QMQ 20KeyPad

Assembly Guide

Said Mrini

www.saidmrini.com

Balaguer. 15/08/2024

Index

Introducing the 20Keypad

The 20Keypad is a customisable macropad designed to streamline your workflow and boost efficiency. This compact, 20-key peripheral leverages the power of QMK firmware, allowing you to program each key with macros, application shortcuts, or frequently used functions.

This could come in handy if you have a laptop or a small size keyboard, the 20Keypad minimises context switching and keeps essential commands readily at your fingertips.

This guide will walk you through the entire process of building your own 20Keypad, from gathering the parts to flashing the QMK firmware and customizing your keys. By the end, you'll have a unique and powerful macropad that perfectly fits your needs.

Materials

- List all the components required for the project including:
    - RP2040zero microcontroller board (Or any similar board)
    - 20 Mechanical keyboard switches (select the switch type that best suits your typing preference – tactile, clicky, or linear)
    - Keycaps (ensure compatibility with your chosen switches)
    - Diodes
    - copper wires
    - 3D-printer (Or Order the enclosure from an online provider)

These are the same ones I have used:

- MicroController: rp2040zero.
- Diodes:1N4148.
- 3D files: Printables.
- Keycaps: Black and white keycaps.
- Switches: Red Switches.

Tools

- Soldering iron.
- Wire cutters and strippers
- Pliers (optional)
- Multimeter (optional, for troubleshooting)

Softwares:

- QMK MSYS
- IDE: VS Code, in case you want to customize the keymap.

Required Skills

- The knowledge of what is a command line.
- An idea about QMK MSYS Commands.
- Previous use of Visual Studio (If you are planning to modify the code).
- Soldering skills

Advanced skills

- C programming.
- .json files.
- qmk firmware.

Technical information (Fun Stuff)

The first thing you need to learn is how keyboards work and what a matrix is. I am too lazy to explain it myself, but I am going to leave some links that can help you figure it out.

How a Keyboard Matrix Works | QMK Firmware

This one would be more entertaining because it contains some colourful things called pictures.

Hand-Wiring Guide

Janzon's projects

These could be more fun to watch because they contain more of those colourful things but in a moving form. I asked the passenger by my side and he told me it has a name and they call it Video. Fascinating isn’t it?

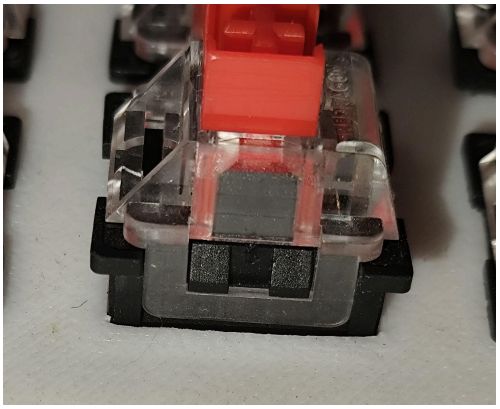
(Don't ask me about the look on his face, I don't know why he is staring at me like that!)

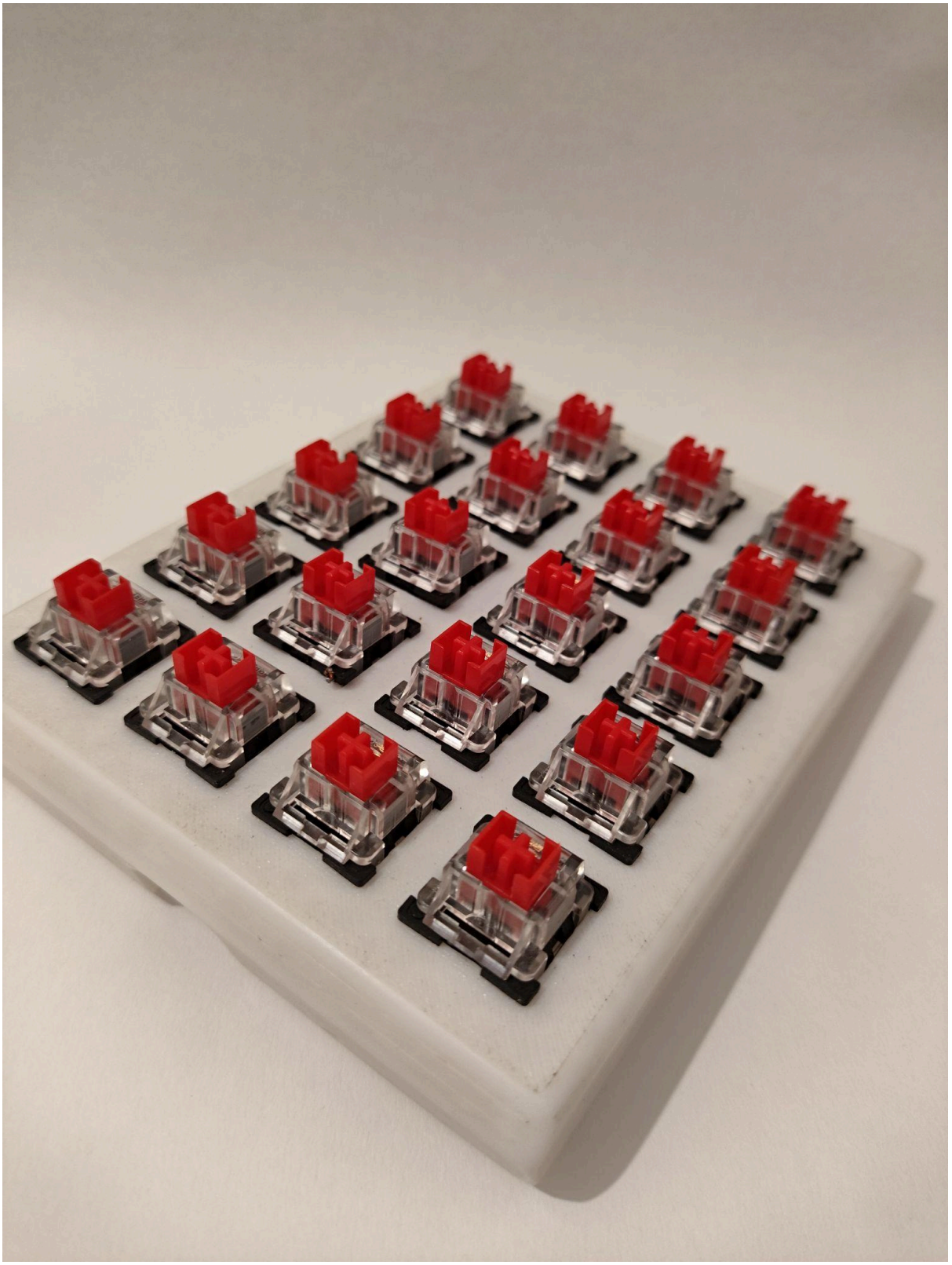
How a Mechanical Keyboard Works (Matrix and Direct Wiring)

How to Build a Handwired Keyboard by Joe Scotto

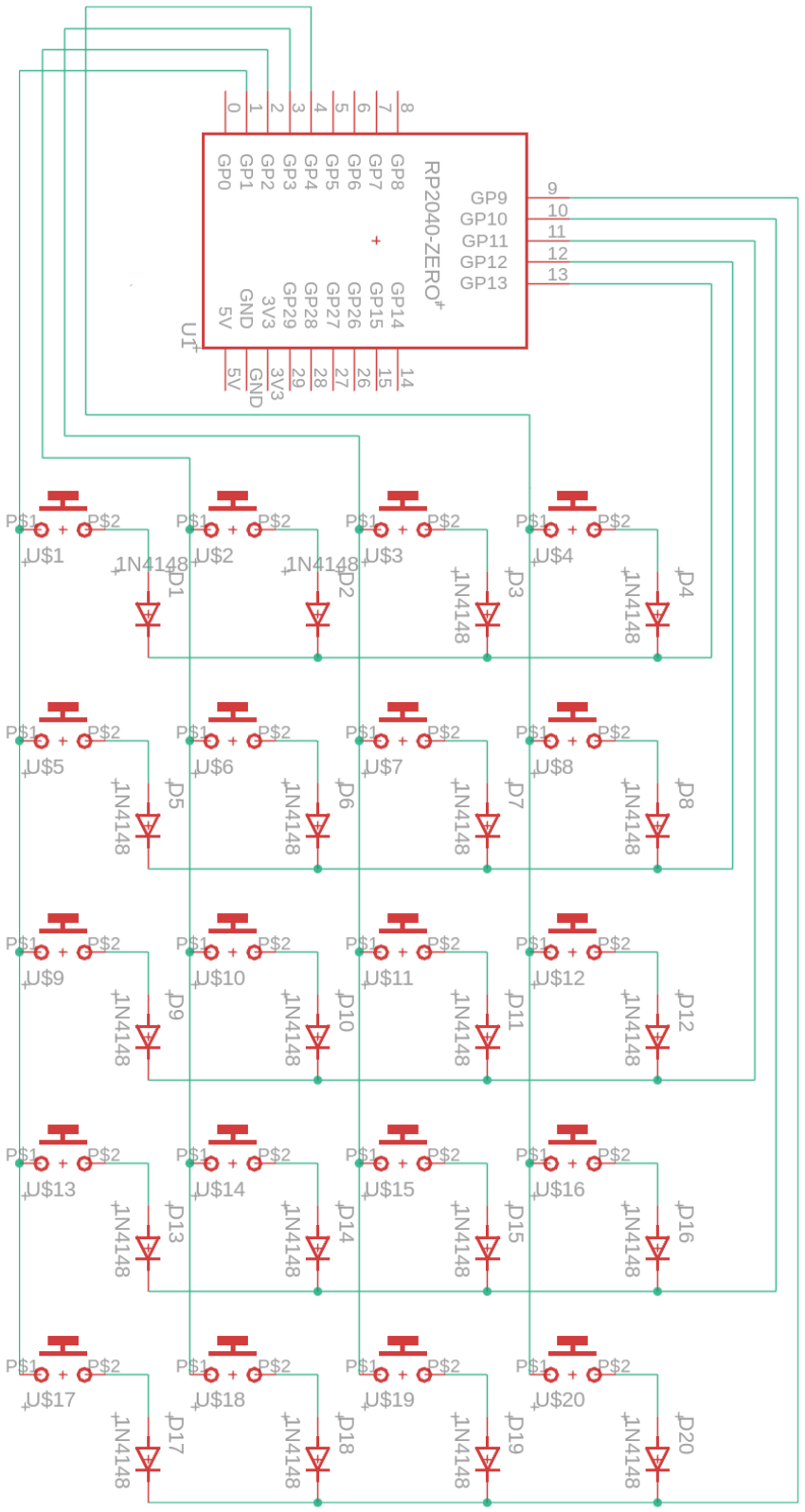
Assembly Guide (The Fun Part)

After you 3D print the enclosure using the files you found on Printables or my GitHub and prepare the rest of the parts, you can start the assembly by placing the keys from the top face of the enclosure.

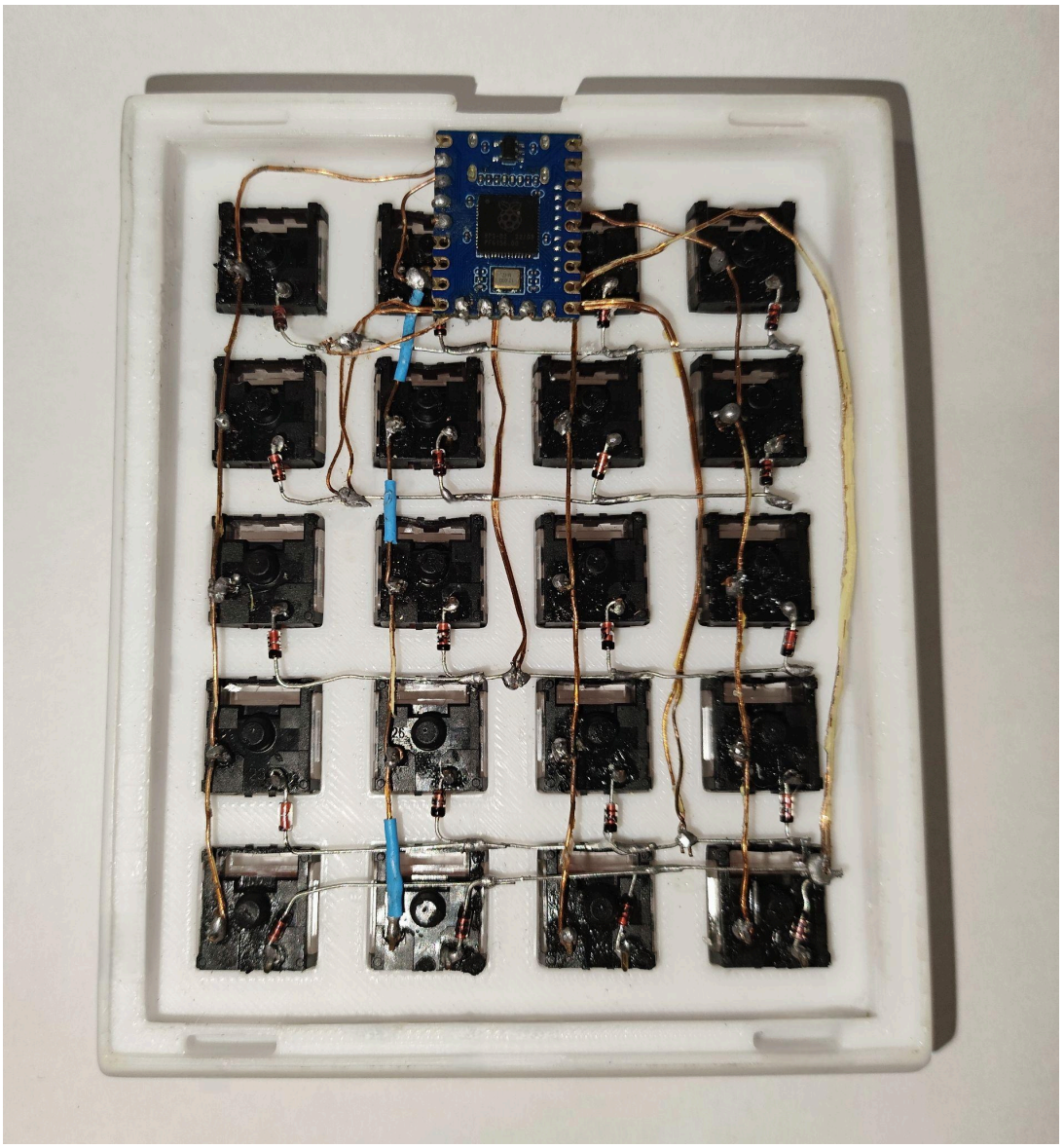




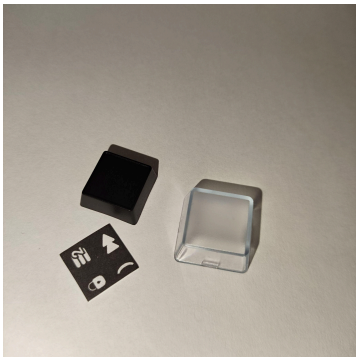
Follow this schematic and connect all the components as they should be.



When you finish you should get something like this or better because mine looks like garbage.

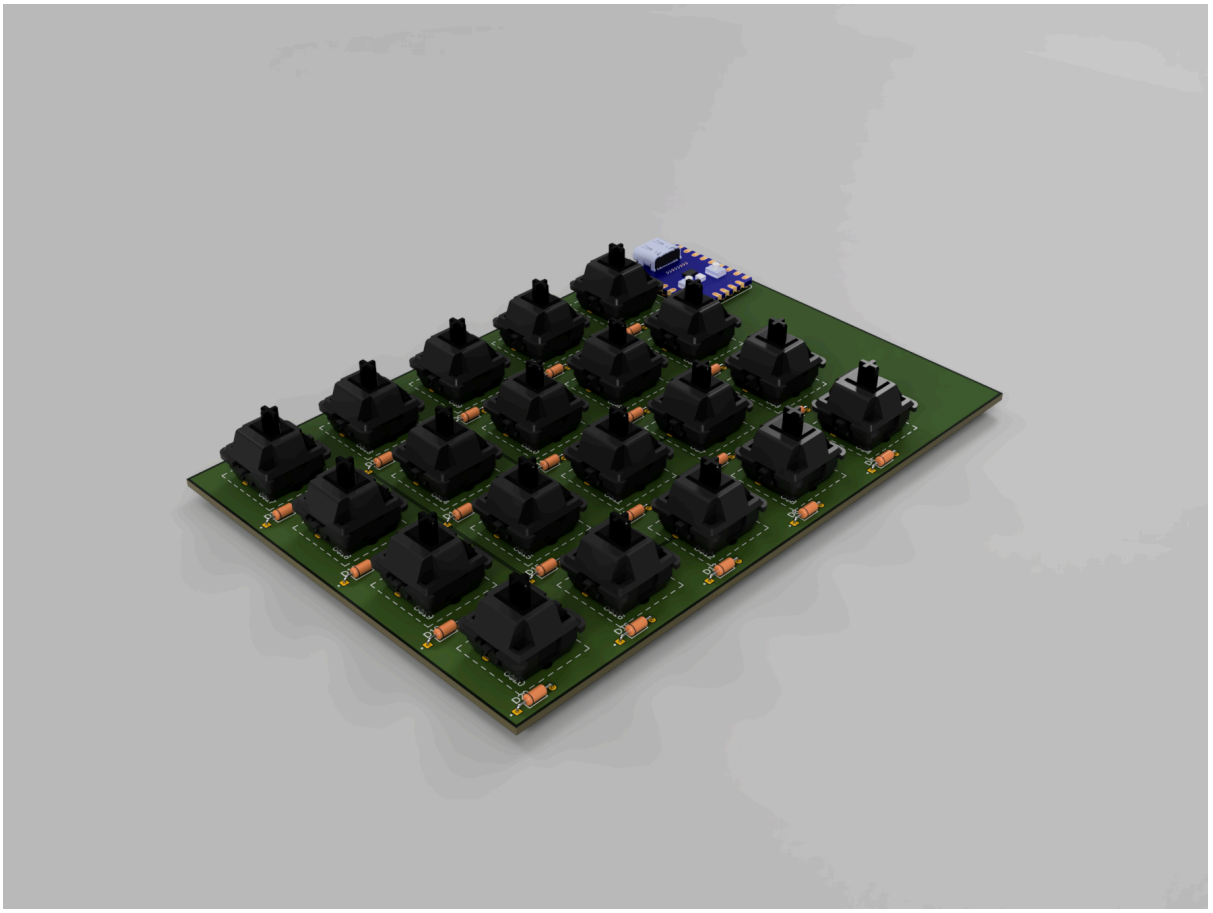


Now Print the labels, that you also found on my Github, cut them to size and put them between the keycap layers.



Put everything together, close it up, and you are done 👌.

If you are wondering why I didn’t use a PCB, it’s going to look like this:



The microcontroller cant be placed on the underside of the PCB and it will have to be placed in a separate place which going to use more space in ower design for nothing.

Also, Handwiring components and soldering are fun, just don’t burn your selfe😊.

Preparing the firmware

1. First, go to the QMK Msys Website, download QMK_MSYS.exe and install it from here.
2. Look for it on the Windows menu and launch it

3. Then you will have to clone the qmk_firmware from my GitHub.

```
  git clone --depth 1
https://github.com/smrini/qmk_firmware.git
"C:\dev\IDEs\qmk_firmware"
```

You can change the Directory `"C:\dev\IDEs\qmk_firmware"` or leave it as it is, You just have to make sure that it ends up with qmk_firmware, and there is no spaces or special characters on the path.

These steps could take some time, 5-15 minutes.

4. When it finishes run this command:

```
qmk doctor
```

If it prompts you to answer Yes or No, just write y and heat enter.

5. Now go to the directory where you installed qmk-firmware, and check that all the needed files are in this directory. if you used the same directory as me, you should find them here.

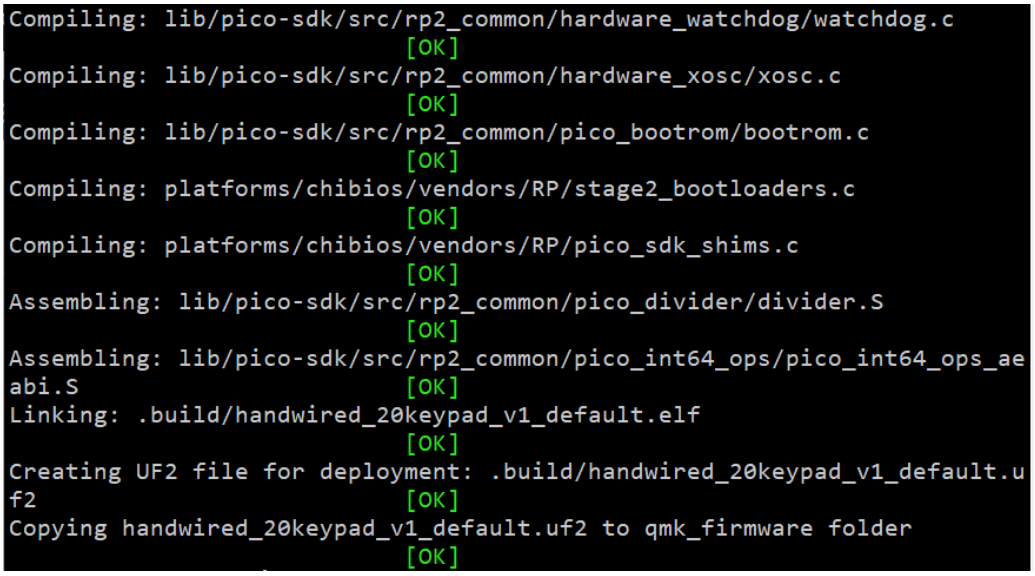
`C:\dev\IDEs\qmk_firmware\keyboards\20keypad\v1`

```
└─────v1
    │   keyboard.json
    │   readme.md
    └─────keymaps
        └─────default
                keymap.c
```

6. now to test everything, run this command:

```
qmk compile -kb 20keypad/v2 -km default
```

If everything is compiled correctly you should see something like this on the CLI



Flashing the Firmware

Method 1:

Flashing the Firmware if you are using the same microcontroller and the same pinout.

1. Put the RP2040 microcontroller board into bootloader mode
    i. Physical reset button: there are 2 buttons on the back of the PCB - press and hold the Boot button, don't leave it yet, press the reset button one time, after you leave it, remove your hand from the boot button.
2. Open the QMK MSYS and write this command

    ```
    qmk flash -kb 20keypad/v1 -km default
    ```

    Where *20keypad/v1* is the directory where our firmware files exist.

3. That's it!

Method 2:

Flashing the Firmware if you are using a different microcontroller, a different pinout, or in case you want to customize the firmware.

1. Go to this Directory.

    ```
    C:\dev\IDEs\qmk_firmware\keyboards\20keypad\v1
    ```

2. Open the keyboard.json file in VS-Code and modify what you want, remember to change this part for the microcontroller and pinout you are using.

    ```
    "matrix_pins": {
        "cols": ["GP4", "GP3", "GP2", "GP1"],
        "rows": ["GP9", "GP10", "GP11", "GP12", "GP13"]
    },
    ```

```
    "diode_direction": "COL2ROW",
    "processor": "RP2040",
    "bootloader": "rp2040",
```

3. Compile the firmware using the QMK MSYS running this command

```
qmk compile -kb 20keypad/v2 -km default
```

5. Put the RP2040 microcontroller board into bootloader mode
    a. There are 2 buttons on the back of the PCB - press and hold the Boot button, don't leave it yet, press the reset button one time, after you leave it, remove your hand from the boot button.
6. Open the QMK MSYS and write this command

```
qmk flash -kb 20keypad/v1 -km default
```

   Where *20keypad/v1* is the directory where our firmware files exist.

7. That's it!

Testing

- Once flashed, connect the macropad to your computer using a USB-C cable.
- Use a keyboard tester to verify that all the keys are working correctly.

Bonus Tip:

After the first successful compilation, you won't need to open the macropad to press the buttons on the microcontroller, you just have to go to layer number 2 by holding the 3rd key from left on the first row, and then press the 4th key on the same row

Useful Resources

First, you will have to install QMK MSYS, you can find it on this website:
HERE

to set the environment, you can follow this guide:
HERE

to build your first project you can follow this guide:
HERE

to load the firmware to your keyboard, you can use QMK ToolBox, which you can download from this link
HERE

or you can flush it using the Command line:
HERE

to configure VS Code for QMK, you can check this link:
HERE

key-codes list:
HERE

A blog that may contain some useful things:
HERE
Space Cadet:
"tap Right Shift on its own and you get the closing one. When held, the Shift keys function as normal."
HERE

Tap Dance:
"Doing different things depends on how many times you taped the key, once, twice, three times, or more..."
HERE

more about .jason file:
HERE

Full list of keycodes used in QMK for Keymaps:
HERE

More about the Keymap.c file:
HERE

RGB Light:
HERE

Instructions for the make command
HERE

Extra Tips

- You have done it, you just successfully built your own custom macropad!
- If you want to get further into it and customize the Keypad more, you can read the qmk documentation.
- Using the same Microcontroller, diode direction, and pinout would make life a lot easier, the other parts won't make any real difference.
- you still can use a different microcontroller, or customize the files, but I don’t recommend that unless you are a pro.
- Try using different switches or keycaps to suit your preferences.
- The links I have provided for buying the parts, aren't an affiliate link and are just an example, so you may be able to find cheaper ones.
- I am really glad you have made it here, and I just want to thank you and say: Few are the people that may come to this point, and make their custom keyboard, but just the idea of making something from the ground means that you are trying to make a difference.

  I am waiting for it 🙃.

Just another thing before you go. you may want to check my profiles from time to time, you could find a project that you may enjoy.

My Personal Website, Behance, Printables…