Creating a Complete Custom Keyboard from Scratch

Elias Glauert

October 7, 2024

Contents

1 Introduction

This paper documents my journey of creating the Pengo-keyboard, a custom-made keyboard from scratch, inspired by a video made by Christian Selig[1]. Watching Selig's video, where he built a custom keyboard, sparked my interest and guided me to embark on a similar project. I highly recommend viewing his video and engaging with his content, as it provides a comprehensive overview of the scope and challenges involved, which can take several weeks to complete.

Custom keyboards can be expensive, particularly the high-end or next-generation models, so it is important to address the cost early on. Creating a keyboard like this typically requires an investment of a few hundred euros. If this is beyond your budget, you might want to reconsider following along this path.

For those familiar with custom keyboards, terms such as switches, keycaps, and modifications will be well-known. Next-generation boards like the Corne or uniquely sized keyboards like the 40% are related to the Pengo-keyboard but will not be covered in detail in this paper. Inspired by the Corne layout, I found the standard key layout didn't quite suit my hands, which led me to design and build a keyboard tailored to my specific ergonomic needs.

The purpose of this paper is to document my journey and record the insights I gathered throughout the process. As a first-timer in such a project, considerable effort went into researching, designing, and building the keyboard. This paper can be read as a guide, with the content section providing an overview to help navigate through the document and focus on the parts most relevant to you.

Feel free to share this work, build upon it, and spread the knowledge about next-generation keyboards. All I ask is that you credit my contributions, just as I have credited Selig, as much of what you will read builds upon his research.

Please note that, as a student living in Germany, some information may not precisely fit your needs or could be slightly biased. This is my first time undertaking a project of this nature, so some details may not be perfect. Use this article as a starting point for your own journey into custom keyboard creation.

[1]`https://www.youtube.com/watch?v=7UXsD7nSfDY` - The video gives a good overview of the tasks that this project includes.

2 Goals and Specifications for the Project

2.1 Important Notice for using this as a Guide

The objective of creating a custom keyboard is to have a personalized and enjoyable experience. While this document will detail my choices, they are based on my needs and preferences and may not suit everyone. Please use this paper as a reference and not a perfect creation, but feel free to adapt or improve upon my design.

2.2 What are my Goals and Specifications?

My main goal was to create a keyboard that made typing easier and more efficient. After learning to touch type on a Mac keyboard, I found that it was unoptimized for my hands. Issues such as excessive movement when pressing backspace, underutilized thumbs, and fingers pressing keys designated for other fingers in when typing inspired this project.

2.2.1 List of all the Goals

1. The board fits my hands so that no hand movement is necessary while typing.
2. Keymap based on and optimized for Windows.
3. The keyboard is split into two parts for ergonomic adjustment.
4. The connection is wireless (Bluetooth).
5. No backlighting to save battery life.
6. The switches are silent to avoid disturbing others.
7. The design is simple, sleek, and fits my room's style and color scheme.

3 Designing the Keyboard Layout

3.1 Paper Prototype that is personalized

Achieving an ergonomic and personalized keyboard layout involves several key considerations:

1. **Ortolinear Layout:** Traditional keyboards have staggered rows where keys are not perfectly aligned beneath each other. Since fingers are best suited for vertical movement rather than lateral, I opted for an ortholinear layout where all keys are aligned in straight columns.

2. **Column Staggered Layout:** This design adjusts key column positions based on the varying lengths of fingers to minimize strain and travel distance.

3. **Fewer Keys:** Reducing the number of keys minimizes hand movement as different layers are used[2]. This also increases thumb functionality.

To implement these ergonomic goals, I began by placing my fingers on a 5mm checkered sheet of paper in the desired typing position. I then drew a 2x2 square around each finger to represent the approximate size of a keycap[3]. These positions formed the base for the keys in the home row.

Based on the movement range and length of each finger, I adjusted the key positions, ensuring comfort and minimizing strain. Inspired by the Corne layout, I added the remaining keys, but deliberately excluded an outer bottom key for the pinkie finger due to its limited range of motion.

This process was applied not only to the left-hand side (often referred to as the center side[4]) but also to the right-hand side (known as the peripheral side[5]). These terms are commonly used in the split keyboard community and will be used throughout this paper.

[2]A layer describes the outcome of a keypress. For example, the base layer outputs a "c" when the C-key is pressed, while a different layer (like the shift layer) outputs a "C".

[3]The actual size of a keycap is around 18mm x 18mm, and with the padding, it measures approximately 19.05mm x 19.05mm. This is similar to the 20mm x 20mm dimension of a 2x2 square on the checkered sheet.

[4]The center side is responsible for sending signals from both itself and the peripheral side to the computer. Thus, it generally consumes more energy and must always be connected for the keyboard to function.

[5]The peripheral side primarily sends its inputs to the center side, consuming less energy and does not need to be used continuously.

3.2 Ergogen - The Software Used to Create the Layout

Ergogen is a free, open-source tool designed to generate files for custom keyboard building by describing the key positions. It simplifies the process of creating a keyboard layout, transforming initial sketches from paper into a usable digital format. Detailed documentation and tutorials can be found online[6]. The interface at `https://ergogen.ceoloide.com` offers additional user-friendly features.

3.2.1 A Little Introduction to Ergogen

To begin with Ergogen, the first step involves defining the rows and columns for the final product. Ergogen operates on a matrix system, allowing for precise control over key placements. This section will outline the basic steps to get started with designing your keyboard layout.

You can do much more than just create the layout in Ergogen, but this paper will focus on the essential steps. For comprehensive tutorials, please refer to the provided documentation or available YouTube videos.

3.2.2 How to Describe the Layout to Look Like We Want It To

The key to using Ergogen effectively starts with correctly defining rows and columns. In my project, I defined my columns as: outer, pinky, ring, middle, index, and inner, and my rows as: bottom, home, and top. Below is a simplified breakdown of the steps:

- **Points:** Individual keys.
- **Zones:** Collections of key-grids, such as the matrix zone and thumb zone.
- **Matrix:** Defines the main layout structure.

For example:

```
matrix:
  columns:
    outer:
    pinky:
    ring:
    middle:
    index:
```

[6]`https://ergogen.xyz`

```
    inner:
  rows:
    bottom:
    home:
    top:

thumbfan:
  anchor: # Defines the starting point for key positioning
  ref: matrix_inner_bottom # Sets the reference point
  shift: [-24, -20] # Moves the reference point
  columns:
    near:
    home:
    far:
  rows:
    thumb:
```

To implement staggered columns and thumb rotations:

```
middle:
  key:
    stagger: 4 # Moves the column vertically
home:
  key:
    spread: 21.25 # Distance between columns
    splay: -12 # Rotates keys based on origin
    origin: [-11.75, -9] # Base positions for keys
```

This is a simplified example to help you get started. For the complete layout files, please refer to my GitHub repository[7].

Remember to consult the Ergogen documentation and community resources for more advanced features and troubleshooting tips.

4 Choosing the Microcontroller

4.1 What is a Microcontroller?

A microcontroller is like the brain of the keyboard, converting key presses into signals that the computer understands. For this project, I chose the

[7] `https://github.com/lultoni/pengo-keyboard`

`nice!nano`[8] microcontroller because of its Bluetooth capabilities, which align with the project goals.

5 The PCB Plate and Routing

The PCB plate connects all the keyboard components like the controller, battery, switches, and more. I used KiCad for routing the electrical signals.

6 Switches and Keycaps

For switches, I chose the Gazzew Boba U4 Silents[9], which are tactile and designed to be quiet. For keycaps, I opted for YMDK DSA Profile blank caps.

7 Soldering and Assembly

The PCB, components, and case are soldered and assembled according to the custom design. Lead-free solder was used for safety, and proper ventilation is essential during the process.

8 Firmware and Keymaps

8.1 Creating the Keymap for Windows and Mac

I designed the keymap using `ZMK`[10] firmware to switch between Windows and MacOS layouts. The firmware was loaded onto the controller, and Bluetooth was used for connection.

9 Conclusion

This document outlines my process for creating a custom keyboard. The steps and resources are provided as a guide to help others with similar projects. For detailed code and files, visit my GitHub repository[11].

[8] `https://nicekeyboards.com/nice-nano`
[9] `https://thocstock.com/switches/gazzew-boba-u4-silents`
[10] `https://zmk.dev`
[11] `https://github.com/lultoni/pengo-keyboard`