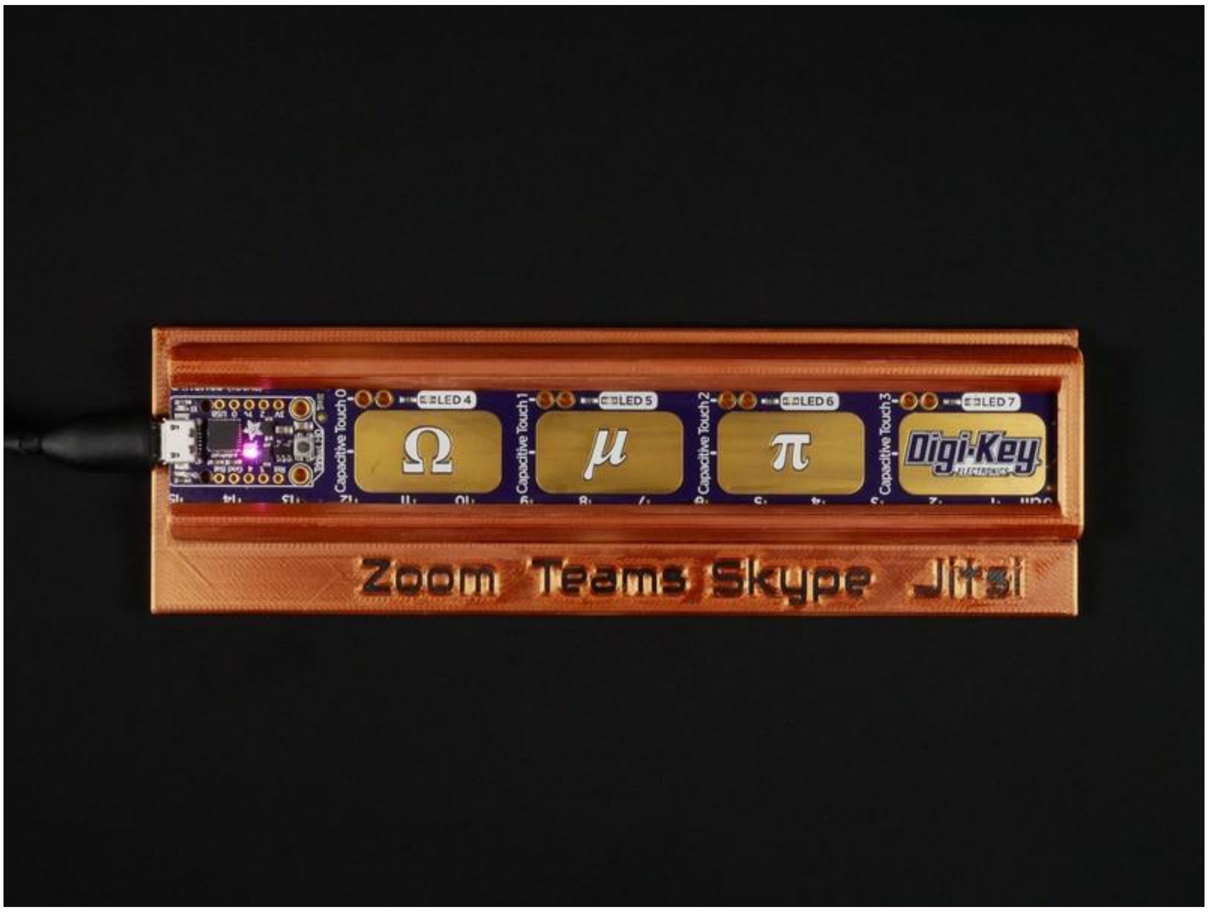
PyRuler Video Conference Panic Buttons

Created by John Thurmond

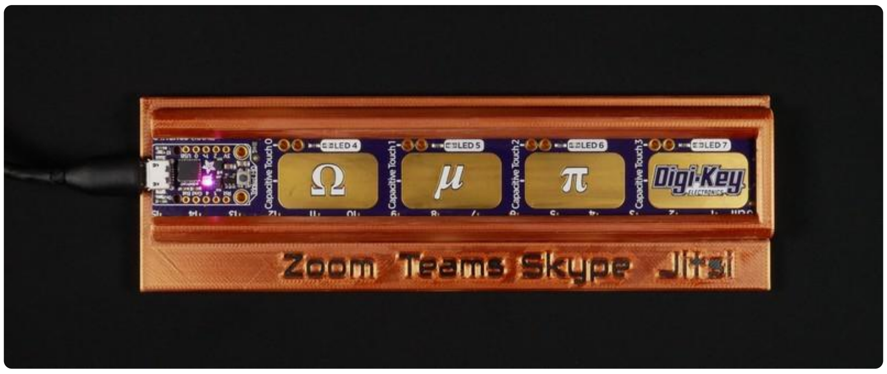


https://learn.adafruit.com/PyRulerVideoPanic

Last updated on 2023-08-29 04:29:50 PM EDT

Table of Contents

Overview



Have you ever been in a video chat, and had some kind of disaster happen that left you scrambling to mute and cut your video feed? I know I have - the most recent being when I spilled hot coffee on my lap. PANIC!

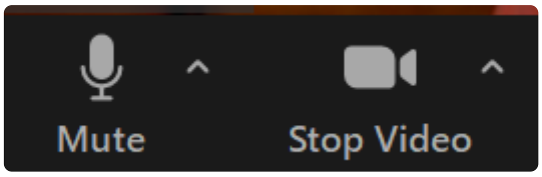
This project was inspired by Simon Prickett's Zoom Panic Button, which allows a user to both mute and cut video at the push of an arcade switch embedded in a Starbucks cup.

Like many of us, I'm spending a lot more time on video calls these days, but Zoom isn't the only video conferencing software I use. Most of my work meetings are on Microsoft Teams or Skype, and personal meetings are on either Zoom or Jitsi.

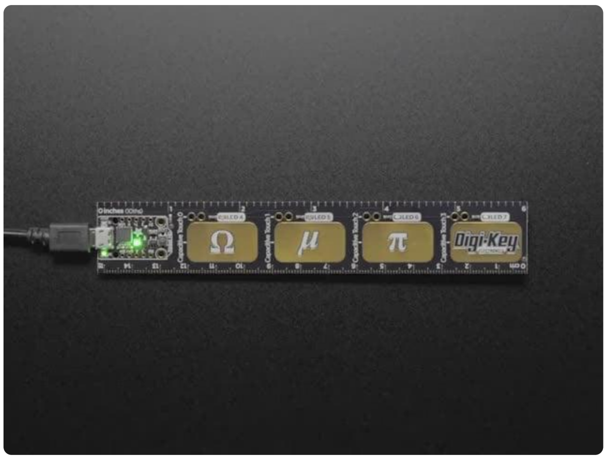
Each one of these applications uses its own hotkeys for muting and stopping the camera, so one button wouldn't do the trick, and my desk is already littered enough with Starbucks cups, so I thought it would be best to build something new!

As it turns out, I had a Trinket M0 already embedded with four buttons in the form of an Adafruit PyRuler. Even better, it's already pre-programmed as a USB HID device, so it was ready to go as a keyboard entry device.

This is a quick and easy project that lets you program your PyRuler as a multi-application panic switch. If you have access to a 3D printer, you can also create a labeled enclosure for it for when you PANIC!



Parts



Adafruit PyRuler - Engineer Reference Ruler with CircuitPython

The first time you soldered up a surface mount component you may have been surprised "these are really small parts!" and there's a dozen of different names...

https://www.adafruit.com/product/4319

USB cable - USB A to Micro-B

This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...

https://www.adafruit.com/product/592

Code

This project doesn't require much (any) knowledge of CircuitPython to get up and running, but if you're new to it, and would like to know more, there is a full getting started guide here (). There's also a great introduction to the PyRuler here ().

If you'd like to edit the code, Adafruit suggests using the Mu editor to edit your code and have an interactive REPL in CircuitPython. You can learn about Mu and installation in this tutorial ().

If you haven't already reprogrammed your PyRuler, it should come pre-programmed as a USB HID keyboard that you can try out. Plug the PyRuler into your computer USB port via an A to microB cable. A drive named CIRCUITPY should appear. Open the code.py file on that drive in Mu () and change this line:

```
ENABLE_KEYBOARD = False
```

to:

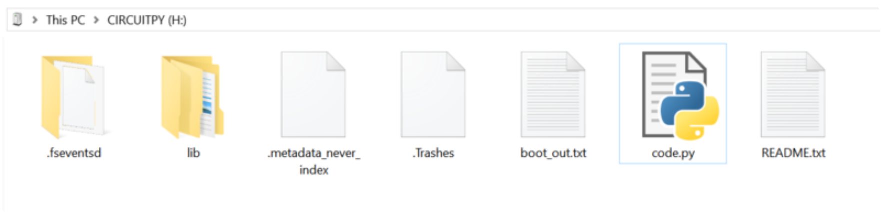
`ENABLE_KEYBOARD = True`

Then when you hit one of the four buttons, it will type Ω, µ, π, or https://www.digikey.com/python () (I totally just typed those on my PyRuler!)

Quick Install

Because we already have everything we need to use the PyRuler as an HID keyboard, getting up and running is very simple:

- Download the code as a code.py file
- Plug the PyRuler into the computer with a microUSB power/data cable
- Once the CIRCUITPY drive appears, just drag the code.py file you downloaded into the CIRCUITPY folder and replace the code.py file that's there
- The PyRuler should immediately reboot, and start working as a panic button for your video conferences!



```
# SPDX-FileCopyrightText: 2020 John Thurmond for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import os
import board
from digitalio import DigitalInOut, Direction
import time
import touchio

# Set this to True to turn the touchpads into a keyboard
ENABLE_KEYBOARD = True

# Used if we do HID output, see below
if ENABLE_KEYBOARD:
    from adafruit_hid.keyboard import Keyboard
    from adafruit_hid.keycode import Keycode
    from adafruit_hid.keyboard_layout_us import KeyboardLayoutUS
    import usb_hid
    kbd = Keyboard(usb_hid.devices)
    layout = KeyboardLayoutUS(kbd)

#print(dir(board), os.uname()) # Print a little about ourselves

led = DigitalInOut(board.D13)
led.direction = Direction.OUTPUT
```

```
touches = [DigitalInOut(board.CAP0)]
for p in (board.CAP1, board.CAP2, board.CAP3):
    touches.append(touchio.TouchIn(p))

leds = []
for p in (board.LED4, board.LED5, board.LED6, board.LED7):
    led = DigitalInOut(p)
    led.direction = Direction.OUTPUT
    led.value = True
    time.sleep(0.25)
    leds.append(led)
for led in leds:
    led.value = False


cap_touches = [False, False, False, False]

def read_caps():
    t0_count = 0
    t0 = touches[0]
    t0.direction = Direction.OUTPUT
    t0.value = True
    t0.direction = Direction.INPUT
    # funky idea but we can 'diy' the one non-hardware captouch device by hand
    # by reading the drooping voltage on a tri-state pin.
    t0_count = t0.value + t0.value + t0.value + t0.value + t0.value + \
               t0.value + t0.value + t0.value + t0.value + t0.value + \
               t0.value + t0.value + t0.value + t0.value + t0.value
    cap_touches[0] = t0_count > 2
    cap_touches[1] = touches[1].raw_value > 3000
    cap_touches[2] = touches[2].raw_value > 3000
    cap_touches[3] = touches[3].raw_value > 3000
    return cap_touches

while True:
    caps = read_caps()
    print(caps)
    # light up the matching LED
    for i,c in enumerate(caps):
        leds[i].value = c
    if caps[0]:
        if ENABLE_KEYBOARD:
            # Zoom
            kbd.press(Keycode.ALT, Keycode.V)
            kbd.release(Keycode.V)
            time.sleep(0.25)
            kbd.press(Keycode.A)
            kbd.release_all()
    if caps[1]:
        if ENABLE_KEYBOARD:
            # Teams
            # Note that video toggle doesn't work in the web app
            kbd.press(Keycode.CONTROL, Keycode.SHIFT, Keycode.M)
            kbd.release(Keycode.M)
            time.sleep(0.5)
            kbd.press(Keycode.O)
            kbd.release_all()
    if caps[2]:
        if ENABLE_KEYBOARD:
            # Skype
            kbd.press(Keycode.CONTROL, Keycode.M)
            kbd.release(Keycode.M)
            time.sleep(0.5)
            kbd.press(Keycode.SHIFT, Keycode.K)
            kbd.release_all()
    if caps[3]:
        if ENABLE_KEYBOARD:
            # Jitsi
            kbd.press(Keycode.M)
```

```
                kbd.release(Keycode.M)
                time.sleep(0.5)
                kbd.press(Keycode.V)
                kbd.release_all()
        time.sleep(.2)
```

More Details

The code is a lightly-modified version of the code that comes on the PyRuler. However, rather than typing the keys for Ω or µ, it presses the combination of hotkeys needed to toggle the camera and microphone in that specific application.

For example, the first button (the Ω) looks like this:

```
# Zoom
kbd.press(Keycode.ALT, Keycode.V)
kbd.release(Keycode.V)
time.sleep(0.25)
kbd.press(Keycode.A)
kbd.release_all()
```

In Zoom, the hotkey for for starting/stopping video is Alt-V, and the hotkey for mute/unmute is Alt-A.

In the code, we're just asking it to press the Alt key, press the V key, release the V key, wait a quarter of a second, press the A key, then release all keys.

"But wait!", you ask, "You didn't hit Alt before the A!"

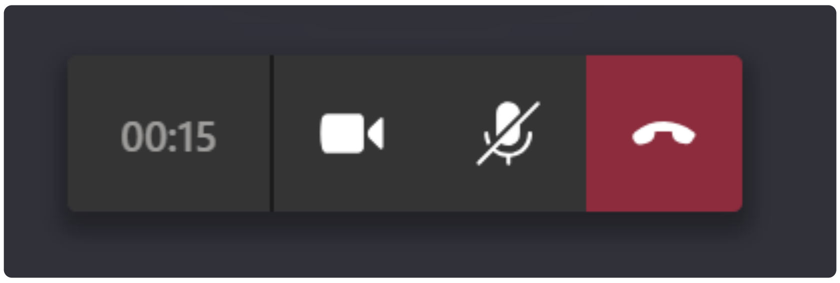
True, but for modifier keys like Alt, they affect all keys pressed while holding them down. They act just like the Shift key, which is just another modifier key. If you hold the Shift key down and START TYPING, all of your letters become capitals - you don't have to press Shift for each one.

The code for Teams looks slightly different:

```
kbd.press(Keycode.CONTROL, Keycode.SHIFT, Keycode.M)
kbd.release(Keycode.M)
time.sleep(0.5)
kbd.press(Keycode.O)
kbd.release_all()
```

Here, we have TWO control keys, both Control and Shift before the M and O keys are pressed. (Ctrl-M is for mute/unmute, and Ctrl-O is for starting/stopping video. You can see why these are difficult to remember in all the apps!

The other thing that's different here is the time between the M and O keys being pressed is increased to half a second. While testing the program, I found that some applications didn't respond very well when the keys were pressed too quickly. Sometimes, this would result in just the microphone being muted, but the video remaining on.



Even worse, as these hotkeys are toggles (they turn things both off and on), if you'd press the button on the PyRuler again, you could unmute the microphone and turn off the video!

Real-world testing is important, and sometimes small tweaks are needed in the code!

Enclosure

Parts

The PyRuler is a little thin to sit nicely on your desk, so I designed a small 3D-printable enclosure for it. This isn't necessary for the project to function, but it makes it a bit nicer to use.

The enclosure is printed in two separate parts that need to be glued together. I recommend cyanoacrylate glue (super glue).

PyRuler_Base.stl () is the base of the ruler with the labels and a groove for the cover.

PyRuler_Cover.stl () sits in the groove on the base and holds the PyRuler in place.

The cover must be printed with supports. These should be easy to remove, but be sure to remove them thoroughly, or the ruler won't slide in easily. The base doesn't require any support. I recommend a layer height of .2mm, and it's a quick print (~2 hours).

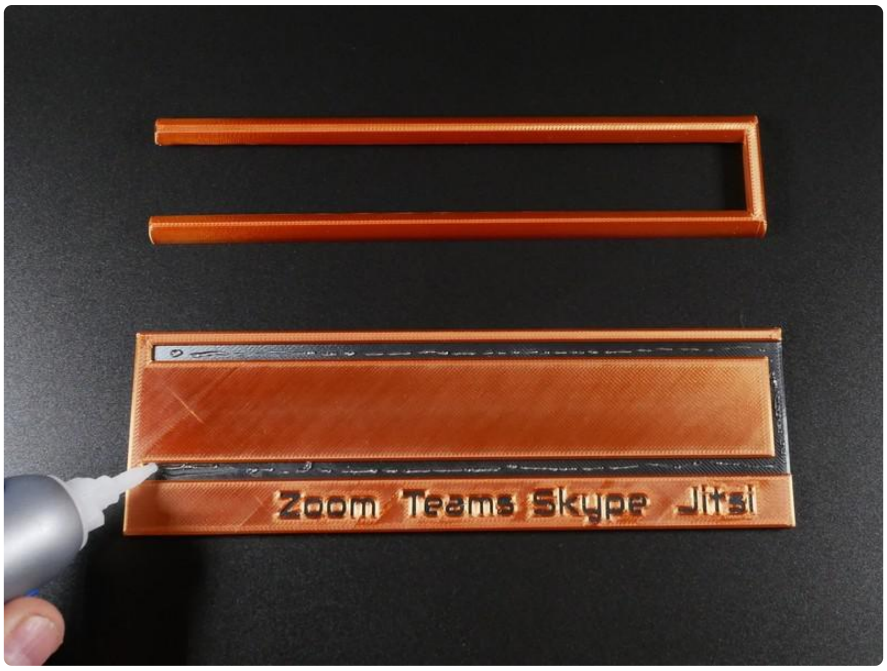
The base can be printed in two different colors by splitting the layers at 2.2mm of height and changing the filament to a contrasting color.

You can of course also print in a single color and use a marker or paint to color in the lettering.

A non-labeled version of the base () is also available, in case you'd prefer to make your own, or customize it!

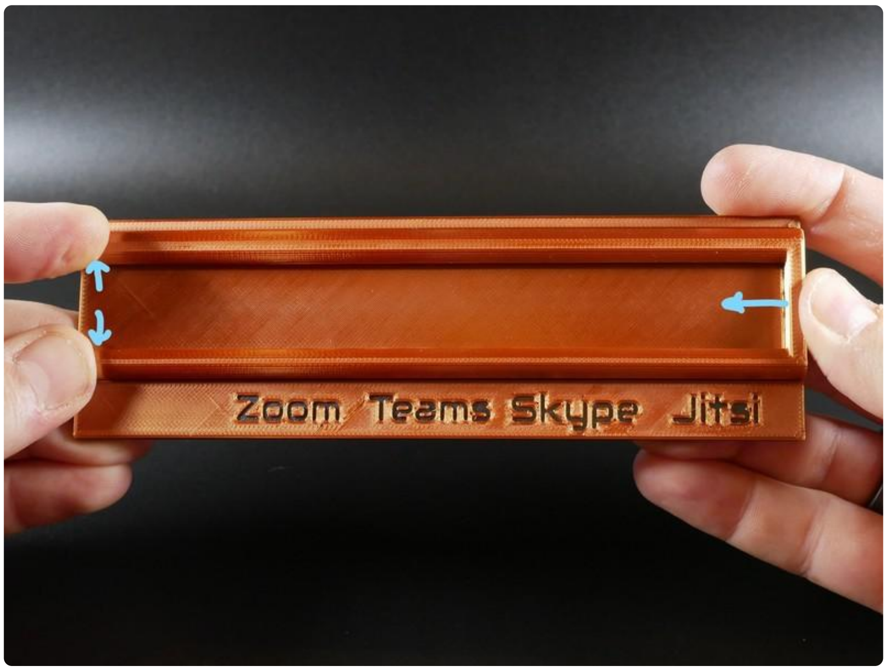
Assembly

Assembling the enclosure is easy, it's just two parts that need to be glued together.



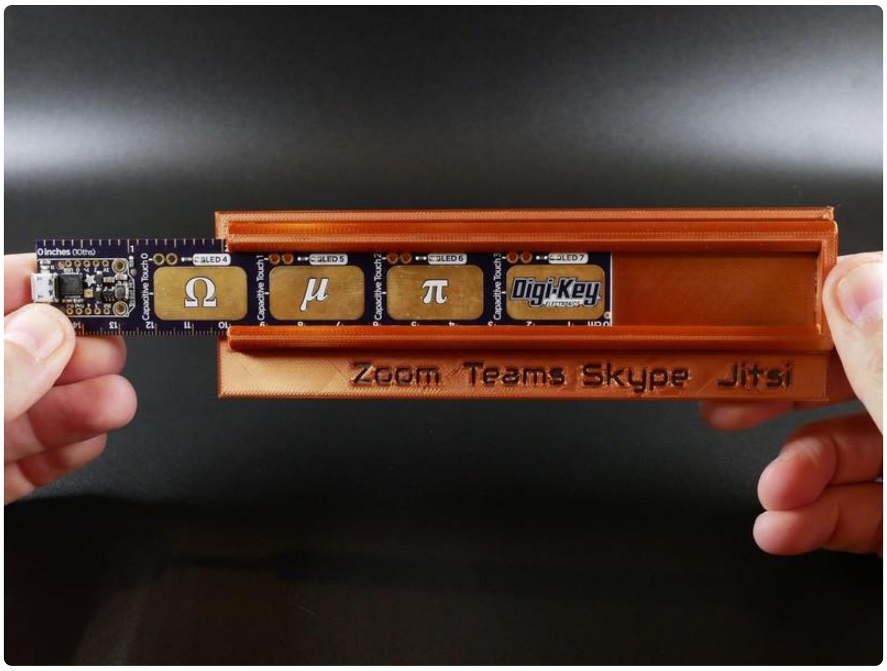
Apply a very thin bead of the cyanoacrylate glue to the 'groove' on the base - a little goes a long way!

The cover should fit into this groove easily, but shouldn't move around much if at all.



Press the cover firmly into place, and apply pressure to the right side, while also holding the open ends on the left side open. This ensures that the ruler will slide in easily, but won't move around inside the enclosure.

This glue sets very quickly, so you should only have to hold this for 30 seconds or so.



Then you simply slide the PyRuler all the way into the enclosure, where it will be held in place by two little bumps on the base.

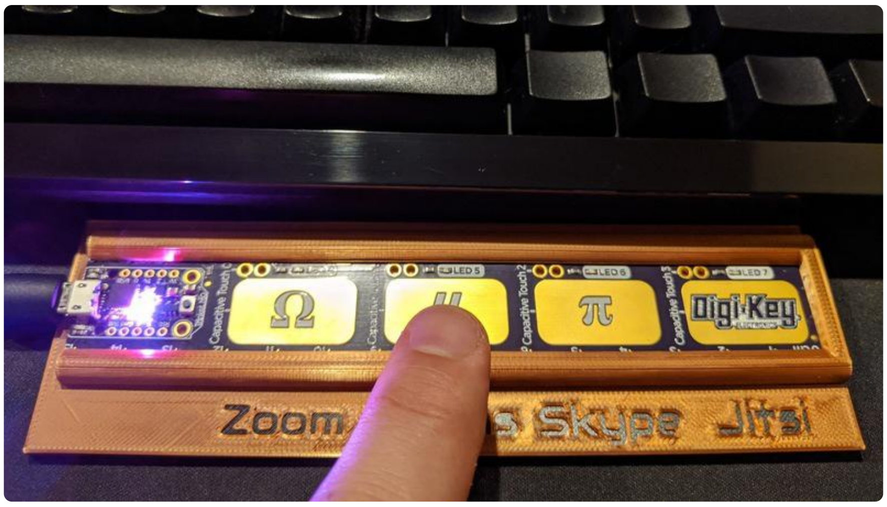
If you ever need to take the ruler out, you may need to use a hobby knife to shave off the bumps to slide it out.

That's it! Plug it in, and now you're ready in case you PANIC!

Usage

Plug a microUSB power/data cable into the PyRuler and then into the computer. The PyRuler should be recognized as both a HID Keyboard and a drive.

Once it's recognized by the computer, just press the capacitive touch sensor corresponding to your video conference application to mute the audio and stop the video. Another press should turn it back on once you've finished panicking!



A couple of usage pointers:

- Make sure your video conferencing application is in focus - it's the window that is in the foreground and you're not accidentally hitting hotkeys in other applications. This could have unexpected results!
- Hit the button that corresponds to the application you're using - hitting the wrong button could also have odd consequences.

Going Further

In the Code section, you've seen how to change the hotkeys, and learned the importance of real-world testing. There are a lot of potential improvements to this project that are worth exploring.

- Do you need to control a different application? Find the hotkeys, change the code, and try it out!
- The LEDs on the PyRuler currently light up when pressing the button. As we're toggling something, perhaps it would be better if the lights instead turned on/off with each button press?
- The buttons are capacitive sensors, and the symbols don't match the functions. Could you put some kind of covering over the buttons that still allow the touch sensors to work?
- Rather than using the touch sensors, you could always wire your own capacitive touch pads or switch. What kind of panic switch would you prefer?
- Can you design a better enclosure - maybe one that can be printed as a single piece?
- Could you use the Trinket M0 with a single big panic button and a rotary switch to change between applications?

If you make any improvements, or even just make your own version, I'd love to see it! You can always reach out to me @grajohnt on Twitter.