# Table of Contents

**Overview**  
- Parts  

**Circuit Diagram**  
- Adafruit Library for Fritzing  
- Wired Connections  

**CircuitPython**  
- CircuitPython Quickstart  
- Safe Mode  
- Flash Resetting UF2  

**Code**  
- Upload the Code and Libraries to the RP2040 Prop-Maker Feather  
- How the CircuitPython Code Works  
- Time of Flight and RGB Matrix Setup  
- DisplayIO on the RGB Matrix  
- Sound Effects  
- NeoPixels and Servo  
- Time Keeping  
- The Loop  

**3D Printing**  
- Parts List  
- Slice with Settings for PLA material  
- Share, Make, Remix  

**Assemble**  
- Remove Tape Cover  
- Matrix Power Cable  
- Assemble Matrix FeatherWing  
- Matrix Cutout  
- LEDs eyes  
- LED eye holes  
- Component cutouts  
- Sensor cutout and covers  
- Servo horn  
- Raven Assemble  
- Attach Raven head  
- Cable Conduits  
- Terminal connections
Overview

Build an interactive tombstone for Halloween with Adafruit's RP2040 PropMaker Feather.

This ghoulish prop features lights and sounds but also an RGB LED matrix and a servo controlled Raven.

The servo, sound effects, and NeoPixels are triggered whenever a time of flight sensor detects something nearby.

The RGB LED matrix displays scrolling text messages to warn or welcome would-be trick-or-treaters.
The tombstone is an off-the-shelf styrofoam prop. It’s easy to work with so we were able to cut out channels and recesses for the electronics and wiring.

The RP2040 PropMaker Feather does most of the work and the Matrix FeatherWing makes it easy to plug in an RGB LED matrix.

CircuitPython has great support for RGB matrices and the time of flight sensor. It’s got adjustable parameters so you can customize just about every attribute of this project.

Parts

Adafruit RP2040 Prop-Maker Feather with I2S Audio Amplifier
The Adafruit Feather series gives you lots of options for a small, portable, rechargeable microcontroller board. By picking a feather and stacking on a FeatherWing you can create...
https://www.adafruit.com/product/5768
Adafruit RGB Matrix Featherwing Kit
Ahoy! It's time to create a dazzling light up project with our new RGB Matrix FeatherWing. Now you can quickly and easily create...
https://www.adafruit.com/product/3036

64x32 RGB LED Matrix - 4mm pitch
Bring a little bit of Times Square into your home with this sweet 64 x 32 square RGB LED matrix panel. These panels are normally used to make video walls, here in New York we see them...
https://www.adafruit.com/product/2278

Adafruit VL53L4CD Time of Flight Distance Sensor - ~1 to 1300mm
The Adafruit VL53L4CD Time of Flight Sensor is another great Time of Flight distance sensor from ST in the VL5 series of chips, this one is great for...
https://www.adafruit.com/product/5396

Speaker - 40mm Diameter - 4 Ohm 3 Watt
Hear the good news! This speaker is a great addition to any audio project where you need a 4 Ohm impedance and 3W or less of power. At 40mm diameter it...
https://www.adafruit.com/product/3968
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Description</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 x</td>
<td><strong>5V 4A (4000mA) switching power supply</strong></td>
<td><a href="https://www.adafruit.com/product/1466">https://www.adafruit.com/product/1466</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>Black LED Diffusion Acrylic Panel</strong></td>
<td><a href="https://www.adafruit.com/product/4749">https://www.adafruit.com/product/4749</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>Micro servo - TowerPro SG92R</strong></td>
<td><a href="https://www.adafruit.com/product/169">https://www.adafruit.com/product/169</a></td>
</tr>
<tr>
<td>2 x</td>
<td><strong>JST PH 2-Pin Cable – Plug Header 200mm</strong></td>
<td><a href="https://www.adafruit.com/product/3814">https://www.adafruit.com/product/3814</a></td>
</tr>
<tr>
<td>2 x</td>
<td><strong>JST PH 2-Pin Cable - Socket Connector 100mm</strong></td>
<td><a href="https://www.adafruit.com/product/261">https://www.adafruit.com/product/261</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>JST PH 2mm 3-Pin Socket</strong></td>
<td><a href="https://www.adafruit.com/product/4046">https://www.adafruit.com/product/4046</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>STEMMA JST PH 2mm 3-Pin Plug</strong></td>
<td><a href="https://www.adafruit.com/product/3893">https://www.adafruit.com/product/3893</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>Silicone Cover Stranded-Core Ribbon Cable</strong></td>
<td><a href="https://www.adafruit.com/product/3889">https://www.adafruit.com/product/3889</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>Split Wire Loom Tubing Wire Conduit</strong></td>
<td><a href="https://amzn.to/3sTADHq">https://amzn.to/3sTADHq</a></td>
</tr>
<tr>
<td>1 x</td>
<td><strong>3.5 Ft Gargoyle Tombstone</strong></td>
<td><a href="https://www.spirithalloween.com/product/3-5-ft-gargoyle-tombstone-decorations/131272.uts">https://www.spirithalloween.com/product/3-5-ft-gargoyle-tombstone-decorations/131272.uts</a></td>
</tr>
</tbody>
</table>

This is the foam tombstone we used - feel free to buy one of your choice or make one with materials at hand.

©Adafruit Industries
Circuit Diagram

The diagram below provides a general visual reference for wiring of the components once you get to the Assembly page. This diagram was created using the software package [Fritzing](https://www.fritzing.org/).

Adafruit Library for Fritzing

Adafruit uses the Adafruit Fritzing parts library to create circuit diagrams for projects. You can download the library or just grab individual parts. Get the library and parts from [GitHub - Adafruit Fritzing Parts](https://github.com/adafruit/Adafruit_Parts).
Wired Connections

- The NeoPixels and speaker are each connected to pins on the screw block terminal
- Sensor plugs into the STEMMA port
- Servo connects to the headers on the Feather
- The Prop-Maker Feather seats on top of the RGB Matrix Wing

CircuitPython

CircuitPython () is a derivative of MicroPython () designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop software downloads. Simply copy and edit files on the CIRCUITPY drive to iterate.

CircuitPython Quickstart

Follow this step-by-step to quickly get CircuitPython running on your board.

Download the latest version of CircuitPython for this board via circuitpython.org

Click the link above to download the latest CircuitPython UF2 file.

Save it wherever is convenient for you.
To enter the bootloader, hold down the BOOT/BOOTSEL button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the reset button (highlighted in blue above). Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!

If the drive does not appear, release all the buttons, and then repeat the process above.

You can also start with your board unplugged from USB, press and hold the BOOTSEL button (highlighted in red above), continue to hold it while plugging it into USB, and wait for the drive to appear before releasing the button.

A lot of people end up using charge-only USB cables and it is very frustrating! Make sure you have a USB cable you know is good for data sync.
You will see a new disk drive appear called RPI-RP2.

Drag the adafruit_circuitpython_etc.uf2 file to RPI-RP2.

The RPI-RP2 drive will disappear and a new disk drive called CIRCUITPY will appear.

That's it, you're done! :)

Safe Mode

You want to edit your code.py or modify the files on your CIRCUITPY drive, but find that you can't. Perhaps your board has gotten into a state where CIRCUITPY is read-only. You may have turned off the CIRCUITPY drive altogether. Whatever the reason, safe mode can help.
Safe mode in CircuitPython does not run any user code on startup, and disables auto-reload. This means a few things. First, safe mode bypasses any code in boot.py (where you can set CIRCUITPY read-only or turn it off completely). Second, it does not run the code in code.py. And finally, it does not automatically soft-reload when data is written to the CIRCUITPY drive.

Therefore, whatever you may have done to put your board in a non-interactive state, safe mode gives you the opportunity to correct it without losing all of the data on the CIRCUITPY drive.

Entering Safe Mode

To enter safe mode when using CircuitPython, plug in your board or hit reset (highlighted in red above). Immediately after the board starts up or resets, it waits 1000ms. On some boards, the onboard status LED (highlighted in green above) will blink yellow during that time. If you press reset during that 1000ms, the board will start up in safe mode. It can be difficult to react to the yellow LED, so you may want to think of it simply as a slow double click of the reset button. (Remember, a fast double click of reset enters the bootloader.)

In Safe Mode

If you successfully enter safe mode on CircuitPython, the LED will intermittently blink yellow three times.

If you connect to the serial console, you'll find the following message.

```
Auto-reload is off.
Running in safe mode! Not running saved code.
CircuitPython is in safe mode because you pressed the reset button during boot. Press again to exit safe mode.
Press any key to enter the REPL. Use CTRL-D to reload.
```

You can now edit the contents of the CIRCUITPY drive. Remember, your code will not run until you press the reset button, or unplug and plug in your board, to get out of safe mode.

Flash Resetting UF2

If your board ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try loading this 'nuke' UF2 which will do a 'deep
clean’ on your Flash Memory. You will lose all the files on the board, but at least you'll
be able to revive it! After loading this UF2, follow the steps above to re-install
CircuitPython.

Download flash erasing "nuke" UF2

Code

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

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

```python
import os
import random
import board
import audiocore
import audiobusio
import audiomixer
import pwmio
from digitalio import DigitalInOut, Direction
import neopixel
from adafruit_motor import servo
from adafruit_ticks import ticks_ms, ticks_add, ticks_diff
from adafruit_led_animation.animation.pulse import Pulse
from adafruit_led_animation.color import RED, BLACK, GREEN
import adafruit_display_text.label
import displayio
import framebufferio
import rgbmatrix
import terminalio
import adafruit_vl53l4cd

distance_trigger = 90  # cm
text="Here lies Fred"
text_color = 0xff0000
# how often to check for a new trigger from ToF
pause_time = 30  # seconds
# speed for scrolling the text on the matrix
scroll_time = 0.1  # seconds

displayio.release_displays()

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

i2c = board.I2C()
vl53 = adafruit_vl53l4cd.VL53L4CD(i2c)

vl53.inter_measurement = 0
vl53.timing_budget = 200

matrix = rgbmatrix.RGBMatrix(
    width=64, height=32, bit_depth=4,
    addr_pins=[board.D25, board.D24, board.A3, board.A2],
    clock_pin=board.D13, latch_pin=board.D0, output_enable_pin=board.D1)

display = framebufferio.FramebufferDisplay(matrix, auto_refresh=True)

linel = adafruit_display_text.label.Label(terminalio.FONT,
    color=text_color,
    text=text)
linel.x = 1
linel.y = 14

def scroll(line):
    line.x = line.x - 1
    line_width = line.bounding_box[2]
    if line.x < -line_width:
        line.x = display.width

g = displayio.Group()
g.append(linel)
display.show(g)

wavs = []
for filename in os.listdir('/tomb_sounds'):
    if filename.lower().endswith('.wav') and not filename.startswith('.'):
        wavs.append('/tomb_sounds/' + filename)

audio = audiobusio.I2SOut(board.I2S_BIT_CLOCK, board.I2S_WORD_SELECT,
    board.I2S_DATA)
mixer = audiomixer.Mixer(voice_count=1, sample_rate=22050, channel_count=1,
    bits_per_sample=16, samples_signed=True, buffer_size=32768)
mixer.voice[0].level = 1
audio.play(mixer)
wav_length = len(wavs) - 1

def open_audio(num):
    n = wavs[num]
    f = open(n, "rb")
    w = audiocore.WaveFile(f)
    return w

PIXEL_PIN = board.EXTERNAL_NEOPIXELS
BRIGHTNESS = 0.3
NUM_PIXELS = 2
PIXELS = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, auto_write=True)
pulse = Pulse(PIXELS, speed=0.05, color=RED, period=3)
COLORS = [RED, GREEN, BLACK]

SERVO_PIN = board.EXTERNAL_SERVO
PWM = pwmio.PWMOut(SERVO_PIN, duty_cycle=2 ** 15, frequency=50)
SERVO = servo.Servo(PWM)
SERVO.angle = 0
clock = ticks_ms()
the_time = 5000
x = 0
scroll_clock = ticks_ms()
scroll_time = int(scroll_time * 1000)
pause_clock = ticks_ms()
pause_time = pause_time * 1000
pause = False
vl53.start_ranging()

while True:
    vl53.clear_interrupt()
    if vl53.distance < distance_trigger:
        if not pause:
            print("Distance: {} cm".format(vl53.distance))
            SERVO.angle = 90
            wave = open_audio(random.randint(0, wav_length))
            mixer.voice[0].play(wave)
            while mixer.playing:
                pulse.color = COLORS[x]
                pulse.animate()
                if ticks_diff(ticks_ms(), scroll_clock) >= scroll_time:
                    scroll(line1)
                    display.refresh(minimum_frames_per_second=0)
                    scroll_clock = ticks_add(scroll_clock, scroll_time)
                x = (x + 1) % 2
                pause = True
                print("paused")
            pause_clock = ticks_add(pause_clock, pause_time)
    else:
        if ticks_diff(ticks_ms(), pause_clock) >= pause_time:
            print("back to sensing")
            pause = False
            print("still paused")
        if ticks_diff(ticks_ms(), scroll_clock) >= scroll_time:
            print("Distance: {} cm".format(vl53.distance))
            scroll(line1)
            display.refresh(minimum_frames_per_second=0)
            scroll_clock = ticks_add(scroll_clock, scroll_time)
            SERVO.angle = 0
            pulse.color = COLORS[2]
            pulse.animate()

Upload the Code and Libraries to the RP2040 Prop-Maker Feather

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

- lib folder
- tomb_sounds folder
Your RP2040 Prop-Maker Feather CIRCUITPY drive should look like this after copying the lib folder, tomb_sounds folder and the code.py file.

How the CircuitPython Code Works

At the top of the code are a few parameters that you can modify to customize the tombstone. `distance_trigger` is the distance in centimeters where the time of flight sensor will trigger. `text` is the text that will scroll across the RGB matrix. `text_color` is the color of the scrolling text. `pause_time` is the time in seconds that the time of flight sensor will delay after being initially triggered. `scroll_time` is the speed in seconds that the text scrolls across the RGB matrix.

```python
distance_trigger = 90 # cm
text="Here lies Fred"
text_color = 0xff0000
# how often to check for a new trigger from ToF
pause_time = 30 # seconds
# speed for scrolling the text on the matrix
scroll_time = 0.1 # seconds
```

Time of Flight and RGB Matrix Setup

The time of flight sensor is instantiated over I2C, and the RGB matrix object is set up to use the FeatherWing pinout for the RP2040.
i2c = board.I2C()
vl53 = adafruit_vl53l4cd.VL53L4CD(i2c)
vl53.inter_measurement = 0
vl53.timing_budget = 200

matrix = rgbmatrix.RGBMatrix(
    width=64, height=32, bit_depth=4,
    addr_pins=[board.D25, board.D24, board.A3, board.A2],
    clock_pin=board.D13, latch_pin=board.D0, output_enable_pin=board.D1)

DisplayIO on the RGB Matrix

The `matrix` is passed as a `FramebufferDisplay` so that `displayio` can be used with the RGB matrix. The text is created as a `Label`. The `scroll` function moves the text by one pixel at a time across the display.

display = framebufferio.FramebufferDisplay(matrix, auto_refresh=True)

line1 = adafruit_display_text.label.Label(
    terminalio.FONT,
    color=text_color,
    text=text)
line1.x = 1
line1.y = 14
def scroll(line):
    line.x = line.x - 1
    line_width = line.bounding_box[2]
    if line.x &lt; -line_width:
        line.x = display.width

        g = displayio.Group()
g.append(line1)

display.show(g)

Sound Effects

The audio files in the `/tomb_sounds` folder are added to the `wavs` list. If you change or add more sound effects to the folder, they will be added to the list with no additional modification in the code. Audio playback is handled with the I2S amp on the Feather. Playback is routed through a `Mixer` object to allow for software volume control. The `open_audio` function takes an index location for a wave file in the `wavs` list and preps it for playback.

wavs = []
for filename in os.listdir('/tomb_sounds'):
    if filename.lower().endswith('.wav') and not filename.startswith('.':
        wavs.append('/tomb_sounds/' + filename)

audio = audiobusio.I2SOut(board.I2S_BIT_CLOCK, board.I2S_WORD_SELECT, board.I2S_DATA)
mixer = audiomixer.Mixer(voice_count=1, sample_rate=22050, channel_count=1, 
    bits_per_sample=16, samples_signed=True, buffer_size=32768)

mixer.voice[0].level = 1
audio.play(mixer)
wav_length = len(wavs) - 1

def open_audio(num):
    n = wavs[num]
    f = open(n, "rb")
    w = audiocore.WaveFile(f)
    return w

NeoPixels and Servo

The NeoPixels and servo are setup using the external NeoPixel and servo pins.

PIXEL_PIN = board.EXTERNAL_NEOPIXELS
BRIGHTNESS = 0.3
NUM_PIXELS = 2

PIXELS = neopixel.NeoPixel(PIXEL_PIN, NUM_PIXELS, auto_write=True)
pulse = Pulse(PIXELS, speed=0.05, color=RED, period=3)
COLORS = [RED, GREEN, BLACK]

SERVO_PIN = board.EXTERNAL_SERVO
PWM = pwmio.PWMOut(SERVO_PIN, duty_cycle=2 ** 15, frequency=50)
SERVO = servo.Servo(PWM)
SERVO.angle = 0

Time Keeping

The `ticks` library is used for time tracking without blocking in the loop. The `scroll_clock` handles the delay for scrolling the text, and the `pause_clock` adds a delay for triggering the time of flight sensor. `ticks` uses milliseconds for time keeping. The `pause_time` and `scroll_time` variables at the top of the code are multiplied by 1000 to convert them from seconds to milliseconds.

x = 0
scroll_clock = ticks_ms()
scroll_time = int(scroll_time * 1000)
pause_clock = ticks_ms()
pause_time = pause_time * 1000
pause = False

The Loop

In the loop, if the time of flight sensor is triggered, then the servo moves to 90 degrees, and a randomized audio file from the `wavs` array is played through the speaker. While the audio is playing, the NeoPixels will pulse in either red or green.
The \(x\) variable allows for switching between the two colors each time the time of flight sensor is triggered.

The \texttt{pause\_clock} is used to avoid multiple triggers back to back in case someone is standing in front of the tombstone for a while. When the time of flight sensor is initially triggered, \texttt{pause} is set to True. It is not reset to False until the \texttt{pause\_clock} is greater than the \texttt{pause\_time} delay.

```python
vl53.clear\_interrupt()
    if vl53.distance < distance\_trigger:
        if not pause:
            print("Distance: {} cm".format(vl53.distance))
            SERVO\_angle = 90
            wave = open\_audio(random\_randint(0, wav\_length))
            mixer\_voice[0]\_play(wave)
            while mixer\_playing:
                pulse\_color = COLORS[x]
                pulse\_animate()
                if ticks\_diff(ticks\_ms(), scroll\_clock) \&\&= scroll\_time:
                    scroll(line1)
                    display\_refresh(minimum\_frames\_per\_second=0)
                    scroll\_clock = ticks\_add(scroll\_clock, scroll\_time)
            x = (x + 1) \% 2
            pause = True
            print("paused")
            pause\_clock = ticks\_add(pause\_clock, pause\_time)
        else:
            if ticks\_diff(ticks\_ms(), pause\_clock) \&\&= pause\_time:
                print("back to sensing")
                pause = False
                print("Still paused")
```

The \texttt{scroll\_clock} allows the matrix to scroll the text by one pixel without blocking the other aspects of the code. Every time the \texttt{scroll\_time} passes, the \texttt{scroll} function is called and moves the text by one pixel to the left.

```python
if ticks\_diff(ticks\_ms(), scroll\_clock) \&\&= scroll\_time:
    print("Distance: {} cm".format(vl53.distance))
    scroll(line1)
    scroll\_clock = ticks\_add(scroll\_clock, scroll\_time)
```

The servo and NeoPixels are reset after being triggered by the time of flight sensor. The servo is moved back to 0 degrees, and the NeoPixel color is changed to \texttt{BLACK} or off.

```python
SERVO\_angle = 0
pulse\_color = COLORS[2]
pulse\_animate()
```
3D Printing

Parts List

STL files for 3D printing are oriented to print "as-is" on FDM style machines. Parts are designed to 3D print without any support material. Original design source may be downloaded using the links below.

Download STLs

Slice with Settings for PLA material

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

- PLA filament 200c extruder
- 0.2 layer height
- 10% gyroid infill
- 60mm/s print speed
- 60c heated bed

Share, Make, Remix

This Halloween Crow was originally designed by danman () on https://www.thingiverse.com/thing:1091364 ()

The parts have been modified to fit all of the electronics and they are available to download.
Assemble

Remove Tape Cover
The Time of Flight breakout has a protective sticker over the sensor that will need to be removed.

Matrix Power Cable
Measure and shorten the included power cable to fit behind the Matrix Display.
Assemble Matrix FeatherWing

Connect power cables to the Matrix Wing: red wire to the + terminal and black wire to the - terminal on the Matrix Wing.

The Matrix Wing is set up with headers for plugging onto the bottom of the RP2040 PropMaker Feather.

The two then plug into the HUB75 socket on the back of the RGB LED Matrix.
Matrix Cutout
Cut a frame out of the foam to fit the Matrix display.

You can use the Template below to print out and trace.

Hot wire foam cutter is used to create the opening for the matrix display.

A piece of black acrylic is placed first to help diffuse the matrix display.

64x32_Matrix_frame_cut-out.jpg.zip
LEDs eyes
Two NeoPixel buttons are wired together for adding a glowing effect to the demon atop the tombstone.

Measure and cut extension cables to reach the LEDs and speaker.

Use socket and plug ends to help easily disconnect components.

LED eye holes
The eyes are then carved out for LEDs to shine through.
Component cutouts
A speaker is fitted into another recess with a press fit cover to keep it in place.
Sensor cutout and covers
Similarly, the time of flight breakout has an opening for the sensor with an accompanying cover.

Servo horn
A piece of foam-core is secured to a servo horn that fits into the head of the 3D printed Raven.

Use foam core to build the servo horn. Trace the Raven neck and cut in half. Attach the foam cutout to the included servo horn.
Raven Assemble
Trace a piece of black foam core around the neck and attach to the bird's neck. Make cut outs for the LEDs and servo.

Route the LED and servo wires through the body.
Attach Raven head
Press fit the head to the foam cutout.

Route wires to the back of the bird.
Cable Conduits
Bundle the LED and servo wires together with split loom tubing conduit.
Terminal connections

Connect the speaker and LEDs to the Prop-Maker Feather terminals.

The Time of Flight Sensor connects to the Feather's on-board stemma QT port.

Everything gets powered by a 5V 4A switching power supply so it can run all night.