Light-Up Reactive Ukulele

Created by Erin St Blaine

https://learn.adafruit.com/light-up-reactive-ukulele

Last updated on 2024-06-03 03:09:35 PM EDT

Table of Contents

Overview

Ukuleles are little joy machines. They're easy to play, portable, light, and really inexpensive - you can get a fairly good sounding one for around $40-$50.

This clear plastic ukulele from Kala simply begs to be upgraded with rainbow lights. Craft an over-the top tacky Tiki musical experience for your audience at your next luau.

This LED ukulele goes even further, with loads of customizable animations and a sound reactive mode. Match your animation playlist with your song repertoire, and bring the music to life as your uke pulses along with your strumming.

We've also used the FFT capabilities of our hardware to add note-sensing to this project. Change animation modes or turn sound reactive mode on or off just by plucking a particular note on your fretboard.

And just for giggles, we've added a Guitar-Hero-style "Rockstar tilt" mode -- tilt the neck of your ukulele up during your big finish to trigger a colorful lightning animation.

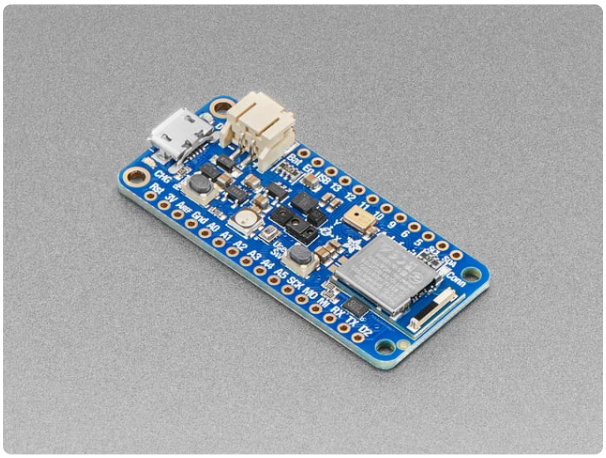
This ukulele is maika'i loa! (That's Hawaiian for awesome..)

Difficulty

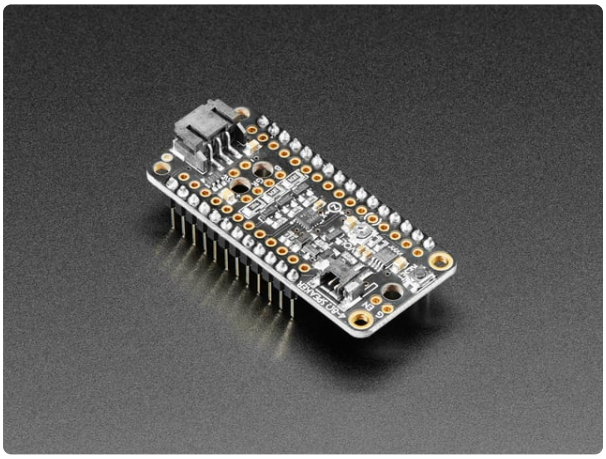
This is a fairly advanced project which took some tight soldering and some tricky maneuvering to get the LEDs in place. I built my ukulele "ship-in-a-bottle" style, placing my LEDs inside the instrument with needle nose pliers and a bent coat hanger. I didn't want to do too much damage to my ukulele by taking it apart, so I took my time and assembled it with care.

It helped a lot to have small hands and a lot of patience. It may be easier for some makers to disassemble the ukulele, put the lights inside and then glue it back together. But my careful method worked! It took me all afternoon.. but it only took me one afternoon.

Adafruit Parts Needed



Adafruit Feather nRF52840 Sense
The Adafruit Feather Bluefruit Sense takes our popular Feather nRF52840 Express and adds a smorgasbord of sensors...
https://www.adafruit.com/product/4516



Assembled Adafruit Prop-Maker FeatherWing
The Adafruit Feather series gives you lots of options for a small, portable, rechargeable microcontroller board. Perfect for fitting into your next prop build! This FeatherWing will...
https://www.adafruit.com/product/4145

1 x Battery
https://www.adafruit.com/product/2011
Lithium Ion Battery - 3.7v 2000mAh

USB DIY Connector - MicroB Female Plug

1 x USB Female — https://www.adafruit.com/product/1829
USB DIY Connector - MicroB Female Plug

---

1 x USB Male — https://www.adafruit.com/product/1826
USB DIY Slim Connector Shell - MicroB Plug

---

1 x Side Light NeoPixels — https://www.adafruit.com/product/3635
Adafruit NeoPixel LED Side Light Strip - Black 90 LED

---

1 x Skinny NeoPixels — https://www.adafruit.com/product/2969
Adafruit Mini Skinny NeoPixel Digital RGB LED Strip - 144 LED/m

---

1 x NeoPixel Connector — https://www.adafruit.com/product/3955
JST PH 4-Pin to Male Header Cable - I2C STEMMA Cable - 200mm

---

1 x Ribbon Cable - 10 Wire — https://www.adafruit.com/product/3890
Silicone Cover Stranded-Core Ribbon Cable - 10 Wire 1 Meter Long

---

1 x Ribbon Cable - 4 Wire — https://www.adafruit.com/product/3889
Silicone Cover Stranded-Core Ribbon Cable - 4 Wires 1 Meter Long

---

1 x Thermoplastic — https://www.adafruit.com/product/2504
Hand-Moldable Plastic - Low Temperature Thermoplastic

---

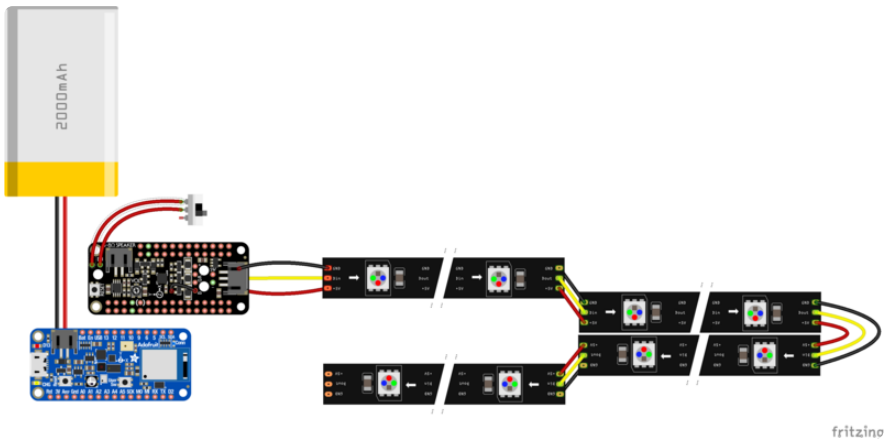
1 x On/Off Switch — https://www.adafruit.com/product/805
Breadboard-friendly SPDT Slide Switch

Additonal Parts & Tools

- A transparent ukulele - Mine is from Kala, the Transparent Ice Soprano Waterman (https://adafru.it/MDU)
- Silicone Adhesive - my favorite is Devcon Silicone Adhesive (https://adafru.it/MDV)
- Krazy glue / superglue
- A Power Drill with a long extension 3/8" spade bit and a few smaller bits
- Rotary tool or sandpaper for drill hole cleanup
- Soldering iron & accessories
- White Industrial velcro
- Needle nose pliers in various shapes and sizes

- Wire coat hanger or bendable 16g steel wire

Circuit Diagram



We're using four pieces of NeoPixel strip total, wired in sequence to fit neatly around the body and neck of the ukulele. The pixels plug into the NeoPixel port on the PropMaker Wing.

The battery plugs into the battery port on the Feather, and our on/off switch will be soldered into the **G** and **EN** ports on the PropMaker Wing.

Finally, the PropMaker Wing and Feather will be stacked and soldered together using the pre-soldered headers, in a nice tidy package.

We'll also wire up a custom USB port extension (not shown).

Software

It's always a good idea to get your software loaded onto your board before the build. That way you'll be able to test your solder joints at each step of the way, and you'll get instant gratification when you plug in the lights.

Getting the software loaded is a 3-step process:

1. Install the operating system (CircuitPython) on the board
2. Copy the required libraries into the /**lib** folder on the board
3. Save the **code.py** file to the board

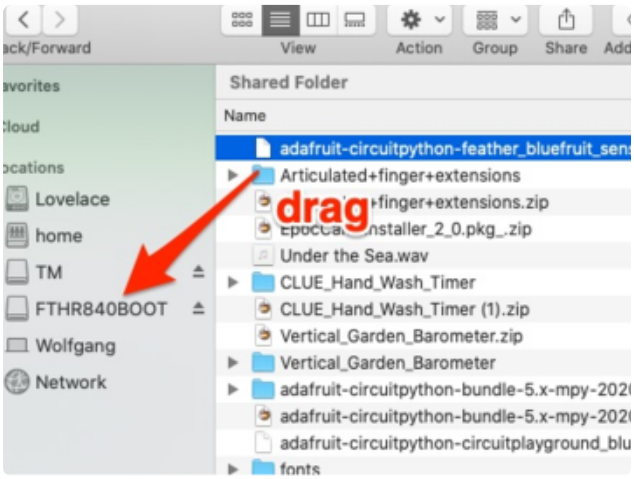
CircuitPython (https://adafru.it/EFq) is a fairly new OS that's changing rapidly. New features are being added and bugs are being fixed all the time, so it's always best to get a fresh version of CircuitPython and the library files before coding.

Install CircuitPython

The Adafruit Feather Sense 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 Bluefruit Sense guide (https://adafru.it/JBU), or click the button below for a direct download link.

Download the Latest CircuitPython Code for the Feather Bluefruit Sense

https://adafru.it/JqE



Download the file, plug your Feather Sense into your computer via the USB port, and double-click the reset button. You'll see a drive appear called **FTHR840BOOT.** Drag the .uf2 file you just downloaded onto this drive to install CircuitPython.

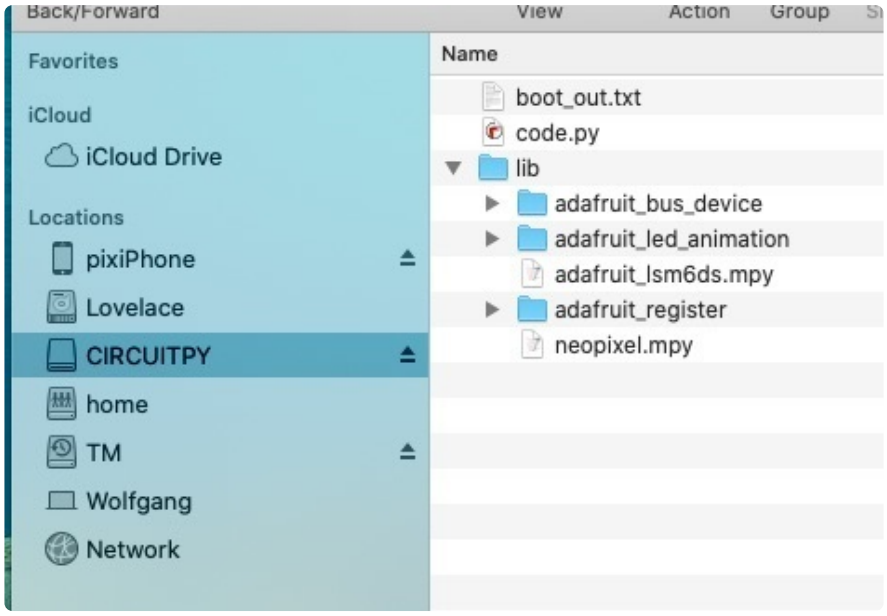
You'll know it worked if the **FTHR840BOOT** drive name changes to **CIRCUITPY**.

Adafruit Circuit Python Libraries

Download the CircuitPython library bundle per the Feather Sense guide instructions here (https://adafru.it/JBV). Unzip the files into a folder on your computer. Create a new folder on the **CIRCUITPY** drive and name it **lib**.

Open the library download folder and find the following files. Copy them into the lib folder on your **CIRCUITPY** drive.

- **adafruit_bus_device** (directory)
- **adafruit_led_animation** (directory)
- **adafruit_lsm6ds.mpy**
- **adafruit_register** (directory)
- **neopixel.mpy**



Upload Files

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

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

```
# SPDX-FileCopyrightText: 2020 Erin St. Blaine for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
```

```
LED Ukulele with Feather Sense and PropMaker Wing
Adafruit invests time and resources providing this open source code.
Please support Adafruit and open source hardware by purchasing
products from Adafruit!
Written by Erin St Blaine & Limor Fried for Adafruit Industries
Copyright (c) 2019-2020 Adafruit Industries
Licensed under the MIT license.
All text above must be included in any redistribution.

MODES:
0 = off/powerup, 1 = sound reactive, 2 = non-sound reactive, 3 = tilt
Pluck high A on the E string to toggle sound reactive mode on or off
Pluck high A♭ on the E string to cycle through the animation modes
"""

import time
import array
import digitalio
import audiobusio
import board
import neopixel

try:
    from ulab.utils import spectrogram
except ImportError:
    from ulab.scipy.signal import spectrogram
from ulab import numpy as np
from rainbowio import colorwheel
from adafruit_lsm6ds import lsm6ds33
from adafruit_led_animation.helper import PixelMap
from adafruit_led_animation.sequence import AnimationSequence
from adafruit_led_animation.group import AnimationGroup
from adafruit_led_animation.animation.sparkle import Sparkle
from adafruit_led_animation.animation.rainbow import Rainbow
from adafruit_led_animation.animation.rainbowchase import RainbowChase
from adafruit_led_animation.animation.rainbowcomet import RainbowComet
from adafruit_led_animation.animation.chase import Chase
from adafruit_led_animation.animation.comet import Comet
from adafruit_led_animation.color import (
    BLACK,
    RED,
    ORANGE,
    BLUE,
    PURPLE,
    WHITE,
)

MAX_BRIGHTNESS = 0.3  # set max brightness for sound reactive mode
NORMAL_BRIGHTNESS = 0.1  # set brightness for non-reactive mode
VOLUME_CALIBRATOR = 50  # multiplier for brightness mapping
ROCKSTAR_TILT_THRESHOLD = 200  # shake threshold
SOUND_THRESHOLD = 430000  # main strum or pluck threshold

# Set to the length in seconds for the animations
POWER_ON_DURATION = 1.3
ROCKSTAR_TILT_DURATION = 1

NUM_PIXELS = 104  # Number of pixels used in project
NEOPIXEL_PIN = board.D5
POWER_PIN = board.D10

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

i2c = board.I2C()  # uses board.SCL and board.SDA
# i2c = board.STEMMA_I2C()  # For using the built-in STEMMA QT connector on a
microcontroller
```

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


# PIXEL MAPS: Used for reordering pixels so the animations can run in different
configurations.
# My LED strips inside the neck are accidentally swapped left-right,
# so these maps also correct for that

# fmt: off
#Bottom up along both sides at once
pixel_map_reverse = PixelMap(pixels, [
    0, 103, 1, 102, 2, 101, 3, 100, 4, 99, 5, 98, 6, 97, 7, 96, 8, 95, 9, 94, 10,
    93, 11, 92, 12, 91, 13, 90, 14, 89, 15, 88, 16, 87, 17, 86, 18, 85, 19, 84, 20,
    83, 21, 82, 22, 81, 23, 80, 24, 79, 25, 78, 26, 77, 27, 76, 28, 75, 29, 74, 30,
    73, 31, 72, 32, 71, 33, 70, 34, 69, 35, 68, 36, 67, 37, 66, 38, 65, 39, 64, 40,
    63, 41, 62, 42, 61, 43, 60, 44, 59, 45, 58, 46, 57, 47, 56, 48, 55, 49, 54, 50,
    53, 51, 52,
    ], individual_pixels=True)

#Starts at the bottom and goes around clockwise
pixel_map_around = PixelMap(pixels, [
    0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
    21, 22, 23, 24, 25, 26, 27, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64,
    63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45,
    44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28,
    76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
    95, 96, 97, 98, 99, 100, 101, 102, 103,
    ], individual_pixels=True)

#Radiates from the center outwards like a starburst
pixel_map_radiate = PixelMap(pixels, [
    75, 73, 76, 27, 28, 74, 77, 26, 29, 73, 78, 25, 30, 72, 79, 24, 31, 71, 80,
    23, 32, 70, 81, 22, 33, 69, 82, 21, 34, 68, 83, 20, 35, 67, 84, 19, 36, 66,
    85, 18, 37, 65, 38, 86, 17, 64, 39, 87, 16, 63, 40, 88, 15, 62, 41, 89, 14,
    61, 42, 90, 13, 60, 43, 91, 12, 59, 44, 92, 11, 58, 45, 93, 10, 57, 46, 94,
    9, 56, 47, 95, 8, 55, 48, 96, 7, 54, 49, 97, 6, 53, 50, 98, 5, 52, 51, 99,
    4, 100, 3, 101, 2, 102, 1, 103, 0,
    ], individual_pixels=True)

#Top down along both sides at once
pixel_map_sweep = PixelMap(pixels, [
    51, 52, 50, 53, 49, 54, 48, 55, 47, 56, 46, 57, 45, 58, 44, 59, 43, 60, 42, 61,
    41, 62, 40, 63, 39, 64, 38, 65, 37, 66, 36, 67, 35, 68, 34, 69, 33, 70, 32, 71,
    31, 72, 30, 73, 29, 74, 28, 75, 27, 76, 27, 77, 26, 78, 25, 79, 24, 80, 23, 81,
    22, 82, 21, 83, 20, 84, 19, 85, 18, 86, 17, 87, 16, 88, 15, 89, 14, 90, 13, 91,
    12, 92, 11, 93, 10, 94, 9, 95, 8, 96, 7, 97, 6, 98, 5, 99, 4, 100, 3, 101, 2,
102, 1, 103, 0
    ], individual_pixels=True)

#Every other pixel, starting at the bottom and going upwards along both sides
pixel_map_skip = PixelMap(pixels, [
    0, 103, 2, 101, 4, 99, 6, 97, 8, 95, 10, 93, 12, 91, 14, 89, 16, 87, 18, 85, 20,
    83, 22, 81, 24, 79, 26, 77, 29, 74, 31, 72, 33, 70, 35, 68, 37, 66, 39, 64, 41,
    62, 43, 60, 45, 58, 47, 56, 49, 54, 51, 52,
    ], individual_pixels=True)
# fmt: on

pixel_map = [
    pixel_map_reverse,
    pixel_map_around,
    pixel_map_radiate,
    pixel_map_sweep,
    pixel_map_skip,
]

# Set up accelerometer & mic
sensor = lsm6ds33.LSM6DS33(i2c)
```

```
mic = audiobusio.PDMIn(
    board.MICROPHONE_CLOCK, board.MICROPHONE_DATA, sample_rate=16000, bit_depth=16
)

NUM_SAMPLES = 256
samples_bit = array.array("H", [0] * (NUM_SAMPLES + 3))


def power_on(duration):
    """
    Animate NeoPixels for power on.
    """
    start_time = time.monotonic()  # Save start time
    while True:
        elapsed = time.monotonic() - start_time  # Time spent
        if elapsed > duration:  # Past duration?
            break  # Stop animating
        powerup.animate()


def rockstar_tilt(duration):
    """
    Tilt animation - lightning effect with a rotating color
    :param duration: duration of the animation, in seconds (>0.0)
    """
    tilt_time = time.monotonic()  # Save start time
    while True:
        elapsed = time.monotonic() - tilt_time  # Time spent
        if elapsed > duration:  # Past duration?
            break  # Stop animating
        pixels.brightness = MAX_BRIGHTNESS
        pixels.fill(TILT_COLOR)
        pixels.show()
        time.sleep(0.01)
        pixels.fill(BLACK)
        pixels.show()
        time.sleep(0.03)
        pixels.fill(WHITE)
        pixels.show()
        time.sleep(0.02)
        pixels.fill(BLACK)
        pixels.show()
        time.sleep(0.005)
        pixels.fill(TILT_COLOR)
        pixels.show()
        time.sleep(0.01)
        pixels.fill(BLACK)
        pixels.show()
        time.sleep(0.03)


# Cusomize LED Animations  -------------------------------------------------------
powerup = RainbowComet(pixel_map[3], speed=0, tail_length=25, bounce=False)
rainbow = Rainbow(pixel_map[4], speed=0, period=6, name="rainbow", step=2.4)
rainbow_chase = RainbowChase(pixel_map[3], speed=0, size=3, spacing=15, step=10)
rainbow_chase2 = RainbowChase(pixel_map[2], speed=0, size=10, spacing=1, step=18)
chase = Chase(pixel_map[1], speed=0.1, color=RED, size=1, spacing=6)
rainbow_comet = RainbowComet(pixel_map[2], speed=0, tail_length=80, bounce=True)
rainbow_comet2 = RainbowComet(
    pixel_map[0], speed=0, tail_length=104, colorwheel_offset=80, bounce=True
)
rainbow_comet3 = RainbowComet(
    pixel_map[1], speed=0, tail_length=25, colorwheel_offset=80, step=4,
bounce=False
)
strum = RainbowComet(
    pixel_map[3], speed=0, tail_length=25, bounce=False, colorwheel_offset=50,
step=4
)
```

```
lava = Comet(pixel_map[3], speed=0.01, color=ORANGE, tail_length=40, bounce=False)
sparkle = Sparkle(pixel_map[4], speed=0.01, color=BLUE, num_sparkles=10)
sparkle2 = Sparkle(pixel_map[1], speed=0.05, color=PURPLE, num_sparkles=4)

# Animations Playlist - reorder as desired. AnimationGroups play at the same time
animations = AnimationSequence(
    rainbow,
    rainbow_chase,
    rainbow_chase2,
    chase,
    lava,
    rainbow_comet,
    rainbow_comet2,
    AnimationGroup(
        sparkle,
        strum,
    ),
    AnimationGroup(
        sparkle2,
        rainbow_comet3,
    ),
    auto_clear=True,
    auto_reset=True,
)


MODE = 0
LASTMODE = 1  # start up in sound reactive mode
i = 0

# Main loop
while True:
    i = (i + 0.5) % 256  # run from 0 to 255
    TILT_COLOR = colorwheel(i)
    if MODE == 0:  # If currently off...
        enable.value = True
        power_on(POWER_ON_DURATION)  # Power up!
        MODE = LASTMODE

    elif MODE >= 1:  # If not OFF MODE...
        mic.record(samples_bit, len(samples_bit))
        samples = np.array(samples_bit[3:])
        spectrum = spectrogram(samples)
        spectrum = spectrum[:128]
        spectrum[0] = 0
        spectrum[1] = 0
        peak_idx = np.argmax(spectrum)
        peak_freq = peak_idx * 16000 / 256
        #        print((peak_idx, peak_freq, spectrum[peak_idx]))
        magnitude = spectrum[peak_idx]
        #         time.sleep(1)
        if peak_freq == 812.50 and magnitude > SOUND_THRESHOLD:
            animations.next()
            time.sleep(1)
        if peak_freq == 875 and magnitude > SOUND_THRESHOLD:
            if MODE == 1:
                MODE = 2
                print("mode = 2")
                LASTMODE = 2
                time.sleep(1)
            elif MODE == 2:
                MODE = 1
                print("mode = 1")
                LASTMODE = 1
                time.sleep(1)
        # Read accelerometer
        x, y, z = sensor.acceleration
        accel_total = x * x + y * y  # x=tilt, y=rotate
        #         print (accel_total)
```

```
        if accel_total > ROCKSTAR_TILT_THRESHOLD:
            MODE = 3
            print("Tilted: ", accel_total)
        if MODE == 1:
            VOLUME = magnitude / (VOLUME_CALIBRATOR * 100000)
            if VOLUME > MAX_BRIGHTNESS:
                VOLUME = MAX_BRIGHTNESS
            #             print(VOLUME)
            pixels.brightness = VOLUME
            #             time.sleep(2)
            animations.animate()
        elif MODE == 2:
            pixels.brightness = NORMAL_BRIGHTNESS
            animations.animate()
        elif MODE == 3:
            rockstar_tilt(ROCKSTAR_TILT_DURATION)
            MODE = LASTMODE
```

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).

Edit Variables

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 your instrument.

```
MAX_BRIGHTNESS = 0.3      # set max brightness for sound reactive mode
NORMAL_BRIGHTNESS = 0.1   # set brightness for non-reactive mode
VOLUME_CALIBRATOR = 50    # multiplier for brightness mapping
ROCKSTAR_TILT_THRESHOLD = 200 # shake threshold
SOUND_THRESHOLD = 430000  # main strum or pluck threshold
```

Set **NORMAL_BRIGHTNESS** to a number between 0 (off) and 1 (full brightness). My uke has a whole lot of densely packed lights, so 10% brightness is really plenty bright -- much more and I'd be blinding people.

MAX_BRIGHTNESS is the cutoff number for sound reactive mode. I've turned it up a little higher to give more variety with my strum, and since it's pulsing along with the volume, it's not quite as blinding.

The volume, tilt, and sound thresholds can also be adjusted here. Further down in the code we'll look at how to read these values from your uke to calibrate it to your own playing style.

```
# Set to the length in seconds for the animations
POWER_ON_DURATION = 1.3
ROCKSTAR_TILT_DURATION = 1
```

Adjust these variables to change the duration of the power-up or tilt animations.

```
NUM_PIXELS = 104  # Number of pixels used in project
NEOPIXEL_PIN = board.D5
POWER_PIN = board.D10
```

Change `NUM_PIXELS` to reflect the actual number of LEDs in your project.

Pixel Mapping

A neat function of the LED Animations library is Pixel Mapping (https://adafru.it/Me-): the ability to run the animations along pixels in whatever order you'd like. This is useful for a couple of reasons.

First, when I wrangled the LED strips inside my ukulele's neck, I somehow managed to get them reversed left-to-right. A light traveling along my four strips in sequence would go around the left side of the body, then up the right side of the neck and back down the left side, then around the right side of the body. This just looked messy. I'd already glued the strips in place, so there was no fixing this without a whole lot of destruction.

Pixel maps to the rescue! We can tell the code to run animations on the pixels in any order we want, so I was able to fix my mistake just by typing in a string of numbers.

Second, with our newfound pixel mapping power, we can set up a list of pixel maps and have a lot more control over the layout of our animations. I set up five different pixel maps:

1. **pixel_map_reverse**: animations start at the base of the ukulele and travel upwards along both sides to the top, symmetrically
2. **pixel_map_around**: animations start at the bottom and travel around the ukulele in a "circle", correcting for my swapped strips inside the neck
3. **pixel_map_radiate**: animations start where the neck and body meet, and radiate out from the center of the ukulele symmetrically
4. **pixel_map_sweep**: basically the inverse of **pixel_map_reverse**, going top-down along both sides
5. **pixel_map_skip**: Same as **pixel_map_reverse** but lighting up every other pixel instead of all pixels

After setting up my maps, I made a list of all of them called `pixel_map`. Further down in the code where I'm setting up animations, I can call a member of the list (i.e `pixel_map[4]`) to make the LED animations run in that configuration.

```
pixel_map = [
    pixel_map_reverse,
    pixel_map_around,
    pixel_map_radiate,
    pixel_map_sweep,
    pixel_map_skip,
]
```

You don't have to use the pixel maps -- you can simply call `pixels.show()` instead of `pixel_map[0].show()` in all the animations if you just want to keep it simple. But using pixel maps can add a whole new dimension to the animations in the LED Animations library.

Customize the Animations & Animation Playlist

Further down in the code, around line 181, you'll find the list of animations. I'm using several animations imported from the LED Animations library including:

- **Rainbow**: a lovely rainbow animation, with customizable spread and speed
- **Comet**: sends a single color "comet" of adjustable length down the strip
- **RainbowComet**: sends a rainbow colored "comet" of adjustable length along the strip
- **RainbowChase**: blocks of LEDs chasing each other around in varying colors
- **Sparkle**: twinkly lights!

There are more animation styles available - check the LED Animations Library Guide (https://adafru.it/LZF) for a comprehensive list and use instructions.

To add more animations:

1. Import them at the top of the code
2. Define and customize them here
3. Add them to the playlist at line 202.

If you want to use different colors, remember to scroll up to the `import` section at the top of the code and add your colors to the color list. Here is a list of all available colors in the library (https://adafru.it/Ld-).

```
# Cusomize LED Animations  ------------------------------------------------------
powerup = RainbowComet(pixel_map[3], speed=0, tail_length=25, bounce=False)
rainbow = Rainbow(pixel_map[4], speed=0, period=6, name="rainbow", step=2.4)
```

```
rainbow_chase = RainbowChase(pixel_map[3], speed=0, size=3, spacing=15, step=10)
rainbow_chase2 = RainbowChase(pixel_map[2], speed=0, size=10, spacing=1, step=18)
chase = Chase(pixel_map[1], speed=0.1, color=RED, size=1, spacing=6)
rainbow_comet = RainbowComet(pixel_map[2], speed=0, tail_length=80, bounce=True)
rainbow_comet2 = RainbowComet(
    pixel_map[0], speed=0, tail_length=104, colorwheel_offset=80, bounce=True
    )
rainbow_comet3 = RainbowComet(
    pixel_map[1], speed=0, tail_length=25, colorwheel_offset=80, step=4,
bounce=False
    )
strum = RainbowComet(
    pixel_map[3], speed=0, tail_length=25, bounce=False, colorwheel_offset=50,
step=4
    )
lava = Comet(pixel_map[3], speed=0.1, color=ORANGE, tail_length=70, bounce=False)
sparkle = Sparkle(pixel_map[4], speed=0.01, color=BLUE, num_sparkles=10)
sparkle2 = Sparkle(pixel_map[1], speed=0.05, color=PURPLE, num_sparkles=4)

# Animations Playlist - reorder as desired. AnimationGroups play at the same time
animations = AnimationSequence(
    rainbow,
    rainbow_chase,
    rainbow_chase2,
    chase,
    lava,
    rainbow_comet,
    rainbow_comet2,
    AnimationGroup(
        sparkle,
        strum,
        ),
    AnimationGroup(
        sparkle2,
        rainbow_comet3,
        ),
    auto_clear=True,
    auto_reset=True,
)
```

Sensitivity

I set threshold variables at the top of the code, but if my values aren't quite what you're looking for, you may want to listen to what your ukulele is doing and adjust the numbers to fit. You can use the REPL in the Mu editor as a serial monitor to print out the sound and movement levels of your ukulele.

In Mu, click the **Serial** button and press **D** to enter the REPL. More about using the REPL here. (https://adafru.it/Awz)

To find a good `SOUND_THRESHOLD`, look for this code around line 245:

```
peak_idx = np.argmax(spectrum)
peak_freq = peak_idx * 16000 / 256
# print((peak_idx, peak_freq, spectrum[peak_idx]))
magnitude = spectrum[peak_idx]
# time.sleep(1)
```

With your ukulele plugged into your computer, uncomment the `print` and `time.sleep` lines and save your code. In the Serial window, you'll see a series of 3 numbers printing out. The third number is `spectrum[peak_idx]` and represents the overall volume of your sound. I chose 43000 for my `SOUND_THRESHOLD` since that's pretty high inside the range between my "quiet" numbers and "loud" numbers. Play with some different numbers until you like the action.

Don't forget to comment out `time.sleep` again when you're done adjusting or your animations will run really slowly.

Note Sensing

The middle number of the three is `peak_freq`. This number represents the pitch, or frequency, of the note being strummed. The code normalizes the frequencies, "squishing" them into levels broken up every 62.5hz. This means you'll get pretty consistent numbers when you strum a particular note, even if your ukulele is a bit out of tune. A high A note played on your ukulele's E string comes out at 875hz, and an A♭ reads at 812.5hz.

This is fun because now we can tell the Feather to run some code whenever it hears that particular pitch. I chose the two highest notes on the instrument because they rarely get played (at least with my novice ability) and won't get strummed accidentally.

High A toggles sound-reactive mode on and off, and high A♭ cycles through the animation modes. Feel free to add more modes centered around different notes!

```
if peak_freq == 875 and magnitude &gt; SOUND_THRESHOLD:
            animations.next()
            time.sleep(1)
        if peak_freq == 812.50 and magnitude &gt; SOUND_THRESHOLD:
            if MODE == 1:
                MODE = 2
                print("mode = 2")
                LASTMODE = 2
                time.sleep(1)
            elif MODE == 2:
                MODE = 1
                print("mode = 1")
                LASTMODE = 1
                time.sleep(1)
```

Tilt Mode / Accelerometer Adjustment

Down around line 256 you'll find a similar `print` line for your accelerometer readout. Uncomment `print(accel_total)` and shake your ukulele around to get some data on which to base your `ROCKSTAR_TILT_THRESHOLD` value.

```
# Read accelerometer
x, y, z = sensor.acceleration
accel_total = x * x + y * y # x=tilt, y=rotate
# print (accel_total)
if accel_total &gt; ROCKSTAR_TILT_THRESHOLD:
    MODE = 3
    print("Tilted: ", accel_total)
```

Troubleshooting

If it doesn't seem to be working, here are a few things to try:

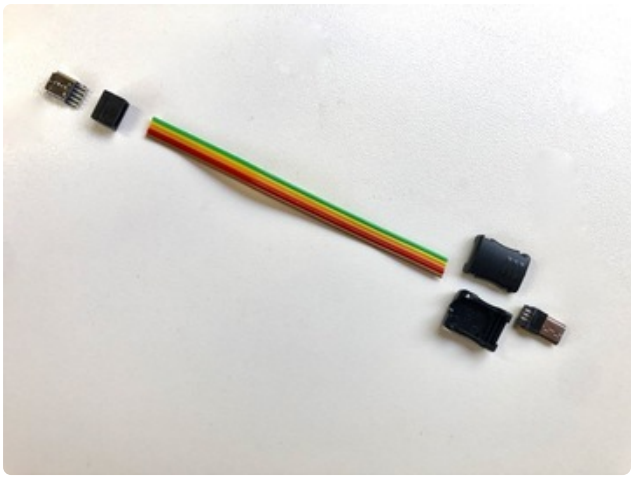
1. Double check you have all the correct libraries installed. Some of the names are really similar -- make sure you've got the right ones.
2. Try reinstalling CircuitPython again
3. Open the **REPL** in the Mu editor by clicking the "serial" button. Press **<ctrl>D**. This will error-check your code and let you know what line number may be the problem. More about the REPL here (https://adafru.it/Awz).

More tips and tricks can be found on the Intro to CircuitPython guide (https://adafru.it/BIM) and the Feather Sense guide (https://adafru.it/L6d).

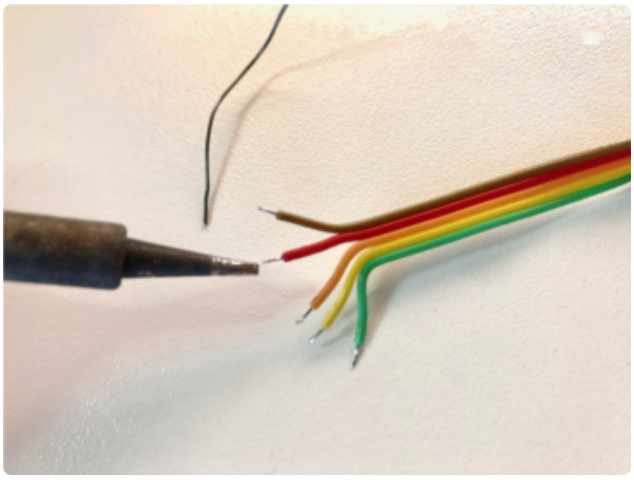
USB Extension Assembly

First we'll build a super low-profile USB extension cable so we can put our charging / programming port on the bottom of the ukulele, where the strap connector would go. This way we can just plug it in to charge the battery without having to take anything apart.

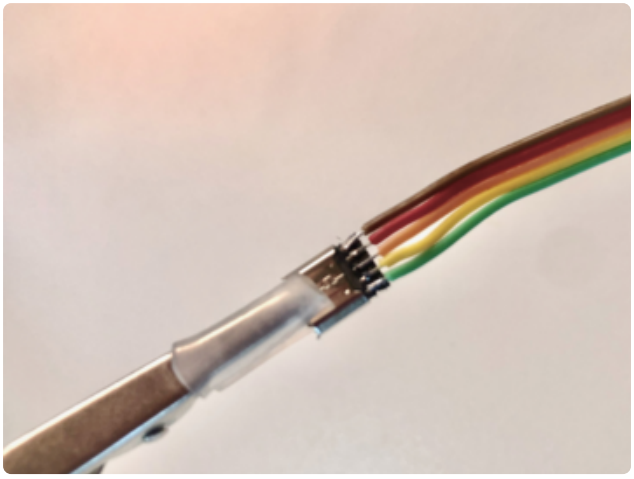
We'll need a 5-strand ribbon cable. I'm using a multicolored one for clarity. Cut the ribbon cable to around 3-4 inches long.



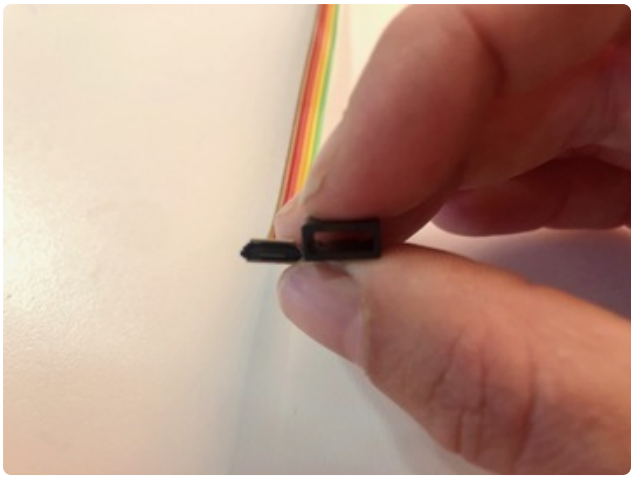
Make sure you've got all the pieces. You'll need a female connector and snap-fit plastic sheath, 4" of ribbon cable, and your male connector and 2-piece plastic sheath.



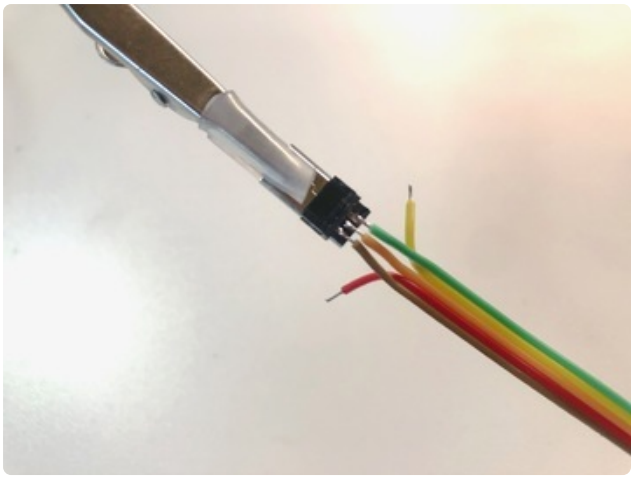
Separate the wires at both ends of your ribbon cable and tin all five wires. Trim just a smidgen off the tips of the wires so they're nice and tidy.

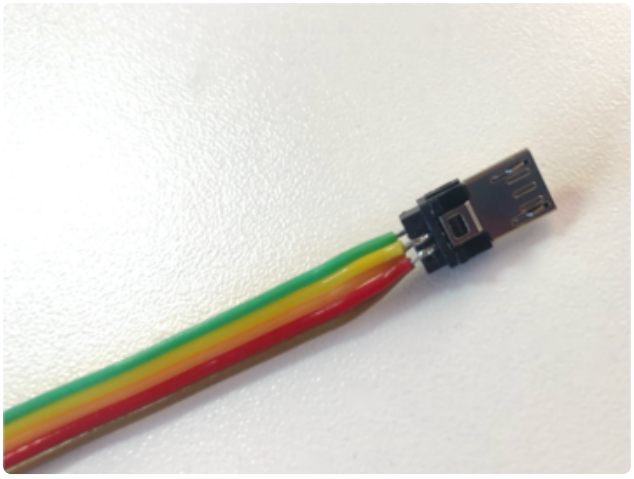


Tin the pads on the female USB connector and solder all five wires to the five pads.



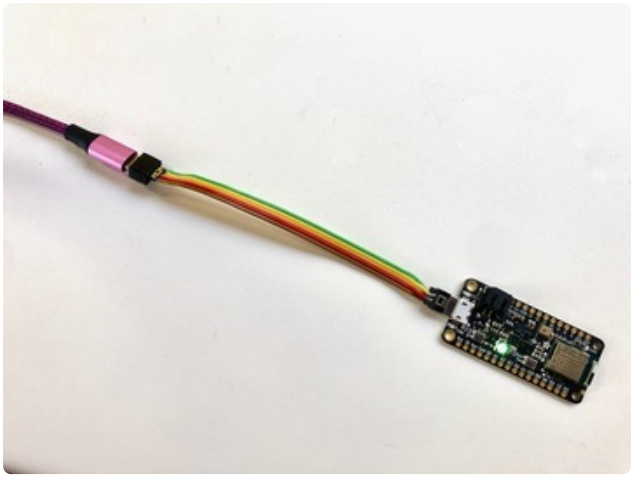
Slip the plastic cover onto the wire and snap it into place around the USB connector.



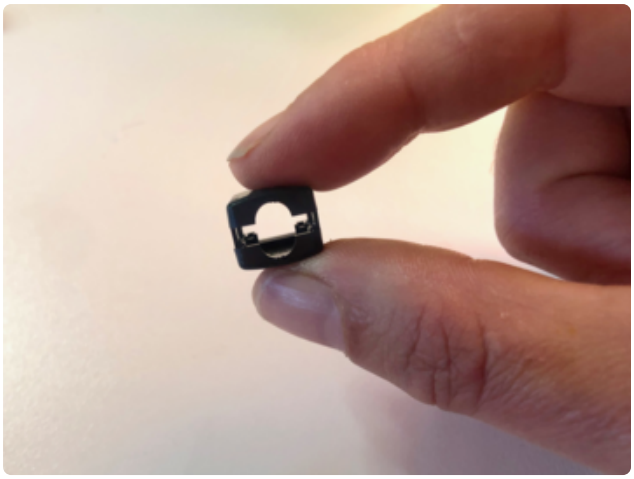


Plug the male side into the female side for a moment, so you're 100% sure you've got both sides the same way up.

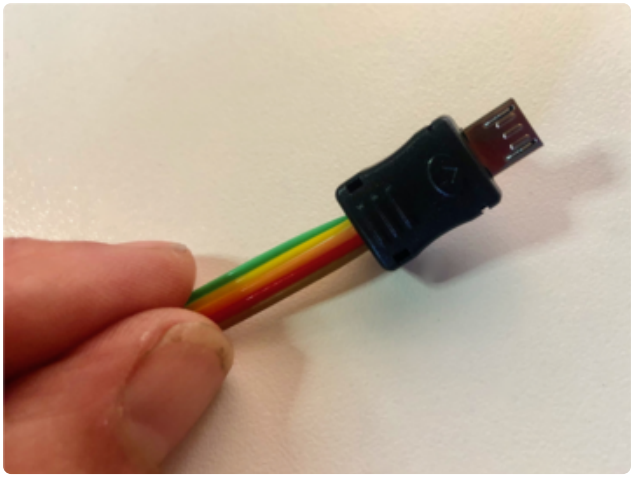
Once you're sure it's aligned right, tin the pads on the male connector on both sides. Solder the wires as shown - the first, third, and fifth wire go on the the 3-pad side, and the second and fourth go on the 2-pad side.



Plug your Feather into your computer's USB port via your newly soldered cable. Press the reset button and be sure your Feather shows up as a drive on your computer, in order to be sure the data lines and power lines are all in working order.



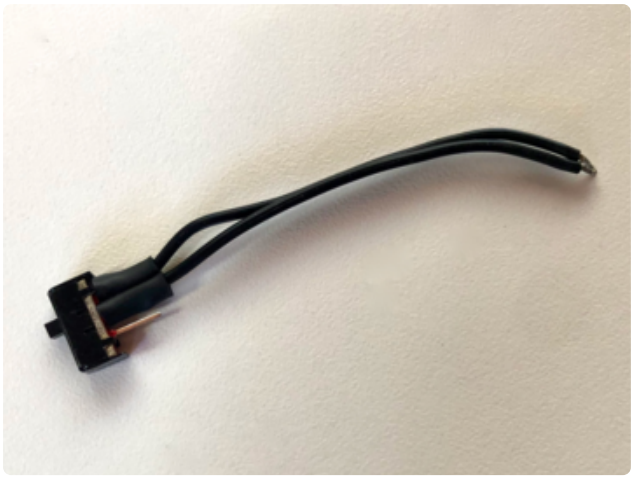
Once you're sure your soldering is 100% solid, clip the housing onto the male end of your cable.



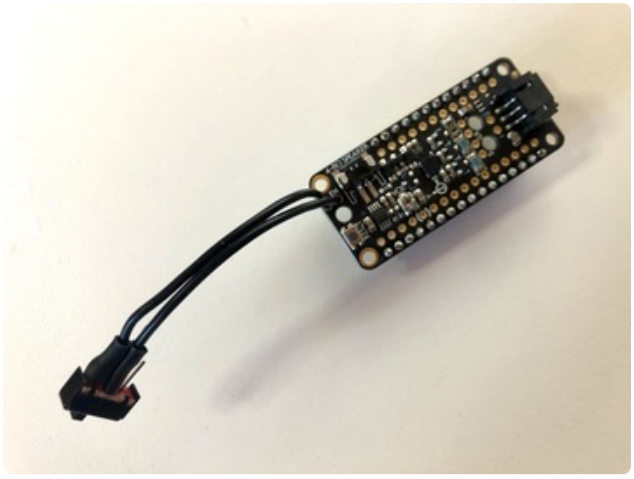
Squeeze a bead of hot glue across the exposed contacts on the female end to keep them nice and secure as well.

Feather Assembly

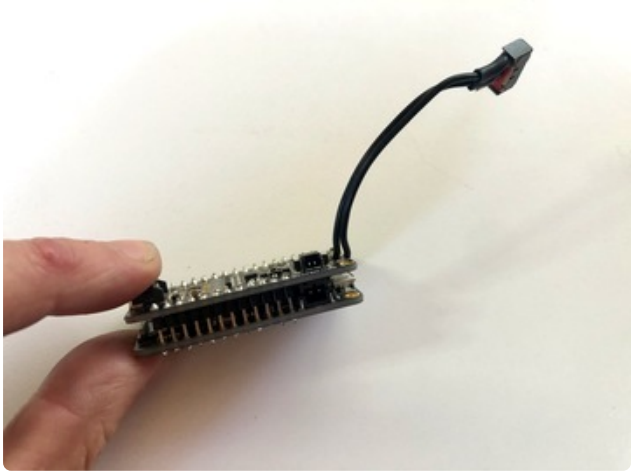
Grab your 4-wire ribbon cable and find the striped wire. Leave that one alone and separate the black wire on the other edge from the ribbon cable. We'll use this single wire for our on/off switch, and the 3-wire ribbon that's left to wire our NeoPixels.



Solder a 3" wire to the middle and one of the side legs of your on/off switch (it doesn't matter which side leg). Cover with heat shrink.



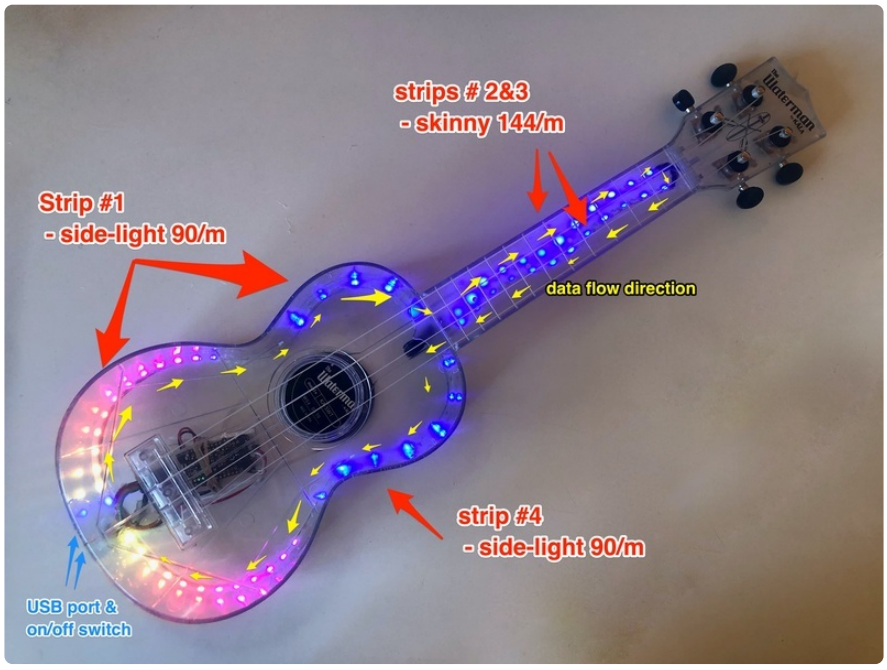
Solder the two wires into the **EN** and **G** holes on your PropMaker Wing as shown. It doesn't matter which wire goes to which hole.



Stack your PropMaker Wing on top of your Feather Sense so all the headers fit into the holes. Solder this little sandwich in place.

Plug in your homemade USB cable, and use that to plug into your computer. Flick your switch on and off to make sure that's working. Also make sure you can see your **CIRCUITPY** drive through your homemade USB cable -- if anything's not working, now is the time to find out and fix it since these connections will become inaccessible later on.

NeoPixel Strip Assembly



Layout

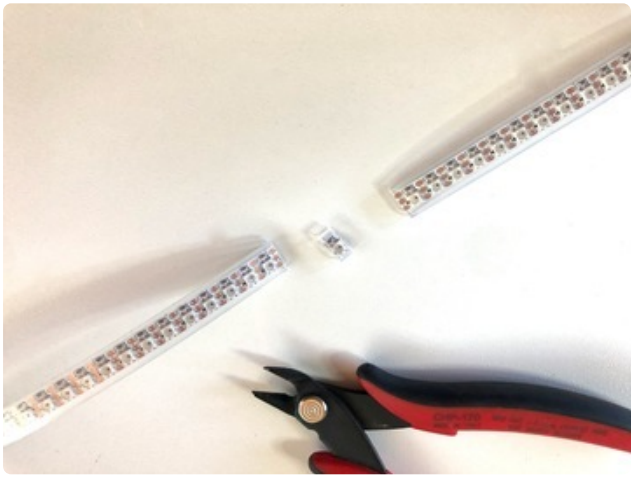
This ukulele uses side-lit NeoPixels (90/m) around the inside edge of the body. Side-lit pixels are great for fitting into curves. They're assembled differently than regular NeoPixels, which flex front-to-back with the pixels shining straight out. These pixels shine to the side, so the flex direction of the strip can curve beautifully along the perimeter of the ukulele body. I have 28 pixels per side.

For the neck, I'm using two strips of Skinny Neopixels (144/m). These are going straight and so can lay flat (more or less) with the light facing straight forward. I have 24 pixels in each strip.

The Feather Sense and PropMaker Wing are attached to the battery in a tidy stack and stuck to the inside back of the ukulele with industrial velcro. Our homemade USB extension plugs into the Feather's USB port and threads through to the outside of the body -- this board has onboard battery charging, so we can charge up our battery as well as update the programming via the USB port.

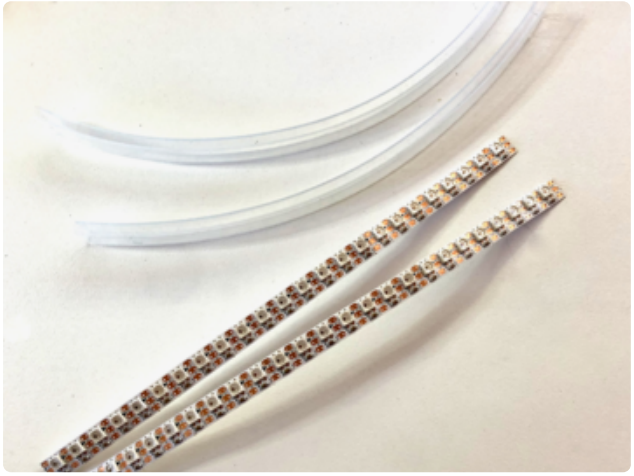
Our on/off switch will also thread through a hole next to the USB port.

Solder NeoPixel Strips

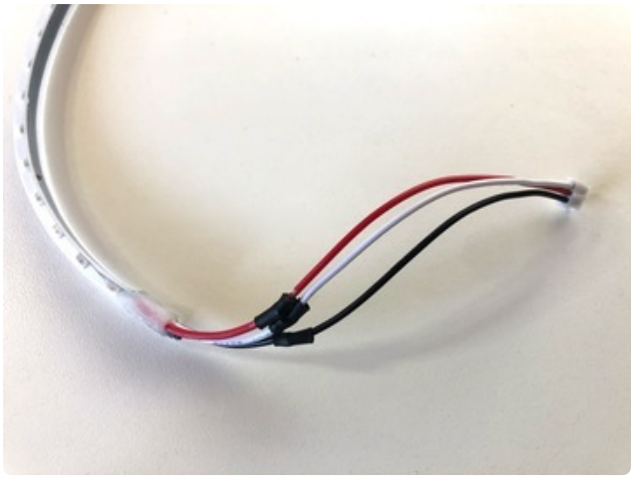


Measure your strips to your ukulele, count out your pixels and cut your strips to length. These high-density strips can be tricky to solder to, so I like to use the "sacrificial pixel" method, leaving a full solder pad on each side while sacrificing the pixel in between.

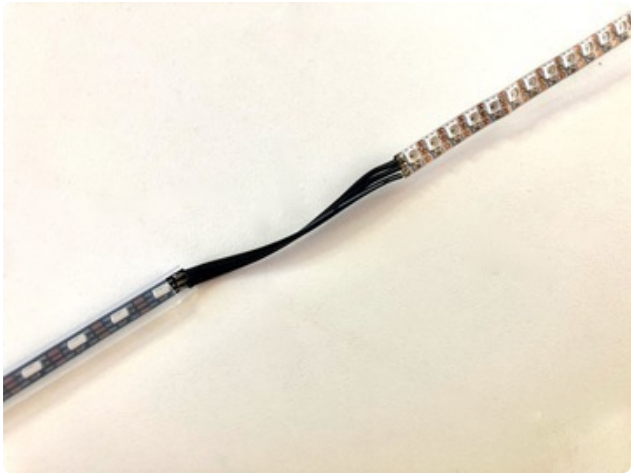
Learn more about how this works in our How To Solder NeoPixels Guide (https://adafru.it/Mf6).



There's not a lot of space inside the neck, so I pulled my Skinny strips out of their silicone casing to better fit them inside. I left the side-light strip cases on.



Grab your 3-pin STEMMA connector and solder it to the color-coded wires on the **IN** end of strip #1 (side-light strip). You can clip off any remaining extra wires.



The next strip in line (#2) is the skinny strip that goes up the neck. Use a 3" piece of ribbon cable to connect it to the side-light strip.

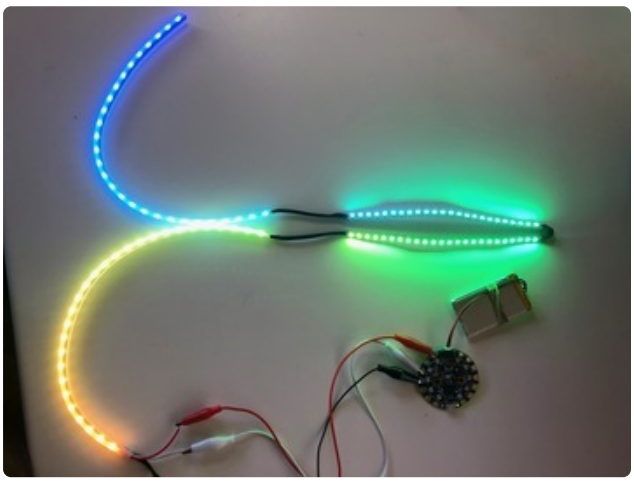
Here's where you need to be extra careful - the pads on the side-light strips are not laid out in the same order as on the skinny strips. I had to put a twist in the wire to get the correct pins connected to the correct pins. Look closely -- you want to match + to +, **G** to **G**, and **DO** (on the side light strip) to **DI** (on the skinny strip).

Next we'll solder the wires for the u-bend in the pixels up by the headstock.

Cut a short piece of 3-wire ribbon cable -- about an inch long.

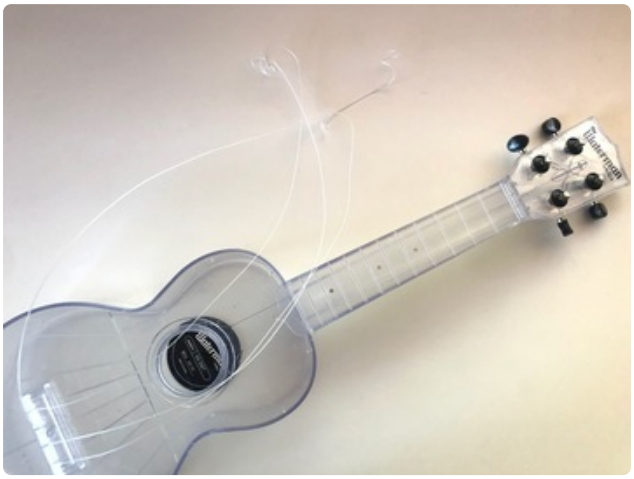
Solder the **OUT** end of one skinny strip (#2 in the diagram above) to the **IN** end of your second skinny strip (#3), matching the pads - + to +, **DI** to **DO**, and **G** to **G**. Bend the strips so they lie next to each other as neatly as possible. Put a little hot glue over the solder pads so they're less likely to come apart during assembly.

Solder strip #4 to the **OUT** end of your second skinny strip to complete the circuit around the ukulele. Don't forget to match the pads with a twist in your ribbon cable.



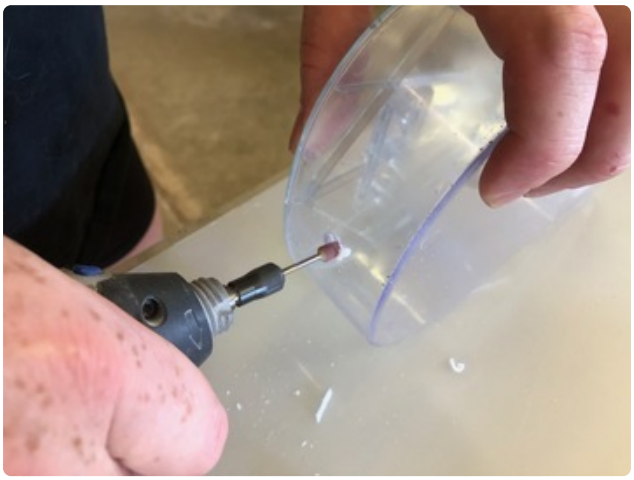
Plug your NeoPixels into your PropMaker and test to be sure everything is working. If you wired something up wrong or backwards, now is the time to catch any mistakes!

Project Assembly



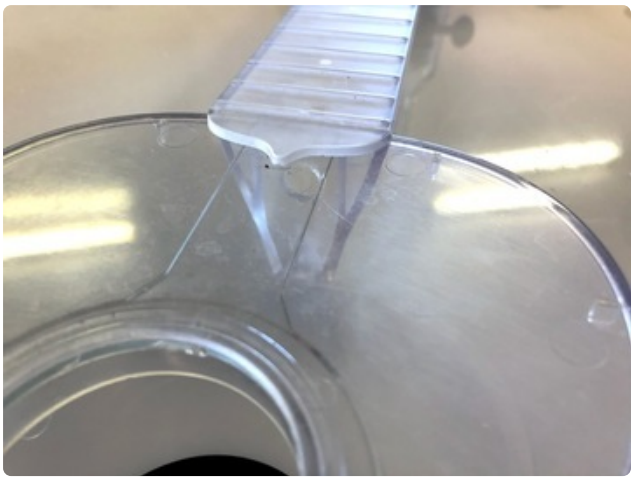
Take the strings off your ukulele. Put them someplace safe where they won't get mixed up.

Drill a pilot hole and then a 3/8" hole in the bottom center of the body, for the USB port. Clean up any stray plastic with a rotary tool or sand paper.

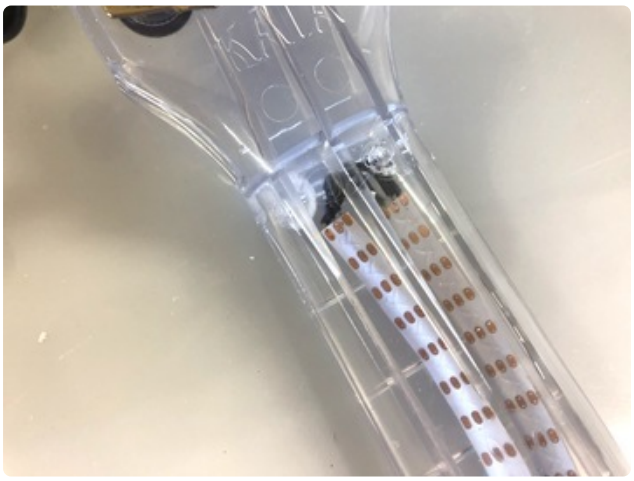


Drill a second hole just below this one for the on/off switch. Don't worry too much about making it fit exactly - you want the hole a bit bigger than the switch and port so we can get them through later.

For the hole into the neck, I'm using a 3/8" spade bit with a long drill extension. Slip it through the bottom hole and carefully drill through into the neck.



This hole needs to be big enough to squeeze the two NeoPixel Skinny strips through. 3/8" seems to be just perfect.



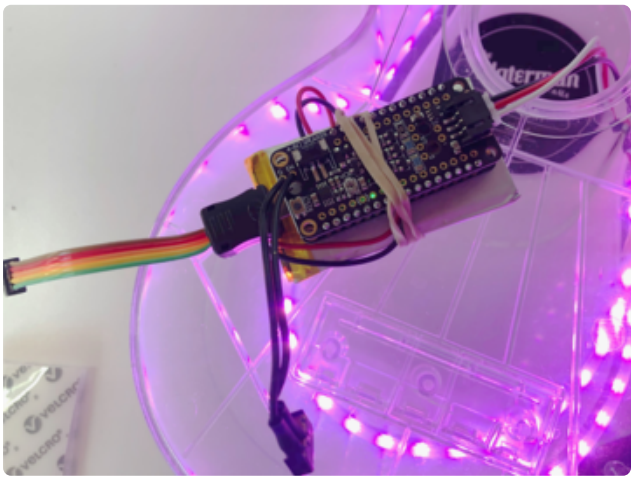
Now it's time for the fiddly part of placing the light strips inside the ukulele. I made judicious use of needlenose pliers, zip ties, q-tips, and anything else I could find to wrangle the strips into place.

I had the most success using a bit of coathanger wire, straightened out and slipped through the bottom hole and then the neck hole, giving me a straight shot at pushing the wires up as far as they would go.

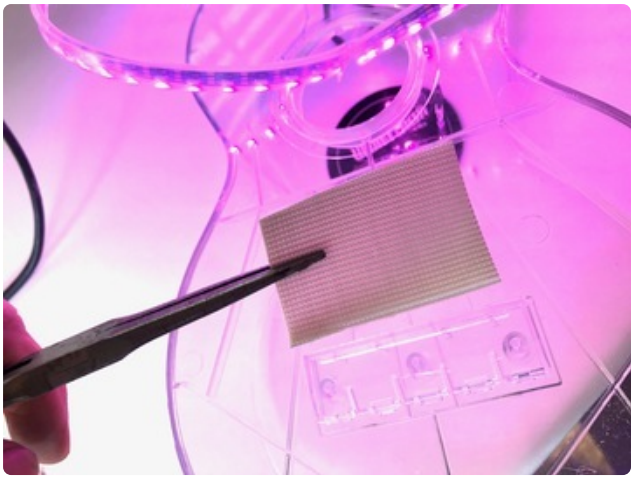
To hold the neck wires in place, I drilled a tiny hole on the back of the neck and emptied a small tube of krazy glue into it.

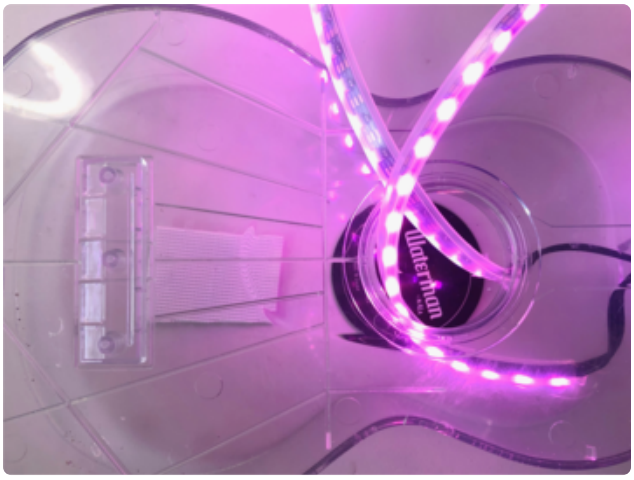
My neck wires don't lay quite flat, but I actually like the wavy effect I get from them quite a lot.

During this process, my neck strips somehow got reversed and placed on opposite sides from what I intended. If this happens to you, no worries - you can fix it in the code with pixel mapping. Check the Software page for instructions on how to do this.

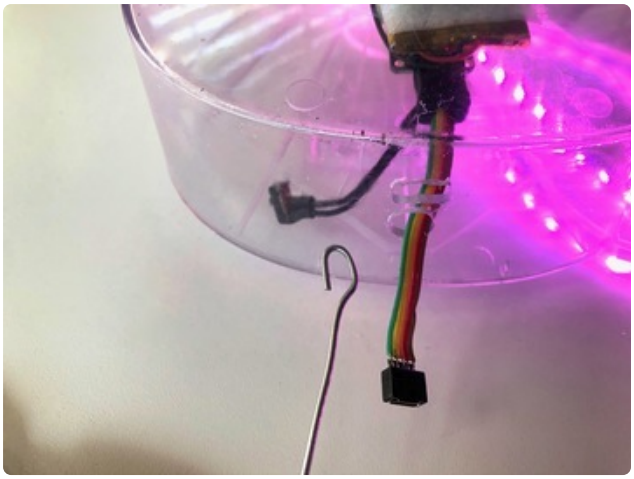


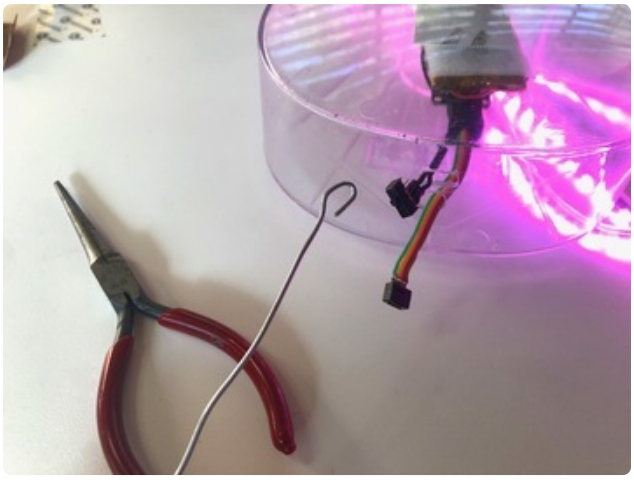
Once the neck wires are in place, plug your battery and USB cable into your Feather stack. Use a rubber band or zip tie to stack the battery underneath the Feather. Make it look as tidy as possible, since it will show a bit through your ukulele.





Cut a piece of industrial velcro to the same size as the battery. Stick one side to the battery and use needlenose pliers to stick the other side to the inside back of the ukulele, as far down as you can reach through the sound hole.

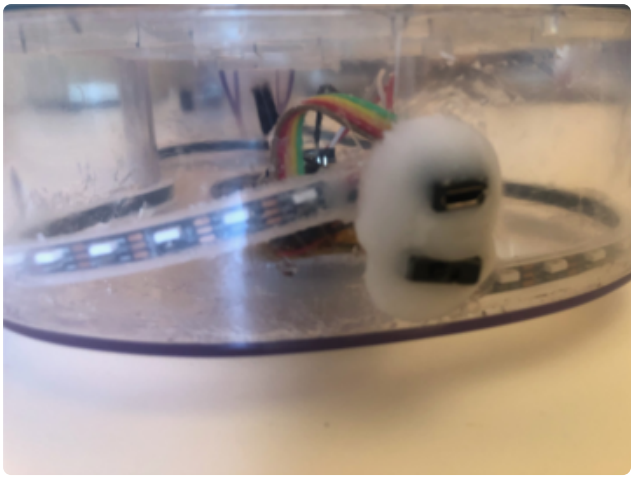


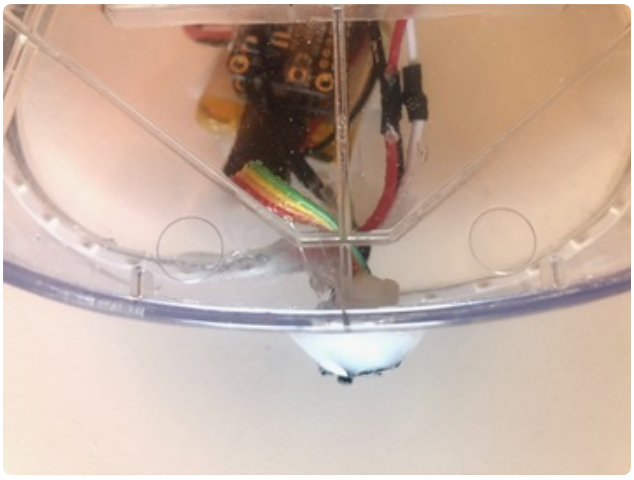


Slip the feather in through the sound hole with the USB cable facing the bottom of the ukulele and press it down onto the velcro.

Now it's time to use your coathanger tool again. Bend it into a hook with your pliers and reach through the holes you drilled to pull the USB port and the switch to the outside. Take your time and be careful not to pull your wire connections too hard or they might break!

Next we'll mount these two components in place using a bit of thermoplastic. Use a heat gun to melt a bit of this down until it gets warm and squishy. It turns transparent when it's moldable, but be careful - it can get hot!





Squish some thermoplastic around the USB port and the switch and smooth it down so your components are more-or-less flush with the bottom of the ukulele. Be sure to push some of the plastic through the hole to the inside, so when it hardens it will grab in place. These components will need to stand up to pushing, so be sure they're pretty solidly mounted - and this thermoplastic seems to reject any and all types of glue, so a physical blob on both the inside and outside is your best bet.

Finally, glue the side-light NeoPixel strips along the inside edges of the ukulele body using silicone adhesive. Silicone adhesive is just about the only thing that will stick to the silicone coating on the LEDs.

If you don't have this or can't get a hold of this, you can take the silicone casing off and glue the strips down with E6000 or another glue, but if you're leaving the protective cases on, you'll need to use silicone glue.

I found my coathanger tool to be the most effective thing for spreading glue and pressing the NeoPixels into it. Wipe off any excess glue right away; this stuff dries rubbery and will be hard to get off later on.

Let everything dry, then shake your ukulele around a bunch to make sure nothing rattles before putting the strings back on.

Play It

Turn your ukulele on with the switch and a power-up animation will run. The ukulele starts up in sound-reactive mode with a rainbow animation.

The sound reactive mode changes the brightness of the LEDs based on the volume of your playing. It works with any of the animation modes. Play loud and the lights will be bright, play quietly or stop playing and they'll go dim or off. To toggle sound-reactive mode on or off, pluck the highest note possible on your ukulele - a high A on the E string.

To cycle between animation modes, play a half-step or whole-step down on the same string - an A flat or high G note will move to the next animation in your playlist.

To trigger rockstar mode, tilt or shake the ukulele quickly and you'll see a lightning bolt flash!