

CONTROLLER - MANUAL

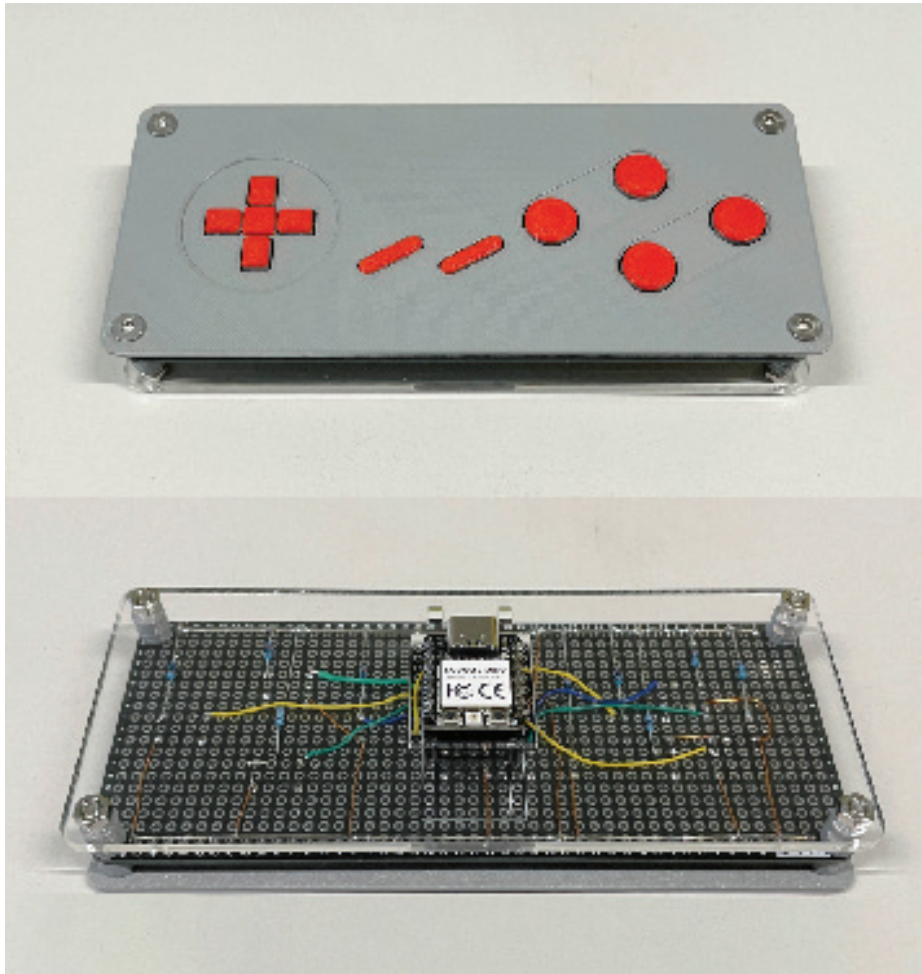
Welcome to this soldering workshop for creating your own USB controller! This controller works with Windows, Mac, and Linux and allows you to play games, including retro games using an emulator.

In this workshop you will learn the basics of perfboard soldering. It's a good basis if you want to start soldering your own electronics projects, for example to include in your Industrial Design projects.

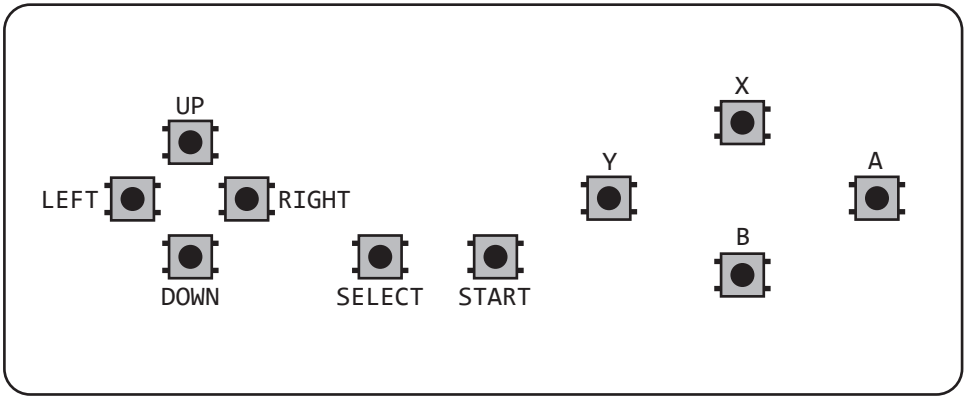
A few handy suggestions:

- Always keep your workspace clean. Place unused bits of wire to the side of your bench so that your working area is always clean.
- Always solder underneath the fume hood!
- When cutting excess wire from components, make sure you hold the excess wire with your non-cutting hand to prevent sharp pieces of wire flying through the air.
- Gently clean the tip of your soldering iron in the brass wire cleaner after every step. Just brush the tip through the brass ball twice to clean it.
- Turn off the soldering iron after completing the soldering in each step. This will prevent the soldering iron tip from becoming damaged.

This workshop is the third in a series of four workshops. These workshops increase in difficulty. If you want to get really good at soldering, you can follow all four of them. They will be taught throughout the year, with a new one every 2 months.

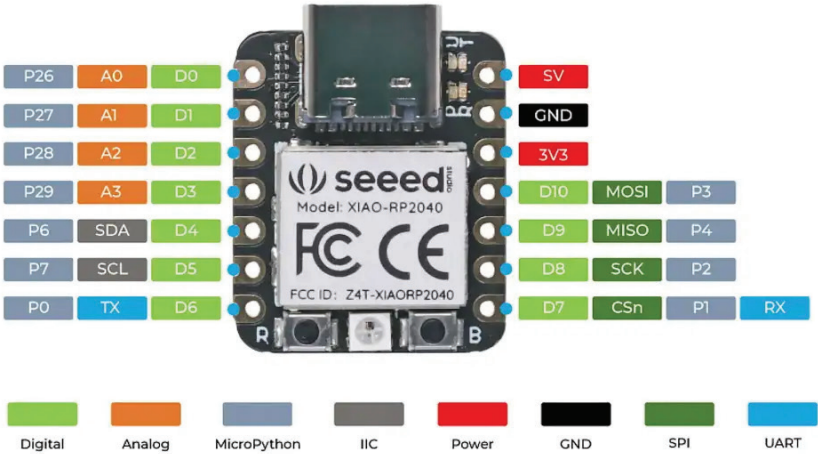


BUTTON OVERVIEW



PINOUT OVERVIEW

XIAO-RP2040 PINOUT DIAGRAM



STEP 1: RAILS

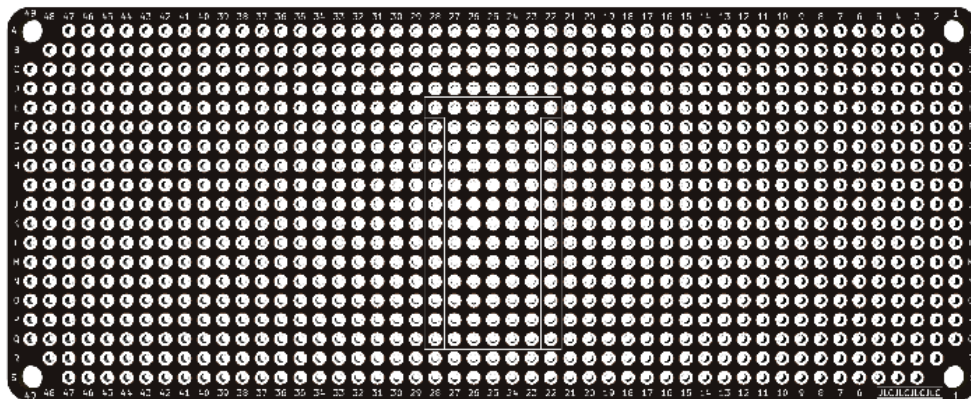
Take the straight tinned copper wires. These wires will become the ground and 3.3V rails.

Solder these wires to the BACK of the perfboard on rows A and S. Make 4 solder joints per rail.

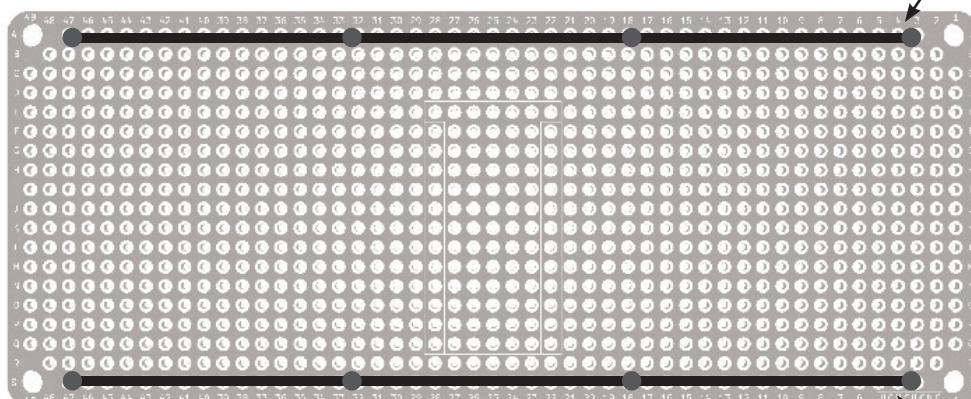
Tip: In your own perfboard projects, it's a good idea to include at least a ground rail if you have multiple components. That way you can easily ground all your components.

2x wire

BACK



BACK



3.3V RAIL

GROUND RAIL

STEP 2: BUTTONS

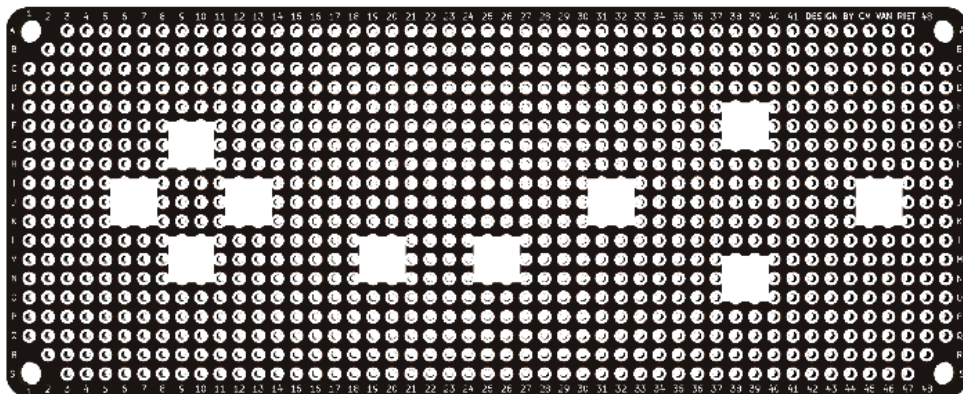
Take the 10 buttons and insert them into the white spaces on the FRONT of the perfboard. Make sure the buttons are oriented horizontally, with the legs protruding on the left and right (not top and bottom!).

Once you're done, check that you now don't see any white squares anymore. Solder the buttons.

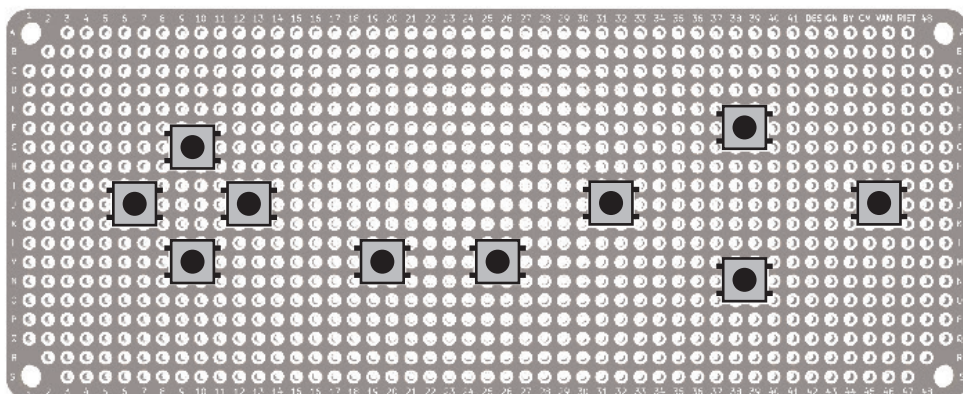
10x pushbutton



FRONT



FRONT



STEP 3: HEADERS

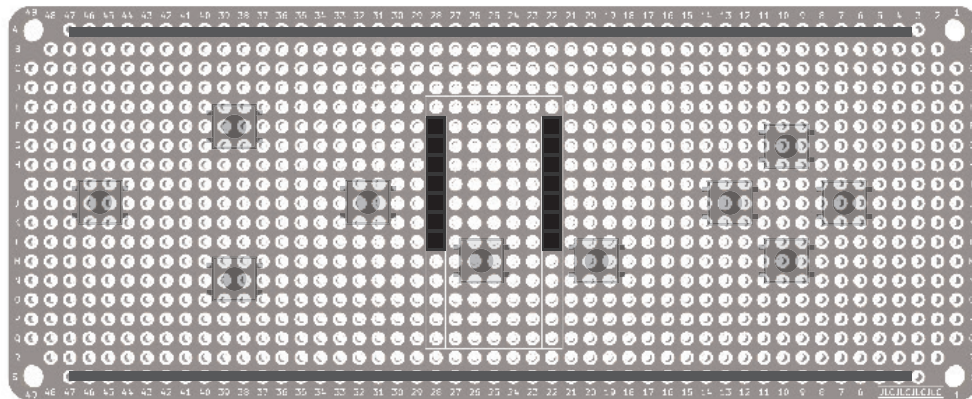
Take the 15-pin header and cut it in half with pliers on pin 8. You will lose one pin, but you will have two headers with 7 pins each.

Solder the two headers to the BACK of the perfboard as shown below.

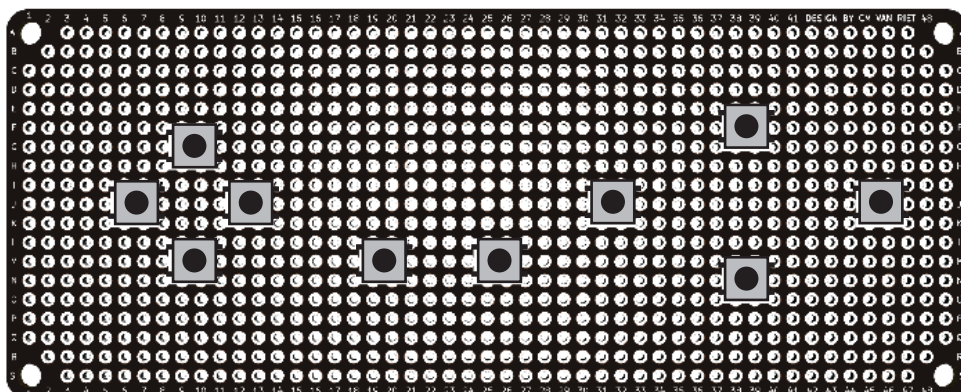
1x header



BACK



FRONT



STEP 4: GROUNDING BUTTONS

Take a black thin, single-core wire and measure it to length from the **bottom** of the button to the ground rail.

Cut the wire and strip the shielding on both ends. On the **BACK** of the perfboard, solder one end of the wire to the button, and the other end to the ground rail. *If it helps, you can use some thin tape to keep the wire in place.*

Repeat 10 times.

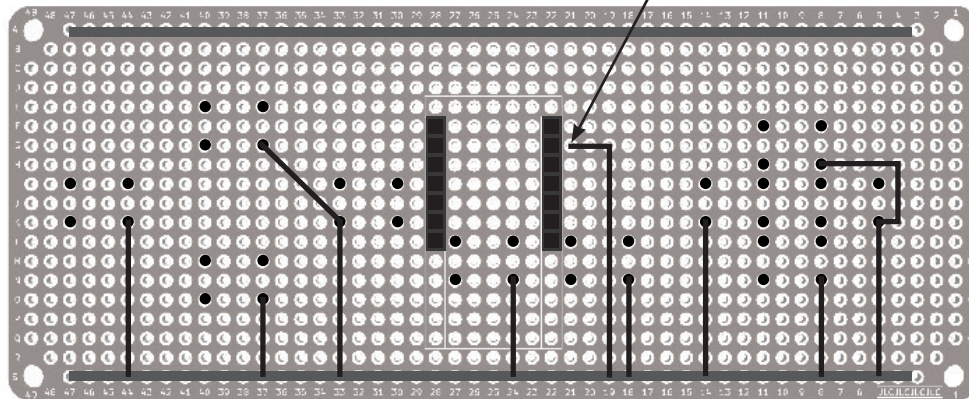
Finally, take a black wire and connect the ground rail to the ground on the microcontroller.

10x black wire



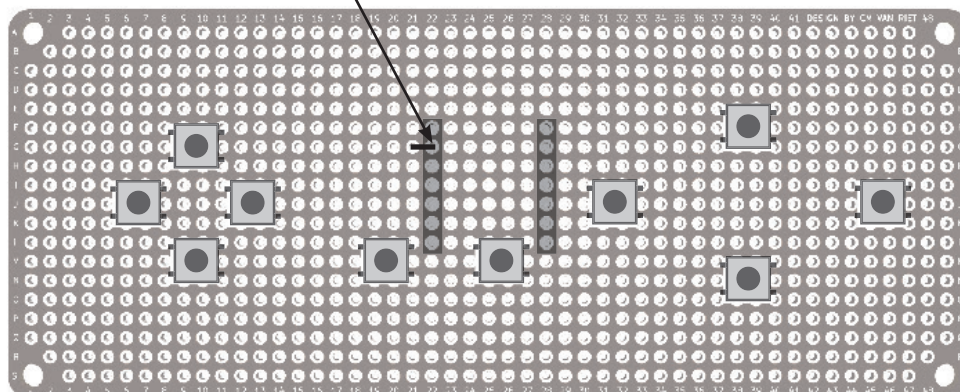
BACK

Wire goes through the perfboard



FRONT

Wire goes to ground on microcontroller

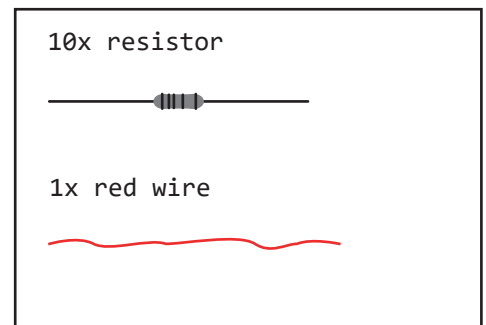


STEP 5: PULL-UP RESISTOR

On the **BACK** of the perfboard, take a resistor and solder it to the **top** of the button and the 3.3V rail. Cut off the ends.

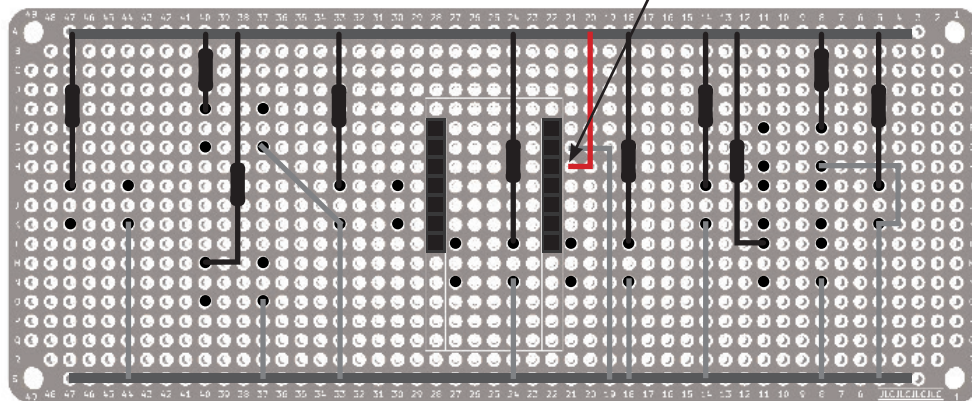
Repeat 10 times.

Take one red wire, strip both ends, and solder it to the 3.3V rail and the 3.3V pin on the microcontroller.



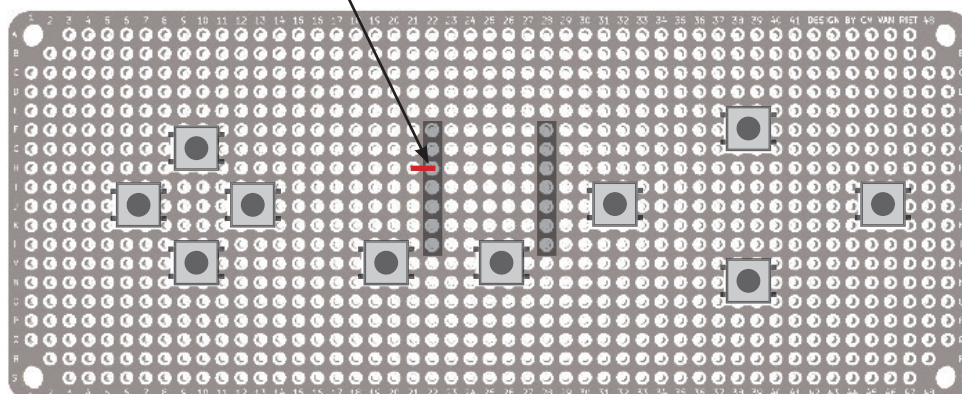
BACK

Wire goes through the perfboard



FRONT

Wire goes to 3.3V on microcontroller



STEP 6: CONNECTING THE BUTTONS

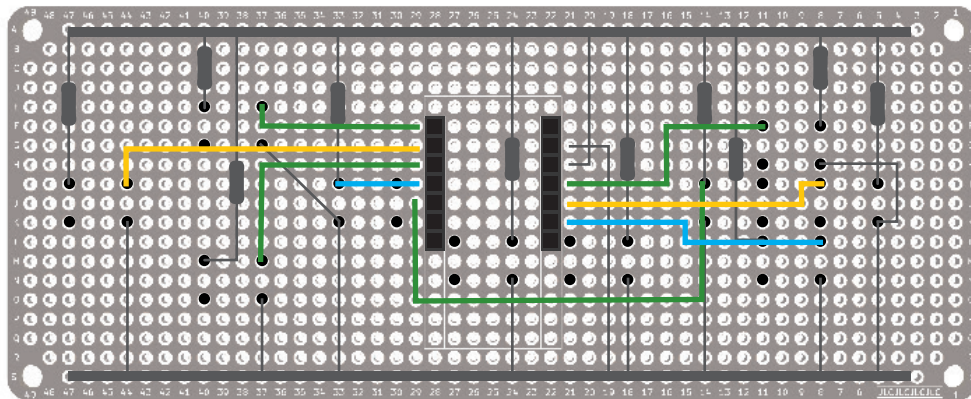
On the **BACK** of the perfboard, using different coloured wires (yellow, green, blue), connect the **tops** of the buttons to the microcontroller pins.

For the **START** and **SELECT** buttons, no wire is needed. The button pins can be soldered directly to pins 7 and 8 of the microcontroller header.

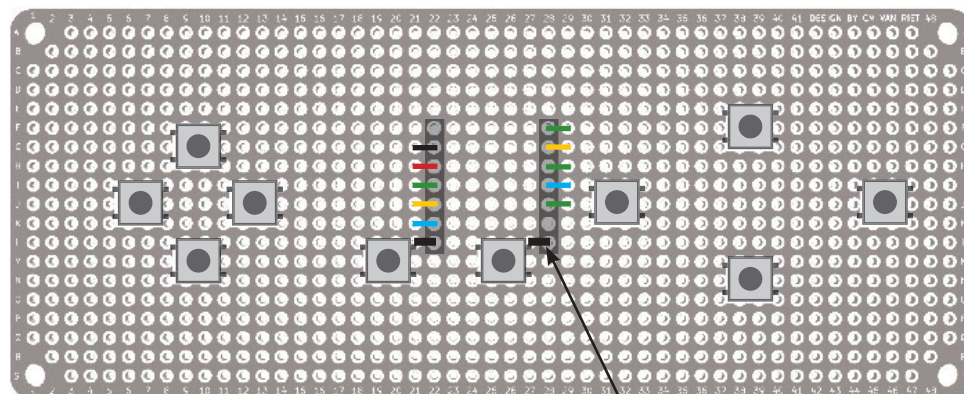
10x coloured wire



BACK



FRONT



Button directly soldered to header

STEP 7: TESTING

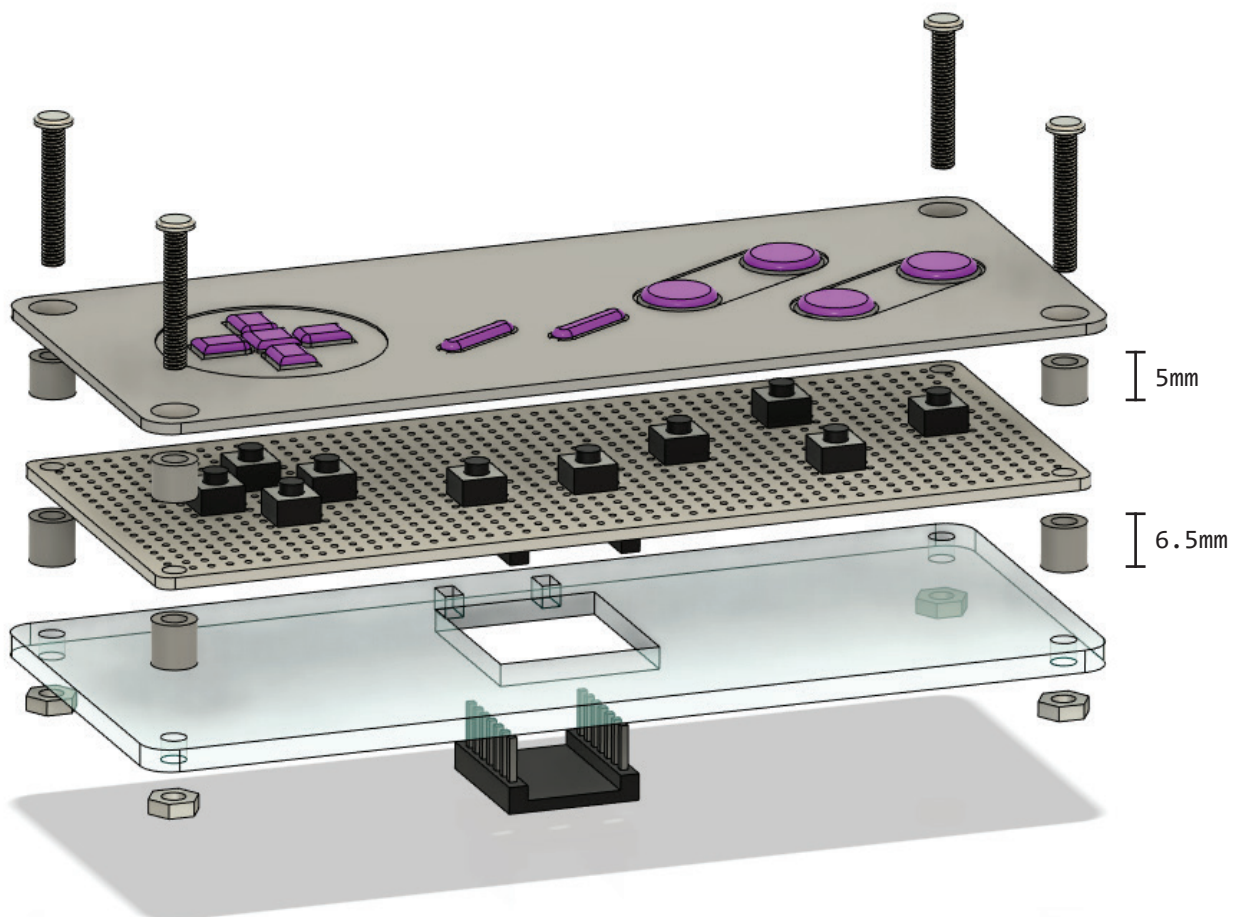
Now is the time to test the connections using a multimeter.

First test the ground connections:


1. Set the multimeter to check for continuity $\bullet)))$
2. While holding one lead of the multimeter to the ground rail, touch all the bottom pins of the buttons on the BACK of the perfboard. Check that you hear a beep at each touch; this tells you that there is a connection. If there's no connection, resolder the wire and check if the connection is restored.
3. Still checking for continuity, insert one lead of the multimeter into the header, and hold the other lead to the other end of the coloured button wire. Check for the beep. If there is no beep, resolder the wire and check for continuity again.
4. Now set the multimeter to measure resistance. You'll be measuring for 10k. Some multimeters set the range automatically, but sometimes you have to set it yourself. In that case, set the range to 20k.
5. Hold one lead of the multimeter to the 3.3V rail and touch all the top pins of the buttons. Check that they all show around 10k in resistance. If a button shows much more (infinite), its resistor is not connected properly. Solder it again and check the connection.

STEP 8: ASSEMBLY

Assemble the controller as pictured below.



STEP 9: UPLOAD THE CODE

Open Thonny, plug in the RP2040, and click the drop-down menu on the bottom right. You should have the option to install **CircuitPython version 7.3.3** on the RP2040. If the RP2040 doesn't show up, on the RP2040 board hold the boot (B) button and press the reset (R) button once. When you're done, the RP2040 should appear as an external drive on your laptop:  **CIRCUITPY (E:)**

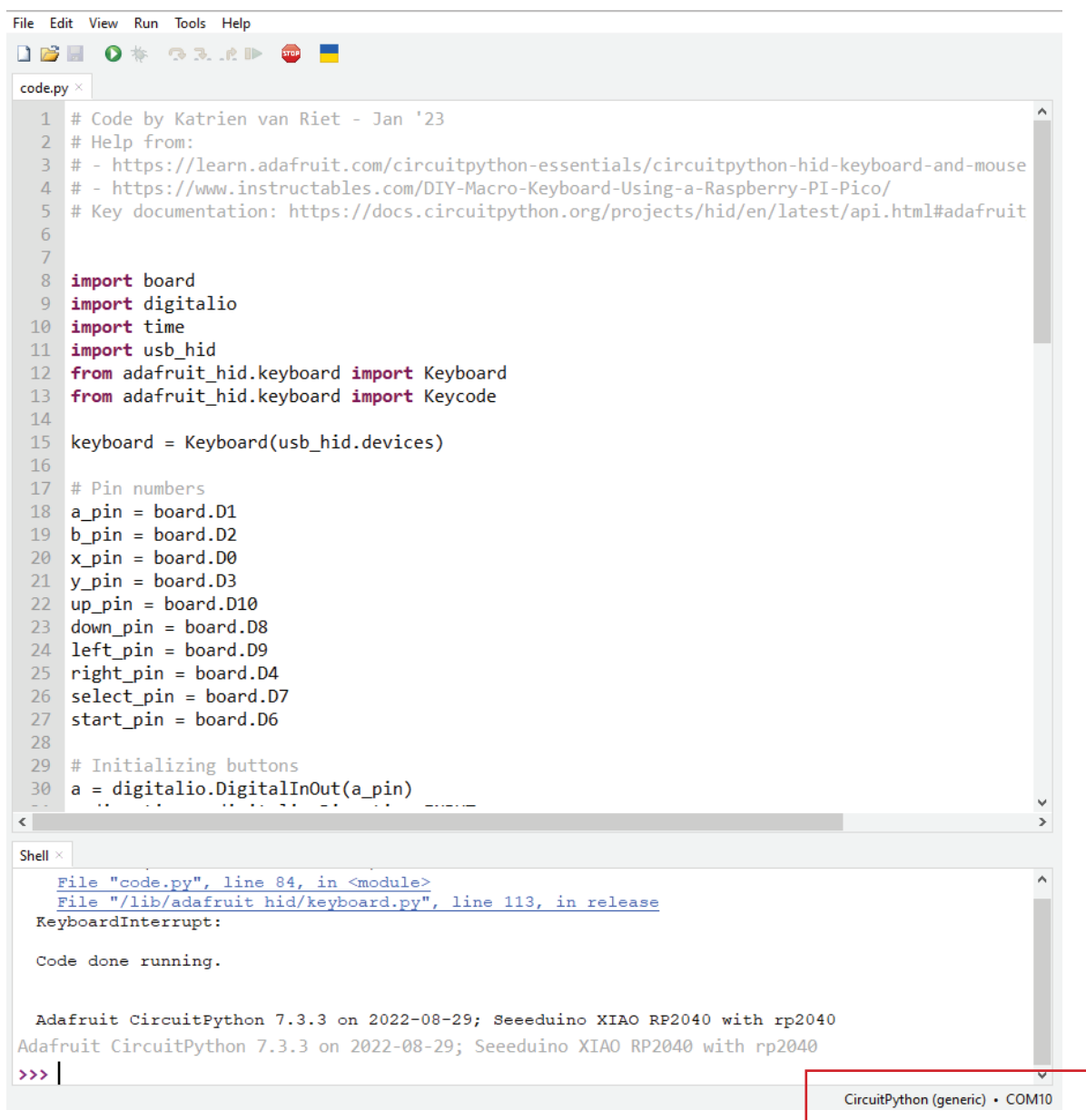
If the RP2040 isn't appearing as a drive, press reset or boot + reset.

Go to the following page:

<https://github.com/kvriet/Soldering-Workshops/tree/main/6%20Controller>

Save the file 'code.py' and the folder 'adafruit_hid' to the CIRCUITPY drive. You're all done!

If you want to make changes to the code, you can open it in Thonny and change the variables, then connect to the RP2040 in Thonny (use the menu on the bottom right), and save the code to the RP2040 as 'code.py'. At the top of the code, some useful websites have been given to help you along with programming in Python.



The screenshot shows the Thonny IDE interface. The top pane displays a Python script named 'code.py' with the following content:

```
1 # Code by Katrien van Riet - Jan '23
2 # Help from:
3 # - https://learn.adafruit.com/circuitpython-essentials/circuitpython-hid-keyboard-and-mouse
4 # - https://www.instructables.com/DIY-Macro-Keyboard-Using-a-Raspberry-PI-Pico/
5 # Key documentation: https://docs.circuitpython.org/projects/hid/en/latest/api.html#adafruit
6
7
8 import board
9 import digitalio
10 import time
11 import usb_hid
12 from adafruit_hid.keyboard import Keyboard
13 from adafruit_hid.keyboard import Keycode
14
15 keyboard = Keyboard(usb_hid.devices)
16
17 # Pin numbers
18 a_pin = board.D1
19 b_pin = board.D2
20 x_pin = board.D0
21 y_pin = board.D3
22 up_pin = board.D10
23 down_pin = board.D8
24 left_pin = board.D9
25 right_pin = board.D4
26 select_pin = board.D7
27 start_pin = board.D6
28
29 # Initializing buttons
30 a = digitalio.DigitalInOut(a_pin)
```

The bottom pane shows the shell output, which includes the following text:

```
File "code.py", line 84, in <module>
File "/lib/adafruit_hid/keyboard.py", line 113, in release
KeyboardInterrupt:

Code done running.

Adafruit CircuitPython 7.3.3 on 2022-08-29; Seeeduino XIAO RP2040 with rp2040
Adafruit CircuitPython 7.3.3 on 2022-08-29; Seeeduino XIAO RP2040 with rp2040
>>> |
```

In the bottom right corner, a red box highlights the text 'CircuitPython (generic) • COM10'.