# Table of Contents

## Overview
- Parts
- Materials & Tools

## Build the Lightsaber
- Creating the Blade
- Tube Guard
- Plastic and NeoPixel Sandwich
- Diffusion
- Insert the Light Strip
- Wire Slit
- Toy Handle
- Blade Reduction
- Battery Storage
- HalloWing Circuit
- Speaker Connection
- NeoPixel Connection
- Battery Connection
- Handle Attachments
- Sink Extension Handle
- Corrugation Slicing
- Make a Coupling
- Add the HalloWing
- Styling

## Program with CircuitPython
- CircuitPython Setup
- Libraries
- Sound Effects
- Lightsaber Code
- Unicorn Lightsaber Code

## Lightsaber Action

Overview

Making your own lightsabers may sound like a highly involved project that takes years of study under a wizened master, and exotic tools and materials -- but it doesn't have to be difficult at all!

In this guide we'll show you how to make your own interactive no-soldering-required lightsabers using common materials from the hardware or craft store and a few simple techniques.

With the HalloWing, we can use the onboard accelerometer to detect saber swinging and hitting motions, control the NeoPixel strip for blade animations, and play sounds through the built in amplifier to a small, yet powerful speaker!
## Parts

1 x **Adafruit HalloWing M0 Express**  
Skull-shaped ATSAMD21 board w 1.44" 128x128 TFT display  
[https://www.adafruit.com/product/3900](https://www.adafruit.com/product/3900)

1 x **NeoPixel Strip 0.5 Meter**  
with 3-pin JST connector  
[https://www.adafruit.com/product/3919](https://www.adafruit.com/product/3919)

1 x **Mini Oval Speaker**  
8 Ohm 1 Watt  
[https://www.adafruit.com/product/3923](https://www.adafruit.com/product/3923)

1 x **Lithium Ion Polymer Battery**  
3.7v 500mAh  
[https://www.adafruit.com/product/1578](https://www.adafruit.com/product/1578)
Materials & Tools

In addition to the components listed above, you'll also need:

- Plastic tube, such as a "bulb guard" used to protect fluorescent tubes, available at most hardware stores or [online](#) 24" long, "1" outside diameter
- Sheet of white corrugated plastic, a.k.a "Coroplast", available at most craft stores, hobby shops, and office supply stores
- Roll of parchment paper or wax paper
- Hobby knife
- Hot glue and hot melt glue gun
Creating the Blade

The blade of the lightsaber will light up from within using a strip of NeoPixels. To get things looking their best, we want a nice stable blade with good diffusion so you don't see the individual LEDs.

For this, we'll use two strips of corrugated plastic (A.K.A., CoroPlast) to sandwich the LED strip. The sandwich will then be inserted into the plastic tube with a roll of parchment paper as a diffuser.

Tube Guard

First, remove the plastic end caps and cut down your plastic tube to 24" in length. You can use a hobby knife or scissors to do this.
Plastic and NeoPixel Sandwich

Measure and cut two strips of translucent corrugated plastic a little shorter than the NeoPixel strip, and just a but less than the inside diameter of your tube.

Make the sandwich, and then use strips of clear tape to secure it.
Diffusion

Measure and cut a length of parchment paper the same length or a few inches shorter than your tube. (Full length is fine, but a bit shorter makes cable management a bit easier later).

Carefully roll the paper, avoiding wrinkling it, to just a bit smaller diameter than the tube, then insert it into the tube.
Insert the Light Strip

Carefully insert the plastic/NeoPixel strip sandwich into the tube, being careful not to bend the diffusion paper.
Wire Slit
Use scissors to cut a short slit in the tube end. This is used to allow the wire to exit the tube without eating up too much of its length!

Toy Handle
First, we'll show how to re-use a toy lightsaber handle. If you want to use a plumbing sink extension instead, scroll down a bit.
If you want to avoid cutting up the toy saber blade, you can skip this step and simply fit the tube over the collapsed blade, and then you'll need to affix the battery to the exterior of the handle.

Blade Reduction

Extend the blade and then use a hand saw or power saw to cut through the end of the bottom section. This will allow us to remove the other, smaller sections.
Battery Storage

Now you can place the battery into the base of the tube blade and run the wiring out of the slit.

Place the tube guard over the remaining toy blade as shown.

Push the tube down into the handle.
HalloWing Circuit

Now we can connect the speaker, battery, and NeoPixel strip to the HalloWing, and then mount it to the lightsaber handle.

**Speaker Connection**

Plug the speaker cable into the HalloWing's A0 SPEAKER JST socket.

**NeoPixel Connection**

Next, plug the NeoPixel cable into the HalloWing's NEOPIX JST socket.
Battery Connection

Plug the battery into the HalloWing's battery JST connector.
Handle Attachments

You can attach the HalloWing and speaker to the lightsaber handle as shown, using double stick foam tape. Zip ties work great as well!

For the sticker, you can peel off the adhesive oval's covering and connect the speaker directly to the handle. Even though the speaker is technically "facing in" it is of nearly the same volume either way -- this may be due to the handle acting as a resonator.
Sink Extension Handle

You can make a nice handle for your saber with all kinds of materials found at the hardware store. I picked a 1-1/2" x 12" sink drain extension tube. Here's how to incorporate it.

The diameters of the tube guard blade and the handle are not the same, so we'll create a tight coupling for them using corrugated plastic.

The key here is to transform the corrugated plastic, which is normally stiff, into a bendable sheet by slicing through the top face.
Corrugation Slicing

Cut a piece of corrugated plastic about 4" wide by 2-1/2" high making sure that the fluting is running vertically as shown.

Being careful not to cut through the bottom face, use a hobby knife to slice the top face between each flute of the corrugation.

You'll now be able to easily bend the plastic, wrapping it around the base of the tube.

Make a Coupling

Use a strip of tape to secure the coupling as shown, and then push the assembly into the handle, making sure to allow the wires from the NeoPixels and battery to clear.
Add the HalloWing

Now, you can connect the HalloWing to the NeoPixel strip, battery, and speaker by plugging them into their respective JST ports.
You can use double stick foam tape to attach the HalloWing and speaker to the handle.
Styling

How you dress up and style your handle is a matter of personal choice. Express yourself!

Here, I decided to go with a retro 80s sci-fi styling by adding an alternating black-and-chrome grip motif to the handle.

I did this by cutting and slicing another piece of corrugated plastic, attaching it with tape, and then using a chrome paint marker.
Program with CircuitPython

CircuitPython Setup

To get started, you'll want to set up your HalloWing by following this guide. When you're ready, and can upload code to the board return here.

Adafruit really likes using the Mu editor to edit the CircuitPython code. See this guide on loading and using Mu.
More details on using with CircuitPython 6 can be found [here](#).

## Libraries

You'll also need a couple of libraries for this project. [Follow this guide](#) on adding libraries. The only two you'll need are the neopixl.mpy and adafruit_lis3dh.mpy files from the Circuit Python bundle in the 'lib' folder, so just drag those two from your downloaded, unzipped 'lib' folder onto the HalloWing.

![HalloWing and lightsaber](image)

## Sound Effects

The code was designed to call on five different sound effects depending on your actions. Here are the actions and related sounds:

- Press A2 to turn on lightsaber: on.wav
- Hold lightsaber mostly still: idle.wav
- Swing lightsaber: swing.wav
- Hit lightsaber: hit.wav
- Press A2 to turn off lightsaber: off.wav

Download these two .zip archives containing the sound effects sets for standard and unicorn modes. Then, unzip them. Depending on which sound effects set you want to use, drag the entire sounds folder for that set onto the HalloWing.

[ sounds_standard.zip ]
Lightsaber Code

Here is the code for a standard lightsaber. This is a "standard" lightsaber, with a single color blade. (Check out the Unicorn Mode code down below for multicolor action!)

You can customize it by changing the `COLOR` values -- we have some presets, but you can use any RGB color combination you like.

Copy the code, then paste it into Mu and save it to your HalloWing with the name code.py (or main.py) the board will automatically run the file named code.py (or main.py) so you must use one of those names.

```python
# SPDX-FileCopyrightText: 2018 John Edgar Park for Adafruit Industries
# SPDX-License-Identifier: MIT

"""LASER SWORD (pew pew) example for Adafruit Hallowing & NeoPixel strip"
# pylint: disable=bare-except

import time
import math
import audioio
import audiocore
import busio
import board
import touchio
import neopixel
import adafruit_lis3dh

# CUSTOMIZE YOUR COLOR HERE:
# (red, green, blue) -- each 0 (off) to 255 (brightest)
```
COLOR = (0, 100, 255)  # jedi
#COLOR = (255, 0, 0)  # sith

# CUSTOMIZE SENSITIVITY HERE: smaller numbers = more sensitive to motion
HIT_THRESHOLD = 250
SWING_THRESHOLD = 125

NUM_PIXELS = 30                        # NeoPixel strip length (in pixels)
NEOPIXEL_PIN = board.EXTERNAL_NEOPIXEL # Pin where NeoPixels are connected
STRIP = neopixel.NeoPixel(NEOPIXEL_PIN, NUM PIXELS, brightness=1, auto_write=False)
STRIP.fill(0)                          # NeoPixels off ASAP on startup
STRIP.show()

TOUCH = touchio.TouchIn(board.A2)      # Rightmost capacitive touch pad
AUDIO = audioio.AudioOut(board.A0)     # Speaker

# Set up accelerometer on I2C bus, 4G range:
I2C = busio.I2C(board.SCL, board.SDA)
try:
    ACCEL = adafruit_lis3dh.LIS3DH_I2C(I2C, address=0x18) # Production board
except:
    ACCEL = adafruit_lis3dh.LIS3DH_I2C(I2C, address=0x19) # Beta hardware

COLOR_SWING = COLOR
COLOR_HIT = (255, 255, 255)  # "hit" color is white

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).
    """
    try:
        wave_file = open('sounds/' + name + '.wav', 'rb')
        wave = audiocore.WaveFile(wave_file)
        AUDIO.play(wave, loop=loop)
    except:
        return

def power(sound, duration, reverse):
    """Animate NeoPixels with accompanying sound effect for power on / off.
    @param sound:    sound name (similar format to play_wav() above)
    @param duration: estimated duration of sound, in seconds (>0.0)
    @param reverse:  if True, do power-off effect (reverses animation)
    """
    start_time = time.monotonic()  # Save function start time
    play_wav(sound)  # Save function start time
    while True:
        elapsed = time.monotonic() - start_time  # Time spent in function
        if elapsed > duration:  # Past sound duration?
            break  # Stop animating
        fraction = elapsed / duration  # Animation time, 0.0 to 1.0
        if reverse:
            fraction = 1.0 - fraction  # 1.0 to 0.0 if reverse
        fraction = math.pow(fraction, 0.5)  # Apply nonlinear curve
        threshold = int(NUM PIXELS * fraction + 0.5)
        for pixel in range(NUM PIXELS):
            if pixel <= threshold:  # Fill NeoPixel strip
                STRIP[pixel] = COLOR_IDLE
            else:
                STRIP[pixel] = 0  # ON pixels BELOW threshold
        if reverse:
            ...
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))

# Main program loop, repeats indefinitely
while True:
    if TOUCH.value:
        if MODE == 0:
            power('on', 1.7, False)  # Power up!
            play_wav('idle', loop=True)  # Play background hum sound
            MODE = 1  # ON (idle) mode now
        else:
            power('off', 1.15, True)  # Power down
            MODE = 0  # OFF mode now
    else:
        while TOUCH.value:
            time.sleep(0.2)  # to avoid repeated triggering
    if TOUCH.value:
        ACCEL_X, ACCEL_Y, ACCEL_Z = ACCEL.acceleration  # Read accelerometer
        ACCEL_SQUARED = ACCEL_X ** 2 + ACCEL_Z ** 2  # (Y axis isn't needed for this, assuming Hallowing is mounted sideways to stick. Also, square root isn't needed, since we're just comparing thresholds...use squared values instead, save math.)
        if ACCEL_SQUARED > HIT_THRESHOLD:  # Large acceleration = HIT
            TRIGGER_TIME = time.monotonic()  # Save initial time of hit
            play_wav('hit')  # Start playing 'hit' sound
            COLOR_ACTIVE = COLOR_HIT  # Set color to fade from
            MODE = 3  # HIT mode
        elif MODE == 1 and ACCEL_SQUARED > SWING_THRESHOLD:  # Mild = SWING
            TRIGGER_TIME = time.monotonic()  # Save initial time of swing
            play_wav('swing')  # Start playing 'swing' sound
            COLOR_ACTIVE = COLOR_SWING  # Set color to fade from
            MODE = 2  # SWING mode
        elif MODE > 1:
            if AUDIO.playing:
                BLEND = time.monotonic() - TRIGGER_TIME  # Time since triggered
                if BLEND > 0:
                    BLEND = abs(0.5 - BLEND) * 2.0  # ramp up, down
                    STRIP.fill(mix(COLOR_ACTIVE, COLOR_IDLE, BLEND))
                    STRIP.show()
                else:
                    # No sound now, but still MODE > 1
                    play_wav('idle', loop=True)  # Resume background hum
                    STRIP.fill(COLOR_IDLE)  # Set to idle color
                    MODE = 1  # IDLE mode now
            else:
                # No sound now, but still MODE > 1
                play_wav('idle', loop=True)  # Resume background hum
                STRIP.fill(COLOR_IDLE)  # Set to idle color
                MODE = 1  # IDLE mode now
Unicorn Lightsaber Code

Here is the code for a Unicorn Mode lightsaber with multicolor rainbow effects!

Copy the code, then paste it into Mu and save it to your HalloWing with the name code.py (or main.py) the board will automatically run the file named code.py (or main.py) so you must use one of those names.

```python
# SPDX-FileCopyrightText: 2018 John Edgar Park for Adafruit Industries
#
# SPDX-License-Identifier: MIT

#"UNICORN SWORD example for Adafruit Hallowing & NeoPixel strip"
#
# pylint: disable=bare-except

import time
import math
import random
import board
import busio
import audioio
import audiocore
import touchio
import neopixel
import adafruit_lis3dh

# CUSTOMIZE SENSITIVITY HERE: smaller numbers = more sensitive to motion
HIT_THRESHOLD = 250
SWING_THRESHOLD = 125
NUM_PIXELS = 30                        # NeoPixel strip length (in pixels)
NEOPIXEL_PIN = board.EXTERNAL_NEOPIXEL # Pin where NeoPixels are connected
STRIP = neopixel.NeoPixel(NEOPIXEL_PIN, NUM_PIXELS, brightness=1, auto_write=False)
STRIP.fill(0)                          # NeoPixels off ASAP on startup
STRIP.show()

TOUCH = touchio.TouchIn(board.A2)      # Rightmost capacitive touch pad
AUDIO = audioio.AudioOut(board.A0)     # Speaker
MODE = 0                               # Initial mode = OFF
FRAMES = 10                            # Pre-calculated animation frames

# Set up accelerometer on I2C bus, 4G range:
I2C = busio.I2C(board.SCL, board.SDA)
try:
    ACCEL = adafruit_lis3dh.LIS3DH_I2C(I2C, address=0x18) # Production board
except:
    ACCEL = adafruit_lis3dh.LIS3DH_I2C(I2C, address=0x19) # Beta hardware
ACCEL.range = adafruit_lis3dh.RANGE_4_G

def hsv_to_rgb(hue, saturation, value):
    # Convert HSV color (hue, saturation, value) to RGB (red, green, blue)
    @param hue:
    0=Red, 1/6=Yellow, 2/6=Green, 3/6=Cyan, 4/6=Blue, etc.
```
@param saturation: 0.0=Monochrome to 1.0=Fully saturated
@param value: 0.0=Black to 1.0=Max brightness
@returns: red, green, blue each in range 0 to 255

```python
hue = hue * 6.0  # Hue circle = 0.0 to 6.0
sxt = math.floor(hue)  # Sextant index is next-lower integer of hue
frac = hue - sxt  # Fraction-within-sextant is 0.0 to <1.0
sxt = int(sxt) % 6  # mod6 the sextant so it's always 0 to 5

if sxt == 0:  # Red to <yellow
    red, green, blue = 1.0, frac, 0.0
elif sxt == 1:  # Yellow to <green
    red, green, blue = 1.0 - frac, 1.0, 0.0
elif sxt == 2:  # Green to <cyan
    red, green, blue = 0.0, 1.0, frac
elif sxt == 3:  # Cyan to <blue
    red, green, blue = 0.0, 1.0 - frac, 1.0
elif sxt == 4:  # Blue to <magenta
    red, green, blue = frac, 0.0, 1.0
else:  # Magenta to <red
    red, green, blue = 1.0, 0.0, 1.0 - frac

invsat = 1.0 - saturation  # Inverse-of-saturation

red = int(((red * saturation) + invsat) * value * 255.0 + 0.5)
green = int(((green * saturation) + invsat) * value * 255.0 + 0.5)
blue = int(((blue * saturation) + invsat) * value * 255.0 + 0.5)

return red, green, blue
```

# Unlike the single-color laser sword example which can compute and fill # the NeoPixel strip on the fly, this version is doing a bunch of color # calculations which would slow things down too much when also trying to # read the accelerometer. Instead, the 'idle' color state of the sword, # plus each of two animations (swinging and hitting) are pre-computed at # program start and stored in bytearrays...these can be quickly issued # to the NeoPixel strip later as needed.

```python
IDLE = bytearray(NUM_PIXELS * STRIP.bpp)
SWING_ANIM = [bytearray(NUM_PIXELS * STRIP.bpp) for i in range(FRAMES)]
HIT_ANIM = [bytearray(NUM_PIXELS * STRIP.bpp) for i in range(FRAMES)]

IDX = 0
for PIXEL in range(NUM_PIXELS):  # For each pixel along strip...
    HUE = PIXEL / NUM_PIXELS  # 0.0 to <1.0
    RED, GREEN, BLUE = hsv_to_rgb(HUE, 1.0, 0.2)
    IDLE[IDX + STRIP.order[0]] = RED  # Store idle color for pixel
    IDLE[IDX + STRIP.order[1]] = GREEN
    IDLE[IDX + STRIP.order[2]] = BLUE

for frame in range(FRAMES):  # For each frame of animation...
    FRAC = frame / (FRAMES - 1)  # 0.0 to 1.0
    HIT_ANIM[frame][IDX + STRIP.order[0]] = RED
    HIT_ANIM[frame][IDX + STRIP.order[1]] = GREEN
    HIT_ANIM[frame][IDX + STRIP.order[2]] = BLUE
    SWING_ANIM[frame][IDX + STRIP.order[0]] = RED
    SWING_ANIM[frame][IDX + STRIP.order[1]] = GREEN
    SWING_ANIM[frame][IDX + STRIP.order[2]] = BLUE

IDX += 3
```

# Go back through the hit animation and randomly set one # pixel per frame to white to create a sparkle effect.
for frame in range(FRAMES):
    IDX = random.randint(0, NUM_PIXELS - 1) * 3
    HIT_ANIM[frame][IDX] = 255
    HIT_ANIM[frame][IDX + 1] = 255
    HIT_ANIM[frame][IDX + 2] = 255

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).

```python
try:
    wave_file = open('sounds/' + name + '.wav', 'rb')
    wave = audiocore.WaveFile(wave_file)
    AUDIO.play(wave, loop=loop)
except:
    return
```

def power(sound, duration, reverse):
    ""
    Animate NeoPixels with accompanying sound effect for power on / off.
    @param sound:    sound name (similar format to play_wav() above)
    @param duration: estimated duration of sound, in seconds (>0.0)
    @param reverse:  if True, do power-off effect (reverses animation)
    ""
    start_time = time.monotonic()  # Save function start time
    play_wav(sound)
    while True:
        elapsed = time.monotonic() - start_time  # Time spent in function
        if elapsed > duration:  # Past sound duration?
            break  # Stop animating
        fraction = elapsed / duration  # Animation time, 0.0 to 1.0
        if reverse:
            fraction = 1.0 - fraction  # 1.0 to 0.0 if reverse
        fraction = math.pow(fraction, 0.5)  # Apply nonlinear curve
        threshold = int(NUM_PIXELS * fraction + 0.5)
        idx = 0
        for pixel in range(NUM_PIXELS):  # Fill NeoPixel strip
            if pixel <= threshold:
                STRIP[pixel] = (  # BELOW threshold,
                    IDLE[idx + STRIP.order[0]],  # fill pixels with
                    IDLE[idx + STRIP.order[1]],  # IDLE pattern
                    IDLE[idx + STRIP.order[2]])
            else:
                STRIP[pixel] = 0  # OFF pixels ABOVE threshold
        if reverse:
            STRIP.fill(0)  # At end, ensure strip is off
        STRIP.show()
        idx += 3
    else:
        neopixel_write(STRIP.pin, IDLE)  # or all pixels set on
        while AUDIO.playing:  # Wait until audio done
            pass

    # Main program loop, repeats indefinitely
    while True:
        if TOUCH.value:  # Capacitive pad touched?
            if MODE == 0:  # If currently off...
                power('on', 3.0, False)  # Power up!
                play_wav('idle', loop=True)  # Play background hum sound
                MODE = 1  # ON (idle) mode now
            else:  # else is currently on...
                power('off', 2.0, True)  # Power down
                MODE = 0  # OFF mode now
        while TOUCH.value:  # Wait for button release
            time.sleep(0.2)  # to avoid repeated triggering
        elif MODE >= 1:  # If not OFF mode...
            ACCEL_X, ACCEL_Y, ACCEL_Z = ACCEL.acceleration  # Read accelerometer
            ACCEL_SQUARED = ACCEL_X * ACCEL_X + ACCEL_Z * ACCEL_Z
            # (Y axis isn't needed for this, assuming Hallowing is mounted
```

©Adafruit Industries  Page 35 of 36
# sideways to stick. Also, square root isn't needed, since we're
# just comparing thresholds...use squared values instead, save math.)
if ACCEL_SQUARED > HIT_THRESHOLD:   # Large acceleration = HIT
    TRIGGER_TIME = time.monotonic() # Save initial time of hit
    play_wav('hit')                 # Start playing 'hit' sound
    ACTIVE_ANIM = HIT_ANIM
    MODE = 3                        # HIT mode
elif MODE == 1 and ACCEL_SQUARED > SWING_THRESHOLD: # Mild = SWING
    TRIGGER_TIME = time.monotonic() # Save initial time of swing
    play_wav('swing')               # Start playing 'swing' sound
    ACTIVE_ANIM = SWING_ANIM
    MODE = 2                        # SWING mode
elif MODE > 1:                      # If in SWING or HIT mode...
    if AUDIO.playing:               # And sound currently playing...
        BLEND = time.monotonic() - TRIGGER_TIME # Time since triggered
        BLEND *= 0.7                # 0.0 to 1.0 in ~1.4 sec
        if MODE == 2:               # If SWING,
            BLEND = abs(0.5 - BLEND) * 2.0 # ramp up, down
            if BLEND > 1.0:
                BLEND = 1.0
        elif BLEND < 0.0:
            BLEND = 0.0
        FRAME = int(BLEND * (FRAMES - 1) + 0.5)
        neopixel_write(STRIP.pin, ACTIVE_ANIM[FRAME])
    else:                           # No sound now, but still MODE > 1
        play_wav('idle', loop=True) # Resume background hum
        neopixel_write(STRIP.pin, IDLE) # Show idle pattern
        MODE = 1                    # IDLE mode now

Lightsaber Action

Now you can use your lightaber! Here are a few notes on operation.

- Depending on battery size, you'll get from 30 minutes to a couple of hours of continuous use -- power is conserved when the blade NeoPixels are "retracted" and when the board is powered off with the on/off switch
- You can charge the battery by simply plugging the HalloWing into USB power
- Deploy and retract the blade by touching the A5 capacitive touch pad as shown in the video above
- Gentle swings trigger the swing effect, fast swings or hitting things (not too hard) will trigger the hit effect
- If the sensitivity for swings and hits is off, you can adjust in the code the `HIT_THRESHOLD` and `SWING_THRESHOLD` values