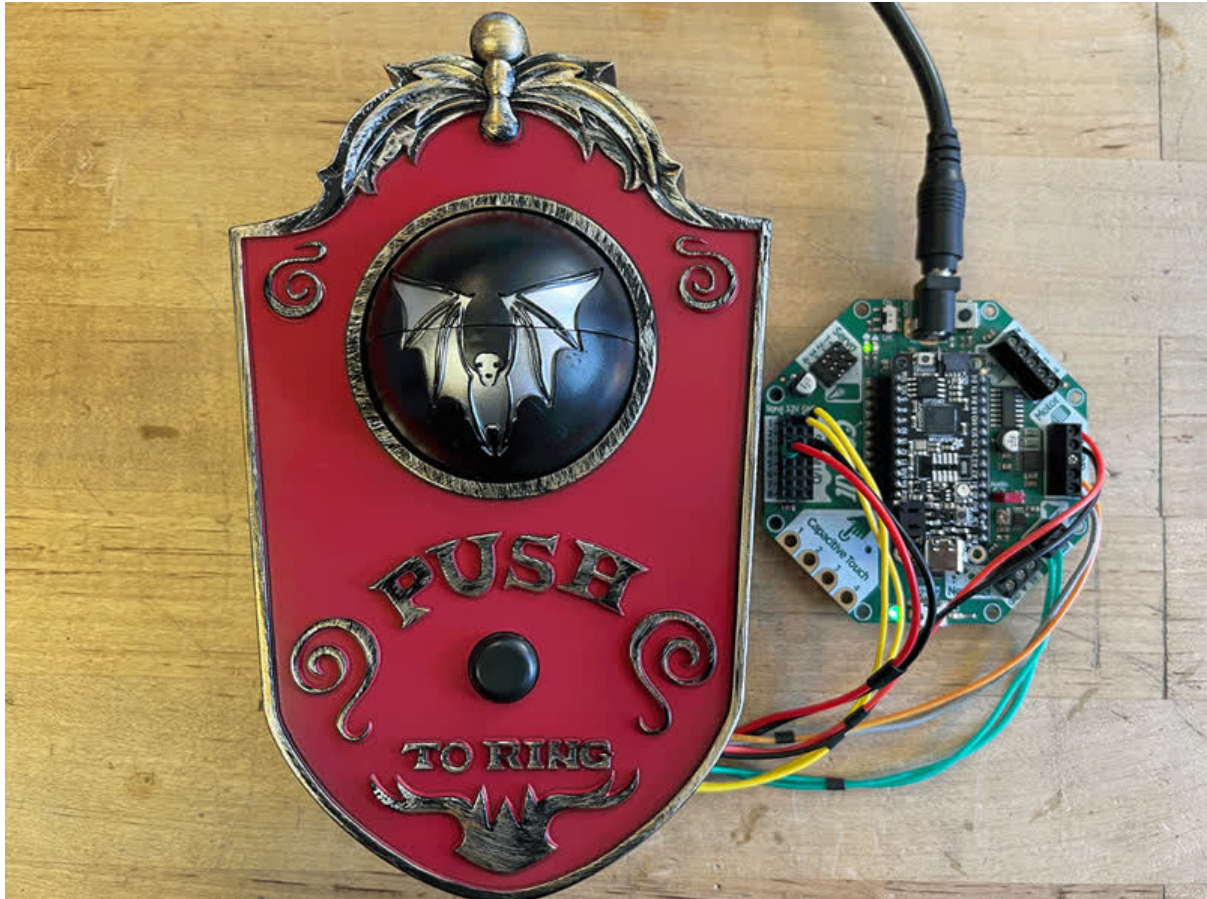




Hacking Holiday Animatronics

Created by John Park



<https://learn.adafruit.com/hacking-holiday-animatronics>

Last updated on 2024-06-03 03:30:05 PM EDT

Table of Contents

Overview	3
• Parts	
Animatronic Anatomy	4
• Motors	
• Voltage	
• Current Draw	
• Speaker Impedance	
• Input	
• LED Light	
Controlling Animatronics	11
• Teardown Documentation	
• LED Wiring	
• Final Touches	
Install CircuitPython	18
• CircuitPython Quickstart	
• Safe Mode	
• Flash Resetting UF2	
Code the Animatronic Eyeball Doorbell	22
• Text Editor	
• Download the Project Bundle	
• Use The Hacked Animatronic	
• Create Your Own Audio Files	
• Make Your Own	

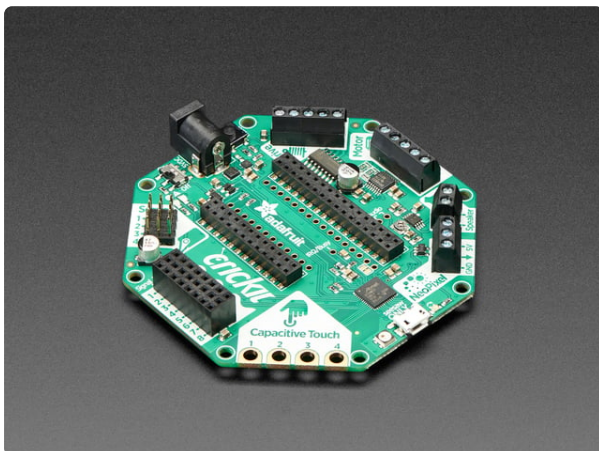
Overview

There's an abundance of fun, well-made, and inexpensive animatronic holiday gizmos available at hardware stores, big-box stores, specialty shops, and online. They work great right out of the package, but sometimes you're looking for that extra bit of personalization, so this guide is all about swapping out the brains of animatronics with your own CircuitPython-based controller.

Why do it? Besides "for the fun of it", building your own animatronic controller allows you to switch out any audio files you like, fine tune the animation, and even introduce new capabilities such as wireless Bluetooth operation, ultrasonic distance sensors, and more.

Only modify battery-powered/low-power DC devices. If your animatronic runs on AC mains power just leave it alone unless you're experienced with high-voltage electronics!

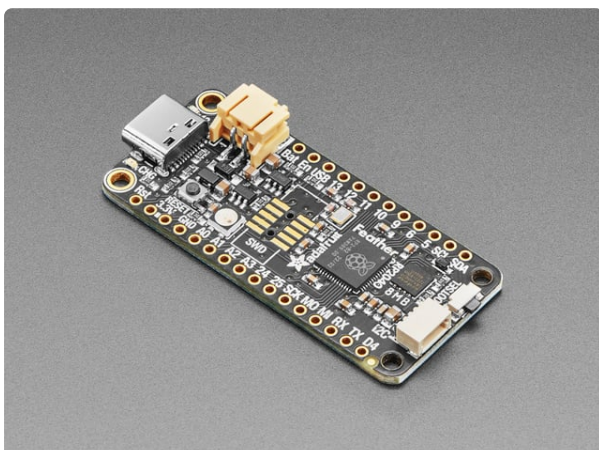
Parts



[Adafruit CRICKIT FeatherWing for any Feather](https://www.adafruit.com/product/3343)

Sometimes we wonder if robotics engineers ever watch movies. If they did, they'd know that making robots into servants always ends up in a robot rebellion. Why even go down that...

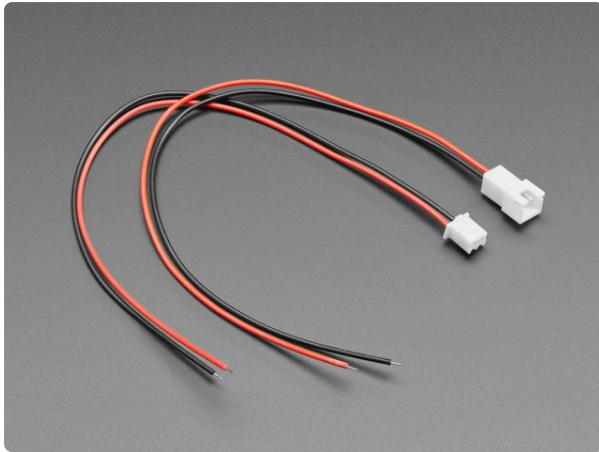
<https://www.adafruit.com/product/3343>



[Adafruit Feather RP2040](https://www.adafruit.com/product/4884)

A new chip means a new Feather, and the Raspberry Pi RP2040 is no exception. When we saw this chip we thought "this chip is going to be awesome when we give it the Feather..."

<https://www.adafruit.com/product/4884>



2.5mm Pitch 2-pin Cable Matching Pair - JST XH compatible

These solid 2.5mm pitch connector cable pairs are great when you need something that can carry a couple amps of current, easy to connect and disconnect, and is just-about 0.1"...

<https://www.adafruit.com/product/4872>

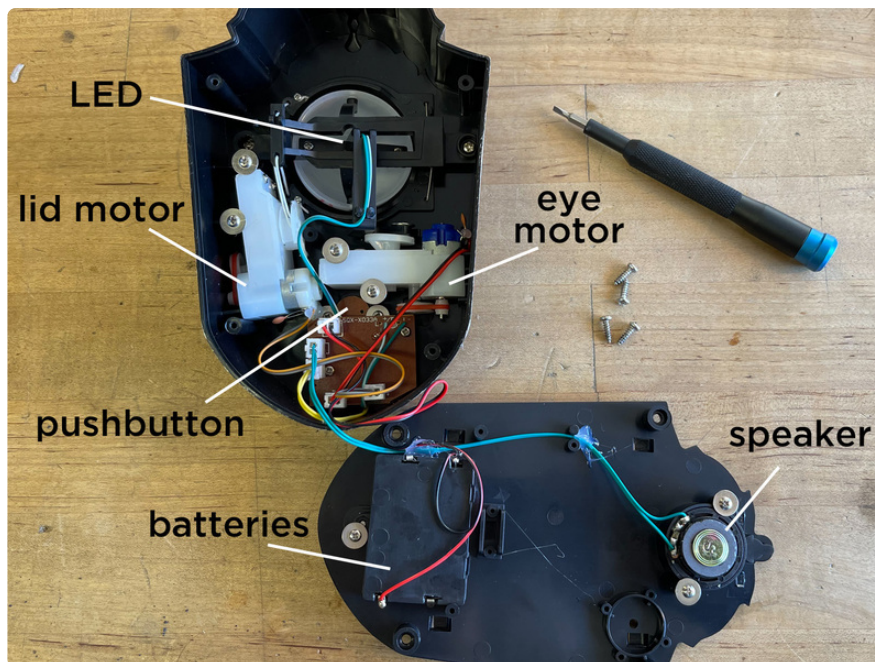


5V 4A (4000mA) switching power supply - UL Listed

Need a lot of 5V power? This switching supply gives a clean regulated 5V output at up to 4 Amps (4000mA). 110 or 240 input, so it works in any country. The plugs are "US..."

<https://www.adafruit.com/product/1466>

Animatronic Anatomy



The typical holiday animatronic will contain inputs, such as buttons and light/motion sensors, and outputs, such as motors, lights, and speakers.

Learning to identify the parts of your animatronic will aid you in transplanting your own control system.

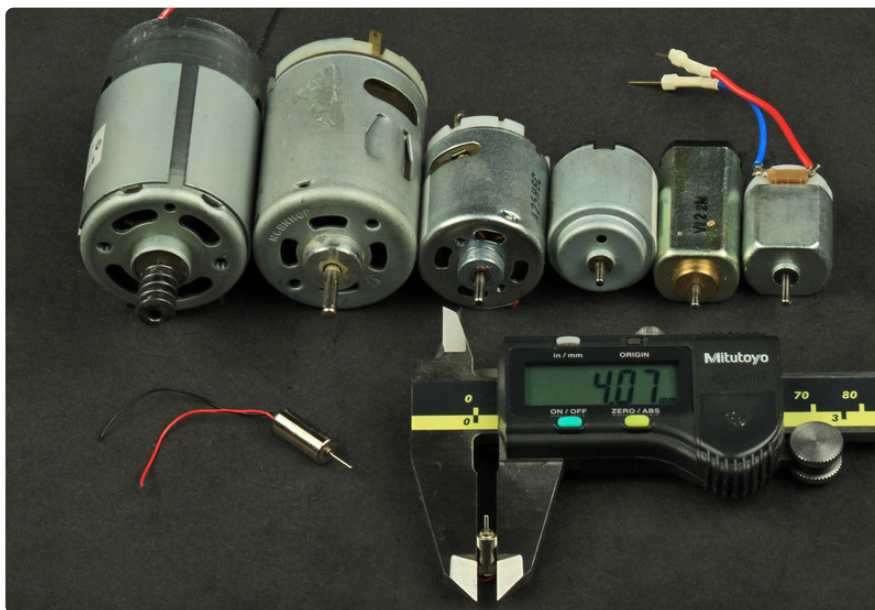
Motors

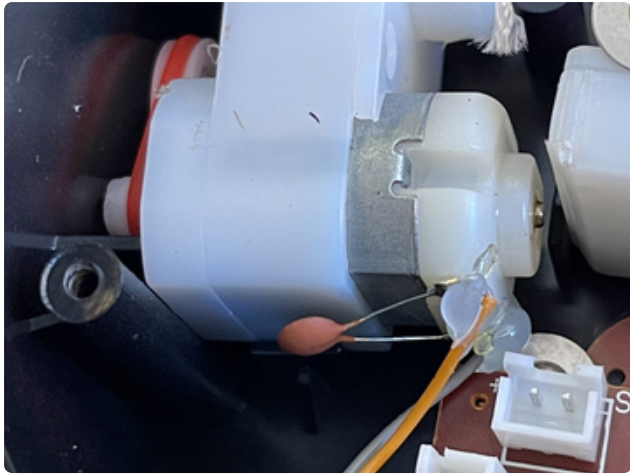
The key things to determine about motors are: motor type, voltage requirements, and current draw.

You can read more about different common motor types [here](https://adafru.it/scm) (<https://adafru.it/scm>).

The vast majority of animatronics from the store will contain brushed DC motors. These are perhaps the "dumbest" of the motor types you'll encounter, as they cannot be told to go to a particular rotational setting the way hobby servos and stepper motors can.

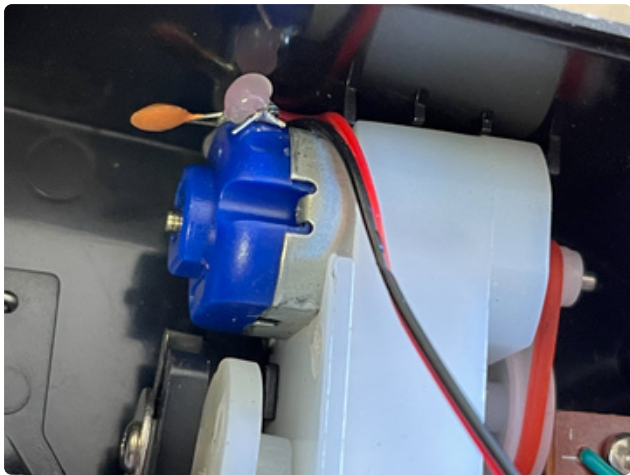
You can typically identify them by their shape -- a metal cylinder or flattened cylinder, with a plastic end cap.



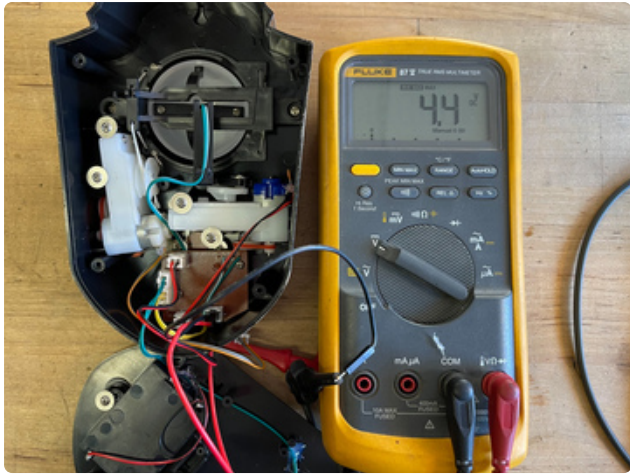


In the Eyeball Doorbell example, there are two brushed DC motors, one to open and close the eyelid and another to rotate the eyeball from side to side.

They are geared down with a pulley and belt system.



Another tell-tale sign of a brushed DC motor is the capacitor soldered across its leads to help reduce electrical noise in the overall circuit.



Voltage

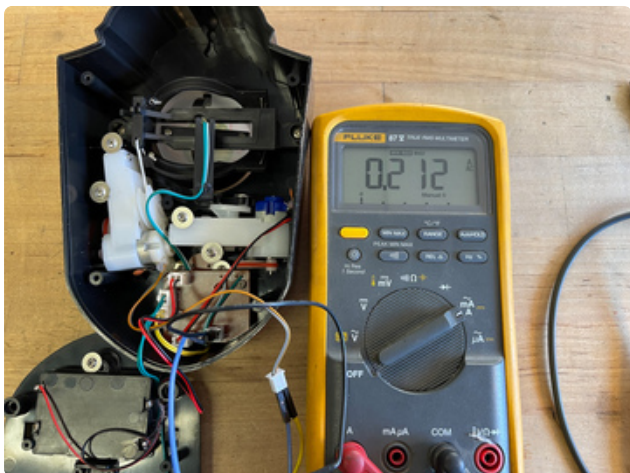
Using a multimeter in DC voltage measuring mode, we can see that the Doorbell sends approximately 4.5V maximum and approx. -4.5V minimum voltage to the motor during operation.



The three AAA batteries supply 4.5V, (1.5V x 3) so this makes sense, and is often another clue as to power requirements, although this can be misleading if the original circuit employs a buck or boost converter.

Current Draw

You also want to determine the maximum current draw of each motor so you can use a power supply or battery pack that provides enough current.

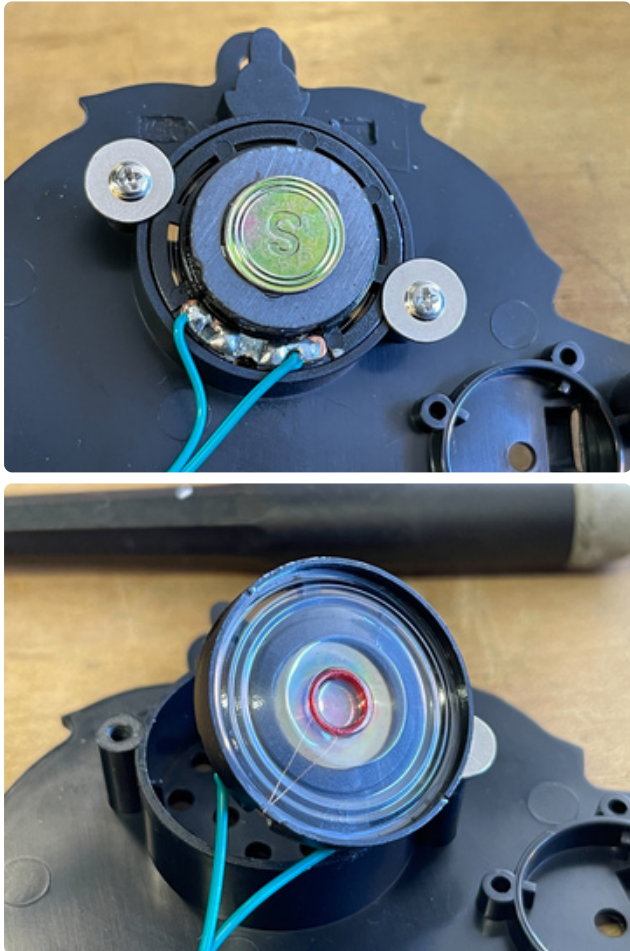


Use your multimeter in current measuring mode (this may require switching one of the test leads on some meters) connect one lead of the motor to the original controller board and the other from the board, in the multimeter's positive lead, then from the multimeter's negative lead to the motor.

Run the animatronic cycle and read the maximum current draw. In this case it is about 212 milliamps.

Measure the greatest draw a motor will pull -- the eyelid motor in the doorbell spins, then stays held with the lid open, drawing peak current.

Both motors combined draw a total maximum of about 250mA, so we'll keep this in mind when picking a motor driver. (Hint: The Crickit can handle this with no problems!)



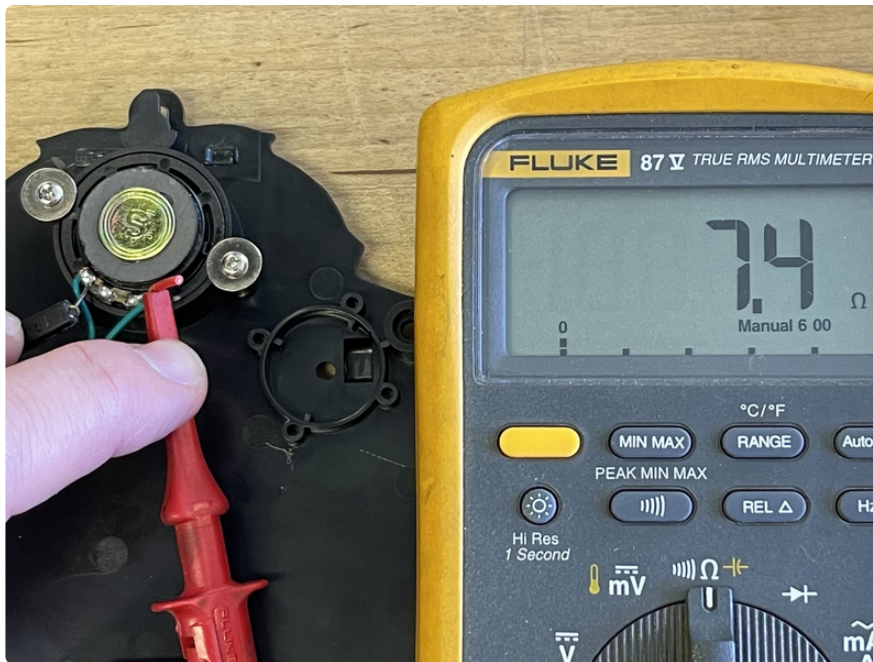
Speaker Impedance

Typical toy/animatronic speakers are either 4Ω or 8Ω speakers, and this is often marked on the back of the speaker.

If not, you can usually measure the DC resistance of the speaker to get a ballpark idea of its impedance.

Unplug the speaker from the rest of the circuit first, then use your multimeter in resistance measuring mode to determine the impedance. In this case, a measurement of 7.4Ω is close enough to say it is an 8Ω speaker.

This means you'll need to use an amplifier that can drive an 8Ω speaker, which happens to be the case with most of the Adafruit amps! (They tend to drive anywhere from 4Ω - 8Ω speakers.)



Input

The trigger for the Doorbell is a simple momentary pushbutton switch. That's about as easy as it gets to re-use! You'll simply use the switch to control a digital input, by connecting the switch to a GPIO pin on the microcontroller (or Crickit) and ground.

If you're faced with a more elaborate input, such as a PIR sensor for motion detection, you may need to test the signals with a multimeter or scope to determine what signal is sent when the trigger is tripped. Often this will still be a simple digital high/low signal, which is easy to integrate, pretty much the same as the momentary switch.



LED Light

If your animatronic contains simple LED lights, you can drive them from a digital output or PWM pin to blink or even fade them. These will look like the typical two-legged LED (although they are sometimes a bit hidden as in the doorbell eye example here).

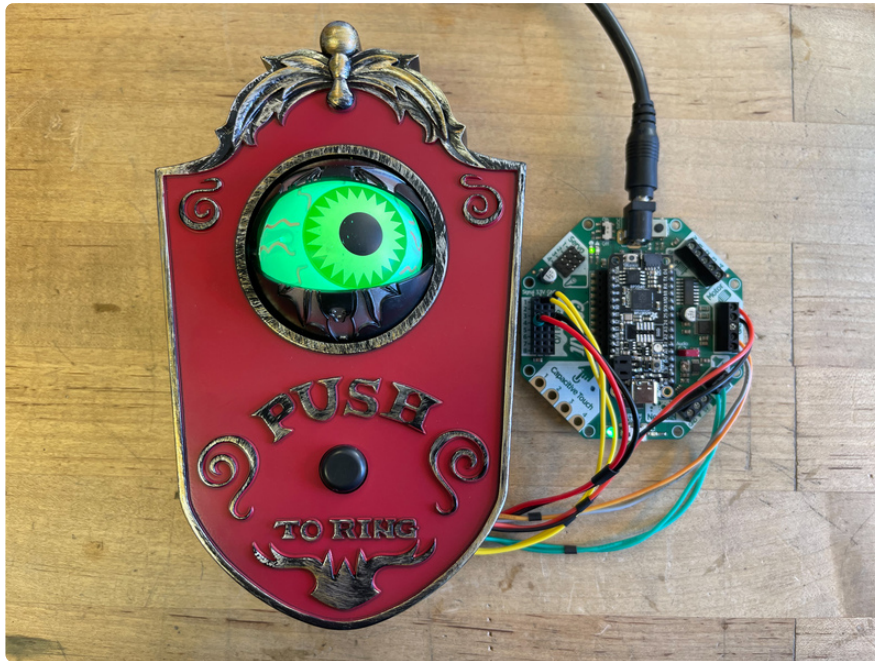
If the animatronic does multicolored lighting with RGB LEDs they will be either analog or digital (NeoPixel-style) packages.

You can determine if a four-legged LED is analog or digital by measuring the voltage between the power leg and each of the three other legs. If they are each operating at roughly 1V-3V, you've probably got an analog LED on your hands, and can control them in the manner shown in [this guide \(https://adafru.it/lCy\)](https://adafru.it/lCy).

If not, you may be dealing with a NeoPixel or similar LED.

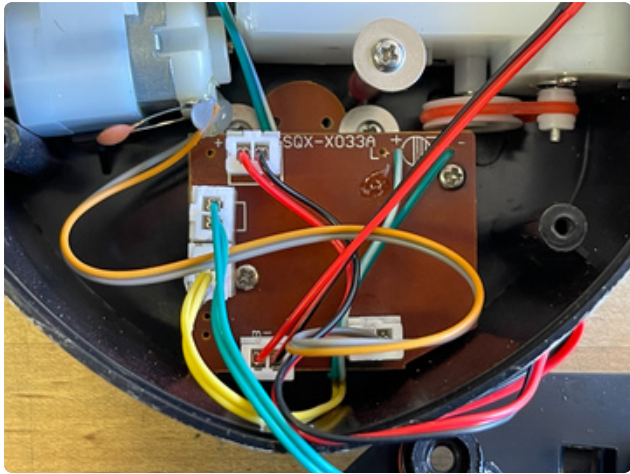
If the existing LEDs seem daunting to control, you may consider swapping them out for your own NeoPixels.

Controlling Animatronics



The Crickit FeatherWing is an excellent choice for controlling the motors, sounds, LEDs, and input necessary for many animatronics.

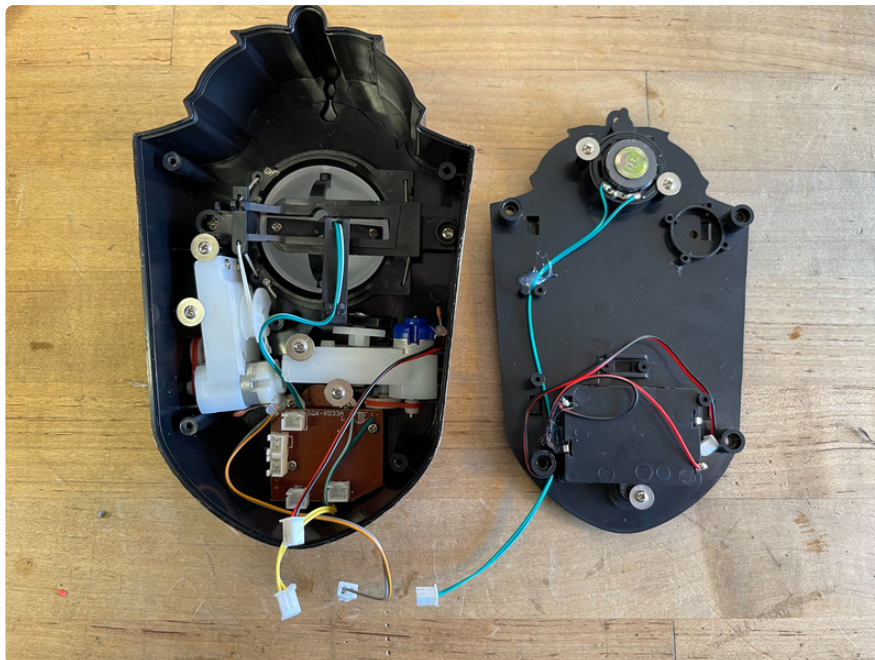
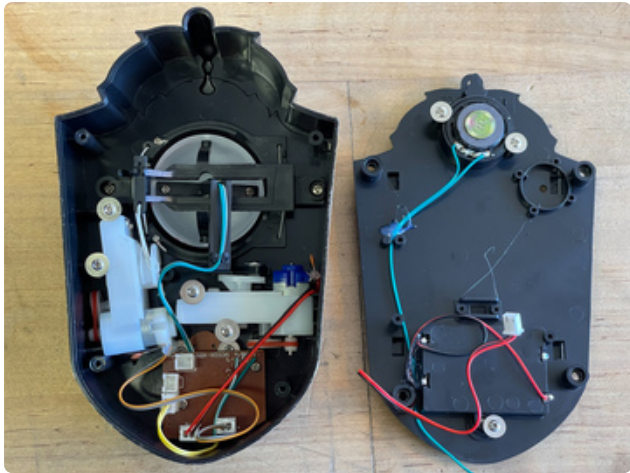
I'll use the Animated Eyeball Doorbell for specific instructions here, but you should be able to apply them to many different cases. Watch out Santa!



Teardown Documentation

It's a good idea to take detailed photos as you teardown the device. The Doorbell wiring is done with JST XH connectors, and has colored wiring, but no indication on the original mainboard as to which plug does what.

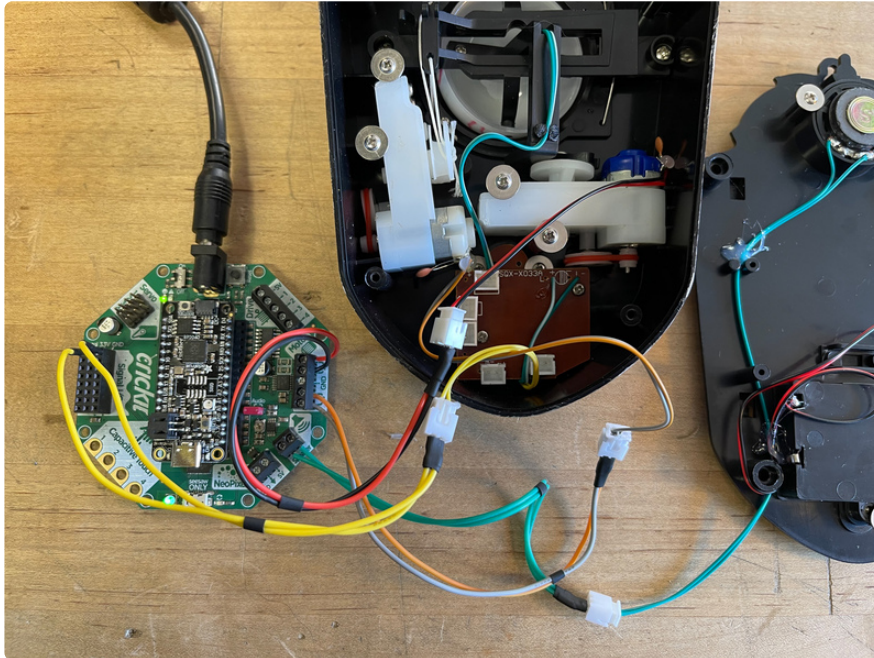
If you need to test things or return the device to stock state, it's incredibly helpful to have "before" photos as guidance!



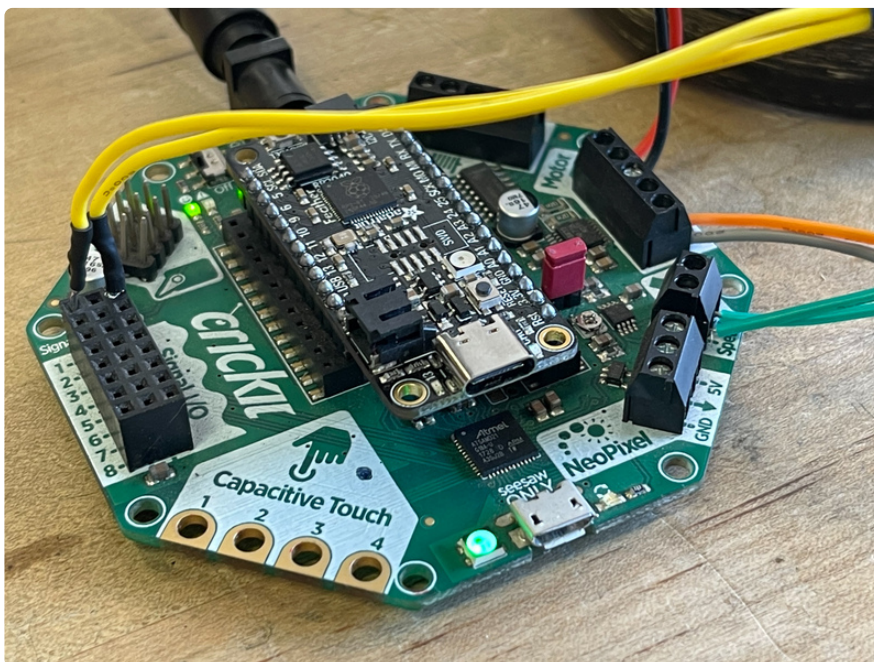
With the peripherals disconnected, it's time to test driving each part from a Crickit and Feather combo.

Use the JST XH pigtails to connect:

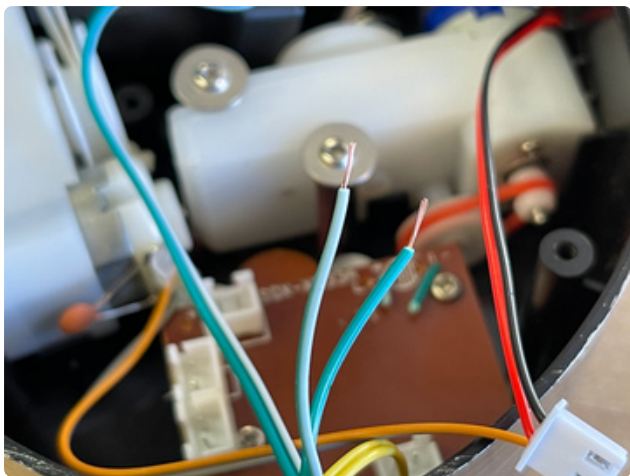
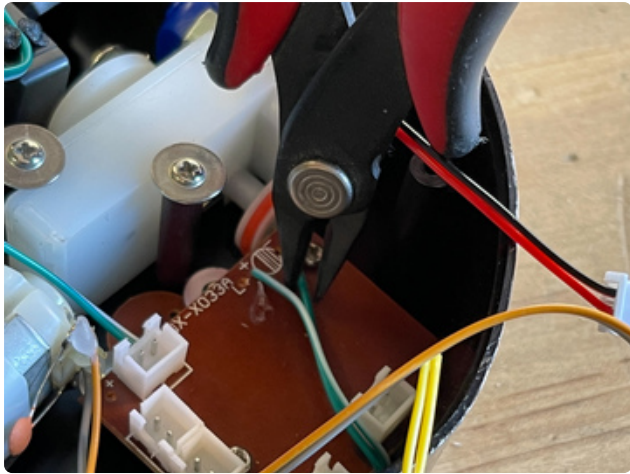
- two motors to the motor driver terminals
- speaker to audio out
- pushbutton to signal 1/GND



Be sure to add a jumper across the audio output header pins on the Crickit in order to enable the amplifier output. (Red jumper shown here.)



Since the LED is wired directly to the board you'll need to cut those wires to insert them into **signal 4/gnd**.

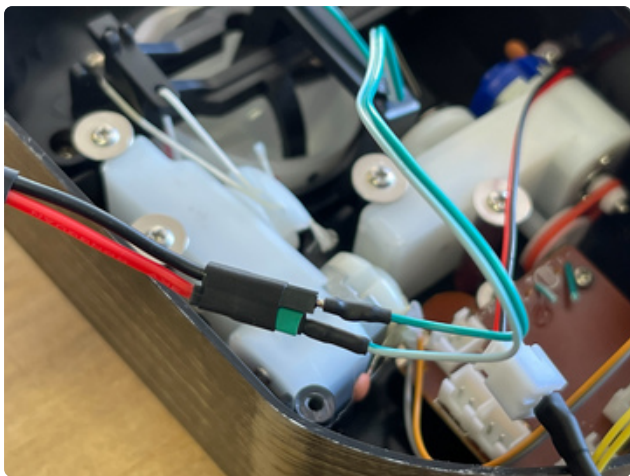
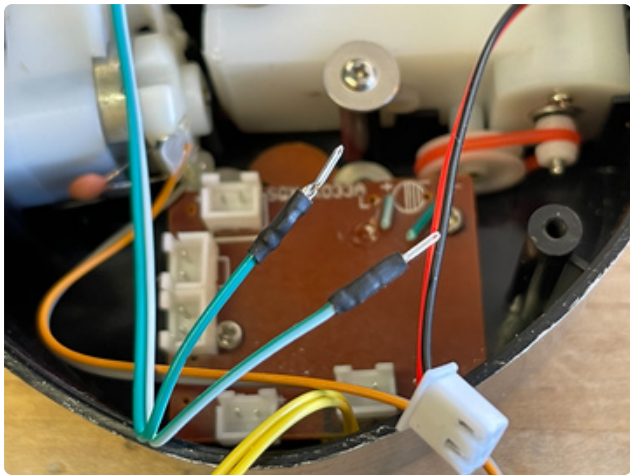
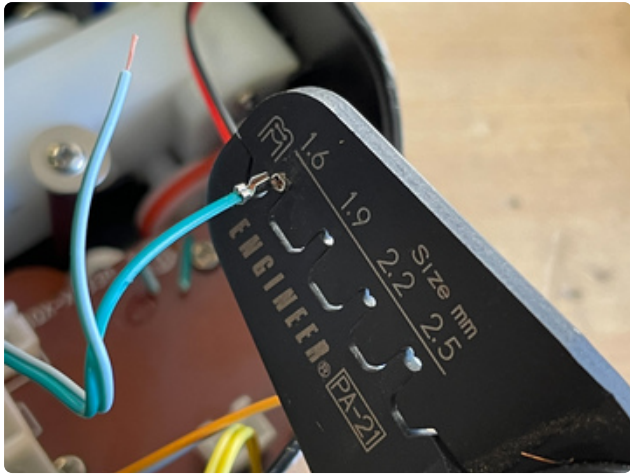


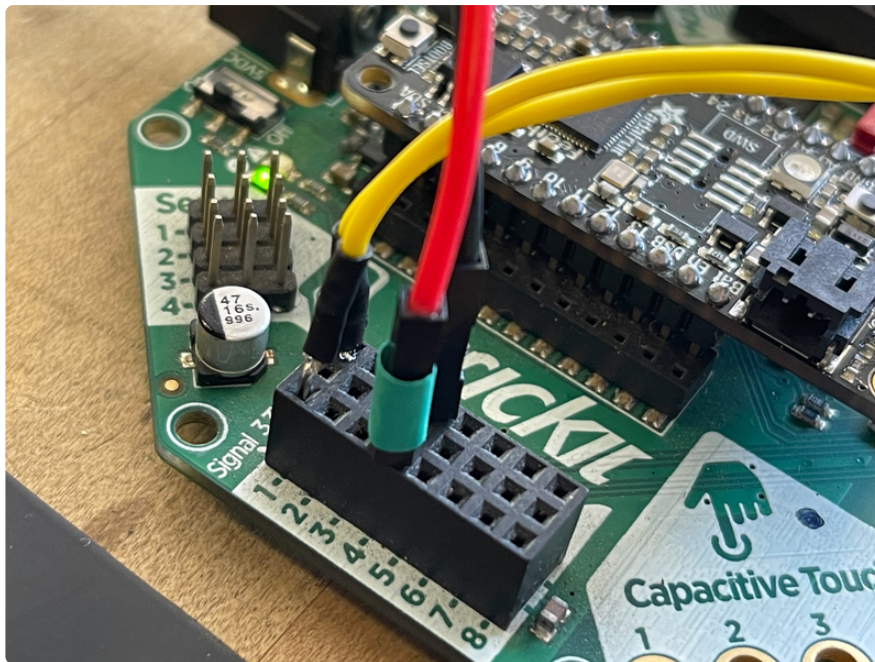
LED Wiring

Note the polarity of the wiring -- **gray** is + and **green** is - here.

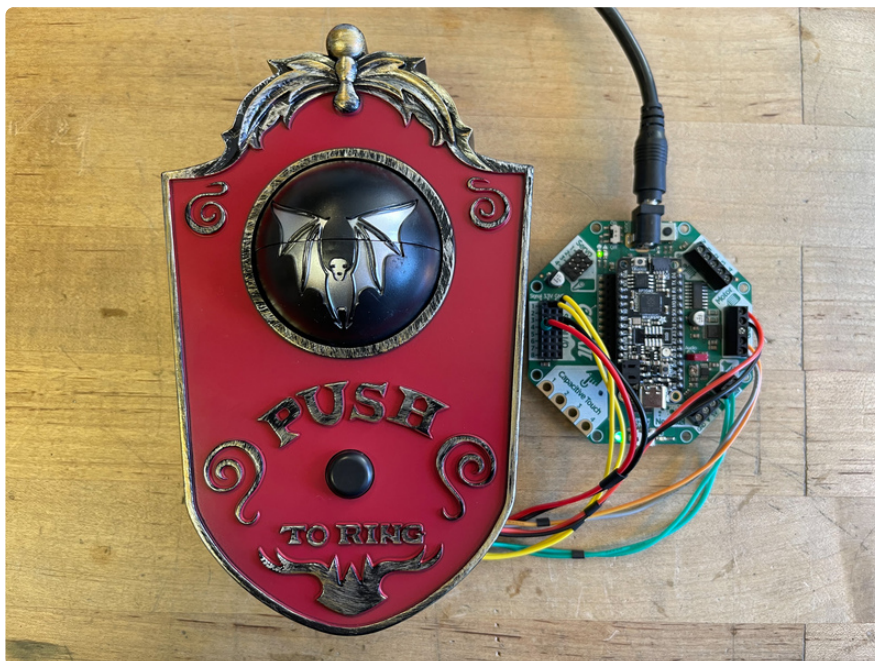
Clip the wires, then strip some insulation.

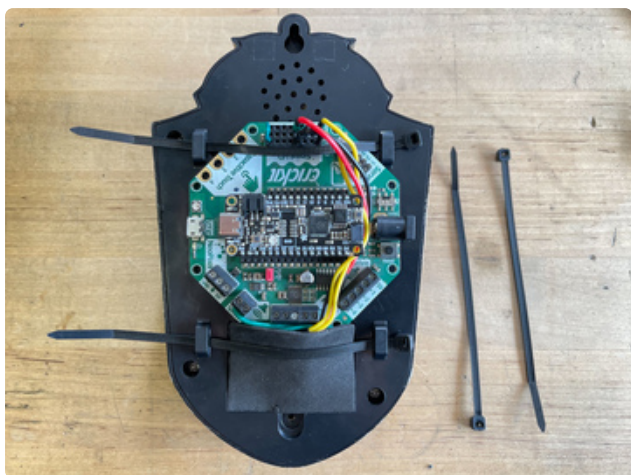
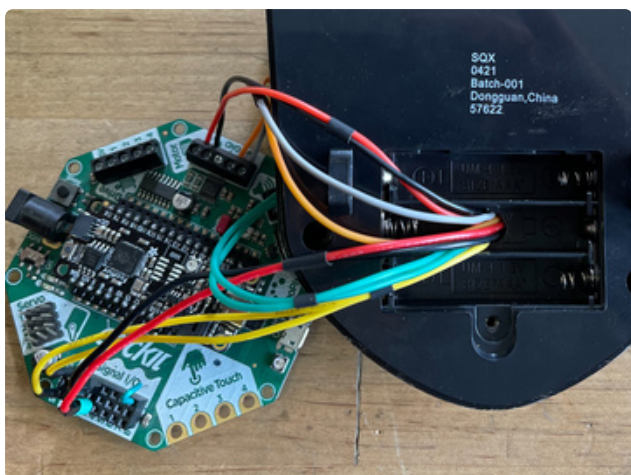
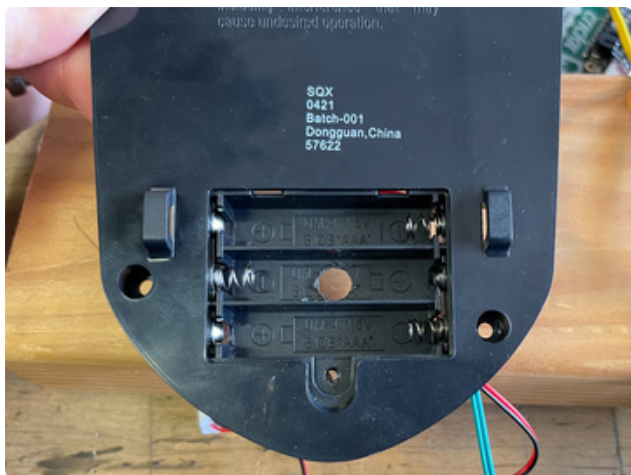
If you want to go the extra mile, you can crimp pins and add header shrouding or heat shrink tubing as shown here.





The transplant is complete! Now, the Feather and Crickit can be coded in CircuitPython or Arduino to receive input from the doorbell pushbutton, and then control both motors, play unique sound files, and blink the LED.





Final Touches

Depending on the space inside your animatronic device, you may be able to fit the Crickit inside. If not, you may want to build a custom back to add space, or run the wiring into a separate enclosure.

Here's an example of drilling out a hole in the battery compartment to run the Crickit externally, using zip ties to hold the Crickit in place.

The also allows access to the power, on/off switch, and USB connector for any re-programming.



Install CircuitPython

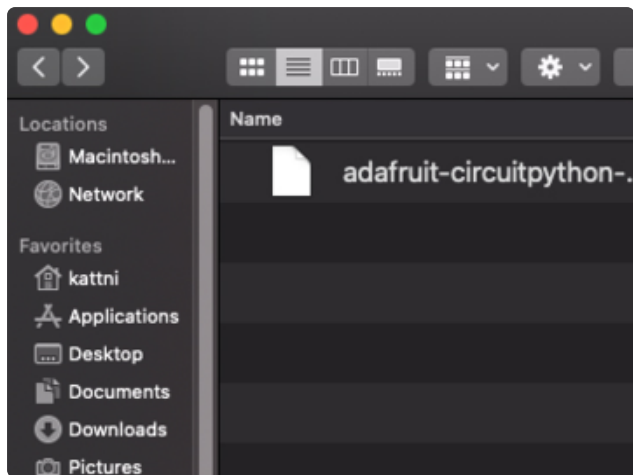
[CircuitPython](https://adafru.it/tB7) (<https://adafru.it/tB7>) is a derivative of [MicroPython](https://adafru.it/BeZ) (<https://adafru.it/BeZ>) designed to simplify experimentation and education on low-cost microcontrollers. It makes it easier than ever to get prototyping by requiring no upfront desktop 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.

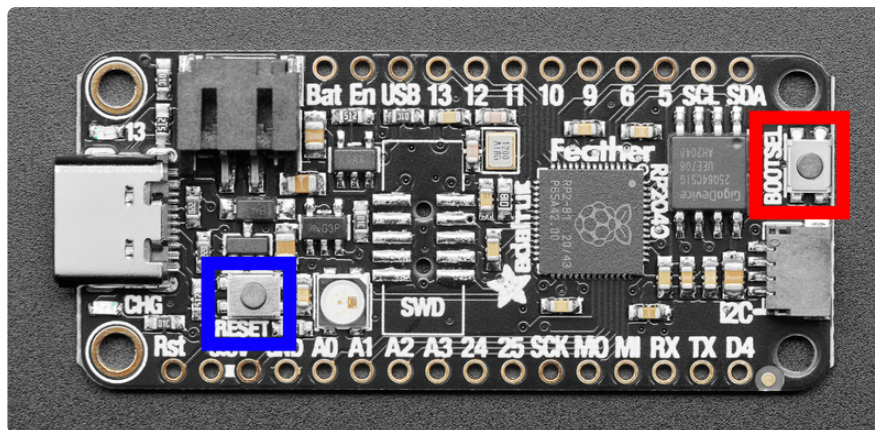
Download the latest version of
CircuitPython for this board via
[circuitpython.org](https://adafru.it/R1D)

<https://adafru.it/R1D>



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

Save it wherever is convenient for you.

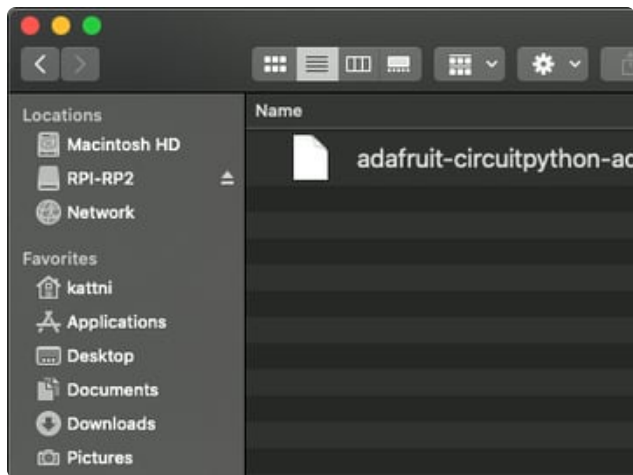


To enter the bootloader, hold down the **BOOT/BOOTSEL** button (highlighted in red above), and while continuing to hold it (don't let go!), press and release the **reset** button (highlighted in blue above). **Continue to hold the BOOT/BOOTSEL button until the RPI-RP2 drive appears!**

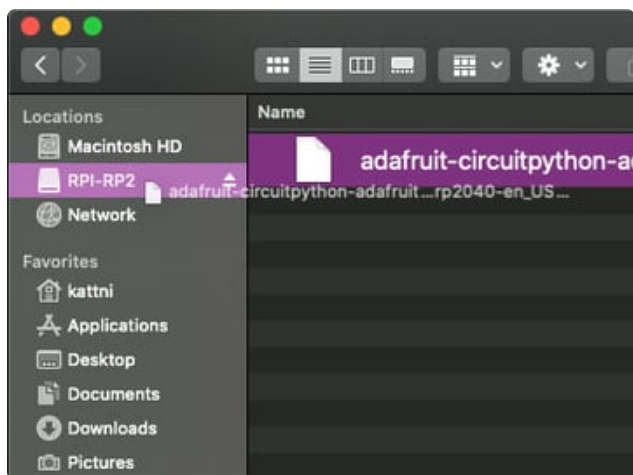
If the drive does not appear, release all the buttons, and then repeat the process above.

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.

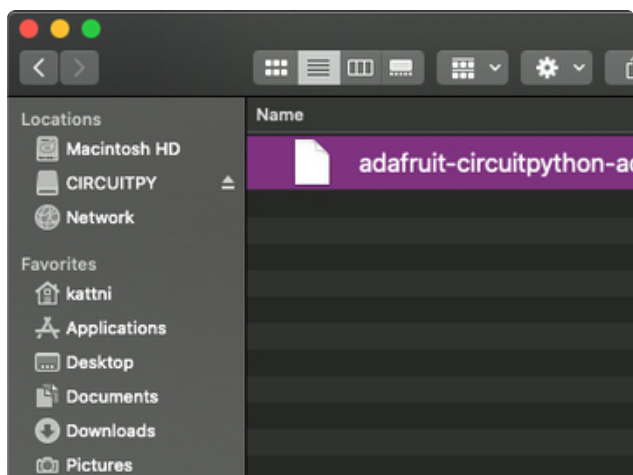
A lot of people end up using charge-only USB cables and it is very frustrating! **Make sure you have a USB cable you know is good for data sync.**



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 CIRCUITPY doesn't show up as a disk drive after installing CircuitPython, try loading this 'nuke' UF2 to RPI-RP2. 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

<https://adafru.it/RLE>

Code the Animatronic Eyeball Doorbell

Text Editor

Adafruit recommends using the **Mu** editor for editing your CircuitPython code. You can get more info in [this guide \(https://adafru.it/ANO\)](https://adafru.it/ANO).

Alternatively, you can use any text editor that saves simple text files.

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file, along with a folder full of key configuration files. To get everything you need, click on the **Download Project Bundle** link below, and uncompress the .zip file.

Drag the contents of the uncompressed bundle directory onto your Feather board's **CIRCUITPY** drive, replacing any existing files or directories with the same names, and adding any new ones that are necessary.

```
# SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
# SPDX-License-Identifier: MIT
import time
import random
import board
import audiomp3
import audiopwmio
from adafruit_crickit import crickit

ss = crickit.seesaw # Crickit seesaw setup

button = crickit.SIGNAL1 # momentary switch to trigger animation
ss.pin_mode(button, ss.INPUT_PULLUP)

LED = crickit.SIGNAL4 # standard LED for eyeball lighting
ss.pin_mode(LED, ss.OUTPUT)

attract_switch = crickit.SIGNAL8 # attract mode switch or jumper
ss.pin_mode(attract_switch, ss.INPUT_PULLUP)

audio = audiopwmio.PWMAudioOut(board.A0) # Feather outputs this pin to Crickit
amplifier
audio_files = [ # use your own mono .mp3 files
```

```

        "phrase_01.mp3",
        "phrase_02.mp3",
        "phrase_03.mp3"
    ]
    current_audio_file = 0

    # two motors
    motor_eye = crickit.dc_motor_1
    motor_lid = crickit.dc_motor_2

    def open_lid():
        motor_lid.throttle = 1 # full speed open
        time.sleep(0.25)
        motor_lid.throttle = 0 # hold

    def close_lid():
        motor_lid.throttle = -1 # full speed closed
        time.sleep(0.25)
        motor_lid.throttle = 0

    def blink(times):
        for _ in range(times):
            ss.digital_write(LED, True)
            time.sleep(0.1)
            ss.digital_write(LED, False)
            time.sleep(0.1)

    def eye_look():
        motor_eye.throttle = random.uniform(0.6, 1.0)
        time.sleep(random.random()) # 0 to 1.0 seconds
        motor_eye.throttle = 0
        time.sleep(random.random())
        motor_eye.throttle = random.uniform(-1.0, -0.6)
        time.sleep(random.random())
        motor_eye.throttle = 0
        time.sleep(random.random())

    while True:
        if ss.digital_read(attract_switch): # regular mode, attract switch not closed/
            shorted
            if not ss.digital_read(button): # button has been pressed
                decoder = audiomp3.MP3Decoder(open("ring.mp3", "rb"))
                audio.play(decoder)
                while audio.playing:
                    pass
                open_lid()
                blink(3)
                ss.digital_write(LED, True) # light the eye
                decoder = audiomp3.MP3Decoder(open(audio_files[current_audio_file],
"rb"))
                audio.play(decoder)
                while audio.playing:
                    eye_look()
                motor_eye.throttle = 0 # audio is finished, pause the eye
                blink(5)
                close_lid()
                current_audio_file = ((current_audio_file + 1) % (len(audio_files))) #
go to next file

            else: # attract mode
                open_lid()
                blink(3)
                ss.digital_write(LED, True)
                for _ in range(4):
                    eye_look()
                time.sleep(1)
                blink(5)

```

```
close_lid()  
time.sleep(random.randint(2, 8))
```

Use The Hacked Animatronic

You can now run your customized animatronic using the Crickit board! Press the original pushbutton to get it started. It will then play the **ring.mp3** audio file, open the eyelid, blink the LED, and start playing the first phrase file.

While the audio file plays, it'll run the eyeball motor back and forth, with some randomized speeds and timing. When the audio file is finished playing, the current loop of eye animation will complete and then the LED will turn off and the lid will close, ready for the next press of the button.

The next time the button is pressed the next audio file in the list will play, and so on until it loops back around to the first one.

Attract Mode: you can use a jumper across the Crickit SIGNAL8 and GND pins to put the device into "attract mode". It will run through eye animation but without playing any sounds.

Create Your Own Audio Files

You can make and load any .mp3 file you like onto your Feather RP2040's **CIRCUITPY** drive. Here's a guide on how to create them.

Make Your Own

To make your own .mp3, use audio software such as [Audacity](https://adafru.it/Lee) (<https://adafru.it/Lee>) to save them with these settings:

- bit rate: anywhere from 16 to 320 kb/s (lower will be smaller, higher is better quality)
- sample rate: 22050Hz or 44100Hz are recommended, although 16kHz, 24kHz, and 48kHz should also work
- channels: mono or stereo
- must be DRM free

Then, simply change the name in the audio file list in code and you can enjoy your own spooky/funny/weird/sassy/you-name-it phrases being uttered by a disembodied eyeball doorbell!