Creating a Complete Custom Keyboard from Scratch.

A Documentation of my first Split Keyboard: pengo-keyboard

Also look how you want to word things and use page breaks.

Elias Glauert

24 September 2024

Table of Contents

Introduction

This paper is about my journey of creating a custom-made keyboard from complete scratch. It all started after I watched a video[1] made by Christian Selig, a video where he himself went about creating one of these custom boards. He is the person that I got inspired from, so do feel free to watch his video and interact with his page. It gives you a great overview of the scope this project has.

I think a lot of people that stumble across a paper like this have had contact with mechanical or non-stock keyboards, maybe even thought something along the lines of „There has to be something else out there that fits me better “. Keyboards are expensive, especially the high-end or custom ones. The price is something that should be made clear early, so if you do not want to invest a few hundred euros or dollars you might not want to follow along the path that I and so many others took.

If you are already informed about the topic of keyboards you might know about what kind of switches, keycaps and mods to customize keyboards exist out there. Some might even know about Corne or 40% Keyboards, which are a topic that I will not cover in detail. I would say that I felt inspired by the Corne variations, just like Cristian Selig was, but for me the forced layout of the keys all stock Corne boards have seemed a bit off for my hands, so I went about following in Cristian's footsteps and made a board from complete scratch.

The paper you are reading is written to document my journey and mark down the findings I made along the way, as a lot of work goes into researching, designing and building, especially for a first timer in this kind of project field. The content section should be a good overview on the general structure of the paper and as this can be read like a guide you should feel free to jump to the parts you find most important and skip those you already know. You can do what you want with this text: share it, build upon it, that does not matter to me, if you give me credit and try to spread the knowledge that are custom keyboards.

If you need a certain source that I used when creating this paper, you can look at references at the end. They include all the material referenced in the footnotes.

Goals and Specifications for the Project

Important Notice

The goal of creating a board from scratch is to have a personalized and enjoyable experience when using it, which explains why I am explicitly stating that these exact

[1] You can his YouTube video here: https://www.youtube.com/watch?v=7UXsD7nSfDY (Last accessed 22 September 2024), a blog about his project is linked in the description (Under the tag “Parts List”).

choices work best for me and do not have to reflect your choices. Each person has different expectations, wishes and most importantly hands and fingers, so please do not just follow and copy my exact steps if you wish to also create a keyboard from scratch without thinking. Really try to question my way of doing things and try to create your own personal keyboard, maybe even improve my ideas.

What are my Goals and Specifications?

I wanted to create a keyboard that makes typing as easy and simple as possible, as the keyboard of the Mac that I started learning typing with 10 fingers on felt weird and unoptimized. My right hand moved too much when using the backspace, messing up my positioning, my thumbs were just sitting there, doing nothing compared to all my other fingers and they still tried pressing the keys that other fingers were supposed to press. All these things were issues I tried to fix with this project.

List of all the Goals I have for this project

1) The board fits my hands, so I don't have to move them when typing
    a) Ortholinear layout: Normal keyboards have their rows moved a little bit, a leftover from the times typewriters were the norm, as back then each individual key was connected via type bars, that needed to be spaced apart, as otherwise the mechanics would break. It is easier for fingers to move up and down than side to side, hence I want to use an ortholinear layout, where all the keys are in a straight line beneath each other.[2]
    b) Column staggered: Adjusted key column positions that are based on the different lengths of the fingers that are assigned to them to minimize strain and travel length.
    c) Less keys, as I would rather use different layers[3]. Thumbs have more functionality when adding this, as they are used to switch between layers and hands don't move optimally, as you just move up or down a layer and press the key same keys as before with a different outcome.
2) Layout can be toggled between Windows and MacOS
    a) Even though I originally only wanted to make a Windows keyboard, as I use MacOS layouts for work and I want to use it for it I need a flexible keymap.
3) The board is split into two parts.

[2] A visual comparrision can be found here: https://tech-fairy.com/staggered-vs-ortholinear-keyboard-what-are-the-differences/ (Last accessed 22 September 2024).
[3] A layer describes the outcome of a keypress. The base layer gives back an "c" when the C-Key is pressed, but when pressing shift the layer is moved to instead give back a "C".

   a) This gives me a more ergonomic experience, as I can adjust the position of both sides to fit my current need, being able to move them around on my desk.

4) Wireless connection (Bluetooth)

   a) I like the aesthetic of this, and it does not bind me to having both side close to each other or the computer they are connected to.

   b) The cable I would have to use otherwise will also not be in the way of other things on my desk.

5) No backlight, like all my keyboards have had until now, as it slumps battery and should not be needed.

   a) You should now where all the keys are without looking in the optimal situation.

   b) If you make the keyboard wired, you can include backlighting as battery life is of no concern for you.

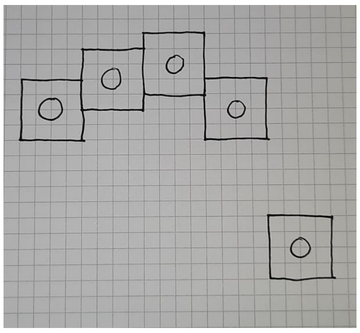
6) Silent switches, as being able to not annoy others around you is a plus in my eyes.

7) Overall, the design should be simple, sleek and clean and fit my setup/room style- and color-wise

Designing the Keyboard Layout based on my Hands and the Components

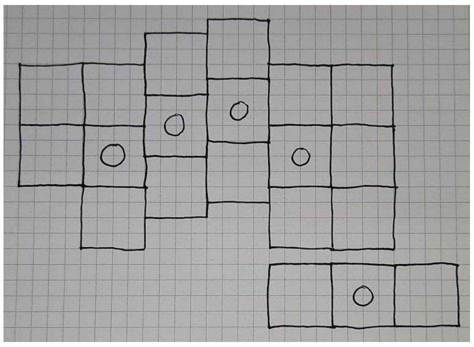
Getting the Layout right and personalized

To implement points number 1 and 3 from the goals I first started by putting my fingers on a 5mm checkered sheet of paper in the way wanted to place my fingers on the keyboard and drew a 2 * 2 square around them, as that is pretty much the real size of a key[4]. These 5 positions are the base for all the keys in the home row.



[4] The real size of a keycap is around 18mm * 18mm and with the padding around the keycap you have a size of 19.05mm * 19.05mm. This is pretty much the same as 20mm * 20mm that a 2 * 2 square has.

From here you can add the movement range of each of your fingers, but that is not really that necessary, except the reasoning on why I did not include an outer bottom key for the pinkie finger. Speaking of this I then added the rest of the keys inspired by Corne Keyboards. This is done not only for the left hand, also known as the center side[5], but also for the right hand, which is called the peripheral side[6].



Ergogen – The Software used to create the Layout

A little Introduction to Ergogen

To get the first rough sketch that was drawn to a format that can be worked with in the rest of the process I used Ergogen[7], a free open-source software that helps with generating files used for custom keyboard building. Ergogen works by you describing what the keyboard will look like, and it does the rest for you. You can do a lot more than just creating the layout in Ergogen, but that will be covered in the later points in the paper when the topic is more relevant. This is also not a full tutorial, as I would not call myself an expert, please read the documentation that is provided on the website or watch some tutorials on YouTube if what I am doing does not match your scope.

How to describe the Layout to look like we want it to

Like mentioned above this is only about getting the key positions right. For this we start with defining the rows and columns we will have in the final product, as Ergogen is based on a matrix system. I defined my columns as these: `outer`, `pinky`, `ring`, `middle`, `index` and `inner` and my rows as these: `bottom`, `home` and `top`.

```
points:
  zones:
    matrix:
      columns:
        outer:
```

[5] The centre side is what sends the signals received by itself and the other side to the computer, which is why it consumes more energy and has to always be connected if you want to use the keyboard.
[6] The peripheral side does not have to be used all the time, as it only sends its inputs to the centre side, making it use less energy.
[7] Ergogen can be found under https://ergogen.xyz, but I like https://ergogen.ceoloide.com/ more, as it gives you a better user interface.

```
        pinky:
        ring:
        middle:
        index:
        inner:
      rows:
        bottom:
        home:
        top:
```

The `points` are pretty much just the individual keys, the `zones` are the just a collection of key-grids, of which `matrix` is one example. Another zone we need to define is the thumb zone with the three thumb keys, which need to be moved out of the way. All the numbers to define positions are based on the metric systems millimeters.

```
    thumbfan:
      anchor: # defines the anchor point for the zone
        ref: matrix_inner_bottom # this sets the base reference for the position to
the bottom of the inner column in the matrix zone
        shift: [-24, -20] # moves this previously defined reference to where we
want along the [x-axis, y-axis], cannot be used for specific keys
      columns:
        near:
        home:
        far:
      rows:
        thumb:
```

Now this itself would only generate the keys without any column stagger, which is not what I want with this keyboard. The rotations for the thumbfan are also not included right now, which makes us use the attributes `key.stagger`, `key.splay`, `key.origin` and `key.spread`.

```
        outer:
          key.stagger: 0
          rows.bottom.skip: true # this is for not having a third key in the outer
column
        middle:
          key.stagger: 4 # moves the column up or down
          key.splay: -0
          key.origin: [-12, -19]
        home:
          key.spread: 21.25 # how far apart all the column should be from the
previous one, or rather the distance between the points of the keys
```

```
          key.splay: -12 # rotates the key based on the
          key.origin: [-11.75, -9] # used to move key base positions
```

This is not the full text that is in the file to complete the description of the layout, rather just some things you can try to implement yourself, as that will have a greater learning effect than just blindly following instructions. The complete files can be found in my GitHub Repository[8], but currently this is not the complete file, only a good start.

Choosing the Microcontroller

What is a microcontroller?

A microcontroller can be explained as a tiny computer inside the keyboard. Just like your brain tells your body what to do, the controller tells the computer how to react to your key presses. It handles all the important tasks your keyboard needs to function. When you press a key, the microcontroller detects this action and translates it into a signal. This signal is then sent to your computer, which understands it as the letter or function you intended.

Which options are there, and which did I choose?

There are a lot of different controllers that all work for different price ranges and use-cases, but for me the nice!nano[9] works best, as I need the Bluetooth capabilities like mentioned in point 4 in the goals, but you could also consider another popular one, the Pro Micro. A list of different microcontrollers based on which use case you have can be found in an article by Golem published in 2021[10]. It might be a good start for you, if you have other goals for your project.

Batteries – Which fit my Build?

Specifications

This is linked to the microcontroller you have chosen, as all of them have different needs and requirements. If your microcontroller does not support wireless usage, then you will not need a battery and can skip this part. The general goal is to find a compact battery with a huge capacity, so I don't have to charge the keyboard that often.

[8] The repository can be found here: https://github.com/lultoni/pengo-keyboard
[9] You can find more information here: https://nicekeyboards.com/nice-nano (Last accessed 22 September 2024)
[10] Article Link: https://golem.hu/guide/controllers/ (Last accessed 22 September 2024)

My Pick for the Project

The nice!nano needs a 3.7 V lithium-ion battery with at least a 100 mAh capacity. I am sure you can find a lot of rechargeable batteries online that fit this description, but as I am okay with this component taking up more space and in following of this changing the positioning of the PCB, I chose the PS3 Controller Battery, as it is compact with a size of 36mm * 58mm * 6mm, but still stores 1800 mAh. The bigger the capacity of your battery is the longer you will have to charge it as well. Using this battery will create a little tilt in the board so that it is higher in the middle of the two parts and lower on the outside, an unintended but fortunate ergonomic feature.

The ZMK Power Profiler[11] is a tool to give you a rough estimate on how long the center and peripheral sides will last for until you must charge them again. If I pluck in my data, the tool calculates that I need to charge my center side around every 6 months and the peripheral side around every year. I cannot confirm these statements yet, as the keyboard has not been in use that long and I charge it occasionally, despite it having more power for months.

The PCB Plate and the Routing on it

SUBHEADINGS?

The PCB plate is where the controller, the battery, the switches and the other small components are all connected with each other. It is very simple in theory, so you could hand wire your keyboard, but from any source I read from, that had created a board with this method said something along the lines of „never again".

WHATCH THE IN-DEPTH VIDEOS FIRST TO GET A GOOD OVERVIEW AGAIN -> ALSO MAKE NOTES NOT RELATED TO THE CURRENT HEADER

Links to the first video that talks about Ergogen and kicad, which you can mention here and in the pcb section: https://www.youtube.com/watch?v=UKfeJrRIcxw
And the second video, no clue right now what is mentioned:
https://www.youtube.com/watch?v=M_VuXVErD6E

How exactly is the grid system included? Especially with the diodes.

Add the Ergogen PCB section in here, instead of the Designing the layout section

[11] The ZMK Power Profiler can be found under this link: https://zmk.dev/power-profiler (Last accessed on 22 September 2024)

- I already talked about points, so you only need to talk about the ‘outlines’ and ‘pcbs’
- Change the outlines to look like you want
- In the PCB section tell the microcontroller which key is which
    - In Christians video at timestamp 7:26
- Add a footprint for the microcontroller, power switch, reset button
    - How is this done exactly so electrical signals do not interfere with each other (look at the kicad files from Christian how his things are placed and how he talked about it in the code
- Find a way to export the outlines (cut out the keys) to make a case
    - Look at the case section for better specifications
    - Do not go into details here tho, only say why you are exporting and what
- Export the PCB files for both the center and peripheral sides

With all the components positions defined all that must be done is to route the electrical signals along, so they do not interfere with one another. For this I used KiCad, also a free software that tells us how many connections are unrouted and where they need to go, just like a big connect the points game, which you must click through to get your final PCB files.

(TRY AND INCLUDE VISUALIZATION FOR THIS STEP)

With the finalized version of the PCB we can go to a supplier of your choice (I would recommend JLCPCB or aisler, those gave me the best quotes, but that can differ for you) and click through the quoting. These were the choices I took:

- Does Christian talk about this in his video?

Which Switch should I use?

SUBHEADINGS?

Key Switches decide a lot of the feel of the keyboard, so I needed to think about this aspect a good bit. I will solder on hotswap sockets to allow the changing of switches, so if I feel like I chose the wrong ones I can just replace the switches I have installed. As I wanted to make my project silent or at least as silent as possible (point 6 of the goals), I chose tactile switches that make as little noise as possible. This can also be influenced by modding the keyboard with PE Foam or Silicone plates beneath the switches. Some people might also like a low-profile switch in comparison to the normal profile, which gives it more of a laptop look and feel. In each category you have high- and low-quality options, where I would personally rather spend a bit more, just to be sure to avoid key chatter[12].

My final pick were the Gazzew Boba U4 Silents[13], which are tactile, have a normal profile and are designed to be quiet[14]. If you want other inspirations in a similar direction, maybe because you did not like my pick, you can look at the following options:

- Akko Penguin Silent
- Zeal PC Zilents V2 62/65g
- Glorious Gateron Brown
- MX RGB Ergo Clear
- TTC Venus 45g Linear
- Everglide Aqua King V3

All the links to where you can buy these can be found in the references.

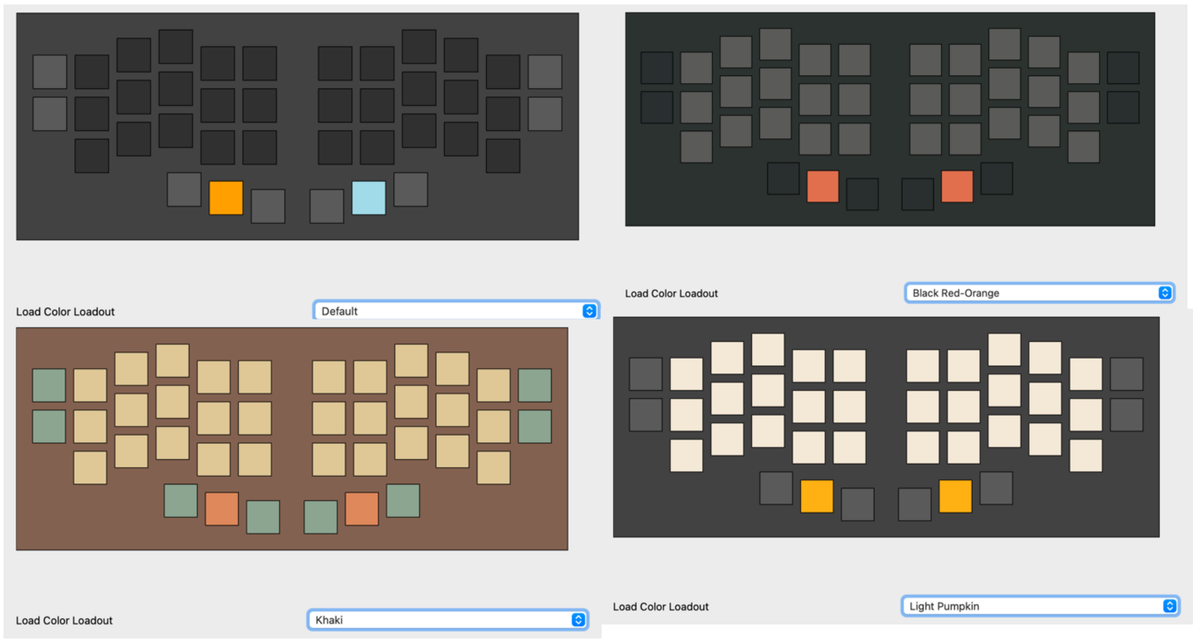
The Keycaps that I will be using for my Keyboard

SUBHEADINGS?

Keycaps play a huge role in how your keyboard looks. They are exchangeable so you do not need to worry about only getting one set of them and having to stick with them, in the end it is about design choices and money. My parameters are that they are normal profile ones, just like my switches, they have the MX/Cherry profile[15] as my switches have the same and that they are made from PBT Plastic, this them feel nicer to the touch.

For me a huge part is also color design, as everything should look good together, like mentioned in point 7 in my goals. I got my inspiration on Pinterest, where I pinned all the designs I could consider in my vision, which gave me a good idea on what I would go for. I created a little program in Python with the help of ChatGPT that would let me display and change colors based on my layout[16]. My favorite start combinations based on the pins on Pinterest were these:

[12] Key Chatter is the problem of one key press giving out multiple actions, so one C-Key press writing a "cc" instead of a "c".
[13] The Gazzew Boba U4 Silents can be found under this link: https://thocstock.com/switches/gazzew-boba-u4-silents (Last Accessed 22 September 2024)
[14] If you want the non-silent version look for the Gazzew Boba U4T Switches.
[15] A medium deep cross sized hole in the center of the backside.
[16] You can find the programm in my GitHub repository under the name 'layout_color_tester.py'.



Load Color Loadout: Default

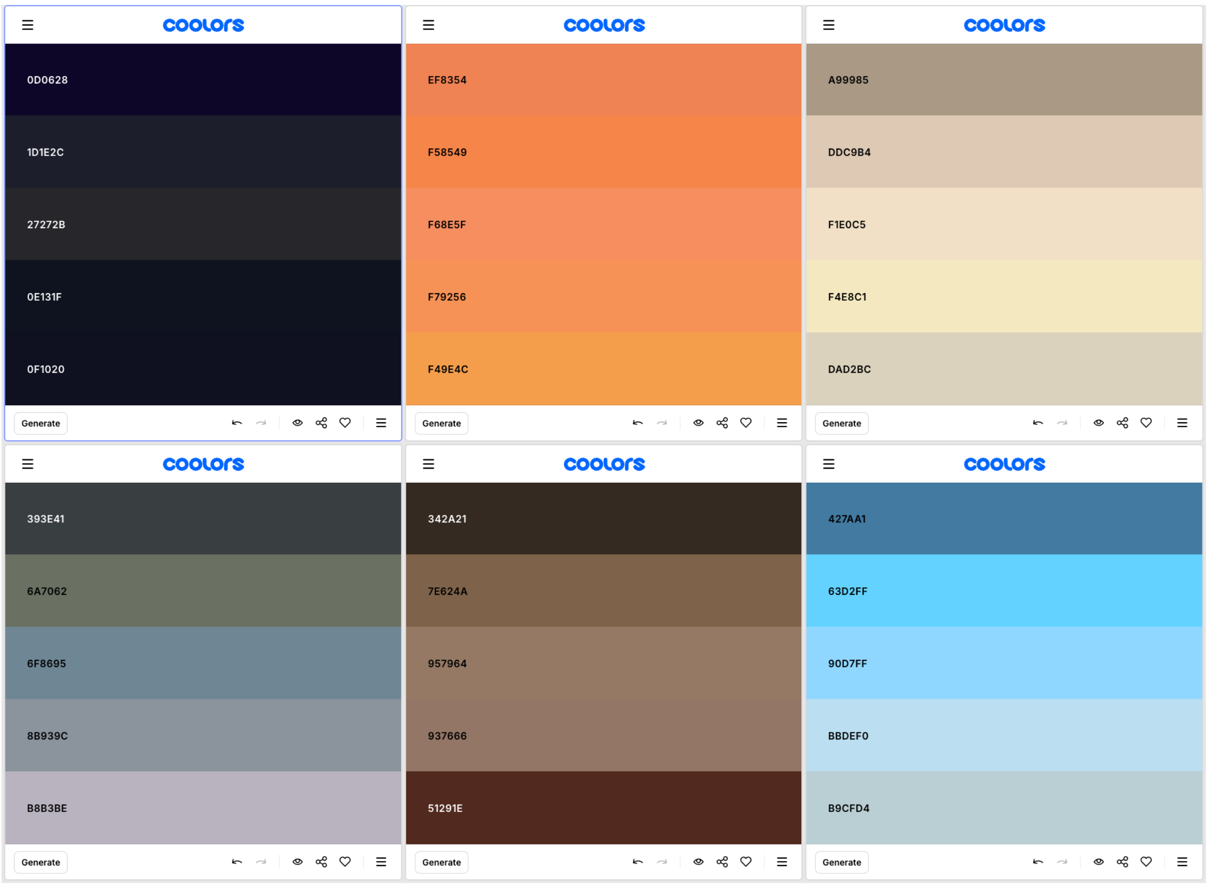
Load Color Loadout: Black Red-Orange

Load Color Loadout: Khaki

Load Color Loadout: Light Pumpkin

There were a few more and these are also not the first draft, just the ones I liked personally. With these in mind I went about finding some nice-looking colors[17] and created the final color schematic from these color bases:

[17] A niche website for this is "coolers" under the link: https://coolors.co/ (Last accessed 23 September 2024)



(IS THIS PART THAT NESSCESSARY?)

Find a retailer that sells you the keycaps in the colors you want or make them yourself, as I do not want to buy an entire set of keys, fuck the keys lowkey bruh

Soldering the electronics onto the PCB

SUBHEADINGS?

WHAT SAUTERING THINGS DO I NEED TO GET? WRITE RESULTS INTO THE SHOPING PART

The Case – A Container for the Components

SUBHEADINGS?

USE ERGOGENS CASE PART, THAT CAN HELP YOU KICKSTART YOUR JOURNEY I THINK

I want to create a case for my board, some like the look of exposed PCB plates and wires and whatnot, but as I want a clean and simple look like mentioned in point 7 in the goals, I want to hide all of that. The case itself is 3D printed, so you can decide yourself if you want to print it yourself or if you want to use an online printing company to do so.

This can be decided as many people told me that self-printed is often of lower quality, hence I went with a company to print one for me.

The case consists of 2 parts: the plate/outside and the bottom, which is screwed to the other part. The plate hides the PCB, as it sits between said PCB and the Keycaps and is usually separated from the outside, but due to myself not wanting to separate the parts in the print I did not, it does not really matter that much.

(THE PLATE SWITCH HOLE IS 14mm * 14mm AND THE DISTANCE BETWEEN SWITCH HOLES IS 5.05mm)

The start of creating a case consists of taking your key positions

Try to measure all your components and where they will be in the case, but only once you have created the PCB and the electronics on there.

WHERE CAN I 3D PRINT THE THINGS, WHAT DO I TELL IN THE DOCS
-> DID I ALREADY WRITE ABOUT IT?

COLORS THAT FIT THE DESIGN – check out the keycaps section

Assembly of the PCB, Keys, Case and the rest

SUBHEADINGS?

Asdf

What is with these goofy things: https://www.caseking.de/glorious-mx-o-ring-key-dampeners-soft-long-key-travel-40a-thin/GAZU-720.html

- ➔ Where are they included and do you need them with your switches?
- ➔ Maybe ask the chat man himself what he thinks of this, as I am sure the research will take a good bit

10x M2x3 screws, and 5x M2x4 standoffs for assembling each side of the case (so 20 screws and 10 standoffs in total) (I'd just grab a small kit on Amazon)

(Optional) 6mm (0.24") diameter x 2mm (0.08") height rubber feet for the bottom of the case

Firmware – Making the Keyboard do what it should

Creating the Keymap for my Windows Computers

Short Notice

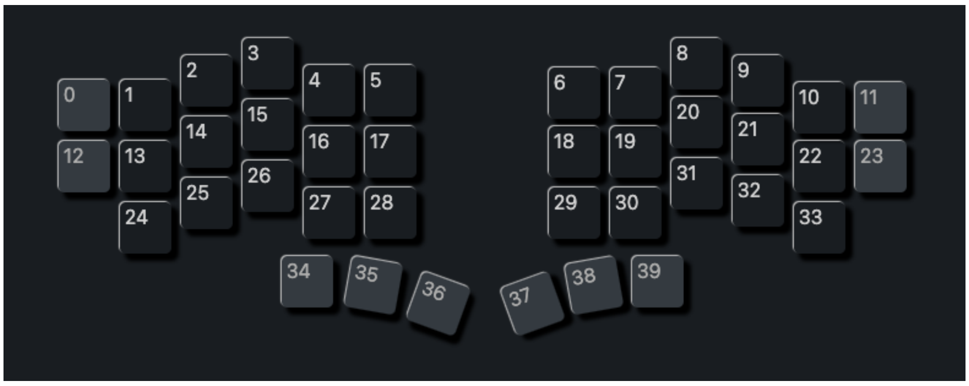
This part covers the general design process that I went through to find the ultimate and best fitting keymap for my needs, but still only talks about the requirements windows keyboards have. The Mac part is further down, which completes point 2 of the goals)

How the most important keys were placed

SUBHEADINGS?

Where can I mention this: Use this website: https://www.keymapper.dev/
What about this: https://github.com/JanLunge/pog (docs: https://pog.heaper.de/)

When creating a keymap there are a lot of things you need to take into consideration, such as finger travel or even the functionality of the different layers. The main problem that I encountered whilst doing this task was with the removed third outer pinkie key, I only had 10 outer/thumb keys that would stay, no matter the layer, as you would need them for shortcuts and such, whilst I needed to incorporate 11 keys: Shift, Control, Windows, Alt, Raise Layer, Lower Layer, Space, Enter, Tab, Backspace and Escape. One key needed to be moved to another layer where it would be accessible and, in the end, I decided to put Escape in another group on the board. The position of the remaining 10 keys was the next hurdle as they had to be sorted into these spots:



The thumb home keys (35 and 38 in this visualization) are reserved for the layer switching, as that was what most people used them for and that seemed to make the most sense when you need to move layers a lot.

I again wrote a simple python program with the help of old reliable ChatGPT to help with finding the perfect combination[18], which would again be helpful when it comes to the

[18] You can find the files in my GitHub repository, where you must, after downloading, put ‘perma_keys_lt_gen.py’, ‘layout_visualizer.py’ and ‘layouts_list.txt’ in a folder called ‘layout_generator’.

Mac part. There were a few rules set in place so that I would not have to press two keys with one finger for the most important windows shortcuts and the keys were only in positions that I wanted them to be in. After scouring the last remaining 16 layouts this is what I ended up with:

```
#-------#                                       #-------#
|       |                                       |       |
| Alt__ |                                       | BckSp |
|       |                                       |       |
#-------#                                       #-------#
#-------#                                       #-------#
|       |                                       |       |
| Shift |                                       | Tab__ |
|       |                                       |       |
#-------#                                       #-------#
         #-------# #-------# #-------# #-------#
         |       | |       | |       | |       |
         | WinKy | | Space | | Enter | | Ctrl_ |
         |       | |       | |       | |       |
         #-------# #-------# #-------# #-------#
```

Here I didn't show the layer shifting keys (35 and 38), as those would not change anyways.

What did I put in the different layers?

SUBHEADINGS?

Like mentioned before we have 3 different layers to put all our different keys. The base, lower und raise layers. When we remove all the keys, we have put in regardless of the layer we are left with 30 free slots per layer.

The base layer, the one that is normally active I have put the letter of the alphabet, the escape key and the most used symbols: comma, colon and single quote. I went with a lightly modified QWERTY layout, so that when I do not have my board I won't be so thrown off. When everything is put together it look like this:

CHANGE THE LAYOUT TO BE QWERTY INSPIRED, BUT OPTIMIZED FOR CODING AND WRITING IN ENGLISH (AND GERMAN)

(INCLUDE THE NEW BASE LAYER (put the escape key where the ‘ is, the ‘ where the / is (/ is killed), the p in the ‘ slot and the esc in the p slot))

The lower layer holds the Bluetooth functionality keys, the function keys (f-keys), and the arrow keys:

(INCLUDE THE LOWER LAYER)

After this you can define the rules after which all possible layouts are generated at the start of ‘perma_keys_lt_gen.py' and show the newly generated layouts with the ‘layout_visualizer.py' script.

The raise layer holds the last symbols and the keys that start/stop music and such:

(INCLUDE THE RAISE LAYER)

ARE ALL THE IMPORTANT KEYS ALREADY INCLUDED?

Creating the Keymap for my MacBook

Short Notice

This part is only about how I implemented the changes between the Windows and Mac Layouts and how I went about the switching between both profiles.

Asdf -> ZMK (?) implementation for this exact point.

Loading the Keymaps onto the Controller

SUBHEADINGS?

Asdf

What is with ZMK? Look at Christians video again for this part as this looks kind of scuffed tbh.

This is a ChatGPT answer for how to switch between mac and windows:

Creating a custom keyboard with ZMK Firmware to support both Windows and Mac layouts can be a satisfying project. ZMK is quite flexible and allows you to define multiple layers and keymaps. Here's a basic guide to help you set this up:

Prerequisites:

1. **Keyboard Hardware**: Make sure you have the necessary hardware — a microcontroller (like an STM32 or nRF52840), keys, diodes, resistors, etc.
2. **Development Environment**: Install the necessary tools and dependencies for building and flashing ZMK Firmware (VS Code, Python, Zephyr, etc.).

Steps:

1. Create Your Keymap

You need to define your keymaps for both Windows and Mac profiles. You can do this by modifying the keymap file in ZMK configuration.

2. Define Layers in Keymap

Create layers for the Windows and Mac layouts. Here's a basic example:

// In your keymap file (e.g., keymap.conf)

```
// Define layers
#define LAYER_WIN 0
#define LAYER_MAC 1

// Define custom keycodes for layer switching
#define MO_WIN  &layer_momentary LAYER_WIN
#define MO_MAC  &layer_momentary LAYER_MAC

// Define your keymap
&keymap {
  compatible = "zmk,keymap";

  keymap {
    // Layer 0: Windows
    LAYER_WIN {
      bindings = <
        &kp A     &kp B     // Replace with your key definitions
        MO_MAC    &kp D     // Key to switch to Mac layer
      >;
    };

    // Layer 1: Mac
    LAYER_MAC {
      bindings = <
        &kp A     &kp B     // Replace with your key definitions
        MO_WIN    &kp D     // Key to switch back to Windows layer
      >;
    };
    };
};
```

In this example, pressing MO_MAC will switch to the Mac layout, and pressing MO_WIN will switch back to the Windows layout.

3. Define Layer Switching Keys

Decide which keys will be used to switch layers. You might want to use a dedicated key or a modifier key combination. Using the example above, if you want specific keys to toggle between layers, define those in your keymap.

4. Configure ZMK and Build Firmware

Ensure your ZMK configuration includes the necessary files and configurations. Here's how you typically set up a basic ZMK configuration:

- **.conf file**: Specify any specific configurations (e.g., Bluetooth settings, power management).
- **board files**: Define your keyboard matrix and pins.

This is an example of how your project directory structure might look:

```
zmk-config
 ├── config
 │   └── board.def
 │   └── keymap.conf
 ├── build
 │   └── (build files here)
 └── boards
     └── shield
         └── your_shield_name
             └── keymap.conf
```

5. Build the Firmware

Run the build process. In your terminal, navigate to your ZMK configuration directory and execute:

```
west build -d build -b your_board -- -DSHIELD=your_shield_name
```

Replace your_board with your specific microcontroller board and your_shield_name with your custom keyboard shield name.

6. Flash the Firmware

Once the firmware is built, use the appropriate tools for your microcontroller to flash the firmware. For example, if you are using nRF52840, you can use nrfutil to flash the firmware.

```
nrfutil dfu usb-serial -pkg your_firmware.zip -p /dev/ttyACM0
```

Replace your_firmware.zip with the firmware package generated from your build and /dev/ttyACM0 with the serial port your keyboard is connected to.

Testing

After flashing, test your keyboard. Ensure that the keycodes are correct and that the layer switching works as expected.

Additional Tips:

- **Tuning and Debugging**: Use ZMK debug features to troubleshoot any issues.
- **Customization**: Continue customizing your keymap and layers as you discover your preferences.
- **Community**: Engage with the ZMK community for guidance and to share your project.

By following these steps, you should be able to create a functional custom keyboard that can easily switch between Windows and Mac layouts using ZMK Firmware.

Connecting the Keyboard to the Computer

SUBHEADINGS?

asdf

Where can I buy all the Parts and what will it cost me?

Important Notice

As this is only about my project, I can’t condone every single item I bought for you personally, you might use the same products, but dislike some of the I linked, so take everything with a pinch of salt. The prices can also vary for you, as this project was realized in late 2024, so things might be cheaper or more expensive or there might even be new products that can be used instead of the ones I used.

Parts List

Please consider that the amounts you need for every part of the board can vary compared to your one. You might need more than just 40 switches or only 1 microcontroller if you do not plan on making a split keyboard. This list only talks about consumables and not tools that you need to create this project.

Component	**Name**	**Amount**	**Final Cost**	**Buying Link**
Controller	nice!nano	2	51,88€	https://shorturl.at/bXQA3
PCB	-	2 + 8	20,00€	https://jlcpcb.com
Key Switch	Gazzew Boba U4 Silents	40 + 0	23,80€	https://shorturl.at/ye4b8
Key Caps	?	40 + 0	0,00€	?
Battery	PS3 Controller	2	35,98€	https://shorturl.at/nsxka
Case		1	0,00€	?
Machine Sockets and Pins	-	2	5,95€	https://shorturl.at/EGez0
Diodes	-	40	3,80€	Link
Battery Jack	-	2	0,95€	https://shorturl.at/O23kq
Reset Button	-	2	1,95€	https://shorturl.at/uzzaJ
Hotswap Sockets	-	40	7,80€	https://shorturl.at/cXdu8
Power Switch	-	2	1,95€	https://shorturl.at/iUYgP
Screws (For the Case)	?	?	0,00€	?
Total Cost	**-**	**-**	**154,06 €**	**-**

3D Printing and Soldering

Some might already have the tools that are needed for 3D printing and soldering or borrowed them from a friend/family member, but not everyone will have an idea what tools they need to have so I have put together a list of all the parts I used to fulfil the project:

- 3D printer or 3D printing company
- Soldering Iron

Credits

I again want to suggest the video made by Christian Selig, as it was the thing that got the stone rolling for me. He has also linked his blog post in the description, so you can go and read that as well, if you want another viewpoint on this topic. There are also a wide plethora of videos and articles about every topic I discussed in this paper so you can do your own research if you feel like I explained something so briefly.

Many thanks go to Tom, Louis and the others that helped me along the way with design choices and motivational support. Without them this project would have been tossed in the bin way before I would have ever really started with it.

I also want to thank you for reading or even just looking at this, as my goal was to inspire others to make their own creations and bring their ideas to real life. If any other questions arise you can always ask in the forums of the many software's and guides that were used, as the people are most likely happy to help you out.

Thank you for reading,
Elias Glauert

References for the Work

SORT ALL OF THESE ONCE YOU HAVE FINISHED THE PAPER

1. "I Built My Dream Keyboard from Absolute Scratch" by Cristian Selig under the link https://www.youtube.com/watch?v=7UXsD7nSfDY where you can also find his parts/files and socials in the description.
2. "Staggered VS Ortholinear keyboard, what are the differences" by Tech Fairy under the link https://tech-fairy.com/staggered-vs-ortholinear-keyboard-what-are-the-differences/ where a nice visualization of staggered and ortholinear keyboards are to be found.
3. The website of nicekeyboards who make the nice!nano under the link https://nicekeyboards.com/nice-nano where you can find all the technical information on the nice!nano and where to buy the microcontroller.

4. “Common controllers for keyboard building 2021” by Golem under the link https://golem.hu/guide/controllers/ which includes a list of controllers that can be used to build keyboards.
5. “ZMK Power Profiler” by ZMK Firmware under this link https://zmk.dev/power-profiler can show you how long your keyboard will last until it the batteries need to be charged again.
6. The Gazzew Boba U4 Silent Switches can be found under this link: https://thocstock.com/switches/gazzew-boba-u4-silents (Last Accessed 22 September 2024)
7. These are the links to the different switch-options that I mentioned in that part of the paper:
    a. Akko Penguin Silent: https://geekboards.de/shop/akko-penguin-akko-penguin-silent-3239?variant=4414
    b. Zeal PC Zilents V2
        i. 62g Version: https://geekboards.de/shop/zt0348-zeal-pc-zilents-v2-62g-413?variant=1197
        ii. 65g Version: https://geekboards.de/shop/zt0349-zeal-pc-zilents-v2-65g-412?variant=1196
    c. Glorious Gateron Brown: https://www.caseking.de/glorious-gateron-brown-switches-120-pieces/GAKC-052.html
    d. MX RGB Ergo Clear: https://www.cherry.de/mx-rgb-ergo-clear-switch-kit
    e. TTC Venus 45g Linear: https://mechanicalkeyboards.com/products/ttc-venus-45g-linear-pcb-mount-switch
    f. Everglide Aqua King V3: https://everglide.co/collections/switches/products/everglide-aqua-king-water-king-switches-v3?variant=42887000326356
    g. All these links were last accessed on 22 September 2024
8. A website for finding some nice color combinations is “coolers” under the link: https://coolors.co/ (Last accessed 23 September 2024)
9. Ergogen can be found under https://ergogen.xyz, but I like https://ergogen.ceoloide.com/ more, as it gives you a better user interface.
10. The GitHub repository can be found here: https://github.com/lultoni/pengo-keyboard, which contains all the files that were used and created in this project.