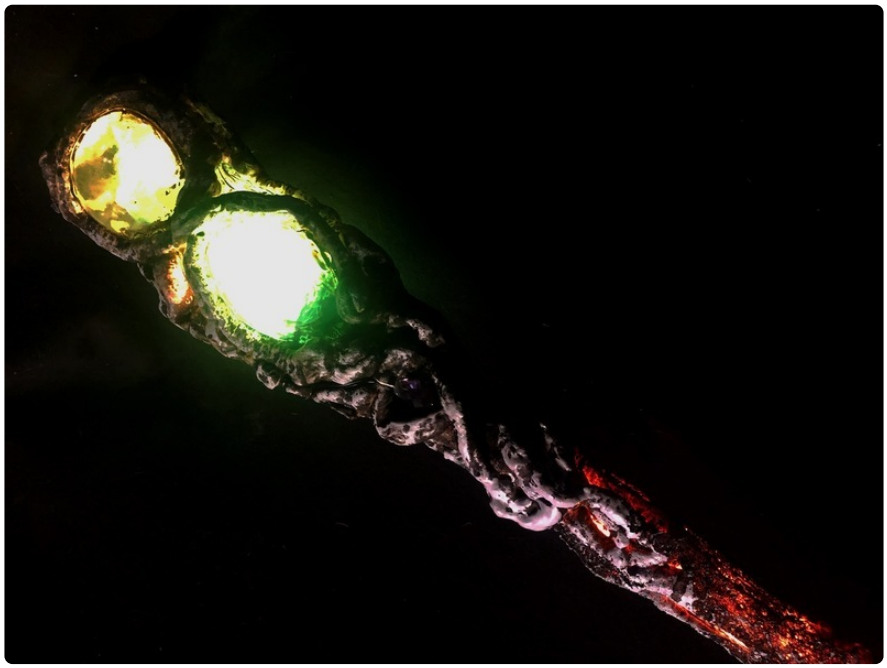
I nestled the FeatherWing assembly and NeoPixel ring carefully into the base of the staff head, so the ring lights shine up into the bottle and the USB port is accessible through my window flap. The board is oriented flat with the floor when the staff is upright. Once I was happy with the placement, and was sure everything was working, I hot-glued the board and ring in place.

This last bit was pretty tricky, and every staff will be different. You can always add more expanding foam if you end up cutting away too much! This material is really forgiving.

Finally, I threaded the on/off switch around until I found a nice hollow for it to nestle into. As a finishing touch, I used silicone glue to glue a jewel above the switch (being careful not to get any glue IN the switch). Silicone glue is rubbery when it dries, so this allows the switch to be activated when I press on the jewel.

Finally, I added a little bit of crumpled iridescent cellophane to the inside of the water bottle, in order to catch the light and give a magical sheen to the NeoPixel colors.

Use It



It takes a little practice to learn to trigger the different modes at will.

Power-Up mode happens when you press the power button. The lights come on at the bottom of the staff and work their way upwards until the whole staff is a gorgeous cyan blue.

Idle mode comes on when the staff is fully powered on but not in motion. The head of the staff is a yellow-green and the handle is orange, and the lights pulse gently while the "idle" sound plays in the background.

The other modes will trigger based on the orientation of your Feather.

Moving the Feather side to side gently will trigger the Swing animation modes -- a whoosh sound, with a photon type animation running down the length of the staff.

Moving the Feather front to back gently will trigger the Yell animation -- the whole staff lights up in purple and a Xena-Warrior-Princess inspired yell will play.

Moving the feather side to side agressively will trigger the Hit animation: a tinkly magic sound will play and the staff will turn blue, then fade back to orange.

Head back to the software page to customize colors and sensitivity, to make the staff work with your own Wizard or Warrior character.

Charging

The Feather M4 has onboard battery charging, so to charge the battery, just plug in a USB cable into the Feather's USB port. Your battery will charge. Magic!