Plug and Play Stenography Keyboard

Team number: 2

Team Members:
Haoqing Zhu (haoqing3@illinois.edu)
Soham Karanjikar (sohammk2@illinois.edu)
Rishi Krishnan (rishik3@illinois.edu)

TA: William Zhang

September 12, 2020

Course: ECE 445

1. Introduction

1.1. Objective

There is currently no way for verbally disabled individuals to communicate with others in real time. Sign languages do provide a way out, but it is very impractical to expect everyone else to learn it in order to communicate. On the other hand, typing on traditional keyboards is too slow, they do not offer any solution to this problem.

Our solution is to build a plug and play stenography keyboard that has all the necessary hardware/software built within, so no extra installation is needed on the host side. This keyboard will offer typing speeds close to what humans speak at (180 - 250WPM).

1.2. Background

Traditional steno keyboards are very expensive [1] as they are generally used only in court. Additionally, they are also bundled with many court-reporting specific softwares which would be useless in everyday communication [2]. Our project on the other hand builds the keyboard for a fraction of the cost while providing the same benefits.

Since our keyboard does not require much technical knowledge, such as installing software or coding, it is very marketable to various demographics. The only necessary knowledge is learning how to use a stenography keyboard. Another benefit is the ability to change the dictionary in our keyboard, so that users can make their own strokes for new words or new phrases, and maybe even use a different language.

1.3. Visual Aid

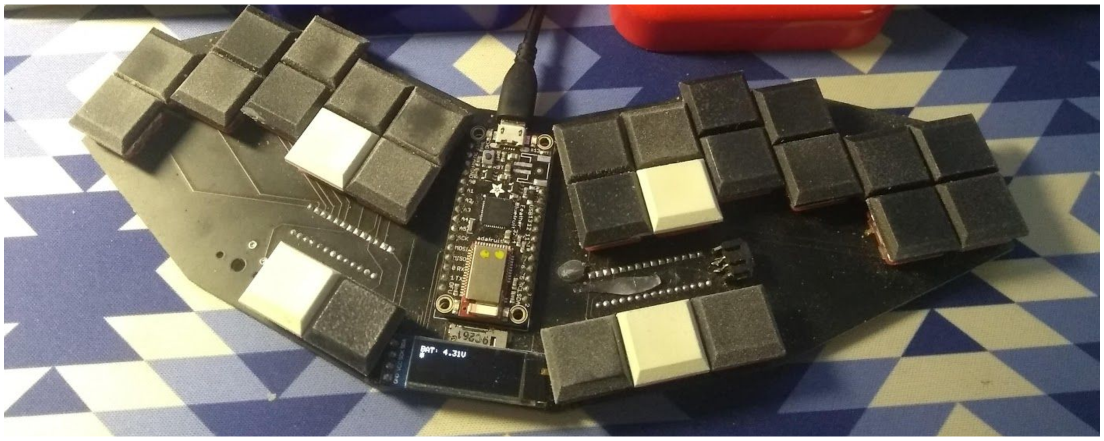

Figure 1: Physical Prototype

1.4. High-level Requirements List

i. The user is able to convert their personal dictionary (assumed in Plover JSON format) using a utility tool into our custom format (for efficiency) within 2 minutes, and load it onboard within 1 minute.
ii. The device is able to translate strokes taken from keyboard to keystrokes according to the corresponding dictionary entry with a max delay of 100ms.
iii. The user is able to dynamically add/delete/modify dictionary entries on the fly within 500ms.

2. Design

2.1. Block Diagram

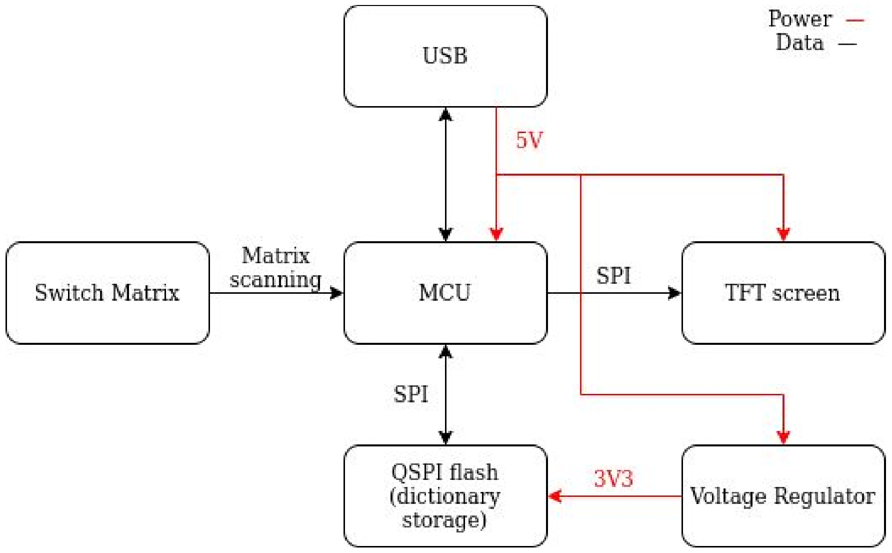

Figure 2: Hardware Block Diagram

The dictionary compiler would allow users to convert their personal dictionary into the version that we would be using. The SPI flash would store the dictionary onboard and the translation engine will translate the strokes from the keyboard into output according to the stored dictionary. The user interface through OLED and the keyboard itself would allow the user to modify the stored dictionary.

2.2. Physical Design

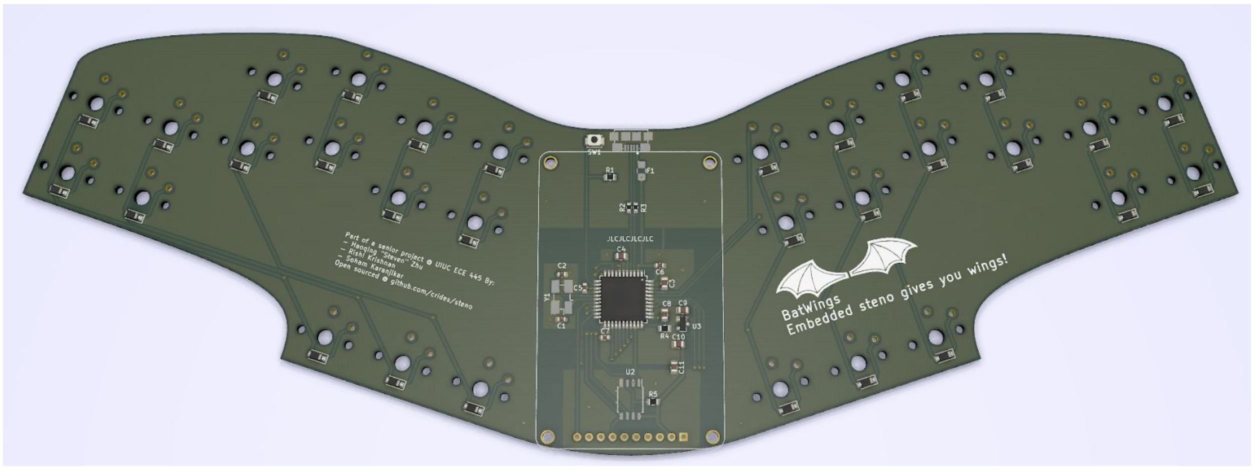
Part of a senior project @ UIUC ECE 445 By:
– Haoqing "Steven" Zhu
– Rishi Krishnan
– Soham Karanjikar
Open sourced @ github.com/crides/steno
BatWings
Embedded steno gives you wings!

Figure 3: 3D board rendering

2.3. Power Module

The keyboard is powered through USB. The MCU and TFT screen are powered directly by the 5V, while the flash is powered by 3.3V through a voltage regulator.

Requirement	Verification
1. VBUS from USB is able to provide 5V ± 5%	1. Measure the VBUS line from USB using an oscilloscope, and see if the maximum and minimum stays within the 5% range
2. Output from voltage regulator is able to provide 3.3V ± 5%	2. a. Connect the regulator with the decoupling caps b. Connect the input of the regulator to VBUS of USB line c. Measure the output of the regulator, and see if it stays within the 5% range

2.4. Dictionary Compiler

The dictionary compiler is required to bootstrap the dictionary onboard. It will take a Plover JSON dictionary and convert it into a binary format that's easy for the MCU to read and manipulate.

Requirement	Verification
1. Able to compile a Plover JSON dictionary to the required format	1. a. Fetch the latest dictionary from Plover's repo b. compile it using the dictionary compiler c. load it onto the flash d. randomly input sequences of strokes and check that the output is the same as the defined entries

2.5. Dictionary Storage

The SPI flash and the associated driver code will take care of storing the binary dictionary. It will also provide fast random read/write access to the dictionary so that the MCU can have enough speed to process the translation.

Requirement	Verification
1. Able to read correctly from the dictionary at least 500kB/s	1. Program the MCU to: a. Generate and write a 1MB sequence to the flash b. Start a hardware timer c. Read continuously from the flash and verify the data d. Stop the hardware timer e. Use the time difference to measure the read speed, and ensure it's higher than 500kB/s
2. Able to write to the storage at least 200kB/s	2. Program the MCU to: a. Start a hardware timer b. Generate and write a predictable 1MB sequence to the flash c. Stop the timer d. Use the time difference to measure the write speed, and ensure it's higher than 200kB/s e. Compute the CRC64 using the flash

	f. Compute the expected CRC64 using the same sequence and ensure the 2 CRCs are the same

2.6. User Interface and Display

The user interface code should communicate with the screen and display related information for the user about the keyboard status. It should also guide the user when editing the dictionary. Additionally, the screen should be large enough to show all the information related to the keyboard status and to facilitate dictionary editing.

Requirement	Verification
1. Able to add entries using the user interface	1. a. Load the with a dictionary b. Use the UI to add an entry c. Stroke the added entry and see if the output is expected d. Repeat from step b. for several times
2. Able to modify entries using the user interface	2. a. Load the with a dictionary b. Use the UI to modify an entry c. Stroke the modified entry and see if the output has changed d. Repeat from step b. for several times
3. Able to remove entries using the user interface	3. a. Load the with a dictionary b. Use the UI to remove an entry c. Stroke the removed entry and see if the output has changed d. Repeat from step b. for several times

2.7. Translation Engine

The translation engine should read the dictionary stored onboard and control the output at a high level when the strokes come in from the keyboard.

Requirement	Verification
1. Able to translate strokes at a rate of at least 4 strokes per second	1. a. Load the keyboard flash with a dictionary b. Modify the firmware to: i. Start a hardware timer when translation starts ii. Stop the timer after output ends iii. Log the time difference to console c. Input random strokes d. Compute the average time difference

2.8. Switch Matrix, Matrix scanning & HID handling

Matrix scanning and handling HID reports are essential parts for a keyboard, and they will be handled by the QMK firmware.

Requirement	Verification
1. The keyboard should have 23 keys for proper steno usage 2. The keys should be arranged in a ergonomic layout	1. Each of the 23 keys on a normal steno keyboard can be reached easily 2. Try typing on the keyboard for extended periods of time, and see how long it takes for the hands to get tired.

2.9. Schematics

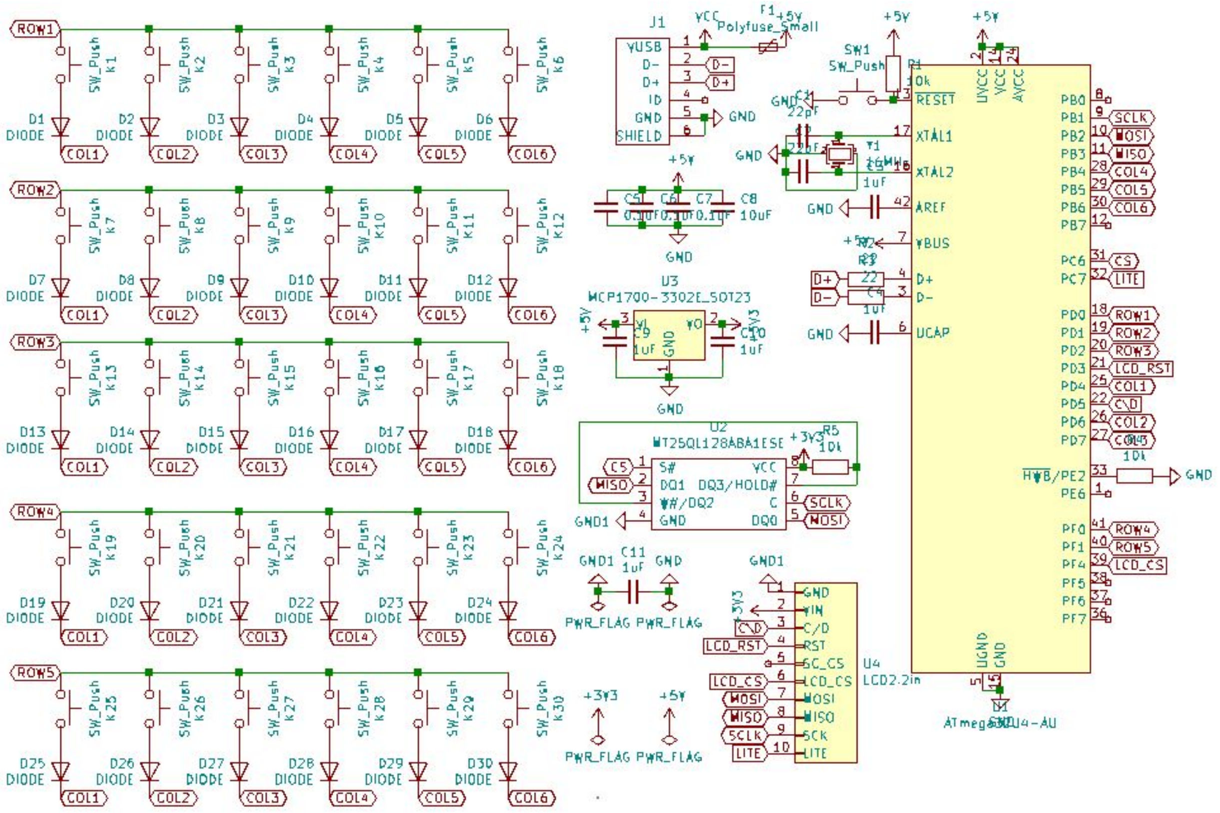

Figure 4: Schematics

2.10. Board Layout

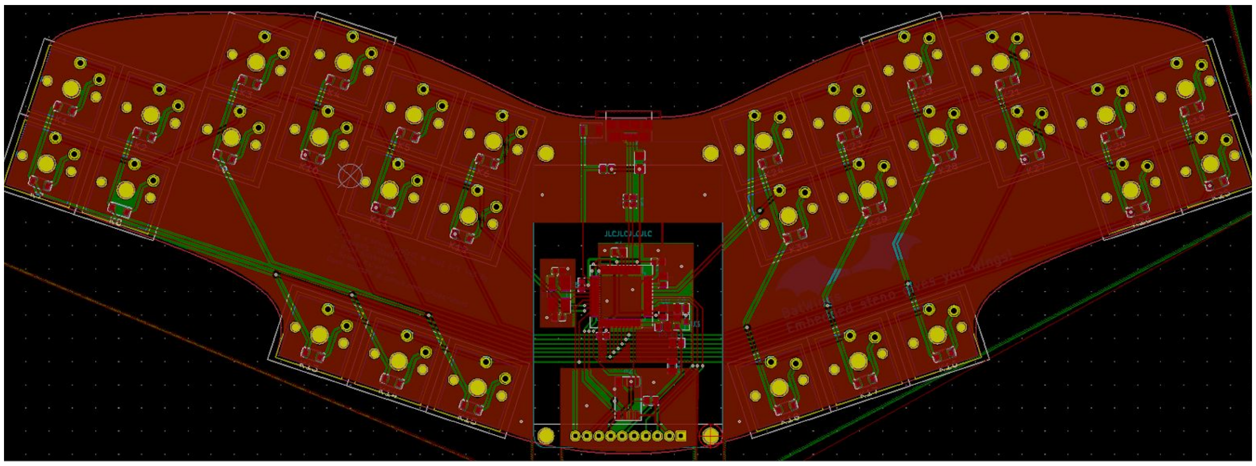

Figure 5: Circuit board layout

2.11. Software Flowchart

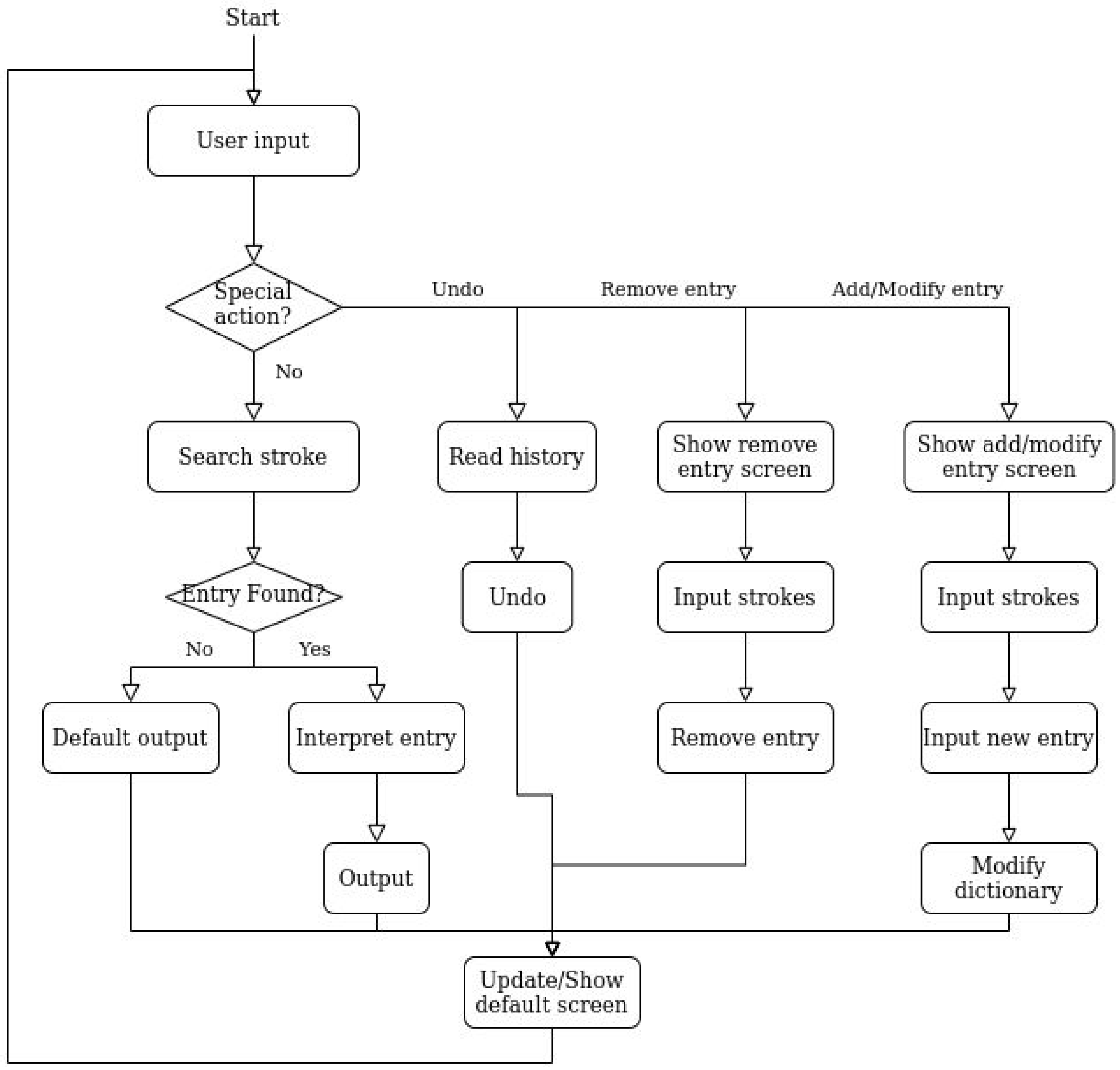

Figure 6: Software flowchart

2.11. Tolerance Analysis

One tolerance we need to maintain is the accuracy of the crystal needs to be good enough so that the USB can operate normally. The USB controller needs a clock source that's accurate within 0.25% in order to operate properly at full speed, according to the MCU datasheet. The crystal we are using has a tolerance of 30ppm at 25 degrees and a temperature tolerance of 50ppm per degree. At a extreme operating temperature of 50 degrees, the accuracy of the crystal would be

$$30 + (60 - 25) * 50 = 1780ppm = 0.178\%$$

which is within the allowed accuracy range.

Another tolerance we need to match is the bandwidth usage for the flash and the TFT screen. In our current design, the ATMega32u4 will be running at 16MHz, which means the SPI hardware will be operating at a maximum of 8MHz, meaning the maximum byte rate would be 1 megabyte per second. Since the MCU will be using the SPI bus to communicate with both the screen and the flash, we need to make sure that we have enough bandwidth.

In our current design, to translate a stroke the MCU needs to search the stroke among at most 8 nodes, then reading the content of the entry which is at most 100 bytes. When searching inside a node, the MCU needs to read the 8 byte header of the node, at most 8 entries of a hashmap each of 8 bytes, and the 8 byte header of the target node. So in the worst case, the number of bytes need to be read from the flash is:

$$8 * ((5 + 8) + (5 + 8) * 8 + 8) + (8 + 100) = 1108\, bytes$$

For the display, let's assume that we want to show the input stroke and the output entry on each stroke. Since we are using a 240x320 display, let's assume that we are using 8x16 fonts, and that each stroke takes up 2 lines and the output takes up 4 lines. So in total the number of bytes we need to write would be:

$$6\, lines * 16\, rows/line * 240\, pixels/row = 23040\, bytes$$

Assuming a maximum of 10 strokes per second, translating strokes will take up 11080 bytes of bandwidth for the flash and 230400 bytes of bandwidth for the screen. Together they will take up 24.1% of the CPU time, which should be enough for the rest of the tasks.

2.12. Contingency Plans

When/If we need to move online, the challenge we would face would almost be the same as what we have now, which is the difficulty to get hardware for testing. This is something we can overcome as we can meet each other on campus even if not permitted to meet in the ECEB lab. In addition, at least one of us has the lab equipment needed (soldering iron, oscilloscope, etc.) So we are not too worried about transitioning to fully online as we are having the same struggles even with the current situation where classes are in-person.

3. Cost and Schedule

3.1. Cost Analysis

3.1.1 Parts

Quantity	Value	Footprint	link	Cost ($)
2	22pF	C0603	link	.2
5	1uF	C0805	link	.5
3	0.1uF (104)	C0603	link	.3
1	10uF (106)	C0805	link	.11
30	DIODE	SOD-123	link	5.1
1	Polyfuse	Fuse 1206	link	0.11
1	USB micro B	AMPHENOL_10118192-0001LF	link	0.21
30	KEYSW	Kailh PG1350		0.6
2	22R	R0603	link	.20
3	10k (103)	R0805	link	.30
1	B3U-1000P	B3U-1000P	link	.92
1	ATmega32U4-AU	TQFP-44_10x10mm_P0.8mm	link	4
1	MT25QL128ABA1ESE	SOP-8_5.28x5.23mm 50mil pitch	link	2.1
1	MCP1700-3302E	SOT-23	link	.37
1	2.2” 18-bit color SPI TFT	100mil pitched headers	link	25
1	16MHz	SMD crystal 4pin 5.0x3.2mm	link	.57
1	74HC4050	SOIC 16	link	.46

3.1.2 Labor

The total estimated time for the project is around 80 man hours, and assuming the development cost is going to be $40/hour, the total labor cost for the project will be: $40/hour * 80hours * 2.5 = $80,000.

3.2. Schedule

Week	Steven	Soham	Rishi
9/28	Initial MSC research	Familiarize with code	
10/5	Assemble boards & testing screens	Fix capitalization & general dict editing UI design	Fix Unicode implementation & testing
10/12	dict editing UI design & implementation	Redesign dict format	Reimplement dict reading
10/19	Add MSC interface	Fix orthography generation	Reimplement dict writing
10/26	Finalize board design	Implement retroactive commands	Implement rest of Plover commands
11/2	Implement ghostFAT	Fix translation engine bugs	Testing
11/9	Improve compiler performance	Implement command parsing	Implement runtime orthography
11/16	Final testing for demo	Prepare for presentation	Begin final report

4. Ethics and Safety

Although injuries risk are almost minimal when using a keyboard, it may still be useful to go over some of the possible physical dangers of using a keyboard [4]. One common issue that people can run into over time is carpal tunnel syndrome, which is caused by pressure in the median nerve of the wrist due to overuse or pressure. This can be easily prevented however, by taking frequent breaks to stretch out the hands and sitting with good posture.

Additionally, you can have standard back and neck pain, which can also be prevented as above, with good posture and frequent brakes. Another physical injury which is unrelated to long term usage is the chance of cutting yourself on the edge of the PCB, as it can be rather sharp. However, with proper care, attention, and handling, this should not be an issue.

As far as electrical errors, the max voltage is only 5V, so there should be minimal risk of injury to the user. However, some components such as the MCU and QSPI flash used in this project will be susceptible to electrostatic discharge (ESD). Precautions must be taken to prevent damaging these parts such as using anti-static gloves and ESD wristbands.

For ethics, we hold responsibility for our project, which is the first rule in the IEEE Code of Ethics [3]. In some sense, our keyboard can be thought of as a giant macropad capable of outputting text or keystrokes at high speeds. Using this keyboard as a tool to conduct abuse such as spam is unethical, according to IEEE Code of Ethics #8 [1]. In addition, because of the same reason, this keyboard can be used in gaming as a form of button macros, which is also a violation of IEEE Code of Ethics #4 [3]. If such use of the device occurs, we do not take any responsibility, as this is the nature of this device, and using the device for spamming or for normal writing is intelligible from firmware.

References

[1] “Writers | Stenograph L.L.C,” *Stenograph L.L.C*. [Online]. Available: https://www.stenograph.com/stenograph-writers/. [Accessed: 30-Sep-2020]

[2] “DigitalCAT Software and Support,” *Stenovations*. [Online]. Available: https://www.stenovations.com/product-category/digitalcat/. [Accessed: 30-Sep-2020]

[3] “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 17-Sep-2020]

[4] “3 Injuries Experienced by Court Reporters and how to Avoid Them,” O'Brien & Bails. [Online]. Available: http://www.obrienandbails.com/3-injuries-experienced-by-court-reporters-and-how-to-avoid-them/. [Accessed: 1-Oct-2020]