Name: Jackson Hagood
UIN: 128006409
Section: CSCE 483-931

Notebook

8/24

First day of class. Initial conception of possible team. Completed resume and entrance questionnaire assignments. The partially formed team began coming up with project ideas in a Google Docs document. Brittany Jenkins mentioned something with accessibility could be rewarding. Andrew came up with the idea of a computer mouse designed for people with Parkinson's. It was also decided that the disability services center could be of aid when coming up with ideas.

8/25

I began brainstorming project ideas by myself. Keeping in mind the accessibility goal and the computer mouse idea I came up with some sort of oversized keyboard for those with disabilities. I have seen oversized keyboards in the past for children (namely a YouTube video). Perhaps an oversized keyboard could be designed for a more mature audience with similar needs. I could not decide on any final designs but I theorized there would be great potential in the idea.

YouTube video: https://youtu.be/t3p4-RJDob8

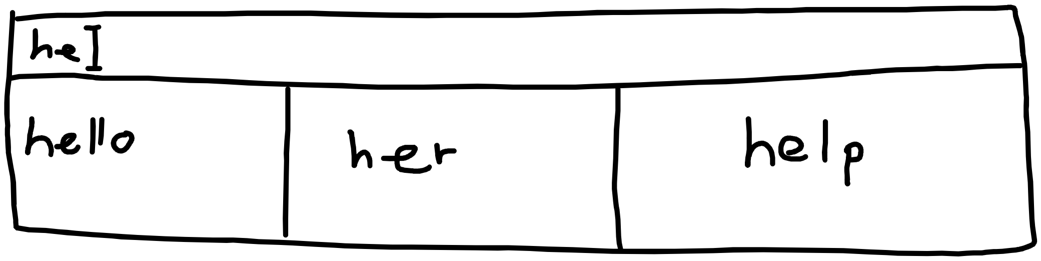
8/29

Final team is formed. Andrew Imwalle, Brittany Jenkins, Connie Liu, Abhishek More, and Max Smith will comprise the team with me. I have worked in the past with Andrew and Max (CSCE 462 project) and am confident in their abilities and our potential when working together. Within our team we have a slight software leaning, but we all still have considerable hardware experience. I presented my keyboard idea to the team with good reception. Together we began discussions on the shapes of the keys, the size of the keyboard, and the potential integration of smart keyboard features. It was also thought that an unconventional keyboard layout may be useful. The QWERTY layout is an artifact from typewriters, indicating a more efficient layout (such as DVORAK) could improve the typing of experience. This appears to be the main candidate at this point. I also created a OneDrive for all of our future documents and placed the proposal template into it.

8/30

I began thinking about the keyboard ideas and came up with the idea of some sort of auto-complete GUI that works in tandem with the hardware. This GUI could be created in something like electron so that it would be compatible with most major operating systems. The GUI would be minimalistic to not impose on other programs.

Extremely rough conception:



Electron: https://www.electronjs.org/

8/31

Discussions on the keyboard continued. I proposed the idea of the GUI to the team. Auto-correct was discussed by decided against. A lot of discussion was had on this idea and two schools of thought were formed. First, the GUI could be implemented in the operating system as its own program (and therefore theoretically be compatible with any keyboard). This would require some sort of interception of signals within the operating system and it is not known how possible this may be. Second, the keyboard could be fitted with LCD displays that would show the auto-complete options. This embedded system approach seems more viable. Despite my initial idea I am leaning towards the embedded system approach. I thought making a front-end in Electron could be rewarding, but it may not suit this project. Other parts of the keyboard were discussed, including some rough ideas for how the circuitry could be accomplished and what kind of microcontroller would be needed (likely Raspberry Pi). Other project ideas were mentioned but the keyboard seems to be the team's main candidate.

9/5

I continued considering the project and decided the embedded approach would work best with most devices.

9/7

The team decided on the oversized keyboard as the final project idea. The team had a long discussion on ideas to implement this. Abhishek mentioned that a trie data structure can be used for auto-complete algorithms. Partial words would be traversed in this tree-like data structure until the end was found. Then, candidates could be found by traversing the remaining nodes. The team discussed how to best implement the keyboard's hardware. For example, the team questioned how best to design the keycaps, what the keyboard's profile should be, how much actuation force should be required to press a key, and other ideas. Then, the discussion on the GUI vs embedded system approach continued. The pros and cons for each approach were discussed and listed as follows:

I. Embedded approach
   a. Pros:
      i. More accessible
      ii. Appeals to older users
      iii. Plug and play
   b. Cons:
      i. Keyboard required micro-controller
      ii. Memory requirements could be tight
      iii. Hardware decisions are necessary
II. Software / GUI approach
   a. Pros:
      i. “Dumb” keyboard
      ii. More customizable
   b. Cons:
      i. OS interactions required
      ii. Software installation is required, potentially tedious

The team as a whole seemed to lean towards the embedded approach, but no final decision was made.

9/12

The team began working on the proposal document. The team decided to choose the embedded approach as it seemed to fit inline with the project’s goals best. The team laid out a tentative schedule to complete this document.

9/14

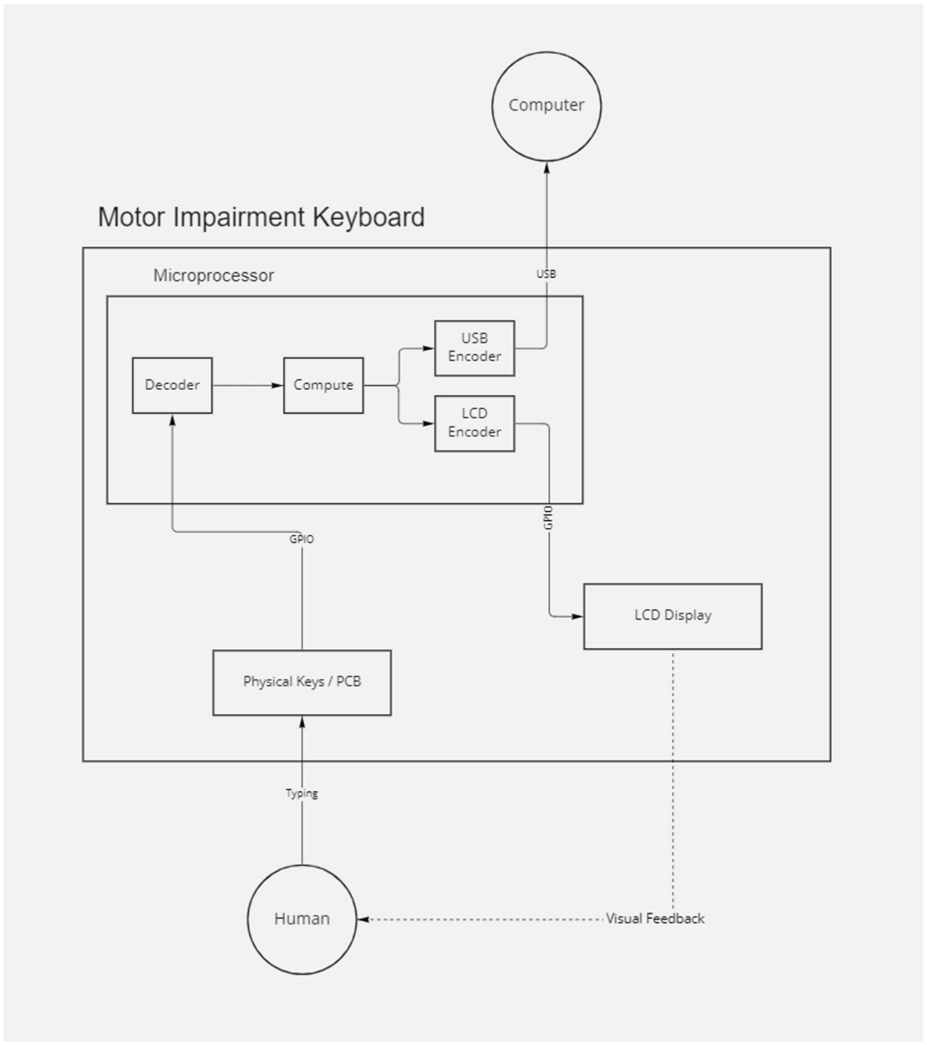
The team began serious work on the proposal document. A lot of discussion was had on what the project’s goal and objectives should be. Alternative solutions and existing technologies were explored. This discussion is best reflected in the proposal document so it will not be repeated here. Brittany also made an appointment with a contact at the Brazos Valley Center for Independent Living. I committed to going to this appointment along with Connie. We were able to complete the Needs Statement, Goals and Objectives, and Design Constraints and Feasibility sections as a team. We divided the remaining document into sections that each team member could finalize and left the Design specifications to be completed together on 9/17. Later that day I finalized the Literature and Technical Survey individually. I was able to divide existing technologies into three major categories and found some useful resources and references (these are reflected in the proposal).

9/15

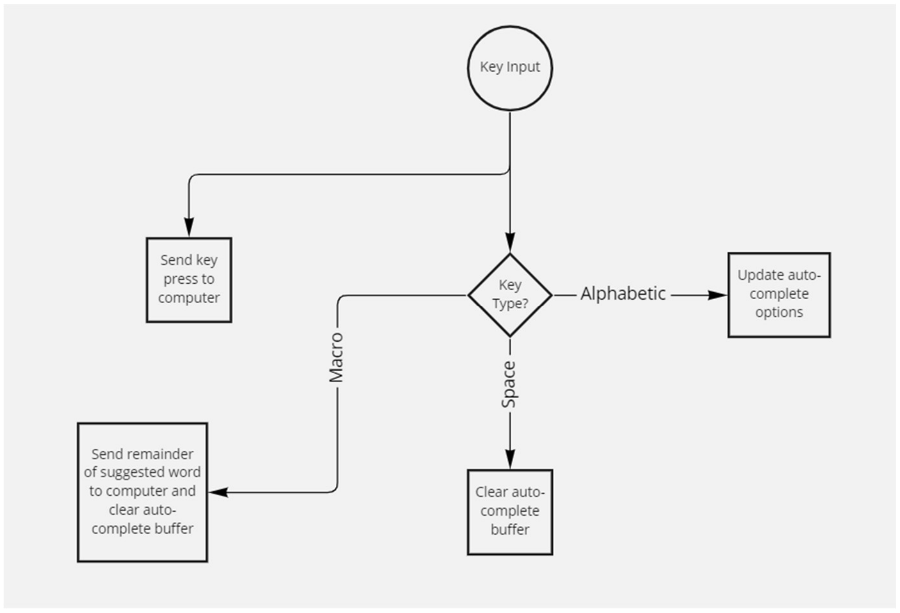
I, Brittany, and Connie met with Mac at the Brazos Valley Center for Independent Living. We has a discussion on our project with Mac and he gave us some input based on his software experience. He mentioned that assistive technology is expensive due to most customers not paying directly (but through insurance or otherwise). He Also mentioned his preference towards software as it can be updated and changed. He agreed with our conception that QWERTY was inefficient, but also agreed that it would be hard for others to transition away from it. Finally, he mentioned that they did not have potential users for us to work with but gave us some contacts.

9/17

The entire team met virtually to complete the majority of the proposal document. Much of this discussion is reflected in the document itself and will not be repeated here. A block diagram was completed as part of the design specifications. This diagram helped us finalize the major components of the project and show a distinction between hardware / embedded software. This diagram was completed as a team.



Max, Andrew, and I also created a rough flowchart on how the software should handle different key presses. This helped us decided on how the software / hardware interactions should be structured.


Key Input
Send key press to computer
Key Type?
Alphabetic
Update auto-complete options
Macro
Space
Send remainder of suggested word to computer and clear auto-complete buffer
Clear auto-complete buffer

The document was not completed, but what little was left is completable in the following days. Brittany was also able to create a Jira board for our project management and I created a GitHub.

GitHub: https://github.com/MIKeyTAMU/MotorImpairmentKeyboard

9/19

The team was divided into groups that worked on the remaining parts of the proposal and the presentation document. Together all the documents were mostly completed, with a few minor elements left to be finished individually. I wrote the executive summary section and helped with the presentation document. We decided to meet again tomorrow and finalize some items. This day also should have marked the start of spring 0, but the team delayed this due to presentation work that was needed.

9/20

The team met virtually to finish the presentation. In this meeting, the presentation was separated into sections for each member to present tomorrow. This short meeting concluded and I began practicing my section.

9/21

The team presented the proposal to the class and received some valuable feedback. The spring was also unofficially started. I began working on the skeleton code for the Trie and Node data structures. I discussed how to best accomplish traversal with Max and we decided on Nodes with an array of children pointers (one for each letter in the alphabet). This would be constant access time. A priority array would be used to keep track of the most likely candidates. I implemented the ideas we discussed and pushed these changed to GitHub. The majority of the Node class was completed. This included the design, constructor, add node function, and getter functions.

Node class:

```
/*
	Each node represents a character, c (technically redundant, but kept for
	simplicity). Each node has the capacity to be a terminal node, meaning it is
	the final character in some word

	An array of pointers to other nodes keeps track of what characters can follow
	the current node. a[] is ordered alphabetically (a[0] -> 'a', a[25] -> 'z')
	with a size of 26

	Array of priorities keeps track of what following characters are highest in
	priority (high to low) each item in p[] corresponds to the index in a[]
	(priority 1 -> a[p[0]]). There is one exception to this where p[i] is 26,
	indicating the priority (i) of the node being terminal (end of the word).
*/

class Node {
	char character;
	bool isTerminal;
	Node* array[26];
	unsigned char prioroity[27];
}
```

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/d55a71f56afd67bff9252a6f5075f9748e1f2955

9/22

I continued some work on the Node class. I added some better comments to explain my thought process and improved some of the code to make it more efficient.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/6f9b7e9905b739fca82a30b851ba933973b8f309

9/26

The team worked on the report form for the upcoming meeting on 9/28. Individual work continued on the project and the team discussed the best layout for the keyboard. I worked on integrating the GitHub with Jira but was unable to do so. I began work on the Trie's skeleton code. I wrote the Trie constructor which reads from a dictionary file to construct a Trie. The dictionary must be in priority order, so I found a sufficient one. I also updated the ReadMe in the GitHub.

Dictionary: https://github.com/first20hours/google-10000-english

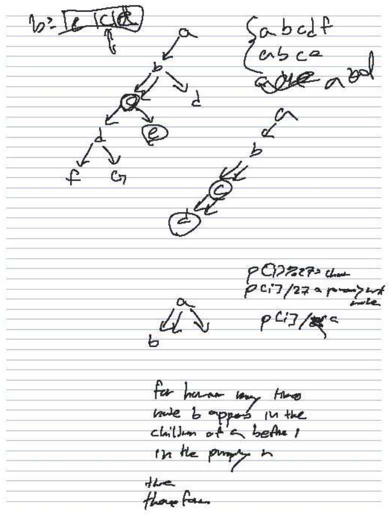
Commit 1:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/1eb5833dbd5ee1afc98dcd90a486486fd315a281

Commit 2:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/79b6a58b1fa70f07438b2bb1eb9cfd542e8d2721

Commit 3:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/6565bbece8ecfe9b3db883410200837031f18463

9/28

The team had their first formal meeting. Afterwards, SCRUM was held for the individual hardware / software teams. I explained my work to Connie and Brittany. Then, Max, Connie, Brittany, and I had a long discussion on algorithms to get three candidates for partial words. We discovered shortcomings in our existing solution and bounced ideas off each other. We pointed out potential algorithms and found flaws in each of them. The algorithm that was decided upon required minimal changes to the existing Node and Trie classes. I then wrote the entirety of the algorithm as well a test program for it.



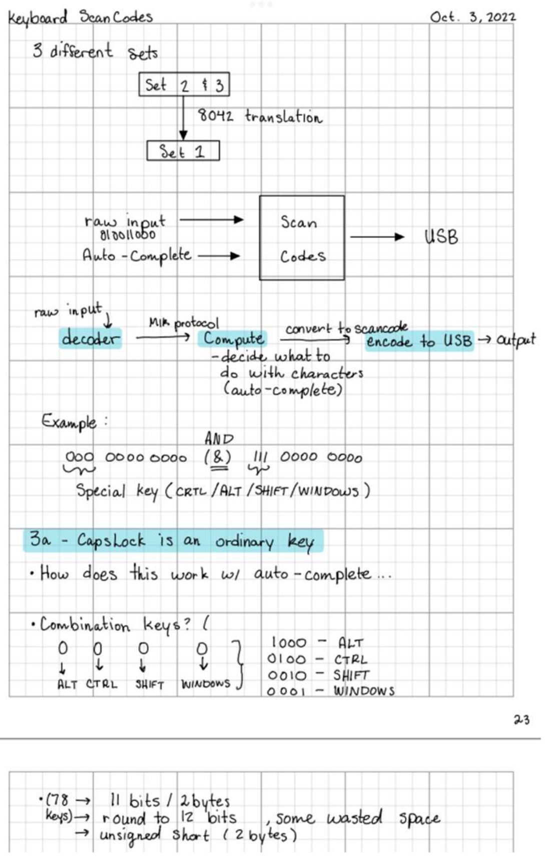
Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/42c7d09391faa4d82352c6bcd9fa201844100d84

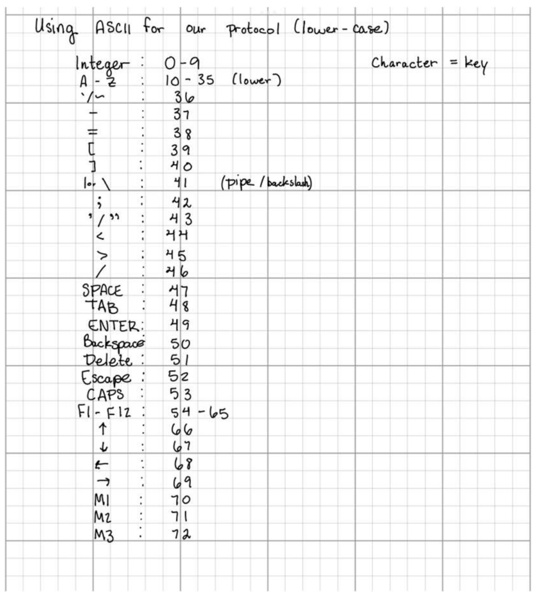
10/2

Added documentation to the Trie and improved efficiency slightly. Continued testing the auto-complete algorithm to a sufficient degree.

10/3

Small changes to auto-complete test file. Restructured directory for the GitHub to allow for better management. Added a ReadMe file for the software section, explaining the implementation up to this point. There was a long discussion during SCRUM about the best way to represent keystrokes (scan codes). It was determined that a special encoding mechanism should be used. This was then designed by me and Max. This takes the form of a short in C++ (2 byte integer). The first 4 bits represent whether special keys have been pressed (ALT, CTRL, SHIFT, and WIN). The final 8 bits are used to represent the character key being pressed (UP, a, 4, etc.). Then, a mechanism to decode such a type was written as a proof of concept by me.





Using ASCII for our protocol (lower-case)

Character = key

Integer	0-9	
A - Z	10 - 35	(lower)
`/~	36	
-	37	
=	38	
[	39	
]	40	
| or \	41	(pipe / backslash)
;	42	
'/"	43	
<	44	
>	45	
/	46	
SPACE	47	
TAB	48	
ENTER	49	
Backspace	50	
Delete	51	
Escape	52	
CAPS	53	
F1 - F12	54 - 65	
↑	66	
↓	67	
←	68	
→	69	
M1	70	
M2	71	
M3	72	

Commit 1:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/259fcec928e7a32499dc88350c15f718312c92fa

Commit 2:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/a53cdabd1e2c1f003e5c9ee056b8795fb717c3b1

Commit 3:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/d4ec630a39c8b1158128b0e9f1a1b713af0d2e4c

10/4

I continued writing code to decode the keystrokes. I played around with the best way to do this and landed on a final implementation. Also added contributions to the weekly meeting report.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/af8f350452df67be70452d2175d0248b6beac732

10/5

Had weekly meeting and presented my work. Did some light testing for MENA.

10/11

Worked on the weekly report.

10/12

Presented weekly report. Discussed with team steps to complete moving forward. Most of the emphasis was placed on hardware progress. Was able to assist with discussions on this topic. Began starting work on the CDR presentation document.

10/16

Worked on the CDR presentation. Completed the software design section explaining the work done up to that point. This included the auto-complete algorithm specification, Trie and Node classes, and MENA protocol (not going to repeat what was said here). Also worked on the project management, validation and testing procedures, and preliminary results sections. This comprised a significant percentage of the overall document.

MENA Structure

Bit(s)	0	1	2	3	4-7	8-15
Key	ALT	CTRL	SHIFT	WIN	(unused)	Character Key

MENA Character Key Table

Integer Value	Character Key
0-9	0-9
10-35	a-z
36	`
37	-
38	=
39	[
40	]
41	\
42	;
43	‘
44	<
45	>
46	/
47	Space Bar
48	Tab
49	Enter
50	Backspace
51	Delete
52	Esc
53	Caps Lock
54-65	F1-F12
66	UP
67	DOWN
68	LEFT

69	RIGHT
70	Macro 1
71	Macro 2
72	Macro 3

10/17

Worked on the CDR presentation document along with Connie. Together, we completed the vast majority of the presentation slides. Continued work with Max on interfacing with hardware. Was able to write a test program to demonstrate the auto-complete algorithm working with a few sample buttons (representing keys). This was then shown during a demonstration that was recorded. Later that day, I worked on standardizing the code. I wrote skeleton code for a driver program that will eventually become the final program. This code incorporates MENA, the auto-complete algorithm, Node and Trie classes, and has spaces for the USB output, LCD output, and GPIO input. Made a make file for this program and also added commands to run the earlier test programs. Also updated the software documentation to the newest specifications. Also worked on assigned presentation slides.

Demo: https://youtu.be/nU4BqcodI7o

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/8357c1331c4a70d305c5565fdde8a7451497cdf7

10/18

Practiced assigned presentation slides for CDR report.

10/19

Conducted the CDR presentation during class. Presented the software design section along with some closing remarks and the demonstration. Afterwards, updated the other team members on my progress and the state of the software portion. Hardware is still taking priority.

10/23

Met with Brittany, Andrew, Abhishek, and Max to work on hardware. Had some discussion on the best way to do hardware decoding and how to go about the PCB design process. I was also able to find a better C++ library for reading GPIO inputs. I made a simple test program for this on my Raspberry Pi.

10/24

Had a long programming sessions where I was able to write code to output keystrokes over USB. I found a library that can accomplish this over the command line and was able to issue these commands from within a C++ program. This was then showed to be working with sufficient success. I demonstrated this to the other team members. With this substantial edition, the final software portions of the project are reading GPIO input and outputting to LCD's. Everything else (auto-complete, USB output, MENA protocol) has now been implemented by me. Spent some time cleaning up code.

USB resource: https://mtlynch.io/key-mime-pi/

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/b4346b53b74c5c94c8d546d12783689941ee6779

10/26

Worked with Max on hardware interrupts. It was determined that the Raspberry Pi can power the PICO and over the same connection get input. This eliminates the need to read over GPIO and GPIO can only be used to output to the LCD's.

10/31

Continued working with Max on getting hardware interrupts working. Several approaches were attempted with several bugs being encountered. Troublingly, it was found out that serial communication between the PICO and Pi is not well supported (the Raspberry Pi has terrible C++ support).

11/1

Worked on weekly report.

11/2

Worked with Max and Abhishek on getting the LCD's working. Abhishek's code was adapted to the current solution. I edited the driver program to utilize this code. Then worked with Max to try and get the hardware interrupts working. Eventually a potential approach was decided upon, but code could not yet be done for this. The approach would be the PICO sending keystrokes to the Pi that contains the MENA encoding. While this is extremely inefficient, due to bad support for C++, and limited time to develop an alternative, it was decided upon.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/6574121c07973743e8de8f582da9a739f7d87fc5

11/7

Worked with Max on getting the Pi to PICO communication working. The previously discussed method was used and was confirmed to be working. This finally demonstrates a communication avenue between the two controllers. The method works as this: the PICO sends a hardware interrupt to the Pi over GPIO. When the Pi receives this interrupt, it runs code to accept a MENA encoding from standard in, which the PICO sends immediately afterwards. This is not preferable, but it does reliably work.

11/9

Worked on the weekly report and cleaned some code up.

11/14

The Pi to PICO communication was finalized. I went ahead and pushed this code that had previously been worked on.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/8e8746d001c089fb581fba129c314520fc9d1112

11/15

Worked on the weekly report and outlined what needed to be done before the project could be finished. All remaining worked had to do with the PCB.

11/16

Worked as a team to get all components of the keyboard working before the break. Every individual component was confirmed to work. End to end functionality confirmed.

11/27

Met with team to finalize the project for demonstration. Several key bugs were identified and fixed. I fixed backspace functionality, worked with Max to get shortcuts fixed, and implemented several other changes. End to end functionality was shown to work with a laptop.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/273ae9b8c7897dc561084 7d3eb32d87eaa064c4b

11/28

Demonstrated the prototype. Afterwards a few bugs were fixed. Then, I worked with Max and we were able to drastically reduce the latency (to within ~10 ms of a normal keyboard). This was done by changing a few parts of the PICO's code. Then, I worked with Brittany on performing some user testing. We spent some time typing passages and recording how many keystrokes were necessary on our keyboard vs a normal keyboard. It was found that the auto-complete algorithm predicted 39 of the 40 typed words in one example and saved on the number of keystrokes that must be pressed drastically. Then, I led a user study with the team members on reaction time. Four of us took turns measuring our reaction time with a normal keyboard and our keyboard. No significant difference was found between the two average measurements, indicating little latency. Later on, I worked on standardizing the GitHub so the other members could add their contributions. I edited a few ReadMe files and changed the directory structure.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/dd21240d889387815b65f 21610b005fa0b913614

11/29

Renamed some files in the GitHub and added all the documents completed up to this point.

Commit:
https://github.com/MIKeyTAMU/MotorImpairmentKeyboard/commit/51c583d5f2dca50c8a19a8 d4dc79e2a2c7dcaadd

11/30

Fixed a small bug where capitalized letters did not work with auto-complete. Did some further user testing in class and practiced by typing a short C++ program using the keyboard. Took the keyboard back with me to complete a demonstration video. Spent time recording different components of the demonstration. Edited them together to show the different aspects of this using Adobe Premiere.

Demonstration video: https://www.youtube.com/watch?v=CSIN_KwBNN4

12/2

Worked on the final report and presentation.