Neo Trinkey Zoom Shortcuts

Created by Liz Clark

https://learn.adafruit.com/neo-trinkey-zoom-shortcuts

Last updated on 2022-12-01 04:02:51 PM EST

Table of Contents

Overview 3

- Parts

CircuitPython 4

- CircuitPython Quickstart

Coding the Project 6

- Installing Project Code

CircuitPython Code Walkthrough 9

- Import the Libraries
- Setup the NeoPixels
- Setup the Capacitive Touch Inputs
- Setup for HID
- NeoPixel Animation and Colors
- State Machines
- The Loop and Touch Input Debouncing
- Setting the Mute, Unmute and Exit Colors
- What Happens When You Press the Top Pad?
- The Great Escape

Usage 14

Overview

Code up a simple, HID shortcut device to use during your Zoom calls.

The Neo Trinkey sticks out from your computer's USB port and gives you access to two capacitive touch pads. When you touch the top pad, you can mute or unmute your webcam and when you touch the bottom pad you can mute or unmute your microphone.

The four NeoPixels are a visual cue to let you know the state of your devices. They'll be red if an input is muted and green when they're unmuted.

For certain Zoom calls, the best part is leaving the call. You can leave quickly by pressing and holding down both touch pads for a little over a second to make a smooth exit. Additionally, the NeoPixels display the classic rainbow cycle animation to celebrate.

Parts

Adafruit Neo Trinkey - SAMD21 USB Key with 4 NeoPixels
It's half USB Key, half Adafruit Trinket...it's Neo Trinkey, the circuit board with a Trinket M0 heart and four RGB NeoPixels for customizable...
https://www.adafruit.com/product/4870

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.

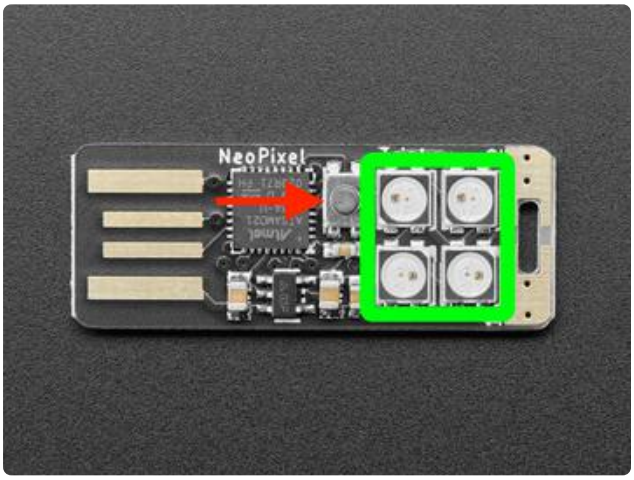
CircuitPython Quickstart

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

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

Click the link above and download the latest UF2 file.

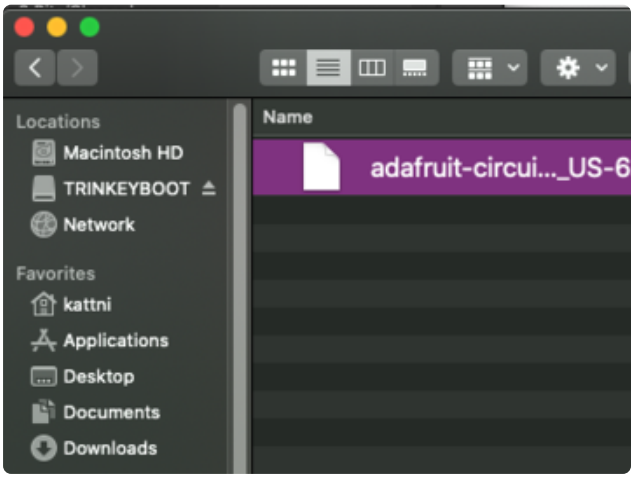
Download and save it to your desktop (or wherever is handy).



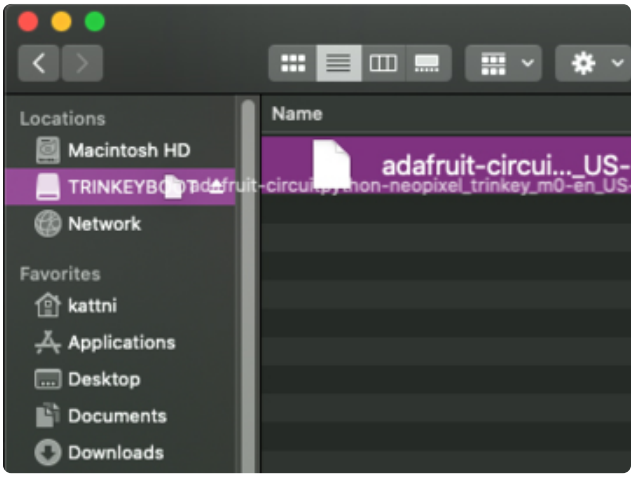
Plug your Neo Trinkey directly into your computer's USB port, or via an adapter if needed.

Double-click the small reset button (indicated by the red arrow), and you will see the NeoPixel RGB LEDs turn green (indicated by the green box in the image). If they turn red, try another port, or if you're using an adapter or hub, try another adapter or hub.

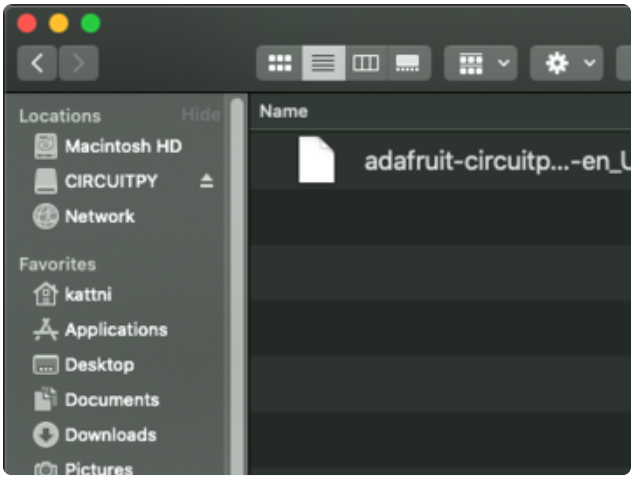
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!



You will see a new disk drive appear called TRINKEYBOOT.



Drag the adafruit_circuitpython_etc.uf2 file to TRINKEYBOOT.



The LEDs will flash red. Then, the TRINKEYBOOT drive will disappear and a new disk drive called CIRCUITPY will appear.

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

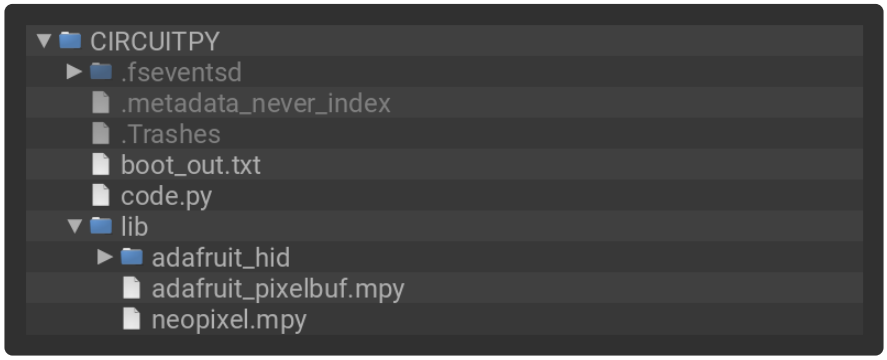
Coding the Project

Installing Project Code

To use with CircuitPython, you need to first install a few libraries, into the lib folder on your CIRCUITPY drive. Then you need to update code.py with the example script.

Thankfully, we can do this in one go. In the example below, click the Download Project Bundle button below to download the necessary libraries and the code.py file in a zip file. Extract the contents of the zip file, open the directory NeoTrinkey_Zoom_Shortcuts/ and then click on the directory that matches the version of CircuitPython you're using and copy the contents of that directory to your CIRCUITPY drive.

Your CIRCUITPY drive should now look similar to the following image:




```
# SPDX-FileCopyrightText: 2021 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
import board
import neopixel
import touchio
import usb_hid
```

```
from rainbowio import colorwheel
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.keycode import Keycode

#  setup for onboard neopixels
pixel_pin = board.NEOPIXEL
num_pixels = 4

pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.05, auto_write=False)

#  setup for cap touch pads
top_touch = touchio.TouchIn(board.TOUCH1)
bot_touch = touchio.TouchIn(board.TOUCH2)

#  HID keyboard input setup
keyboard = Keyboard(usb_hid.devices)
keyboard_layout = KeyboardLayoutUS(keyboard)
#  variable for the ALT key
alt_key = Keycode.ALT


def rainbow_cycle(wait):
    for j in range(255):
        for i in range(num_pixels):
            rc_index = (i * 256 // num_pixels) + j
            pixels[i] = colorwheel(rc_index & 255)
        pixels.show()
        time.sleep(wait)

#  variables for colors
RED = (255, 0, 0)
GREEN = (0, 255, 0)

#  state machines
#  cap touch debouncing
bot_pressed = False
top_pressed = False
#  default mute states
mic_mute = True
vid_mute = True
#  time.monotonic() tracker
clock = time.monotonic()
#  tracking for initiating an exit from the meeting
escape = False
escape_1 = False
escape_2 = False

while True:
    #  cap touch debouncing
    if not top_touch.value and top_pressed:
        top_pressed = False
    if not bot_touch.value and bot_pressed:
        bot_pressed = False
    #  if your mic is muted...
    if mic_mute:
        #  neopixels are red
        pixels[0] = RED
        pixels[1] = RED
        pixels.show()
    #  if your camera is muted...
    if vid_mute:
        #  neopixels are red
        pixels[2] = RED
        pixels[3] = RED
        pixels.show()
    #  if your mic is NOT muted...
    if not mic_mute:
        #  neopixels are green
```

```
        pixels[0] = GREEN
        pixels[1] = GREEN
        pixels.show()
    #  if your camera is NOT muted...
    if not vid_mute:
        #  neopixels are green
        pixels[2] = GREEN
        pixels[3] = GREEN
        pixels.show()
    #  if you are leaving the meeting...
    if escape:
        #  neopixels are rainbow
        rainbow_cycle(0)
        #  resets exit states
        escape = False
        escape_1 = False
        escape_2 = False
        mic_mute = True
        vid_mute = True
    #  if you press the top touch cap touch pad...
    if (top_touch.value and not top_pressed):
        top_pressed = True
        #  start time count for exit
        clock = time.monotonic()
        #  slight delay so that you don't automatically mute/unmute
        #  if your intent is to exit
        time.sleep(0.12)
        #  if after the delay you're still pressing the cap touch pad...
        if top_touch.value and top_pressed:
            print("escape top")
            #  initial escape state is set to true
            escape_1 = True
        #  if you aren't still pressing the cap touch pad...
        else:
            #  if your camera was muted...
            if vid_mute:
                print("top")
                #  your camera is NOT muted
                vid_mute = False
                #  resets escape state just in case
                escape_1 = False
            #  if your camera was NOT muted...
            elif not vid_mute:
                print("top")
                #  your camera is muted
                vid_mute = True
                #  resets escape state just in case
                escape_1 = False
            #  sends camera mute/unmute shortcut
            keyboard.send(alt_key, Keycode.V)
    #  if you press the top touch cap touch pad...
    if (bot_touch.value and not bot_pressed):
        bot_pressed = True
        #  start time count for exit
        clock = time.monotonic()
        #  slight delay so that you don't automatically mute/unmute
        #  if your intent is to exit
        time.sleep(0.12)
        #  if after the delay you're still pressing the cap touch pad...
        if bot_touch.value and bot_pressed:
            print("escape bot")
            #  initial escape state is set to true
            escape_2 = True
        #  if you aren't still pressing the cap touch pad...
        else:
            #  if your mic was muted...
            if mic_mute:
                print("bot")
                #  your mic is NOT muted
```

```
                mic_mute = False
                #  resets escape state just in case
                escape_2 = False
            #  if your mic was NOT muted...
            elif not mic_mute:
                print("bot")
                #  your mic is muted
                mic_mute = True
                #  resets escape state just in case
                escape_2 = False
            #  sends mic mute/unmute shortcut
            keyboard.send(alt_key, Keycode.A)
    #  if you held down both cap touch pads and 2 seconds has passed...
    if ((clock + 2) < time.monotonic()) and (escape_1 and escape_2):
        print("escape")
        #  full escape state is set
        escape = True
        #  sends exit meeting shortcut
        keyboard.send(alt_key, Keycode.Q)
        #  brief delay for confirmation window to open
        time.sleep(0.1)
        #  sends enter to confirm meeting exit
        keyboard.send(Keycode.ENTER)
```

CircuitPython Code Walkthrough

Import the Libraries

First, the libraries are imported.

```
import time
import board
import neopixel
import touchio
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
from adafruit_hid.keycode import Keycode
```

Setup the NeoPixels

Then, the NeoPixels are setup. The four onboard NeoPixels can be accessed with `board.NEOPIXEL`.

```
#  setup for onboard neopixels
pixel_pin = board.NEOPIXEL
num_pixels = 4

pixels = neopixel.NeoPixel(pixel_pin, num_pixels, brightness=0.05, auto_write=False)
```

Setup the Capacitive Touch Inputs

The two capacitive touch pads are accessed with `board.TOUCH1` and `board.TOUCH2`. They're setup as touch inputs with the `touchio` library.

```
#  setup for cap touch pads
top_touch = touchio.TouchIn(board.TOUCH1)
bot_touch = touchio.TouchIn(board.TOUCH2)
```

Setup for HID

`keyboard` is setup as the USB HID keyboard object. Zoom uses the ALT key for a lot of its built-in shortcuts, so `alt_key` is setup as a variable to hold `Keycode.ALT`.

```
#  HID keyboard input setup
keyboard = Keyboard(usb_hid.devices)
keyboard_layout = KeyboardLayoutUS(keyboard)
#  variable for the ALT key
alt_key = Keycode.ALT
```

NeoPixel Animation and Colors

The classic `rainbow_cycle` NeoPixel animation is included, followed by `RED` and `GREEN` RGB color values that will be used to show if your inputs are muted or unmuted.

```
#  rainbow cycle animation
def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if pos &lt; 0 or pos &gt; 255:
        return (0, 0, 0)
    if pos &lt; 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos &lt; 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)


def rainbow_cycle(wait):
    for j in range(255):
        for i in range(num_pixels):
            rc_index = (i * 256 // num_pixels) + j
            pixels[i] = wheel(rc_index &amp;amp; 255)
        pixels.show()
        time.sleep(wait)

#  variables for colors
```

```
RED = (255, 0, 0)
GREEN = (0, 255, 0)
```

State Machines

A few state machines are used in the loop and their functions are commented in the code.

```
#  state machines
#  cap touch debouncing
bot_pressed = False
top_pressed = False
#  default mute states
mic_mute = True
vid_mute = True
#  time.monotonic() tracker
clock = time.monotonic()
#  tracking for initiating an exit from the meeting
escape = False
escape_1 = False
escape_2 = False
```

The Loop and Touch Input Debouncing

The loop begins with some debouncing for the cap touch inputs.

```
while True:
    #  cap touch debouncing
    if not top_touch.value and top_pressed:
        top_pressed = False
    if not bot_touch.value and bot_pressed:
        bot_pressed = False
```

Setting the Mute, Unmute and Exit Colors

The `mic_mute` and `vid_mute` states affect whether the two corresponding NeoPixels are green or red.

```
#  if your mic is muted...
    if mic_mute:
        #  neopixels are red
        pixels[0] = RED
        pixels[1] = RED
        pixels.show()
    #  if your camera is muted...
    if vid_mute:
        #  neopixels are red
        pixels[2] = RED
        pixels[3] = RED
        pixels.show()
    #  if your mic is NOT muted...
    if not mic_mute:
```

```
        #  neopixels are green
        pixels[0] = GREEN
        pixels[1] = GREEN
        pixels.show()
    #  if your camera is NOT muted...
    if not vid_mute:
        #  neopixels are green
        pixels[2] = GREEN
        pixels[3] = GREEN
        pixels.show()
```

The `escape` state cues the start of the `rainbow_cycle` animation when you exit a meeting. It also resets the `escape_1`, `escape_2`, `mic_mute` and `vid_mute` states to their defaults.

```
#  if you are leaving the meeting...
    if escape:
        #  neopixels are rainbow
        rainbow_cycle(0)
        #  resets exit states
        escape = False
        escape_1 = False
        escape_2 = False
        mic_mute = True
        vid_mute = True
```

What Happens When You Press the Top Pad?

When you press the top touch pad, `time.monotonic()` is called to begin a time count. This is followed by a delay of `0.12` seconds. This tracks if you are holding the touch input for an exit. Otherwise, the code would go right to muting or unmuting your webcam.

```
#  if you press the top touch cap touch pad...
    if (top_touch.value and not top_pressed):
        top_pressed = True
        #  start time count for exit
        clock = time.monotonic()
        #  slight delay so that you don't automatically mute/unmute
        #  if your intent is to exit
        time.sleep(0.12)
```

If you are inputting a long press to trigger an exit from the meeting, the `escape_1` state is set to `True`, which will begin initiating the exit. This also keeps your webcam's mute status unchanged so that you don't suddenly mute or unmute while you're leaving the meeting.

```
#  if after the delay you're still pressing the cap touch pad...
        if top_touch.value and top_pressed:
            print("escape top")
            #  initial escape state is set to true
            escape_1 = True
```

If you aren't inputting a long press, then the `vid_mute` status will change depending on the previous state and the shortcut for muting/unmuting the video (ALT+V) is sent.

```
#  if you aren't still pressing the cap touch pad...
        else:
            #  if your camera was muted...
            if vid_mute:
                print("top")
                #  your camera is NOT muted
                vid_mute = False
                #  resets escape state just in case
                escape_1 = False
            #  if your camera was NOT muted...
            elif not vid_mute:
                print("top")
                #  your camera is muted
                vid_mute = True
                #  resets escape state just in case
                escape_1 = False
            #  sends camera mute/unmute shortcut
            keyboard.send(alt_key, Keycode.V)
```

The same logic is in place for the bottom capacitive touch input, which controls the microphone.

```
#  if you press the top touch cap touch pad...
    if (bot_touch.value and not bot_pressed):
        bot_pressed = True
        #  start time count for exit
        clock = time.monotonic()
        #  slight delay so that you don't automatically mute/unmute
        #  if your intent is to exit
        time.sleep(0.12)
        #  if after the delay you're still pressing the cap touch pad...
        if bot_touch.value and bot_pressed:
            print("escape bot")
            #  initial escape state is set to true
            escape_2 = True
        #  if you aren't still pressing the cap touch pad...
        else:
            #  if your mic was muted...
            if mic_mute:
                print("bot")
                #  your mic is NOT muted
                mic_mute = False
                #  resets escape state just in case
                escape_2 = False
            #  if your mic was NOT muted...
            elif not mic_mute:
                print("bot")
                #  your mic is muted
                mic_mute = True
                #  resets escape state just in case
                escape_2 = False
            #  sends mic mute/unmute shortcut
            keyboard.send(alt_key, Keycode.A)
```

The Great Escape

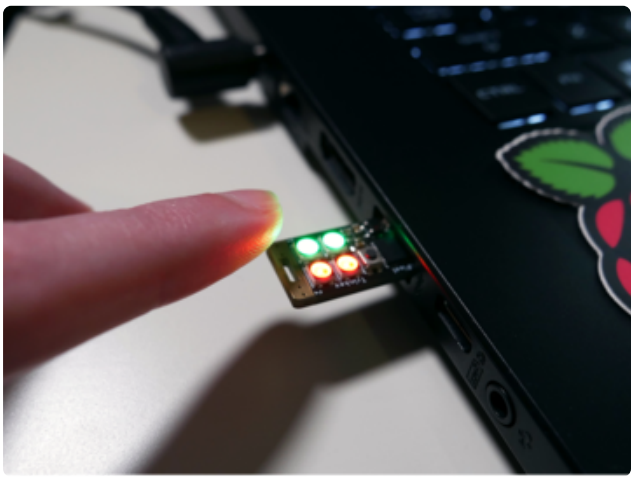
If you hold down both capacitive touch pads at the same time, escape is set to `True` after 2 seconds. This delay ensures no accidental exits from the meeting.

Then, the shortcut for exiting a meeting (ALT+Q) is sent, followed by a slight delay before sending the Enter key. Zoom puts up a confirmation window before you leave or end a meeting, so by sending Enter it fully automates your exit.

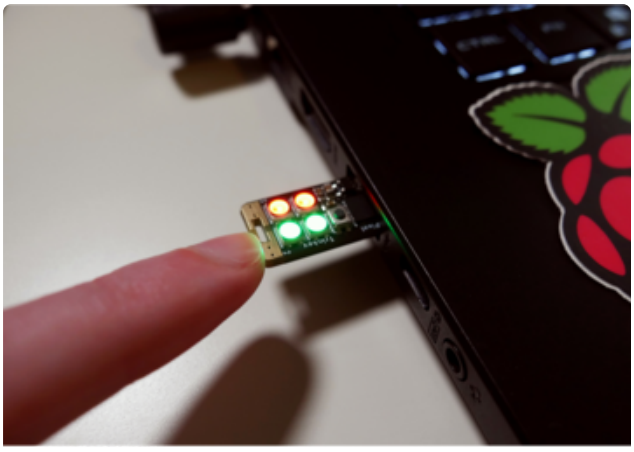
```
#  if you held down both cap touch pads and 2 seconds has passed...
    if ((clock + 2) &lt; time.monotonic()) and (escape_1 and escape_2):
        print("escape")
        #  full escape state is set
        escape = True
        #  sends exit meeting shortcut
        keyboard.send(alt_key, Keycode.Q)
        #  brief delay for confirmation window to open
        time.sleep(0.1)
        #  sends enter to confirm meeting exit
        keyboard.send(Keycode.ENTER)
```

Usage

Before using the Neo Trinkey for Zoom shortcuts, its recommended to setup your default video and microphone mute settings in Zoom so that they match the code.

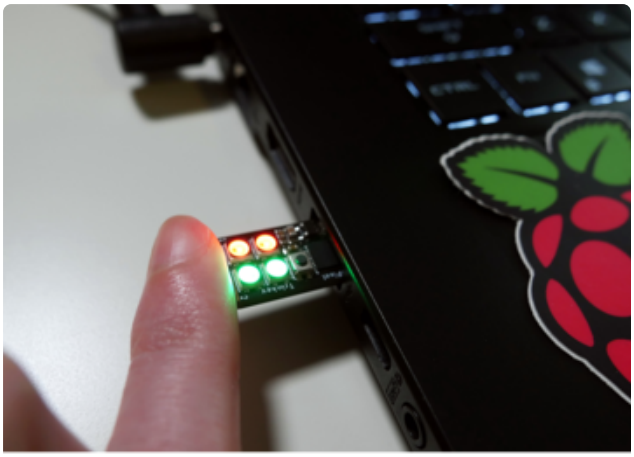


When you're in a meeting, you can tap the top touch pad to mute or unmute your camera. The two NeoPixels directly next to the pad will light up either green or red depending on the state of the camera.



You can tap the bottom touch pad to mute or unmute your microphone. The bottom two NeoPixels will light up either green or red to show you if you're muted or unmuted.

No more having to hear that you're muted by your fellow Zoom meeting participants and no more delay in unmuting by fiddling with your mouse or keyboard.



To exit a meeting, press down on both touch pads at the same time and hold for about one second. Two seconds later, the keyboard shortcut to exit will be sent and you can have a mini celebration with the rainbow cycle animation on the NeoPixels.

If you use a different video conferencing software, you can change the keyboard shortcuts in the code to match. You can also change which touch pad is controlling the mic or webcam. Hopefully this project makes interacting with your virtual meetings easier and fun.