DIY Pico Mechanical Keyboard with Fritzing and CircuitPython

Created by John Park

https://learn.adafruit.com/diy-pico-mechanical-keyboard-with-fritzing-circuitpython

Last updated on 2021-11-16 03:20:33 PM EST

Table of Contents

Overview 5

• Parts 6
• PCB 6

Installing CircuitPython 10

• CircuitPython Quickstart 10
• Flash Resetting UF2 12

Installing Libraries 12

• Installing the Adafruit HID Library 12

Installing the Mu Editor 13

• Download and Install Mu 13
• Starting Up Mu 14
• Using Mu 14

Code the Pico Keyboard 15

• Text Editor 15
• CircuitPython Code 16
• Testing 17
• HID Keyboard Basics 18
• USB Keyboards versus CircuitPython HID Keyboards 19
• Keyboard Press/Release 19
• Media Control 20
• Key Assignements 21

Make a Custom PCB with Fritzing 21

• Custom Parts 23
• Import Parts 23
• Breadboard View 24
• Add Pico Part to Breadboard View 24
• Add Keyswitch 25
• Wire the Switch to the Pico 26
• Route the Schematic View 26
• Part Layout in PCB View 27
• Pads and Holes 28
• Draw a Trace 28
• Bottom Layer Via a Via 29
• Bottom Trace 30
• View Management and Layers 31
• Trace Adjustment 32

Making a More Complex Board 33

• Routing a Board with More Switches 33
• Schematic Tips 33
• Symbolic Connections 35
• GPIO Connections 36
• PCB Rerouting 37
• Keyswitch Spacing 37
• Copper Ground Fill 37
• Ground Plane Prep 38
• Set Ground Fill Seed 39

• Ground Fill 40
• Pin Traces 42
• Silkscreen Label 43
• Reset Button 44

Ordering PCBs 46

• DRC Check 47
• Paper Test 48
• Gerber -- Not Just For Babies 51
• Export Gerber Files 52
• Zip It Good 52
• Upload and Verify 53

21-Key Pico Keyboard 55

• Net Labels 56
• PCB Layout 56

Pico Keyboard Assembly 58

• Solder the Pico 59
• Test the Joints 60
• Reset Button 60
• Switch Plate 60
• Screw Prep 64
• Seat the Switches 65
• Solder the Switches 67
• Testing 67
• Final Assembly 67
• Laser Case Bottom 68
• Laser Case Top 70
• Optional Creamy Center 71
• 3D Printed Case 72
• 3D Base Attachment 73
• 3D Case Top 75

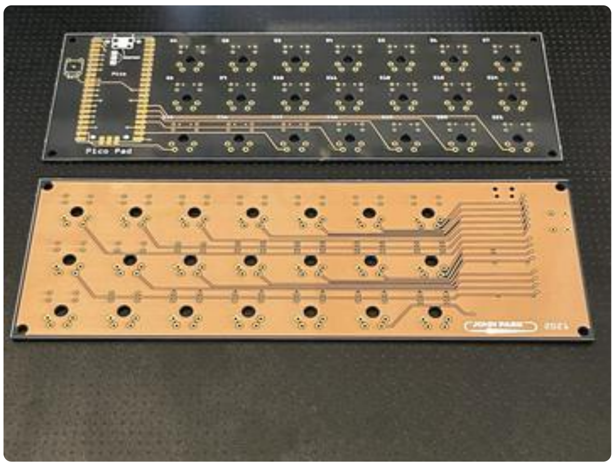
Overview

Build your own custom mechanical keyboard that runs CircuitPython on the RaspberryPi Pico RP2040!

With lots of pins, the RaspberryPi RP2040 Pico (https://adafru.it/QOF) makes for a great brain of a mechanical keyboard/macro pad -- no scan matrix required. Up to 26 keys can be used with direct GPIO pins. You'll learn how to make your own PCB design in Fritzing to send off for fabrication. A 3D printed or laser cut case finishes it off in style.

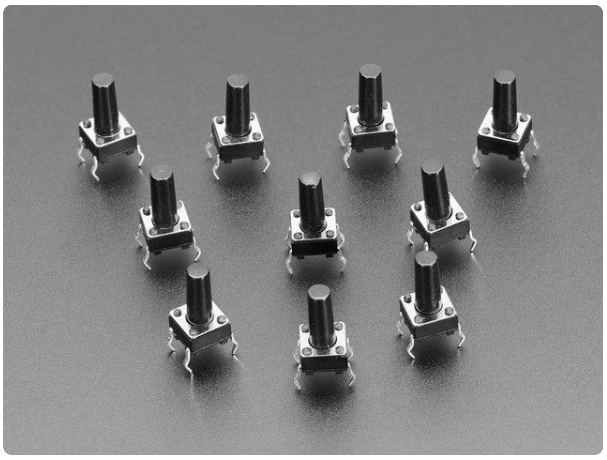
Once you've built your own custom keyboard from scratch, run over to Reddit r/mechanicalkeyboards and show it off!

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.



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

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

Cherry MX Compatible Keyswitches

The classic mechanical keyswitch. They come in three styles -- linear, tactile, and clicky, and in different strengths.

You can get them many places, including here (https://adafru.it/QNc) from Digi-Key, or here (https://adafru.it/QNd).

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

MX Keycaps

Pick your style! Blank XDA or DSA profile keycaps look clean and stylish on this build, as it is an ortholinear (non-staggered) layout. Make sure all the keys are 1u (square) size.

I got some here (https://adafru.it/QUe) and here (https://adafru.it/QNe). There's a whole world of keycaps out there, just make sure they fit the Cherry MX stems. These 1u Extras in pink and purple (https://adafru.it/QNf) look pretty great...

Brass M2.5 Standoffs 16mm tall - Black Plated - Pack of 2
Finally we get our very own super sweet sixteen... 16 millimeters that is! Our Black Plated M/F Brass 16mm Standoffs are engineered specifically...
https://www.adafruit.com/product/2337

Black Nylon Machine Screw and Stand-off Set – M3 Thread
Totaling 420 pieces, this M3 Screw Set is a must-have for your workstation. You'll have enough screws, nuts, and hex standoffs to fuel...
https://www.adafruit.com/product/4685

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

M2.5 x 16mm screws x4

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

Little Rubber Bumper Feet - Pack of 4
Keep your electronics from going barefoot, give them little rubber feet! These small sticky bumpers are our favorite accessory for any electronic kit or device. They are sticky, but...
https://www.adafruit.com/product/550

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

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

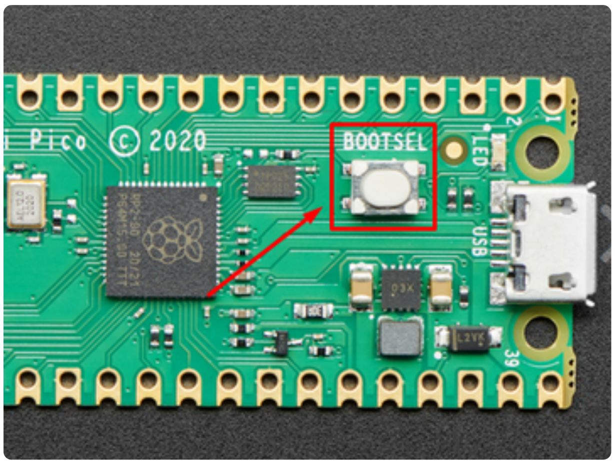
Follow this step-by-step to quickly get CircuitPython working on your board.

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.

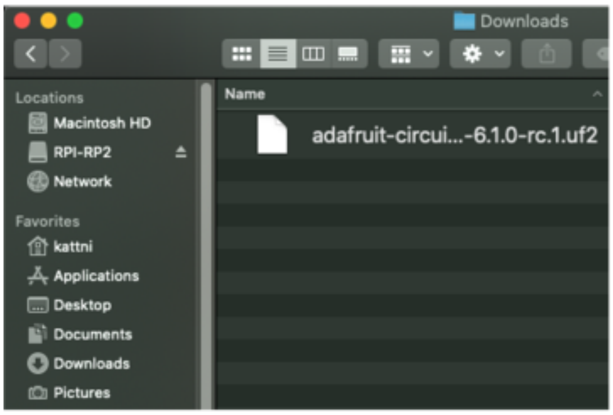
Download and save it to your desktop (or wherever is handy).



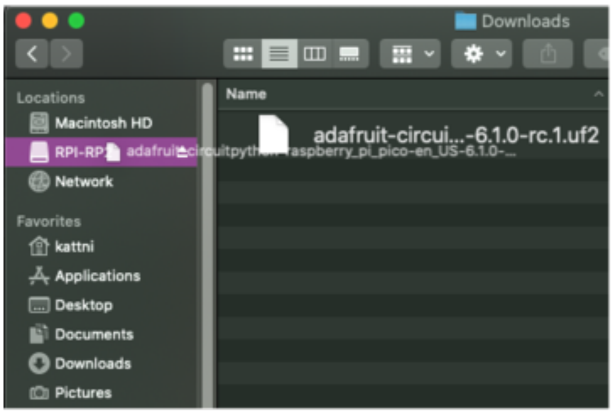
Start with your Pico unplugged from USB. Hold down the BOOTSEL button, and while continuing to hold it (don't let go!), plug the Pico into USB. Continue to hold the BOOTSEL button until the RPI-RP2 drive appears!

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

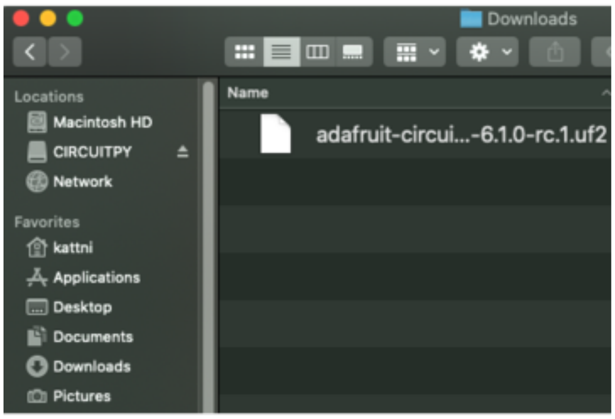
A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.



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/QAJ

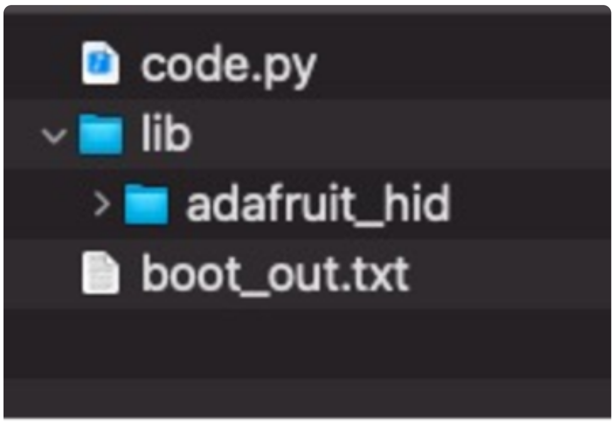
Installing Libraries

Alongside the core CircuitPython libraries (which are baked into CircuitPython), you'll also add the Adafruit HID Library to add keyboard features.

Installing the Adafruit HID Library

Download the library bundle (https://adafru.it/ENC) here.

Copy the adafruit_hid folder from the bundle to the lib folder on your CIRCUITPY driv e.



Before continuing make sure your board's lib folder has the adafruit_hid library folder copied over.

Installing the Mu Editor

Mu is a simple code editor that works with the Adafruit CircuitPython boards. It's written in Python and works on Windows, MacOS, Linux and Raspberry Pi. The serial console is built right in so you get immediate feedback from your board's serial output!

Mu is our recommended editor - please use it (unless you are an experienced coder with a favorite editor already!).

Download and Install Mu

Code with Mu: a simple Python editor for beginner programmers.
Download
Start Here

```
print("Hello from Mu!")
```

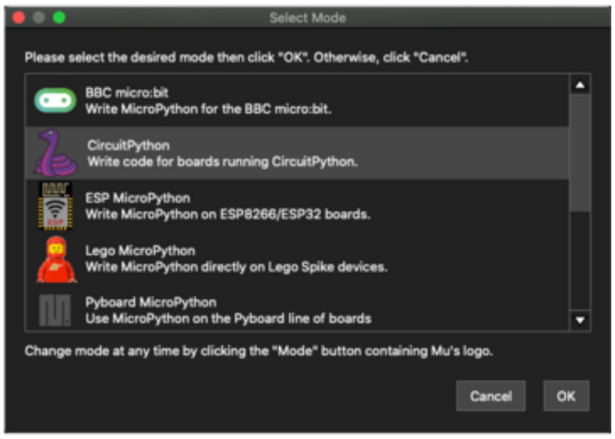
Download Mu from https://codewith.mu (https://adafru.it/Be6).

Click the Download link for downloads and installation instructions.

Click Start Here to find a wealth of other information, including extensive tutorials and and how-to's.

Windows users: due to the nature of MSI installers, please remove old versions of Mu before installing the latest version.

Starting Up Mu



Please select the desired mode then click "OK". Otherwise, click "Cancel".

BBC micro:bit
Write MicroPython for the BBC micro:bit.

CircuitPython
Write code for boards running CircuitPython.

ESP MicroPython
Write MicroPython on ESP8266/ESP32 boards.

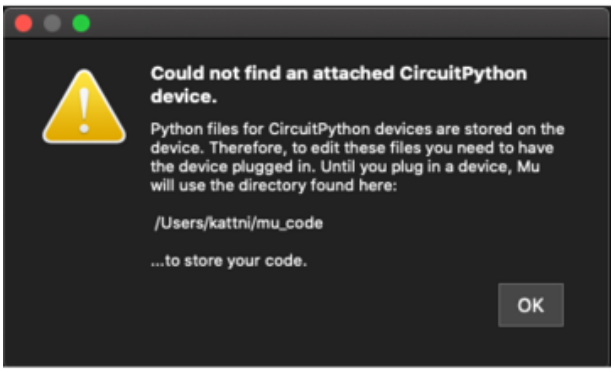
Lego MicroPython
Write MicroPython directly on Lego Spike devices.

Pyboard MicroPython
Use MicroPython on the Pyboard line of boards

Change mode at any time by clicking the "Mode" button containing Mu's logo.

The first time you start Mu, you will be prompted to select your 'mode' - you can always change your mind later. For now please select CircuitPython!

The current mode is displayed in the lower right corner of the window, next to the "gear" icon. If the mode says "Microbit" or something else, click the Mode button in the upper left, and then choose "CircuitPython" in the dialog box that appears.



Could not find an attached CircuitPython device.

Python files for CircuitPython devices are stored on the device. Therefore, to edit these files you need to have the device plugged in. Until you plug in a device, Mu will use the directory found here:

/Users/kattni/mu_code

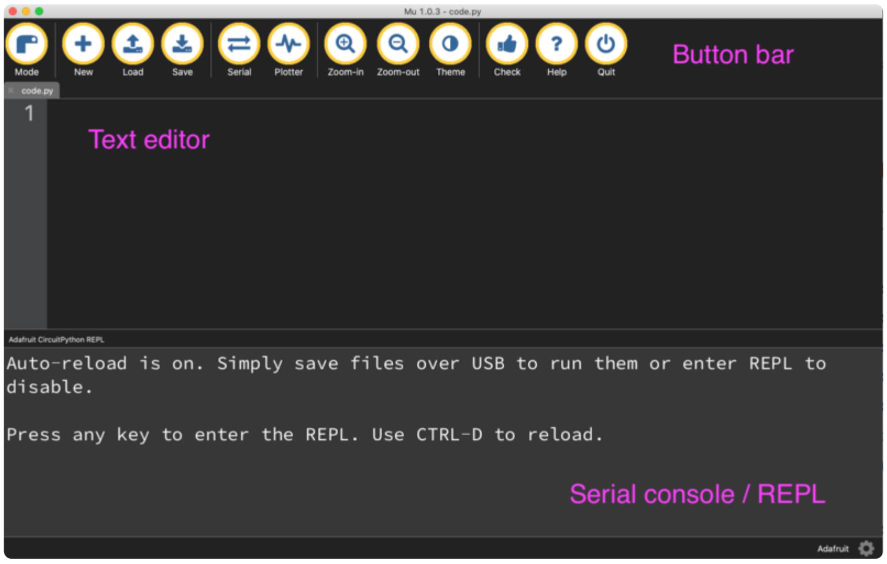
...to store your code.

Mu attempts to auto-detect your board on startup, so if you do not have a CircuitPython board plugged in with a CIRCUITPY drive available, Mu will inform you where it will store any code you save until you plug in a board.

To avoid this warning, plug in a board and ensure that the CIRCUITPY drive is mounted before starting Mu.

Using Mu

You can now explore Mu! The three main sections of the window are labeled below; the button bar, the text editor, and the serial console / REPL.



```
Auto-reload is on. Simply save files over USB to run them or enter REPL to
disable.

Press any key to enter the REPL. Use CTRL-D to reload.
```

Now you're ready to code! Let's keep going...

Code the Pico Keyboard

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.

CircuitPython Code

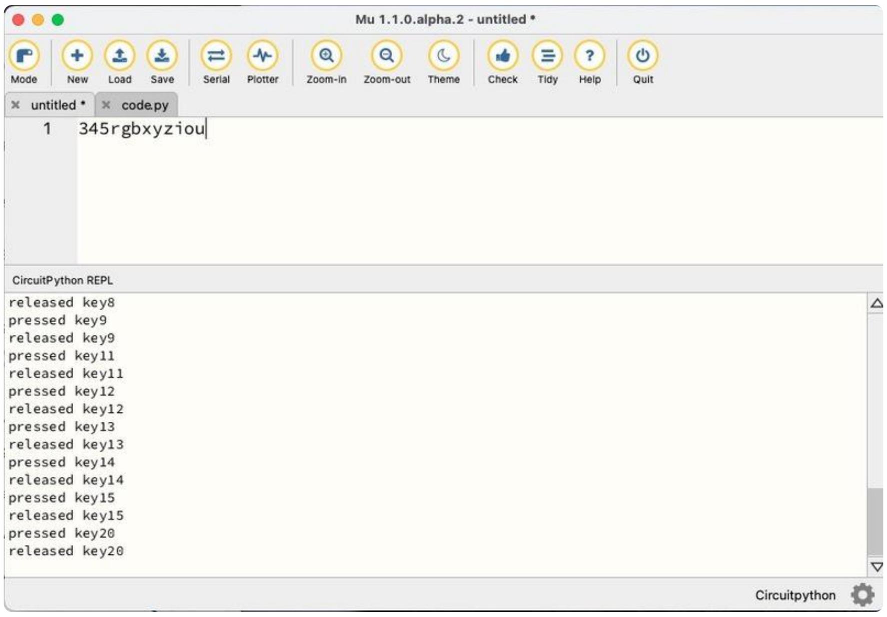
Copy the code below and paste it into Mu. Then, save it to your Pico as code.py.

```
# SPDX-FileCopyrightText: 2021 John Park for Adafruit Industries
# SPDX-License-Identifier: MIT
# RaspberryPi Pico RP2040 Mechanical Keyboard

import time
import board
from digitalio import DigitalInOut, Direction, Pull
import usb_hid
from adafruit_hid.keyboard import Keyboard
from adafruit_hid.keycode import Keycode
from adafruit_hid.consumer_control import ConsumerControl
from adafruit_hid.consumer_control_code import ConsumerControlCode

print("---Pico Pad Keyboard---")

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

kbd = Keyboard(usb_hid.devices)
cc = ConsumerControl(usb_hid.devices)

# 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,
]

MEDIA = 1
KEY = 2

keymap = {
    (0): (KEY, (Keycode.GUI, Keycode.C)),
    (1): (KEY, (Keycode.GUI, Keycode.V)),
    (2): (KEY, [Keycode.THREE]),
    (3): (KEY, [Keycode.FOUR]),
    (4): (KEY, [Keycode.FIVE]),
    (5): (MEDIA, ConsumerControlCode.VOLUME_DECREMENT),
    (6): (MEDIA, ConsumerControlCode.VOLUME_INCREMENT),

    (7): (KEY, [Keycode.R]),
    (8): (KEY, [Keycode.G]),
    (9): (KEY, [Keycode.B]),
    (10): (KEY, [Keycode.UP_ARROW]),
    (11): (KEY, [Keycode.X]),  # plus key
    (12): (KEY, [Keycode.Y]),
    (13): (KEY, [Keycode.Z]),

    (14): (KEY, [Keycode.I]),
    (15): (KEY, [Keycode.O]),
    (16): (KEY, [Keycode.LEFT_ARROW]),
    (17): (KEY, [Keycode.DOWN_ARROW]),
    (18): (KEY, [Keycode.RIGHT_ARROW]),
    (19): (KEY, [Keycode.ALT]),
    (20): (KEY, [Keycode.U]),

}
switches = [0, 1, 2, 3, 4, 5, 6,
            7, 8, 9, 10, 11, 12, 13,
            14, 15, 16, 17, 18, 19, 20, 21]

for i in range(21):
    switches[i] = DigitalInOut(pins[i])
    switches[i].direction = Direction.INPUT
    switches[i].pull = Pull.UP

switch_state = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

while True:
    for button in range(21):
        if switch_state[button] == 0:
            if not switches[button].value:
                try:
                    if keymap[button][0] == KEY:
                        kbd.press(*keymap[button][1])
                    else:
                        cc.send(keymap[button][1])
                except ValueError:  # deals w six key limit
                    pass
                switch_state[button] = 1

        if switch_state[button] == 1:
            if switches[button].value:
                try:
                    if keymap[button][0] == KEY:
                        kbd.release(*keymap[button][1])

                except ValueError:
                    pass
                switch_state[button] = 0

    time.sleep(0.01)  # debounce
```

Testing

Even before we've attached the mech keyswitches and PCB, you can test the code by shorting each of the GPIO pins used to ground. This is the same as pressing a key. Just be careful not to short power to ground!

Open the serial monitor for your board in Mu, or use a text editor to watch the keys get typed each time you short one of the GPIO pins.



```
345rgbxyziou
```

```
released key8
pressed key9
released key9
pressed key11
released key11
pressed key12
released key12
pressed key13
released key13
pressed key14
released key14
pressed key15
released key15
pressed key20
released key20
```

USB keyboards and mice show up on your computer as 'HID' devices, which stands for 'Human Interface Device'

HID Keyboard Basics

This guide page (https://adafru.it/DaD) has a great intro to CircuitPython HID Keyboard.

For even more details, check out the documentation at https://circuitpython.readthedocs.io/projects/hid/en/latest/ (https://adafru.it/B-7) which includes all of the keycodes and media codes you can use.

By importing the adafruit_hid library into the program, you can make calls to send keyboard keys and media keys.

USB Keyboards versus CircuitPython HID Keyboards

Certain keys you might want to emulate from your standard existing keyboard, such as multi-media (volume, play/pause, etc.) keys, are not actually regular keyboard keys. They are consumer control keys. All keys on a standard USB keyboard are not necessarily sending regular keycode values, but instead may be sending consumer control values on a separate consumer control device. In CircuitPython this is the difference between sending `Keycode` values via a `Keyboard`, and sending `ConsumerControlCode` values via a `ConsumerControl`. Therefore, if, in CircuitPython, you intend to send, for example, a "mute" command in your HID example, you should use `ConsumerControl` instead of an equivalent keyboard key based on your standard existing keyboard.

Consumer control keys, such as multi-media keys, do not require "focus" to function. For example, sending a `VOLUME_DECREMENT` consumer control command will decrease the volume regardless of which window currently has keyboard focus.

Keyboard Press/Release

Using the HID library in CircuitPtyhon, you can send this command to "type" the letter 'a':

`kbd.press(Keycode.A)`

`kbd.release(Keycode.A)`

This would send a lowercase 'a' to the computer just as if you had typed it yourself. To send a capital 'A', we'd add the shift key to the command like this:

`kbd.press(Keycode.SHIFT, Keycode.A)`

`kbd.release(Keycode.SHIFT, Keycode.A)`

This is pretty cool, since it means we can layer on lots of keys all at the same time, just like you do on your physical keyboard when using keyboard shortcuts!

So, if there's some keyboard shortcut you want to use (or create for yourself in something like Quicksilver or AutoKeys) that is command+option+ctrl+a the CircuitPython code would look like this:

`kbd.press(Keycode.GUI, Keycode.ALT, Keycode.CONTROL, Keycode.A)`

`kbd.release(Keycode.GUI, Keycode.ALT, Keycode.CONTROL, Keycode.A)`

The adafruit_hid library allows for operating system specific names such as 'Keycode.COMMAND' on macOS which is 'Keycode.WINDOWS' on Windows. Or, you can use the generic 'Keycode.GUI' on any operating system. Same goes for 'ALT/OPTION'

Media Control

There is a second command to use to adjust volume, play/pause, skip tracks, and so on with media such as songs and videos. These are often represented on a physical keyboard as icons silkscreened onto the rightmost function keys.

In USB HID speak, these are known as "Consumer Control codes". To play or pause a track, use this command:

`cc.send(ConsumerControlCode.PLAY_PAUSE)`

```
keymap = {
    (0): (KEY, (Keycode.GUI, Keycode.C)),
    (1): (KEY, (Keycode.GUI, Keycode.V)),
    (2): (KEY, [Keycode.THREE]),
    (3): (KEY, [Keycode.FOUR]),
    (4): (KEY, [Keycode.FIVE]),
    (5): (MEDIA, ConsumerControlCode.VOLUME_DECREMENT),
    (6): (MEDIA, ConsumerControlCode.VOLUME_INCREMENT),

    (7): (KEY, [Keycode.R]),
    (8): (KEY, [Keycode.G]),
    (9): (KEY, [Keycode.B]),
    (10): (KEY, [Keycode.UP_ARROW]),
    (11): (KEY, [Keycode.X]),  # plus key
    (12): (KEY, [Keycode.Y]),
    (13): (KEY, [Keycode.Z]),

    (14): (KEY, [Keycode.I]),
    (15): (KEY, [Keycode.O]),
    (16): (KEY, [Keycode.LEFT_ARROW]),
    (17): (KEY, [Keycode.DOWN_ARROW]),
    (18): (KEY, [Keycode.RIGHT_ARROW]),
    (19): (KEY, [Keycode.ALT]),
    (20): (KEY, [Keycode.U]),

}
```

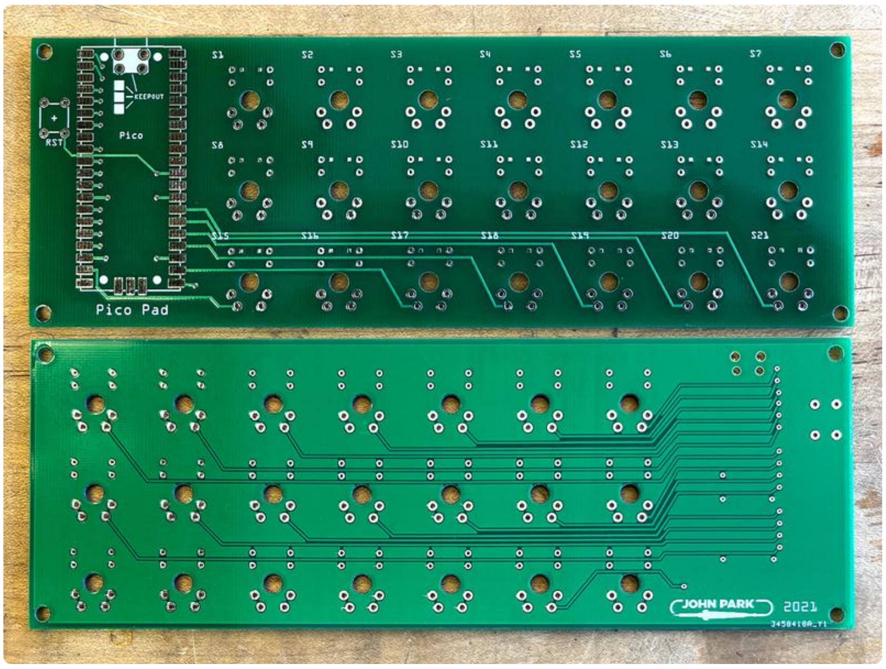
Key Assignements

With that in mind, you can now fully customize the function of each key by editing this section of the code:

Note, in order to use a single stroke keycode, you'll surround it in [brackets], while a multi-stroke keycode will have its own (parentheses) as shown here:

```
(1): (KEY, (Keycode.GUI, Keycode.V)),
(2): (KEY, [Keycode.THREE]),
```

Make a Custom PCB with Fritzing



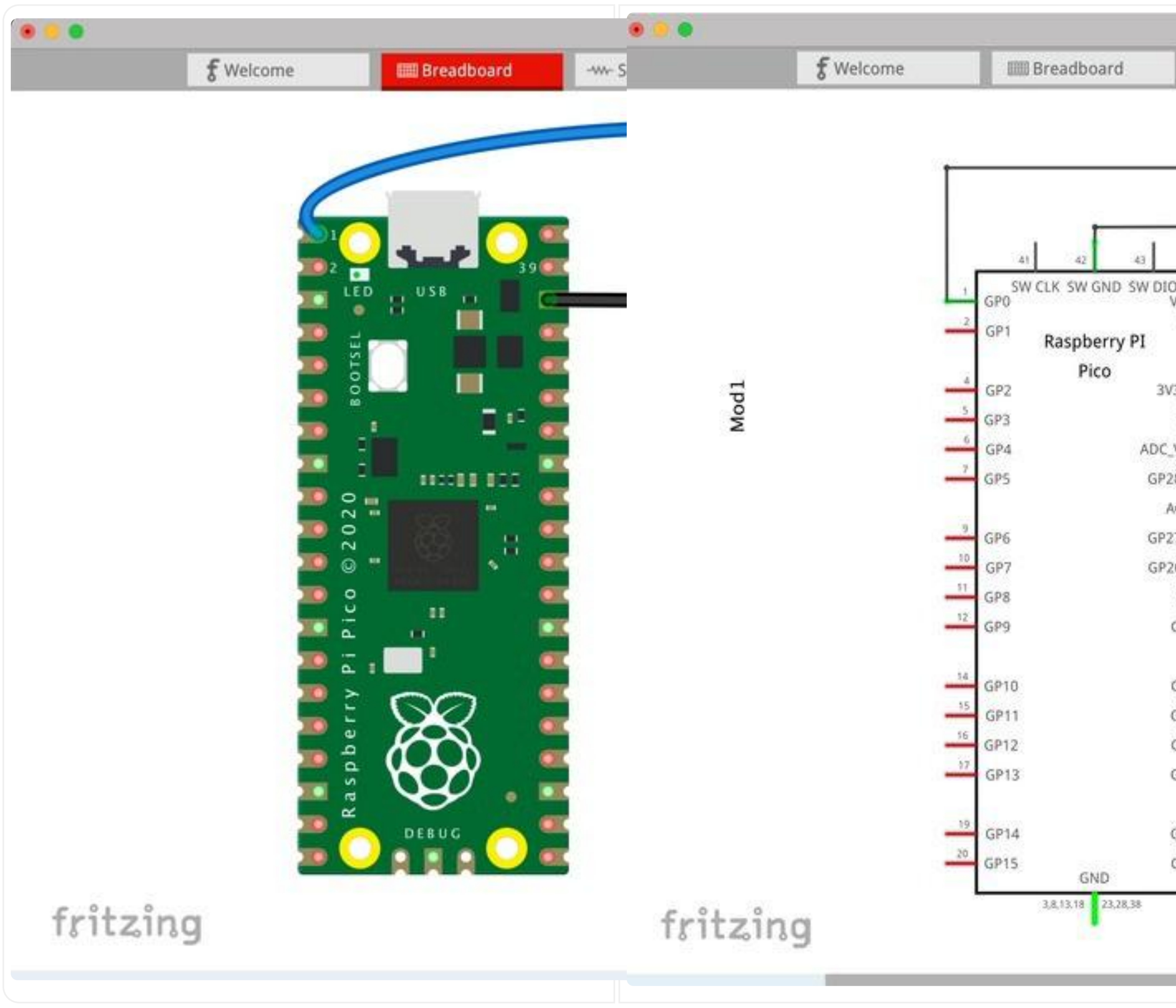
Printed circuit boards (PCBs) are incredibly neat, compact, and sturdy ways to connect components when compared to a breadboard full of wires and tenuous connections held in place by friction and optimism, or the chaos of a hand wired protoboard. Once you know how to design PCBs, you can make them real via chemical etching, milling, or sending the design files to a PCB production house, such as OSH Park, JLCPCB, or others who will make professional boards for you quickly and inexpensively.

Design software such as Eagle or Kicad are capable of creating very sophisticated designs, but have fairly steep learning curves. Fritzing is very popular among makers for creating breadboard diagrams, but also has a easy-to-use PCB design capabilities that are often overlooked. In this guide, you'll use Fritzing to make your own Pico mechanical keyboard PCB!

You'll find some terrific Fritzing Tricks from Radomi 'deʃhipu' Dopieralski here (https://adafru.it/QUf). Big thanks to Peter 'vanepp' Van Epp for creating the excellent RaspberriPi Pico Fritzing parts, more info here (https://adafru.it/QUA).

Fritzing is comprised of three main views of the same design -- Breadboard view, Schematic view, and PCB (printed circuit board) view.

You'll work in these three views to first create a prototype representation of the circuit in Breadboard view, then optimize the circuit in Schematic view, and finally to design the electrical traces and physical layout of the board in PCB view.



Moving among the three views in Fritzing is encouraged! Make adjustments in any view whenever you like, just remember that they all represent the same design, so changes in one view will likely impact the others.

Custom Parts

There are two custom Fritzing parts you'll use while designing your keyboard PCB -- a Cherry MX keyswitch and the SMD (surface mount) version of the RP2040 Pico. Read more about this part here (https://adafru.it/QNB).

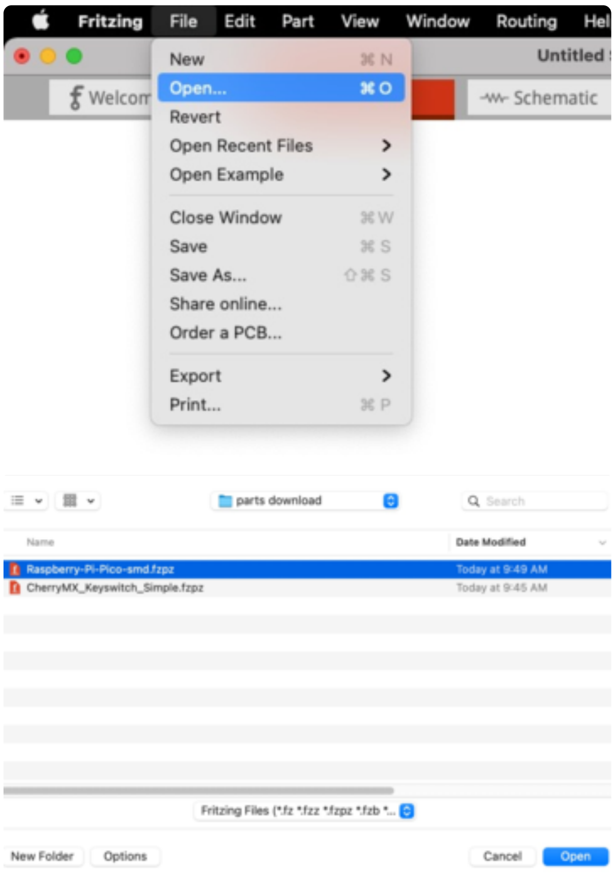
Download the .zip compressed files linked below, and then uncompress them. Once uncompressed the file names will end in `.fzpz`

CherryMX_Keyswitch_Simple.zip

https://adafru.it/QNC

Raspberry-Pi-Pico-smd.zip

https://adafru.it/QND



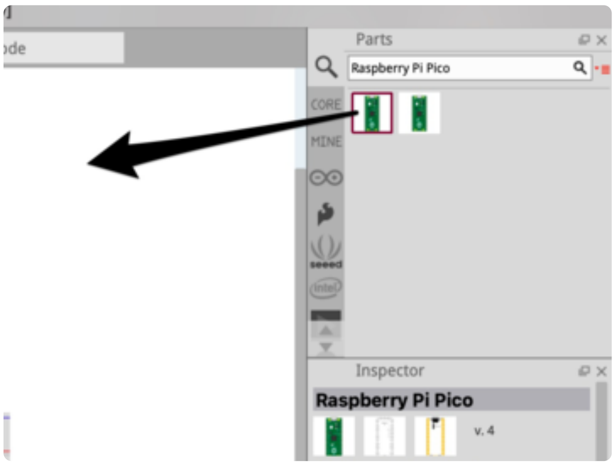
Import Parts

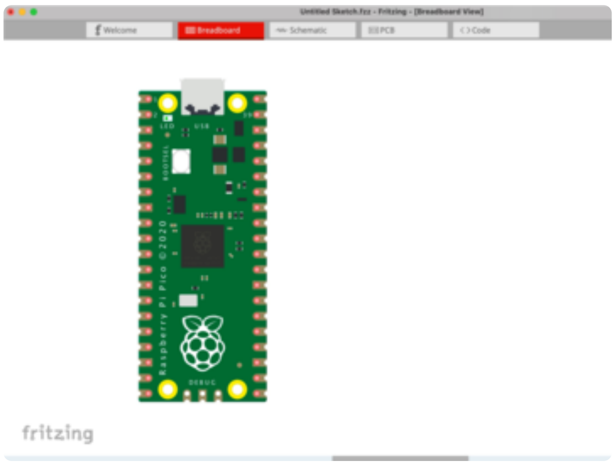
In Fritzing, click File > Open... and navigate to the Raspberry-Pi-Pico-smd.fzpz part, then click the Open button.

This will add the part to your Parts bin.

Breadboard View

The breadboard view in Fritzing resembles real-world physical parts wired together directly or on a breadboard.





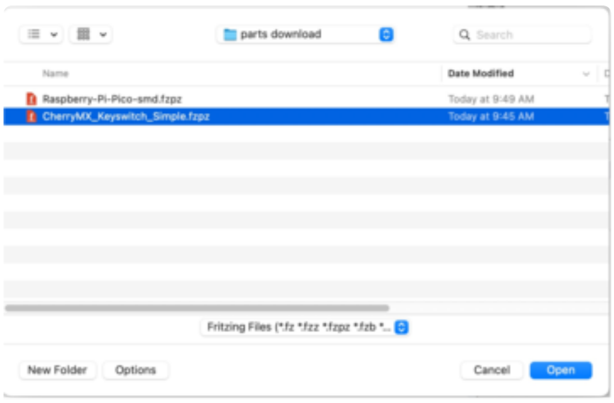
Add Pico Part to Breadboard View

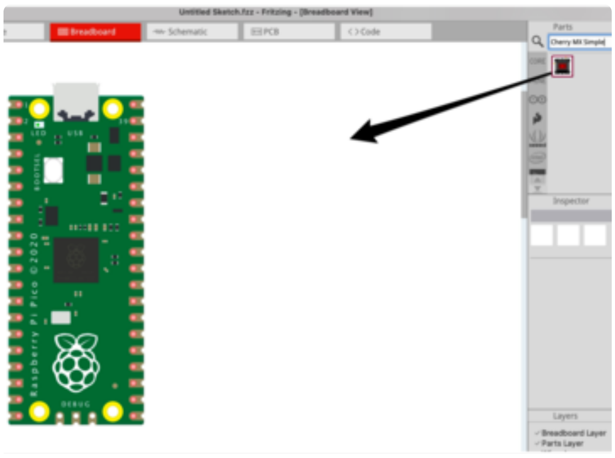
In the Part bin, the Pico part should appear (if not, you can use the search feature to find it). Drag the Pico into your breadboard view.

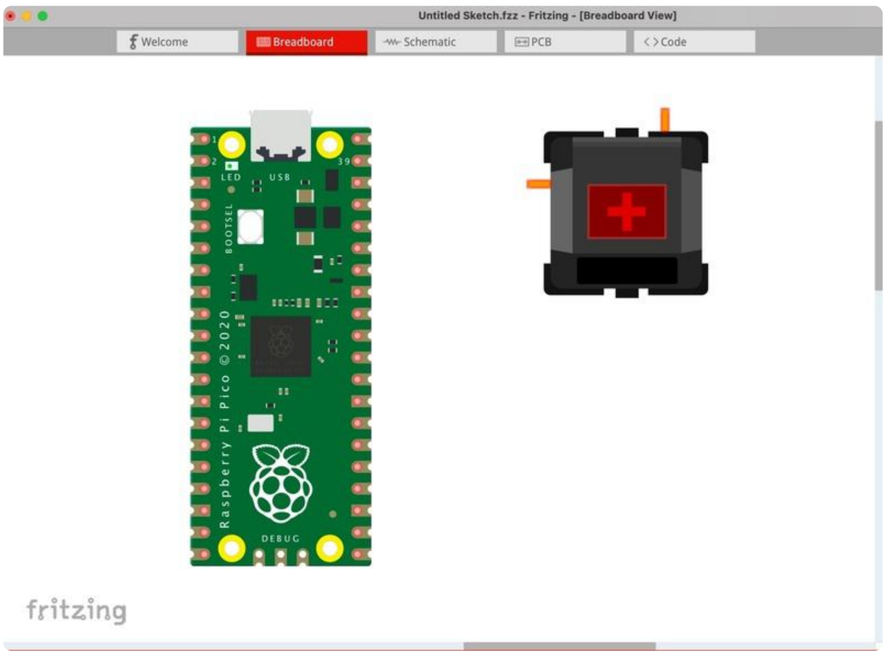
Note, there are two different Pico parts available, one is designed for SMD (surface mount) use using the castellated pads soldered to the top of the PCB, the other is THT (through hole), designed for use with header pins. This design calls for the SMD part.

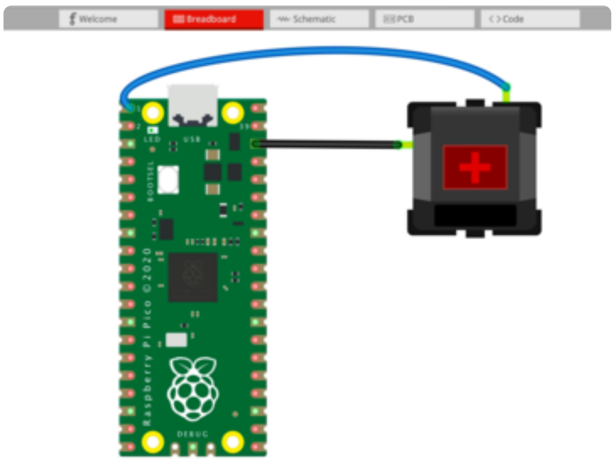
Add Keyswitch

Repeat the previous steps to add the Cherry MX Keyswitch Simple part.


parts download
Search
Name
Date Modified
Raspberry-Pi-Pico-smd.fzpz
Today at 9:49 AM
CherryMX_Keyswitch_Simple.fzpz
Today at 9:45 AM
Fritzing Files (*.fz *.fzz *.fzpz *.fzb *...
New Folder
Options
Cancel
Open







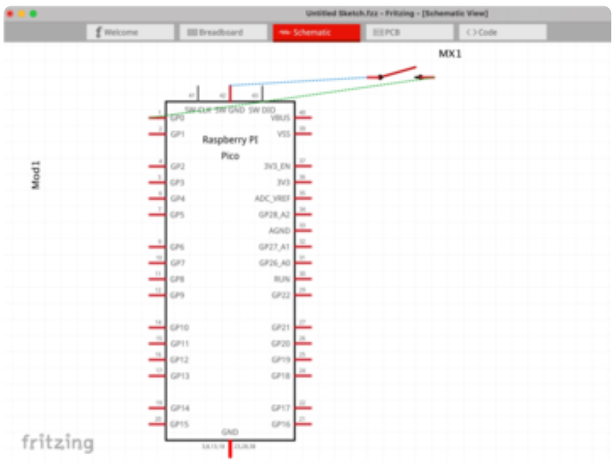
Wire the Switch to the Pico

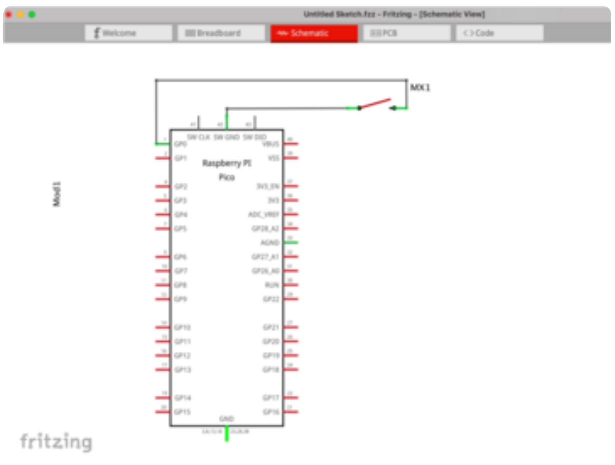
Click-drag a wire from the Pico's GND (pin 38) to connect to the keyswitch pin 1 leg. You can right-mouse click the wire to change its color to black.

Connect the Pico's GPIO 0 pad (pin 1) to the pin 2 leg of the keyswitch, then color the wire blue.

Remember so save your work regularly!

Next, you'll have a look at the Schematic view representation of your circuit. You won't need to do too much here now, but as your circuit becomes more complex, you'll use this view to optimizing things in ways that will make life easier in the PCB view.





Route the Schematic View

Click the Schematic tab at the top of Fritzing. Here you'll see a neat, organized version of your parts made with standard electronic component symbols.

The connections you made in Breadboard view appear here as dotted lines between the Pico and the switch. Sometimes these connections are just as you'd like, other times you'll do some rerouting here for clarity and optimization.

For now, you can click on each of the dotted connections to create solid wires, and include bend points by clicking and dragging to make a clear, organized diagram.

You'll do more work in this view later once you've added more switches, but for now you can move on to PCB view.

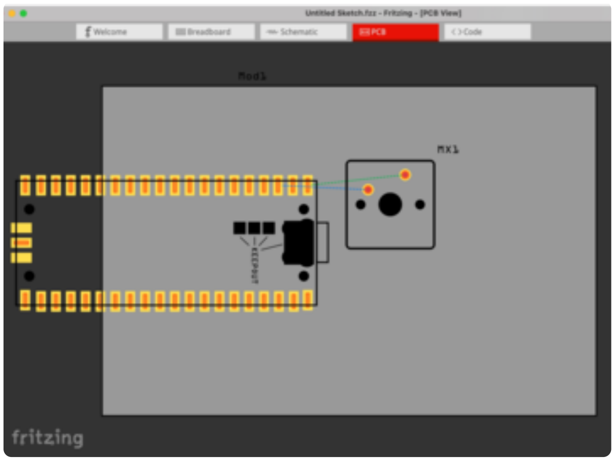
Now that you have the parts wired in Breadboard view, and routed in Schematic view, it's time to orient them and lay out their electrical traces in PCB view.

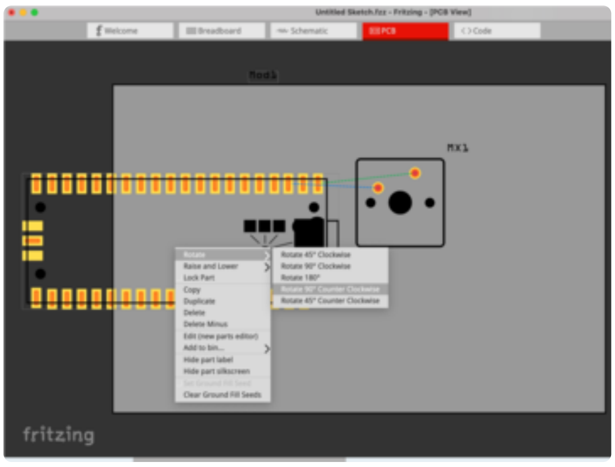
Part Layout in PCB View

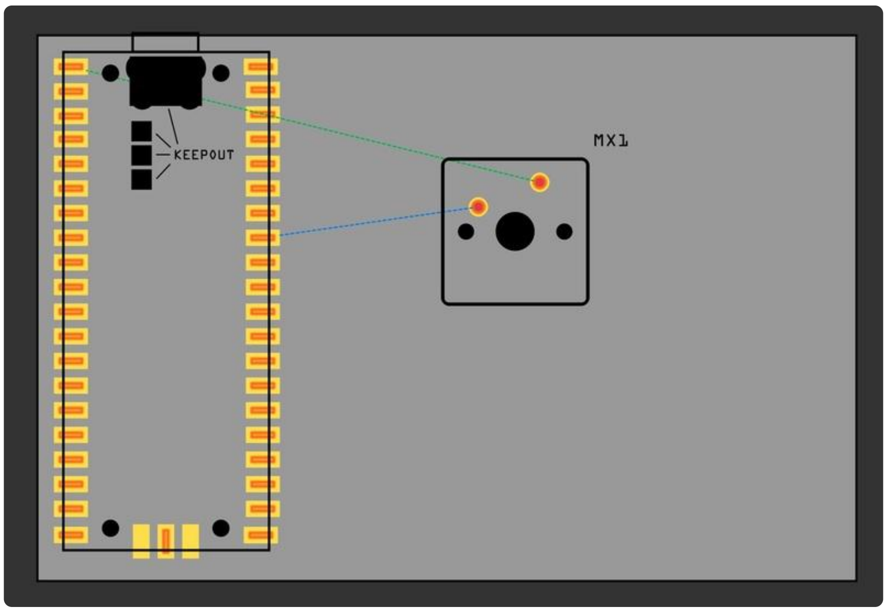
Switch to PCB view. You'll see the Pico and the keyswitch parts as well as a gray rectangle for the PCB board itself.

In this view you can re-position any part by click-dragging it.

You will rotate the Pico by right-mouse clicking on it and choosing the Rotate > Rotate 90° Counter Clockwise menu item.





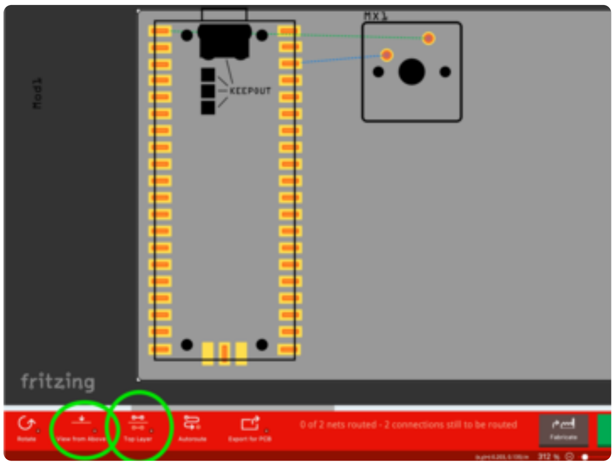


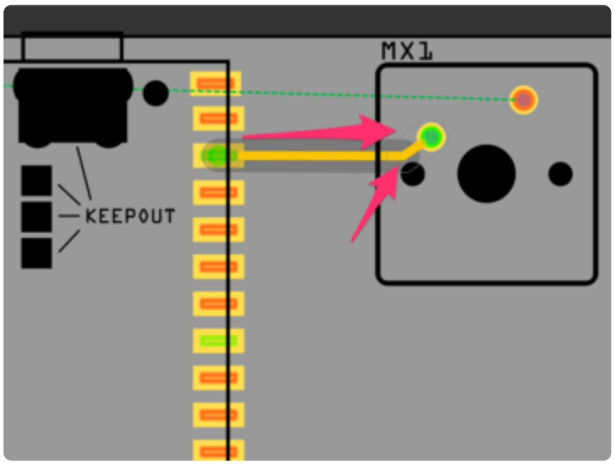
Pads and Holes

The part footprints include all of the necessary copper pads to mount the Pico to the top of the PCB as well as the drilled, plated (copper-lined) holes and circular pads to mount the keyswitches to the top of the board by soldering them to the bottom of the board.

Just as in the Schematic view, any existing connections between parts that haven't been routed are shown as dotted lines. These are called "air wires" or "ratsnest" lines. If you were to fabricate this PCB right now it wouldn't function because the air wires haven't been converted to actual copper traces.

The PCB view is a bit more complex than the other views because the board has both a top and a bottom side which must be considered. You can view it from either side (as well as both at the same time using a sort of "x-ray" view), and you can place your copper traces on either side.

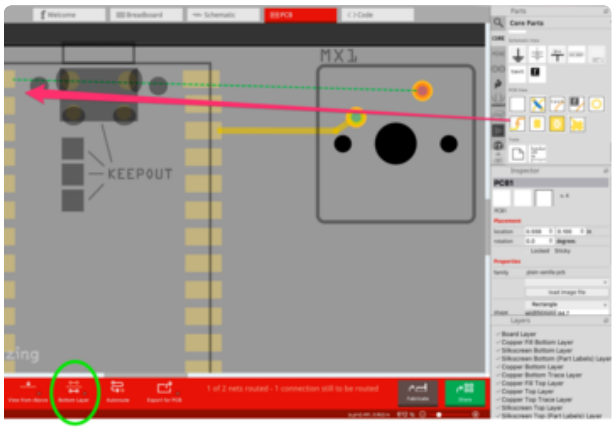


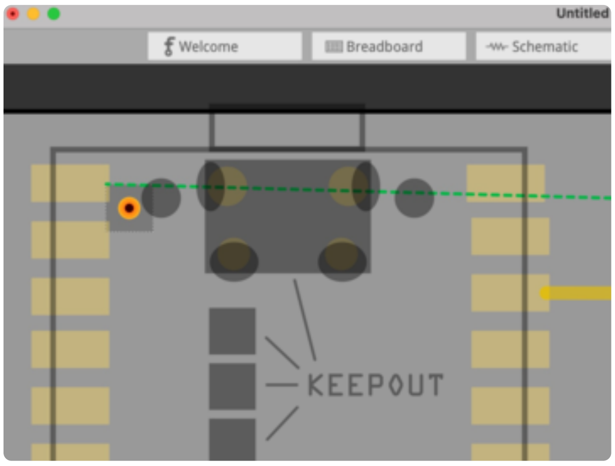


Draw a Trace

With the board view set at "View from Above" and the copper layer set to "Top Layer", click and drag the air wire that runs from the Pico GND to the keyswitch pin 1. This will create a copper trace on the top layer of the board connecting these parts.

If you click drag on the trace you can add and position a bend point as shown here.



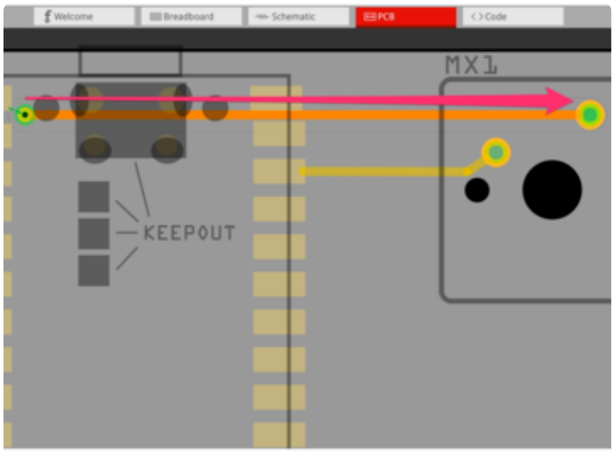


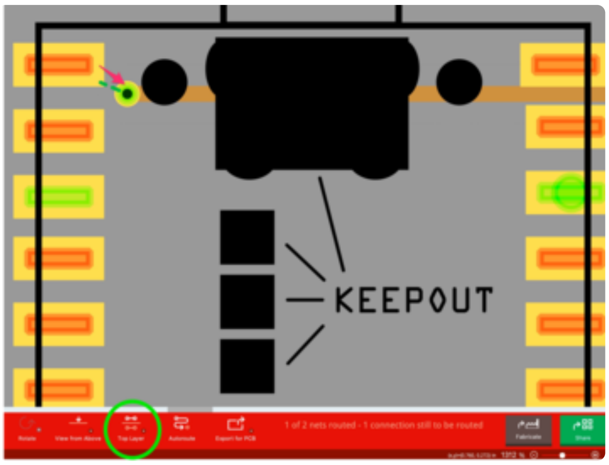
Bottom Layer Via a Via

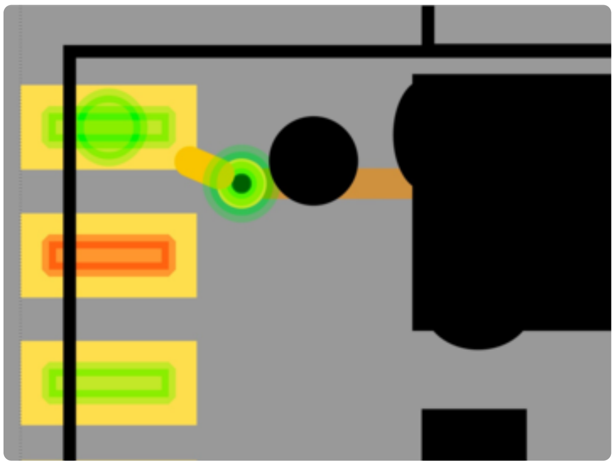
As you can imagine, placing all of your traces on just the top side of the board will become difficult when you have a lot of components. Traces can't cross one another without creating problems for your circuit. Running some connections on the bottom of the board makes life much easier!

In order to run a connection from one of the Pico pads at the top of the board through to the bottom of the board where you can then create a trace to connect to the keyswitch pin 2, you'll create a small plated connection hole called a via. You can move between layers via a via!

Switch to Bottom Layer -- vias can be placed on either layer since they pass between them, but we'll be continuing this trace on the bottom in a moment. In the Core part bin, look for the via icon in the PCB view section and then drag a via on onto the PCB next to Pico pin 1 as shown.





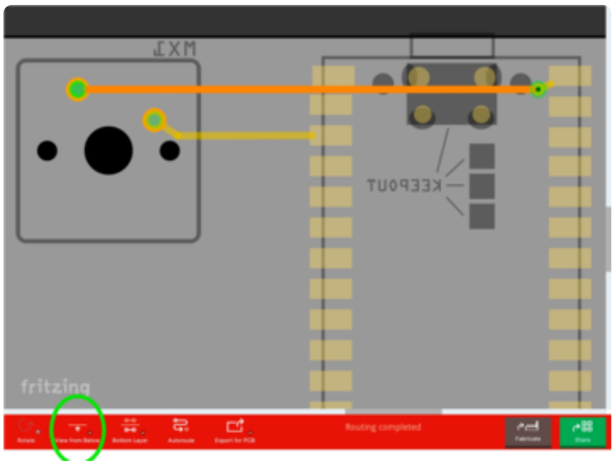


Bottom Trace

Drag a trace from the via to the keyswitch pin 2. Notice how the copper traces on the bottom layer are orange, while the top layer traces are yellow.

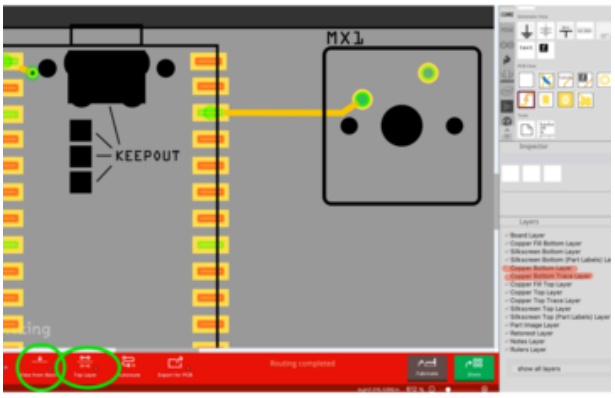
Remember that you're looking at a sort of x-ray view of the board's bottom. Since the Pico is on the top of the board and uses SMD pads, there are no worries about this new trace touching any of the pads it appears to be crossing beneath.

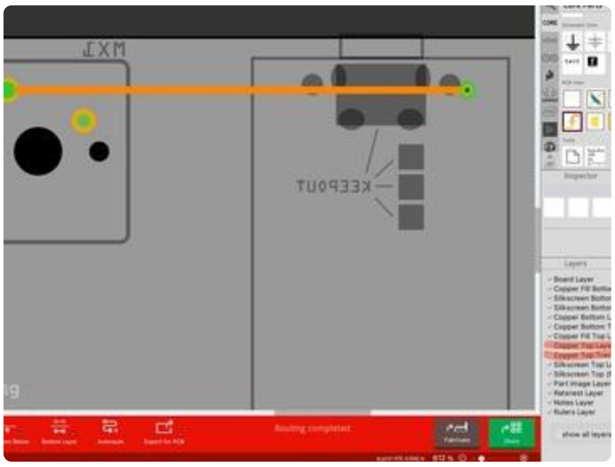
Switch back to the Top Layer and make a small trace from Pico pad 1 to the via.



View Management and Layers

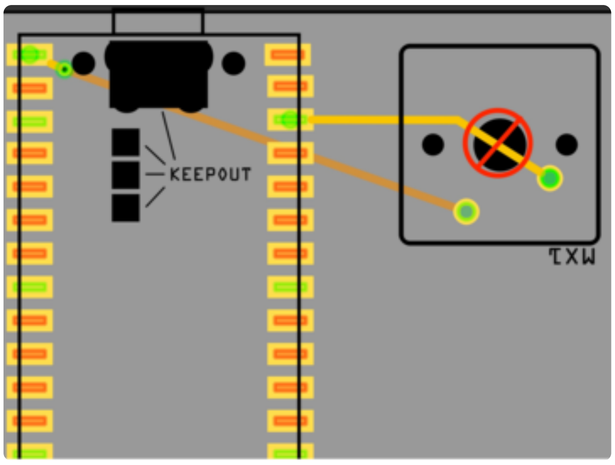
For visual clarity, you can switch between top and bottom views of the board, as well as use the Layers panel to turn on and off the various layer of the PCB, including top and bottom copper, top and bottom silkscreen (the stuff printed in ink on the board), and more.

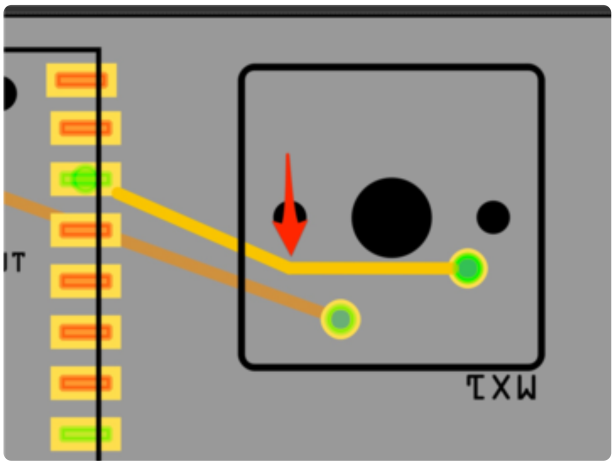


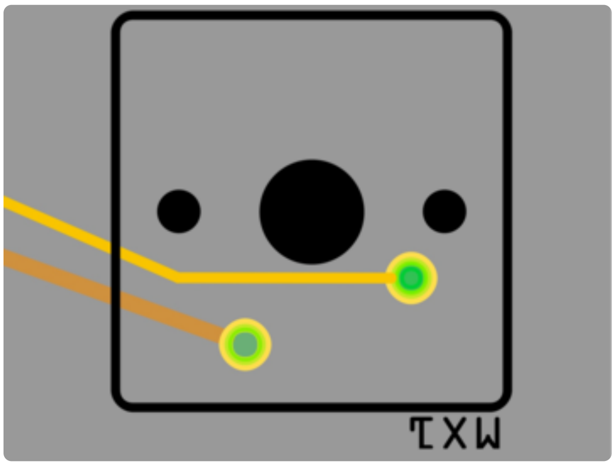


Parts can be moved and rotated even after you've placed copper traces, however this may lead to some issue that require fixes.

Pick the keyswitch and rotate it 180° to create such a situation.







Trace Adjustment

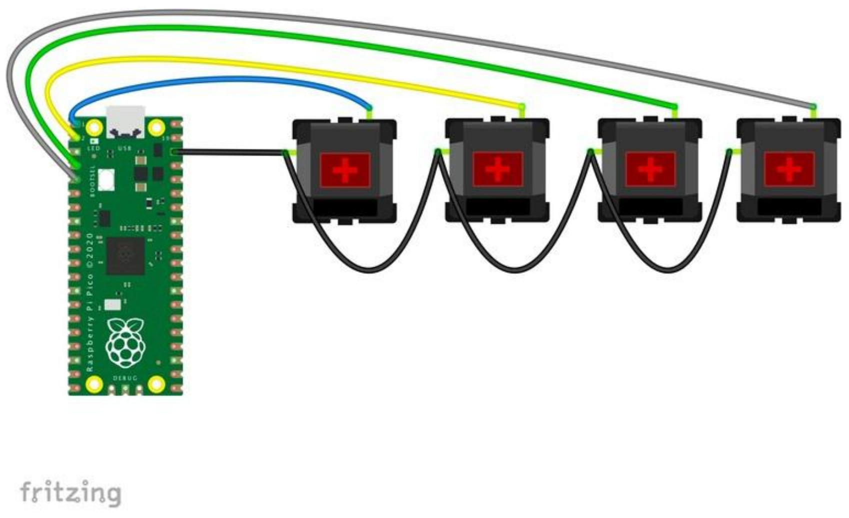
Oh no! Here you'll see that after rotating the part, the GND trace is now running right through the drilled mounting hole. (Note, the red warning sign was added here for effect, Fritzing does not warn you about this type of issue.)

Fix this issue by dragging the bendpoint down until the trace clears the hole.

If the trace still looks a bit close for comfort, you can change the trace width in the Inspector Properties panel from the default of 24 mil down to 16 mil or even 12 mil. Thinner traces can be difficult to etch or mill at home, but aren't a problem for PCB fab houses.

On the next page, more switches will be added and you'll learn some tricks for dealing with boards of greater complexity.

Making a More Complex Board



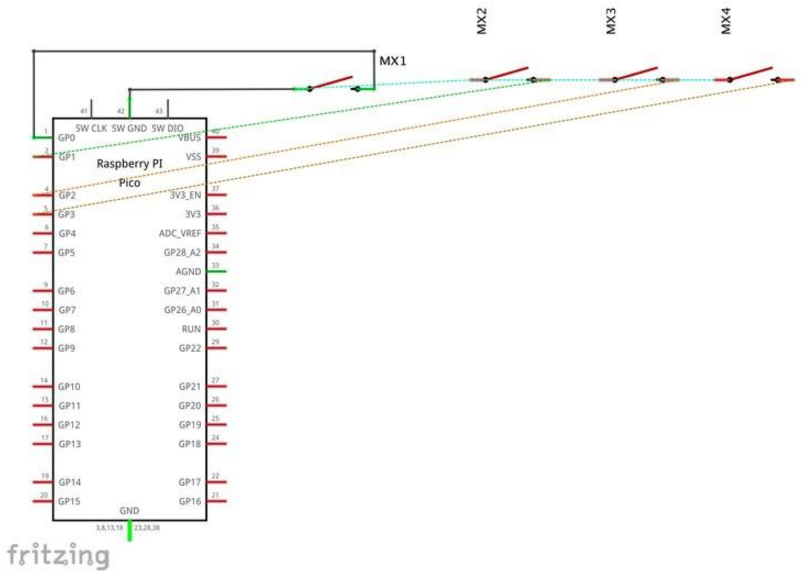
Routing a Board with More Switches

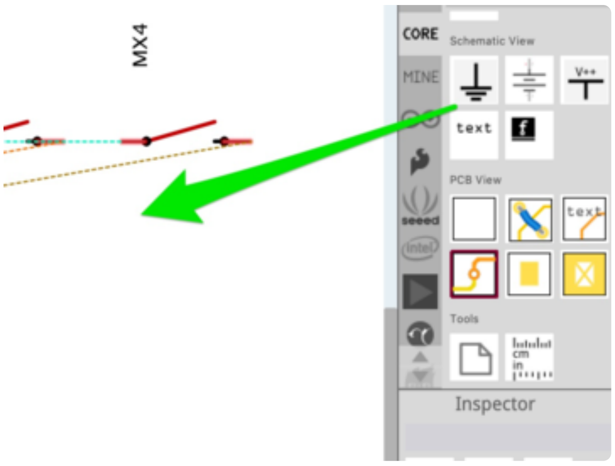
You'll likely want a few more switches on your Pico keyboard. Let's go for four switches right now and try out some optimizations in Schematic and PCB views to make things more efficient.

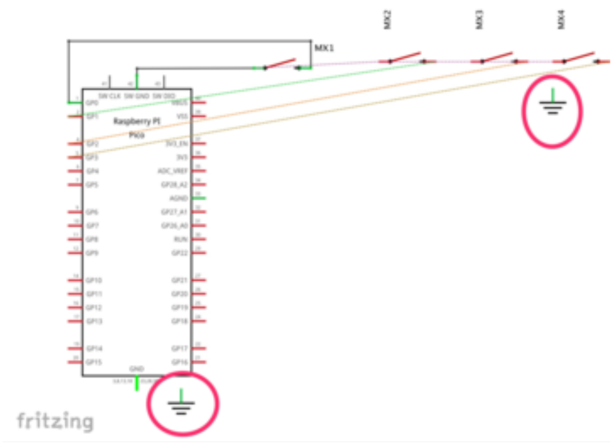
To begin, add three more keyswitches and wire them to GPIO 2, 3, & 4 of the Pico. Run the ground wires in series.

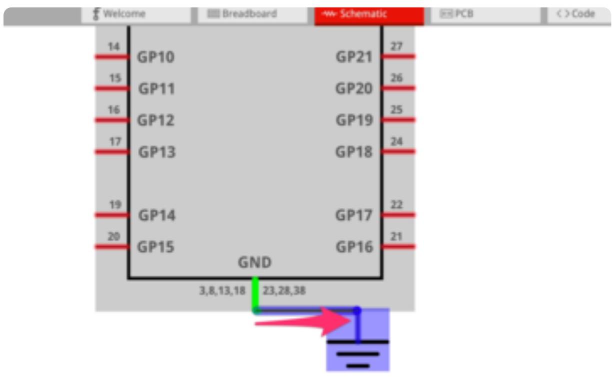
Schematic Tips

Switching to Schematic view we can see the circuit now has some airwires to route.


MX1
MX2
MX3
MX4
Raspberry PI
Pico
SW CLK
SW GND
SW DIO
GP0
GP1
GP2
GP3
GP4
GP5
GP6
GP7
GP8
GP9
GP10
GP11
GP12
GP13
GP14
GP15
VBUS
VSS
3V3_EN
3V3
ADC_VREF
GP28_A2
AGND
GP27_A1
GP26_A0
RUN
GP22
GP21
GP20
GP19
GP18
GP17
GP16
GND
fritzing







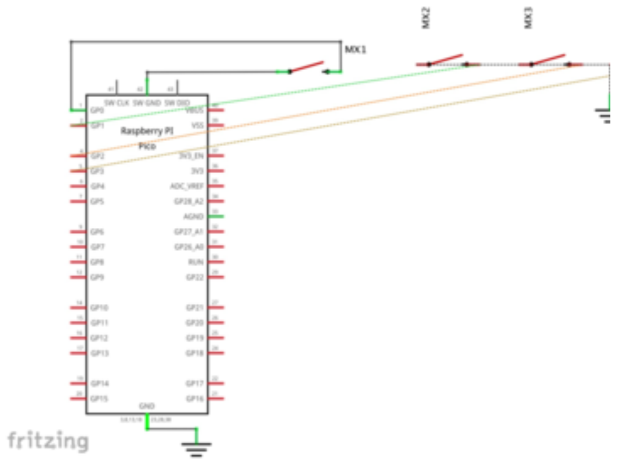
Symbolic Connections

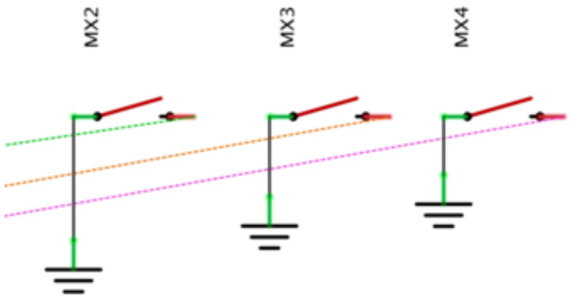
Running all of those ground connections on the switches directly to the Pico (or to each other) can become a bit messy and hard to read. Instead, you'll use the GND symbol. These represent a network of ground connections that are symbolically connected to each other.

In the Core parts bin, find the ground symbol and drag two of them into the Schematic view, one near keyswitch 4 and one next to a GND connection on the Pico.

Route a wire from the Pico GND connection to the GND symbol, then route a wire from the switch 4 pin 1 to the other GND symbol.

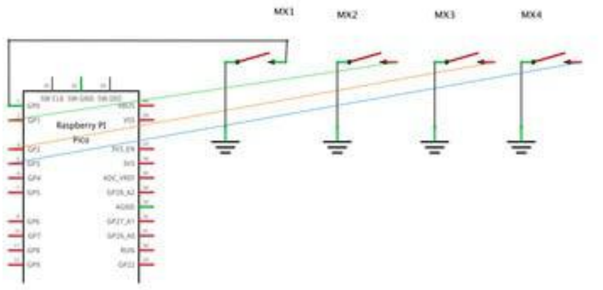
Bring in three more GND symbols and connect each switch to its own. They all now share the common GND network without a mess of wires!

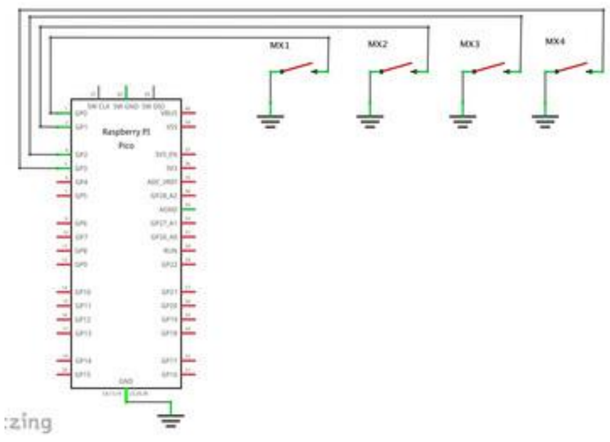




GPIO Connections

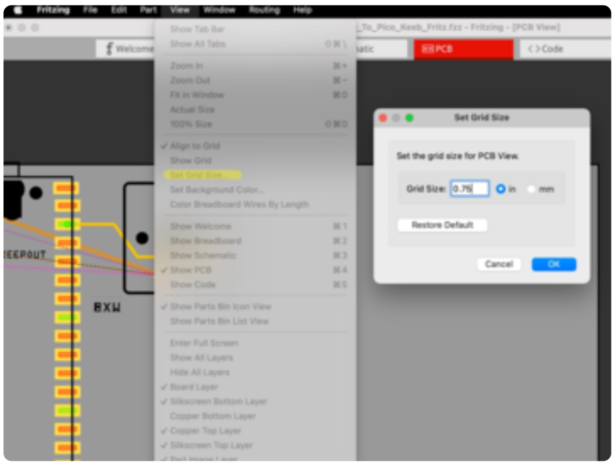
You can now wire the other GPIO to switch connections. Later we'll look at using symbolic net labels for these as well, but for now a direction connection is fine.





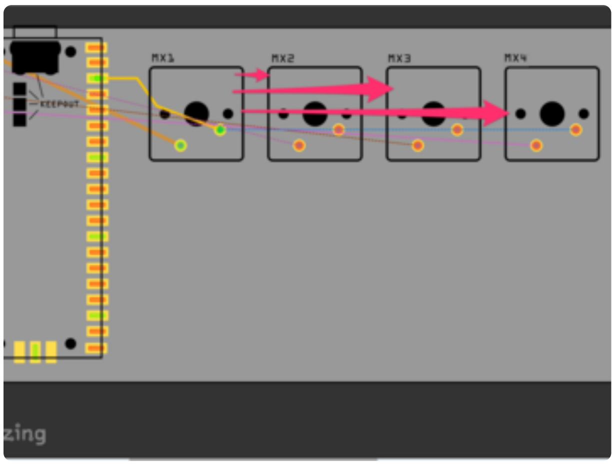
PCB Rerouting

Now that you've added more components you'll switch to PCB view to rework the board.



Set the grid size for PCB View.

Grid Size: 0.75 [✓ in] mm



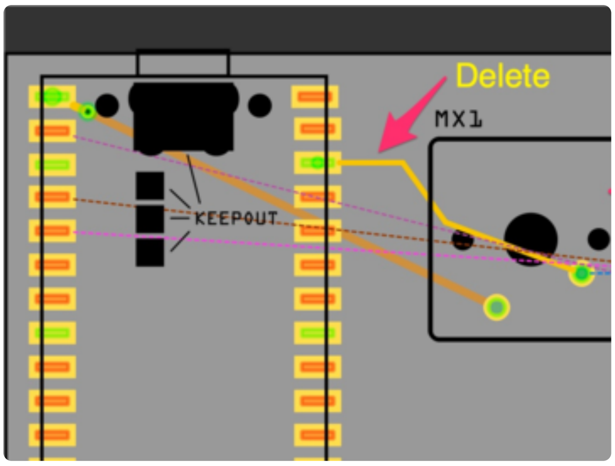
Keyswitch Spacing

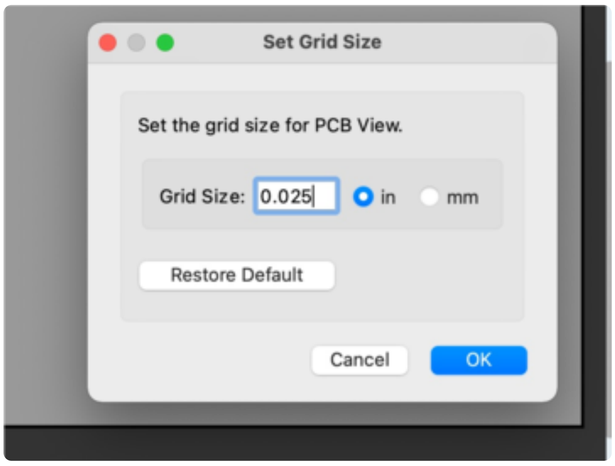
A typical spacing between keyswitches is 0.75". Set the Fritzing grid spacing to this value by going to View > Set Grid Size... and filling in the Grid Size field.

Now, pick and grab each switch to align them in a row -- they will snap to the grid, giving you perfect spacing.

Copper Ground Fill

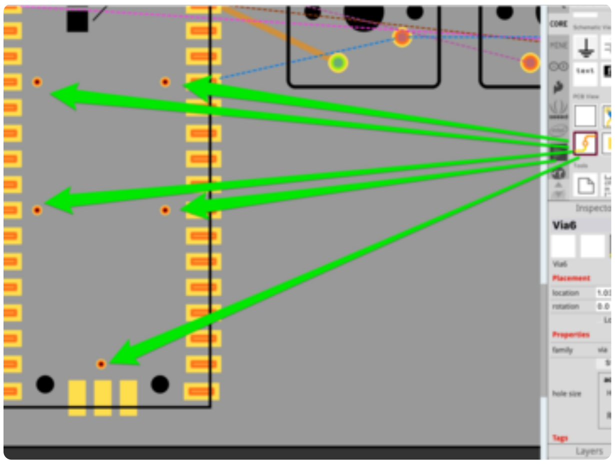
Instead of running individual traces to the ground pins on each switch, you'll created a copper fill on the bottom layer of the board. This is a large plane of copper that is connected to the Pico GND and all of the component grounds. This greatly simplifies making all of those ground connections, since no individual traces will need to be drawn and routed.

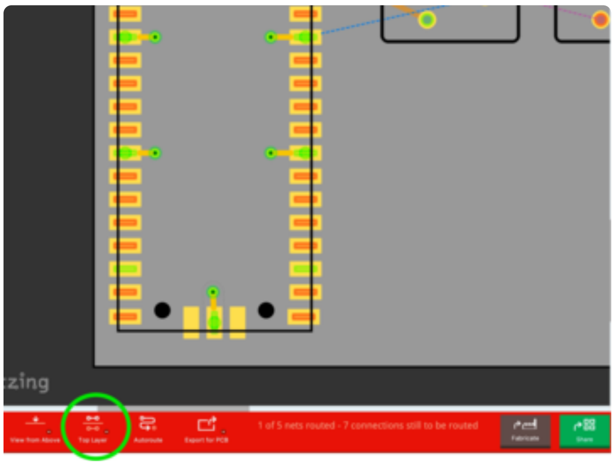



Set Grid Size

Set the grid size for PCB View.

Grid Size: 0.025 [✓ in] mm





Ground Plane Prep

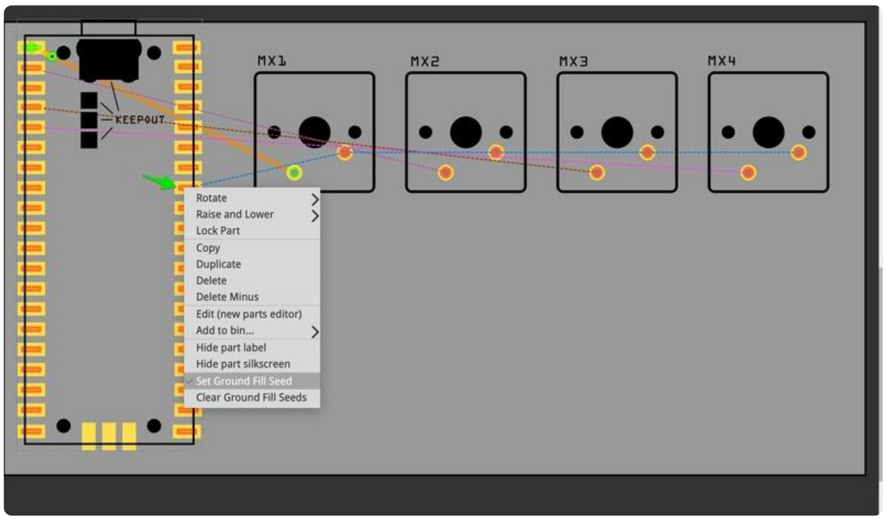
First, deleted the existing ground trace.

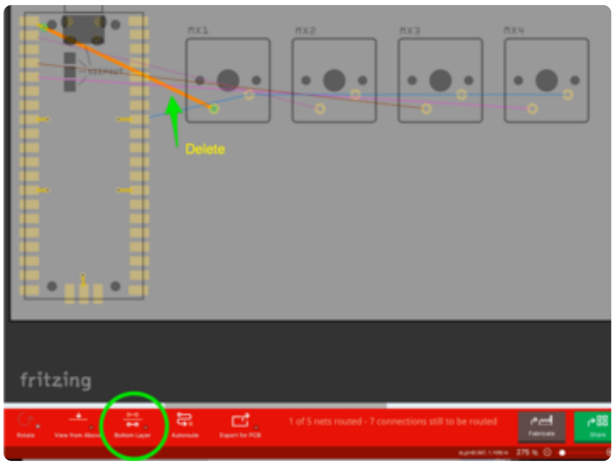
You'll create vias to run the Pico's GND pads to the bottom layer. Switch the grid spacing to 0.025", then drag vias next to a few of the Pico's GND pads.

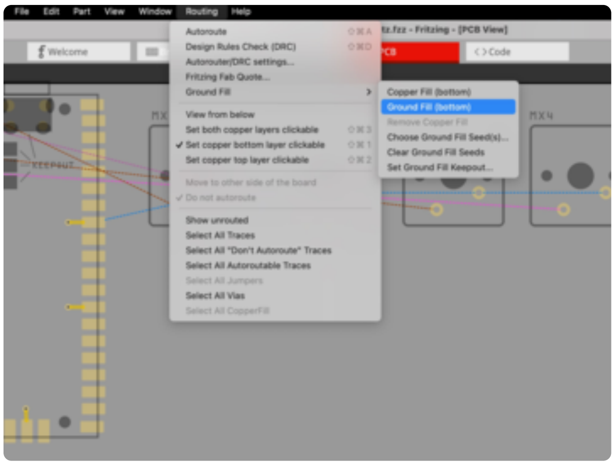
With the copper layer set to Top Layer, run a trace from each Pico GND pad to its neighboring via.

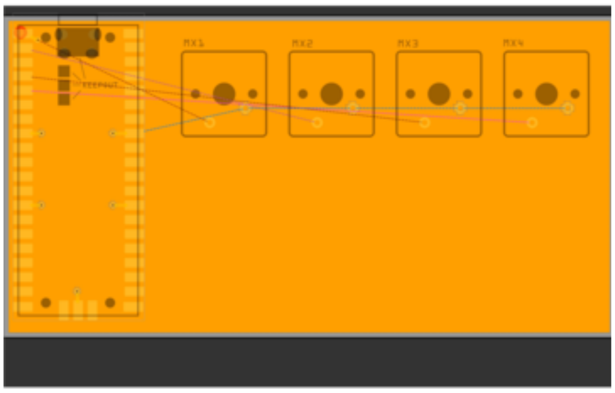
Set Ground Fill Seed

Since all of the pins we want to connect to common ground are on one network, all we need to do is specify one of them as the "ground fill seed" before we pour the fill. Right-mouse click on a Pico GND pad and pick Set Ground Fill Seed from the pop-up menu.









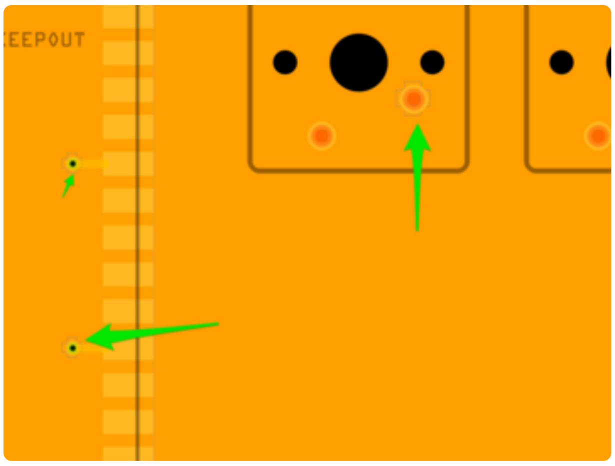
Ground Fill

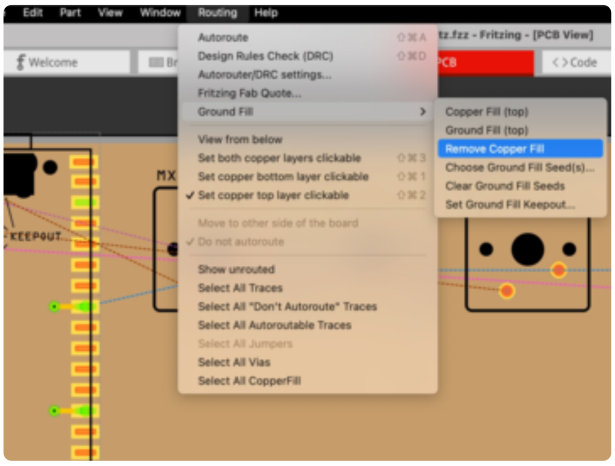
Switch to the bottom layer and delete the GPIO 0 to switch trace you made earlier. You can click it and press Backspace on the keyboard to do so.

With the Bottom Layer still active, click on Routing > Ground Fill > Ground Fill (bottom)

This will create the ground copper fill and connect each of the ground vias and pads to it.

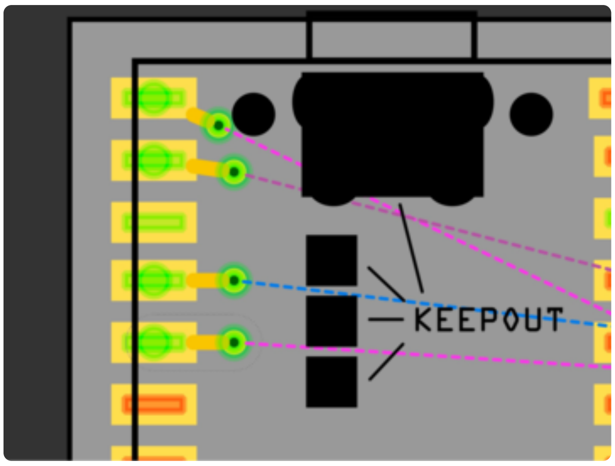
However! We have other traces to create before we can finalize the Ground Fill, so you'll remove it for now and re-create it later. Click Routing > Ground Fill > Remove Copper Fill to remove it.

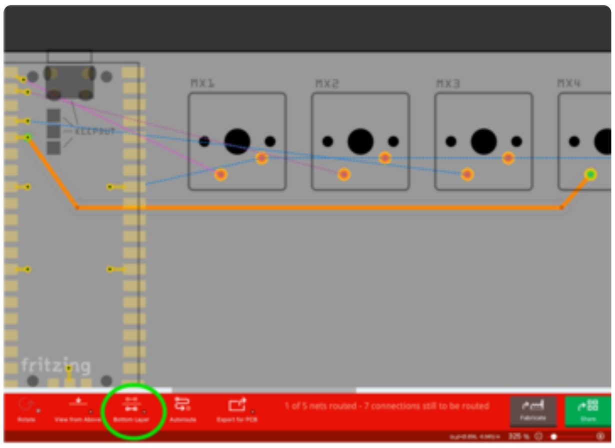


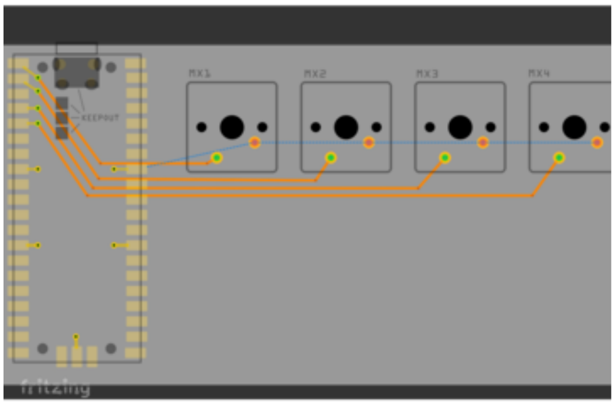


You can run the GPIO pins to the keyswitches on the bottom layer of the board to avoid having them cross the Pico's pads on the right side of the board, as well as the "keepout" area that is marked on the Pico's footprint indicating where there are pads, holes, and the USB connector pads to avoid.

(Later, when you use more keyswitches you'll route some of those pads on the top layer.)





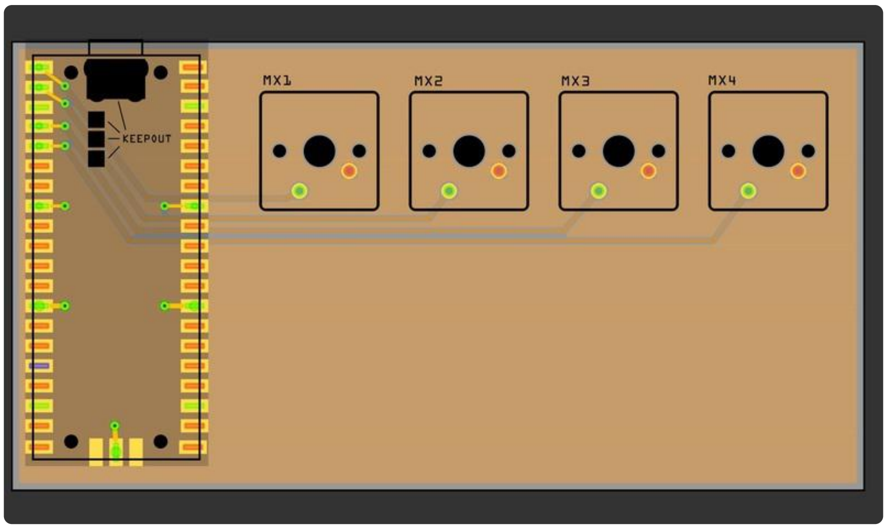


Pin Traces

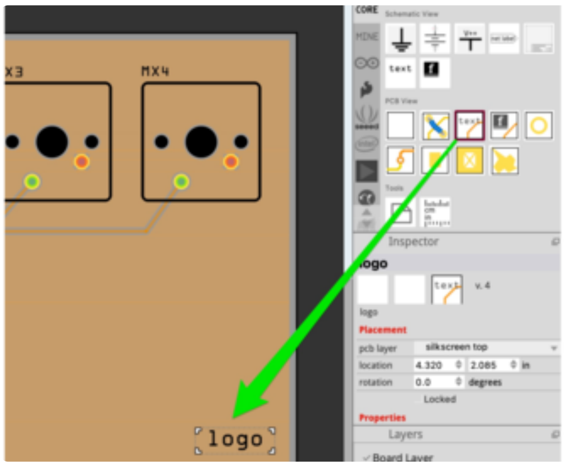
As before, add vias next to the pads, and connect them to the top layer pads for GPIO 0-3.

On the bottom layer, begin running traces from the Pico pads to the keyswitches as shown.

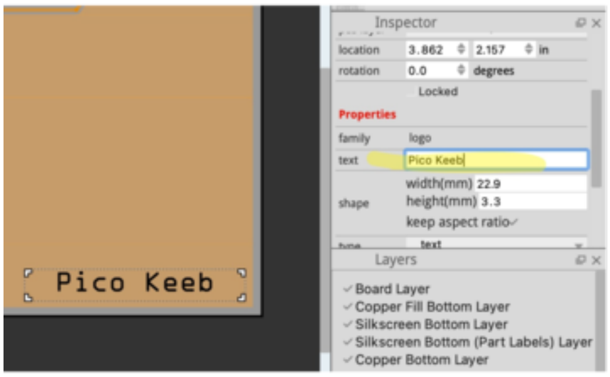
Now that the traces are all in place, re-pour the ground fill on the bottom layer.



This time, the copper ground fill has left space for all of the non-ground traces while connecting all of the ground pads to the fill.



pcb layer silkscreen top
location 4.320 2.085 in
rotation 0.0 degrees



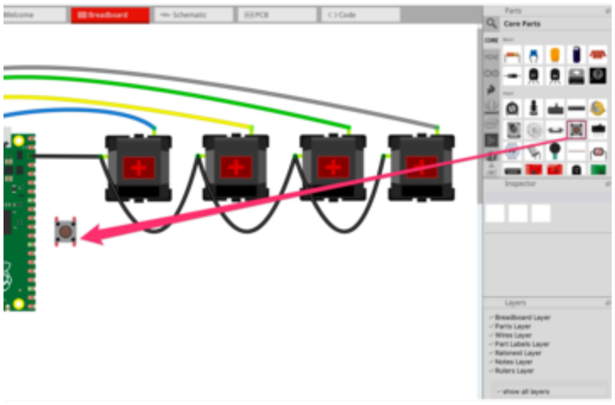
Inspector
location 3.862 2.157 in
rotation 0.0 degrees
Locked

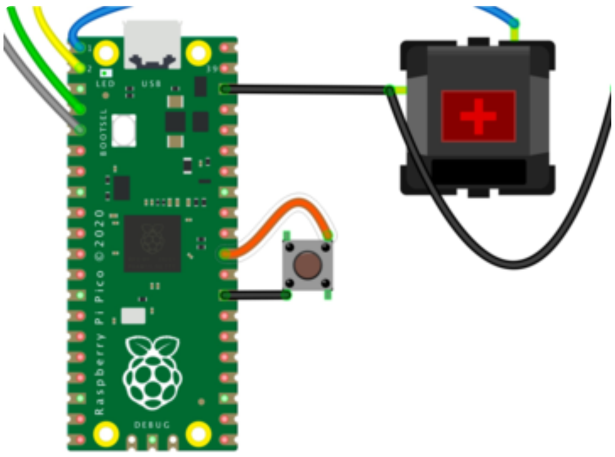
Properties
family logo
text Pico Keeb
shape width(mm) 22.9
height(mm) 3.3
keep aspect ratio

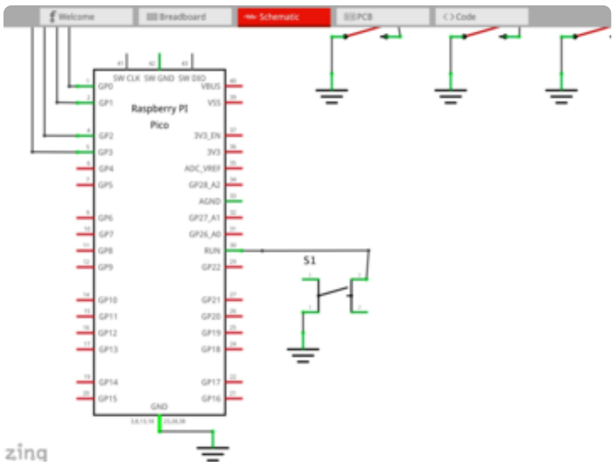
Layers
Board Layer
Copper Fill Bottom Layer
Silkscreen Bottom Layer
Silkscreen Bottom (Part Labels) Layer
Copper Bottom Layer

Silkscreen Label

If you'd like to add text to the board, you can drag the logo part from the Core PCB View parts bin on to the board and customize it.





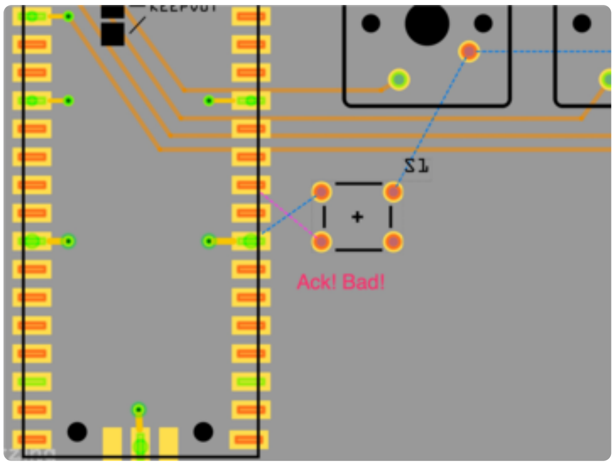


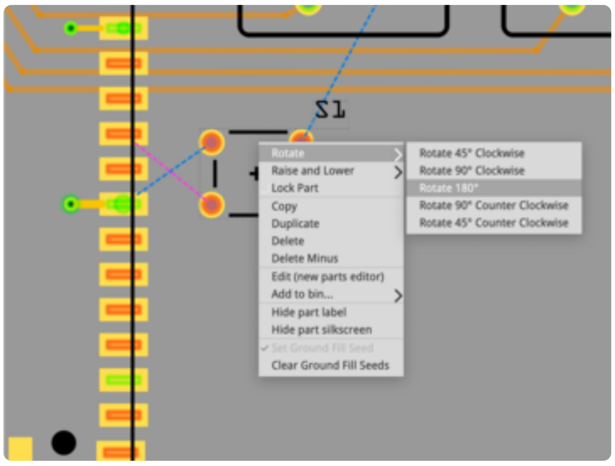
Reset Button

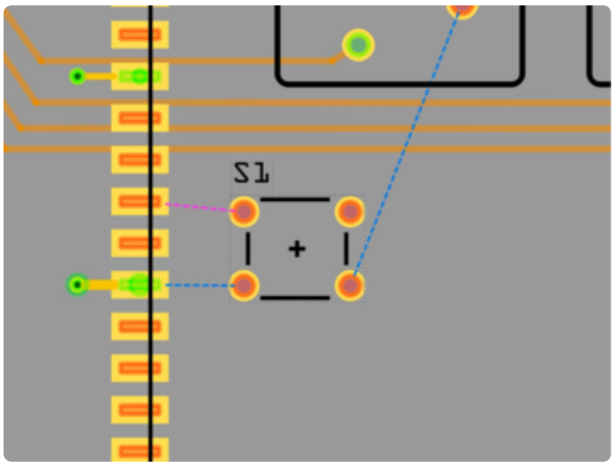
The Pico has a RUN/RESET pin which will reset the board when grounded. We'll add a small button to make this easier to use.

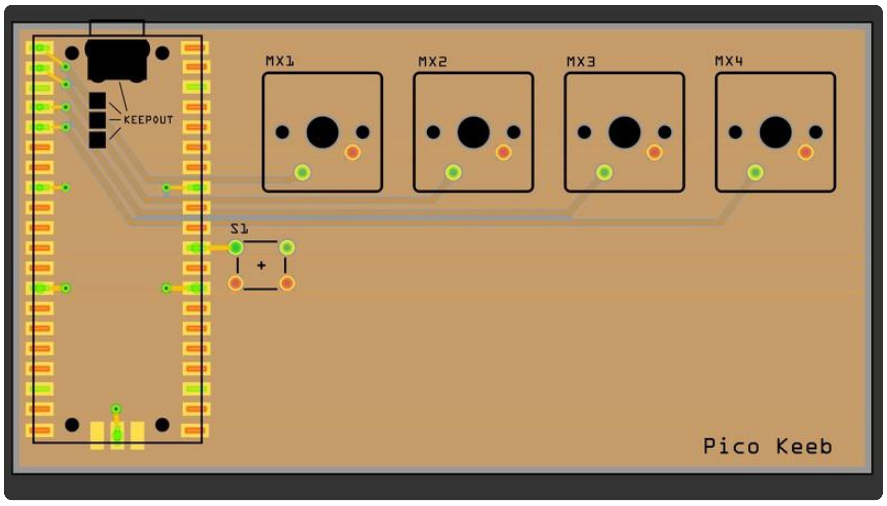
Add and wire the button the same way you've done before with the keyswitches.

Run a trace from the RESET pin to the button, and then remove and re-pour the ground fill.


S1
Ack! Bad!

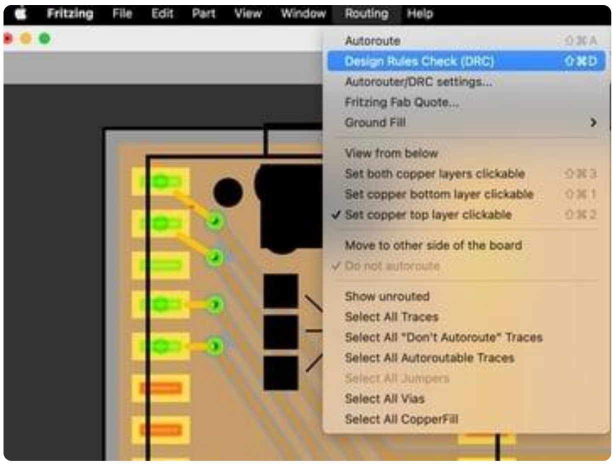

S1
Rotate
Raise and Lower
Lock Part
Copy
Duplicate
Delete
Delete Minus
Edit (new parts editor)
Add to bin...
Hide part label
Hide part silkscreen
Set Ground Fill Seed
Clear Ground Fill Seeds
Rotate 45° Clockwise
Rotate 90° Clockwise
Rotate 180°
Rotate 90° Counter Clockwise
Rotate 45° Counter Clockwise

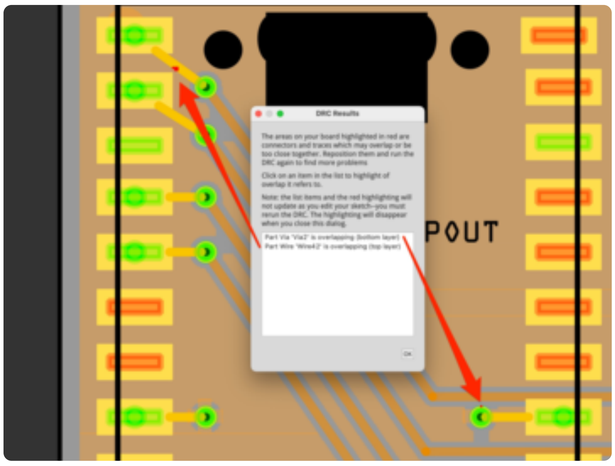

S1



Ordering PCBs

You can take your PCB design from Fritzing and export it to a set of standard Gerber files for fabrication at a PCB factory.



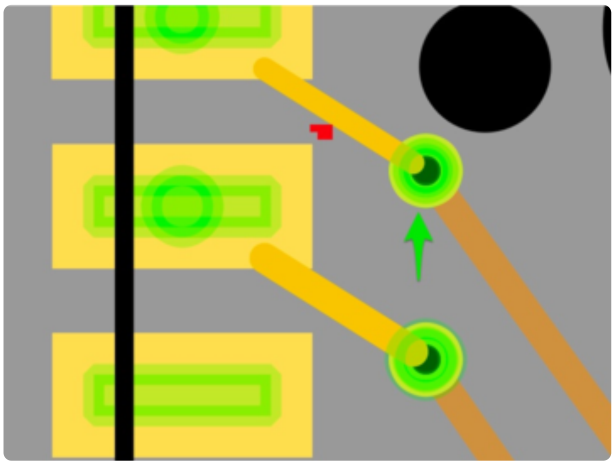


The areas on your board highlighted in red are connectors and traces which may overlap or be too close together. Reposition them and run the DRC again to find more problems

Click on an item in the list to highlight of overlap it refers to.

Note: the list items and the red highlighting will not update as you edit your sketch--you must rerun the DRC. The highlighting will disappear when you close this dialog.

Part Via 'Via2' is overlapping (bottom layer)
Part Wire 'Wire42' is overlapping (top layer)



DRC Check

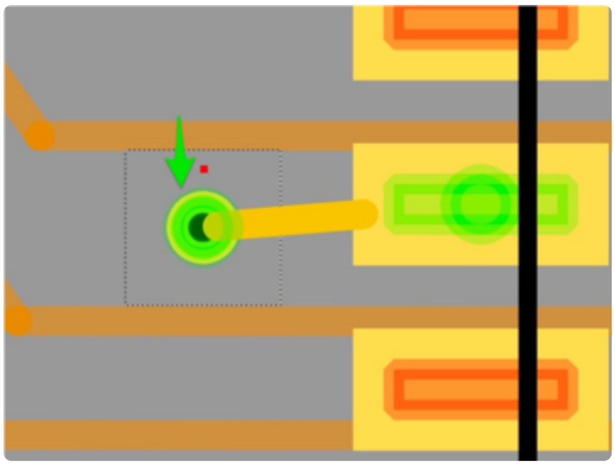
First, run a Design Rules Check (DRC) to see if Fritzing spots any errors with the board that could cause problems in manufacturing. This will check for things like accidentally overlapping traces or elements that are possibly too close together.

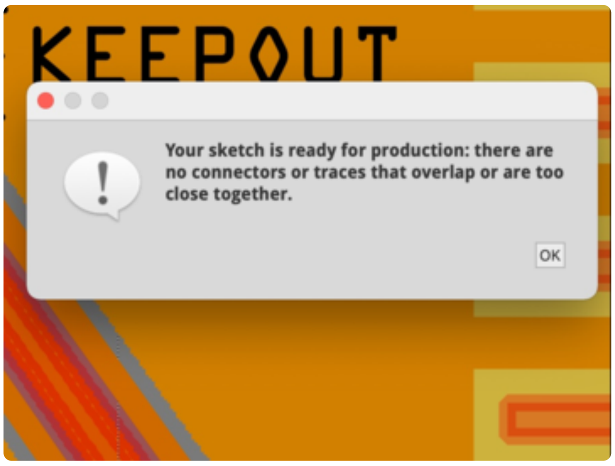
Click File > Routing > Design Rules Check (DRC) and wait for the popup window and tests to run.

There were two errors found -- a wire (trace) and a via that are a bit close for comfort to some pads.

Remove the ground fill, move the offending elements a bit and then re-pour the ground fill.

Re-run the DRC and pass with flying colors!





Your sketch is ready for production: there are no connectors or traces that overlap or are too close together.

You can download the full Fritzing file here:

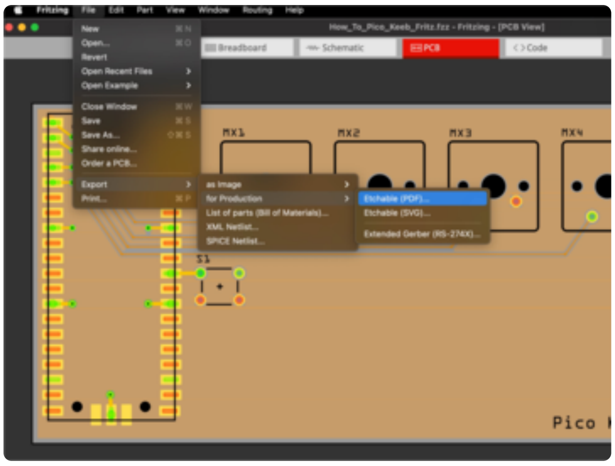
How_To_Pico_Keeb_Fritz.fzz

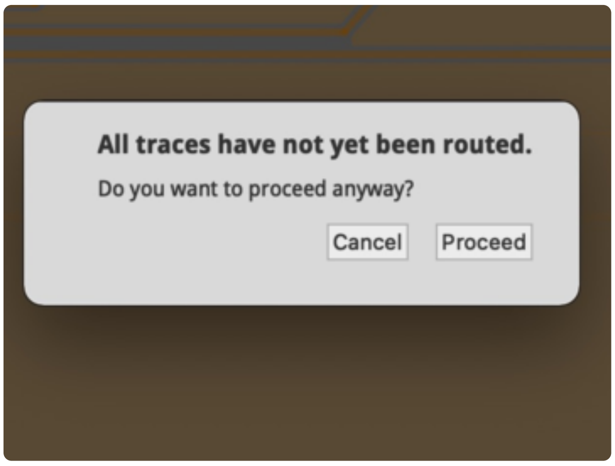
https://adafru.it/QUb

Paper Test

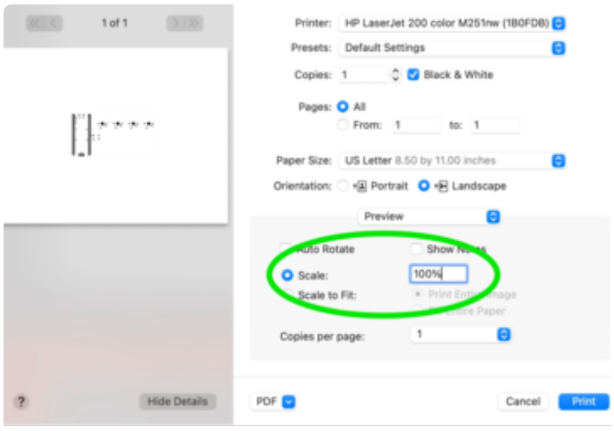
The first step is to export the design as a set of PDF files and print the copper top file at a 1:1 scale on a sheet of regular printer paper.

This is a terrific reality check for scale and placement of your physical parts.




All traces have not yet been routed.

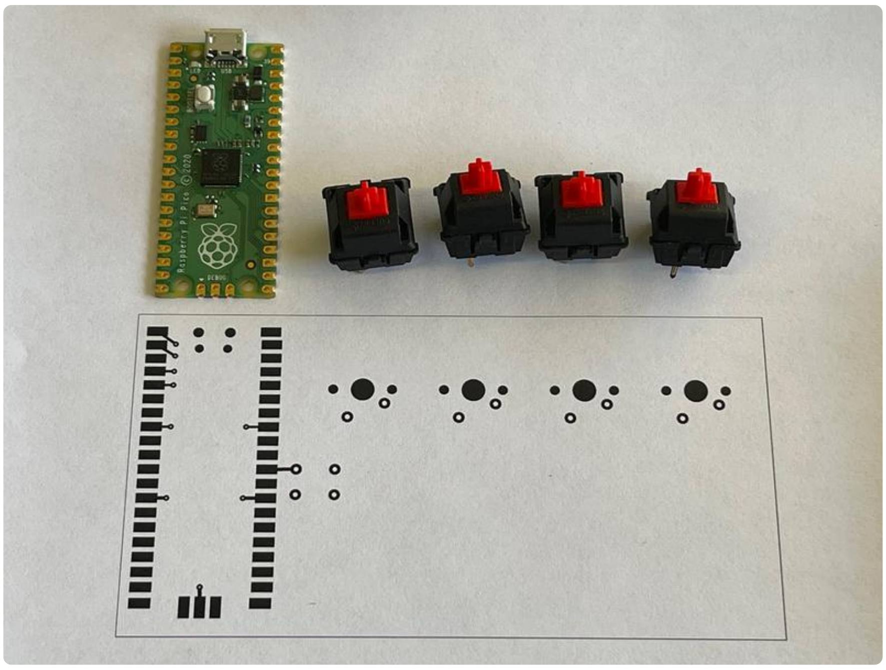
Do you want to proceed anyway?

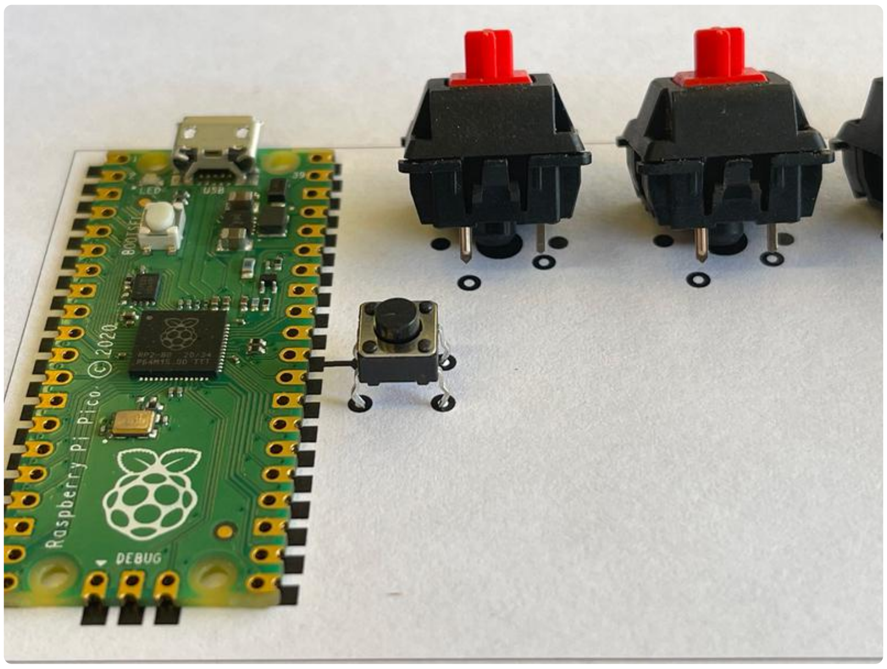

Printer: HP LaserJet 200 color M251nw (180FD8)
Presets: Default Settings
Copies: 1 Black & White ✓
Pages: All
Paper Size: US Letter 8.50 by 11.00 inches
Orientation: Landscape
Scale: 100%
Copies per page: 1

To do this, click File > Export > for Production > Etchable (PDF) and select a location to export the files. Since this creates a dozen or so files, it's good to put them in their own directory.

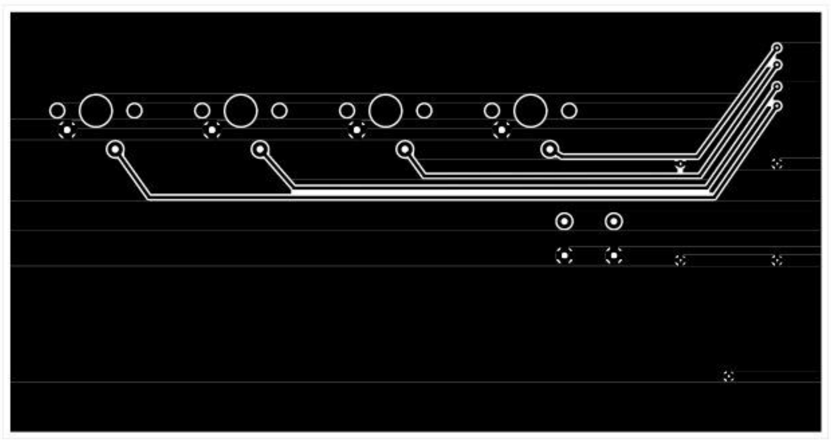
Note, due to a bug with Ground Fill and ratsnest wires, you will see a message pop up saying not all traces have been routed. If you're sure all traces have been routed it's safe to click Proceed.

Lay out your parts on the printed paper for a reality check!





The bottom layer copper mirrored view is also a good one to inspect before moving on.

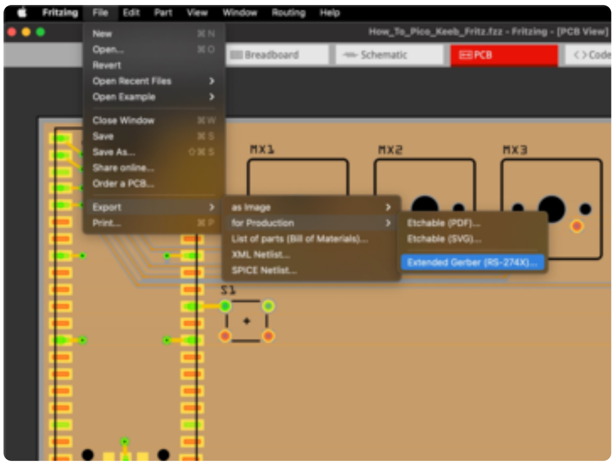


Don't be alarmed by the copper view of some elements such as the mounting holes for the switches -- you'll verify the drill hole layer during the PCB ordering step.

Gerber -- Not Just For Babies

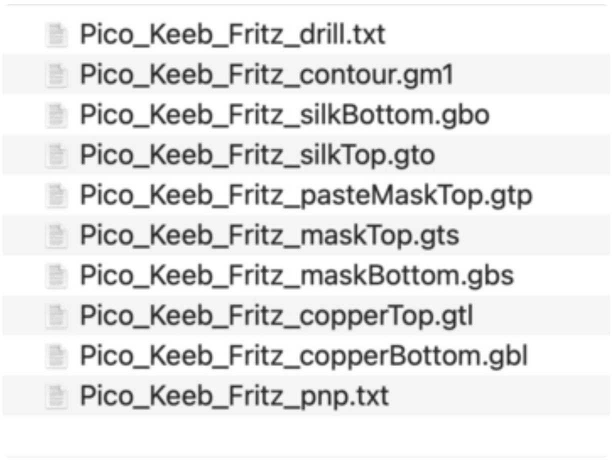
You're now ready to prep your board files for the PCB fab house! You verified your design using PDF files, but now you'll need to export Gerber files for fabrication. Gerber is the standard used in PCB fabrication, and is a set of ASCII text files that represent the different layers of the board -- typically:

- Drill holes
- Board outline (for milling)
- Top and bottom silk screen printing layers
- Top and bottom copper layers
- Top and bottom solder mask layers

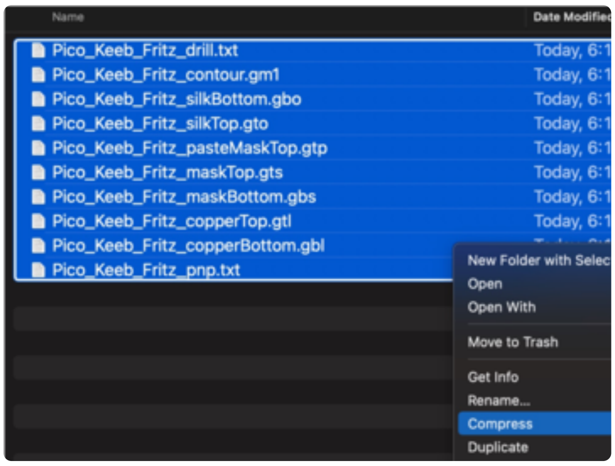


Export Gerber Files

To export your Gerber files, click on File > Export > for Production > Extended Gerber (RS-274X) and pick a location -- again it's helpful to create a fresh, empty, new-car-scented folder to store them together neatly.



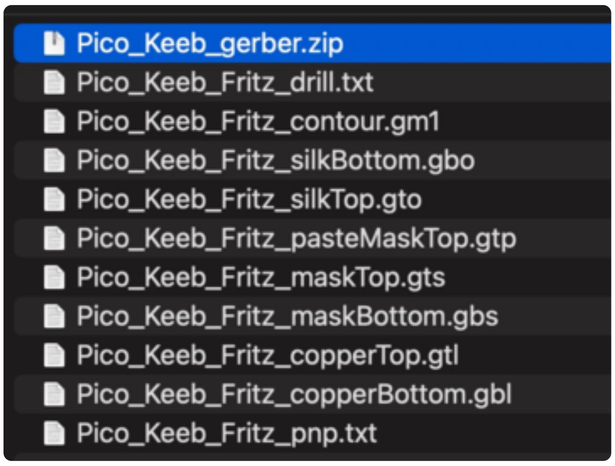
Pico_Keeb_Fritz_drill.txt
Pico_Keeb_Fritz_contour.gm1
Pico_Keeb_Fritz_silkBottom.gbo
Pico_Keeb_Fritz_silkTop.gto
Pico_Keeb_Fritz_pasteMaskTop.gtp
Pico_Keeb_Fritz_maskTop.gts
Pico_Keeb_Fritz_maskBottom.gbs
Pico_Keeb_Fritz_copperTop.gtl
Pico_Keeb_Fritz_copperBottom.gbl
Pico_Keeb_Fritz_pnp.txt



Name	Date Modified
Pico_Keeb_Fritz_drill.txt	Today, 6:1
Pico_Keeb_Fritz_contour.gm1	Today, 6:1
Pico_Keeb_Fritz_silkBottom.gbo	Today, 6:1
Pico_Keeb_Fritz_silkTop.gto	Today, 6:1
Pico_Keeb_Fritz_pasteMaskTop.gtp	Today, 6:1
Pico_Keeb_Fritz_maskTop.gts	Today, 6:1
Pico_Keeb_Fritz_maskBottom.gbs	Today, 6:1
Pico_Keeb_Fritz_copperTop.gtl	Today, 6:1
Pico_Keeb_Fritz_copperBottom.gbl	
Pico_Keeb_Fritz_pnp.txt	

Zip It Good

Some PCB houses prefer to ingest your Gerber files as a single .zip archive, so go ahead and pick all of the files you just exported and compress them into a .zip file named something like Pico_Keeb_gerber.zip

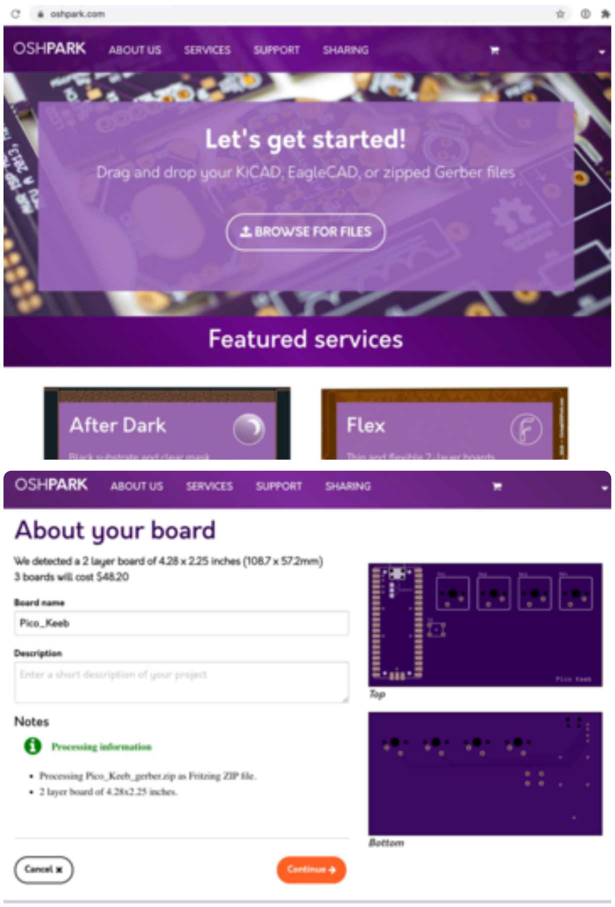


Pico_Keeb_gerber.zip
Pico_Keeb_Fritz_drill.txt
Pico_Keeb_Fritz_contour.gm1
Pico_Keeb_Fritz_silkBottom.gbo
Pico_Keeb_Fritz_silkTop.gto
Pico_Keeb_Fritz_pasteMaskTop.gtp
Pico_Keeb_Fritz_maskTop.gts
Pico_Keeb_Fritz_maskBottom.gbs
Pico_Keeb_Fritz_copperTop.gtl
Pico_Keeb_Fritz_copperBottom.gbl
Pico_Keeb_Fritz_pnp.txt

Upload and Verify

There are lots of places to have your PCBs made -- I'm a fan of both OSHPark and JLCPCB in particular, and I know people who like PCBWay a lot too. I'd recommend OSHPark for your first boards as they have terrific customer service and a great UI for helping you through the process.

Head to oshpark.com (https://adafru.it/e2G) and then drag your Pico_Keeb_gerber.zip file onto the "Let's get started!" box. They'll ingest the zip, extract the files, and invite you to inspect the layers.



About your board

We detected a 2 layer board of 4.28 x 2.25 inches (108.7 x 57.2mm)
3 boards will cost $48.20

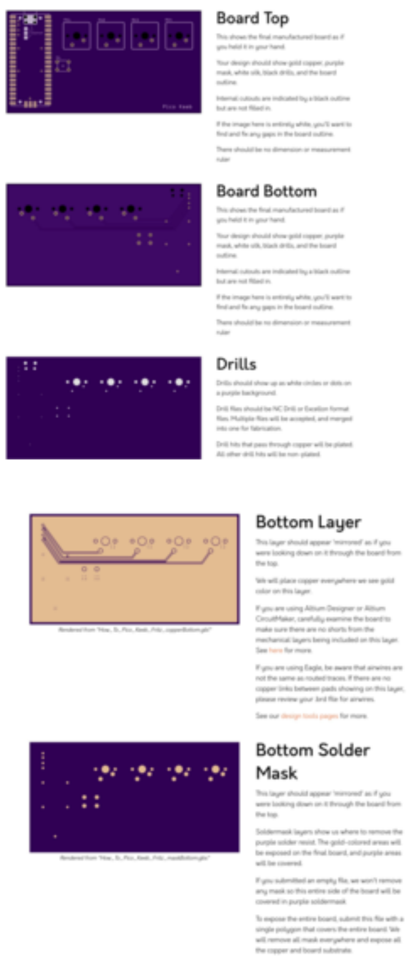
Board name: Pico_Keeb

Description: Enter a short description of your project

Notes

Processing information

- Processing Pico_Keeb_gerber.zip as Fritzing ZIP file.
- 2 layer board of 4.28x2.25 inches.



Board Top

This shows the final manufactured board as if you held it in your hand.

Your design should show gold copper, purple mask, white silk, black drills, and the board outline.

Internal cutouts are indicated by a black outline but are not filled in.

If the image here is entirely white, you'll want to find and fix any gaps in the board outline.

There should be no dimension or measurement ruler.

Board Bottom

This shows the final manufactured board as if you held it in your hand.

Your design should show gold copper, purple mask, white silk, black drills, and the board outline.

Internal cutouts are indicated by a black outline but are not filled in.

If the image here is entirely white, you'll want to find and fix any gaps in the board outline.

There should be no dimension or measurement ruler.

Drills

Drills should show up as white circles or dots on a purple background.

Drill files should be NC Drill or Excellon format files. Multiple files will be accepted, and merged into one for fabrication.

Drill hits that pass through copper will be plated. All other drill hits will be non-plated.

Bottom Layer

This layer should appear 'mirrored' as if you were looking down on it through the board from the top.

We will place copper everywhere we see gold color on this layer.

If you are using Altium Designer or Altium CircuitMaker, carefully examine the board to make sure there are no shorts from the mechanical layers being included on this layer. See here for more.

If you are using Eagle, be aware that airwires are not the same as routed traces. If there are no copper links between pads showing on this layer, please review your .brd file for airwires.

See our design tools pages for more.

Bottom Solder Mask

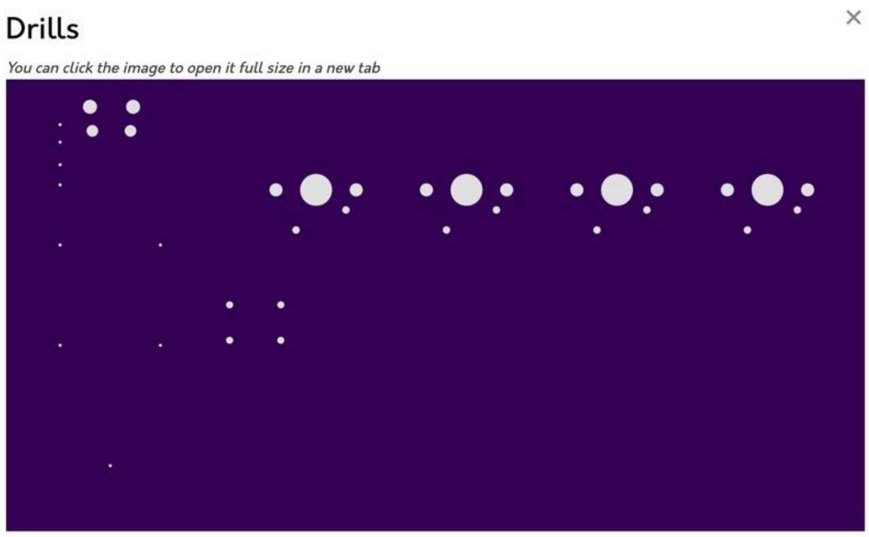
This layer should appear 'mirrored' as if you were looking down on it through the board from the top.

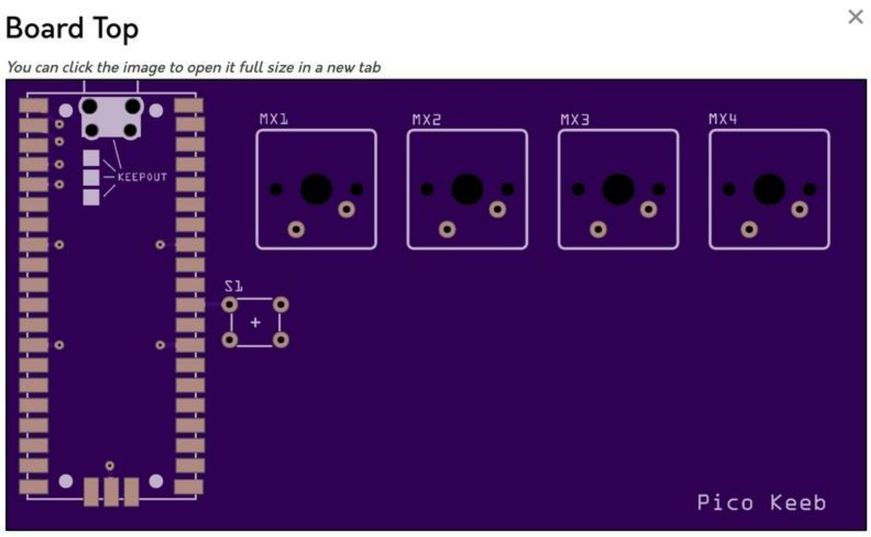
Soldermask layers show us where to remove the purple solder resist. The gold-colored areas will be exposed on the final board, and purple areas will be covered.

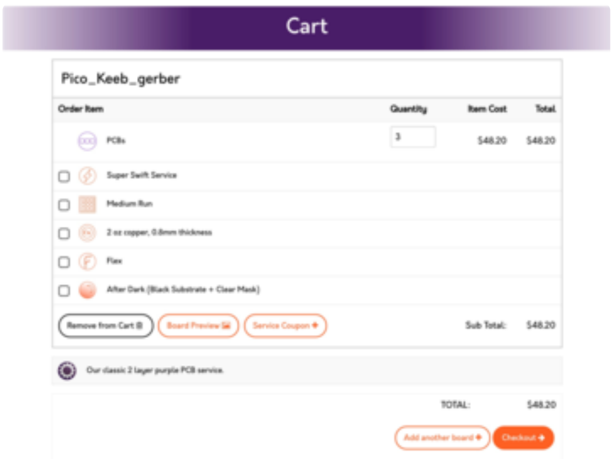
If you submitted an empty file, we won't remove any mask so this entire side of the board will be covered in purple soldermask.

To expose the entire board, submit this file with a single polygon that covers the entire board. We will remove all mask everywhere and expose all the copper and board substrate.

Hey look! The Drills layer looks correct, whew.







Cart

Pico_Keeb_gerber

Order Item	Quantity	Item Cost	Total
PCBs	3	$48.20	$48.20
☐ Super Swift Service			
☐ Medium Run			
☐ 2 oz copper, 0.8mm thickness			
☐ Flex			
☐ After Dark (Black Substrate + Clear Mask)			
		Sub Total:	$48.20

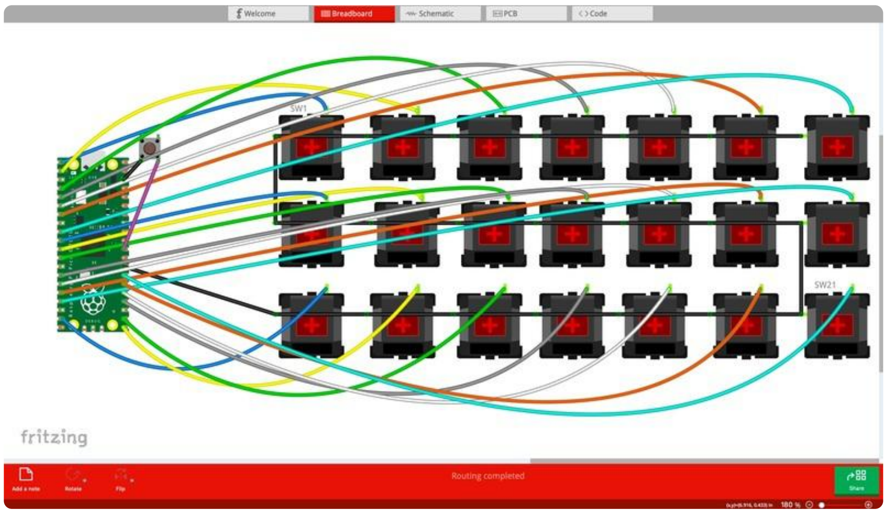
Our classic 2 layer purple PCB service.

TOTAL: $48.20

If you're happy with the board, you can go ahead and order (minimum of three boards) with the default settings, or if you don't mind waiting a bit longer, pick the After Dark option for that stylish black and copper look.

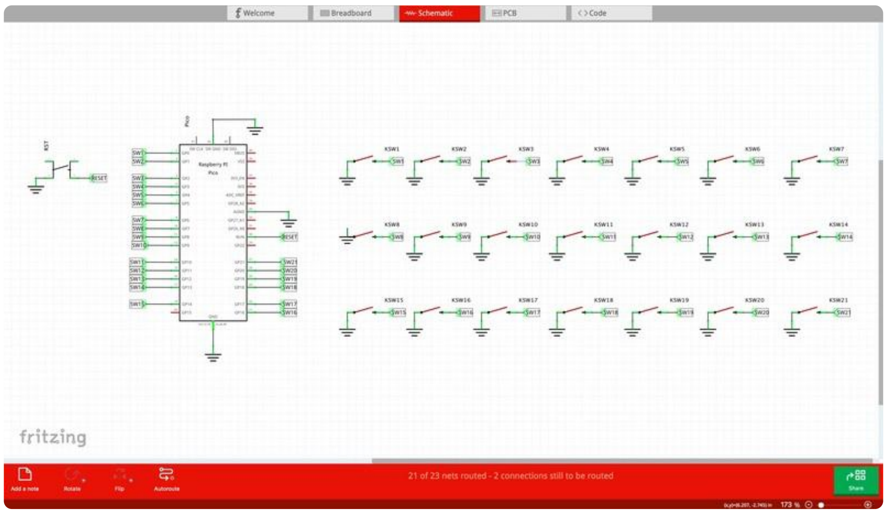
If you're in a real hurry, and don't mind "classic" green PCBs, you can get your boards made quickly and inexpensively at JLCPCB (https://adafru.it/QNE), just use the default options. The boards are made in 1-2 days, and DHL shipping to the US from China takes less than a week. I got five of the 21-key Pico Keyboard PCBs we'll look at on the next page in less than a week for $26.10 total, including shipping!

21-Key Pico Keyboard

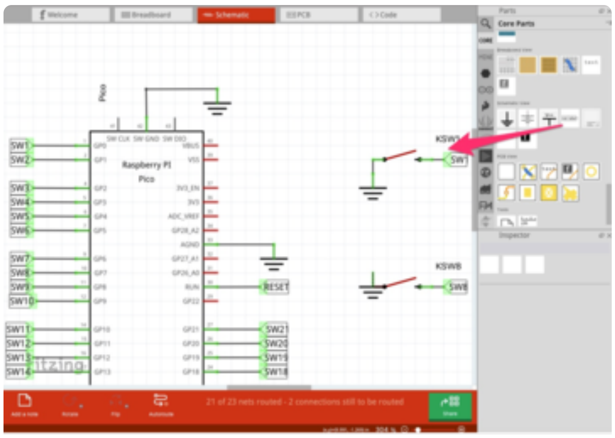


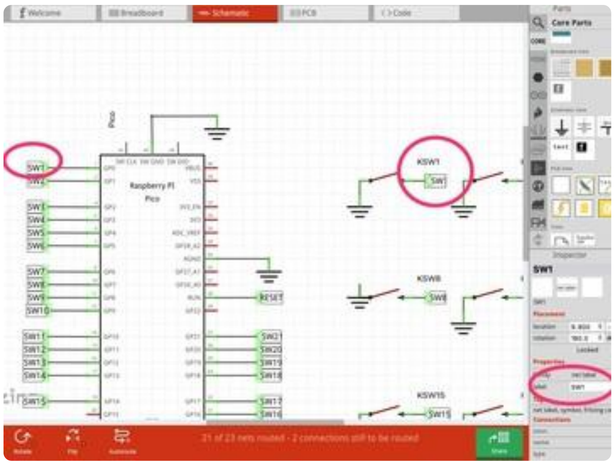
You can take things a step farther in complexity by adding more keys! I decided to build a 21-key keyboard, which isn't too different from the previous example, but offers an opportunity to look at some Fritzing tips as your project becomes more complex.

Here's a look at the Schematic view for the above design:



You'll notice the lack of wires running everywhere. This is because you can use net labels to do the same thing with the GPIO pin connections that we did earlier with the GND symbols. This makes the connections a bit less obvious, so sometimes the practice is frowned upon, but in the case of a lot of nearly identical keyswitches, it's a nice solution to the dreaded "web of wires" problem.



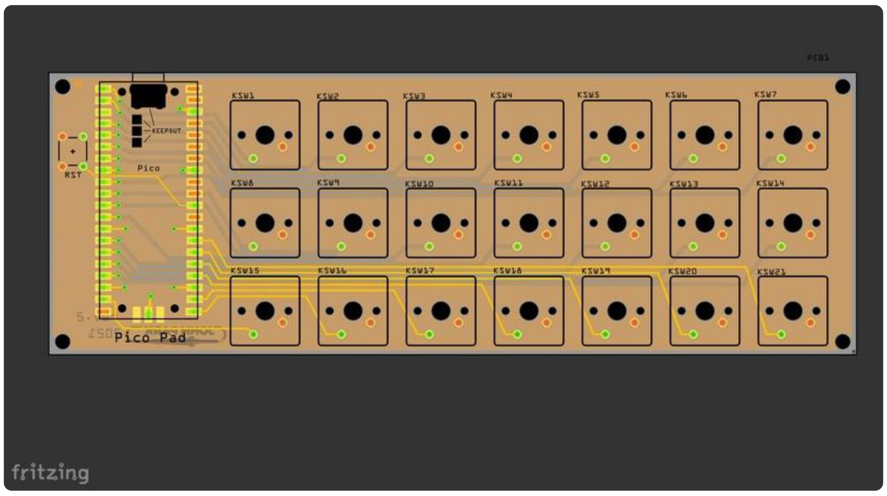


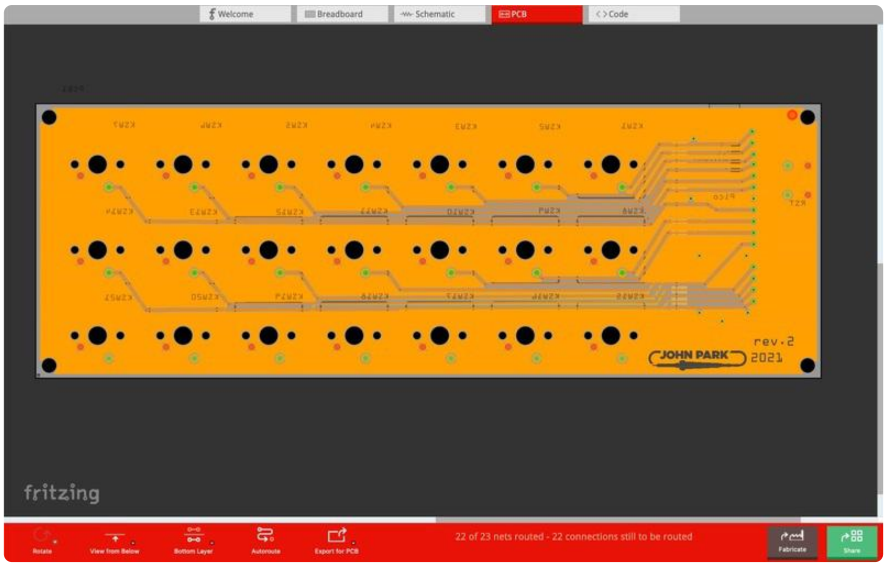
Net Labels

Drag the net label object from the Core parts bin Schematic View section onto your design. You'll need two of them per wire and their connectivity is based upon naming the networks. For example, two net labels named `SW1` are used to connect keyswitch 1 to Pico GP0

PCB Layout

In the PCB view, you'll lay out the parts and traces just the same as before, only more! Note how both the top and bottom layers are used this time to run the traces without overlapping.





Here's the full Fritzing file for this board:

Pico_21_Keyboard.fzz

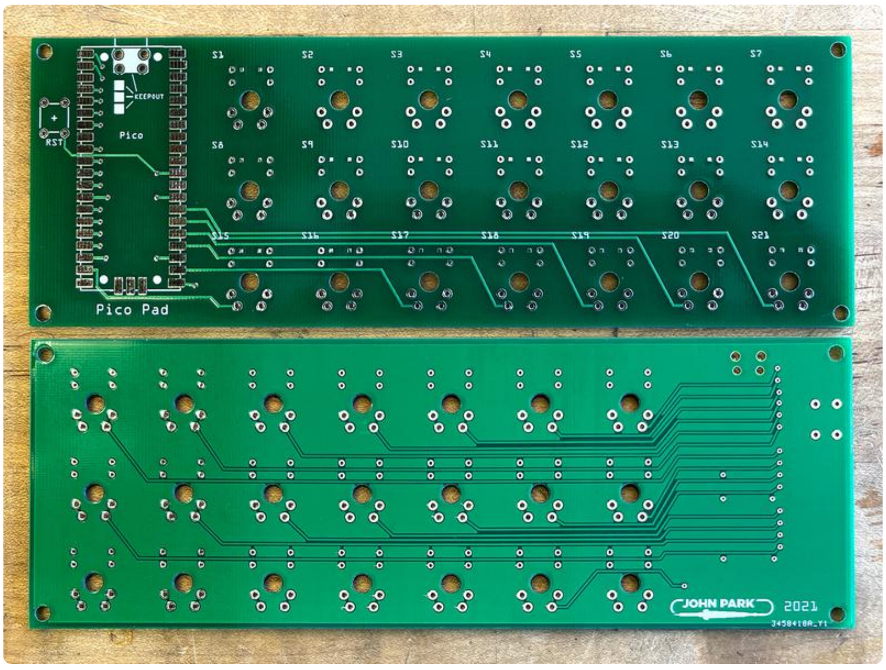
https://adafru.it/QUc

Here is the set of Gerbers for this design:

Pico_21_keyboard_gerbers.zip

https://adafru.it/QUd

Here's a set of the PCBs (note these used a different keyswitch footprint that had extra unnecessary holes that were removed in subsequent versions of the part).



You can order a set directly from this OSH Park link (https://adafru.it/TBP) if you like.

Pico Keyboard Assembly

Now you can assemble the keyboard. You've got four options here:

1. No case at all, just raw PCB, Pico, keyswitches, keycaps, and a dream!
2. 3D printed case
3. Laser cut case using either acrylic or wood
4. Combo laser cut and 3D printed case (a.k.a. "The Ice Cream Sandwich")

If you'd like to 3D print your case and switchplate, go ahead and get the model files from the link below.

Pico Keyboard Case 3D Model

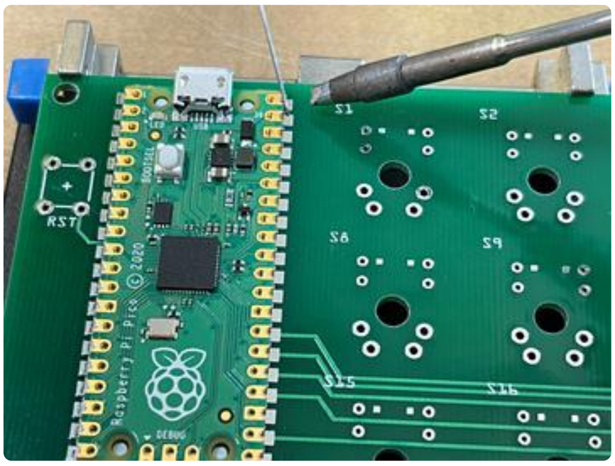
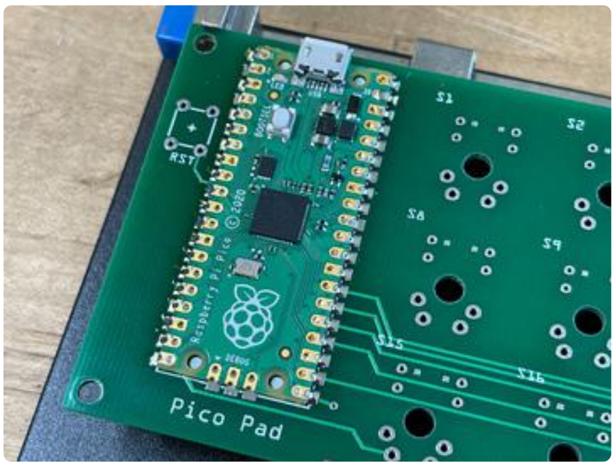
https://adafru.it/QOa

If you'd rather use 2D dimensional drawings to laser cut, hand craft, or mill your case, use this .svg file linked below.

The key plate should be made from 1.5mm material, while the top and bottom can be whatever you like so long as you have the standoffs and screws needed. I made the top and bottom of my case from 3mm acrylic.

Pico_Pad_laser_case.svg

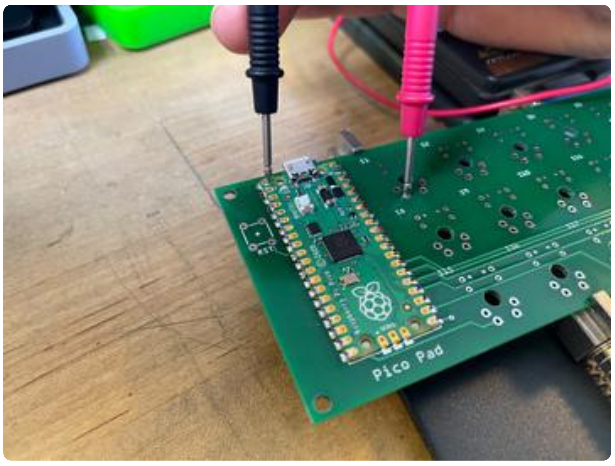
https://adafru.it/QOb

Solder the Pico

To begin, solder the Pico to the PCB. Heat up the joint between the two boards and then flow solder in, making sure to keep the boards aligned while soldering the first corner.

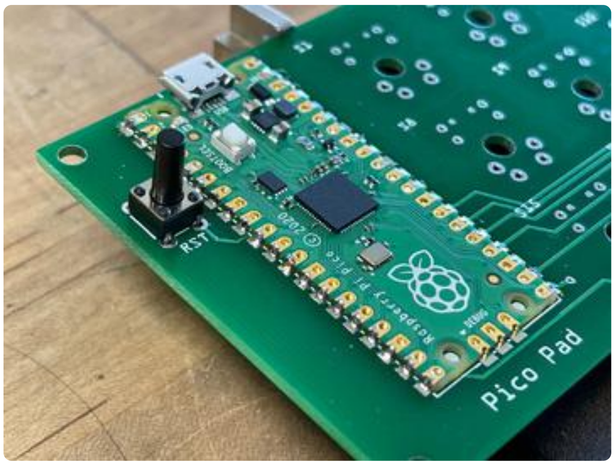
Solder all four corners and then work your way through the rest of the pads.



Test the Joints

Once the boards are soldered, use a multimeter in continuity mode to check the connections between each GPIO pin and its associated keyswitch.

It's a good idea to also double check that there are no shorts between ground and power pins.

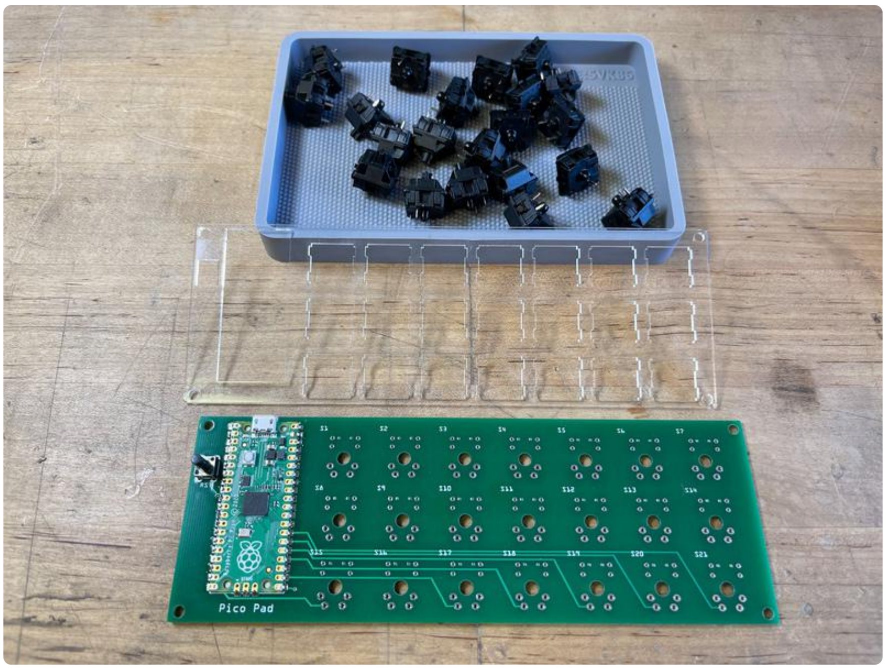


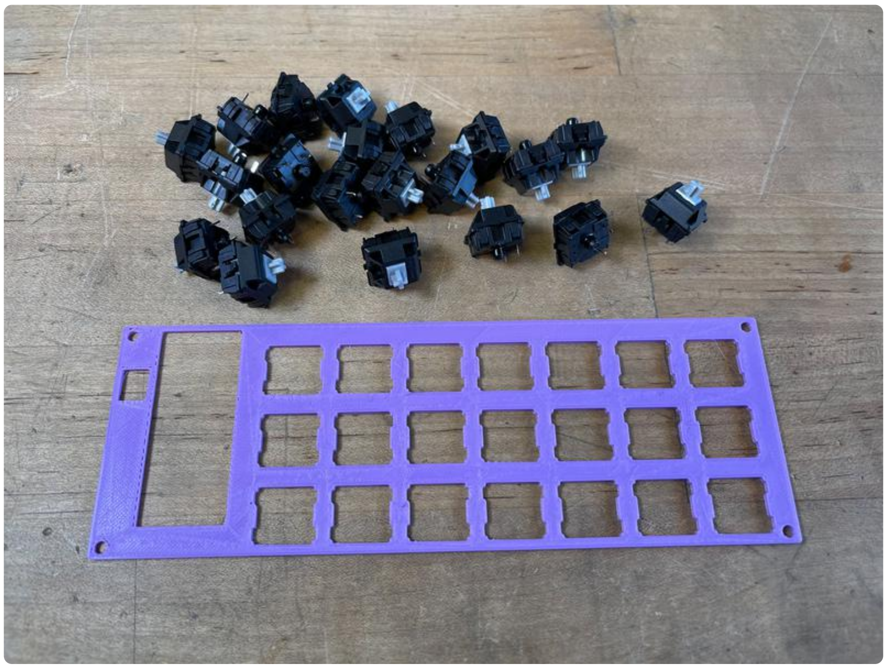
Reset Button

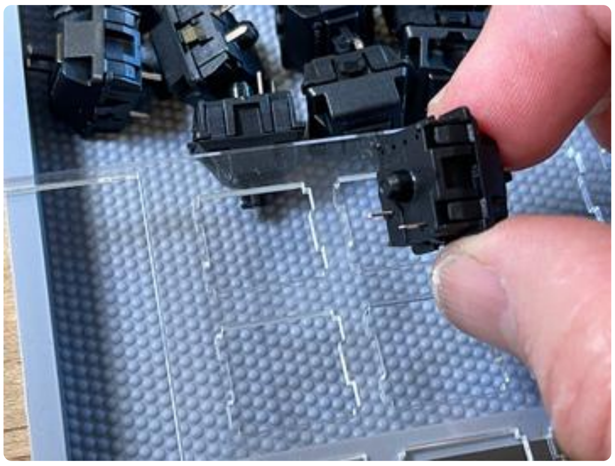
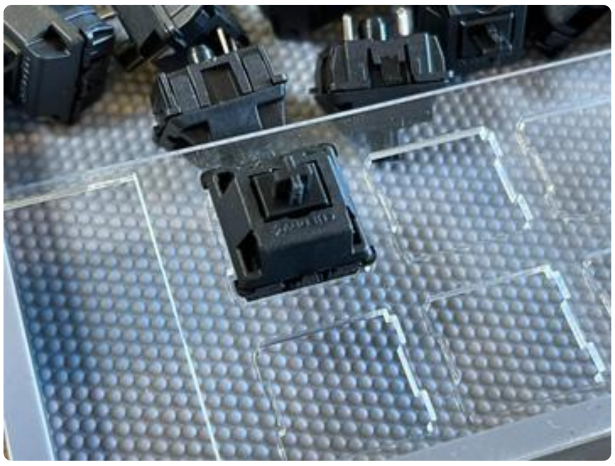
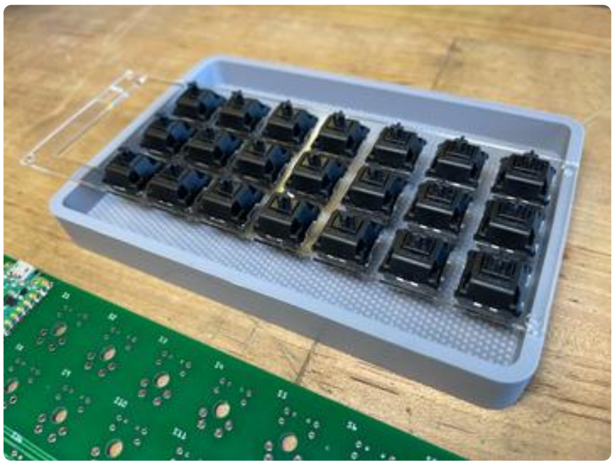
Solder the reset button in place next.

Switch Plate

The switch plate holds all of the keyswitches in places so they are stable and well aligned. Relying on the soldered pins and plastic pegs alone is definitely possible, but tends to lead to misalignment and wobbliness.


Pico Pad



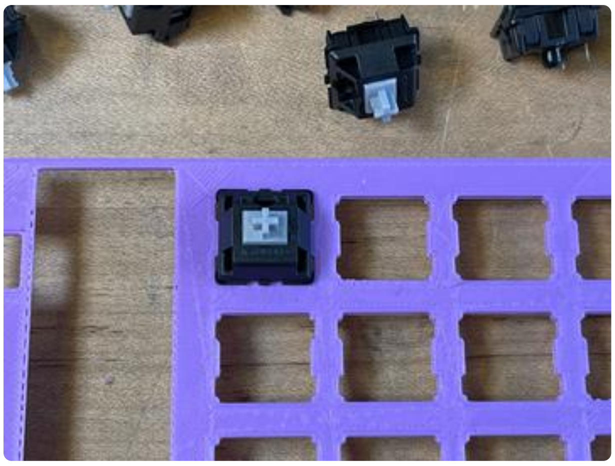
Add Switches to Plate

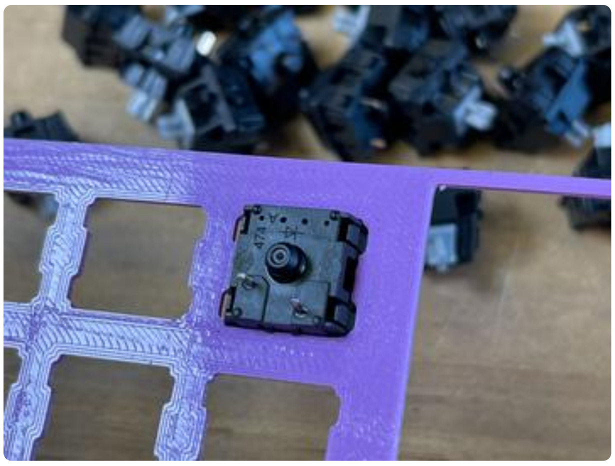
You'll need to push the keyswitches through the switchplate before mounting and soldering them to the PCB.

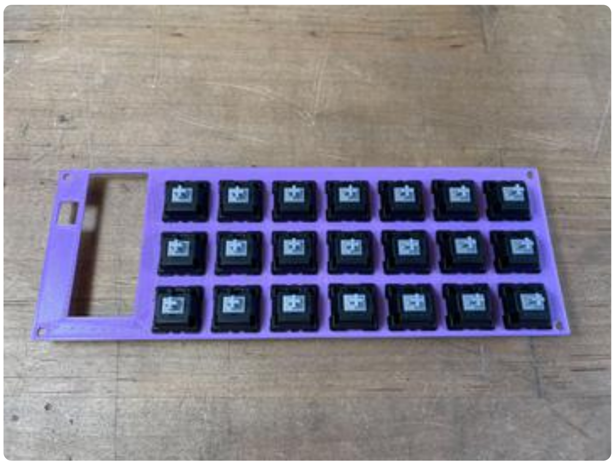
Orient the plate with the Pico cutout on the left and the reset switch cutout in the proper upper position.

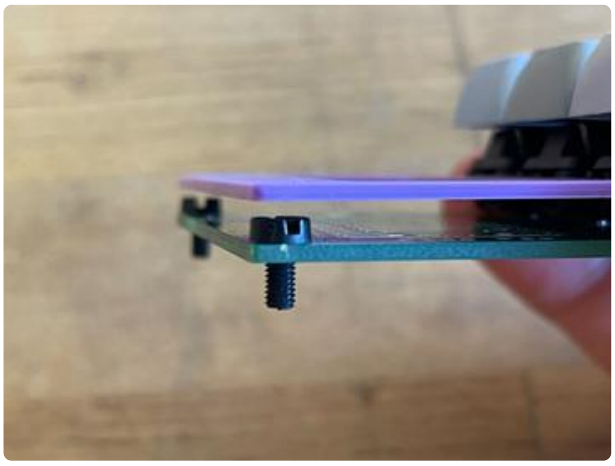
Orient the keyswitches to match the PCB -- pins at the bottom.

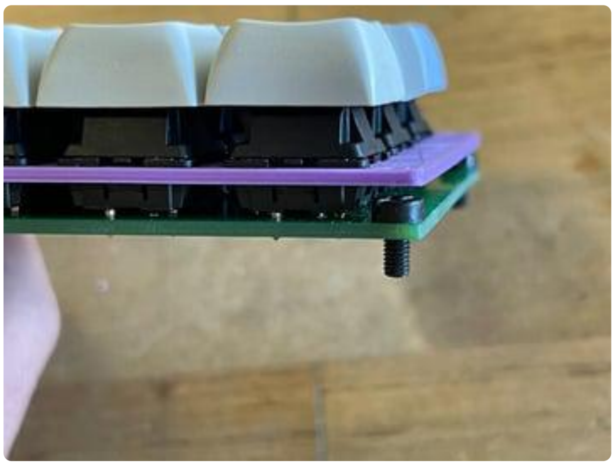
Push the keyswitches through from the top of the plate, being sure they click satisfyingly into place. Careful about applying too much force to the acrylic or it can break.







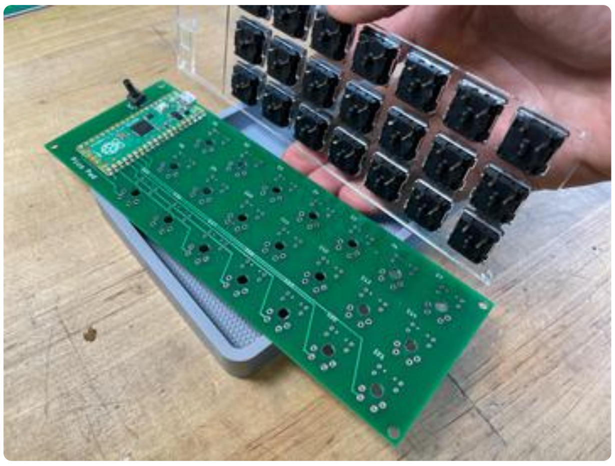
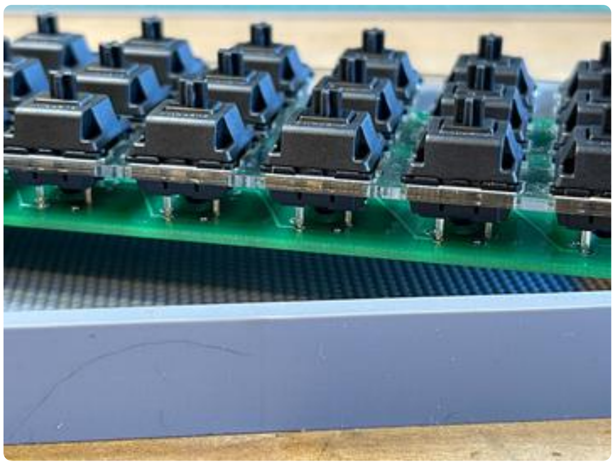
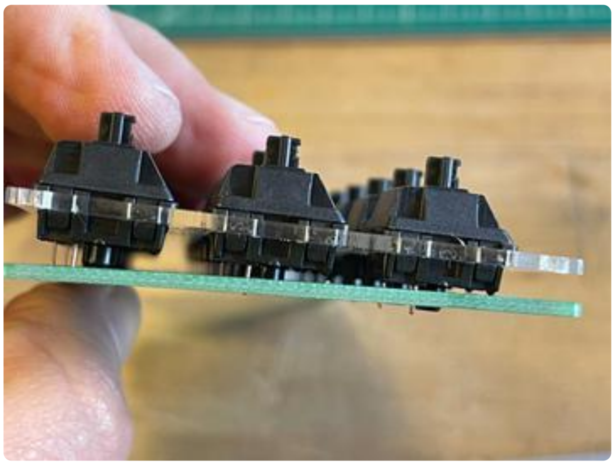




Screw Prep

Before proceeding, place the four M3 screws into the PCB mounting holes as shown.

These mount the PCB to the case bottom.

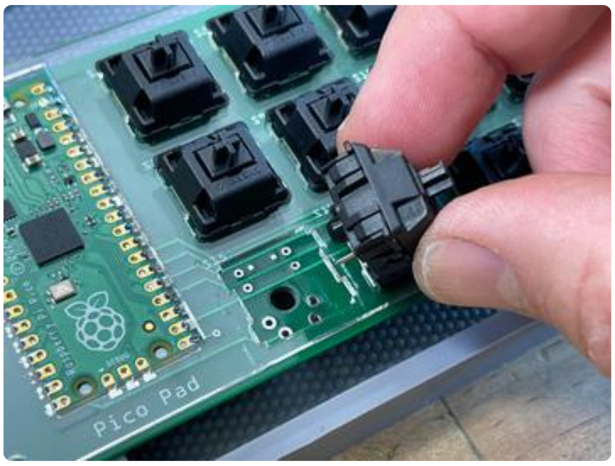
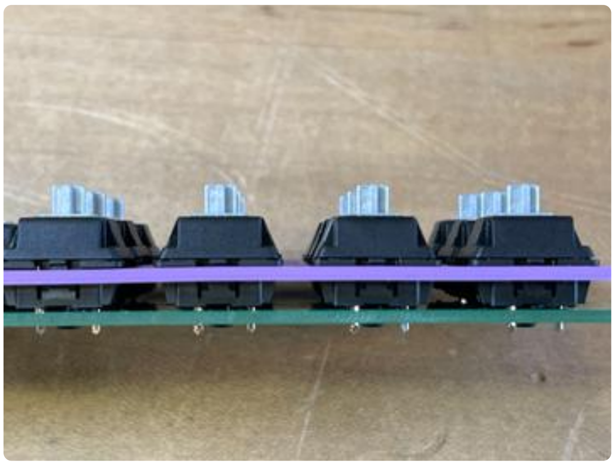
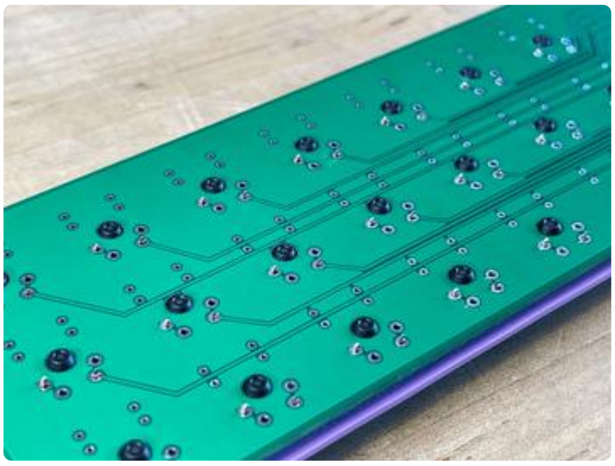




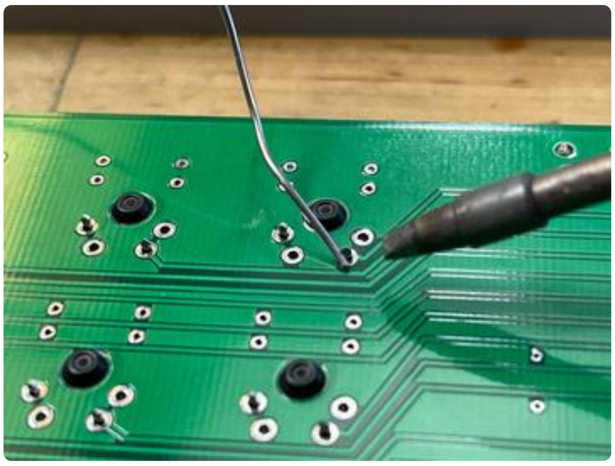
Seat the Switches

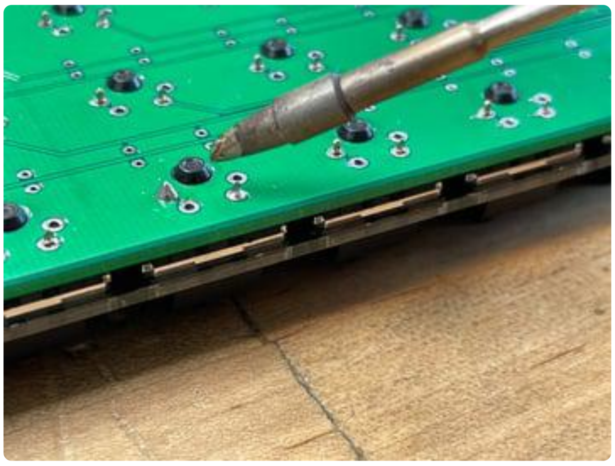
Carefully inspect each switch leg to make sure they are all straight. It's crucial to the next step that none of them are bent, or they won't seat properly in to the holes.

Align the switches with the PCB from the top and push all of the switches into place. Take it slowly and make sure none of the legs bend! You can use a thin pick/probe to help them along if needed.

If you have issues with a particular switch, squeeze its retention tabs and pull it from the plate, then adjust the pins and re-seat it.


Pico Pad





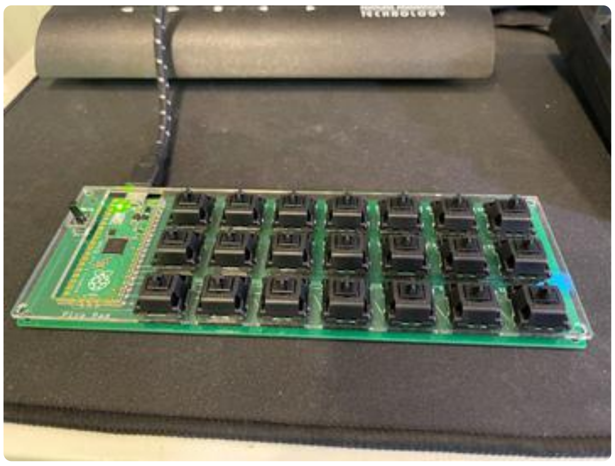


Solder the Switches

Flip the board over and start soldering! Make sure you have a clean, hot iron, a ventilation fan, and some good quality solder.

Squeeze the key toward the PCB as you solder the first few switches to make sure they're fully seated (I did the four corner switches first). After that the switchplate should assist you in keeping things level and snug.

Make your way through all 42 solder joints like a boss.



Testing

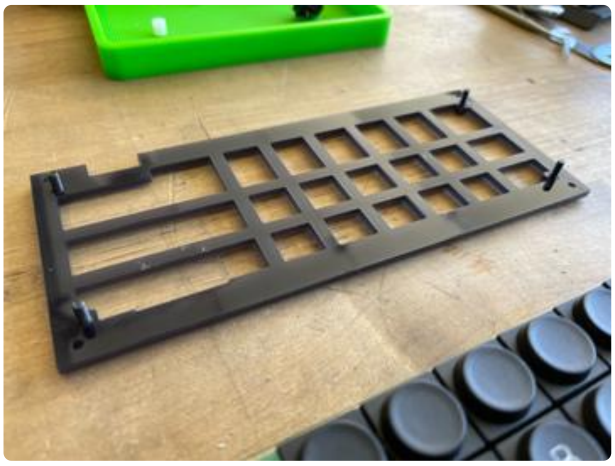
This is a great time to test your work, before enclosing the keyboard in a case.

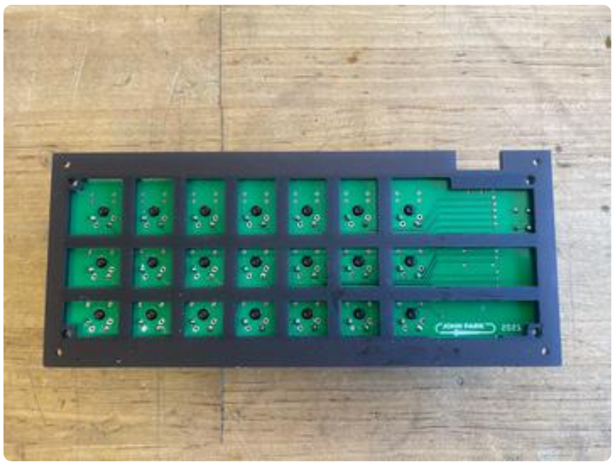
Plug in the Pico over USB and try out each key by typing into a text editor, or just watching the serial output of your code editor.

Final Assembly

With the functional parts of the keyboard built and tested, you now get to add the keycaps and case. Such fun!

First, the laser/milled style case, then the 3D one.

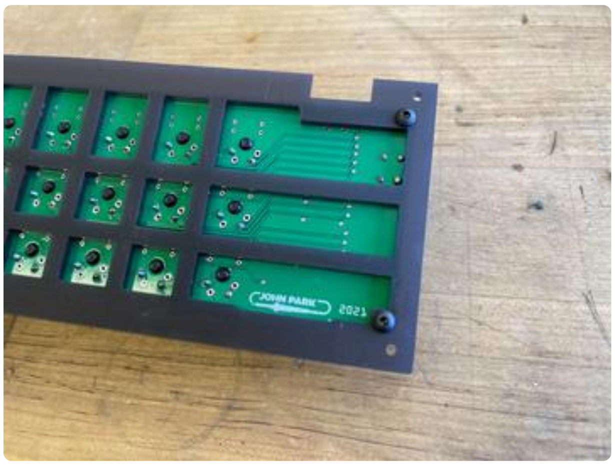
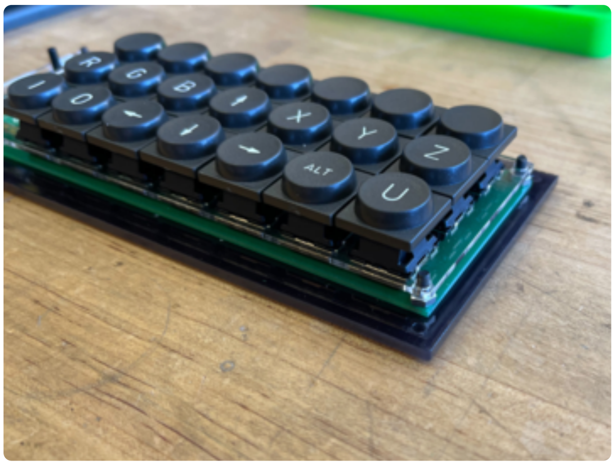


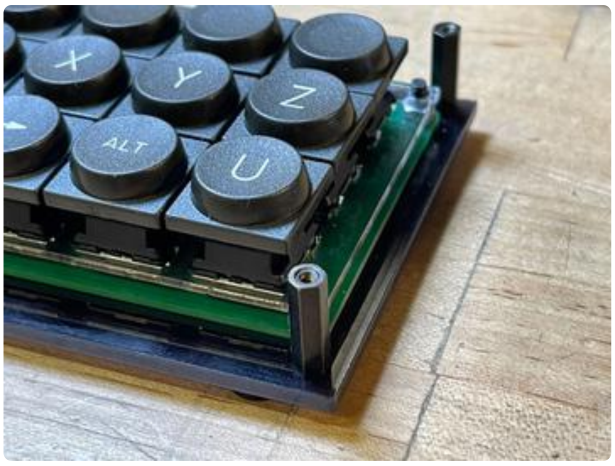


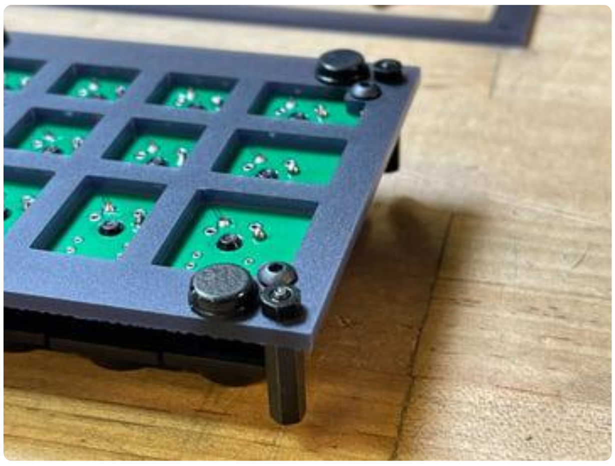
Laser Case Bottom

Use the M3 screws and nuts to affix the PCB to the case bottom.

The patterns in the base allow for the keyswitch posts and soldered pin legs clearance for a nice flush fit against the PCB.


JOHN PARK
2021

R
G
B
I
O
X
Y
Z
ALT
U



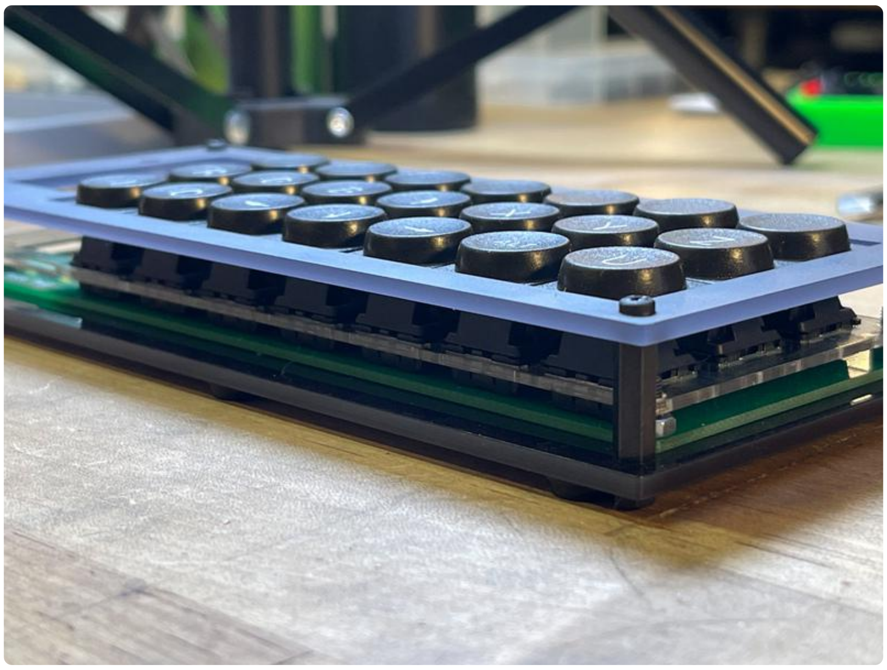


Laser Case Top

Add the four M2.5 x 16mm brass standoffs to the four corners, tightening the nuts from the bottom.

Place six rubber feet under the base as well.

Screw the case top onto the standoffs from above using M2.5 x 6mm nylon screws.

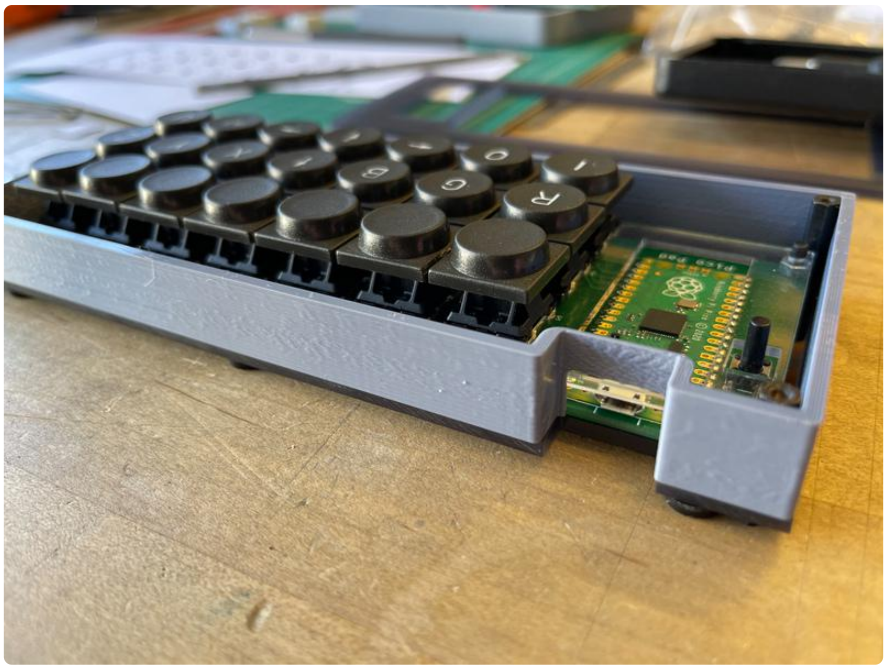


Optional Creamy Center

If you'd like to 3D print a middle section for this version of the case, use the file attached below.

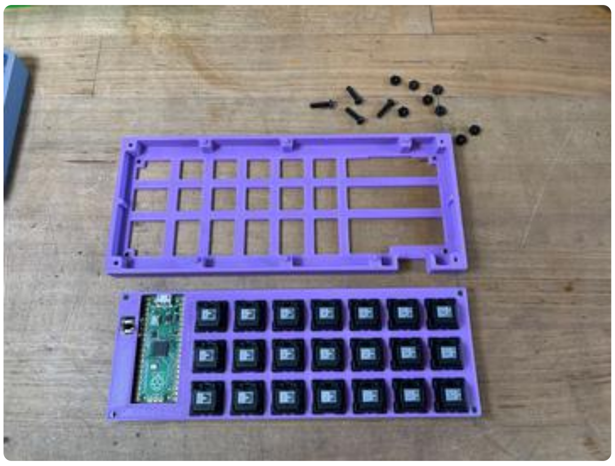
Pico_Pad_sandwich_center.stl

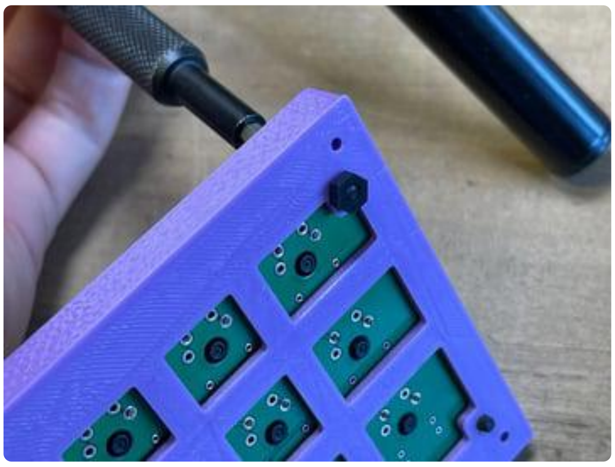
https://adafru.it/QOc



3D Printed Case

The 3D printed case assembles in a similar way.



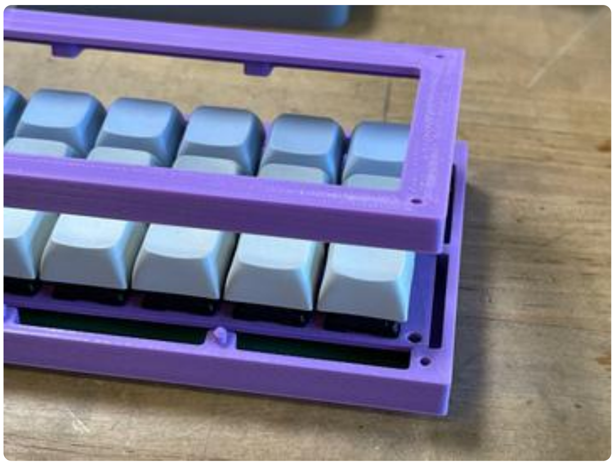


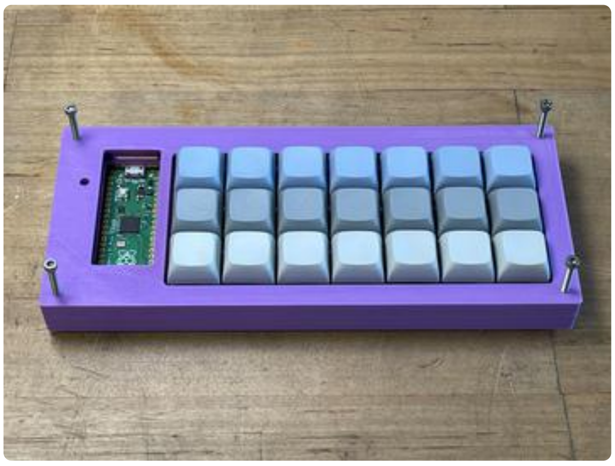
3D Base Attachment

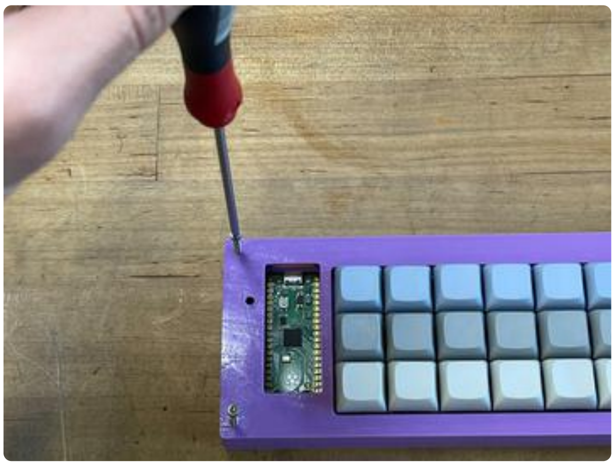
Place the PCB into the base, alining the M3 screws you added earlier to the holes in the base.

Use the nylon nuts to fasten them.

JOHN PARK
2021







3D Case Top

Place the case top onto the bottom -- there are alignment posts to make it fit neatly.

Screw in the four M2.5 x 16mm screws from the top (or bottom, your choice). No nuts are needed, the screws will self tap into the plastic.