Mini VOTE Keyboard

Created by Collin Cunningham

https://learn.adafruit.com/vote-keyboard

Last updated on 2024-06-03 03:10:08 PM EDT

Table of Contents

Overview

This guide will show you how to build a **mechanical keyboard** designed for typing one thing only - “VOTE”! Perfect for responding to heated political debates on social media, or simply reminding friends & strangers of the most important action they can take as citizens of a democratic society. Bonus – anyone named Veto or Tove can use it to type their name as well.

What you’ll need

Parts

6 x 1N4148 Diodes — https://www.adafruit.com/product/1641
Small Signal Diodes for switch matrix

1 x Trinket M0 — https://www.adafruit.com/product/3500
Microcontroller Board for running CircuitPython code

1 x Black Nylon Screw and Stand-off Set — https://www.adafruit.com/product/3299
For raising PCB

1 x Printed Circuit Board

8 x Mechanical Keyswitches, Cherry MX compatible, PCB mount type

5 x 1u Keycaps, Cherry MX compatible (see notes below)

1 x 3u Spacebar Keycap, Cherry MX compatible (see notes below)

Tools

1 x Flush Cutters — https://www.adafruit.com/product/152
For clipping excess leads

1 x Soldering Iron — https://www.adafruit.com/product/180
For making connections

1 x Solder Spool — https://www.adafruit.com/product/1886
Solder Wire - 60/40 Rosin Core

1 x Panavise Jr. - PV-201 — https://www.adafruit.com/product/151
PCB vise for soldering

1 x Micro USB Cable — https://www.adafruit.com/product/2185
For programming Trinket M0

Key switches

The PCB is designed to be used with Cherry MX compatible PCB mount keyswitches (https://adafru.it/19Av) – as opposed to **plate mount** keyswitches. PCB mount switches have two extra **plastic posts** on the bottom that keep the switch from twisting horizontally during installation & while in use. Plate mount style switches can be used instead, but they will exhibit the aforementioned twisting behavior.

Keycaps

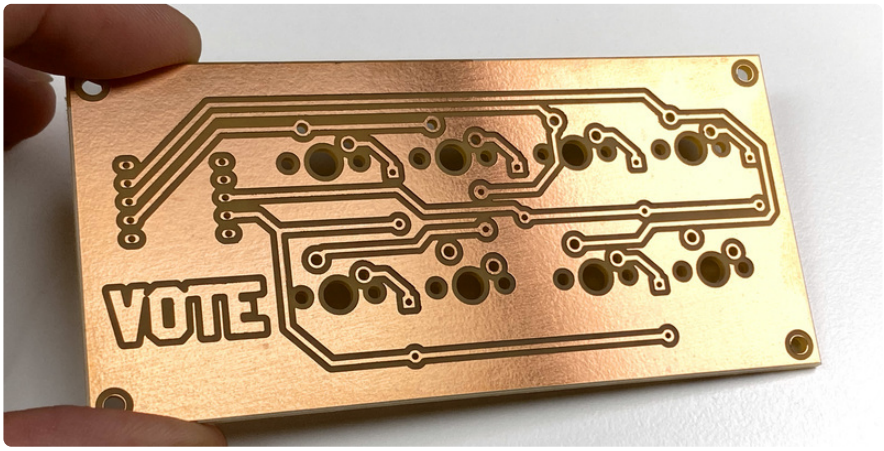
Keycap lengths are measured in units relative to a single character key. So, for example: the 'A' key is denoted as being 1u in length, while the TAB key is 1.5u, and the standard Spacebar is 6.25u

Due to the unusual layout, finding proper keycaps for this project can be a challenge – particularly the 1u Return & 3u spacebar caps.

The **V, O, T, E**, & **Return** caps I used are from a limited-run set called XDA Oblique (https://adafru.it/MpF) and the 3u **spacebar** is from an add-on kit for this set (https://adafru.it/Mqa).

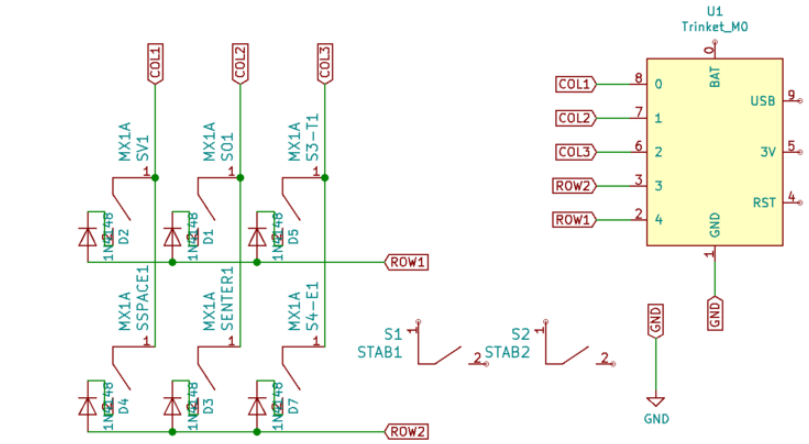
To create something similar without scouring r/mechmarket (https://adafru.it/Mqb), you can pick up the **English Spacebar** & **Base** kits from this set (https://adafru.it/Mqa). Alternatively, you could use caps from this complete set (https://adafru.it/Mqc) which costs a bit more. Whichever keycaps you end up using, just make sure they're **Cherry MX compatible** – which is the most common type.

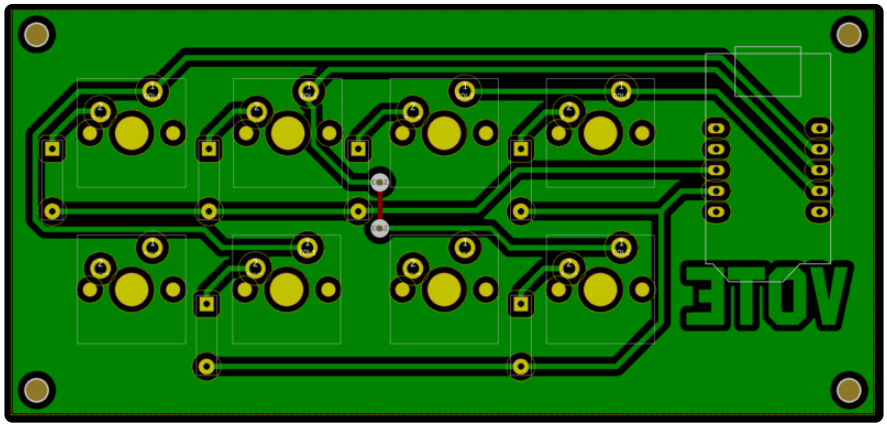
PCB



The PCB for this project was designed in Kicad (https://adafru.it/VtD) with drill sizes and trace widths optimized for a **1/32" flat end mill**. The board you see in the photos was created with an **Othermill,** ancestor of the Bantam Tools PCB Milling Machine (https://adafru.it/Mub).

These files have only been tested with a milling machine and may require changes before before being sent to a PCB fab house for production.





Kicad project

Click the button below to download the Kicad source project with schematic, board, & library files

VOTE_Keyboard-Kicad.zip

https://adafru.it/Muc

Gerber files

Click the button below to download the milling/fabrication files in Gerber format

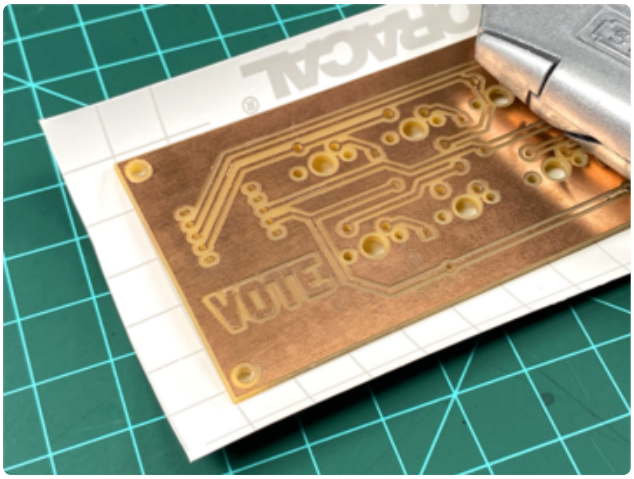
VOTE_Keyboard-gbr.zip

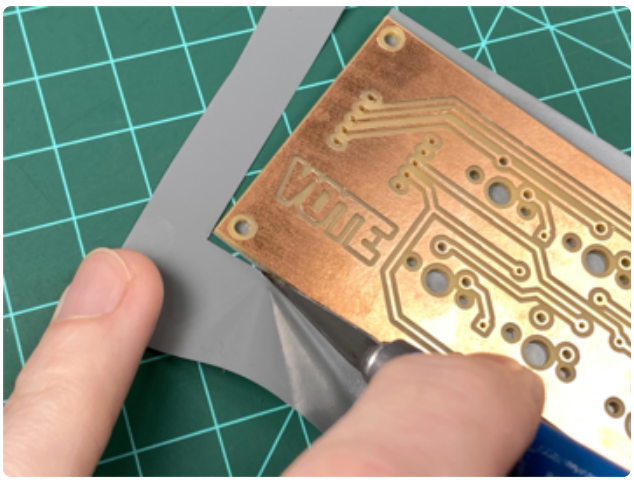
https://adafru.it/Mud

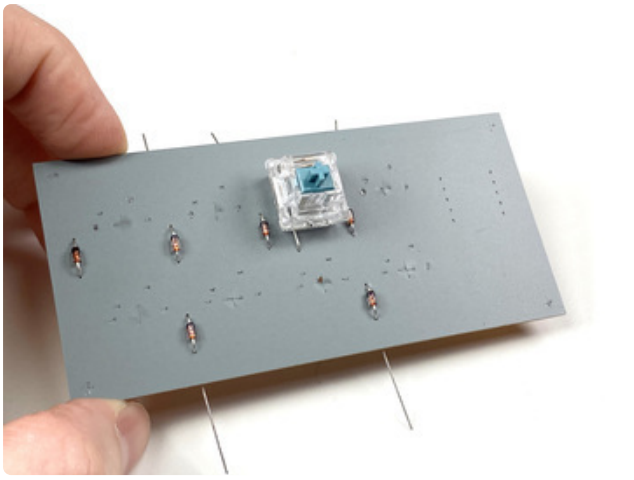
Optional color

Since the PCB will play a major role in this keyboard's aesthetic, you may want to dress it up a bit. You could use paint to cover the top side in a custom color, but it's far easier to use **adhesive vinyl**.

I used Oracal 631 (https://adafru.it/Mue) adhesive vinyl, which is **repositionable** and somewhat forgiving during application – similar to drawer liner or contact paper. Consider using Oracal 651 type (https://adafru.it/Muf) if you want something more **durable & permanent**. Whichever type you choose, the following process should apply:







Cut a piece of vinyl **significantly larger** than the PCB – ~13cm x 8cm

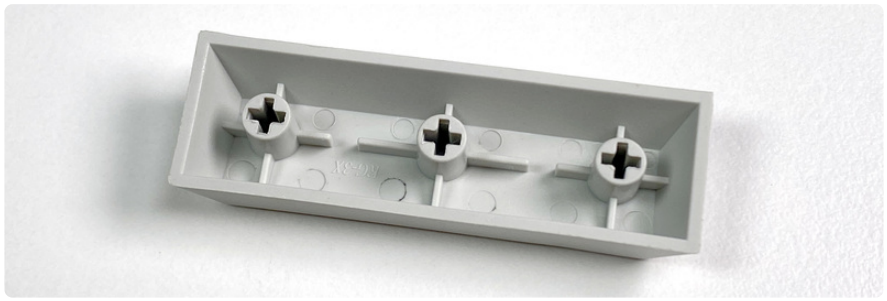
Remove the adhesive backing and apply vinyl to topside of PCB

Turn the covered PCB over and use a sharp knife along the board edges to remove excess vinyl

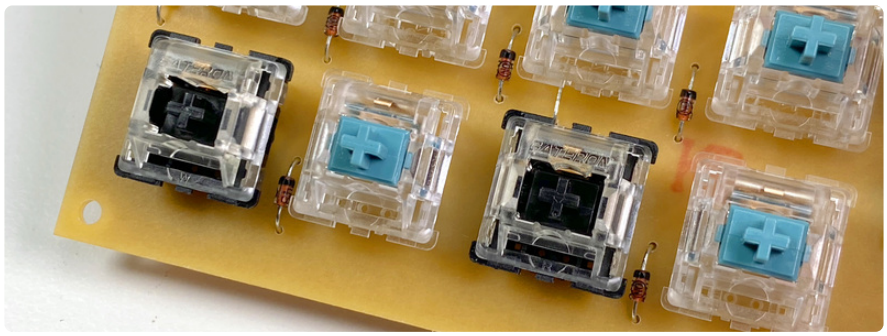
Slice holes through the vinyl for each component lead & use cross-cuts for the larger openings (switch posts, standoffs, etc)

Assembly

A note about spacebar switches

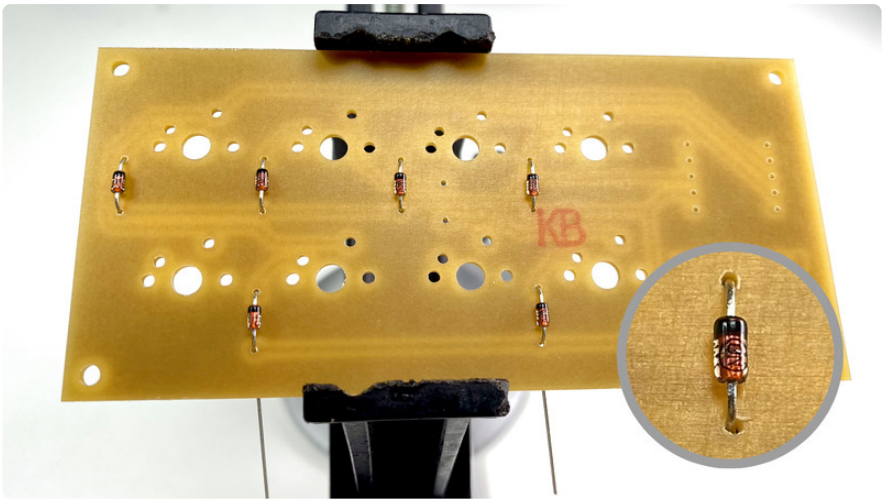


Normally, keycaps with a length of 2u or greater require stabilizers (https://adafru.it/MuA) which would complicate this relatively simple build. The **3u spacebar** we're using can be mounted directly on 3 **individual keyswitches**. This means we get to skip the stabilizers, but it also means the spacebar will be harder to press than the other keys.

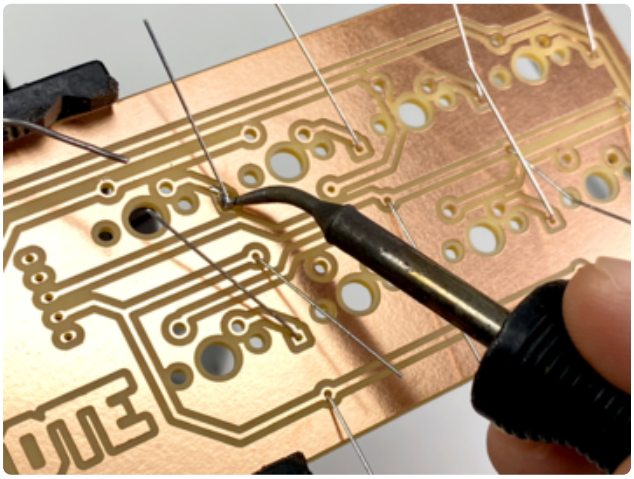


If a harder-to-press spacebar bothers you, you can use a small flathead screwdriver to open the keyswitches (https://adafru.it/MuB) on **either end** of the spacebar and remove their springs before soldering. Doing so will leave only the force of the center switch's spring required for actuation.

Diodes

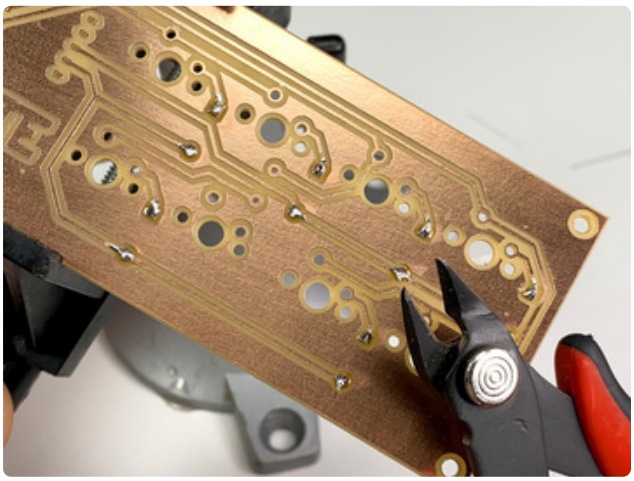


Mount the diodes on the PCB with each diode's **stripe** pointing toward the **upper edge** of the board as seen in the photo above.



Once you've double-checked each diode's orientation, **bend** their leads outward to hold them in place.

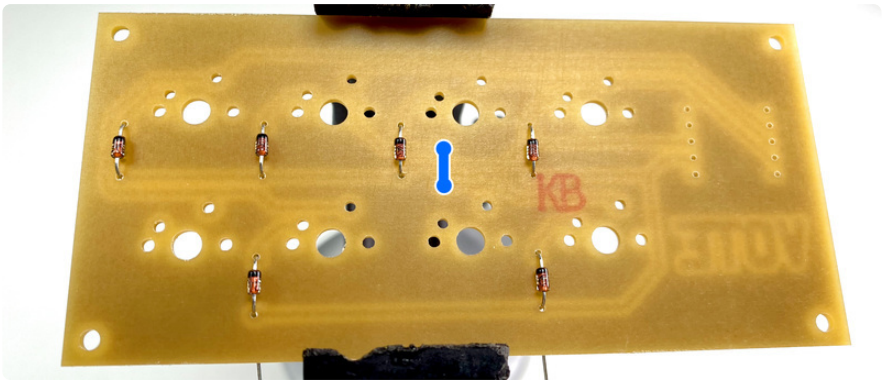
Solder each diode in place.



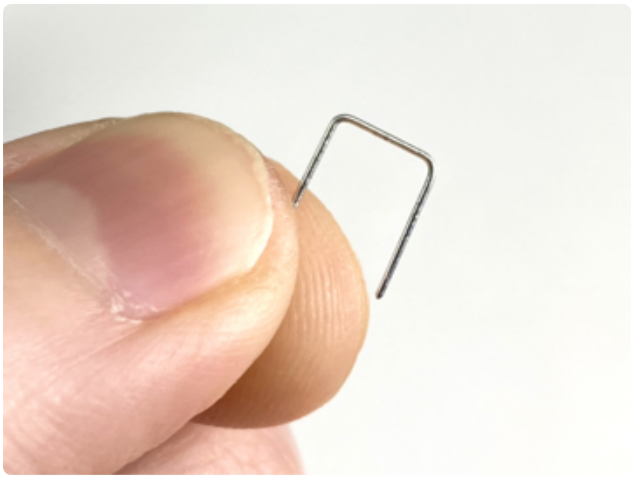
Clip the excess leads off the back of the board.

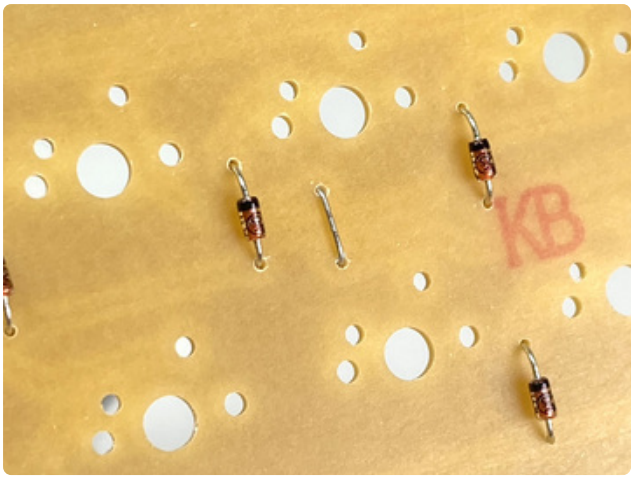
Save one of the clipped leads to use as a jumper in the next step.

Jumper



If you're working with a single-sided PCB, you'll need to install a **single jumper wire** in the location shown above in blue.

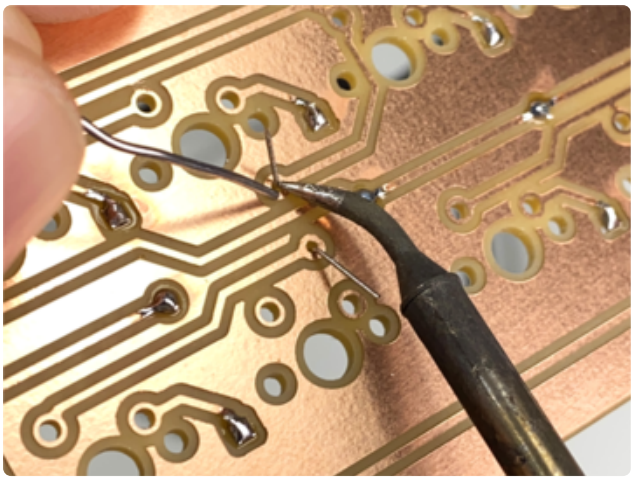


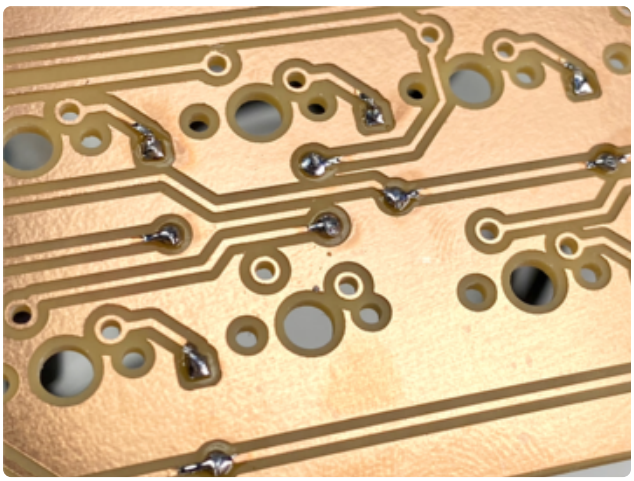


Bend a **clipped diode lead** from the previous step into a ~5mm **U shape**.

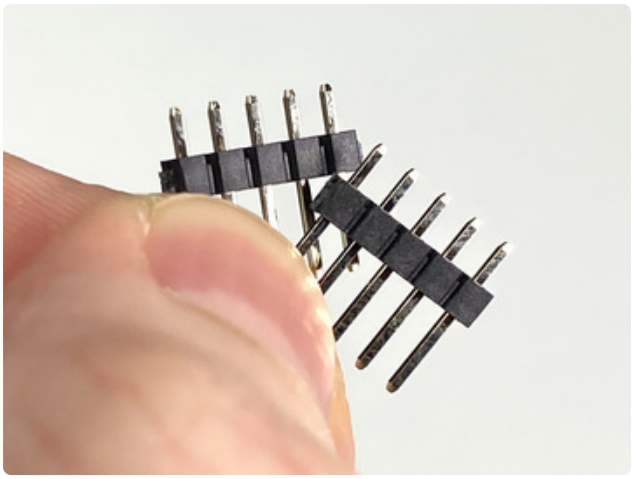
Mount the jumper in the center of the top side of the PCB.

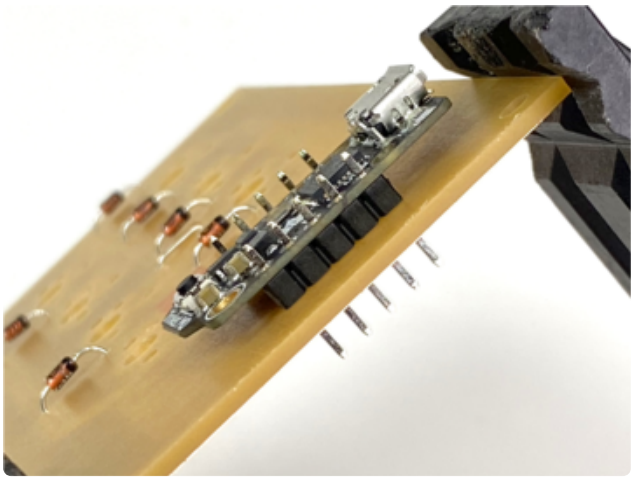
Bend the jumper leads outward, solder, and clip any excess from the board.

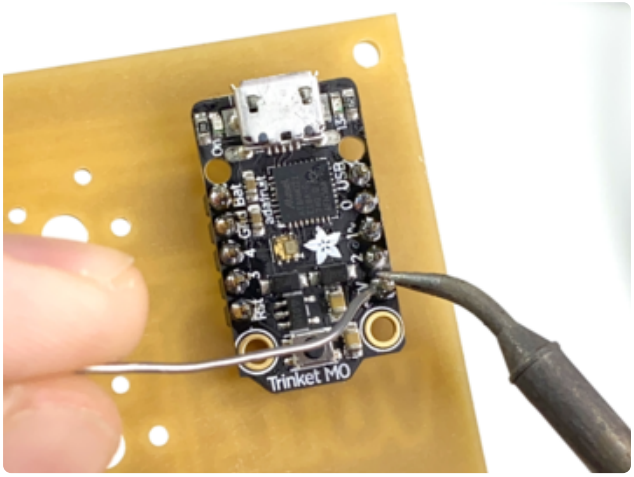




Trinket M0







Use flush cutters to cut a strip of male **header pins** into two **5-pin** length pieces.

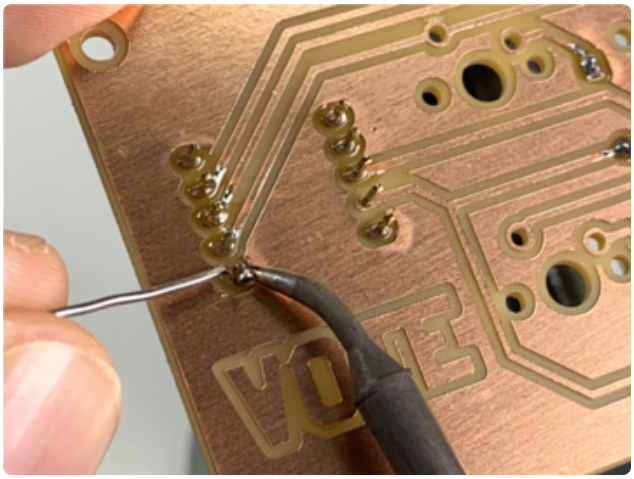
Mount the header pins on the PCB with the **longer pin ends** protruding out the **bottom copper side** of the board.

Mount the Trinket M0 using the pins on the top side of the board as seen in the photo.

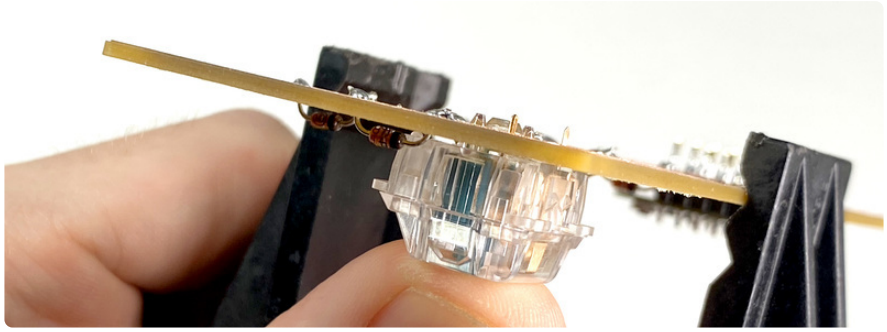
Carefully **solder each pin** to the Trinket M0.

Turn the board over and solder the pins to the bottom side of the board.

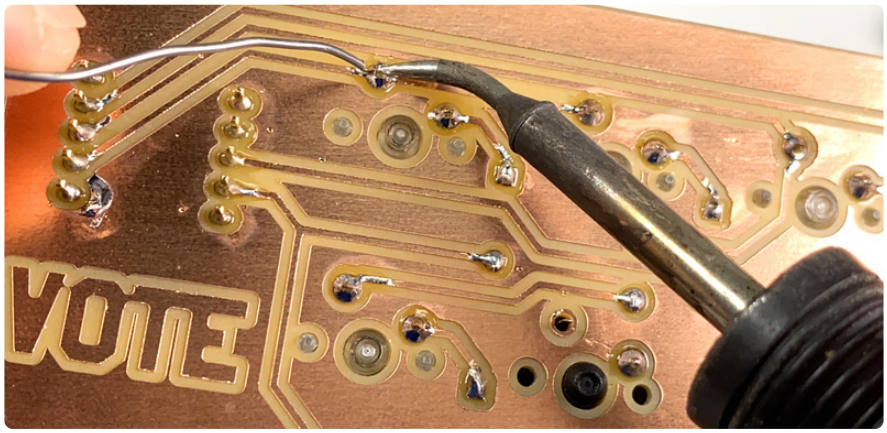
Clip the excess pin lengths on the bottom side of the board.



Keyswitches



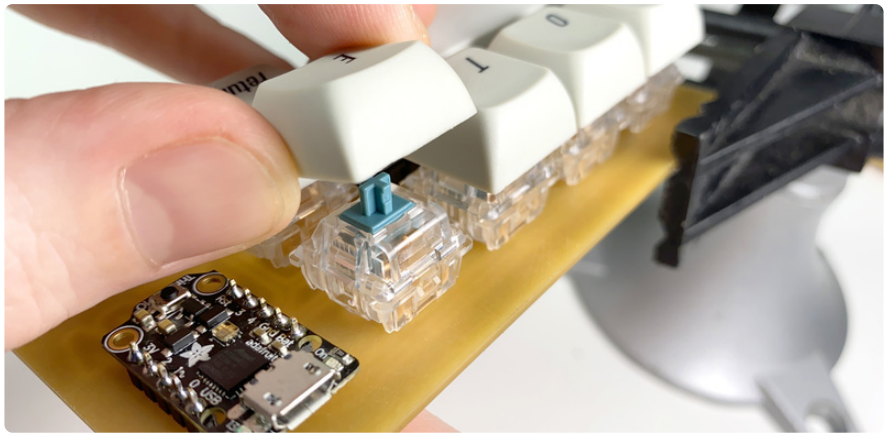
Mount a keyswitch on the PCB, ensuring that all of its pins & plastic posts are poking out the bottom side of the board.



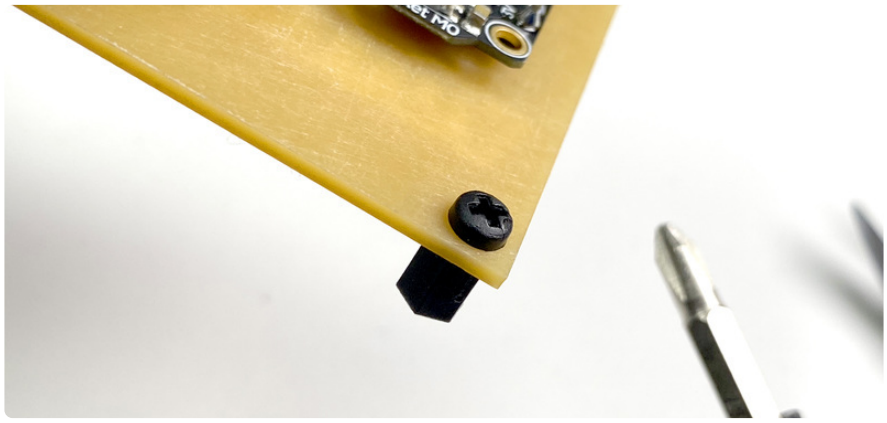
Solder both of the switch's **pins** to the bottom side of the PCB.

Repeat this process for each of the remaining switches.

Finishing up



Once soldering is complete, go ahead and install the **keycaps** by pressing each one firmly onto the switch stems.



Finally, install the **standoffs** at each corner of the PCB using a small Phillips head screwdriver

Software

Trinket M0 comes **preloaded with CircuitPython**. If you've since used it to run Arduino code, or you'd like to upgrade to the latest version, follow the instructions here to install the latest CircuitPython (https://adafru.it/ABS).

Connect Trinket M0 to your computer using a micro USB Cable. A drive named **CIRCUITPY** should appear on your computer.

Open the **CIRCUITPY** drive and create a folder named **lib** inside (if it doesn't already exist).

Libraries

The project code requires two code libraries. **Click the link below** to download the CircuitPython library **bundle** which matches the version of CircuitPython you are running. You can check the **boot_out.txt** file on the **CIRCUITPY** drive to determine the major version of CircuitPython you are using.

Library Bundle

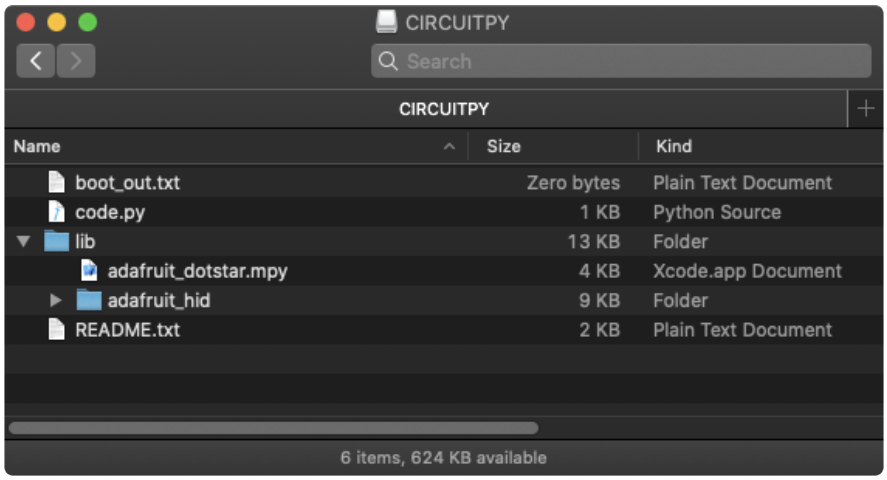
https://adafru.it/ENC

Unzip the library bundle, and open the **lib** folder inside.

You'll need to copy two libraries from this folder to the **CIRCUITPY** drive's **lib** folder:

- Locate the folder named **adafruit_hid** and copy it to the **CIRCUITPY** drive's **lib** folder.
- Locate the file named **adafruit_dotstar.mpy** and copy it to the **CIRCUITPY** drive's **lib** folder.

Your CIRCUITPY drive's file structure should now look like this:



Project code

CircuitPython code for this project was adapted from the rather excellent MiniKbd (https://adafru.it/MuC) by Andy Clymer.

Copy the code below and paste it into a **new text file**.

Save the text file as **code.py** to the root of the **CIRCUITPY** drive, overwriting any preexisting file.

```
# SPDX-FileCopyrightText: 2020 Collin Cunningham for Adafruit Industries
#
# SPDX-License-Identifier: MIT

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.keyboard_layout_us import KeyboardLayoutUS

kbd = Keyboard(usb_hid.devices)
kbdLayout = KeyboardLayoutUS(kbd)
state = []
pins = {}
buttonMap = [
    dict(row="D4", col="D0", id=1),
    dict(row="D4", col="D1", id=2),
    dict(row="D4", col="D2", id=3),
    dict(row="D3", col="D2", id=4),
    dict(row="D3", col="D0", id=5),
    dict(row="D3", col="D1", id=6)]

# Set up row pins
for pin in ["D4", "D3"]:
    p = DigitalInOut(getattr(board, pin))
    p.direction = Direction.OUTPUT
    pins[pin] = p

# Set up column pins
for pin in ["D0", "D1", "D2"]:
    p = DigitalInOut(getattr(board, pin))
    p.direction = Direction.INPUT
```

```
    p.pull = Pull.DOWN
    pins[pin] = p

buttonIDtoKeycode = {
    1: Keycode.V,
    2: Keycode.O,
    3: Keycode.T,
    4: Keycode.E,
    5: Keycode.SPACE,
    6: Keycode.ENTER}

while True:
    # Compare old and new state
    oldState = state
    newState = []
    newBtn = None
    for button in buttonMap:
        r = pins[button["row"]]
        r.value = True
        if pins[button["col"]].value:
            newState += [button["id"]]
            if not button["id"] in oldState:
                newBtn = button["id"]
        r.value = False
    # Press & release keys
    for oldID in oldState:
        if not oldID in newState:
            kbd.release(buttonIDtoKeycode[oldID])
    if newBtn:
        kbd.press(buttonIDtoKeycode[newBtn])
    state = newState
```

Usage & customization

Once you've saved **code.py** to your Trinket M0 the code will start running and the board will be seen by your computer as any other **USB keyboard** – no reboots or special tricks required.

Want to make your keyboard say something other than "vote"? No sweat. You can easily change what keycodes are sent to your computer by editing the code.

Edit **lines 37-42** of **code.py** by swapping the current keycodes with the new ones you want. You can see a list of all possible keycodes using this reference in the CircuitPython documentation (https://adafru.it/MuD).

For example, if you wanted to be able to type "cool" and exchange the return key for a shift key, it would look like this:

```
buttonIDtoKeycode = {
    1: Keycode.C,
    2: Keycode.O,
    3: Keycode.O,
    4: Keycode.L,
    5: Keycode.SPACE,
    6: Keycode.SHIFT}
```