Modal MIDI Keyboard

Created by John Park

https://learn.adafruit.com/modal-midi-keyboard

Last updated on 2025-03-04 10:17:07 AM EST

Table of Contents

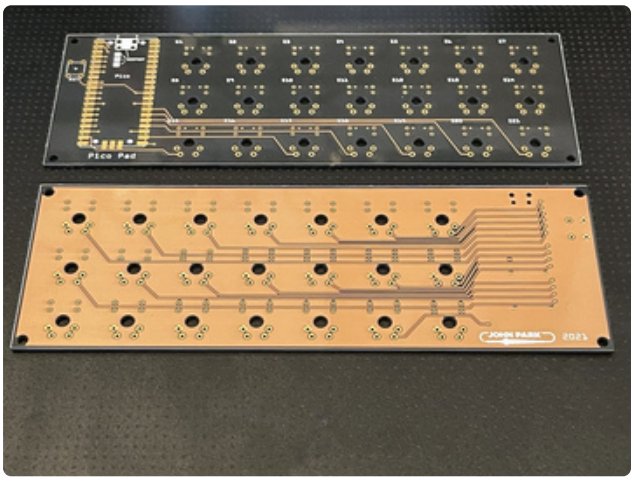
Overview

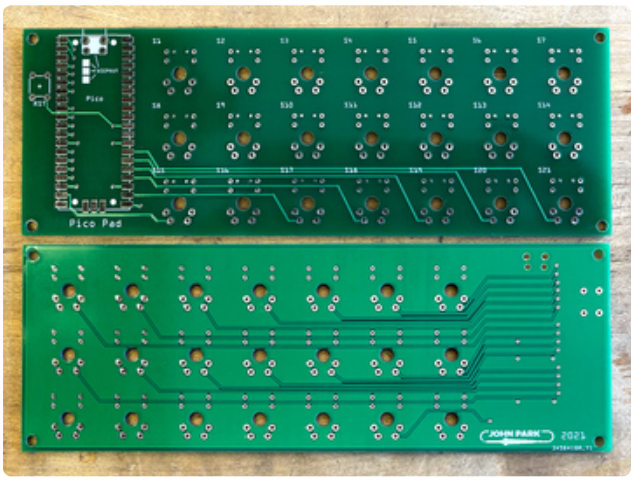
Play great sounding melodies and chords on a synthesizer without lots of piano lessons by sticking with notes that sound good together! Modes, such as **Major/ Ionian, Minor/Aeolian, Dorian,** and **Mixolydian**, to name a few, are sets of relative note intervals designed for this purpose, and now you can build your own keyboard that will play within whichever key and mode you choose -- you can't hit a wrong note!

Your modal keyboard sends notes over USB MIDI to any software synthesizer, or hardware synth with USB MIDI Host capabilities. Pick your key and mode on startup and then start your jam!

For more on modes, check out this video (https://adafru.it/TCh) and take a look at this page (https://adafru.it/TCd).

Parts





PCB

You can order these using the Gerber files found later in the guide from a board house such as JLCPCB, or by visiting this OSH Park link (https://adafru.it/TBP). You only need one PCB per keyboard, but most board houses make them in multiples of three or five for a minimum order.

The keyboard uses 21 keyswitches and keycaps.

Kailh Mechanical Key Switches - 10 packs - Cherry MX Compatible

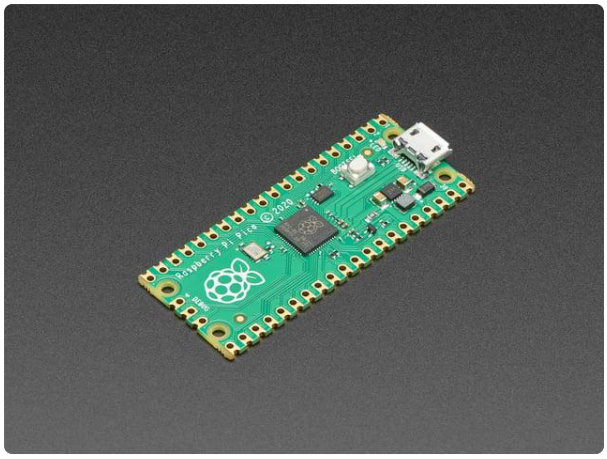
For crafting your very own custom keyboard, these Kailh mechanical key switches are deeee-luxe!Come in a pack of 10 switches, plenty to make a...

https://www.adafruit.com/product/4996

DSA Keycaps for MX Compatible Switches in Various Colors

Dress up your mechanical keys in your favorite colors, with a wide selection of stylish DSA key caps. Here is a 10 pack different colored keycaps for your next mechanical keyboard or...

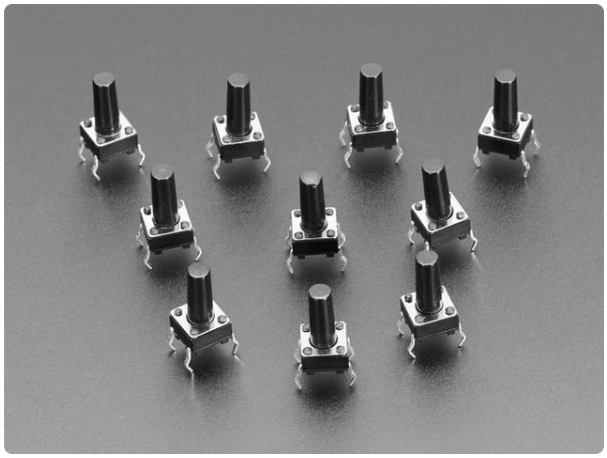
https://www.adafruit.com/product/5097



Raspberry Pi Pico RP2040

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to...

https://www.adafruit.com/product/4864



Tactile Switch Buttons (6mm tall) x 10 pack

Super-tall clicky momentary switches are standard input "buttons" on electronic projects. These work best in a PCB but

https://www.adafruit.com/product/1490

M2.5 x 16mm screws x4

Get at a hardware store or from McMaster-Carr here (https://adafru.it/QNA).

1 x Little Rubber Bumper Feet Pack of 4	https://www.adafruit.com/product/550
1 x USB cable USB A to Micro-B	https://www.adafruit.com/product/592
2 x Brass M2.5 Standoffs 16mm tall pack of 2	https://www.adafruit.com/product/2337
1 x Black Nylon Machine Screw and Stand-off Set M2.5	https://www.adafruit.com/product/3299
1 x Black Nylon Machine Screw and Stand-off Set – M3 Thread Black Nylon Machine Screw and Stand-off Set M3	https://www.adafruit.com/product/4685

Installing 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** drive to iterate.

CircuitPython Quickstart

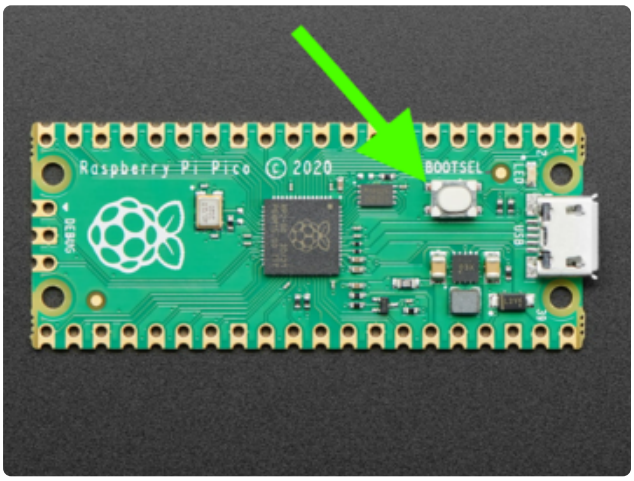
Follow this step-by-step to quickly get CircuitPython working on your board.

Download the latest version of CircuitPython for the Raspberry Pi Pico from circuitpython.org

https://adafru.it/QaP

Click the link above and download the latest UF2 file.

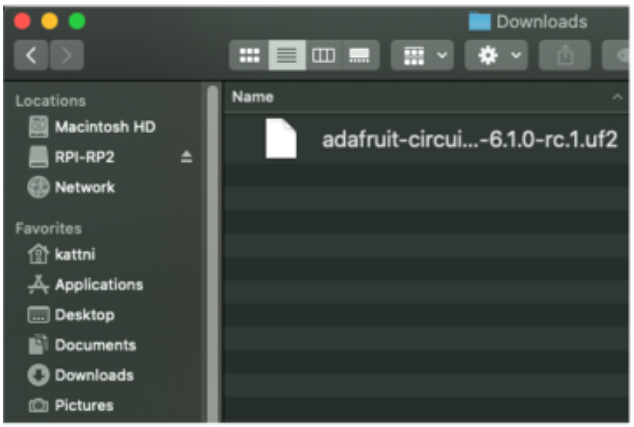
Download and save it to your desktop (or wherever is handy).



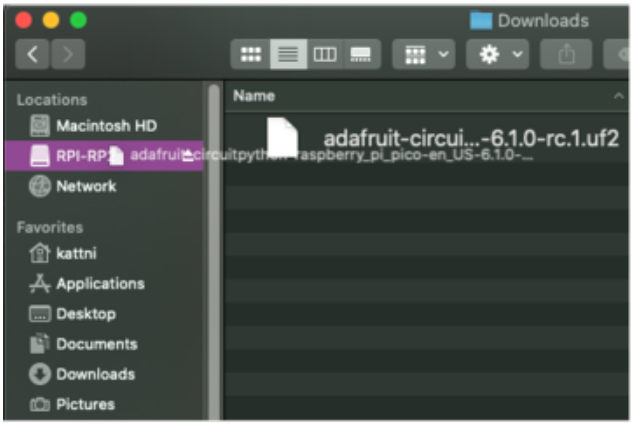
Start with your Pico unplugged from USB. Hold down the **BOOTSEL** button, and while continuing to hold it (don't let go!), plug the Pico into USB. **Continue to hold the BOOTSEL button until the RPI-RP2 drive appears!**

If the drive does not appear, unplug your Pico and go through the above process again.

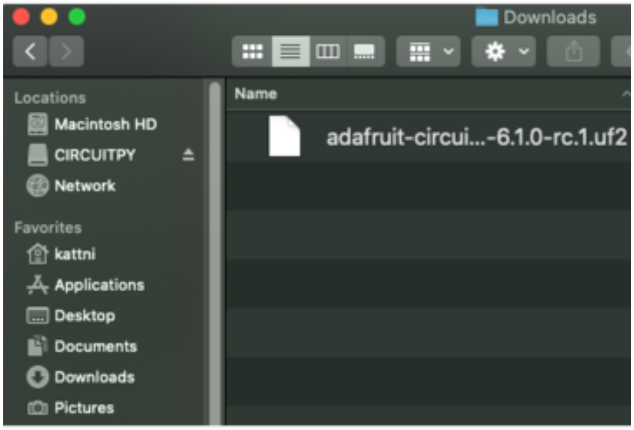
A lot of people end up using charge-only USB cables and it is very frustrating! **So make sure you have a USB cable you know is good for data sync.**



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! :)

Flash Resetting UF2

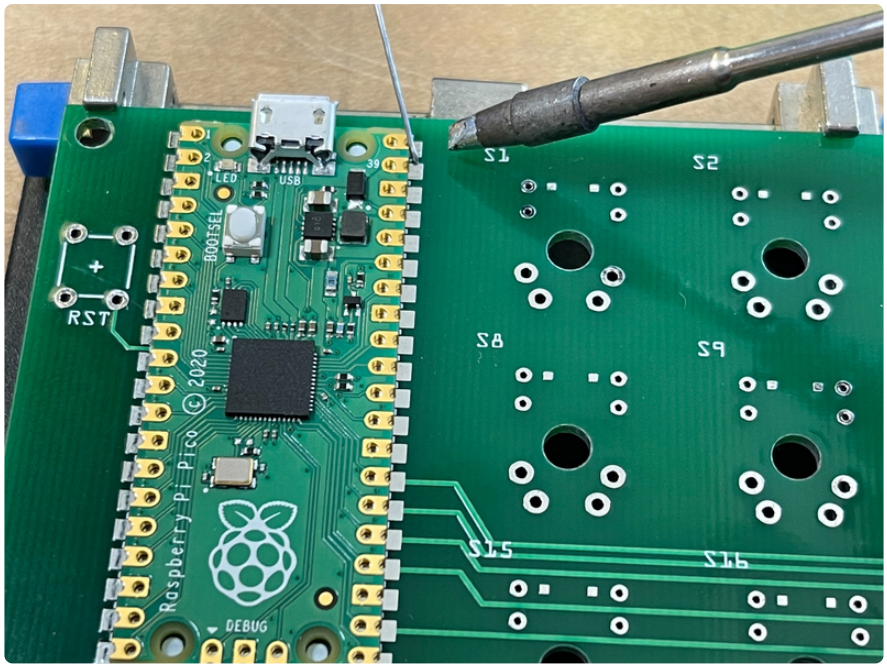
If your Pico ever gets into a really weird state and doesn't even show up as a disk drive when installing CircuitPython, try installing this 'nuke' UF2 which will do a 'deep clean' on your Flash Memory. You will lose all the files on the board, but at least you'll be able to revive it! After nuking,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

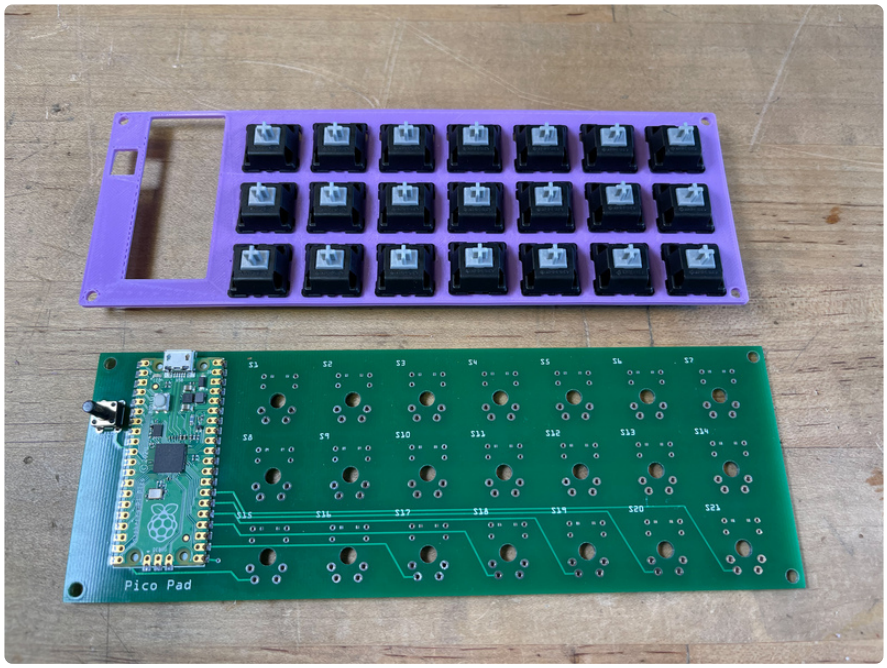
flash_nuke.uf2

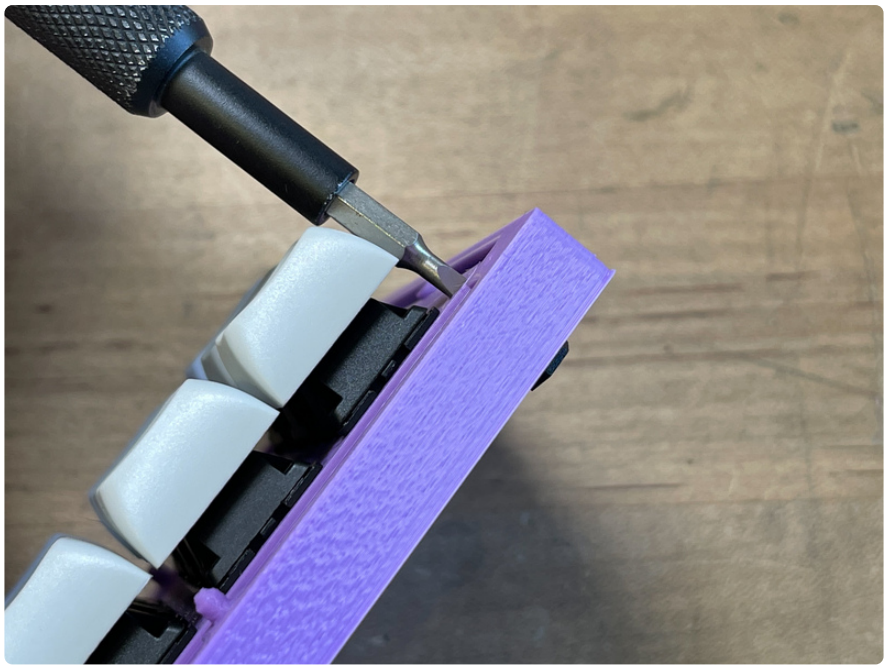
https://adafru.it/1afi

Build the Modal MIDI Keyboard

This guide (https://adafru.it/TCe) is dedicated to designing and building the 21-Key Pico Keyboard. You can jump straight to ordering PCBs using this section of the guide (https://adafru.it/TCf), and then follow the steps for assembly on this page (https://adafru.it/TCg).







Code the Modal MIDI Controller

Text Editor

Adafruit recommends using the Mu editor for using your CircuitPython code with the Pico. You can get more info in this guide (https://adafru.it/ANO).

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

Download the Project Bundle

Your project will use a specific set of CircuitPython libraries and the **code.py** file. In order to get the libraries 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 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

# Pico RP2040 Mechanical MIDI Modal Keyboard
# 7x3 mech keyboard
# Each key sends MIDI NoteOn / NoteOff message over USB
# Can be any scale/mode
# Key combo sends MIDI panic (see bottom section of code)

import time
import board
from digitalio import DigitalInOut, Direction, Pull
import usb_midi
import adafruit_midi
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff
from adafruit_debouncer import Debouncer

print("---Pico MIDI Modal Mech Keyboard---")

MIDI_CHANNEL = 1  # pick your MIDI channel here

midi = adafruit_midi.MIDI(midi_out=usb_midi.ports[1], out_channel=MIDI_CHANNEL-1)

def send_midi_panic():
    print("All MIDI notes off")
    for x in range(128):
        midi.send(NoteOff(x, 0))

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT
led.value = True

num_keys = 21

# list of pins to use (skipping GP15 on Pico because it's funky)
pins = (
    board.GP0,
    board.GP1,
    board.GP2,
    board.GP3,
    board.GP4,
    board.GP5,
    board.GP6,
    board.GP7,
    board.GP8,
    board.GP9,
    board.GP10,
    board.GP11,
    board.GP12,
    board.GP13,
    board.GP14,
    board.GP16,
    board.GP17,
```

```
    board.GP18,
    board.GP19,
    board.GP20,
    board.GP21,
)


keys = []
for pin in pins:
    tmp_pin = DigitalInOut(pin)
    tmp_pin.pull = Pull.UP
    keys.append(Debouncer(tmp_pin))

root_notes = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59)  # used during config
note_numbers = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
                60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
                72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83)
note_names = ("C2", "C#2", "D2", "D#2", "E2", "F2", "F#2", "G2", "G#2", "A2", "A#2",
"B2",
              "C3", "C#3", "D3", "D#3", "E3", "F3", "F#3", "G3", "G#3", "A3", "A#3",
"B3",
              "C4", "C#4", "D4", "D#4", "E4", "F4", "F#4", "G4", "G#4", "A4", "A#4",
"B4",)
scale_root = root_notes[0]  # default if nothing is picked
root_picked = False  # state of root selection
mode_picked = False  # state of mode selection
mode_choice = 0


#  ----- User selection of the root note ----- #
print("Pick the root using top twelve keys, then press bottom right key to enter:")
print(". . . . . .")
print(". . . . o o")
print("o o o o o o .")

while not root_picked:
    for i in range(12):
        keys[i].update()
        if keys[i].fell:
            scale_root = root_notes[i]
            midi.send(NoteOn(root_notes[i], 120))
            print("Root is", note_names[i])
        if keys[i].rose:
            midi.send(NoteOff(root_notes[i], 0))
    keys[20].update()

    if keys[20].rose:
        root_picked = True
        print("Root picked.\n")

#  lists of mode intervals relative to root
major = ( 0, 2, 4, 5, 7, 9, 11 )
minor = ( 0, 2, 3, 5, 7, 8, 10 )
dorian = ( 0, 2, 3, 5, 7, 9, 10 )
phrygian = ( 0, 1, 3, 5, 7, 8, 10 )
lydian = (0 , 2, 4, 6, 7, 9, 11 )
mixolydian = ( 0, 2, 4, 5, 7, 9, 10)
locrian = ( 0, 1, 3, 5, 6, 8, 10)

modes = []
modes.append(major)
modes.append(minor)
modes.append(dorian)
modes.append(phrygian)
modes.append(lydian)
modes.append(mixolydian)
modes.append(locrian)

mode_names = ("Major/Ionian",
```

```
              "Minor/Aeolian",
              "Dorian",
              "Phrygian",
              "Lydian",
              "Mixolydian",
              "Locrian")

intervals = list(mixolydian)  # intervals for Mixolydian by default

print("Pick the mode with top seven keys, then press bottom right key to enter:")
print(". . . . . . .")
print("o o o o o o o")
print("o o o o o o .")

while not mode_picked:
    for i in range(7):
        keys[i].update()
        if keys[i].fell:
            mode_choice = i
            print(mode_names[mode_choice], "mode")
            for j in range(7):
                intervals[j] = modes[i][j]
            # play the scale
            for k in range(7):
                midi.send(NoteOn(scale_root+intervals[k], 120))
                note_index = note_numbers.index(scale_root+intervals[k])
                print(note_names[note_index])
                time.sleep(0.15)
                midi.send(NoteOff(scale_root+intervals[k], 0))
                time.sleep(0.15)
            midi.send(NoteOn(scale_root+12, 120))
            note_index = note_numbers.index(scale_root+12)
            print(note_names[note_index], "\n")
            time.sleep(0.15)
            midi.send(NoteOff(scale_root+12, 0))
            time.sleep(0.15)

    keys[20].update()
    if keys[20].rose:
        print(mode_names[mode_choice], "mode picked.\n")
        mode_picked = True

scale = []  # create the base scale
for i in range(7):
    scale.append(scale_root + intervals[i])

midi_notes = []  # build the list with three octaves
for k in range(7):
    midi_notes.append(scale[k]+24)
for l in range(7):
    midi_notes.append(scale[l]+12)
for m in range(7):
    midi_notes.append(scale[m])

led.value = False
print("Ready, set, play!")


while True:

    for i in range(num_keys):
        keys[i].update()
        if keys[i].fell:
            try:
                midi.send(NoteOn(midi_notes[i], 120))
                note_index = note_numbers.index(midi_notes[i])
                print("MIDI NoteOn:", note_names[note_index])
            except ValueError:  # deals w six key limit
                pass
```

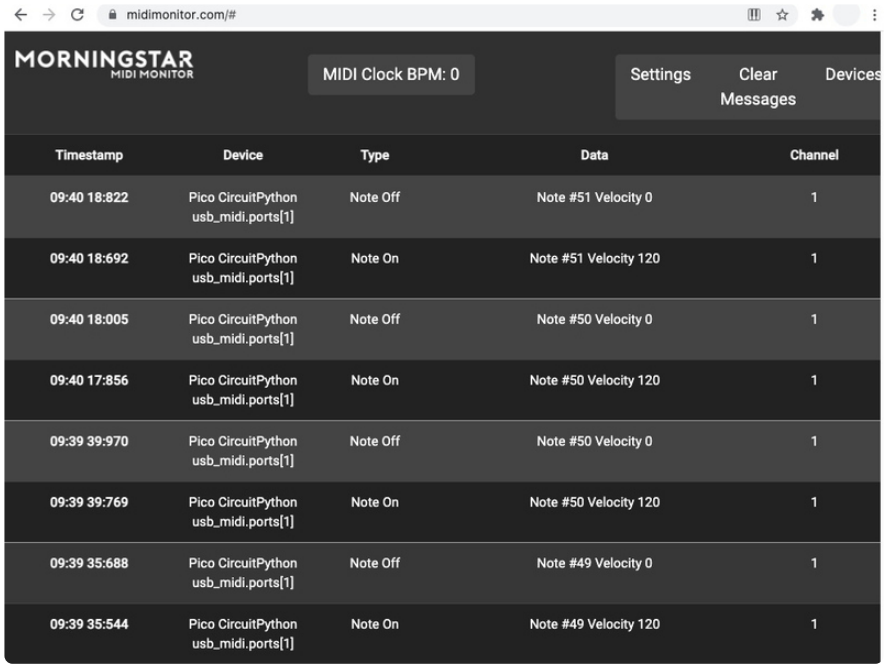
```
        if keys[i].rose:
            try:
                midi.send(NoteOff(midi_notes[i], 0))
                note_index = note_numbers.index(midi_notes[i])
                print("MIDI NoteOff:", note_names[note_index])
            except ValueError:
                pass

    #  Key combo for MIDI panic
    # . o o o o o .
    # o o o . o o o
    # . o o o o o .

    if (not keys[0].value and
            not keys[6].value
            and not keys[10].value
            and not keys[14].value
            and not keys[20].value):
        send_midi_panic()
        time.sleep(1)
```

Use the Modal MIDI Keyboard

To test the keyboard, plug it into your computer and launch this handy Chrome browser MIDI Monitor web app (https://adafru.it/C-3) to check that it is working.



Make Some Sound

MIDI note messages are fun to look at, but even better when the make some sound! Use a software synthesizer that accepts MIDI messages (pretty much all of them do!).

Here are some examples of free, open source synths for Linux, Windows, and mac os:

- Helm (https://adafru.it/C-a)

- VCV Rack (https://adafru.it/C-b)
- Pure Data (https://adafru.it/C-c)
- Ardour (https://adafru.it/C-d)

Launch your software synth and select the Pico CircuitPython keyboard as your MIDI source.

This video shows how root and mode selection work on startup. You can start over again at any time by pressing the reset button.

Root Note Selection

On startup, you can press each of the first 12 keys starting from the upper left corner of the keyboard to preview/select your scale root note.

Press the bottom right key to enter/commit the most recently previewed note.

Mode Selection

Once your root note is picked, the Modal MIDI keyboard goes into mode selection configuration. Press each of the keys on the top row of the keyboard to preview each mode:

- Major/Ionian
- Minor/Aeolian
- Dorian
- Phrygian
- Lydian
- Mixolydian
- Locrian

You'll hear each note of the selected mode play. Once you like your choice, press the lower right key to enter.

Now, you can start playing all three octaves!

How It Works

Libraries

First, you'll import libraries for `time`, `board`, `digitalio`, `usb_midi`, `adafruit_midi`, and the `adafruit_debouncer`.

Next, you set your `MIDI_CHANNEL` variable to whichever real-world MIDI channel you want to use. This can be anything from 1-16.

The `midi` object is created to send over USB.

```
import time
import board
from digitalio import DigitalInOut, Direction, Pull
import usb_midi
import adafruit_midi
from adafruit_midi.note_on import NoteOn
from adafruit_midi.note_off import NoteOff
from adafruit_debouncer import Debouncer

print("---Pico MIDI Modal Mech Keyboard---")

MIDI_CHANNEL = 1  # pick your MIDI channel here

midi = adafruit_midi.MIDI(midi_out=usb_midi.ports[1], out_channel=MIDI_CHANNEL-1)
```

MIDI Panic

The `send_midi_panic()` function can be used to send a **noteOff** command on all 128 MIDI notes, which is used in rare cases where a note or notes get "stuck" in the on state. You'll trigger this function with a special keyboard shortcut.

```
def send_midi_panic():
    print("All MIDI notes off")
    for x in range(128):
        midi.send(NoteOff(x, 0))
```

Key Setup

You'll create a list of the 21 GPIO `pins` that will be used on the Pico, and then set them all as digital input debouncer objects in a list named `keys[]`.

```
pins = (
    board.GP0,
    board.GP1,
    board.GP2,
    board.GP3,
    board.GP4,
    board.GP5,
    board.GP6,
    board.GP7,
    board.GP8,
    board.GP9,
    board.GP10,
    board.GP11,
    board.GP12,
    board.GP13,
    board.GP14,
    board.GP16,
    board.GP17,
    board.GP18,
    board.GP19,
    board.GP20,
    board.GP21,
)

keys = []
for pin in pins:
    tmp_pin = DigitalInOut(pin)
```

```
    tmp_pin.pull = Pull.UP
    keys.append(Debouncer(tmp_pin))
```

Note Lists

These variables are lists of MIDI note numbers and names, as well as state variables.

```
root_notes = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59)  # used during config
note_numbers = (48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
                60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71,
                72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83)
note_names = ("C2", "C#2", "D2", "D#2", "E2", "F2", "F#2", "G2", "G#2", "A2", "A#2",
"B2",
              "C3", "C#3", "D3", "D#3", "E3", "F3", "F#3", "G3", "G#3", "A3", "A#3",
"B3",
              "C4", "C#4", "D4", "D#4", "E4", "F4", "F#4", "G4", "G#4", "A4", "A#4",
"B4",)
scale_root = root_notes[0]  # default if nothing is picked
root_picked = False  # state of root selection
mode_picked = False  # state of mode selection
mode_choice = 0
```

User Config: Note Selection

You'll allow the user to select a root note using this section of code. It will wait until the user presses the bottom right key, a.k.a. `keys[20]`, until it move on.

```
print("Pick the root using top twelve keys, then press bottom right key to enter:")
print(". . . . . . .")
print(". . . . . o o")
print("o o o o o o .")

while not root_picked:
    for i in range(12):
        keys[i].update()
        if keys[i].fell:
            scale_root = root_notes[i]
            midi.send(NoteOn(root_notes[i], 120))
            print("Root is", note_names[i])
        if keys[i].rose:
            midi.send(NoteOff(root_notes[i], 0))
    keys[20].update()

    if keys[20].rose:
        root_picked = True
        print("Root picked.\n")
```

Mode Lists

You'll create lists of the interval formulas of the seven modes, which are relative to the root note. The `modes[]` list is a dictionary of these.

```
major = ( 0, 2, 4, 5, 7, 9, 11 )
minor = ( 0, 2, 3, 5, 7, 8, 10 )
dorian = ( 0, 2, 3, 5, 7, 9, 10 )
phrygian = ( 0, 1, 3, 5, 7, 8, 10 )
lydian = (0 , 2, 4, 6, 7, 9, 11 )
mixolydian = ( 0, 2, 4, 5, 7, 9, 10)
locrian = ( 0, 1, 3, 5, 6, 8, 10)
```

```
modes = []
modes.append(major)
modes.append(minor)
modes.append(dorian)
modes.append(phrygian)
modes.append(lydian)
modes.append(mixolydian)
modes.append(locrian)

mode_names = ("Major/Ionian",
              "Minor/Aeolian",
              "Dorian",
              "Phrygian",
              "Lydian",
              "Mixolydian",
              "Locrian")

intervals = list(mixolydian)  # intervals for Mixolydian by default
```

User Config: Mode Selection

The user now picks among the seven modes, with a preview played for each. The bottom right key confirms the selected mode and then moves on.

```
print("Pick the mode with top seven keys, then press bottom right key to enter:")
print(". . . . . . .")
print("o o o o o o o")
print("o o o o o o .")

while not mode_picked:
    for i in range(7):
        keys[i].update()
        if keys[i].fell:
            mode_choice = i
            print(mode_names[mode_choice], "mode")
            for j in range(7):
                intervals[j] = modes[i][j]
            # play the scale
            for k in range(7):
                midi.send(NoteOn(scale_root+intervals[k], 120))
                note_index = note_numbers.index(scale_root+intervals[k])
                print(note_names[note_index])
                time.sleep(0.15)
                midi.send(NoteOff(scale_root+intervals[k], 0))
                time.sleep(0.15)
            midi.send(NoteOn(scale_root+12, 120))
            note_index = note_numbers.index(scale_root+12)
            print(note_names[note_index], "\n")
            time.sleep(0.15)
            midi.send(NoteOff(scale_root+12, 0))
            time.sleep(0.15)

    keys[20].update()
    if keys[20].rose:
        print(mode_names[mode_choice], "mode picked.\n")
        mode_picked = True
```

Main Loop

In the main loop of the program the keys are checked for updates with the debouncer. If a key is pressed (fell) the associated **noteOn** message is sent, and when it it released (rose) the **noteOff** message is sent.

```
for i in range(num_keys):
        keys[i].update()
        if keys[i].fell:
            try:
                midi.send(NoteOn(midi_notes[i], 120))
                note_index = note_numbers.index(midi_notes[i])
                print("MIDI NoteOn:", note_names[note_index])
            except ValueError:  # deals w six key limit
                pass

        if keys[i].rose:
            try:
                midi.send(NoteOff(midi_notes[i], 0))
                note_index = note_numbers.index(midi_notes[i])
                print("MIDI NoteOff:", note_names[note_index])
            except ValueError:
                pass
```

Panic Key Combo

If the five key pattern of the outer corners and the center key are pressed, the `send_midi_panic()` function runs, turning off all notes.

```
#  Key combo for MIDI panic
    # . o o o o o .
    # o o o . o o o
    # . o o o o o .

    if (not keys[0].value and
            not keys[6].value
            and not keys[10].value
            and not keys[14].value
            and not keys[20].value):
        send_midi_panic()
        time.sleep(1)
```