PyPortal Guitar Tuner

Created by Ruiz Brothers

https://learn.adafruit.com/pyportal-guitar-tuner

Last updated on 2024-06-03 03:08:24 PM EDT

Table of Contents

Overview 3

- CircuitPython Power!
- 3D Printed PyPortal Case
- Parts

3D Printing 5

- Parts List
- CAD Assembly
- Slicing Parts
- Design Source Files

Install CircuitPython 6

- Set up CircuitPython Quick Start!
- PyPortal Default Files

Coding the PyPortal Guitar Tuner 9

- Installing Project Code

CircuitPython Code Walkthrough 11

- Setup
- The Loop

Assembly 14

- Hardware Setup
- Install Standoffs
- Screen Cover
- Install Screen Cover
- Installed Screen Cover
- Installing Plate and Speaker
- Connect Speaker
- Install Speaker
- Secure PyPortal
- Install Frame
- Secure Frame
- Final Build
- Going Further

Overview

CircuitPython Power!

Build a simple Guitar Tuner with CircuitPython and Adafruit PyPortal! Use the touch screen to tap on tuning pegs and play music notes. The notes are pre-recorded wav audio files of guitar strings. The graphic of the head stock is a single bitmap. This uses the displayio library for CircuitPython and can be customized to make a unique sound board.

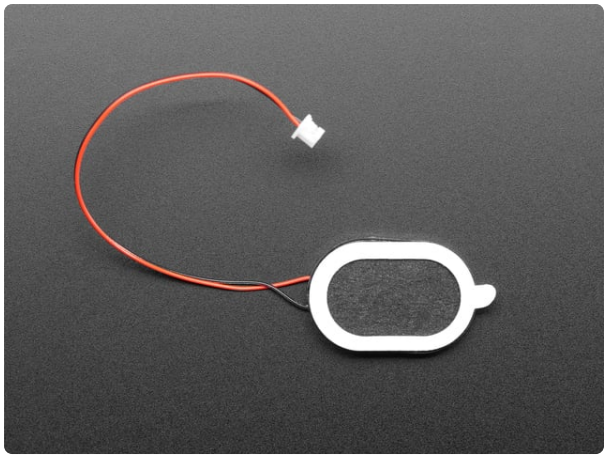
3D Printed PyPortal Case

The Adafruit PyPortal and a mini oval speaker are secured to a 3D printed enclosure using M2.5 screws and standoffs. The PyPortal is mounted vertically and features a built-in holder for a speaker.

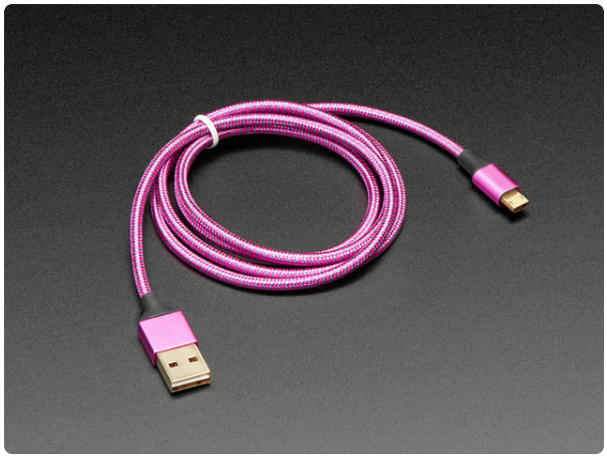
Parts

Adafruit PyPortal - CircuitPython Powered Internet Display

PyPortal, our easy-to-use IoT device that allows you to create all the things for the “Internet of Things” in minutes. Make custom touch screen interface...
https://www.adafruit.com/product/4116



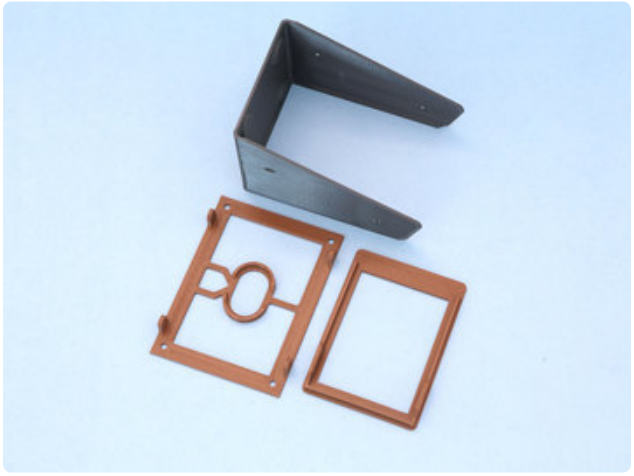
[Mini Oval Speaker - 8 Ohm 1 Watt]
Hear the good news! This wee speaker is a great addition to any audio project where you need 8 ohm impedance and 1W or less of power. We particularly like...
https://www.adafruit.com/product/3923



[Fully Reversible Pink/Purple USB A to micro B Cable - 1m long]
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...
https://www.adafruit.com/product/4111

[Black Nylon Machine Screw and Stand-off Set – M2.5 Thread]
Totaling 380 pieces, this M2.5 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel your maker...
https://www.adafruit.com/product/3299

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.

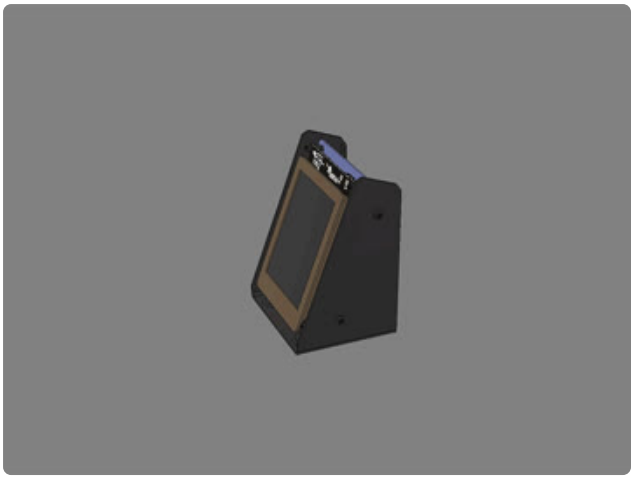
frame.stl
screen-cover.stl
pcb-plate.stl

Download CAD files from Fusion 360

https://adafru.it/LBu

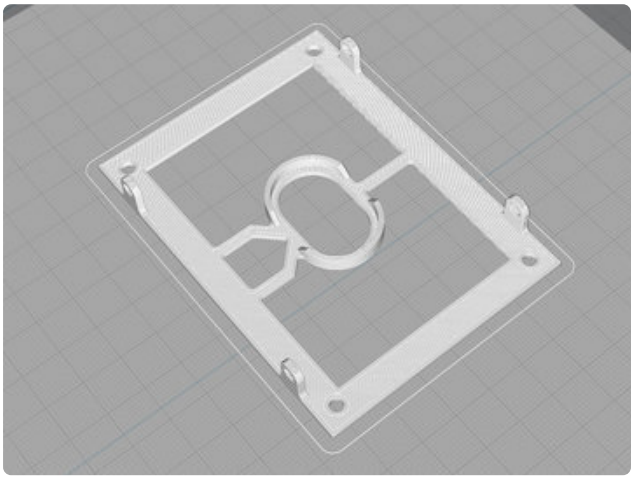
Download CAD files from Thingiverse

https://adafru.it/LBv



CAD Assembly

The PyPortal is secured to the PCB plate using M2.5 hardware screws and standoffs. The screen cover is press fitted over the PyPortal's display. The PCB plate is secured to the frame using M2.5 hardware screws and hex nuts. The speaker is press fitted into the holder in the center of the PCB plate.

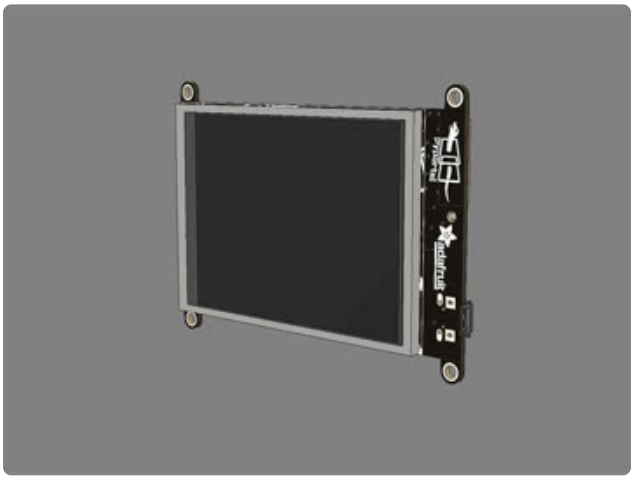


Slicing Parts

No supports are required. Slice with settings for PLA material.

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

PLA filament 220c extruder
0.2 layer height
10% gyroid infill
60mm/s print speed
60c heated bed



Design Source Files

The project assembly was designed in Fusion 360. This can be downloaded in different formats like STEP, STL and more. Electronic components like Adafruit's board, displays, connectors and more can be downloaded from the Adafruit CAD parts GitHub Repo (https://adafru.it/AW8).

Install CircuitPython

CircuitPython (https://adafru.it/tB7) is a derivative of MicroPython (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** "flash" drive to iterate.

The following instructions will show you how to install CircuitPython. If you've already installed CircuitPython but are looking to update it or reinstall it, the same steps work for that as well!

Set up CircuitPython Quick Start!

Follow this quick step-by-step for super-fast Python power :)

Download the latest version of CircuitPython for the PyPortal via CircuitPython.org

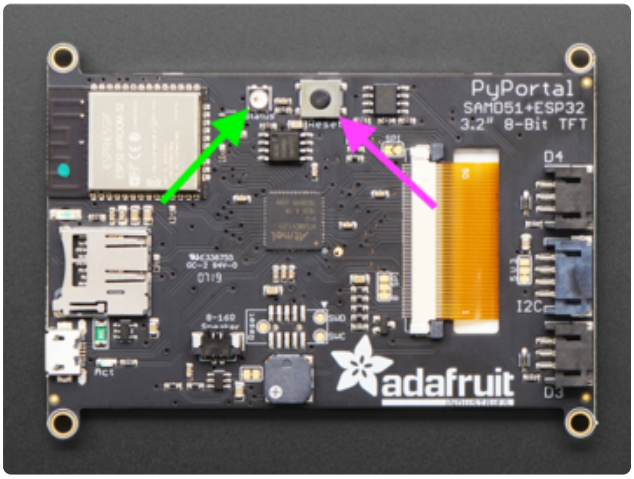
https://adafru.it/Egk

Download the latest version of CircuitPython for the PyPortal Pynt via CircuitPython.org

https://adafru.it/HFd

Click the link above to download the latest version of CircuitPython for the PyPortal.

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

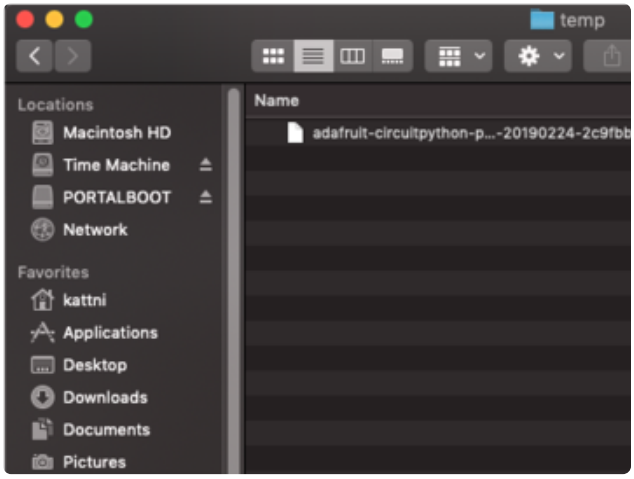


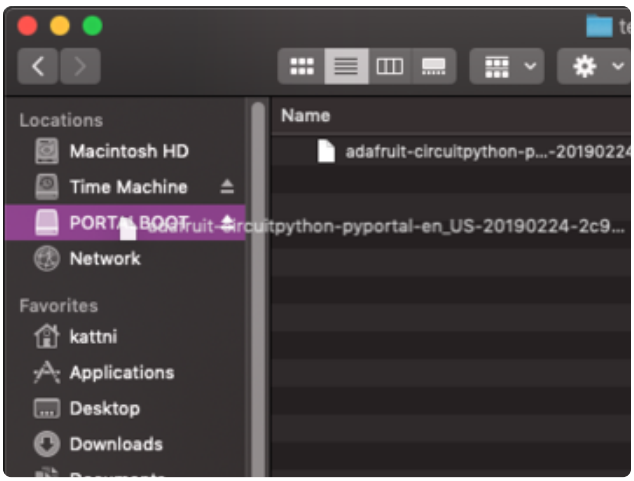
Plug your PyPortal into your computer using a known-good USB cable.

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

Double-click the **Reset** button on the top in the middle (magenta arrow) on your board, and you will see the NeoPixel RGB LED (green arrow) turn green. If it turns red, check the USB cable, try another USB port, etc. **Note:** The little red LED next to the USB connector will pulse red. That's ok!

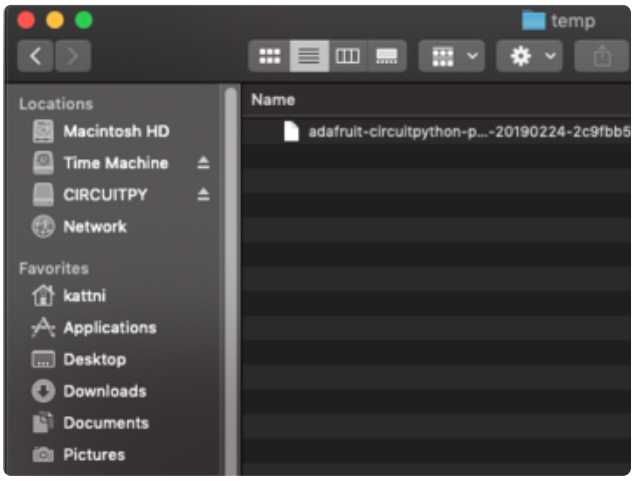
If double-clicking doesn't work the first time, try again. Sometimes it can take a few tries to get the rhythm right!





You will see a new disk drive appear called **PORTALBOOT**.

Drag the **adafruit-circuitpython-pyportal-<whatever>.uf2** file to **PORTALBOOT.**



The LED will flash. Then, the **PORTALBOOT** drive will disappear and a new disk drive called **CIRCUITPY** will appear.

If you haven't added any code to your board, the only file that will be present is **boot_out.txt**. This is absolutely normal! It's time for you to add your **code.py** and get started!

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

PyPortal Default Files

Click below to download a zip of the files that shipped on the PyPortal or PyPortal Pynt.

PyPortal Default Files

https://adafru.it/UF-

PyPortal Pynt Default Files

https://adafru.it/UGa

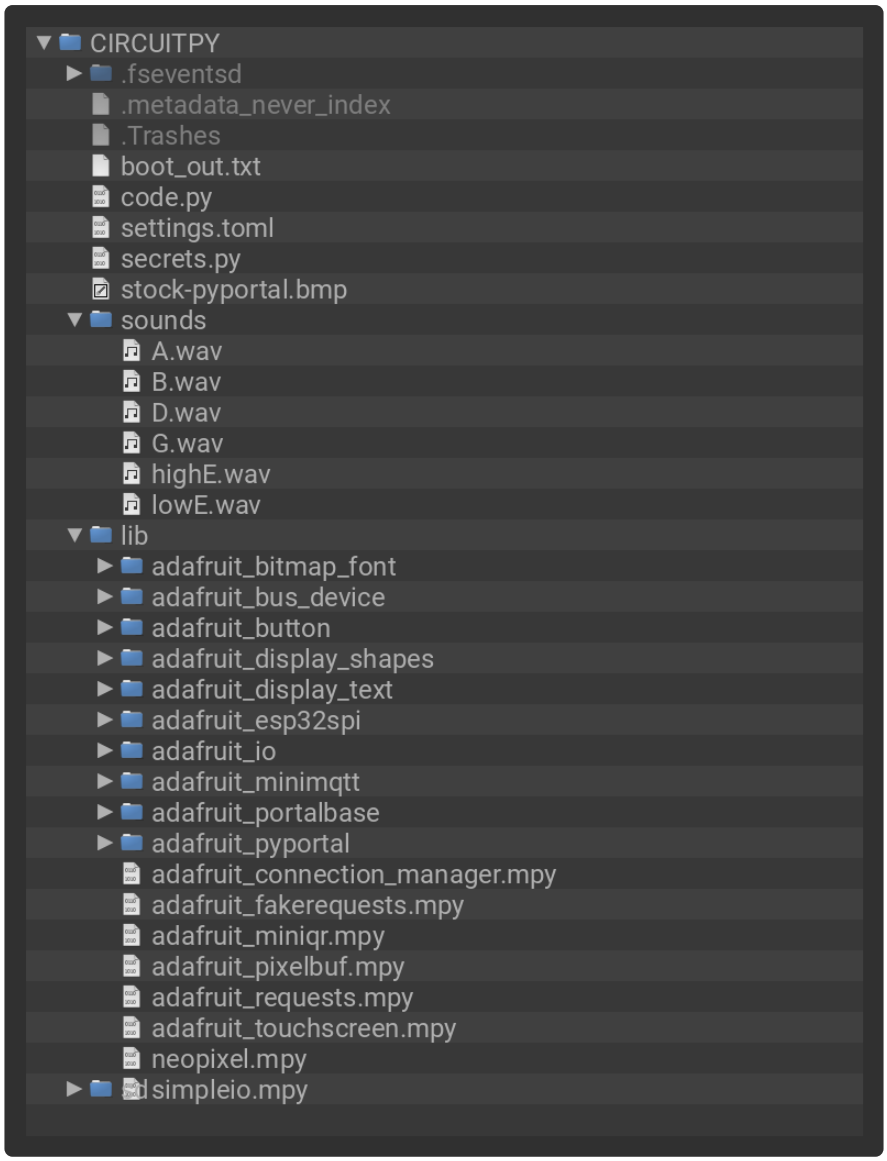
Coding the PyPortal Guitar Tuner

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 **PyPortal_Guitar_Tuner/** 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: 2020 Liz Clark for Adafruit Industries
#
# SPDX-License-Identifier: MIT

import time
from adafruit_button import Button
from adafruit_pyportal import PyPortal

pyportal = PyPortal(default_bg="/stock-pyportal.bmp")

lowE = "/sounds/lowE.wav"
A = "/sounds/A.wav"
D = "/sounds/D.wav"
G = "/sounds/G.wav"
B = "/sounds/B.wav"
highE = "/sounds/highE.wav"

notes = [lowE, A, D, G, B, highE]

pegs = [
    {'label': "lowE", 'pos': (53, 0), 'size': (65, 90)},
    {'label': "A", 'pos': (124, 0), 'size': (65, 90)},
    {'label': "D", 'pos': (194, 0), 'size': (65, 90)},
    {'label': "G", 'pos': (194, 150), 'size': (65, 90)},
    {'label': "B", 'pos': (124, 150), 'size': (65, 90)},
    {'label': "highE", 'pos': (53, 150), 'size': (65, 90)}
```

```
    ]

buttons = []
for peg in pegs:
    button = Button(x=peg['pos'][0], y=peg['pos'][1],
                    width=peg['size'][0], height=peg['size'][1],
                    style=Button.RECT,
                    fill_color=None, outline_color=0x5C3C15,
                    name=peg['label'])
    pyportal.splash.append(button.group)
    buttons.append(button)

note_select = None

while True:
    touch = pyportal.touchscreen.touch_point
    if not touch and note_select:
        note_select = False
    if touch:
        for i in range(6):
            tuning = notes[i]
            button = buttons[i]
            if button.contains(touch) and not note_select:
                print("Touched", button.name)
                note_select = True
                for z in range(3):
                    pyportal.play_file(tuning)
    time.sleep(0.1)
```

CircuitPython Code Walkthrough

Setup

CircuitPython Libraries

The CircuitPython code begins by importing the libraries.

```
import time
from adafruit_button import Button
from adafruit_pyportal import PyPortal
```

Display Setup

First, `pyportal` is setup as a PyPortal object. The PyPortal's default background is also setup to be the guitar headstock image, **stock-pyportal.bmp**. This means that on boot, the PyPortal will display the bitmap with just one line of code.

```
pyportal = PyPortal(default_bg="/stock-pyportal.bmp")
```

Loading Audio Files

The audio file locations are assigned to variables so that they can be easily referenced in the code. Then, files are put into the `notes` array in order from low to high.

```
lowE = "/sounds/lowE.wav"
A = "/sounds/A.wav"
D = "/sounds/D.wav"
G = "/sounds/G.wav"
B = "/sounds/B.wav"
highE = "/sounds/highE.wav"

notes = [lowE, A, D, G, B, highE]
```

Touchscreen Buttons

The PyPortal will have six buttons that sit on top of the guitar headstock image. Instead of setting each button up individually, their parameters are setup as a group in an array, called `pegs`, of dictionary entries. This way you can easily denote the buttons' label, position and size.

```
pegs = [
    {'label': "lowE", 'pos': (53, 0), 'size': (65, 90)},
    {'label': "A", 'pos': (124, 0), 'size': (65, 90)},
    {'label': "D", 'pos': (194, 0), 'size': (65, 90)},
    {'label': "G", 'pos': (194, 150), 'size': (65, 90)},
    {'label': "B", 'pos': (124, 150), 'size': (65, 90)},
    {'label': "highE", 'pos': (53, 150), 'size': (65, 90)}
    ]
```

Finally, this information for the buttons are assigned as `Button` objects using the adafruit_button CircuitPython library. Using the `pegs` array, all of the information from the dictionaries can be pulled in to complete the setup for the buttons. These buttons are then added to the PyPortal's display.

```
buttons = []
for peg in pegs:
    button = Button(x=peg['pos'][0], y=peg['pos'][1],
                    width=peg['size'][0], height=peg['size'][1],
                    style=Button.RECT,
                    fill_color=None, outline_color=0x5C3C15,
                    name=peg['label'])
    pyportal.splash.append(button.group)
    buttons.append(button)
```

Button State

The `note_select` state will be used to debounce the touchscreen buttons.

```
note_select = None
```

The Loop

The loop begins with `touch` setup to hold the PyPortal's touchscreen functionality.

Next, touchscreen button debouncing is setup. You only need to setup one instance rather than one for each individual button because you are essentially checking to see if the touchscreen is being touched in any location.

```
while True:
    touch = pyportal.touchscreen.touch_point

    if not touch and note_select:
        note_select = False
```

Play Notes

The final portion of the loop is how notes are played through the PyPortal. First, it checks to see if the touchscreen has been touched.

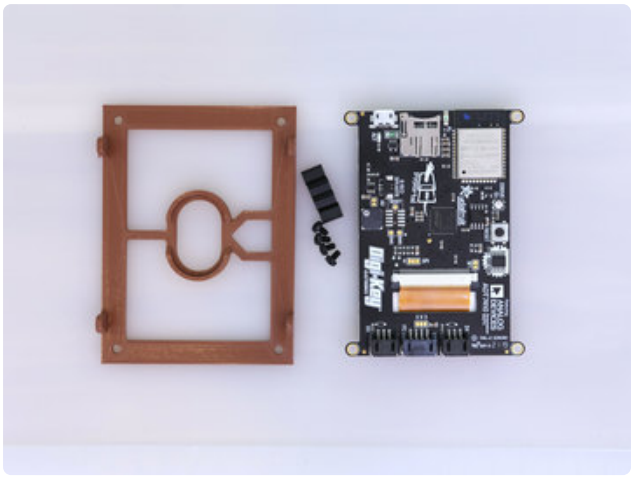
This is followed by a `for` statement. In this `for` statement, `tuning` and `button` are setup to hold the array index locations for the `notes` and `buttons` arrays. Since there are six indexes in each of these arrays, this allows for the sound files to match up with the touchscreen buttons and play when pressed.

Finally, an `if` statement checks if the touchscreen was touched in the proximity of the coordinates of one of the button locations. If it was, then the name of the button is printed to the REPL. This is followed by a final `for` statement. This `for` statement allows for the corresponding note's audio file to play three times.

This is followed by a delay. The length of the delay will determine the amount of time between each time the audio files are played. You may want to adjust it depending on your guitar tuning preferences.

```
if touch:
        for i in range(6):
            tuning = notes[i]
            button = buttons[i]
            if button.contains(touch) and not note_select:
                print("Touched", button.name)
                note_select = True
                for z in range(3):
                    pyportal.play_file(tuning)
    time.sleep(0.1)
```

Assembly



Hardware Setup

Use 4x M2.5 x 6mm long FF standoffs and 4x M2.5 x 6mm long screws to secure the PCB plate to the PyPortal.

Install Standoffs

Insert M2.5 x 6mm screws through the top of the mounting tabs on PyPortal. Fasten M2.5 x 6mm long standoffs onto the threads of the screws.

Screen Cover

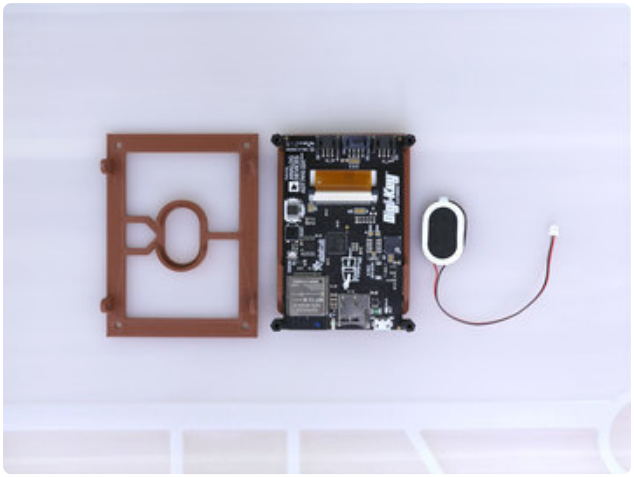
Orient the screen cover with the display on the PyPortal. Use the photo to reference the correct orientation.

Install Screen Cover

Fit the PyPortal display into the recess on the screen cover. Press the screen to fully seat into the screen cover.

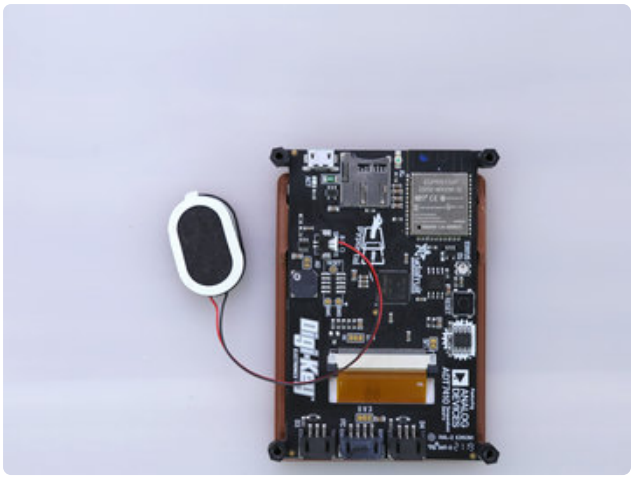
Installed Screen Cover

The screen cover helps to keep the display attached to the PyPortal. The viewing area is exposed and does not obstruct display. This also hides the screens bezel.



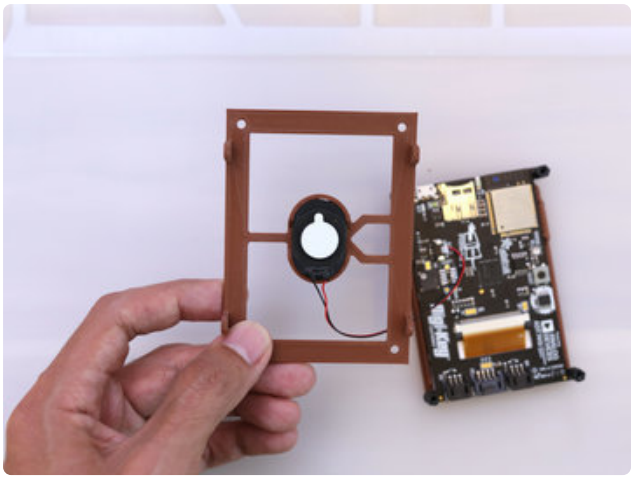
Installing Plate and Speaker

The PyPortal is secured to the PCB plate using 4x M2.5 x 6mm screws. The mini oval speaker is press fitted onto the speaker holder on the PCB plate.



Connect Speaker

Plug in the molex pico connector from the speaker to the speaker port on the back of the PyPortal.

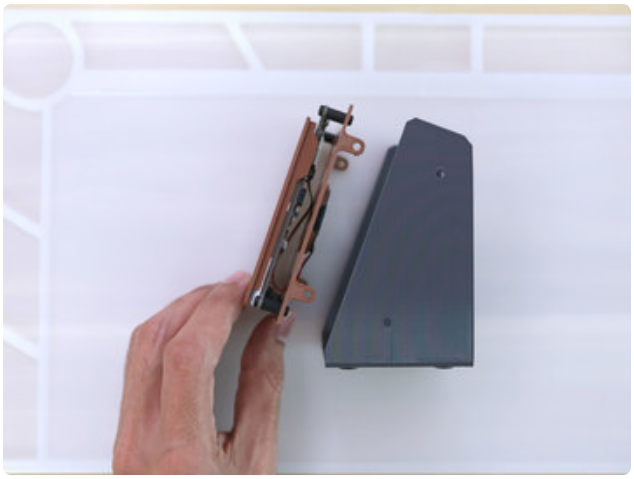


Install Speaker

Press fit the body of the speaker into the speaker holder in the center of the PCB plate. Reference the photo for best orientation.

Secure PyPortal

Place the PCB plate to the back of the PyPortal with the mounting holes lined up with the standoffs.



Install Frame

Orient the frame with the mount tabs on the PCB plate. Line up the mounting holes in the frame with the mounting tabs on the PCB.

Secure Frame

Insert 4x M2.5 x 6mm screws into the side of the frame and through the mounting tabs on the PCB plate. Insert and fasten 4x M2.5 hex nuts onto threads of the screws and tighten.

Final Build

And now we're ready for some guitar tuning!

Going Further

The frame has adequate amount of space for a USB battery or other components.