BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING
FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE
ÚSTAV AUTOMATIZACE A INFORMATIKY

DESIGN OF MECHANICAL KEYBOARD
NÁVRH MECHANICKÉ KLÁVESNICE

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR — **Bc. Andrej Pillár**
AUTOR PRÁCE

SUPERVISOR — **Ing. Petr Šoustek**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Master's Thesis

Institut:	Institute of Automation and Computer Science
Student:	**Bc. Andrej Pillár**
Degree programm:	Applied Computer Science and Control
Branch:	no specialisation
Supervisor:	**Ing. Petr Šoustek**
Academic year:	2022/23

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Master's Thesis:

Design of mechanical keyboard

Brief Description:

A mechanical keyboard is a type of keyboard that uses mechanical switches instead of membrane switches.
Mechanical keyboards can provide different levels of resistance and feedback, which can lead to a more satisfying typing experience. For this reason, mechanical keyboards are becoming increasingly popular among many users. These keyboards have their advantages, whether it's a much longer life, faster typing speeds or the wide range of sizes and layouts available. In contrast, conventional membrane keyboard, pressing a key pushes down on a rubber dome, which then pushes against a circuit board to register the keystroke.

The subject of this thesis is the actual design and construction of a mechanical keyboard.

Master's Thesis goals:

A overview of mechanical keyboard research and design.
Familiarization with developments for a selected microcontroller platform and QMK firmware.
Design and construction of mechanical keyboard.
Real verification of the achieved solution (typing test).

Recommended bibliography:

MONK, Simon. Programming Arduino: getting started with sketches. New York: McGraw-Hill, c2012. ISBN 978-0071784221.

GARCIA-RUIZ, Miguel Angel; MANCILLA, Pedro Cesar Santana. DIY Microcontroller Projects for Hobbyists: The ultimate project-based guide to building real-world embedded applications in C and C++ programming. Packt Publishing Ltd, 2021.

Deadline for submission Master's Thesis is given by the Schedule of the Academic year 2022/23

In Brno,

L. S.

doc. Ing. Radomil Matoušek, Ph.D. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

ABSTRACT

This work presents a construction of a computer keyboard using mechanical key switches. The first part provides a brief overview of computer keyboard history, used technologies, standards and options. Follows an overview of the keyboard designed as part of this work. The implemented keyboard is based on JWK Everglide Oreo Cherry MX style switches mounted on a PVC plate. The keyboard is powered by QMK firmware running on a 32-bit ARM MCU. The keyboard is provided with a custom keymap and is fully programmable. Additional features include a programmable rotary encoder and multi-color key underglow.

ABSTRAKT

Tato práce představuje konstrukci počítačové klávesnice využívající mechanické spínače kláves. V první části je uveden stručný přehled historie počítačových klávesnic, použitých technologií, standardů a možností. Následuje přehled klávesnice navržené v rámci této práce. Realizovaná klávesnice je založena na spínačích JWK Everglide Oreo ve stylu Cherry MX umístěných na PVC desce. Klávesnici ovládá firmware QMK běžící na 32-bitovém ARM mikrokontroleru. Klávesnice je vybavena upravenou mapou kláves a je plně programovatelná. Mezi další funkce patří programovatelný otočný enkodér a vícebarevné podsvícení kláves.

KEYWORDS

mechanická klávesnice, QMK, Cherry MX, Human Interface Device

KLÍČOVÁ SLOVA

mechanical keyboard, QMK, Cherry MX, Human Interface Device

2023

BIBLIOGRAPHIC CITATION

PILLÁR, Andrej. *Design of mechanical keyboard.* Brno, 2023. Available at: `https://www.vut.cz/studenti/zav-prace/detail/145800`. Master's Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science, Supervised by Ing. Petr Šoustek

AUTHOR'S DECLARATION

I hereby declare that the master's thesis *Design of mechanical keyboard* was prepared as an original author's work under the supervision of Ing. Petr Šoustek. All the relevant information sources which were used during preparation of this thesis, are properly cited and included in the list of references.

V Brně dne 22. 5. 2023 .....

Andrej Pillár

ACKNOWLEDGEMENT

Thanks for the company and the treats, you know who you are.

CONTENTS

1 Introduction **15**
2 Computer keyboards **19**
2.1 Physical layout 20
2.2 Logical layout 23
2.3 Debouncing 26
2.4 Ghosting and rollover 27
2.5 Keyboard switches 29
2.6 Keycaps 37
2.7 Additional inputs 41
2.8 Communication Protocol 43
3 Implementation **53**
3.1 Tools 53
3.2 Switches 54
3.3 Key caps 56
3.4 Additional inputs 57
3.5 Layout 57
3.6 Controller 57
3.7 Firmware 60
3.8 Board and other components 68
3.9 Key underglow 68
3.10 Case 72
3.11 Production 77
4 Conclusion **81**
Bibliography **83**
List of Figures **91**
List of Tables **93**
List of Appendices **95**
A BOM **97**
B Electrical connection schematics. **99**

1 Introduction

Human-machine interfaces are mostly based around text. The input devices evolved along with the machines and the complexity of input data. Starting with the single-contact telegraph key, through the full-alphabet mechanical typewriter and on to the first computing peripherals, or rather paraphernalia at the time, in the form of keypunches and teletypes. These devices, though used in relation to computers were not a direct input. As the name suggests, keypunches were a specialized device used in the process of creating punch cards for the early computers. As the technology progressed and computers were able to process direct input, this input was provided in the form of the familiar alphanumeric keyboard. The general layout remained close to the fully mechanical predecessors with some special keys for sending control sequences inherited from the teletype. These new keyboards, unlike their predecessors, were no longer entirely mechanical, rather electrical, but their staggered rows remain to this day. Up until the invention of the computer mouse in the 1980s, the keyboard was the only form of input.

The first major developments in the keyboard switch space came about with the proliferation of home computer systems. Micro Switch division of Honeywell claims to have introduced the worlds first „solid state keyboard“ in 1968 [1], using micro switches with hall effect sensors [2]. Cherry, the now renowned manufacturer of switches which started in the 1950s in the automotive industry [3], entered the market with a 1971 patent number US3715545A, „Momentary push button switch with improved non-conductive cam for normally retaining movable leaf spring contacts in a non-operative position“ [4]. These switches first found their application in industrial machinery control panels but soon found their way into terminal and later PC keyboards. Whereas the other manufacturers used coil spring designs and direct electrical contact, IBM at the time went with a beam spring and capacitive pads. In addition to established industrial manufacturers, new computer switch focused companies also start to appear around this time such as Datanetics, who provided switches for the original Apple II computers [5] and who were awarded one of the first patents for what would later become widespread in the form of rubber dome over membrane keyboards [6]. A number of other, mostly industrial manufacturers produced switches are used in terminal and sometimes computer keyboards, for example the RAFI RS series.

The early keyboard layouts were not subject to any standardization and often the alphanumeric keys were the only commonality. Most early home computers integrated a keyboard into the system itself (e.g. Commodore64, ZX Spectrum, etc.). Keyboards separate from systems were only compatible with the intended computer. The incompatibility ranged from electrical, through physical connector differences

to protocols. Some keyboards had their key matrix directly connected (Commodore 128D), but the majority contained onboard logic and communicated keycodes via a serial interface. The first de-facto standardization of the layout came in with the IBM Model F and later Model M the form of which was adopted by the wide range of IBM-compatible products. Similar standardization came about on the connection side as well, with the also IBM-originated PS2 port. Introduced in 1987 with the Personal System/2, the 6-pin DIN style connector worked for both keyboards and mice and replaced the previous larger 5-pin DIN connectors and DE-9 connectors respectively. The new port implemented a bidirectional synchronous serial channel, though device transmission is slightly favored. This port remained a de-facto standard for PC hardware up until the introduction of the USB Human Interface Device protocol, or HID for short, around 2001.

The personal computer market accelerated in the 1980s and that meant progress for peripherals as well. 1981 saw the introduction of IBM's buckling spring capacitive keyboard with the Model F and the improved membrane switch Model M a few years later. While not entirely mechanical per se, the key feel is regarded as pleasant. The Model M keyboards have since attained a cult following and reproductions of the original design are available to this day. In 1983 Cherry introduced their „MX“ line of key switches, initially offering one variant with smooth linear action. The variants expanded a few years later to include a tactile option. The switches were recognized by their color-coded stems. These would later become the basis for an entire family of key switches with varied characteristics that remain a popular option many years down the line. The Japanese manufacturer Alps also introduced a line of micro switches aimed at computer peripherals around the start of 1980s. While not as popular as the MX line, switches of the Alps SK design can still be found in some custom boards. The original manufacturer discontinued their production sometime around the late 1990s but the switches gained a following and a number of clones from different brands is still available, most notably the Canadian company Matias [7]. The capacitive dome-and-spring Topre switch was also introduced in the 1980s.

The following decade did not see much new technology but a lot of improvements. IBM replaces the buckling spring with a rubber sleeve, essentially starting the trend of simplification for mass-production which brought about the plentiful cheap rubber dome keyboards. Cherry introduces a tactile non-clicky variant with a brown stem, initially available to keyboard OEMs. The low profile Cherry ‚ML‘ line becomes available in the 90s as well.

Rapid proliferation of personal computing meant cost-optimization, and as a result cheap mass produced rubber dome keyboards became the standard, replacing

the dedicated mechanical switch keyboards with soft rubber dome monolithic designs with printed membrane matrices. While certainly simpler, more compact and lighter, the typing feel leaves a lot to be desired. Popularity of mechanical keyboards waned rapidly through the early 2000s, falling out of favor mostly due to their high prices and general lack of interest in „old tech“. They were relegated to the high-end segment and small „enthusiast“ communities. This trend started seeing a reversal in the 2010s though mostly due to renewed interest from the computer gaming community, especially in e-sports. Mainstream manufacturers slowly started offering mechanical options again. The situation further changed for the better in 2014 with the expiration of Cherry's patent. More companies were free to create their own switch types and the market expanded significantly. Exclusively switch-focused compaines such as Gateron or Kailh-kaihua started offering a wide range of switches and even mainstream manufacturers started engineering their own modifications. Cherry MX became the de-facto standard mechanical switch format and continues to be popular.

With seemingly ever expanding availability of different switch types and novel layouts, mechanical keyboards once again enjoy a high degree of popularity among a broad range of users.

2 Computer keyboards

The keyboard switches are arranged into a matrix of rows and columns. The exact number depends on the keyboard layout but usual counts are 6 rows and between 12 to 21 columns. One switch terminal is commoned across a row and the other across the column. The matrix interconnections are usually realized on a printed circuit board. The switches are mounted to a plate sandwiched between the top and bottom shell, or directly to the circuit board without a plate. Plates are usually made of metal but different plate materials can provide variation to the overall typing feel. A diode is installed in the direction of the row or column connection to prevent unwanted input when pressing multiple keys, known as ghosting, described in Section 2.4. The aforementioned matrix of switches is often formed according to a standardized layout. Basic physical layouts in use today are descended from the IBM PC layouts thanks to the wide clone market. Key functions are then defined by the logical layout. Functions can be printed on the key caps but they are entirely up to the interpretation of host software, allowing the use of multiple layouts on one keyboard. The key switches are single pole, single throw momentary switches. Since most switches depend on physical contacts, steps must be taken to avoid erroneous reading due to contact oscillations. This is called debouncing, further described in Section 2.3. Various switch constructions are available. The switches are capped by so-called key caps which, as mentioned, usually carry a visual information about the key function. These can take various forms as well and are mostly classified by switch compatibility and material. Finally, since USB HID is a broad specification, keyboards can and sometimes do include additional interface controls such as integrated pointers, trackballs, rotary or multi pole switches. A schematic of a typical mechanical keyboard construction can be seen in Fig. 1. Fig. 9a depicts an alternative low-cost rubber-dome over membrane keyboard construction.

The keyboard controller scans the matrix with a high frequency, detecting and processing key presses. Matrix scan duration depends on the microcontroller and the matrix configuration but generally the matrix is scanned between 500 to 2000 times per second. First the pins configured as columns are set as output and rows as input (or vice-versa, depending on diode polarity). The columns are then sequentially pulled high, reading the input from row pins each time. If a high bit is read it means a key on the corresponding row was pressed. The controller then processes the resulting scancode and communicates it to the host computer. Protocols used when communicating with the host are described in Section 2.8.

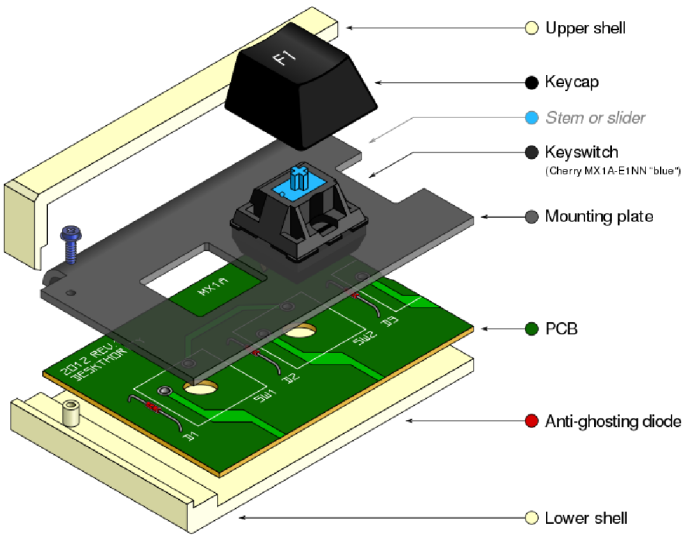
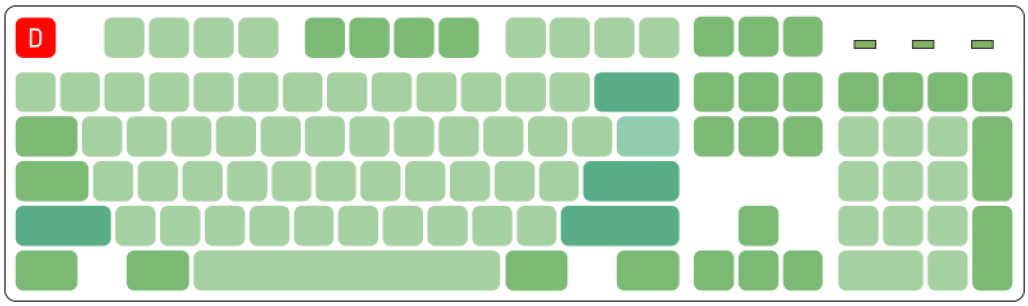


Fig. 1: Construction of a typical mechanical keyboard [8].

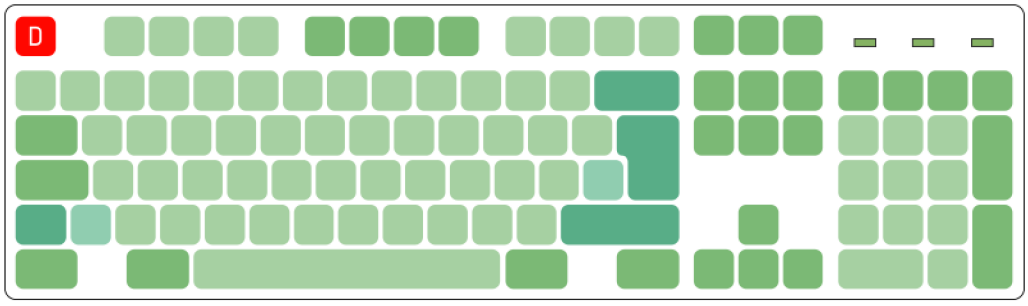
2.1 Physical layout

The standard physical layouts are ANSI 101 also known as the standard US layout (Fig. 2a), ISO 102 with larger two-row enter key and an extra key next to left shift widely used throughout Europe (Fig. 2b), and Asian 101 with even larger two-row enter key and a single width backspace key (Fig. 2c). The layouts can be seen in Fig. 2. Three indicator LEDs are usually included, usually placed in the the top right corner over the number pad. These signal the activation of from left to right num lock, caps lock and scroll lock. Num lock controls whether the number pad keys output scancodes for numbers or navigation. Caps lock is used to change the default for alphabetical keys to upper case. Scroll lock was used to change behavior of the arrow keys from moving the cursor to moving the content in a focused window. This functionality is nowadays needed only very rarely. Since scroll lock is a vestigial function, both the key and the indicator position are sometimes used for other purposes or omitted altogether. These base layouts are not rigidly defined and are often customized by subtracting or sometimes adding a number of keys. Other common customizations include rearranging of key blocks, most often the cursor keys and navigation block.
Four main modifications, selection shown in Fig. 3, of the base layout are widespread:

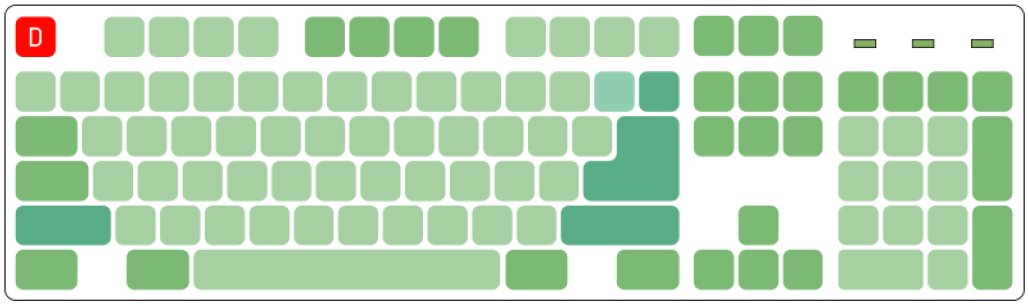
- Tenkeyless: Omitting the number pad (Fig. 3a)
- 75%: Compact layout mainly used in laptop keyboards, usually achieved by repositioning navigation keys and eliminating gaps between key blocks (Fig. 3b)
- 60%: Omitting the function row and navigation block (Fig. 3c)
- 40%: Further shedding even the number row, mostly custom made, commercially rare



(a) ANSI 101 standard layout [9].

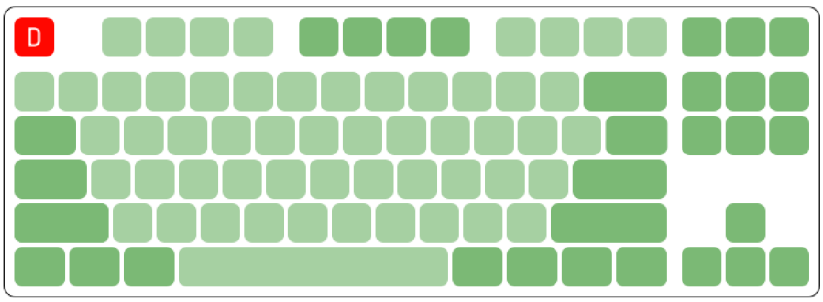


(b) ISO 102 standard layout [10].

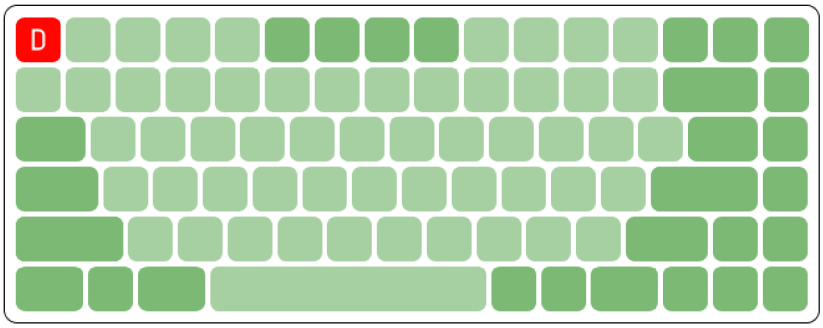


(c) Asian 101 standard layout [11].

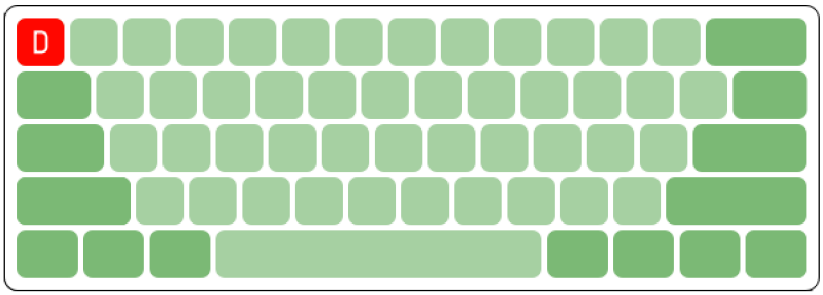
Fig. 2: The standard full-size layouts



(a) ANSI tenkeyless [12].



(b) ANSI 75% [13].



(c) ANSI 60% Modified from [14].

Fig. 3: Miniaturized ANSI layouts

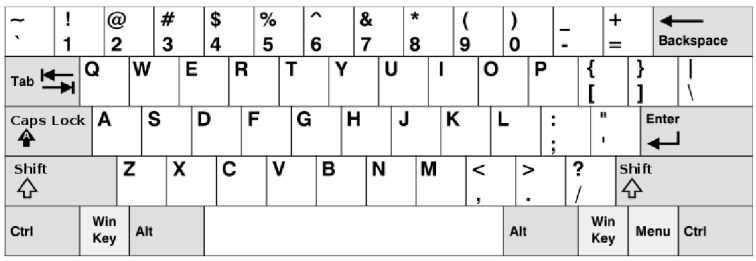
There is not a limit to the amount of customization and some keyboard peripherals do not conform to any standards on shape. Some keyboards even physically split the keys into two halves for more ergonomic positioning. Other non-standard examples include the various angled ergonomic layouts or the ortholinear (all keys in straight rows and columns) layout popular with 60% and smaller DIY keyboards.

2.2 Logical layout

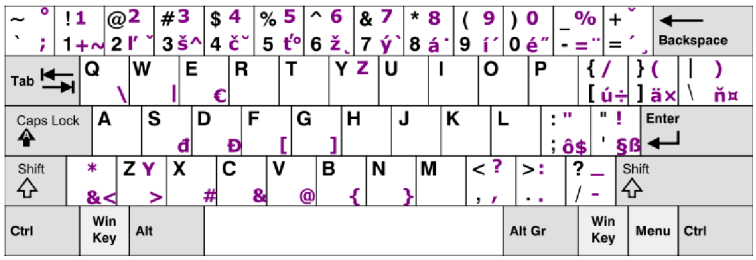
As mentioned above, the logical layout describes the key functions. The widespread and de-facto standard layout called QWERTY is inherited from the widely popular Remington no.2 typewriter, complete with the staggered rows [15]. It is named after the first six letters of the first alphabetical row. The layout survived with only minor changes from the late 1870s when it was first commercially introduced. The base layout, shown in Fig. 4, contains the letters, numbers and some more frequently accessed symbols. A heatmap of the used keys can be seen in Fig. 5a.

Many regional adaptations exist, replacing some symbols with regionally more appropriate ones such as accented characters and in some cases rearranging the letters. For example the first six letters are organized as AZERTY in French or QWERTZ in German, Slovak and Czech layouts as can be seen in Fig. 4b. The regional layouts are standardized by local standard bodies. Alternate layouts are available for languages that do not use the latin script such as Russian or Greek. These include the appropriate symbols in addition to the latin ones, though sometimes with a different layout i.e. the local symbols need not respond to the latin symbols they share a key with. Every layout supports working with layers, enabling access to more symbols by combining a keypress with a modifier key. The obvious example of this behavior is the case change when combining letters with the Shift modifier. Regional layouts often define additional layers, accessible for example using the Alt Gr modifier on the European ISO layouts. Custom keyboard controller firmware can further enable the use of additional modifiers or modifier combinations to access several defined layers. This functionality is used extensively in keyboards with more compact layouts. Accented characters can be also typed with the use of so called dead keys. This term comes from the typewriter era where some accent keys could be typed without moving the carriage allowing the typist to combine them with a letter. In computers this functionality is usually accessed via a compose key. While not usually a part of the standard layout, the functionality can be assigned. Similarly to the typewriter, pressing this key allows the user to combine an accent symbol with a letter. Some common examples include single quote for acute accents, backtick for graves or caret for circumflex. The available combinations are determined by the OS implementation and the character's availability in the unicode set.

Several alternative layouts exist, of which the best known are Dvorak, proposed by August Dvorak in 1936 and Colemak, proposed by Shai Coleman in 2006. Both layouts aim to optimize finger and hand movements to improve typing ergonomics. The Dvorak layout is based on studies of letter frequencies and human hand physiology, aiming to reduce typing effort [18]. This is attempted by sorting the letters in a manner that requires the use of both hands to type most common words. The



(a) QWERTY ANSI layout [16].

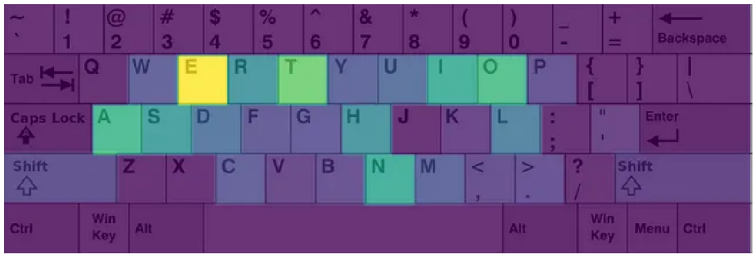


(b) Regional modification, Slovak QWERTY/Z [17].

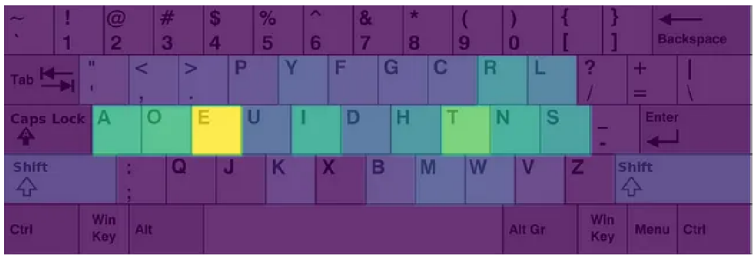
Fig. 4: The QWERTY layout.

most common letters are placed on the middle row thus theoretically reducing finger travel. An illustration of letter frequencies can be seen in Fig. 5b. Studies were performed in the USA in an attempt to standardize this typing layout but the bid was ultimately rejected since not enough advantages were found over the ubiquitous QWERTY. Later, in the 1980s was the layout included by ANSI as an alternative standard layout. Several modifications exist including regional and a programmer's Dvorak with rearranged special symbols. Nowadays is the layout available for use on all major operating systems.

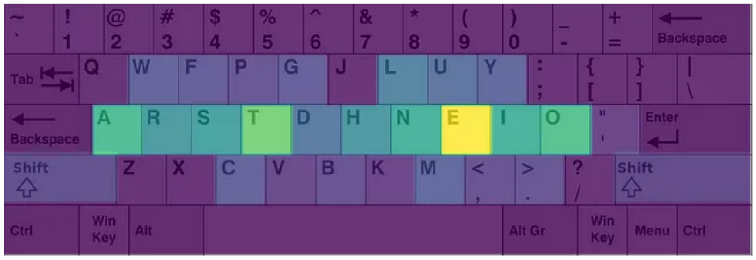
The more recent Colemak layout is based upon the same idea of reorganizing the letters by frequencies and reducing typing effort. The other keys are left in their original positions. Another change is the replacement of the caps lock key with an additional backspace. The character heatmap for Colemak can be seen in Fig. 5c. The main points raised by its author about the shortcomings of the Dvorak layout include the difficult transition from QWERTY, the layout's bias towards the right hand and the uncomfortable use of some common shortcuts [19]. The layout attracted an active community who further experiment and modify the layout. The Colemak layout is ranked by the carpalx project near the top in terms of typing effort. The carpalx project is an effort to test and optimize keyboard layouts. The project ranks the layouts by performance in a typing effort model. The evaluated metrics include frequency of hand, finger and keyboard row usage, asymmetry of hand usage and the number of successive presses by one finger, hand or on one row. A number of more or less optimized layouts has been created and evaluated, with some available for in the Unix environment [20].



(a) Character frequency heatmap for QWERTY layout.



(b) Character frequency heatmap for Dvorak layout.



(c) Character frequency heatmap for Colemak layout.

Fig. 5: Character frequency heatmaps; purple least frequent, yellow most frequent. Modified from [21].

2.3 Debouncing

Every switch suffers from oscillations due to the contact movement. Steps must be taken to avoid unintended input resulting from multiple rapid contact closures. Time required for the contacts to settle is specified by the switch manufacturer. In the case of modern Cherry MX switches for example, this time is given as sub 1 ms [22]. Mitigation of these oscillations is called debouncing and usually entails averaging the input value over a period of time. Debouncing can be managed in hardware or software.

Hardware debouncing with discrete components means a significant increase of component count. The (component-wise) simplest implementation consists of two NAND gates forming a set-reset latch. This setup however requires a double-throw switch which is not at all common in general computer input applications. This approach, even if very effective, is therefore used only rarely. Another option is to use an RC filter network. The filter schematic can be seen in Fig. 6. Since charging and discharging a capacitor takes time, the appropriate logic level is reached smoothly. Placing such circuit with a proper time constant in front of the logic input therefore eliminates the effects of contact bounce. An inverter is required as well since the capacitor charges with the switch open. The time required for the charge or discharge is controlled by the component values. These need to be carefully chosen to obtain a balance between time required for the contacts to settle and input responsiveness. Another caveat of this approach is the voltage threshold specified for an input to be interpreted as a logic one or zero. The capacitor charges in a linear fashion which means that the voltage level will spend some time in the undefined region. It is therefore recommended to use an inverter with input hysteresis i.e. a Schmitt trigger to avoid undefined behavior on the processor side [23].

According to the diagram in Fig. 6, five additional components are required per switch. Such component count increase is untenable for a keboard consisting of tens of keys even considering an inverter with multiple I/O in one package. Integrated circuits, such as MC14490 which is a dedicated debouncer with 6 I/Os [24], do exist but their high price makes them non-competitive to a software-based approach [25].

Modern microcontrollers are easily able to cope with the minimal extra processing required for software input debouncing. The simplest approach to software debouncing is to lower the sampling rate. This however introduces unwanted input delay and a more sophisticated approach is required for all but the most rudimentary applications. One approach is to sample the switch status periodically and look for a number of sequential stable readings. The switch state is considered stable and switched after a satisfactory number of readings show one state, otherwise is the

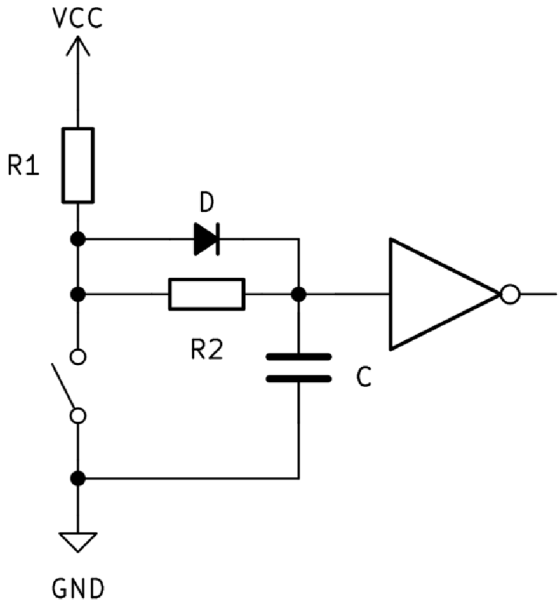
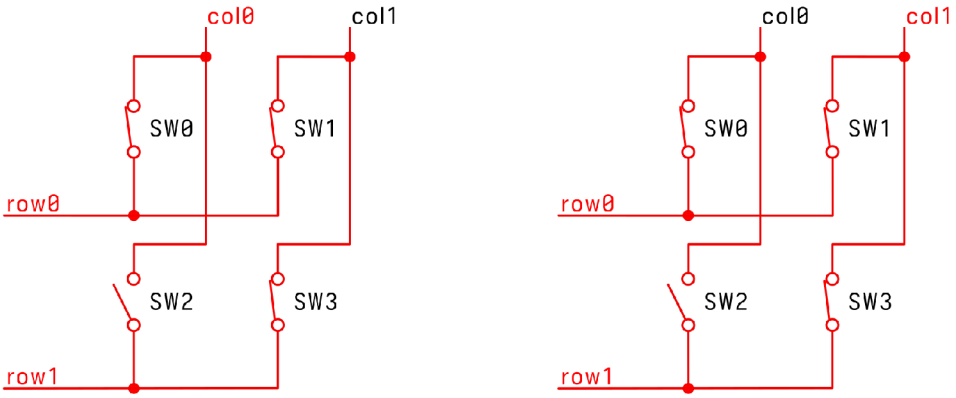


Fig. 6: Schematic of a hardware-debounced switch, modified from [23].

state reverted to its initial value. The number of matching readings and the sampling rate are adjusted based on the switch settle time and required reponse time. A response time of up to 80 ms is seen as isntantaneous, times over 100 ms can feel noticeably delayed even in general usage. Response time requirements tend to be higher for fast-paced actions such as video gaming.

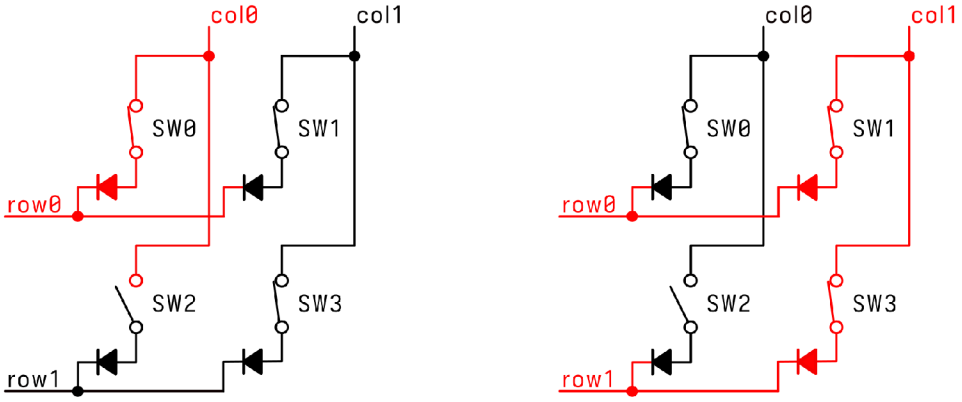
2.4 Ghosting and rollover

When switches are connected directly across the matrix rows and columns, pressing three or more keys on two different rows and columns results in the controller reading unintended additional input. This is caused by voltage present on one column when scanning being short-circuited through the pressed keys to another row. A simple two-by-two matrix in Fig. 7 illustrates this process. Lines with voltage present are highlighted in red. As can be seen in the figure, voltage applied to column 0 finds its path trough the adjacent switches. This results in an erroneous reading as if both SW0 and SW2 were pressed even when that is not the case. Simply adding a diode in series with the switches rectifies this issue, as illustrated in Fig. 8. This issue can also be avoided by directly connecting each key to the controller or otherwise electrically isolating the keys. When ghosting remediation is present, the diode approach is the most common [26]. Another way to avoid ghosting issues without increasing component count is to ignore key presses that could potentially be a result of ghosting. This approach is called blocking and is found in cheaper products. Since the underlying matrix need not correspond with the key layout and ghosting can



(a) Scanning column 0 (b) Scanning column 1

Fig. 7: Ghosting when pressing multiple keys



(a) Scanning column 0 (b) Scanning column 1

Fig. 8: Series diodes prevent ghosting

occur with three or more keys pressed, the safe limit of simultaneously pressed keys is only two. There are some attempts to improve rollover on the cheap by skipping anti-ghosting and instead optimizing the matrix. The optimization usually involves connecting keys that could be used together in a way that should circumvent the ghosting issue.

The amount of pressed keys able to be registered at one time is called rollover. On the protocol side, both PS/2 and USB in full HID mode support reading any number of pressed keys. This of course requires mitigation of the ghosting issue. Ability of a keyboard to read and report an arbitrary number of pressed keys is usually called N-key rollover, NKRO for short.

2.5 Keyboard switches

As mentioned in the introduction, computer keyboards use single pole, single throw momentary switches. All switches consist of a stem for holding the keycap, a slider or stem that actuates the mechanism and a shell in addition to the mechanism itself. Design and construction of these elements determines the switch characteristics such as actuation force, actuation point, pre- and over-travel or tactility. Mechanical switches are usually compared based on these characteristics. The actual construction can vary between manufacturers and switch types. The mechanical contact switch has been used widely since the first computer keyboards but later gave way to the membrane switches as manufacturers sought a cost optimization. Many more or less complicated designs came and went over the years. Rubber dome over membrane is the most common nowadays mostly due to its very low cost and easy mass production. On the mechanical side, the Cherry MX design is the most widespread. Some other approaches include capacitive pads, hall effect or optical sensors. The following section describes the main switch technologies in use. Since a mechanical keyboard is the theme of this work, the main focus will be on mechanical switches.

Membrane switch

A very simple form of switch based on two layers of thin plastic membrane printed with conductive ink and a separator in between as depicted in Fig. 9b. The original design was patented in 1968 by Datanetics [6], and used a more complicated approach with several layers and gold-plated contacts. The design was later simplified and led to the Datanetics DC-50 stand-alone membrane switch where the membrane and an actuating mechanism were placed in the individual switches' enclosures. Some manufacturers combined the membranes with a traditional circuit board serving as the base layer such as in the case of the Atari 800 computer. The simple design known today did not become commonplace until the 1990s. The de-

sign is widely used in a broad range of input applications from industrial panels, through all kinds of consumer electronics to computer keyboards. The actuation force is very low and pressing such keys provides no tactile feedback whatsoever. In computer keyboards are the switch membranes usually used in conjunction with additional layer of rubber domes which provides longer key travel and a slight improvement to the actuation force required though tactile feedback is still missing. An exploded view of such setup can be seen in Fig. 9a. Membrane keys have their actuation point at the bottom of travel and provide no overtravel.

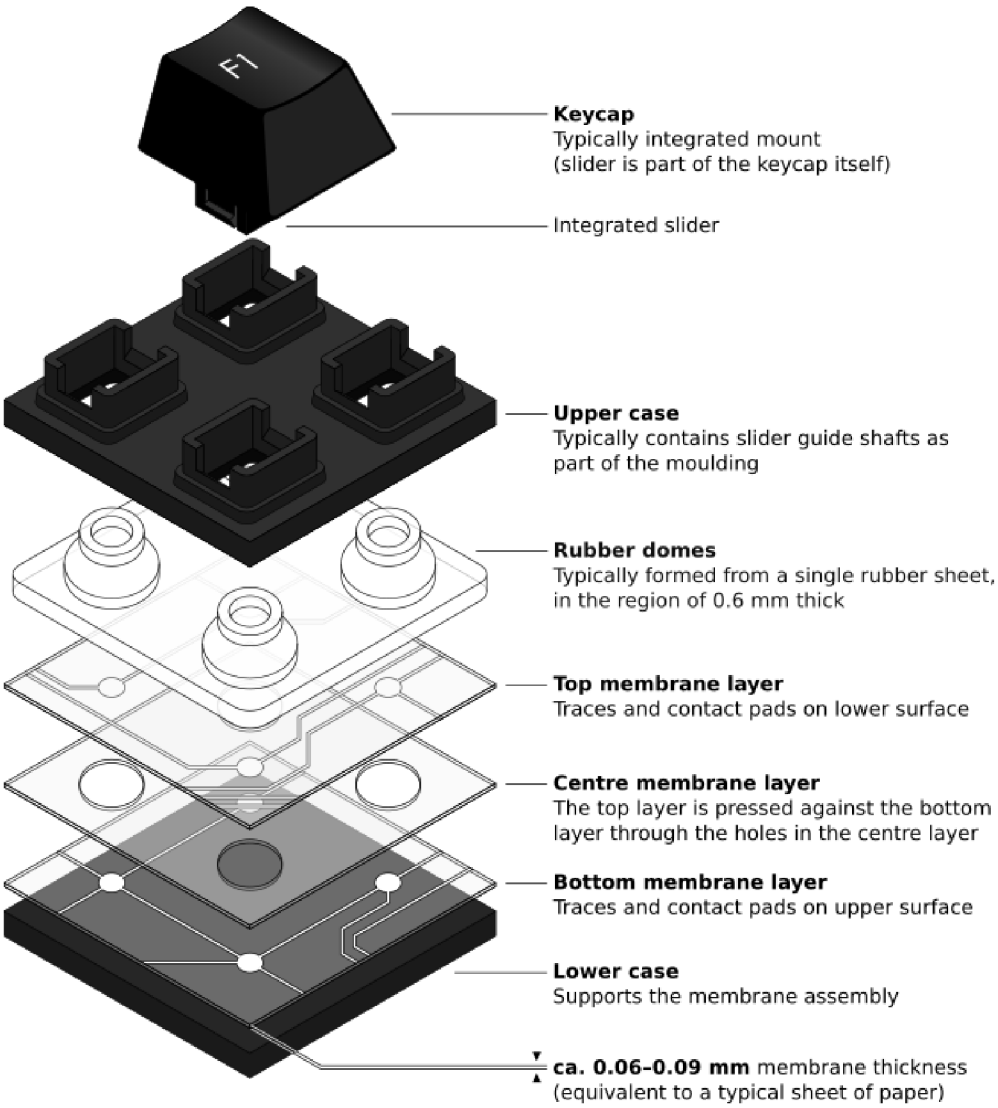
Capacitive switch

This type of switch works based on detecting a change in capacitance. The movable part and the base PCB make up two plates of a capacitor; pressing the key decreases their distance, therefore increasing capacitance. Switches of this construction do not suffer from oscillations (switch bounce) as there are no parts to oscillate since the press detection does not rely on mechanical contact. The earlier IBM PC-AT keyboard Model F used switches based on this principle. Options avialable nowadays include primarily switches from the Japanese brand Topre, used predominantly in the Realforce keyboard lineup. Originally introduced in 1984, Topre switches use a combination of a capacitive pad with a coil spring and rubber dome [29]. The construction as depicted in the original patent filing is shown in Fig. 10. The rubber domes come in varying stiffness levels providing for a tunable actuation force. The key feel is smooth with a tactile bump at the stroke start. The switches are quieter than a typical mechanical switch and enjoy high popularity among a small but dedicated fanbase. The debate around whether to consider them a mechanical switch is (and probably will remain) ongoing. Generally they are regarded as „halfway“ mechanical.

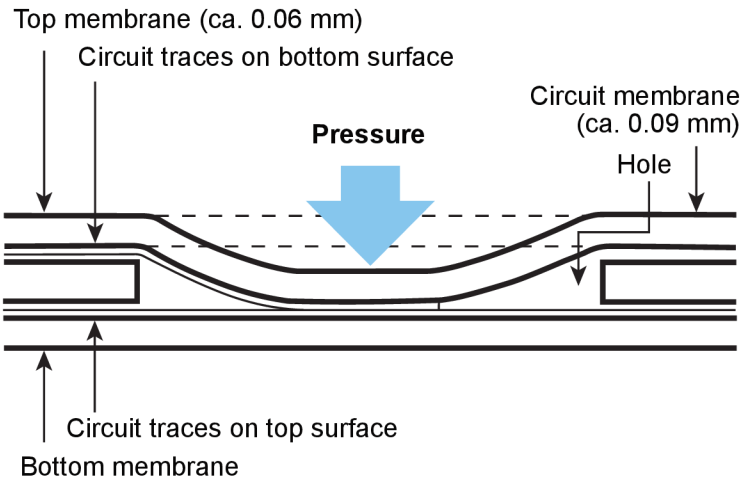
Other switches

Other than space and resulting practicality, there is no limit to the type of switch that can be used for a keyboard. Some of the more common mechanisms include hall effect and optical sensors. Both have the advantage of longer lifespan since the only mechanically stressed component is the return spring, and the absence of contact bounce again stemming from the contactless nature of press detection. The advantages are however in practice offset by the increased cost, and in case of magnetic switches also their size.

The magnetic switch works on the principle of detecting a magnetic field by means of a hall effect sensor. The field is genereated by a permanent magnet attached to the switch stem. Depending on the detector setup, these switches theoretically allow for a variable actuation point setting depending on the magnetic field intensity.



(a) Rubber dome over membrane switch schematic [27].



(b) Schematic of a pressed membrane key (detail) [28].

Fig. 9: Membrane key switch schematics

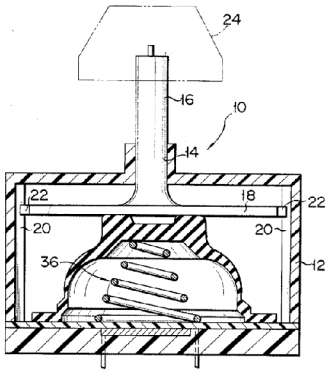


Fig. 10: Cutaway schematic of a Topre switch [29].

Optical switches replace the metal contacts by a photodiode and an IR LED. A key press is registered whenever the switch stem breaks the light beam.

Mechanical switch

Metal contacts in this type of switch are mechanically closed by the stem movement, hence the name. Many different variants and constructions exist coming from a variety of manufacturers as described earlier. This section focuses on the Cherry MX style as it is the most common and one used in the presented keyboard.

Mechanical switches are usually categorized based on their „feel“ or actuation response. The main criterion here is the tacile feedback. The feedback is provided by the interaction between the stem and contacts along the actuation path. The three feedback types are:

- Linear - no tactile feedback, stem has a smooth ramp
- Tactile - stem contains more or less pronounced bump along the ramp, usually just before actuation point
- Clicky - same tactile bump with additional audible click, achieved by multi-part stem released just after the activation point

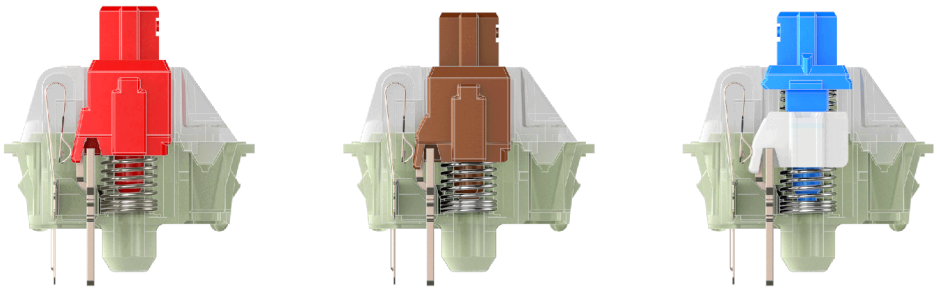


Fig. 11: Cutaway view of Cherry MX switches. From left to right: linear, tactile and clicky. [30–32].

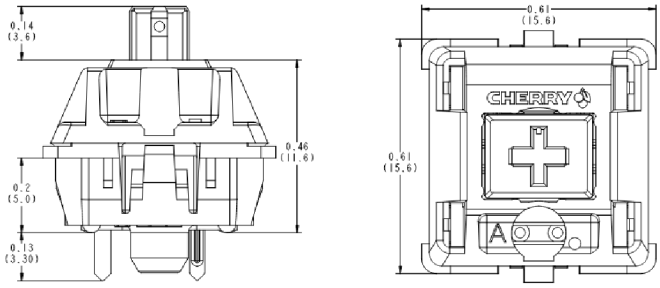


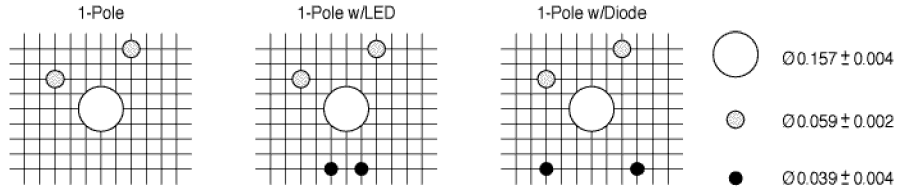
Fig. 12: Dimensions of an MX type switch [33]. Dimensions in inches with millimeters in parentheses.

Differences in the stem construction are illustrated in Fig. 11. Cherry MX style switches share the same basic construction with two part housings. The height of the switch from bottom to the top of the stem is 15.2 mm with the contacts protruding further 3.3 mm. The upper part of the bottom housing has standard dimensions starting with a 1 mm thick 15.6x15.6 mm flange followed by a 14x14 mm contact surface for plate mounting. The contact surface continues for about 2.5 mm with a detent at 1.5 mm as standard plates are 1.5 mm thick. Two sides then taper slightly towards the center. A mechanical drawing of the switch is shown in Fig. 12. Two variants of the bottom housing exist: 5-pin and 3-pin. 5-pin switches include two additional plastic pins on the bottom intended to provide better stability when the switch is mounted directly to the PCB. Comparison of the footprints can be seen in Fig. 13. Most plate-mount hot-swapable keyboard designs provide these holes in the PCB for greater compatibility and one presented in this work is no exception.

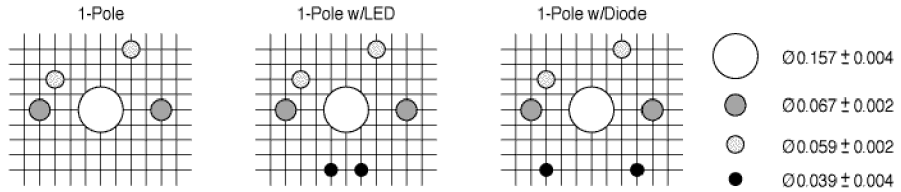
Another important characteristic is the actuation force or switch stiffness, measured in centi-Newtons [cN] or gram-force [gf], often denoted simply g. These two units are almost identical as 1 gram force is equal to 0.98 cN. The stiffness is determined by the used return spring and the stem profile also plays a role for non-linear switches. The nominal actuation force seen in datasheets usually corresponds to force required to bring the switch to the actuation point. A selection of various springs and stems can be seen in Fig. 14.

Other, less pronounced characteristics but still contributing to the overall feel are:

- Total travel - distance until the stem bottoms out on the shell
- Activation point - depth at which a keystroke is registered
- Bump position - where along the stroke is the bump located
- Overtravel - distance after activation before bottoming out



(a) Three-pin switch footprint.



(b) Five-pin switch footprint.

Fig. 13: Comparison of MX switch footprints [33]. Grid spacing 1.27 mm, hole dimensions in inches.

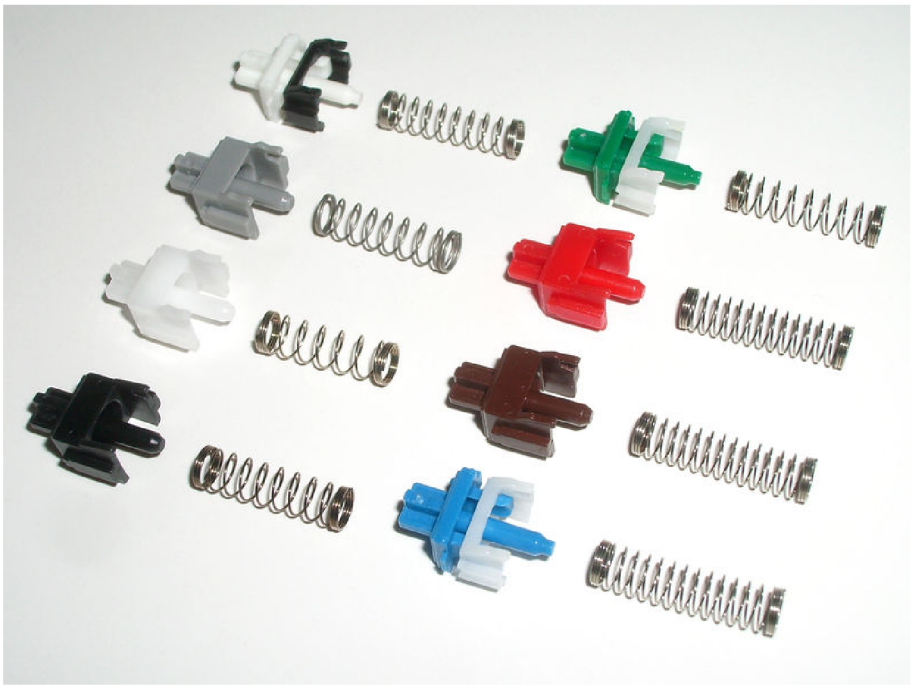


Fig. 14: Selection of various Cherry MX sliders and springs. [34]

Tab. 1: Switch characteristic comparison [22, 35–42]

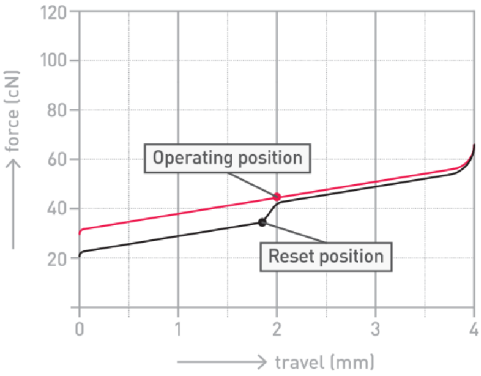
Switch	Type	Actuation force [cN/gf]	Pre-travel/ Actuation point [mm]	Total travel [mm]
Cherry MX Red	Linear	45 cN	2	4
Cherry MX Black	Linear	60 cN	2	4
Cherry MX Brown	Tactile	55 cN	2	4
Cherry MX Clear	Tactile	65 cN	2	4
Cherry MX Blue	Clicky	60 cN	2.2	4
Cherry MX Green	Clicky	80 cN	2.2	4
Kailh Brown	Tactile	60 gf	1.9	4
Gateron Phantom Red	Linear	45 gf	2	4
Keychron K Pro Mint	Tactile	67 gf	2.2	3.3

All measured in millimeters. „Correct“ combination of these characteristics is mostly determined by the typing style. An overview of basic switch specs can be seen in Table 1. Fig. 15 depicts the actuation force along the travel path. A composite chart for all colors is presented in Fig. 15d.

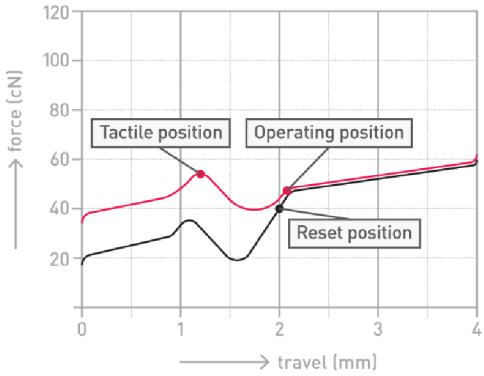
The contacts in an MX style switch usually consist of a static plate and a leaf that deflects based on stem movement. The contact pattern is cross point i.e. a vertical pad on one side and horizontal on the opposite. The contact pads are often gold or gold-plated. A close-up view of the contacts can be seen in Fig. 25. The service lifetime for modern Cherry MX switches is specified at over 100 million actuations [22].

Besides the aforementioned measurable characteristics, materials also play a role in the key feel. Material choices for the stem and housing have an effect on the operation smoothness and contribute to sound as well. Nylon and polycarbonate are the most common materials for housings. The stems are usually polyoxymethylene or POM, also known as acetal. POM has the advantage of low friction making it well suited for this application. Some switches also come with lubricated stems to further help with actuation smoothness. Adding lubrication is one of fairly easy and common ways to improve key feel.

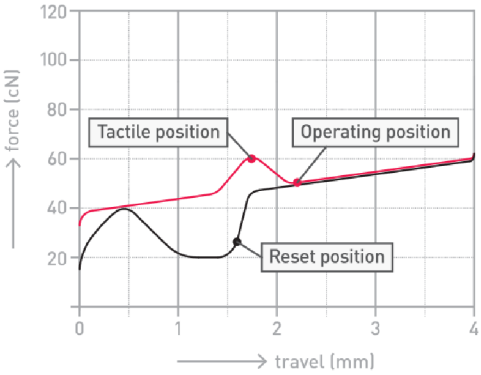
Mechanical switches are fairly loud compared to the rubber dome keyboards. While the acoustics are part of their attractiveness for some typists, increased loudness is not always desirable. Some, typically linear, switches are offered in quieter variants with damped stems as can be seen in Fig. 16a left. Additional methods of reducing noise include wrapping o-rings around keycap stems (Fig. 16b) or placing soft pads



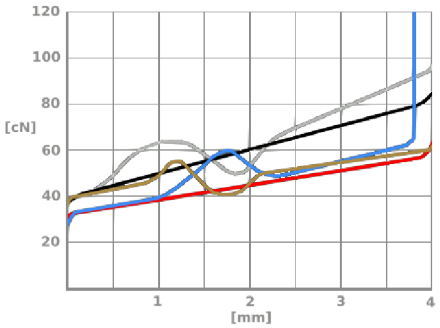
(a) Cherry MX Red actuation force [30]



(b) Cherry MX Brown actuation force [31]



(c) Cherry MX Blue actuation force [32]



(d) Composite chart. Plot line colors correspond to switch colors. Adapted from [43].

Fig. 15: Actuation force over travel distance.

(a) Damped MX Red and Black sliders. (b) Damping O-rings under keycaps.

Fig. 16: Noise reduction approaches. [44, 45]

on upper switch housings to further reduce sound when bottoming out, but a clicky switch will not be a good choice for a crowded office environment regardless of modifications.

The switch types can be identified by the color of their stem. Cherry originally introduced the linear switches with red, tactile with brown and clicky with blue stems. These main colors are well established and followed by other manufacturers as well. The available color palette has since expanded to include for example a differently weighted linear switch with a black stem or a tactile switch with white one. The amount of presently available switches from various manufacturers covers a good chunk of the color palette but the colors have retained their purpose for actuation feel indentification.

2.6 Keycaps

Keycaps sit on top of the switches and provide a comfortable interface for pressing the keys. The caps usually carry a printed information about the key function called a legend, but blank caps are sometimes used as well. Keycaps need to interface with the switch stem and are therefore specific to each stem design. From the aftermarket point of view, four main cap mounts are available corresponding to switch popularity: Cherry cross mount and low profile mount, Alps-style mount and Topre mount. Cheap rubber dome keyboards usually use caps with the stem molded in making them non-replaceable. Besides the mounting style, keycaps can be categorized by their profile and overall keyboard profile, their size and shape and construction material.

Keycap sizes are measured in units equating to about 19 mm measured at the base. Keys are usually rectangular, 1 U high with a slightly narrower top surface with the exception of ISO and JIS Enter keys. Most of the keys are capped by 1 U wide caps

with the modifiers being wider as depicted in Fig. 2. Common cap sizes are listed in Table 2.

Tab. 2: Common keycap sizes

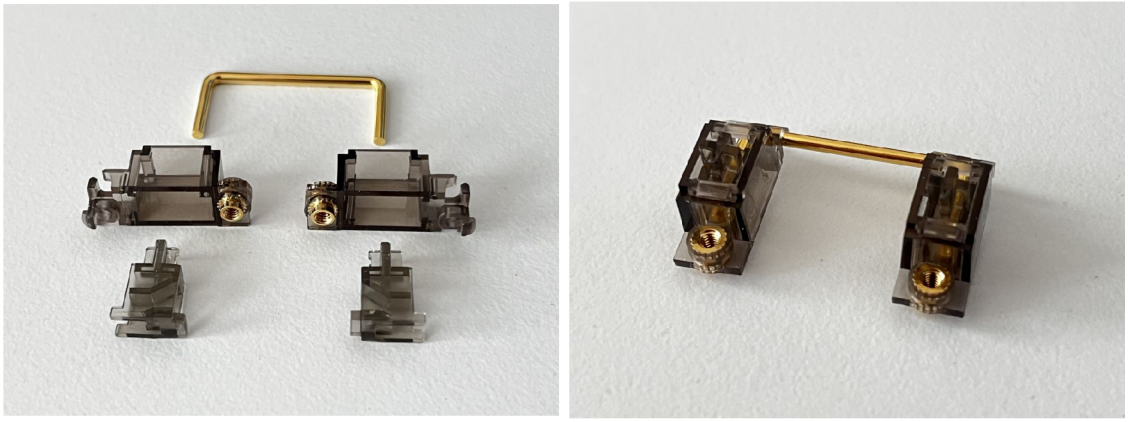
Width	Usage
1 U	General alphanumeric keys
1.25 U	Bottom row modifiers, ISO left Shift
1.5 U	Tab, | key in ANSI layout
1.75 U	Caps lock
2 U	Backspace
2.25 U	ANSI Enter, left Shift
2.75 U	Right Shift
6/6.25 U	Spacebar

Caps that are wider than 1.75 units require additional support or they would be very unpleasant if not impossible to press off center. This problem is solved by stabilizers, often termed „stabs“ in the keyboard community. There are several approaches to stabilizing the keys. Some approaches include additional dummy switches, guide shafts or wire stabilizers where a length of wire connects two points on the cap besides the center mount. Wire stabilizers are most widely used. For Cherry MX style caps the stabilizers usually consist of two additional sliders with the cross mount connected by a wire. Simpler implementations omit the sliders and use clips molded into the caps for retaining the wire. Adapters from MX cross to clip style exist.

The MX style slider stabilizers have a 6 unit spacing for the spacebar and 1.25 unit spacing for the other wide keys. Mounting options include clip-in PCB mount, clip-in plate mount or screw-in PCB mount. An example of a screw-in type stabilizer can be seen in Fig. 17. The stabilizers are often lubricated to improve smoothness and eliminate unpleasant ratting since the stabilizer stems loosely slide in a housing.

Caps on different keyboard rows can have different profiles or the entire keyboard base PCB can be angled or curved to provide a more ergonomic typing position. The basic row profiles are depicted in Fig. 18.

A completely flat profile is mostly used in space constrained situations such as laptop keyboards. Generally more common with low profile switches and caps.



(a) Wire stabilizer parts.

(b) Assembled stabilizer.

Fig. 17: MX cross mount 1.25 U screw-in PCB mount wire stabilizer.

The contoured profile is the most common in full height keyboards. Each row of keys has caps with a differing top and bottom angles in addition to the top angles differing between rows. This is done in order to simulate the curved profile on a flat PCB a and slightly even the distance fingers have to travel to the further rows. This setup does not allow for changing keycaps between rows, making certain alternate layouts such as Dvorak impossible without a custom printed set. Row numbering differs between cap manufacturers, some count row 1 from the top, others from the bottom. Top two rows and the two lower middle rows often share the same cap profile, resulting in 4 unique profiles. This however also tends to vary between manufacturers and sometimes between cap sets. A curved profile uses the same cap shape on all rows and instead the base layer is angled.

A staircase or stepped layout was more common on the early keyboards in 70s and 80s. The profile mimics a common typewriter setup with an angled board and horizontal key surfaces. The same result can be achieved with flat caps and angled switch stems.

Chiclet or island keys are very similar to the flat profile with the difference of added space between the keycaps. Chiclet style is only found in low-profile keyboards.

As with the rest of the mechanical keyboard community, keycap profiles are also dynamically evolving with the manufacturers experimenting with tweaking the profiles, creating new patterns in addition the established ones.

The individual keycaps usually have a profiled touch area as well. The touch profile is concave with the exception of spacebar which has a convex profile. Earlier caps had a spherical profile while modern caps usually have a cylindrical profile. Keycap sets with the spherical profile are still available but not very often found on OEM

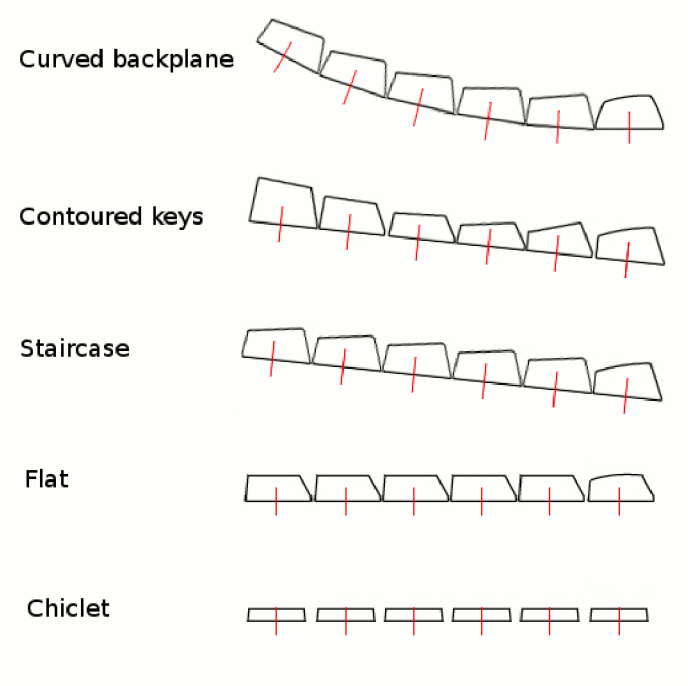


Fig. 18: Common keyboard profiles. [46]

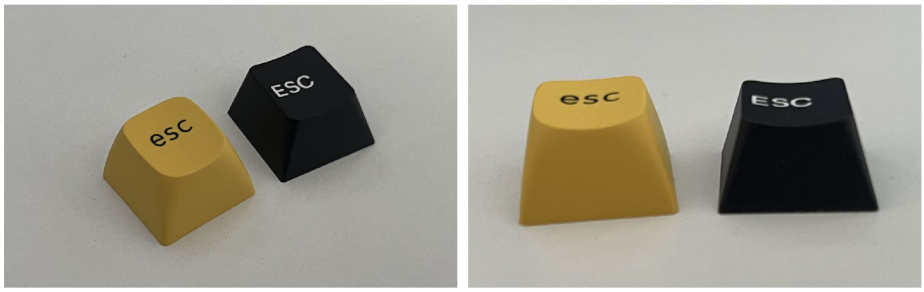


Fig. 19: Keycap profile comparison.

boards anymore. The difference is depicted in Fig. 19: Spherical profile on the left (yellow), cylindrical on the right (black). Both caps are double-shot PBT.

Legends carrying the function information are typically printed in an easily legible sans-serif font. The legend placement on the key corresponds to the modifier layer. Primary function is printed in the bottom left cap corner, shift-layer function in the top-left. For layouts utilizing the graph alt key, symbols accessible on this layer are printed bottom-right, and graph+shift layer symbols in the top-right corner. Alphabetical keys are an exception and are typically printed capitlaized in the top-left corner. Keyboards with legends printed for use with a regional layout often place the regional symbols in the graph layer space. The legends are sometimes printed in cap centers, nowadays mostly seen in single-layout custom cap sets. Alternatively for full height caps, legends can also be printed on the front face of the cap.

The legends are most commonly pad-printed but a wide variety of methods is used by the manufacturers: laser etching, engraving, dye sublimation, masking or direct molding in plastic also called double-shot molding.

Double-shot molding is a process where the legend is molded first and plastic for the keycap shell, usually of different color, is injected around the initial molding in a subsequent step. This process provides the best legend legibility and highest wear resistance though comes at a higher cost due to the increased complexity of molding. If a backlit legend is desired, the legend molding is done in translucent plastic. Alternatively the entire caps are molded in translucent plastic and subsequently painted with the legends masked off. The resulting effect allows the light to shine through the uncovered parts. Depending on the cap quality, the paint used is reasonably wear resistant but this finish has the least wear resistance overall.

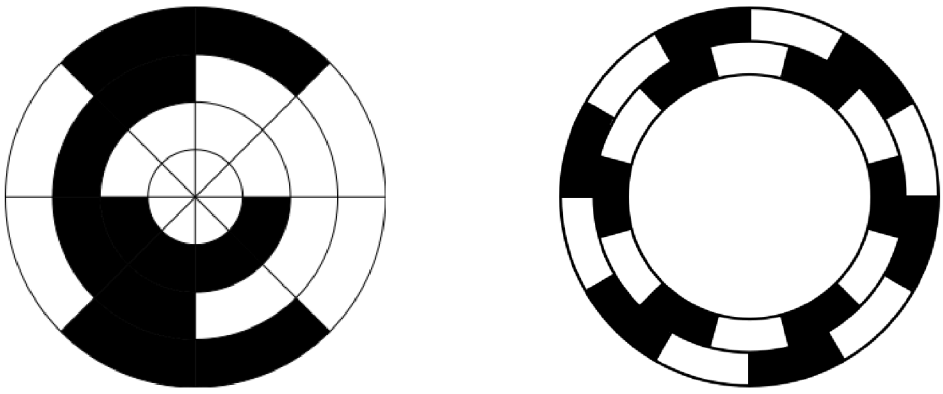
Keycaps are made of several kinds of thermoplastics. Acrylonitrile Butadiene Styrene, or ABS, keycaps are the most common as the material has good molding properties and acceptable wear resistance. Newer ABS caps often wear to a shiny finish. Older caps are more resistant but white-colored keycaps are prone to yellowing. Legends on ABS caps can be pad-printed (decaled), laser etched, engraved or the caps can be double-shot molded.

The second most common plastic is PVC, or Poly-vinyl Chloride, used in mass market cheap keyboards produced by OEMs like HP, Dell or Logitech. It has average mechanical properties. Legends on PVC caps are most commonly pad-printed and rarely laser-etched, stemming from their widespread use in cheap products.

PBT, or Polybutylene Terephthalate, is another popular cap material. It is harder and more durable than ABS but also more brittle and does not have as good molding properties. The plastic is especially prone to shrinkage. It is still used for double-shot caps as its other properties make the caps desirable even at a higher cost. The cost increase comes from the improved tooling required and higher than average defect rate in production. In addition to the other methods, hiher heat resistance of PBT allows dye sublimation legend printing; a process where dye particles are directly embedded into the plastic substrate. PBT keycaps are usually sold in aftermarket cap sets and not generally seen in OEM boards. They are often considered a premium feature.

2.7 Additional inputs

The USB input class specification is quite flexible allowing for a variety of input devices. Dedicated control surfaces such as audio mixing panels or video color grading consoles take advantage of this extensively. Non-button inputs are less



(a) A 3-bit gray-coded binary encoder wheel. [47]

(b) A 2-bit binary incremental encoder wheel.

Fig. 20: Encoder wheels.

often seen in general computer keyboards but some, usually custom-built, keyboards do include additional input devices. These include pointing devices, various linear sliders or rotary encoders. The integrated pointing devices are either of the pointing stick type utilizing strain gauges or miniature trackballs as the requirement for the keyboard to remain static while in use is a limiting factor. Rotary potentiometers are not very well suited for general purpose usage due to their limited travel. Encoders do not have such limitation and are seen included in keyboards in various forms ever more often.

Encoders

Encoders are devices that convert angular position or motion into a digital or analog signal. Two main categories exist: absolute encoders which always provide information about the angular position and incremental encoders which provide instant information about rotation speed and direction. Absolute encoders are more complex, costly devices that find their application primarily in precise industrial machinery. Functionality provided by incremental encoders is sufficient for general use as an input device.

Similarly to key switches, encoders use a variety of methods to achieve their task. Common encoder sensing technologies include magnetic, capacitive, optical or mechanical. The sensing works on the principle of detecting binary segments along a number of tracks. The segments are gray-coded, meaning only one bit changes between any two subsequent positions. An example of a three-track encoder wheel can be seen in Fig. 20a. Black segments represent 1 and white 0. Similarly to analog-digital converters, higher density contacts, i.e. more bits, provide higher resolution. The number of positions an encoder can sense comes out to 2^n where n is

the number of contacts [48]. The 3-bit wheel depicted can track 2^3 i.e. 8 positions and detect motion in 45 degree increments.

The simpler incremental encoders require only a two-bit wheel. For optical encoders, this is commonly done using two plates with radial slots where one is moving with the shaft and the other remains static. In mechanical conatact encoders the two tracks alternate between 1 and 0 with a 50% phase shift as shown in Fig. 20b. Moving the encoder effectively creates two square waveforms with offset phase. Rotation speed can be determined from the waveform frequency and phase shift between the tracks determines rotation direction [49]. The encoders may include a zero-angle mark which can be used to determine when a full turn is completed. It is therefore possible to obtain information about position from an incremental encoder but this information must be tracked separately. The denser the alternating segments the lower the speed required to generate same amount of pulses, resulting in finer control and better feel. Information about the number of segments is specified as pulses per revolution.

2.8 Communication Protocol

The keyboard controller detects keypresses and processes this information in a way that is understood by the host system – the next step is to transmit the data. Since keyboards are universal devices that can be used with a variety of hosts, the transmission is done according to a standard protocol. The next section takes a closer look at the major communication protocols in use with keyboards.

PS/2

As mentioned in the introduction, first standardization of the input peripheral interface was started by IBM on the PS/2 system. The interface even inherited its name from the shorthand system name. While the protocol was present on the earlier PC-AT line, PS/2 was the first lien of computers offering the same connector for both keyboard and mouse. The physical connector took shape of a 6-pin mini-DIN with two pins left unconnected. The four connected pins are data, ground, VCC and clock. Data and clock lines are pulled up to VCC. Connector colors are standardized as purple for the keyboard and green for the mouse. The interface is not designed to be hot-pluggable, i.e. the peripherals must be connected before the computer starts and should not be removed while it is running [50].

PS/2 implements bidirectional synchronous serial protocol. The clock signal in the range of 10–16.7 kHz is generated by the connected device. Data sent from a connected device to the host is read on the rising edge and communication from

the host is read on the falling edge. The host (computer) has control over the bus, connected devices can send data only when both clock and data lines are high signifying an idle bus. The host can request communication to the device by first pulling clock low, then pulling data low and releasing the clock line. Device-to-host communication is sent in 11-bit frames consisting of a zero start bit, 8 data bits followed by 1 odd parity bit and lastly a one stop bit. Host-to-device frames add one extra acknowledgement bit at the end. Host-to-device communication includes for example commands to toggle lock indicator LEDs or to set a scancode set [51].

Since the physical connector contiains two unconnected pins, some computers provide a single connector for both keyboard and mouse. Such connectors then require a splitter cable in order to use two devices at once. Using a splitter cable on a regular PS/2 port is not possible since the protocol only allows for communication between a single host and device.

The PS/2 standard is nowadays largely superseded by USB HID. Motherboard manufacturers however continue to provide the port, mostly for backwards compatibility. While USB is more practical for most usage applications, one advantage of the legacy protocol is the absence of polling which could theoretically lead to a decrease in input delay.

USB

Universal Serial Bus or USB for short is an industry standard codifying specifications for connectors, protocols, cables for serial communication between computers and a wide variety of peripherals. Introduced in 1996, it came about from a joint effort of Compaq, DEC, IBM, Intel, Microsoft and Norhern Telecom. The aim of this effort was to provide an easy to use, flexible, standard and easy to integrate interface that would support real-time data transfer and remain extensible. The standard specifications are freely available and continue to be updated and developed under the oversight of USB Implementers Forum. A number of revisions came out over the years since its inception, each bringing an increase in transfer speeds while remaining backwards-compatible.

Original USB spec defines two speed tiers or rates: 1.5 Mb/s or Low speed for low bandwidth devices requiring no cable shielding and 12 Mb/s or Full speed. The transfer speed is indicated by pulling data lines either high or low. Additional revisions added more rates from 480 Mb/s in USB 2, through 5 Gb/s in USB 3 up to 20 Gb/s in USB 3.2. Backwards compatibility is achieved by only using a subset of features common to both the host and the connected device. Computer keyboards do not require a large amount of bandwidth and generally fall into the Low speed

bracket. Features of revisions beyond 2.0 are not relevant to the presented work and will not be further elaborated upon.

Electrically, USB communication occurs over a twisted differential pair of wires. Two additional wires are usually present, carrying the bus voltage and ground. USB bus voltage is typically 5 V. The USB bus has a tiered star or tree topology, as depicted in Fig. 21. A single hub is at its root and each additional hub represents another tier. Up to 127 physical devices can be present on the bus [52].

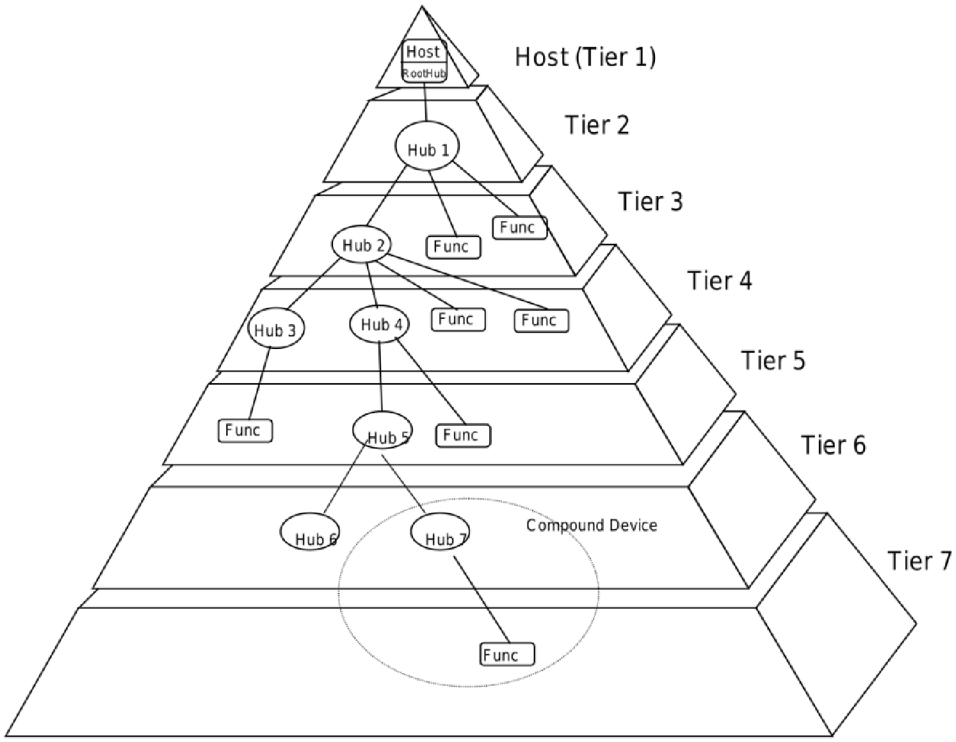


Fig. 21: USB bus topology [52].

A major advantage in the ease of use category is USBs ability to operate in Plug and Play fashion. Class-compliant devices do not need any prior configuration, are ready to be used as soon as they are connected and are hot-pluggable. Plug and Play achieves this by managing I/O and memory address ranges and IRQs, based on information retrieved from device descriptors stored in a segment of the device's ROM. Descriptors identify the device configuration, its interfaces, available endpoints and indicate belonging to a certain class. An example descriptor structure for an HID device can be seen in Fig. 22. A device can belong to multiple classes. This is possible since the class can be defined at the interface descriptor level and

not directly in the device descriptor even though it contains a class field. Each descriptor provides information as follows:

- Device descriptor: Information about the device; class, subclass, vendor, version etc.
- Configuration descriptor: Information about different possible device configurations.
- Interface descriptor: Specifies all available endpoints and optionally classes and subclasses. More descriptors can be specified under the class descriptor.
- Endpoint descriptor: Information about each endpoint type; required transfer type, packet size, polling interval etc.

Device classes are defined by the USB IF. An overview of defined classes as of January 7, 2022 is presented in Table 3. The first column lists the class code, second column lists whether is the class applicable to a USB device, interface or both and the last column offers a brief description of the class.

USB is a polled bus meaning the host initiates all transfers. Communication transactions usually involve up to three packets. A scheduled packet also called a token packet describing the type and direction of communication, USB device address and endpoint number comes first. The addressed device then selects itself and responds by sending its data or indicating that it has no data to send. Lastly, the destination may respond with a packet indicating a successful transfer. Communication stream between a host and a device endpoint is also called a pipe. USB defines two pipes, stream and message. Only the message pipe has a defined strucure and one message pipe always exists in all connected devices. This pipe is connected to endpoint number zero and provides information completely describing the device. This includes information about the standard, class and the device vendor. Several interface descriptors can share endpoint 0. Pipes have attributes defining their bandwidth, directionality, endpoint characteristics etc. Any number of pipes can be defined when a USB device is configured.

Since USB devices are hot-pluggable, the bus must be able to accomodate dynamic topology changes. All devices are connected to the bus via hubs as per the topology diagram in Fig. 21. The host queries hubs' status bits to determine device attachment or removal. Ports with attached devices are enabled and the attached devices are given an address using a control pipe at a default address. The host then establishes a control pipe at endpoint 0. Attachment notifications are sent out after determining the attached devices' function from its descriptor. These are subsequently handled by the host software appropriate for each function. When hub status bits indicate a removal, the host software is similarly notified and the appropriate software must handle the function removal. The activity of identifying devices

Tab. 3: Defined USB class codes [53]

Base class	Descriptor Usage	Description
00h	Device	Special: Use class information in the Interface Descriptors
01h	Interface	Audio
02h	Both	Communications and CDC Control
03h	Interface	HID (Human Interface Device)
05h	Interface	Physical
06h	Interface	Image
07h	Interface	Printer
08h	Interface	Mass Storage
09h	Device	Hub
0Ah	Interface	CDC-Data
0Bh	Interface	Smart Card
0Dh	Interface	Content Security
0Eh	Interface	Video
0Fh	Interface	Personal Healthcare
10h	Interface	Audio/Video Devices
11h	Device	Billboard Device Class
12h	Interface	USB Type-C Bridge Class
3Ch	Interface	I3C Device Class
DCh	Both	Diagnostic Device
E0h	Interface	Wireless Controller
EFh	Both	Miscellaneous
FEh	Interface	Application Specific
FFh	Both	Vendor Specific

and assigning addresses and detecting removals is known as bus enumeration. It is an ongoing activity due to USB devices' ability to be connected or disconnected at any time.

The USB protocol recognizes four types of data transfer using a set of uni- or bi-directional pipes:

- Control transfer: Used at connect time to configure the device, can be used for other purposes including controlling other pipes on the device.
- Buld data transfer: Sequential exchange of large amounts of data, variable bandwidth.
- Interrupt data transfer: Can be requested anytime, used for timely reliable data delivery, usually event notifications.
- Isochronous transfer: Real time data streaming using a negotiated bandwidth, no error correction.

Data flow through a pipe is generally dependent on communication taking place via another pipe. Communication takes place in time frames of 1 millisecond for low and full speed devices or 125 microseconds for high speed devices. The number of transactions in a frame is specified by the transfer type.

USB HID Class

Since the topic of the presented work is a keyboard, the next section describes the relevant USB class, the Human Interface Device class or HID for short. The HID device class descriptor identifies other HID descriptors and their sizes. HID devices contain report descriptors describing the generated data and its meaning, and optionally a physical descriptor providing information about which parts of the human body interact with the device. The report descriptor contains a set of items each describing e.g. a joystick position or a switch state. Host HID driver determines the size and content of data reports coming from the device based on information in this descriptor. An illustration of possible descriptor scheme for an HID device is presented in Fig. 22.

Since input peripherals can provide multiple functions, for example keybords with integrated pointing devices, the HID class in not strictly subclassed. Instead, provided data types and required protocls are described in the report descriptor. The subclassing facility is however used to indicate support for boot mode. Boot mode is, as the name suggests, intended for use in pre-boot BIOS environments and implements a simplified communication model. This is done to avoid the rather large overhead of parsing descriptors.

HID devices communicate with the host using either the default control pipe or an interrupt pipe. The control pipe is used for USB control data; transmitting data

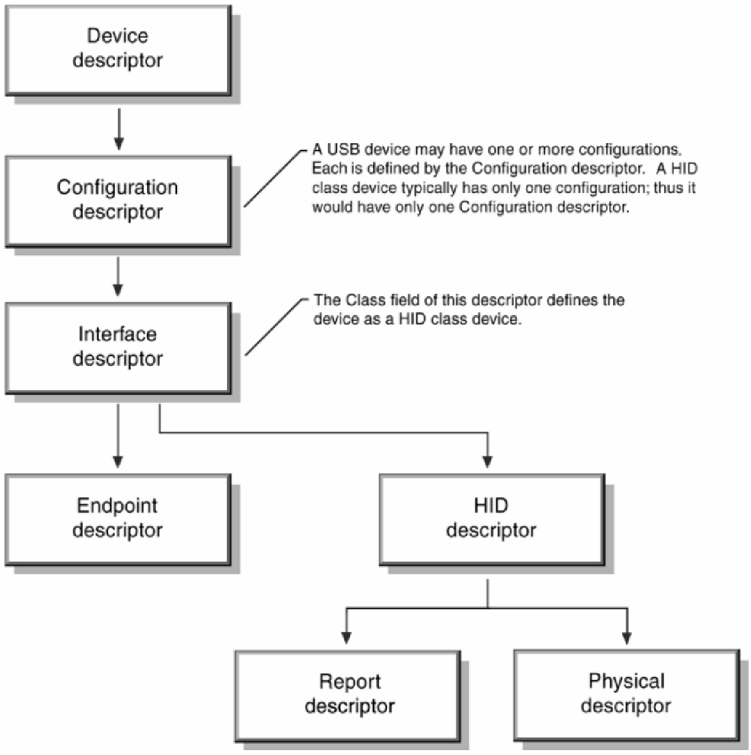


Fig. 22: Example descriptor hierarchy for HID class device [54].

when polled for a report or receiving data from host. Asynchronous data is sent sent to the host using the interrupt pipe. Additionally, the host may use this pipe to send low latency data to the device. Communication is established by first requesting a configuration descriptor from the device using the control pipe and then selecting the endpoint for interrupt pipe. The host driver saves the endpoint ID and the report interval. The report interval indicates the minimum time between report requests. For example a 10ms report interval indicates that a report can be requested every 10ms. Reading a report earlier than that could result in repeating an already sent packet or an empty response. A third optional pipe may be defined and, if present, is used by the host to transmit stream data to the device. Otherwise the host uses the default control pipe for transmitting reports to the device.

HID report descriptors do not take the usual form of data blocks of tables providing key-value combinations but rather are composed of so-called items. An item is one piece of information about a device. Items have a one-byte prefix specifying the item size in bits 0–1, item type in bits 2–3 and an item tag in bits 4–7. A byte of data follows. Optionally, items can contain additional data the size of which is determined by the item type. Short and long items are defined; short items can contain 0,1,2 or 4 bytes of additional data, while long items can contain up to 258. Long items are specified by a size value of 2.

Three item types are defined: main, global and local. Global items do not carry input data itself but rather describe it. Local items have the same function but different behavior in the context of state table creation. Main items can contain the following five item tags:

- Input: Data from a control e.g. position on axis, array of pressed buttons etc.
- Output: Data to a control e.g. setting an LED state.
- Feature: Data not presented to the user e.g. state of software toggle.
- Collection: A relevant grouping of the previous three tags.
- End collection: Tag denoting an end of item collection.

Items forming a report descriptor completely describe the incoming data; a host-side application knows how to handle the data and what is the data used for after parsing it. Several main items can be contained in the descriptor. To form a complete information package, the descriptors must contain the following elements:

- The aforementioned main item.
- Usage.
- Usage page.
- Logical minimum.
- Logical maximum.
- Report size.
- Report count.

A number of optional elements is defined as well, these include for example the specification of a unit for the incoming data. Usages are items supplying information about an input purpose. This allows for consistent assignment of functions between applications. General usages are grouped in pages e.g. joystick axes or keyboard button scancodes. The scancodes and other usage tags are defined in a supplementary document called HID usage tables for USB. The document specifies codes not only for keyboard keys but any other HID device as well, from mice through game controllers to specialized peripherals. The scancodes are assigned to a peripheral type using an 8-bit hexadecimal usage id. The scancodes themselves are 8-bit as well. The keyboard scancodes for alphanumeric, special character and function keys occupy the upper range from 01 to 64, with modifier keys in the lower part from E0 to E7. These codes are part of PC-AT, Mac and Unix standards, and should be implemented in the boot mode as well. The range from 65 to E0 specifies several different operations such as additional function keys or dedicated codes for operations like copying and pasting or media control. These features can be included on a keyboard as dedicated keys but are not part of any standard [55]. Where some early keyboards transmitted the ASCII representation of the pressed keys in order to bypass compatibility limits, scancodes merely indicate a key position. This allows

for dynamic mapping of characters independent of the character set printed on the keyboard keycaps.

Items in the report descriptor are parsed by the host-side HID class driver. Information is extracted by reading the items and storing the information in an item state table. An example of a parsed HID device is shown in Fig. 23.

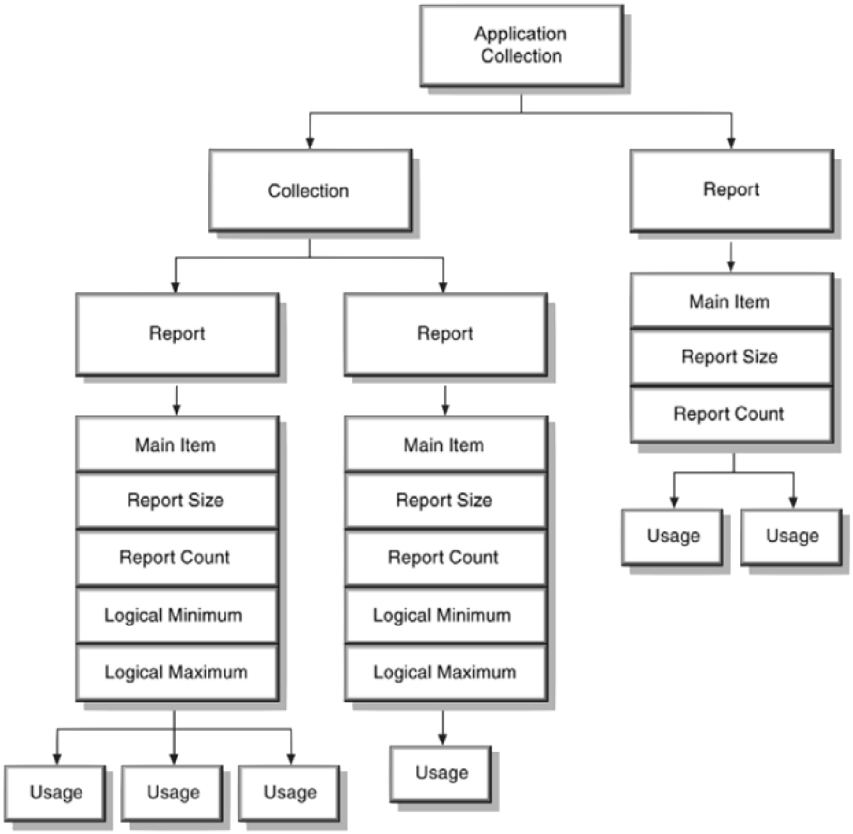


Fig. 23: Example of parsed HID descriptors [54].

The actual reports can be sent at most every USB frame as mentioned above. Reports can contain multiple packets but are limited in size based on USB speed. The limits are 8 bytes for low speed devices and 64 for high-speed. Each report must transfer meaningful data. In case of more input types on a single endpoint such as for a keyboard with integrated pointing device, a 1-byte report ID is sent at the start to identify which report structure is relevant to the incoming data.

As mentioned, the boot protocol is much simpler sending report packets as eight 8-bit bytes where the first byte encodes the modifier key status and bytes 2–7 are the read scancodes. Byte 1 is reserved in this mode [54].

3 Implementation

The keyboard created as part of this work uses Cherry MX style switches, is of TKL form factor and the provided PCB files allow for both ISO and ANSI layouts. A rotary encoder is included in addition to the standard keys. The keyboard has provision for controllable key backlighting. The key switches are mounted on a plate and are hot-swappable. The keys are topped by a cherry profile double-shot PBT cap set from Keychron. The board is powered by an ARM microcontroller running the QMK firmware. The following chapter covers the design process and parts used.

3.1 Tools

Creating a new custom keyboard is a quite involved task. Thankfully the active community makes available a number of tools supporting the creation process. Keyboard-layout-editor [56] can be considered a foundational one as it allows a quick and easy visualisation of any layout. Output of this tool can be then used in subsequent steps, making the process that much more straightforward.

After creating a key layout, the next step is to lay out a matrix for it onto a circuit board. Hand wiring a switch matrix is of course always an option, however with the advent of affordable printed circuit board production services an ever less appealing one. A number of tools is available for this task, from commercial solutions like EAGLE or Altium designer to free open-source projects like KiCad or even a web-browser solution like EasyEDA which also offers a convenient access to fab house catalogs for seamless ordering of prototypes. In line with the author's affinity for open-source projects, the presented board is laid out using KiCad. A number of custom keyboard specific component libraries and helper plugins is available for this tool as it is commonly used in the community. Of those, several were used in this realization: the switch footprint library by ai03 [57] and a plugin for laying them out by yskoht [58], the LED footprints come from the marbastlib library by ebastler [59].

Open-source software was used for the mechanical design side of the solution as well with the 3D models realized using FreeCAD software, and mechanical drawings for the plate and stacked case drawn in QCAD.

3.2 Switches

Key switches are the main component deciding the overall keyboard feel. The choice is highly subjective depending on indivdual preferences rather than any objective metric. The author prefers tactile switches on the stiffer side and the final choice was among three models: Keychron K pro brown, Kailh Box Burnt Orange and JWK Everglide Oreo. The basic switch specs are summed in Table 4.

Tab. 4: Comparison of considered switches [42, 60, 61]

Switch	Type	Actuation force [gf]	Actuation point [mm]	Total travel [mm]
Keychron K Pro Brown	Tactile	50	2	4
Kailh Box Burnt Orange	Tactile	70	2	3.6
JWK Everglide Oreo	Tactile	55	2	4

The keyboard is populated by Everglide Oreo switches, chosen mainly for their satisfying bump which begins practically right at the start of travel. The switches also have a nice characteristic sound when bottoming out. Their actuation force sits at 55 gf and they bottom out at 62 gf. The overall travel is standard at 4 mm with actuation point midway through at 2 mm. Both halves of the switch housing are made of smoky gray semi-transparent nylon with acceptable tolerances but slightly looser fit of the top housing with variation between individual switches. The looseness does not affect key stability noticeably but does have an effect on switch sound. The bottom housing is of the 5-pin variety providing two additional stabilisation pins so the switch can be mounted either directly on the PCB as well as on a plate. The housing provides space for mounting a 2-pin 2 mm LED. The stem is made of opaque white POM and combined with the light lubrication applied from the factory provides very smooth travel. The return spring is gold-plated. Keycap mount is standard cherry cross but seems to be overall slightly larger as two separate keycap sets had a little tighter fit. The switch and its internal components can be seen in Fig. 24.

The switch contact plate and leaf are stamped out of copper with gold contact points. A close-up of the contacts is shown in Fig. 25a. The closed cross-point can be seen in Fig. 25b.

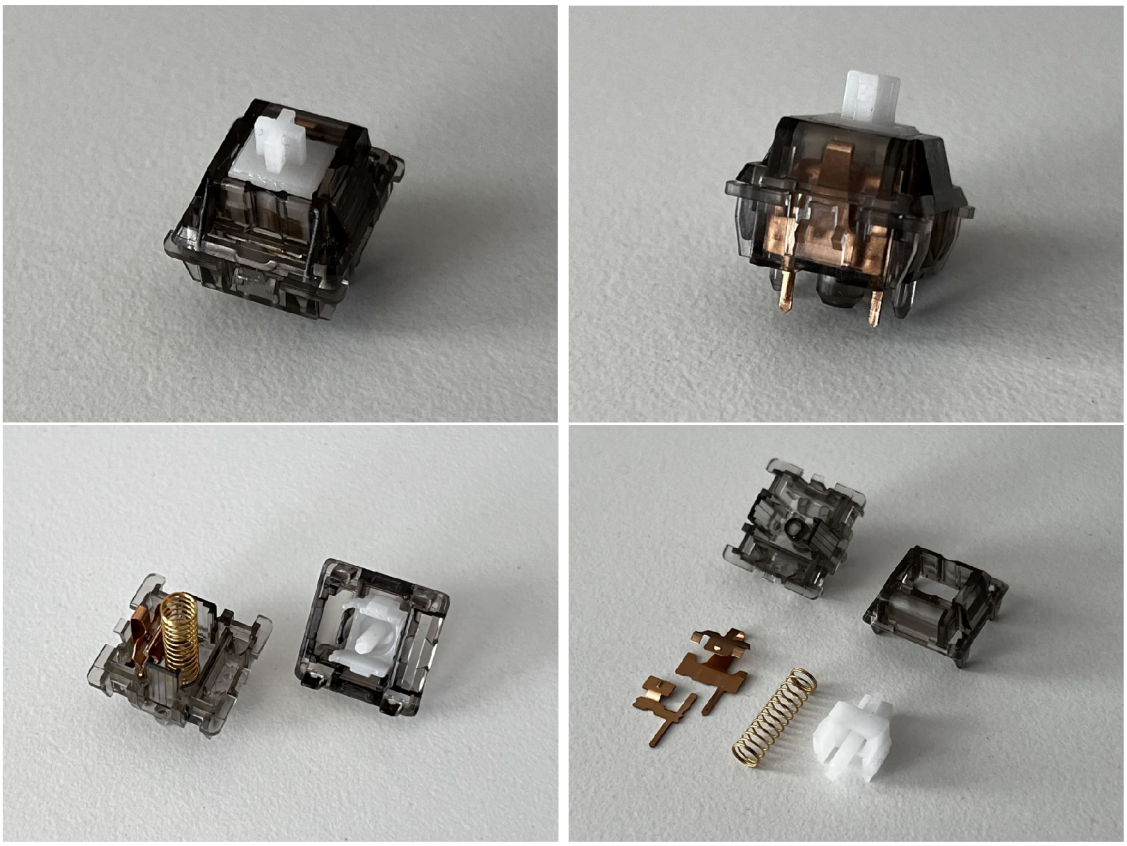
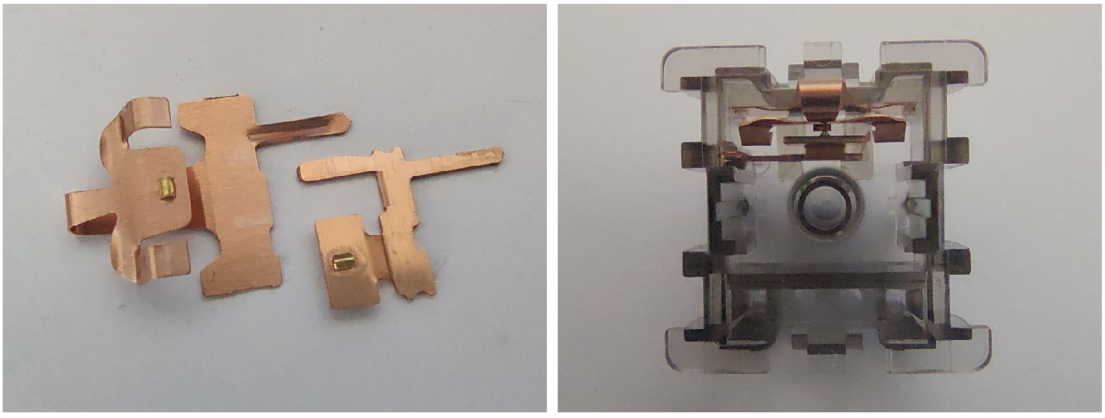


Fig. 24: JWK Everglide Oreo switch and its internal components.



(a) Contact leaf and plate.

(b) Contact cross point detail.

Fig. 25: Everglide Oreo switch contacts.

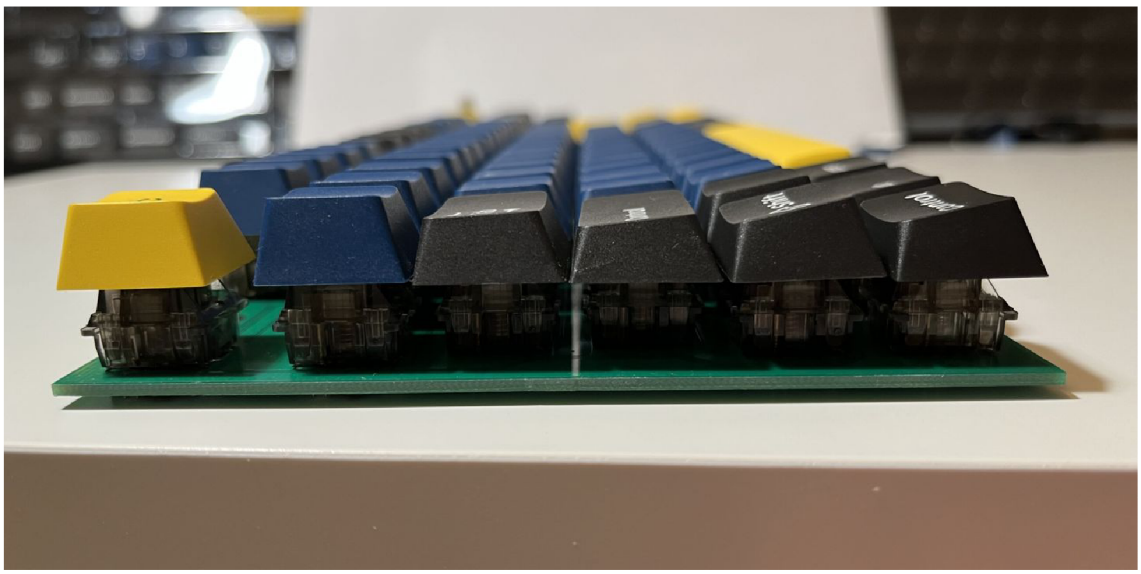


Fig. 26: The keycap profile.

3.3 Key caps

The caps are not only immediately visible, but are the part in direct contact with the user. It is therefore necessary to choose a good-looking and good-feeling set. A set of available options is outlined in Section 2.6. Aftermarket cap availability is fairly wide starting with close-to-OEM options all the way to artisan hand-made sets. Choosing a cap set is a fairly subjective matter similarly to preferred key feel. The obvious factors here are aesthetics, ergonomics and for the discerning typist also acoustics. Colorway and legend font fall into the aesthetics category, ergonomics include the cap profile both in terms of contact surface and overall row profile. Cap material covers perhaps all of the categories as it affects appearance, touch feel and the sound emitted when the cap contacts the switch housing at the bottom of a stroke. Sound is further also affected by the wall thickness and to some degree also the row profile.

The cap set chosen for the presented keyboard is made of PBT using the double-shot technique. The ISO version of the board is using a Keychron Royal set with a dark-blue and black colorway with yellow accents [62]. The keycaps have a contoured row profile and use the Cherry height profile putting the highest edges at only 9 mm. The profile as installed on the board is shown in Fig. 26. While not the preferred profile, the thick 1.5 mm walls and the colorway sufficiently make up for the lower height. The set has the standard cylindrical touch profile and large ISO style legend font. The caps are finished with a frosted texture offering a nice feel.

The presented keyboard is using screw-in type cap stabilizers styled similarly to the chosen switches as can be seen in Fig. 30a.

3.4 Additional inputs

In addition to the keys, a rotary encoder is included on the keyboard in the top-right position in place of the break key in standard layout. The encoder can be mapped to a variety of functions from simple volume control to scrolling on-screen content. The part used is of the incremental type with 2-bit gray coding, uses physical contacts and provides a resolution of 20 pulses per rotation. The service life is specified as 30000 cycles [63]. The encoder stem can be pressed to actuate a tactile switch which is wired as part of the keyboard matrix and can be assigned any function.

3.5 Layout

A TKL layout was chosen for the presented keyboard which means 88 keys for ISO version and 87 for ANSI, with standard layout and no number pad. PCBs for both ISO and ANSI versions are provided. The layout is further slightly more compact vertically which was achieved by reducing the gaps between key blocks to 0.25 U across the board. All key blocks are laid out as standard with the exception to the top right key in the navigation block, usually a pause/break key, which has been replaced by a rotary encoder that can be mapped to any function. This is done in attempt to better use the otherwise mostly useless key in an easy-to-reach position. The encoder can be pressed down to actuate a built in switch so the change comes at no cost to the number of mappable keys.

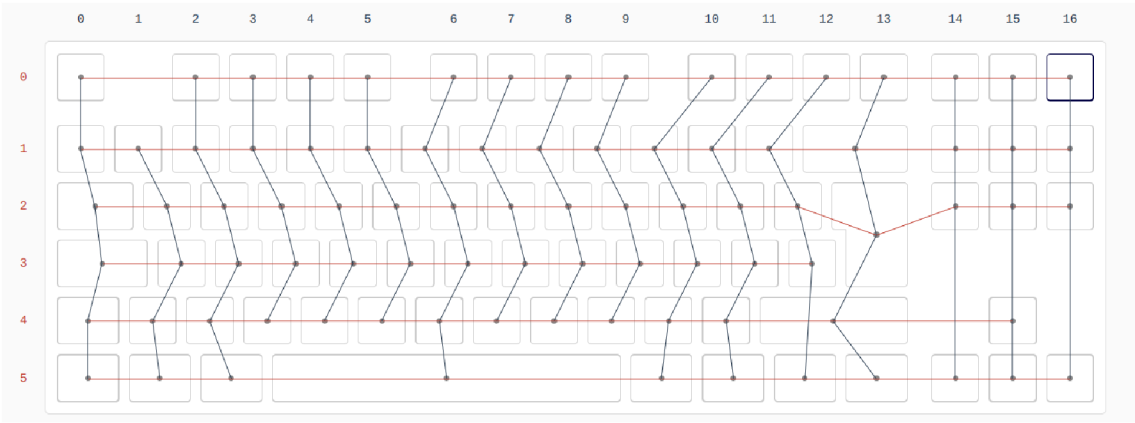
The logical layout provided is QWERTY with additional functions such as media and backlight control enabled on a function layer accessed by the Fn key. Additional key maps and layers can be defined by modifying the firmware sources. The keyboard also supports on-the-fly remapping using the VIA tool [64].

Key matrix

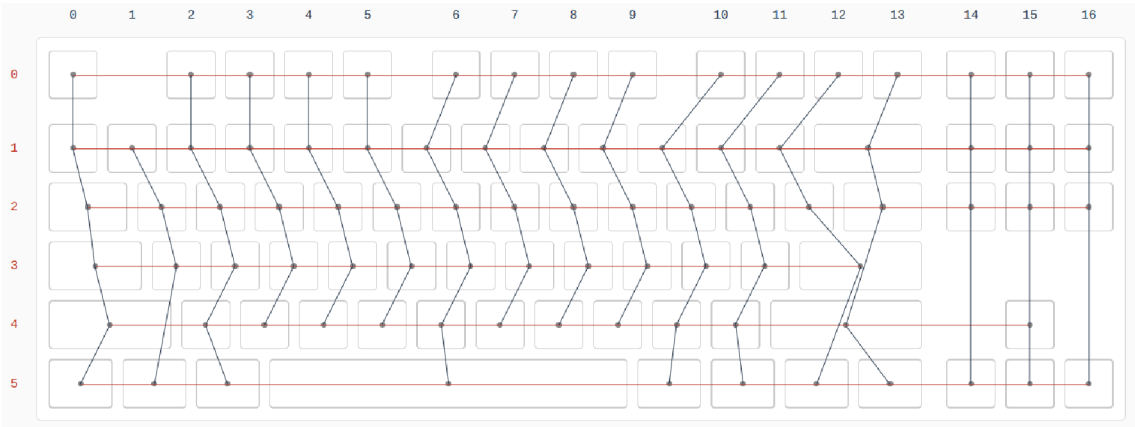
The keys are arranged in a matrix of 6 rows and 17 columns as can be seen in Fig. 27. Starting column 9 at the top row instead of row 1 resulted in one less necessary column saving one trace and an I/O pin. The matrices and boards for both layouts are the same save for differences in key placement. The matrix layout visualization was created using the kbfirmware tool [65].

3.6 Controller

A keyboard controller is responsible for interpreting pressed keys, constructing and sending messages containing the resulting scancodes to the host. This job is given to a more or less capable microcontroller ranging from mass market bare-bones cheap



(a) ISO layout key matrix.



(b) ANSI layout key matrix.

Fig. 27: Key matrix connection.

Tab. 5: Considered microcontrollers [67–69].

Controller	ATmega32U4	AT90USB1286	STM32F303xC
Architecture	AVR 8-bit	AVR 8-bit	ARM 32-bit
Clock Speed	16 MHz	16 MHz	72 MHz
Voltage	5 V	5 V	3.3 V (some pins 5V capable)
Flash size	32 kB	128 kB	256 kB
EEPROM size	1 kB	4 kB	4 kB
Operating memory	2.5 kB	8 kB	48 kB
Digital IO	25 pins	46 pins	37 pins

epoxy blob units to powerful ARM MCUs with plenty of I/O and processing power. More refined dedicated solutions exist as well, aimed mainly at higher-end OEMs, such as Holtek HT82K94 [66]. Advanced microcontrollers enable complex functionality such as addressable lighting, embedded information screens or additional input devices such as encoders or toggles thanks to the USB protocol flexibility. In the DIY/custom community, general RISC MCUs are the primary choice. Most manufactuers have a model with USB support available and the selection is wide.

On the 8-bit AVR side ATMega 32U4 is one of the popular controllers but it is limited by its smaller RAM and flash size. The MCU is used standalone or on an Arduino Pro Micro dev board. A more powerful alternative devboard is the PJRC Teensy 2++ with AT90USB1286 MCU.

The ARM side has a very broad selection of applicable microcontrollers depending on particular needs for I/O size, storage or performance. Custom keyboards often utilize offerings from the popular ARM Cortex-M line from manufacturers like ST Micro and Freescale/NXP. Firmware compatibility is another criterion in controller choice since self-written firmware is not often utilized given the existence of a powerful alternative in TMK/QMK and their various forks. A comparison of considered MCUs is presented in Table 5.

The ARM-Based STM32F303CCT6 was chosen for the presented keyboard. The controller offers increased flexibility, the option for DFU programming via USB connection, a usable amount of I/O and is fully compatible with the QMK firmware. The additional resources allow for features like multiple modifier layers and programmable backlighting.

3.7 Firmware

Firmware running on the microcontroller is responsible for servicing the keyboard functions. There are several established community projects besides the option to write own firmware from scratch. The most popular open source keyboard firmware project is QMK [70]. The firmware is compatible with both AVR 8-bit microcontrollers as well as the more powerful 32-bit ARM STM32 ones. The firmware can, and sometimes is, ported to additional microcontrollers but as it stands, the mostpopular choices for keyboard controllers are covered. QMK is short for Quantum Mechanical Keyboard and the project is a community effort to provide flexible firmware for a variety of input devices from keyboards though mice to MIDI peripherals. Originally forked from a keyboard firmware project of Jun Wako called TMK for the purpose of extending support for Olkb Planck ortholinear board by Jack Humbert. The project continued to diverge and add more fetures and in 2015 was renamed to QMK and established as standalone [71]. The new project started gaining traction as a community effort in contrast to the original TMK which is mostly managed by the original author with little community involvement. TMK is still used in a number of custom keyboard designs but the communities remain isolated in single-topic projects. QMK on the other hand embraces a philosophy where anyone is welcome to power a keyboard with the firmware, configurations and keymaps are stored and shared centrally in a monolithic repository [70]. Code is freely shared between the projects as appropriate.

A number of tools exist to aid with QMK usage. From the basic ones there is command-line interface wrapping commands for creating, compiling a flashing the firmware to various keyboards. For users that wish to configure an existing QMK-compatible keyboard and do not feel comfortable with CLI tools, QMK provides the QMK toolbox. The toolbox is a graphical front-end for flashing various supported microcontrollers available for MS Windows and macOS. Keymaps can be easily configured using the online QMK configurator. The configurator provides a convenient graphical representation of the keys as well as the defined matrix. Assigning functions is as straightforward as clicking the key position and choosing an action from the provided list. The new map can then be compiled and downloaded for flashing [72]. Using these two tools enables a user of nearly any technical level to customize their keyboard and enjoy it to the fullest.

Additional support tools exist for advanced users as well, for example a tool for directly translating a keyboard-layout-editor output to a QMK JSON layout format.

Due to its great flexibility and active community support, QMK was chosen as the firmware for the presented keyboard.

QMK setup

Working with the QMK CLI tools is very straightforward as well, the initial setup creates a full build environment for the supported platforms. This includes checking and installing dependencies such the compiler toolchains and flashing utilities. Udev rules setting proper permissions for the devices are conveniently included as well. To power a keyboard with the QMK firmware, a new project with the board's information must be created. This is done by simply creating a subfolder under keyboards/ in the checked out qmk_firmware repository. Alternatively, a qmk CLI command `new-keyboard` is provided which interactively pre-fills some information.

QMK config

QMK firmware provides almost infinite customization possibilities. Configurations for various boards are handled as projects, one per board. The files included in every project are: feature configuration, build rules and a readme file providing information about the board. A subdirectory keymaps/ containing, as the name suggests, the key maps has to be present as well. Any number of maps can be defined, stored in subdirectories of keymaps/. Each project can also contain additional subdirectories of modifications, each with its own set of rules and configs.

The board setup such as matrix pin mapping and enabled features is configured via header files or alternatively in a data-driven fashion using JSON files. Both options are supported at the moment but data-driven configuration is encouraged for new keyboards. Header file definitions take precedence when compiling a firmware that includes both. The basic hardware configuration as well as feature configuration both reside in a file called config.h or info.json for the data-driven approach:

- USB information – vendor and product IDs.
- Key matrix information – row and column count, MCU pin assignments.
- Anti-ghosting diode direction.
- Key repeat and hold behaviors.
- Split keyboard connections and behaviors.
- Various other pin assignments for the enabled features – Audio, backlight, RGB control, additional controls, displays, etc.

A comparison between the methods can be seen in Listing 1 and Listing 2 for the header file and info.json respectively. Listing 3 shows the definition of a simple 2x4 matrix.

Listing 1: Example config.h snippet

```
#define VENDOR_ID 0x7ffb
#define PRODUCT_ID 0xa9f9
#define DEVICE_VER 0x0002
#define MATRIX_ROWS 6
#define MATIX_COLS 17
#define MATRIX_ROW_PINS { A7, B0, B1, B2, B10, B11 }
#define MATRIX_COL_PINS { A2, A3, A4, A5, A6, B12, B13, B14,
    ↪ B15, A8, A9, A13, A14, A15, B3, B8, B9 }
#define DIODE_DIRECTION COL2ROW
```

Listing 2: Example info.json snippet

```
{
    "usb": {
            "device_version": "0.0.2",
            "pid": "0x7FFB",
            "vid": "0xA9F9"
        },
    "matrix_pins": {
        "cols": ["A2", "A3", "A4", "A5", "A6", "B12", "B13", "
           ↪ B14", "B15", "A8", "A9", "A13", "A14", "A15", "
           ↪ B3", "B8", "B9"],
        "rows": ["A7", "B0", "B1", "B2", "B10", "B11"]
    },
    "diode_direction": "COL2ROW"
}
```

Listing 3: Example info.json matrix definition

```
{
    "layouts": {
        "LAYOUT_ortho_2x4": {
            "layout": [
                { "matrix": [0, 0], "x": 0, "y": 0 },
                { "matrix": [0, 1], "x": 1, "y": 0 },
                { "matrix": [0, 2], "x": 2, "y": 0 },
                { "matrix": [0, 3], "x": 3, "y": 0 },
                { "matrix": [1, 0], "x": 0, "y": 1 },
                { "matrix": [1, 1], "x": 1, "y": 1 },
                { "matrix": [1, 2], "x": 2, "y": 1 },
                { "matrix": [1, 3], "x": 3, "y": 1 }
            ]
        }
    }
}
```

The layout definition is a list of dictionaries describing each key. The key description dictionaries have required elements specifying the absolute position in x and y coordinates and the key position in the matrix. The matrix position is a list of coordinates [row, col]. The coordinate system starts in the upper left corner, the x axis has its positive direction to the right and the y axis down.

Build options are contained in a make file rules.mk. Build options include directory layout setup, adding extra libraries or source files, specifying various key layouts, firmware format or optimizations. Feature configurations can be specified here as well such as RGB light driver. As program storage space is limited in MCUs and QMK functionality so broad, any functions or fatures that are not utilized can also be disabled in this makefile. The feature options control features such as bluetooth support, audio, midi or mouse pointer control. Anything that is not integral to keyboard operation can be excluded from compilation to save program space. The options flags can also enable debug features, custom matrix scanning routines or NKRO support. USB endpoints can be enabled or disabled as required here as well since they are a limited resource in the MCUs. The configuration set in directly in the keyboard directory is considered default. Additional configurations, provided in subdirectories of the keyboard directory, can partially or completely override these settings.

Key maps

The mapping of functions to the keys is usually specified in a file `keymap.c`. This file can also contain custom functions such as servicing additional features. The keymap is specified as an array of layers. Each layer must contain a key code for each key defined in the matrix. The keycodes are a list of arguments to a function named the same as the layout definition in info.json. A keymap for assigning keycodes to the matrix from Listing 3 can be seen in Listing 4:

Listing 4: A keymap.c file for the example 2x4 layout.

```
#define _BASE 0
#define _FN 1
const uint16_t PROGMEM keymaps[][MATRIX_ROWS][MATRIX_COLS] = {
    [_BASE] = LAYOUT_ortho_2x4(
        KC_1,    KC_2,    KC_3,    KC_4,
        KC_5,    KC_6,    KC_7,    MO(_FN)
    ),
    [_FN] = LAYOUT_ortho_2x4(
        KC_MPRV,   KC_MPLY,   KC_MNXT,   KC_MSTP,
        KC_VOLU,   KC_VOLD,   KC_TRNS,   MO(_FN)
    )
};
```

The example illustrates basic keycodes, layers and multimedia functions. When defining layers, it is a good practice to give the array elements a readable name. Another good practice which helps with readability is to visually organize the keycodes as they appear on the board.

In the base layer at position 0 keys are mapped to output numbers 1 – 7 and the rightmost bottom key switches to the function layer. In the function layer the top key row maps to media playback controls: previous, play/pause, next and stop. The lower row controls volume up and down. The seventh key contains a „transparent“ keycode which means that function from the lower layer will be used.

QMK includes a wide variety of keycodes from basic functionality to the option to define macros and custom key sequences or sending a code with modifier in one stroke. A so-called magic keycodes can be used to change the keyboard behavior while powered on. The available options include changing key functions, e.g caps-lock key emits a control modifier code, changing the default layer, enabling debug mode or even swapping key functions. Programmable buttons that do not correspond to scancodes but can be sent as as events to be processed host-side are also available. A special category are the so-called quantum keycodes, usually mapped to a specific key combination, which can reboot the keyboard controller, reinitialize persistent memory or even reflash the keyboard firmware [73]. This feature is called bootmagic in QMK.

Should the demo keyboard contain an encoder, its mapping could be defined in keymap.c as shown in Listing 5:

Listing 5: An example encoder function assignment.

```
...
#if defined(ENCODER_MAP_ENABLE)
const uint16_t PROGMEM encoder_map[][NUM_ENCODERS][2] = {
    [_BASE] =  { ENCODER_CCW_CW(KC_VOLU, KC_VOLD) },
    [_FN] =  { ENCODER_CCW_CW(KC_MS_WH_UP, KC_MS_WH_DOWN) },
};
#endif
```

The encoder mapping is again an array with elements for each layer. The mapping is an array as well, allowing assignment of function to more encoders as additional elements. Here the encoder controls volume in base layer and acts as a mouse wheel in the function layer. The mapping is wrapped in a define guard so that the code is not compiled if encoder support is turned off as a build option.

An example configuration of an SK6812 RGB backlight for the demo keyboard can be seen in Listings 6 and 7:

Listing 6: Enabling the RGB backlight feature in rules.mk.

```
...
RGBLIGHT_ENABLE = yes
RGBLIGHT_DRIVER = WS2811
```

Listing 7: Backlight configuration in info.json.

```
{
...,
    "rgblight": {
        "pin": "B4",
        "led_count": 8,
        "brightness_steps": 16,
        "hue_steps": 16,
        "saturation_steps": 16,
        "animations": {
            "alternating": true,
            "breathing": true,
            "rainbow_mood": true
        }
    }
}
```

QMK provides drivers for the WS2811 and APA102 addressable RGB LED families. The SK6812 family utilizes the WS2811 driver as the components are related. A number of effects for the backlight comes pre-programmed as well, with a selection shown in the listing under the `"animations"` key. Routines for unused effects can be optionally excluded from compilation using `#undef` statements to reduce the firmware size. A number of wrapper functions is provided and documented for the purpose of creating own effects if desired.

Rather than using RGB, QMK controls the backlight colors using a Hue, Saturation, Value or HSV model. The model defines a color circle, shown in Fig. 28. Value of the Hue parameter moves around the circle, Saturation moves between the inner and outer sections choosing the color intensity and Value sets the brightness. Increment steps when changing these values are defined in info.json. Keycodes are defined for controlling the RGB lighting. These include enabling or disabling the lights, the mentioned HSV parameter tuning as well as the option to jump to a specific effect or cycling through the enabled effects.

As with all QMK options, RGB lighting is extensively configurable with options to assign specific effects or modes to a layer or using the lights for status signaling. As the LEDs pass the inforamtion packets along the chain they are connected in, the packet order can be modified if the chain does not correspond to order of the key

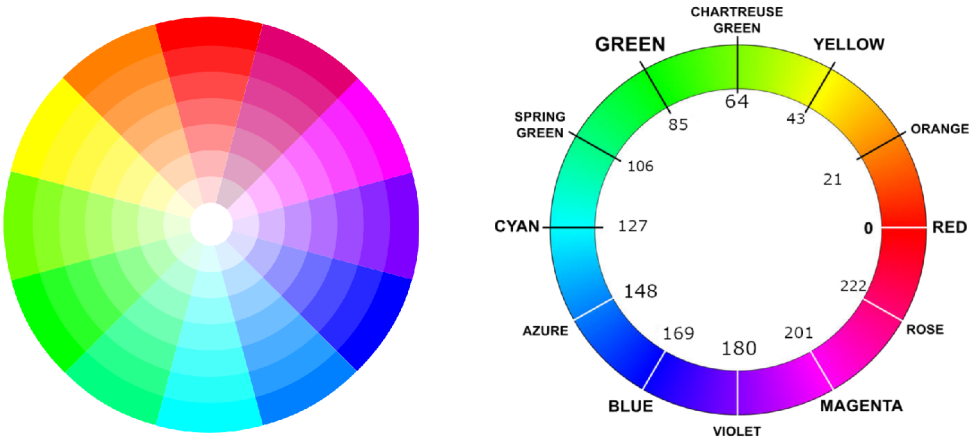


Fig. 28: HSV model color wheel [74].

matrix. A separate LED matrix can be defined in the config header file allowing for an arbitrary LED ordering. The header file can also contain additional configuration such as the maximum allowed brightness, default HSV values or effects upon reset or the option to turn the backlight off when the host computer goes to sleep. Some effects have configurable parameters that can be set in the header file as well [74].

VIA

VIA is a tool for confguring QMK compatible keyboards on-the-fly without the need to re-flash [64]. The tool is available in Chromium-based web browsers for maximum accessibility. Its functionality is achieved through the bootmagic feature. Some setup is required when creating a QMK keyboard to use it with VIA: The `VIA_ENABLE=yes` directive must be specified in build options, the keyboard must have a unique USB IDs for vendor and product and must have a product name defined. Further, a default keymap must be provided with the desired amount layers. The keymap is present so that known values get assigned to the keys on startup and can be entirely transparent. Enabling VIA support also limits the use of QMK features that change the `qmk_keycodes` enum [75].

To use the VIA configurator, a key matrix information must be provided. The matrix configuration is of the keyboard-layout-editor format, and must contain the positions in row,column format. Additional information can be provided here as well, such as alternate layouts with labels or encoder positions. Similarly to QMK, these layout files for supported keyboards are stored in a repository. Using a custom keyboard is possible after providing this configuration file to the configurator directly.

Operation internals

The QMK firmware begins its execution loop by first setting up the platform. These tasks include setting up timers, interrupts and I/O ports. Next is the protocol initialization which sets up the USB pipes and endpoints. This part differs between microcontrollers depending on the compile-time settings used. AVR MCUs run on the lufa platform while ARM parts use ChibiOS RTOS. After the initial setup is complete, the controller enters the main event servicing loop.

Internally, the event servicing state-machine in QMK can run at a maximum tick rate of 1KHz. The loop first checks for events without keycode processing such as updating the RGB light matrix. The key matrix is scanned next and keycode events processed. The matrix state is saved between scans and changes are determined by comparing the current state with the previous one. The matrix state is internally represented as an array of rows with length equal to the number of columns. The event scan result for the demonstration 2x4 keypad with the top rightmost key pressed can be seen in Listing 8.

Listing 8: Matrix scanning result

```
{
    {0,0,0,1},
    {0,0,0,0}
}
```

If a state change is detected, the resulting event is processed. Based on the enabled features, this is done in several steps. A simplified flow can look as follows:

- Pre-process combos.
- Map to a keycode defined for the position.
- Update typing speed information.
- Process a macro.
- Process layout changes.
- Process backlight.
- Process key overrides.
- Process RGB.
- Process programmable buttons.
- Detect and process Quantum-specific keycodes.

The list can be shorter or much longer based on the enabled modules and features as mentioned. If at any point during the processing a function returns `false`, the processing is halted. After all steps complete, a post-process routine is called which can be used to handle any required cleanup [76].

3.8 Board and other components

The PCB for the keyboard is realized in two parts: the main board carrying the switches, LEDs and MCU, and a daughterboard containing the USB interface and voltage regulator. This approach was chosen to enable greater flexibility when designing a case. The daughterboard can be mounted along any of the sides and the main board can be tilted to any angle without compromising the ergonomics of the connector.

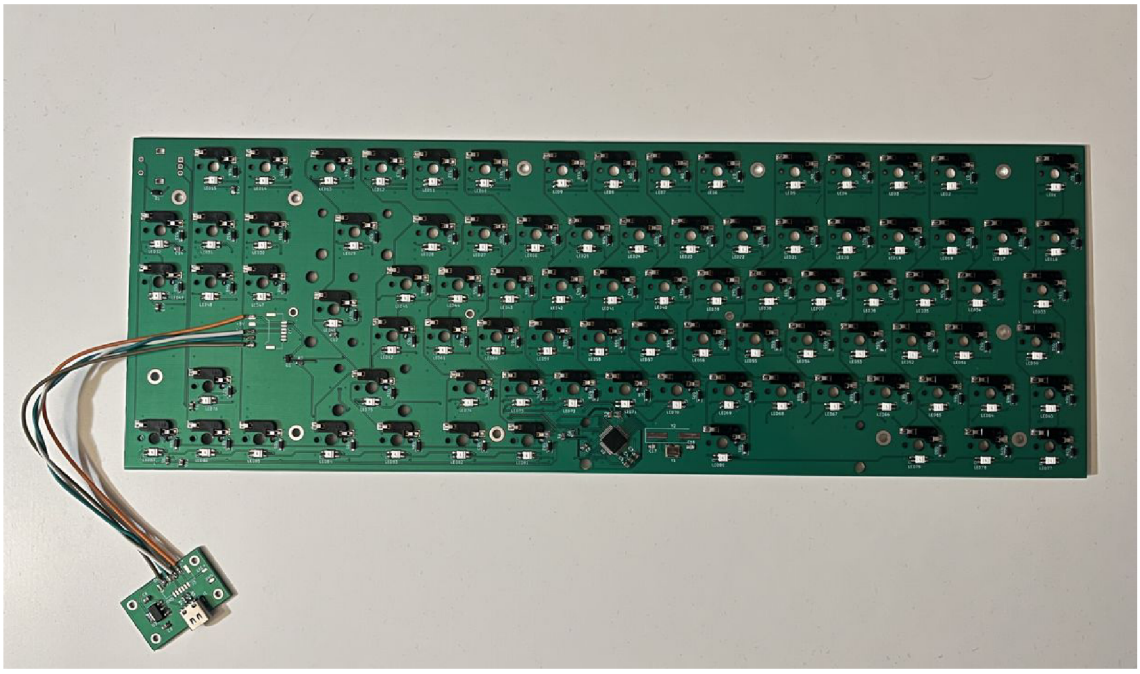
The connector board contains a through-hole mounted USB-C receptacle, a TS1117 3.3 V low-dropout voltage regulator in an SOT-223 package with its decoupling capacitors and footprints for wires or 6-pin 1.25 mm pin pitch SMD connector.

The main board contains a few different components. The most numerous are of course the switch sockets and associated diodes along with the key backlight LEDs. Next there is the MCU in an LQFP-48 package placed in the lower middle part, its decoupling capacitors, boot option and reset buttons, some pull-up and pull-down resistors, and a 12 MHz crystal oscillator. Two footprints are provided for the oscillator giving a choice between a four-pad grounded-case low-profile one or an SMD HC49 package. A single SOT-23 package p-channel MOSFET acts as a high-side switch for the backlight power. The full board BOM can be seen in Appendix A Table 6. A view of the board populated by SMD components is shown in Fig. 29.

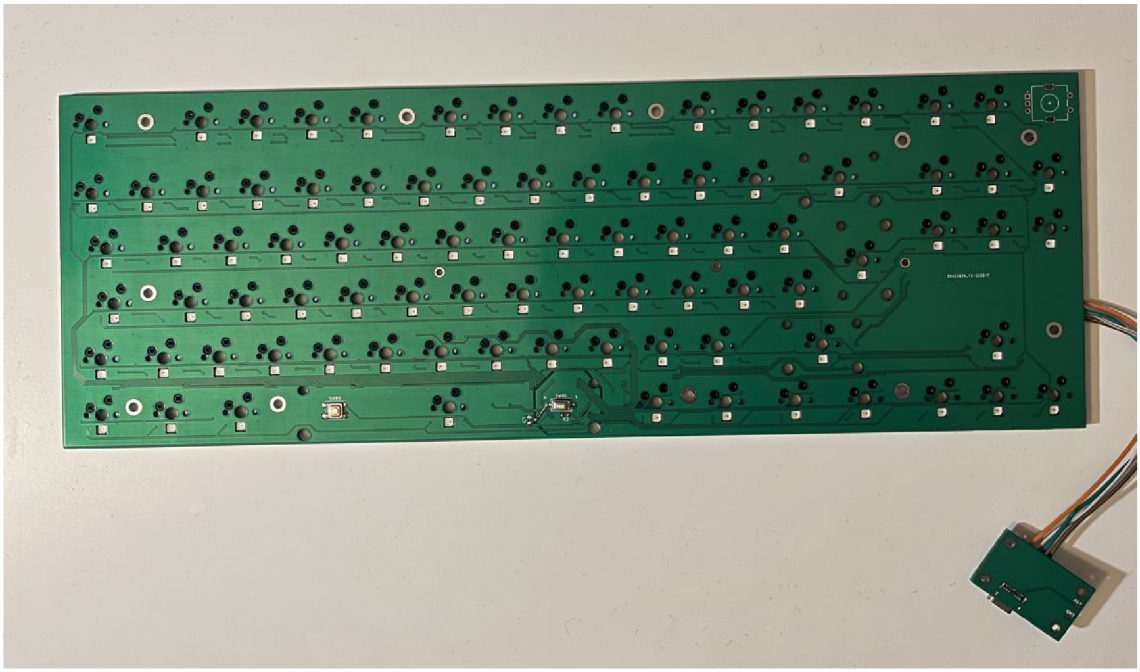
The anti-ghosting diodes used are 1N4148 in SOD-123 package. This diode type is widely used in keyboards and other signal switching applications due to its very fast recovery time of 4 ns [77]. The diode is available in a variety of packages, SOD-123 was chosen as a number of them were on hand and the footprint is is still hand-solderable. All other passive components are of the SMD variety in an 0805 package, again chosen because of general availability and the possibility to solder manually while being space-economical. The switch sockets are also SMD, mounted on the bottom side of the board. They are made by Kailh Kaihua [78]. The sockets do add to the board design complexity but the trade-off is a greater flexibility when setting up the keyboard.

3.9 Key underglow

The key backlight is provided by addressable RGB LEDs. The light color and pattern can be set in firmware and dynamically adjusted. The LEDs used are SK6812 in the mini-e package. The package is compact, measuring only 3.2 mm by 2.8 mm without leads, reverse surface mount. The PCB needs to have cutouts for the light to shine through, but no parts protrude above the board surface. This is

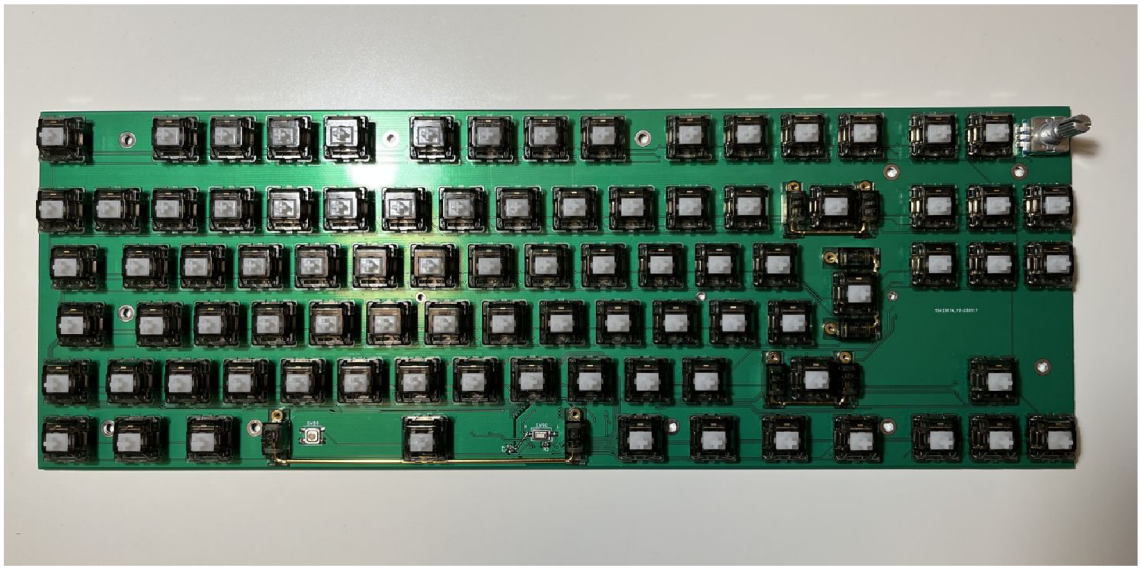


(a) The populated board, reverse side.



(b) Board without switches installed, front side.

Fig. 29: PCB views, front and backside.



(a) The board populated by switches and stabilizers.

(b) Complete board with keycaps.

Fig. 30: Completed board

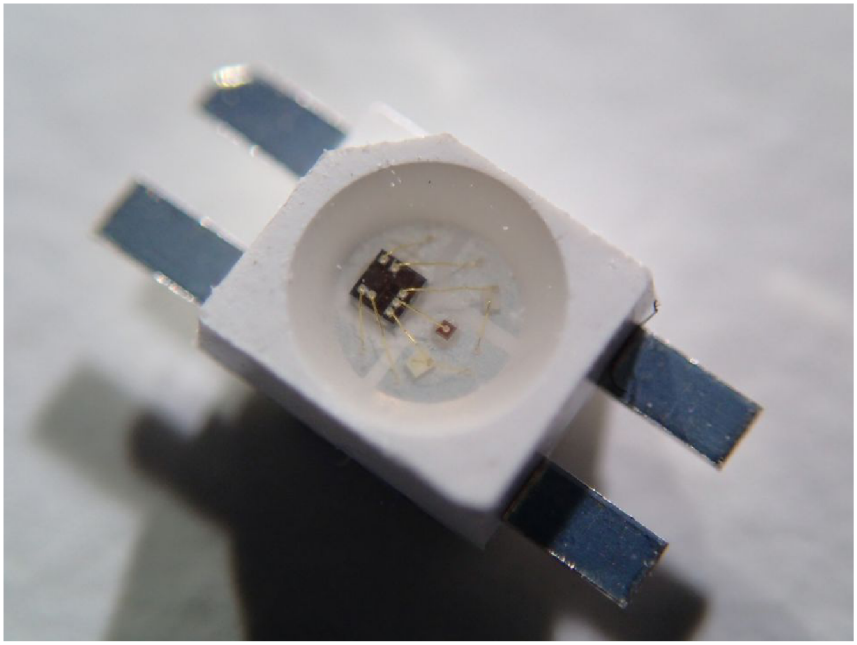


Fig. 31: Detail of the SK6812 LED mini-e package.

an advantage as it is universally compatible even with switches that do not have provision for an SMD backlight LED in their shell.

The LEDs are powered by 5 V and have a standby power consumption of 1 mA per device. The power consumption when in use varies per channel. When all channels are active, i.e. the LED is set to white color, the maximum measured power reached around 50 mA per device. With 8-bit control per channel, the LED brightness can be theoretically set in steps of $\frac{1}{256}$ but actual values differ per chip and the steps are not equidistant in practice. Considering 87 backlit keys, to fit in the standard USB power budget of 500 mA each LED is limited to about 5 mA total. This results in a usable though not very bright light. The brightness can be set higher when dynamic patterns are used as not every LED is lit with the same intensity at the same time. When no backlight is desired, power to the LED chain can be turned off via the high side switch mosfet to save an additional 87 mA.

Each package contains three flip-chip LEDs, one for each color channel, and a control IC. A close-up view of the internals is shown in Fig. 31. The LEDs are controlled by a one wire PWM-like protocol. The devices' data-in and data-out are chained together, the control IC reads a packet from data-in and re-sends the rest on the data-out pin to the next device. With 8-bits per channel as mentioned above, the packets come out to 24-bits. The communication is unipolar meaning the high and low bits are indicated by how long is the signal held high or low each period. To indicate a zero, the data line will be high for $\frac{1}{3}$ of the period and low for the rest.

A one is indicated by a data line high for $\frac{2}{3}$ of the period. Minimum symbol period is specified as 1.2 µs [79]. The communication specification is quite tolerant even of less than ideal conditions making these LEDs easy to work with in a variety of setups.

3.10 Case

The case for the presented keyboard is realized in clear gloss acrylic. The case measures at 366.5 mm wide and 138 mm deep. The construction is stacked, with the outlines laser-cut from 2 and 4 mm thick sheets. The case is 24 mm tall which makes the keyboard require an elevated palm rest for ergonomic usage. The USB-C connector is placed in the middle of the case and is slightly recessed. 3D models of the acrylic case are show in Fig. 32 below. The finished keyboard in the case can be seen in Fig. 33.

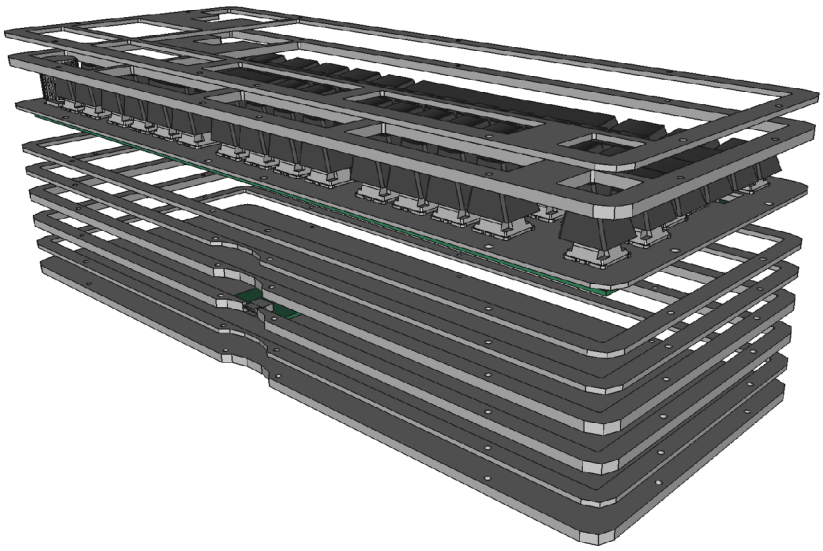
Several issues came up during testing of this case setup. Chief among them the lack of tilt, which combined with the rather high front edge necessitates the use of an elevated palm rest. A more robust alternative case is therefore being explored, preliminary visualization of which is presented in Fig. 34. The new case has a 9° base tilt combined with a slightly lower front edge. The plan for this case is to machine it out of aluminum.

Switch mounting

When mounting switches, two options are available: PCB mount where the switch is either soldered directly to the board or socketed without any additional support, or plate mount where the switch is clipped into a hole in a carrier plate that is then mounted to the rest of the case. Fig. 35 depicts differences between the mounting styles. PCB mounted switches have an advantage of lesser construction complexity but generally at a price of lower key stability. While general stability is always desirable, there are typists who prefer a less rigid mount allowing for some vertical flex when pressing the keys. Some PCB designs even include lengthwise cutouts to enable the board to flex better. Slight flex does provide a softer typing experience and enhances the overall feel but the issue remains a matter of personal preference.

Switches mounted to a plate are held firmly in place and key instability is largely removed save for whatever looseness is present in the switch stem. Some flex can be allowed depending on material used. In this scenario is the PCB then typically mounted to the plate and the whole assembly is encased. Socketed switches are usually plate mounted as the sockets by design do not hold the switch too firmly

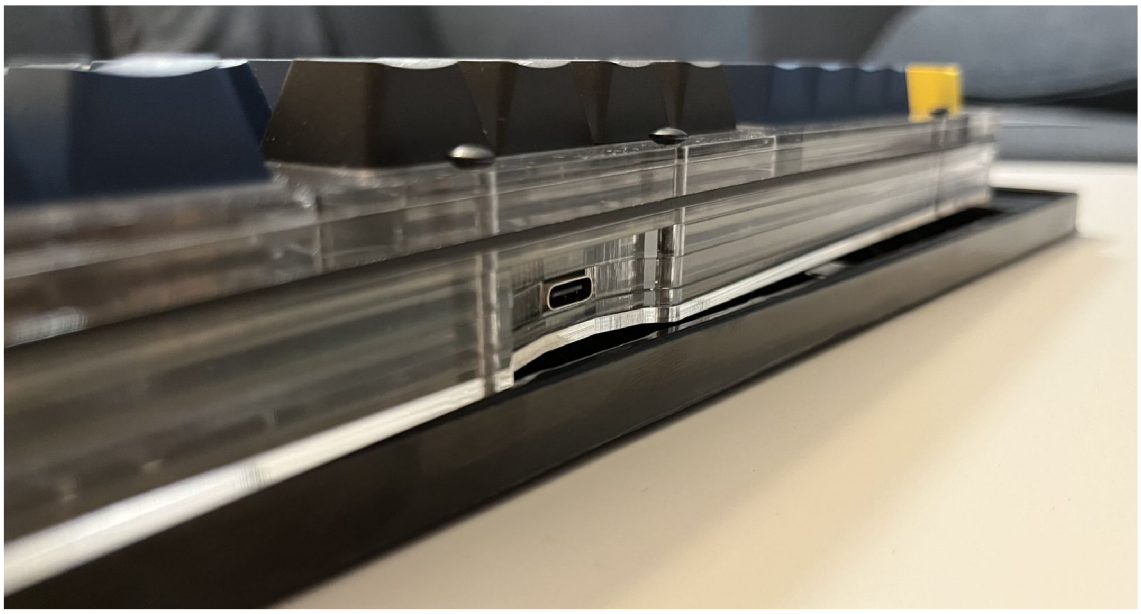
(a) Front view.



(b) Back view, exploded.

Fig. 32: Acrylic case models.

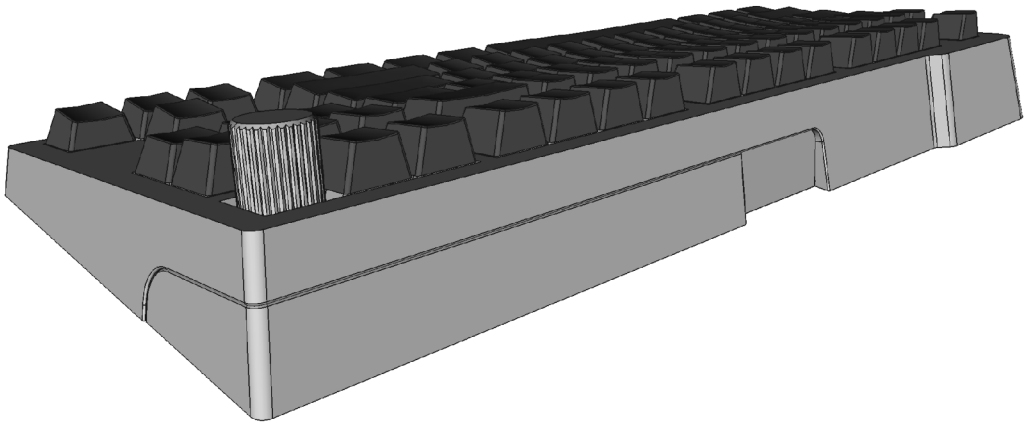
(a) Finished keyboard, front view.



(b) Detail of the USB connector area.

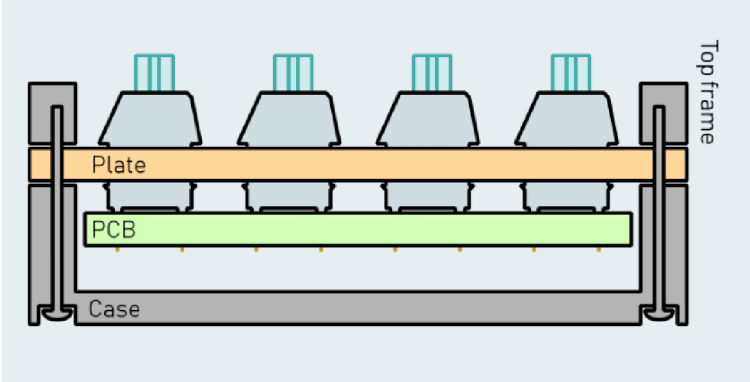
Fig. 33: The finished keyboard.

(a) Front view.

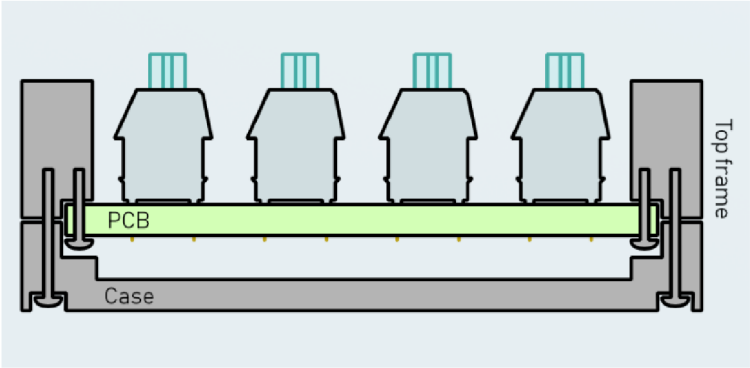


(b) Back view.

Fig. 34: Alternative case prototype.



(a) Sandwich mount plate.



(b) Plateless board.

Fig. 35: Comparison of possible plate and plateless switch mounting; modified from [80].

compared to a soldered connection. The presented keyboard uses plate mounted switches.

Plate mounting

There are several possible ways to mount a plate to the case with the most common as follows [80]:

- Tray mount: The plate sits inside the case, fastened to mounting posts.
- Top mount: the plate is fastened to the top shell.
- Bottom mount: the plate is fastened to the bottom shell.
- Sandwich mount: the plate is sandwiched between outer shell parts.
- Integral mount: the plate is also the top shell.
- Gasket mount: a layer of flexible material is placed around the mounting points and the plate is sandwiched with as few points of contact as possible.

Tray mounted plates are quite common as the plate is very simple and the shell is not very complex to machine. Based on the positions of mounting posts, this approach can result in uneven stiffness and sound. Top and bottom mount plates

provide even stiffness and sound but are not very often seen in custom keyboards as the mounting increases the complexity of both the plate and shells. Sandwich mount often provides a stiffer feel. The setup is however easy to design and manufacture and is often feel in small volume custom boards with varied shapes. Integral mount is usually seen in floating keycap designs. Depends on the particular implementation but this mounting style often results in stiff feel and loud key presses. Gasket mount plates provide pleasant soft sound and typing feel. The setup can theoretically bear looser manufacturing tolerances for the shells but with the increased complexity is often found in more expensive offerings.

The constructed keyboard uses a sandwich plate mount in an acrylic case. The actual PCB can be wither suspended below the plate and held in by the switches themselves which provides a comfortable yield when typing and improves sound somewhat but comes with the disadvantage of requiring complete disassembly when changing the switches. Alternatively, the PCB can be mounted on M3 threaded standoffs holes for which are provided in an alternate case bottom design. The PCB also offers an option to be suspended from the plate by several M2 screws in addition to the installed switches.

Besides the mounting style, plate material plays a large role in the resulting feel as well. Plates are usually manufactured from metals for their rigidity, but good results cane be achieved with plastic plates too. Options include polycarbonate, PVC or even FR4 which is a common PCB substrate material. Aluminum is the most common on the metal side with steel as another common option. Bronze and brass are less commonly available but paired with good case and damping provide a nice variety in acoustic performance. Best results are achieved by experimenting with different materials and styles to find a combination that provides the most satisfying result. The presented keyboard uses a 1.6 mm PVC plate. The plate was generated using the ai03 plate generator service, again based off of the layout created in keyboard-layout-editor [81].

3.11 Production

Producing the keyboard was a chapter in and of itself. The board designs were prototyped and then produced by a custom small volume manufacturing service in China. A number of such services exist and are a good way to produce designs in a relatively cost-effective fashion. Lately the services expanded from simple circuit-board manufacturing to even include option of full board assembly, with some fab houses providing access to industrial-level parts catalogs. They are not without downsides however, some of which were experienced during the production process:

very limited communication with the fab house, and for the last few years seemingly ever-present supply-chain issues. Another caveat lies with the shipping and customs part; it is sometimes not easy to receive an order in a timely fashion. Even through all the limitations, these services are indispensable in creating low-volume custom boards efficiently.

Rapid prototyping for a board this size is a sizable expense even with access to services as mentioned above. Given the minimum order volume of five boards, these larger more complex designs commonly approach costs of 25 – 30 US Dollars per batch for the bare boards only. Combined with international shipping, prototype runs easily approach costs in the hundreds of dollars in three rounds. All in all, despite a rather negative experience with delays due to parts shortage and the exceedingly difficult process of communicating to cancel the delayed order, the net experience with low-cost fab houses remains positive. Parts problems can be avoided by researching the published catalog if available and not relying on an automated BOM processing when ordering.

Components

As mentioned in an earlier section, most passive components were chosen and used because they were readily available in the author's component stash. The rest of the components, namely a number of the hotswap sockets, a reel of the sk6812 leds and the microcontroller were obtained from Chinese e-commerce sites. This approach is attractive mostly because of the offered prices but the results are often a mixed bag. In case of the microcontrollers this was the only option as a component shortage meant that big name suppliers were out of stock across the board. While the sellers usually promise great quality, the components often come from dubious sources behind the scenes and often also provide an equally dubious result in the application. This was unfortunately the case when prototyping the board. A total of five microcontrollers was purchased, of which two came out immediately defective with an internal short-circuit on the power pins. Further two had other internal faults with one failing within about two days due to excessive current consumption and overheating, and one being quite difficult to flash and in the end completely unable to execute the flashed program due to memory corruption. Of the batch purchased only one microcontroller works and even then is not entirely reliable as the DFU interface sometimes fails to process a new program. The alternative method of serial flashing via an ST-Link tool fortunately works even if this approach is not ideal. As the controllers were purchased at only slightly below MSRP, the net result was largely negative with excessive amount of time spent on detecting issues. Paying higher price up-front at a reputable seller can result in net savings in less

time required for troubleshooting and implementing workarounds. That is of course only possible if the parts are available to begin with.

Populating the board

As the attempt to utilize a PCB assembly service was unsuccessful, the prototypes had to be assembled manually. The component choices did take this into account so it did not present an insurmountable challenge. The entire board can be hand-soldered over the course of about four hours, a good chunk of which was spent on the LQFP-48 packaged MCU.

Case

Compared to trouble with PCB production and assembly, producing the case was refreshingly straightforward. A local laser cutting company was able to process the order in a matter of days after promptly resolving minor issues with the source file compatibility.

4 Conclusion

A research of computer keyboard history, components and design approaches with focus on mechanical keyboards was done in the first part of the presented work. The next part then describes a process of designing and implementing a fully customizable mechanical keyboard. The entire work is open-source under the terms of MIT license.

The keyboard is realized in a compact layout without a dedicated numberpad, and includes an additional control element in the form of a rotary encoder. The keyboard utilizes tactile mechanical switches. The switches are mounted in sockets for easy removal and change if so desired. Board designs for both regular ISO and ANSI layouts are provided. Each individual switch has a programmable RGB underglow implemented using a reverse-mount SK6812 LED chain. The switches are capped by a double-shot PBT caps in blue and black colorway with yellow accents manufactured by Keychron. The board has a stacked acrylic case with center mounted recessed USB-C port. The board controller is an ARM STM32F3 series MCU running the open-source QMK firmware. The used firmware is very powerful and customizable down to every little detail. Customizations include a modified base keymap with a Control key in position of the standard Caps-Lock and and RGB control key in the Scroll-Lock position. Further, the Pause/Break key is replaced with the rotary encoder mapped to control volume level. The base keymap can be easily modified to taste using the dynamic configuration tool VIA.

The board was manufactured using a prototyping service but unfortunately due to an ongoing component shortage had to be assembled by hand. The design takes into account such possibility however and the assembly is possible if somewhat challenging in places. The mentioned shortage also affected the available microcontrollers which were sourced from sub-optimal vendors resulting in a rather high failure rate.

The prototype presented in this work lays a foundation for further development. Especially on the ergonomics front as the prototype acrylic case requires a palm rest and does not offer an easy way to adjust tilt. In view of these shortcomings, an alternative machined aluminum option is being explored. The board itself can be further optimized by including overvoltage, overcurrent and static discharge protections in addition to a general optimization of component values for easier production. Further experiments with mounting and flex cuts are already underway as well. And the story continues.

Bibliography

[1] Honeywell. *Micro Switch division history* [online]. 1999 [cit. 2022-03-19]. Available at: http://web.archive.org/web/19990422030528/http://www.honeywell.com/sensing/about/history.stm.

[2] Joseph T Maupin, Everett A Vorthmann. *Hall effect contactless switch with prebiased schmitt trigger* [online]. [cit. 2022-03-19]. Available at: https://patents.google.com/patent/US3596114.

[3] Cherry corp.. *History - Cherry* [online]. [cit. 2022-03-19]. Available at: https://www.cherryamericas.com/company/history.

[4] E Long. *Momentary push button switch with improved non-conductive cam for normally retaining movable leaf spring contacts in a non-operative position* [online]. [cit. 2022-03-19]. Available at: https://patents.google.com/patent/US3715545.

[5] Mike Willegal. *Early Apple Keyboards* [online]. [cit. 2022-03-19]. Available at: http://www.willegal.net/appleii/early-a2-keyboards.htm.

[6] Meryl E Miller. *Electrical interconnection system for multilayer circuitry* [online]. [cit. 2022-03-19]. Available at: https://patents.google.com/patent/US3594684.

[7] Matias. *Matias Mechanical Keyswitches* [online]. [cit. 2022-03-19]. Available at: https://matias.ca/switches/.

[8] Daniel Beardsmore. *Keyswitch illustration.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Keyswitch_illustration.svg.

[9] Daniel Beardsmore. *Physical layout – Asian 101.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Physical_layout_--_ANSI_101.svg.

[10] Daniel Beardsmore. *Physical layout – ISO 102.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Physical_layout_--_ISO_102.svg.

[11] Daniel Beardsmore. *Physical layout – Asian 101.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Physical_layout_--_Asian_101.svg.

[12] Daniel Beardsmore. *Tenkeyless keyboard layout.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Tenkeyless_keyboard_layout.svg.

[13] Daniel Beardsmore. *75% keyboard layout.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:75%25_keyboard_layout.svg.

[14] Daniel Beardsmore. *Tenkeyless vs 60 percent.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Tenkeyless_vs_60_percent.svg.

[15] Yasuoka, M. On the Prehistory of QWERTY. *ZINBUN*. Mar 2011, vol. 42, p. 161 – 174. DOI: 10.14989/139379.

[16] Danelson83. *KB_United_States.svg* [online]. [cit. 2022-03-19]. Available at: https://commons.wikimedia.org/wiki/File:KB_United_States.svg.

[17] j vanovcan. *KB_Slovak.svg* [online]. [cit. 2022-03-19]. Available at: https://commons.wikimedia.org/wiki/File:KB_Slovak.svg.

[18] Randy Cassingham. *The Dvorak Keyboard: the Basics* [online]. [cit. 2022-03-19]. Available at: https://dvorak-keyboard.com/.

[19] Shai Coleman. *Colemak keyboard layout FAQ* [online]. [cit. 2022-03-19]. Available at: https://colemak.com/FAQ#What.27s_wrong_with_the_Dvorak_layout.3F.

[20] Martin Krzywinski. *Carpalx keyboard layout optimizer* [online]. [cit. 2022-03-21]. Available at: http://mkweb.bcgsc.ca/carpalx/?about.

[21] Jarry Xiao. *Data mining typeracer* [online]. [cit. 2022-03-19]. Available at: https://medium.com/@jarryxiao/data-mining-typeracer-part-2-5bfd0726f87e.

[22] Cherry Corp.. *CHERRY MX1A-GxxA/B* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/1/3/6/1/13618248706cd28e75ab9bdf9e55e9f8794611c1/EN_CHERRY_MX_BROWN.pdf.

[23] Jack G Ganssle. *A guide to debouncing* [online]. [cit. 2022-04-11]. Available at: https://my.eng.utah.edu/~cs5780/debouncing.pdf.

[24] onsemi. *onsemi MC14490 datasheet* [online]. [cit. 2022-04-11]. Available at: https://www.onsemi.com/pdf/datasheet/mc14490-d.pdf.

[25] Digikey. *Onsemi MC14490*. [cit. 2022-04-11]. Available at: https://www.digikey.com/en/products/detail/onsemi/MC14490DWR2G/1478851.

[26] Dave Dribin. *Keyboard matrix help* [online]. [cit. 2022-04-11]. Available at: https://www.dribin.org/dave/keyboard/one_html/.

[27] Daniel Beardsmore. *Rubber dome over membrane, exploded.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Rubber_dome_over_membrane,_exploded.svg.

[28] Daniel Beardsmore. *Pressure membrane assembly.svg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Pressure_membrane_assembly.svg.

[29] Takao Nagashima. *Keyboard switch* [online]. [cit. 2022-03-19]. Available at: https://patents.google.com/patent/US4584444A/en.

[30] Cherry Corp.. *Cherry MX Red - The fastest gaming keyswitch.* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/en/cherry-mx/mx-original/mx-red.html#techSpecs.

[31] Cherry Corp.. *Cherry MX Brown - Ideal keyswitches for precise typing.* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/en/cherry-mx/mx-original/mx-brown.html#techSpecs.

[32] Cherry Corp.. *Cherry MX Blue - The keyswitch with the ideal feedback.* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/en/cherry-mx/mx-original/mx-blue.html#techSpecs.

[33] Cherry Corp.. *CHERRY MX - Developer Information* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/en/dev.html.

[34] Daniel Beardsmore. *Cherry MX – sliders and springs.jpg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Cherry_MX_--_sliders_and_springs.jpg.

[35] Cherry Corp.. *CHERRY MX1A-LxxA/B* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/b/7/a/b/b7ab2f72bd5686e1e1e759a9f3703536d7fb1e18/EN_CHERRY_MX_RED.pdf.

[36] Cherry Corp.. *CHERRY MX1A-1xxA/B* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/0/7/f/6/07f6a966a4b95053db5691e73faa401f67d2eb5e/EN_CHERRY_MX_BLACK.pdf.

[37] Cherry Corp.. *CHERRY MX1A-Cxxx* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/1/e/6/d/1e6d4479ea3c692473ae8dd3f0b825bd568ecadb/EN_CHERRY_MX_CLEAR.pdf.

[38] Cherry Corp.. *CHERRY MX1A-ExxA/B* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/a/5/3/1/a531cb6598bc849cbcf131fd7a31814282b74545/EN_CHERRY_MX_BLUE.pdf.

[39] Cherry Corp.. *CHERRY MX1A-Fxxx* [online]. [cit. 2022-04-21]. Available at: https://www.cherrymx.de/_Resources/Persistent/a/a/3/1/aa31c11f193a199eea05c8897d9decc539ce9b7f/EN_CHERRY_MX_GREEN.pdf.

[40] Kailh. *Kailh Switch Set* [online]. [cit. 2022-04-21]. Available at: https://www.kailh.net/products/kailh-switch-set?variant=43776339804402.

[41] Gateron. *Gateron Phantom Switch set* [online]. [cit. 2022-04-21]. Available at: https://www.gateron.co/products/gateron-phantom-switch-set.

[42] Keychron. *Keychron K Pro Switch set* [online]. [cit. 2022-04-21]. Available at: https://www.keychron.com/products/keychron-k-pro-switch.

[43] findecanor. *CherryMXForceTravel.png* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:CherryMXForceTravel.png.

[44] Daniel Beardsmore. *Cherry MX slider – damped.jpg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:Cherry_MX_slider_--_damped.jpg.

[45] Daniel Beardsmore. *O-rings_fitted.jpg* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:O-rings_fitted.jpg.

[46] findecanor. *KeyProfile.gif* [online]. [cit. 2022-04-21]. Available at: https://deskthority.net/wiki/File:KeyProfile.gif.

[47] jjbeard. *Encoder Disc (3-Bit).svg* [online]. [cit. 2022-04-21]. Available at: https://commons.wikimedia.org/wiki/File:Encoder_Disc_(3-Bit).svg.

[48] Dynamic Research Corporation. *Techniques For Digitizing Rotary and Linear Motion*. 2002.

[49] Mike Murray. *How Rotary Encoders Work – Electronics Basics* [online]. [cit. 2022-04-21]. Available at: https://www.thegeekpub.com/245407/how-rotary-encoders-work-electronics-basics/.

[50] Adam Chapweske. *PS/2 Mouse/Keyboard Protocol* [online]. [cit. 2022-04-14]. Available at: https://www.burtonsys.com/ps2_chapweske.htm.

[51] Giorgos Lazaridis. *The PS2 protocol* [online]. [cit. 2022-04-14]. Available at: https://pcbheaven.com/wikipages/The_PS2_protocol/?p=1.

[52] USB Implementers Forum. *Universal Serial Bus Specification, Revision 2.0* [online]. [cit. 2022-04-20]. Available at: https://www.usb.org/document-library/usb-20-specification.

[53] USB Implementers Forum. *Defined Class Codes* [online]. [cit. 2022-04-20]. Available at: https://www.usb.org/defined-class-codes.

[54] USB Implementers Forum. *Device Class Definition for Human Interface Devices (HID)* [online]. [cit. 2022-04-20]. Available at: https://www.usb.org/sites/default/files/hid1_11.pdf.

[55] USB Implementers Forum. *HID Usage Tables* [online]. [cit. 2022-04-29]. Available at: https://www.usb.org/sites/default/files/hut1_4.pdf.

[56] Ian Prest. *Keyboard Layout Editor* [online]. [cit. 2023-05-11]. Available at: http://keyboard-layout-editor.com/.

[57] ai03. *MX Alps Hybrid* [online]. [cit. 2023-05-11]. Available at: https://github.com/ai03-2725/MX_Alps_Hybrid.

[58] yskoht. *Keyboard Layouter* [online]. [cit. 2023-05-11]. Available at: https://github.com/yskoht/keyboard-layouter/.

[59] ebastler. *marbastlib* [online]. [cit. 2023-05-11]. Available at: https://github.com/ebastler/marbastlib.

[60] Kailh. *Kailh BOX Heavy switxh set* [online]. [cit. 2022-04-21]. Available at: https://www.kailh.net/products/kailh-box-heavy-switch-set.

[61] Switches.mx. *Everglide Oreo* [online]. [cit. 2022-04-21]. Available at: https://switches.mx/everglide-oreo.

[62] Keycrhon. *Cherry Profile Double - Shot PBT Full Set Keycaps - Royal* [online]. [cit. 2022-04-21]. Available at: https://www.keychron.com/products/double-shot-pbt-cherry-full-set-keycap-set-royal.

[63] TT Electronics. *Rotary Encoder Model EN11 Series* [online]. [cit. 2022-04-21]. Available at: https://www.ttelectronics.com/TTElectronics/media/ProductFiles/Datasheet/EN11.pdf.

[64] VIA Team. *VIA* [online]. [cit. 2022-04-21]. Available at: https://www.caniusevia.com/.

[65] Ruiqi Mao. *Keyboard Firmware Builder* [online]. [cit. 2023-05-11]. Available at: https://kbfirmware.com/.

[66] Holtek. *HT82K94E/HT82K94A USB Multimedia Keyboard Encoder 8-Bit MCU* [online]. [cit. 2022-04-21]. Available at: https://www.holtek.com.tw/documents/10179/11842/HT82K94xv240.pdf.

[67] Microchip/Atmel. *ATmega16U4/ATmega32U4 8-bit Microcontroller with 16/32K bytes of ISP Flash and USB Controller* [online]. [cit. 2022-04-21]. Available at: https://ww1.microchip.com/downloads/en/devicedoc/atmel-7766-8-bit-avr-atmega16u4-32u4_datasheet.pdf.

[68] Microchip/Atmel. *8-bit AVR Microcontroller with 64/128K Bytes of ISP Flash and USB Controller* [online]. [cit. 2022-04-21]. Available at: https://www.pjrc.com/teensy/at90usb1286.pdf.

[69] ST Micro. *STM32F303CC* [online]. [cit. 2022-04-21]. Available at: https://www.st.com/resource/en/datasheet/stm32f303cc.pdf.

[70] Jack Humbert. *Quantum Mechanical Keyboard Firmware* [online]. [cit. 2023-05-11]. Available at: https://github.com/qmk/qmk_firmware.

[71] QMK Contributors. *General FAQ* [online]. [cit. 2023-05-11]. Available at: https://docs.qmk.fm/#/faq_general.

[72] QMK Contributors. *QMK Configurator* [online]. [cit. 2023-05-11]. Available at: https://config.qmk.fm/.

[73] QMK Contributors. *QMK Keycodes* [online]. [cit. 2023-05-11]. Available at: https://docs.qmk.fm/#/keycodes.

[74] QMK Contributors. *RGB Lighting* [online]. [cit. 2023-05-11]. Available at: https://github.com/qmk/qmk_firmware/blob/master/docs/feature_rgblight.md.

[75] VIA Team. *Configuring QMK* [online]. [cit. 2022-04-21]. Available at: https://www.caniusevia.com/docs/configuring_qmk.

[76] QMK Contributors. *Understanding QMK's code* [online]. [cit. 2023-05-11]. Available at: https://github.com/qmk/qmk_firmware/blob/master/docs/understanding_qmk.md.

[77] Semtech Electronics. *1N4148W Silicon Epitaxial Planar Switching Diode* [online]. [cit. 2022-04-21]. Available at: https://datasheet.lcsc.com/lcsc/1811061725_ST-Semtech-1N4148W_C81598.pdf.

[78] Kailh Kaihua Electronics. *KH-PS1607-10 Product specification* [online]. [cit. 2022-04-21]. Available at: https://www.kailhswitch.com/Content/upload/pdf/201815927/PG151101S11.pdf.

[79] Shenzhen Normand Electronic. *SK6812MINI-E LED Datasheet* [online]. [cit. 2022-04-21]. Available at: http://www.normandled.com/upload/202004/SK6812MINI-E%20LED%20Datasheet.pdf.

[80] Thomas Baart. *Cheat sheet: Custom keyboard mounting styles* [online]. [cit. 2022-04-21]. Available at: https://thomasbaart.nl/2019/04/07/cheat-sheet-custom-keyboard-mounting-styles/.

[81] ai03. *ai03 Plate Generator* [online]. [cit. 2023-05-11]. Available at: https://kbplate.ai03.com/.

List of Figures

1 Construction of a typical mechanical keyboard. 20
2 The standard full-size layouts 21
3 Miniaturized ANSI layouts 22
4 The QWERTY layout. 24
5 Character frequency heatmaps. 25
6 Schematic of a hardware-debounced switch. 27
7 Ghosting when pressing multiple keys 28
8 Series diodes prevent ghosting 28
9 Membrane key switch schematics 31
10 Cutaway schematic of a Topre switch 32
11 Cutaway view of Cherry MX switches. 32
12 Dimensions of an MX type switch. 33
13 Comparison of MX switch footprints. 34
14 Selection of various Cherry MX sliders and springs. 34
15 Actuation force over travel distance. 36
16 Noise reduction approaches. 37
17 MX cross mount 1.25 U screw-in PCB mount wire stabilizer. 39
18 Common keyboard profiles. 40
19 Keycap profile comparison. 40
20 Encoder wheels. 42
21 USB bus topology. 45
22 Example descriptor hierarchy for HID class device. 49
23 Example of parsed HID descriptors. 51

24 JWK Everglide Oreo switch and its internal components. 55
25 Everglide Oreo switch contacts. 55
26 The keycap profile. 56
27 Key matrix connection. 58
28 HSV model color wheel. 66
29 PCB views, front and backside. 69
30 Completed board 70
31 Detail of the SK6812 LED mini-e package. 71
32 Acrylic case models. 73
33 The finished keyboard. 74
34 Alternative case prototype. 75
35 Comparison of possible plate and plateless switch mounting. 76

List of Tables

1 Switch characteristic comparison 35
2 Common keycap sizes 38
3 Defined USB class codes 47

4 Comparison of considered switches 54
5 Considered microcontrollers. 59

6 The board BOM 97
7 Board interconnect cable. 98
8 Switches and keycaps. 98

List of Appendices

A BOM **97**
B Electrical connection schematics. **99**

A BOM

Grand total component count: 488
Board component count: 311

Component	Value	Footprint	Quantity
Capacitor, ceramic	4.7 uF	SMD 0805	1
Capacitor, ceramic	10 uF	SMD 0805	2
Capacitor, ceramic	10 pF	SMD 0805	2
Capacitor, ceramic	100 nF	SMD 0805	23
Resistor	10k	SMD 0805	5
Resistor	0	SMD 0805	2
Resistor	1k	SMD 0805	1
Resistor	1.5k	SMD 0805	1
Resistor	5.1k	SMD 0805	2
Microcontroller	STM32F303CCT6	LQFP-48	1
Voltage regulator	3.3 V	SOT223	1
Crystal	12 MHz	3.2x2.5 mm SMD	0 or 1
Crystal	12 MHz	HC49 SMD	0 or 1
Mosfet	P-Channel IRLML6402	SOT-23	1
LED	SK6812	mini-e	87
Switch socket	-	Cherry MX	87
Diode	1N4148	SOD-123	88
Rotary encoder	-	Alps EC11	1
Switch, SPST	-	5x5 mm SMD	1
Switch, DIP	-	DIP SMD	1
Connector	Female	SMD 1.25mm pin-pitch	2
Connector	USB-C female	through-hole 4085	1

Tab. 6: The board BOM

Component	Value	Footprint	Quantity
Connector, male	-	6x1	2
Connector pin	-	-	12
Conductor wire	D 0.5 mm	-	6

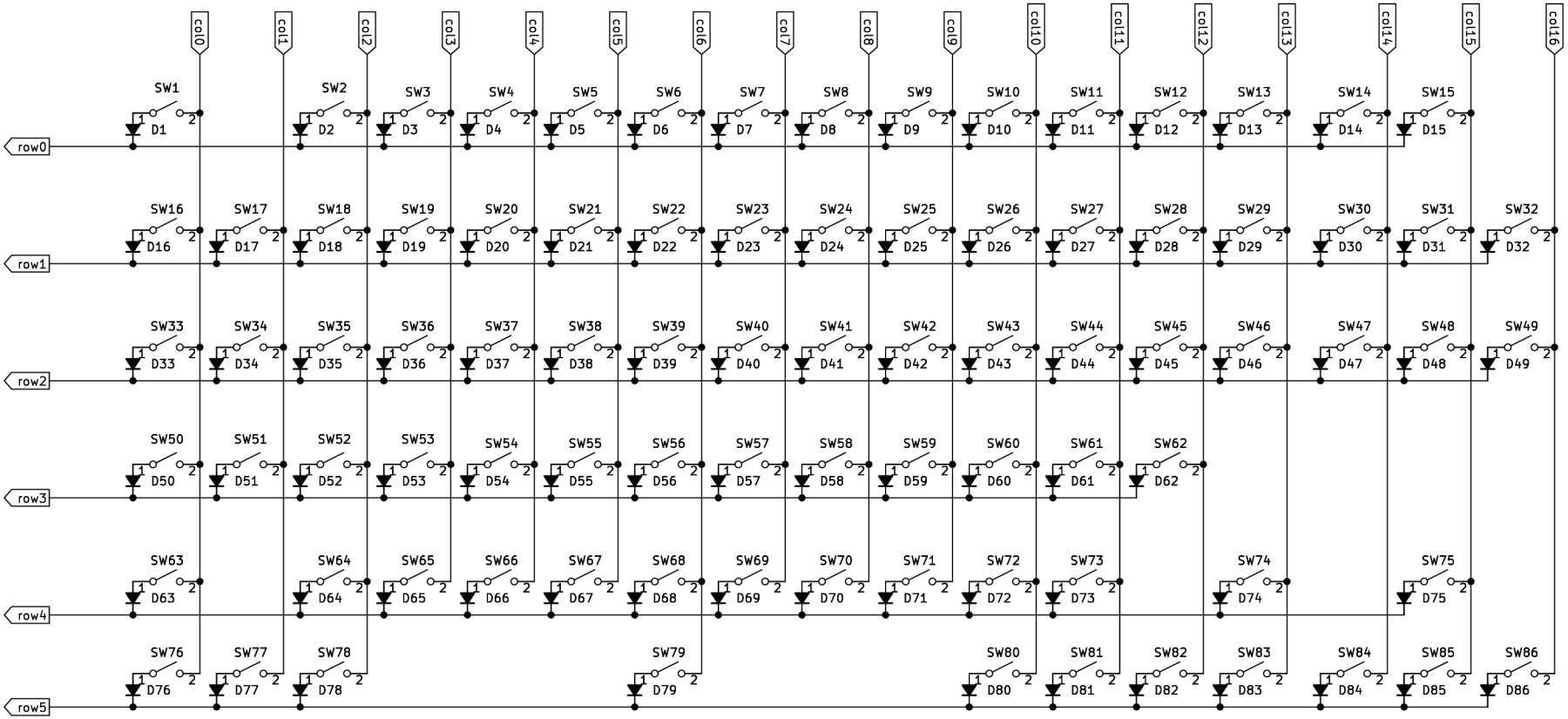
Tab. 7: Board interconnect cable.

Component	Value	Footprint	Quantity
Key Switch	Everglide Oreo	Cherry MX	87
Key cap	1 U	Cherry MX	73
Key cap	1.25 U	Cherry MX	8
Key cap	1.5 U	Cherry MX	1
Key cap	1.75 U	Cherry MX	1
Key cap	2 U	Cherry MX	1
Key cap	2.75 U	Cherry MX	1
Key cap	6.25 U	Cherry MX	1
Key cap	ISO enter	Cherry MX	1
Encoder cap	-	6 mm shaft	1

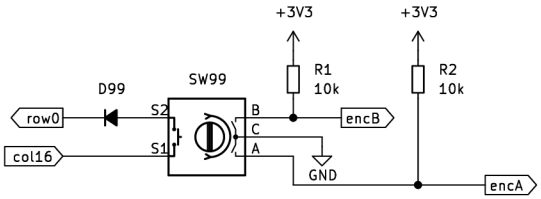
Tab. 8: Switches and keycaps.

B Electrical connection schematics.

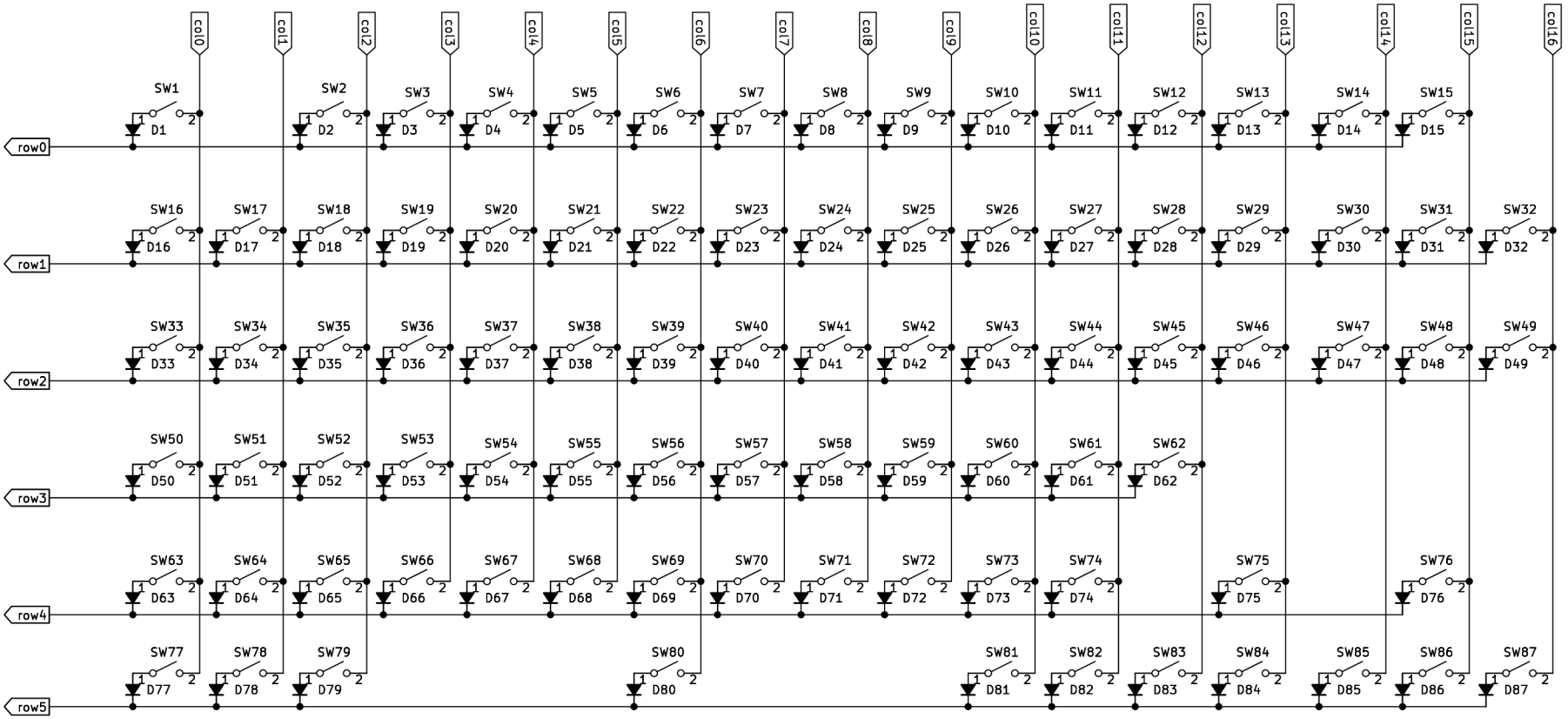
Key matrix ANSI



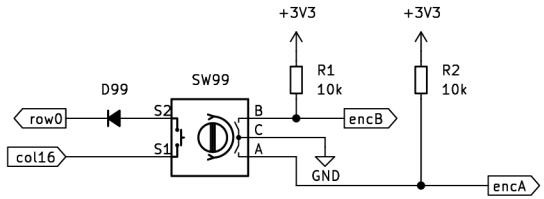
Encoder



Key matrix ISO

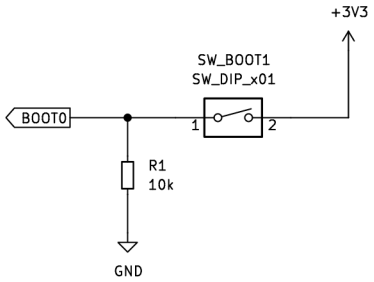


Encoder

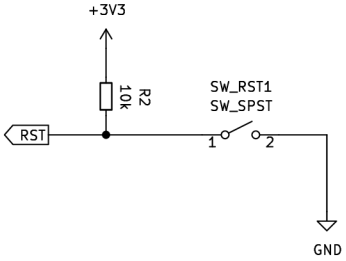


Microcontroller

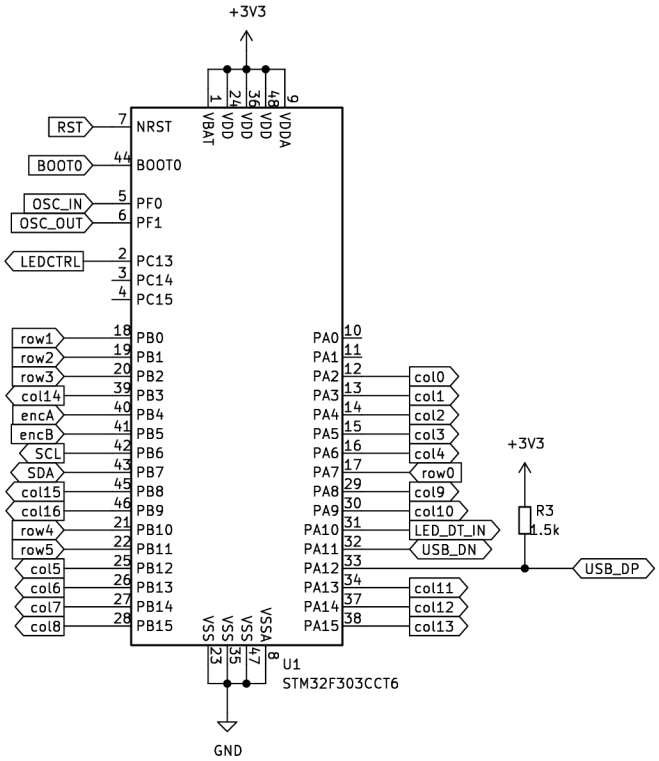
Boot mode selection



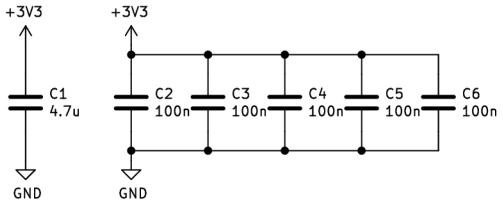
Reset button



STM32F303CCT6

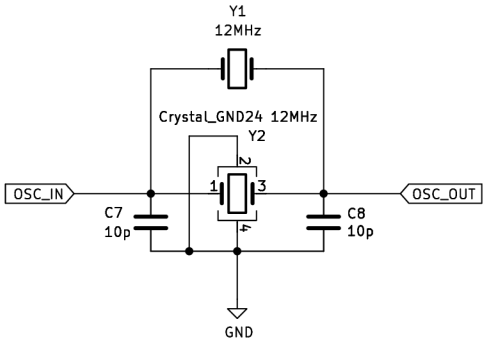


Power decoupling

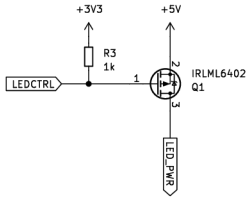


Crystal oscillator

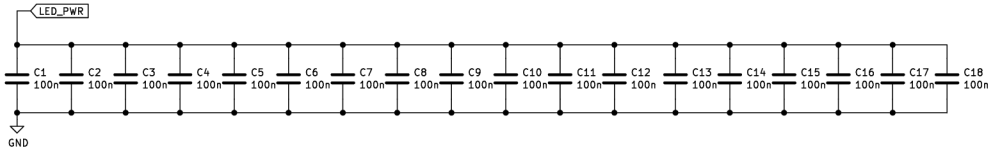
Choose and populate one footprint



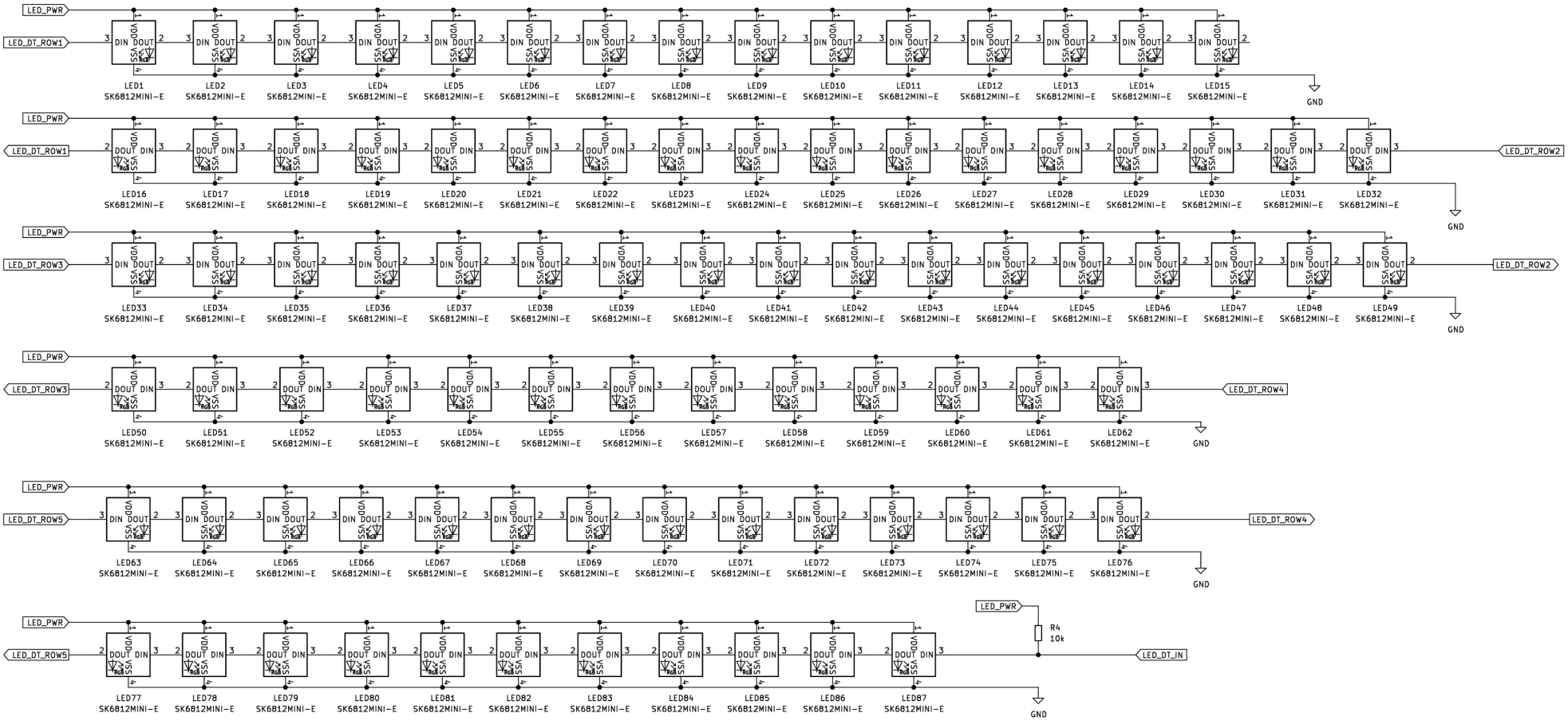
High side switch



Power decoupling

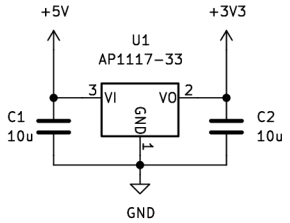


LED daisychain



USB daughterboard

3V3 regulator



USB-C socket