

Motor Impairment Keyboard



Critical Design Review

MIKey

Jackson Hagood

Andrew Imwalle

Brittany Jenkins

Connie Liu

Abhishek More

Max Smith

Department of Computer Science

Texas A&M University

10/19/2022

Table of Contents

1	Introduction.....	3
2	Proposed design	3
2.1	Updates to the proposal design (if applicable).....	3
2.2	System description.....	4
2.3	Complete module-wise specifications	5
2.3.1	Hardware.....	5
2.3.2	Software	6
2.3.3	Modeling.....	8
2.3.4	Parts list.....	8
3	Project management.....	9
3.1	Updated implementation schedule.....	10
3.2	Updated validation and testing procedures	12
3.3	Updated division of labor and responsibilities.....	13
4	Preliminary results	15

1 Introduction

The world is in a digital age where usage and navigation of computers is significantly important for the population. Unfortunately, not everyone is able to efficiently use computers, as they are not designed for those with physical disabilities.

People with tremors or motor function impairments experience trouble when interacting with traditional computer peripherals. As use of technology becomes more prevalent, there exists a need for a simple assistive technology that provides better computer hardware accessibility to individuals with limited motor function. Existing assistive technologies are often too restrictive for those with mild to moderate motor function impairments and tend to sacrifice efficiency for the user. As a result, there is a need for a more traditional form of interaction that is easier for those with mild impairments to utilize.

The goal of a potential solution would be to make the typing experience easier for those who have motor function impairments. The solution should be cost-effective, accessible, and should also improve the experience of interacting with their computer. Priority must be placed on ensuring the solution is easy to learn, as a large percentage of those with motor function impairments are typically older in age and likely to be less proficient with computers. Therefore, avoiding the need to install device drivers or other software is preferable. The solution should also be generic enough to be customizable depending on the specific needs of each user. The design should not compromise any of the abilities present in a normal keyboard. Ideally, a solution would add to a normal keyboard's capabilities. The aim of these capabilities should be to reduce the user's opportunity to make mistakes while typing and improve their typing speed. Another important objective is to keep the solution as cost effective as possible in order to keep the product in reach of the general user. A final objective is to follow a "helping the disabled helps all" philosophy. Many devices for those with disabilities end up helping others in unintended ways. A solution to this problem could end up having uses for a wide variety of circumstances, even for those without motor function impairments.

Several constraints presented in the needs statement involve the price, functionality, and compatibility of the product. In terms of price, we aim to deliver a final product that does not exceed \$200. Additionally, users will ideally be able to set up and take advantage of core functionalities of the product on their own. We also want the user to be able to use this product with any computer in a variety of settings, so it must be reasonably portable and compatible across Linux, Windows, and Mac. To do this, our solution will be based on existing technologies with the form factor similar to a standard USB-A, QWERTY American-English keyboard. Lastly, our development time is constrained at 10 weeks. We have determined that a majority of these constraints are feasible given our technical expertise, but our biggest challenges will likely be the final cost of the product and limitations on development time.

To ensure that our design is valid and successful, we will utilize user testing from the targeted demographic. For conducting the testing, two major types of user feedback will be obtained. Initially, the users will be asked to type a sample passage on a standard keyboard they are familiar with. Their typing speed and accuracy will be measured using a typing speed program/website. Then the user repeats this process for another passage of similar length and difficulty using the keyboard designed by the team. The measured results will be compared. After this quantitative data is gathered, a semi-structured interview will be conducted. The interview will center around their experience with the keyboard along with a questionnaire that will ask the user several questions regarding their experience using the keyboard. This will give qualitative insight into the user's typing experience.

2 Proposed design

2.1 Updates to the proposal design (if applicable)

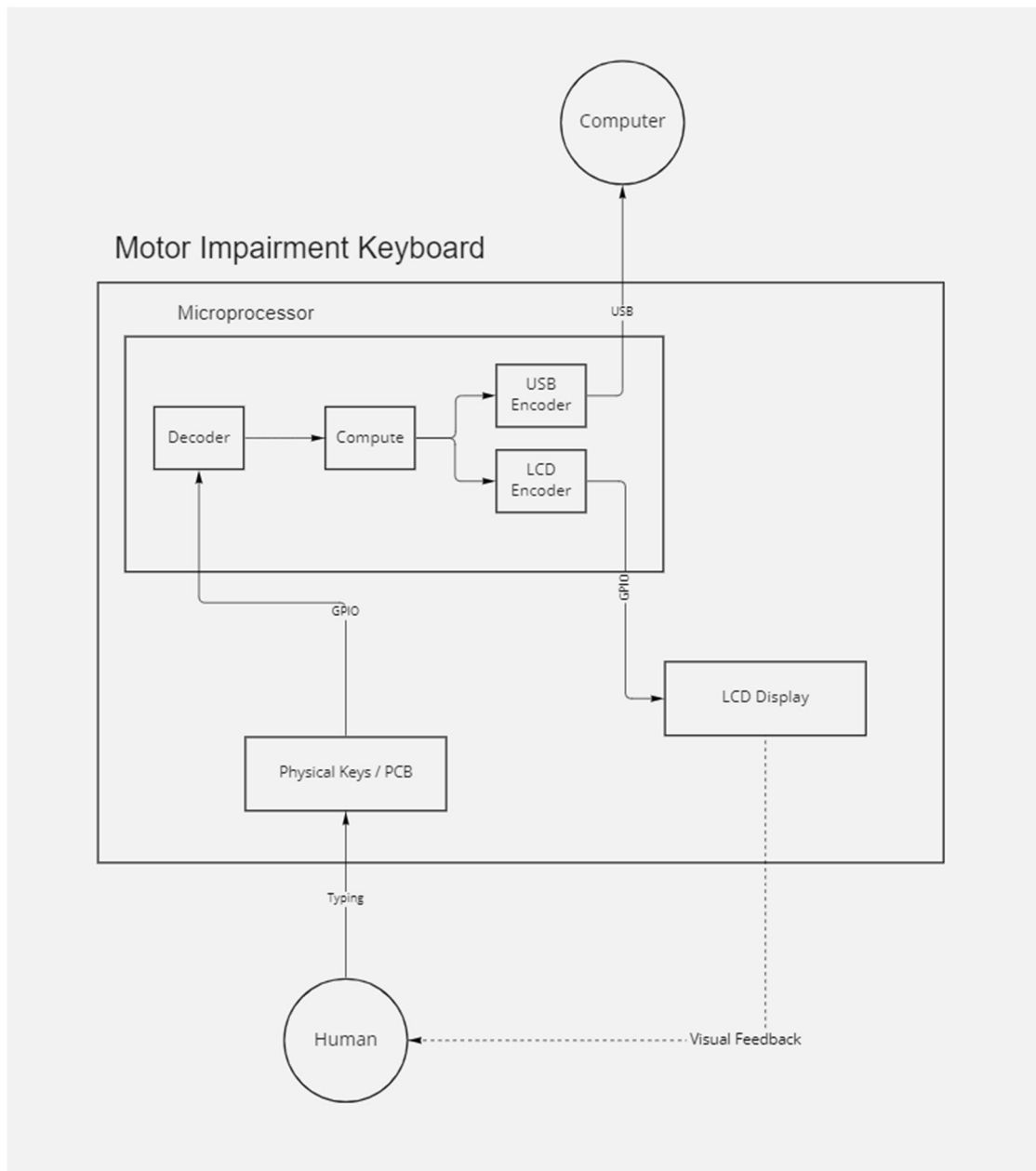
As of this time, no significant changes to the overall objective described in the proposal design have been made. Also there have been no new findings or alternatives discovered for this project that

have caused us to redirect our attentions. The only major updates to the proposal design involve several budget changes.

Discussions were held on the best switches to use. It was determined that slim profile mechanical key switches would work best for the form factor that was desired. One effect of this choice, however, was that O-rings are no longer compatible with the chosen switches. Thankfully, this was a secondary priority and will not likely have a large effect on the final product.

When looking into options regarding the Raspberry Pi, the group found a cheaper alternative that will perform the necessary tasks in the Raspberry Pi Pico. This is a smaller device designed to run physical computing projects.

2.2 System description



Our product comes in the form of a traditional keyboard, with some modifications that make typing easier for motor-impaired users. The user-facing hardware (the keycaps, key switches, and the external frame) is described further in section 2.3.1.

The Motor Impairment Keyboard (MIK) is surrounded by an input and output. To start off the input stage, the user will type their desired text on the keyboard, which will send a signal input to the Pi via the GPIO pins. Our text suggestion algorithm requires text input, so a decoder is needed to convert the raw GPIO input into text characters and send the data to our software. The user can also choose to select three macro keys that correspond to the three text suggestions that we will deliver. If a macro key press is detected, the suggested text is directly sent to the user's computer.

The compute module will run our algorithm and discover possible text suggestions. The algorithm is further described in detail in section 2.3.2. These suggestions must be available to the user so an LCD encoder is used to convert the text into GPIO signals, which are passed to the LCD display. This LCD display shows the three best text suggestions to the user. Simultaneously, the text is also sent to the USB encoder which converts the text into data that can be sent through the USB protocol to the user's computer.

2.3 Complete module-wise specifications

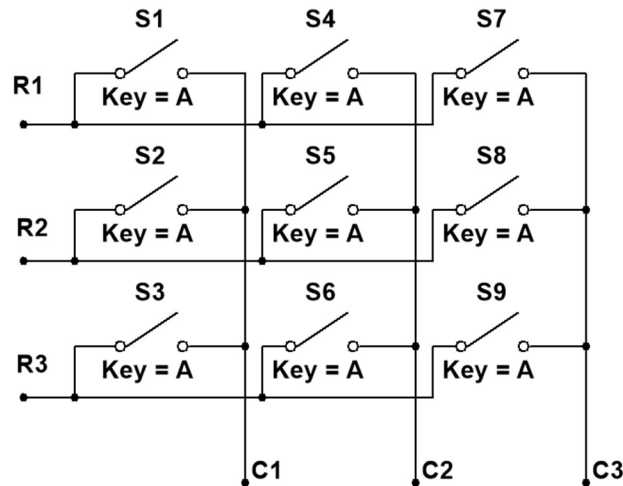
Progress has been made in all areas (hardware, software, and modeling design) of the project. Including, but not limited to, an updated polling/interrupt scheme within the hardware system, a finalized custom keyboard layout, a custom keystroke encoding system, and modification to the keycap model design.

2.3.1 Hardware



A custom layout for the keys has been created by the team to ensure that only the necessary keys would be present in the keyboard. This layout is a slightly modified QWERTY layout to ensure that the user is familiar with the positioning of the keys. The keyboard and individual keys will be oversized to assist with accuracy. The keys will be sunken into the keyboard housing, similar to a sunken keyboard, also to assist with accuracy and speed. The keycaps will be a slim design as to not make the keyboard too tall, which is an aesthetic as well as functional decision. The three macros placed underneath the spacebar is to allow the user to select from the auto complete options.

The LCD display will be built into the keyboard and will receive GPIO signal input from the microprocessor. It will display up to three suggestions that the user can select. As the user continues to type, the LCD will update. The font will be large and high contrast to ensure that the user can read the suggestions clearly.



In order to retrieve key presses from the keyboard, MIK will use a key scanning matrix, as shown above. This circuitry will be driven by a secondary microprocessor that will generate interrupts to send to the primary processing unit, along with a MENA code. In order read key presses, the secondary microprocessor will continuously poll the hardware by individually powering each row of the matrix and checking each column to see if the switch at that specific row/column coordinate has been closed. If a high signal has been detected at a specific row, the row and column are used to index a mapping from the key scanning matrix location to a MENA encoding. As mentioned above, an interrupt is then generated and the MENA code is sent to the primary processing unit to be handled. In order to reduce the number of GPIO pins used, a decoder will likely be used for the powering of the rows. This will reduce the number of pins from $R + C + N$ to $\lg(R) + C + N$, where R and C are the number of rows and columns respectively, and N is the number of pins used for other purposes.

2.3.2 Software

A new keystroke encoding system, MENA (MENA Encoding Not ASCII), was devised to encode the specific keys that we have included on our finalized keyboard layout within our developed auto-complete algorithms. 73 standard “character” keys must be trackable in MENA (1 byte). In addition to this, the encoding must be able to store whether special keys (ALT, CTRL, SHIFT, and WIN) have been pressed in combination with these regular keys. Therefore, the first 4 bits of the encoding value are used to keep track of whether these 4 special keys are pressed. To store all this information MENA requires two bytes (a `short`) with the first byte keeping track of the special keys and the last byte keeping track of standard (character) key presses. The following tables outline this encoding and key-integer mapping.

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	`

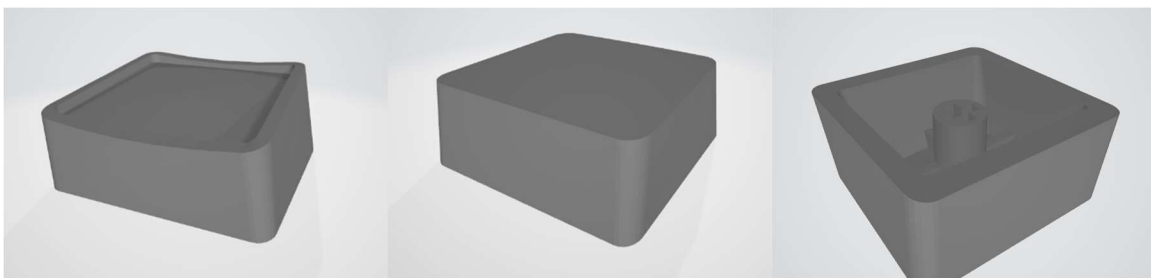
37	-
38	=
39	[
40]
41	\
42	;
43	'
44	<
45	>
46	/
47	Space Bar
48	Tab
49	Enter
50	Backspace
51	Delete
52	Esc
53	Caps Lock
54-65	F1-F12
66	UP
67	DOWN
68	LEFT
69	RIGHT
70	Macro 1
71	Macro 2
72	Macro 3

The design for the auto-complete algorithm has been refined but not drastically altered since the proposal stage. The auto-complete algorithm makes use of a Trie data structure which itself has instances of Node objects. Looking first at the Node class, each node represents a single character that is part of one or more words in a dictionary (stored as a `char` in the `character` member variable). This node has 26 possible children, one for each character in the alphabet. These children are stored as pointers in a fixed length array, `children`, in alphabetic order (index 0 is the address of child 'a'). Whenever an index in this array is `nullptr`, no child exists for that character. In addition to having children, each node can also be the terminal node for a word in the dictionary. This is kept track of using a `boolean` variable `isTerminal`. The final member variable in the node is an array of 4 `unsigned chars` used to keep track of the highest priority children. This is important for the auto-complete algorithm's traversal later. These member variables are accessed with a constructor, getter functions, and setter functions.

The Trie class makes use of these node objects. The Trie's only member variable is an array of 26 root node pointers, one for each character in the alphabet. This class works through two central functions: a constructor and `getCandidates()` function. Looking at the constructor, a filename is used to access a dictionary file for construction. This dictionary is expected to be composed of words ordered from highest frequency to lowest frequency. The dictionary chosen for this project is composed of the 10 thousand most used English words. The constructor first initializes each of the 26 root nodes and then opens the dictionary file. Then, the file is traversed for each word in the dictionary. For each word, the constructor searches for an existing node and moves a pointer to it if it exists. If no such node exists, it is created, and a pointer is then moved there. Before moving to the next character in the word, the constructor also sets the priority of the previous node. If one of the 4 free priority slots are free in the previous node's priority array, the node is added in the next available slot. This is done using modulus such that the existence of the node earlier in the array is accounted for so during the auto-complete calculation the same word is not traversed twice.

The other major component of the Trie class is the `getCandidates()` function. This function accepts a string representing user input (a partial word) and gives the three most likely candidates. First, the function traverses through the Trie (starting at a root) using the partial word to find the last known node (or decide if no such words exist). Then, the algorithm will traverse subsequent nodes using the priority arrays until a terminal node is found of sufficiently high priority. The values of the priority array are setup such that no word is ever traversed twice. The algorithm also avoids suggesting only what the user input if they gave a valid word. This is performed three times before the function terminates.

2.3.3 Modeling



The initial design of the keycap was similar to a traditional mechanical keyboard keycap with a few changes. These changes were to enlarge the surface area in order to allow the key to be easier to press, and to inset the top of the key into the outer rim to ensure that when a key is being pressed, the user would not accidentally press another key.

When the first set of keycaps were printed, there were a few issues. The first of which was the fact that the stem which interfaces with the switch was not consistent and uniform in the sizing. This proved to be a non-issue as the switches still fit inside the keycap consistently. Another issue was due to the shape of the top of the keycap. The keycap is printed upside down to ensure that the stem of the part would be printed with the best orientation to ensure a successful print. This means that there needs to be support on the inset part of the key. This proved to be difficult to remove from the model. When printing 80 keycaps for a prototype, this would not be efficient. The benefit that they inset model provided would also be covered by the fact that the keys will sit inside a rigid frame. This ensures that the user does not have any issues pressing the correct key and removes the need for an inset keycap. Due to these issues, the top of the keycap was changed so that it would be flat while remaining oversized.

2.3.4 Parts list

Below is the updated itemized list of the parts that will be used for our custom mechanical keyboard.

Product Name	Cost (dollars)	Purchase Site
--------------	----------------	---------------

Raspberry Pi Pico	13.99	Amazon
LCD Displays	23.79	Amazon
Keyboard Switches	47.98	Amazon
PCB Material	~90	jlcpcb.com
Total	177.9	

All of the above parts have been ordered and have arrived, except the PCB material. The PCB is currently a lower priority as we are primarily working on the software and early stages of the hardware.

3 Project management

Due to the early time dedication to organize the team, the roles and responsibilities have not deviated from the original plan. This has worked well for our team and allowed us the flexibility to float between tasks as well as be accountable for the results of certain items. The responsibilities of the team have been divided such that each person has at least one technical and non-technical responsibility. Everyone on the team has software experience, so the hardware technical roles were divided among those who have the most hardware experience. This responsibility will fall on Andrew, Max, and Abhishek. Connie, Brittany, and Jackson will be the team members who will prioritize software development. Finally, due to the nature of our product, 3D Printing and Modeling is another technical role that must be filled. Fortunately, Andrew and Jackson have experience and skills to fill this role.

Technical Role	Members Involved
Hardware Team	Andrew, Max, Abhishek
Software / Embedded	Connie, Brittany, Jackson
3D Printing / Modeling	Andrew, Jackson

Non-technical duties have also been divided among the team. Brittany has been the point of contact for outside resources for mentorship, assistive technology, and disabilities resources, and she will continue to do so. Brittany also has experience with project management and will do the planning for the team. The team has decided to utilize agile software development, and Andrew and Jackson will be the SCRUM masters due to each being on either the hardware or software team. Connie and Max will handle most of the technical reports and the meeting notes because they also span both the hardware and software teams. Finally, Abhishek will handle the finances and purchasing for the team's prototype.

Non-Technical Role	Members Involved
Project Manager	Brittany
Hardware SCRUM Master	Andrew
Software SCRUM Master	Jackson
Communications	Brittany
Finance	Abhishek
Technical Reports / Meeting Notes	Connie, Max

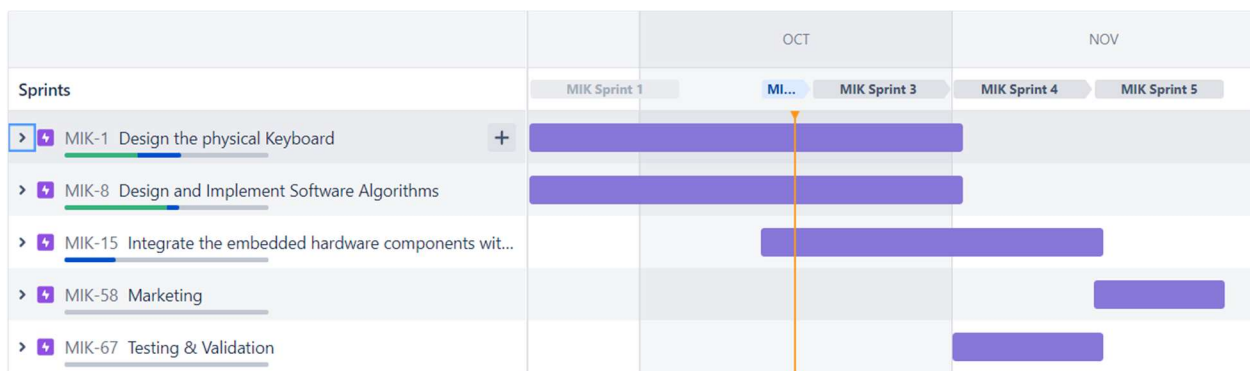
The team has maintained use of the same coordination tools including Github, Jira, OneDrive, and Discord. There was some learning curve to utilizing Jira to its fullest potential, but most team members have now had enough experience with the platform to create, update, and assign tasks as needed. This helps us keep track of the progress of sprints with a reasonable amount of effort. Github has allowed us to maintain version control for our project and organize the code in accordance with sub-team tasks.

3.1 Updated implementation schedule

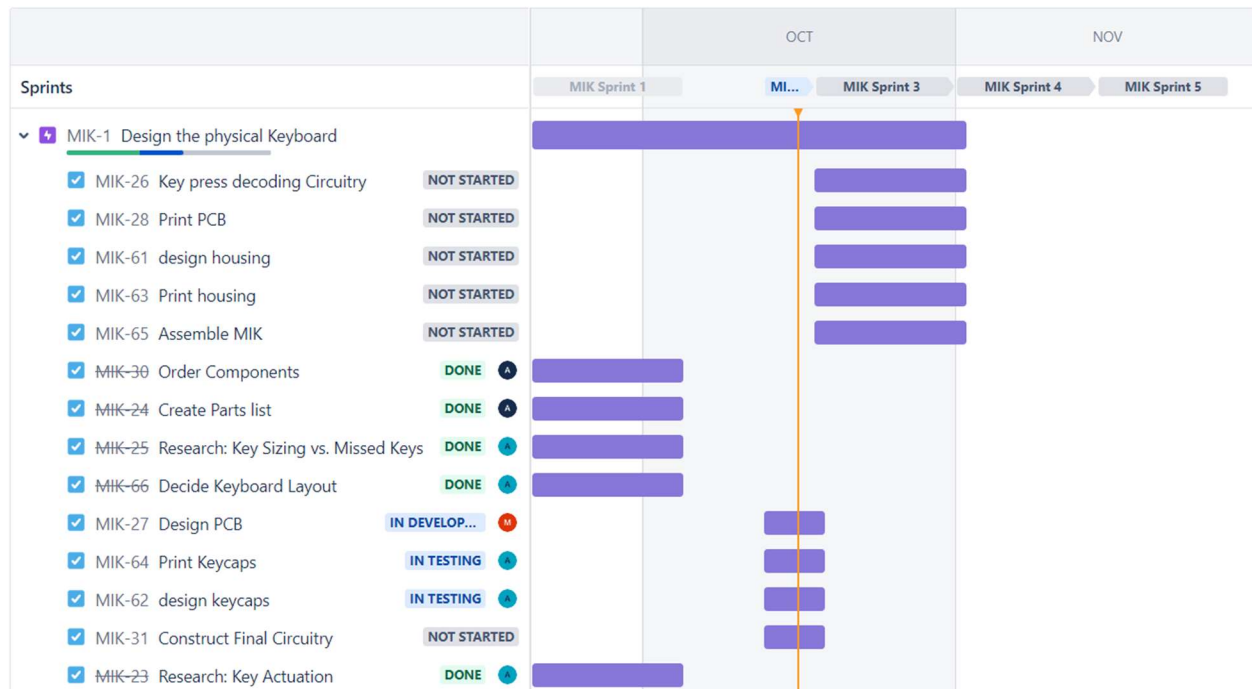
Our timeline has not changed since the beginning of Sprint 1. Below is the outline of the beginning and end of each sprint.

Sprint #	Dates	Status
Sprint 1	09/20/2022 - 10/04/2022	Complete
Sprint 2	10/04/2022 - 10/18/2022	In Progress
Sprint 3	10/18/2022 - 11/01/2022	Not Started
Sprint 4	11/01/2022 - 11/15/2022	Not Started
Sprint 5	11/15/2022 - 11/27/2022	Not Started
Final Reporting	11/27/2022 - 12/05/2022	Not Started

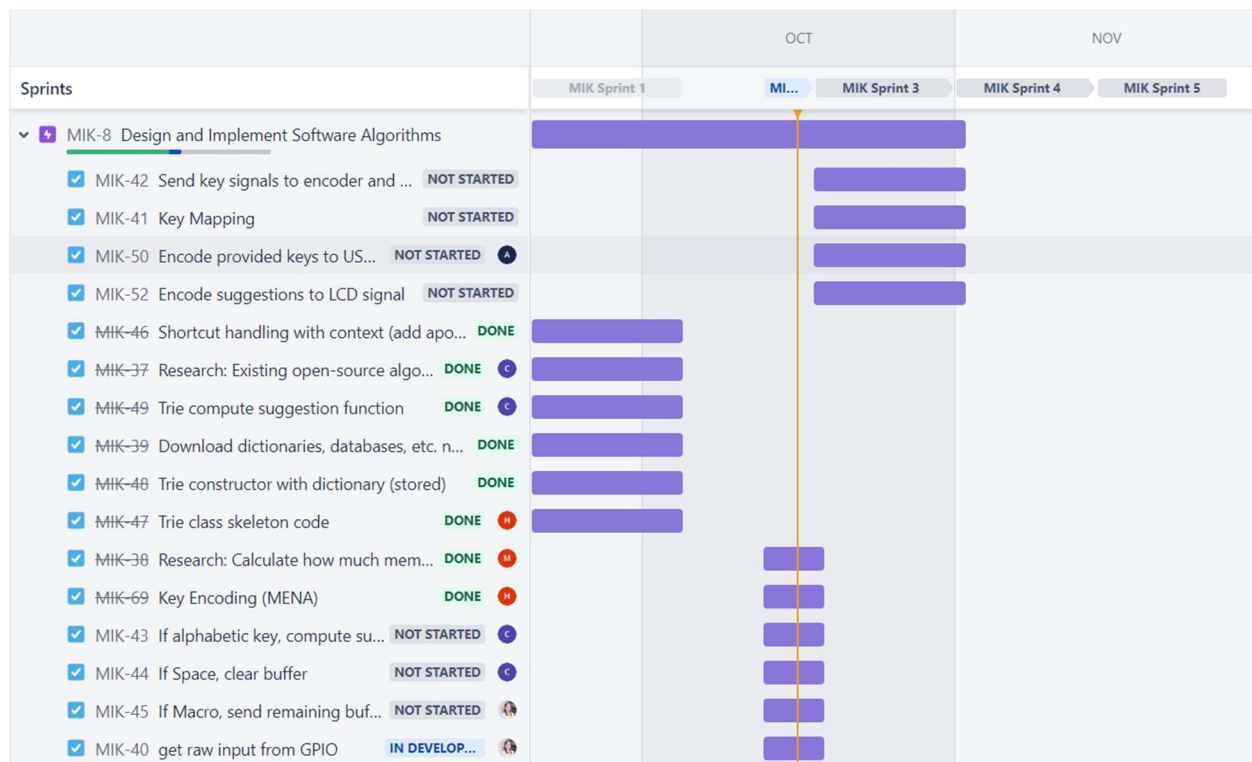
Below is the overall outline of tasks for sprints 1 through 5. The tasks are organized into epics, or larger portions of the project, that correspond to sub-team tasks. The Gantt Chart below is from our Jira board, and it shows that we intend to front load hardware and software tasks in the first 3 sprints. This allows us to focus on integration and testing for sprints 4 and 5 in November. This will be extremely helpful during Thanksgiving break, because we have users for prototype testing in a team member's hometown. Finally, Sprint 5 will be used for marketing evaluation and a second iteration of testing after adjustments have been made to the keyboard.



The next photo expands the hardware tasks covered in sprint 1 through 3, several of these tasks were completed in sprint 1, but due to supply chain issues we had to push back several hardware tasks to sprint 2. With the supply chain issues, there was a difficulty where the parts were expected to ship by a date, but when they took longer than expected, a second order was placed and the original parts were refunded. This did not cancel the shipment of the original parts and resulted in 2 sets of the ordered parts arriving for the team. Our parts arrived over a week later than we expected them, which fortunately caused minimal setbacks due to Fall Break at the university. Because of this, Sprint 3 will be hardware heavy, while integrating with the software.

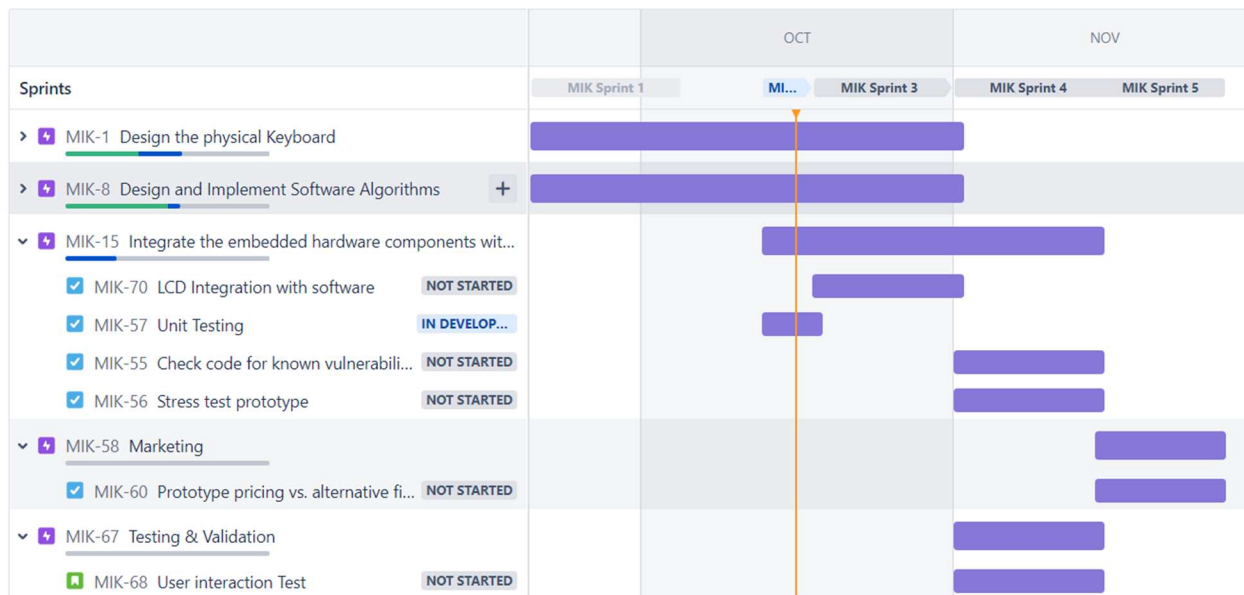


Much of the software has been handled in the first 2 sprints, and sprint 3 code focuses on integration and lower-level functionality with the circuitry. We were able to get ahead of schedule with the software, which will help us since we had setbacks ordering parts.



Finally, modeling, marketing, and testing will mainly take place in sprints 3,4, and 5. Modeling focuses on assembling the keyboard once the hardware and software have been implemented. We would

like to have 2 iterations of testing, so MIK-67 will roll over into sprint 5 and adjustments will be made to our product based on the prototype testing done in those sprints. Marketing will focus on finalizing our product and begin reporting on the project and future implementations to drive down costs to the buyers.



3.2 Updated validation and testing procedures

Once a sufficient prototype ready for testing is completed, user testing will begin. There will be two avenues for this to occur. Firstly, one of the team members has a family friend afflicted with Parkinsons. This person has agreed to assist with testing around later November. Secondly, the team's point of contact, Brittany, has been in correspondence with several organizations centered around related disabilities since the beginning of the project. Effort has been made to find candidates for testing the prototype through one of these organizations.

For conducting the testing, two major types of user feedback will be obtained. First of all, the users will be asked to type a sample passage on a standard keyboard they are most likely familiar with. Their typing speed and accuracy will be measured using a typing speed program/website (such as typing.com) and recorded. Then they will repeat this process for another passage of similar length and difficulty using the keyboard designed by MIKey. Again, their speed and accuracy will be recorded. This will be repeated several times for each user as they become more familiar with the MIK. After this quantitative data is gathered, a semi-structured interview will be conducted. The interview will center around their experience with the keyboard. Also, a questionnaire will be prepared to ask the user several questions regarding their experience using the keyboard. This will give qualitative insight into the user's typing experience.

One important consideration we have taken for the interview, is whether our research needs to be approved by an Internal Review Board (IRB) for Human Research. We believe that our experiment will qualify for exemption according to IRB standards. First, we needed to ensure that our research did not require any HIPAA identifiers. This means we will not record any personal health information (PHI) or personally identifiable information (PII) when conducting research. Therefore, all data will be recorded anonymously by not taking a testers name, geolocation, date of birth, or any diagnosis dates. In general, age range, how long a user has had a medical condition, and whether a user is retired are not subject to HIPAA when provided by the users during an interview. We plan to give users a notice of their rights and let them know that no identifiable HIPAA data will be recorded.

We have self-determined that our experiment is likely exempt from IRB oversight, but we will still need to submit an approval form to iRIS, TAMU's IRB platform, before we begin testing our prototype. We have made this preliminary determination by deciding demographic questions will remain vague and user testing will only require minimal risks, since they will only be expected to type on a keyboard.

Interview/Demographic Questions





- Q1: Do you suffer from a motor function impairment disability?
- Q2: What is the severity of your motor impairment?
- Q3: How often do you type using a keyboard?
- Q4: Do you currently use any assisted keyboard technology?
- Q5: What is your age range?
 - 18-25
 - 26-35
 - 35-50
 - 51-65
 - 66-75
 - 76+




Questionnaire

Q1: The Motor Impairment Keyboard was easier to use than the normal keyboard.				
Strongly Agree	Slightly Agree	Neutral	Slightly Disagree	Strongly Disagree
Q2: I felt I typed faster on the Motor Impairment Keyboard than the normal keyboard.				
Strongly Agree	Slightly Agree	Neutral	Slightly Disagree	Strongly Disagree
Q3: The auto-complete screen on the Motor Impairment Keyboard helped me type faster.				
Strongly Agree	Slightly Agree	Neutral	Slightly Disagree	Strongly Disagree
Q4: If this keyboard was on the market, I would consider purchasing it for personal use.				
Strongly Agree	Slightly Agree	Neutral	Slightly Disagree	Strongly Disagree

3.3 Updated division of labor and responsibilities

Sprint 2 focuses preparing for the CDR by implementing initial hardware and software sub-tasks separate from one another. These tasks are due by October 18th, and they are assigned to members in the picture below. The Legend shows which user icon belongs to which team member.

Jira Icon	Member Name
	Brittany Jenkins
	Max Smith
	Andrew Imwalle
	Abhishek More

	Connie Liu
	Jackson Hagood
	Unassigned/Multiple team member's responsible

It is likely that some of the software tasks will be moved to sprint 3 at this point due to dependencies, and our focus has shifted to getting hardware caught up to allow for both sub-teams to move forward in sprint 3. Unit testing and Constructing circuitry are both unassigned because multiple team members will be working on these tasks.

MIK Sprint 2 13 Oct – 18 Oct (11 issues)			9	18	0	Complete sprint	...
<input checked="" type="checkbox"/>	MIK-38 Research: Calculate how much memory is needed for storage	DESIGN AND IMPLEMENT SOFTW...	2	DONE		M	
<input checked="" type="checkbox"/>	MIK-69 Key Encoding (MENA)	DESIGN AND IMPLEMENT SOFTW...		DONE		H	
<input checked="" type="checkbox"/>	MIK-43 If alphabetic key, compute suggestions	DESIGN AND IMPLEMENT SOFTW...	2	NOT STARTED		C	
<input checked="" type="checkbox"/>	MIK-44 If Space, clear buffer	DESIGN AND IMPLEMENT SOFTW...	2	NOT STARTED		C	
<input checked="" type="checkbox"/>	MIK-45 If Macro, send remaining buffer and clear buffer	DESIGN AND IMPLEMENT SOFTW...	2	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-40 get raw input from GPIO	DESIGN AND IMPLEMENT SOFTW...	4	IN DEVELOPMENT			
<input checked="" type="checkbox"/>	MIK-27 Design PCB	DESIGN THE PHYSICAL KEYBOARD	5	IN DEVELOPMENT		M	
<input checked="" type="checkbox"/>	MIK-64 Print Keycaps	DESIGN THE PHYSICAL KEYBOARD	2	IN TESTING		A	
<input checked="" type="checkbox"/>	MIK-62 design keycaps	DESIGN THE PHYSICAL KEYBOARD	3	IN TESTING		A	
<input checked="" type="checkbox"/>	MIK-57 Unit Testing	INTEGRATE THE EMBEDDED HARD...	4	IN DEVELOPMENT			
<input checked="" type="checkbox"/>	MIK-31 Construct Final Circuitry	DESIGN THE PHYSICAL KEYBOARD	3	NOT STARTED			

Sprint 3 is dense, and it will focus on integrating much of the hardware and software. While tasks are not assigned, The members of the software sub-team will work on MIK-42, MIK-41, MIK-50, and MIK-52. The hardware sub-team will complete MIK-26, MIK-28, MIK-61, MIK-63, and MIK-65. The team will work together for integration and testing tasks. This holds true for sprints 4 and 5 as well.

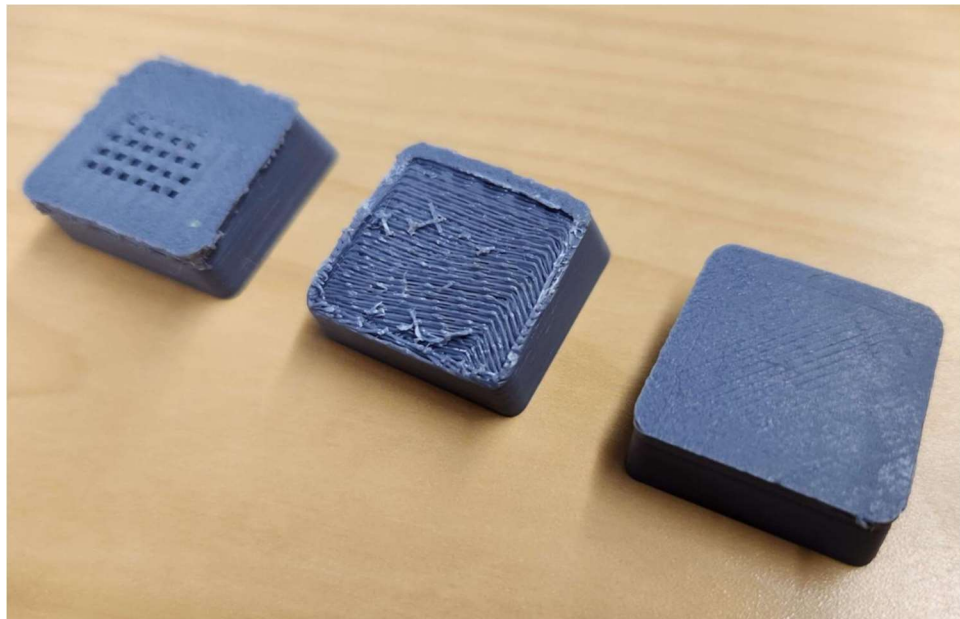
MIK Sprint 3 18 Oct – 1 Nov (10 issues)			28	0	0	Start sprint	...
<input checked="" type="checkbox"/>	MIK-42 Send key signals to encoder and key decision maker	DESIGN AND IMPLEMENT SOFTW...	1	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-41 Key Mapping	DESIGN AND IMPLEMENT SOFTW...	3	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-50 Encode provided keys to USB protocol and send signal	DESIGN AND IMPLEMENT SOFTW...	5	NOT STARTED		A	
<input checked="" type="checkbox"/>	MIK-26 Key press decoding Circuitry	DESIGN THE PHYSICAL KEYBOARD	3	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-28 Print PCB	DESIGN THE PHYSICAL KEYBOARD	2	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-61 design housing	DESIGN THE PHYSICAL KEYBOARD	4	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-70 LCD Integration with software	INTEGRATE THE EMBEDDED HARD...	2	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-63 Print housing	DESIGN THE PHYSICAL KEYBOARD	2	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-52 Encode suggestions to LCD signal	DESIGN AND IMPLEMENT SOFTW...	4	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-65 Assemble MIK	DESIGN THE PHYSICAL KEYBOARD	2	NOT STARTED			
MIK Sprint 4 1 Nov – 15 Nov (3 issues)			5	0	0	Start sprint	...
<input checked="" type="checkbox"/>	MIK-68 User interaction Test	TESTING & VALIDATION		NOT STARTED			
<input checked="" type="checkbox"/>	MIK-55 Check code for known vulnerabilities	INTEGRATE THE EMBEDDED HARD...	2	NOT STARTED			
<input checked="" type="checkbox"/>	MIK-56 Stress test prototype	INTEGRATE THE EMBEDDED HARD...	3	NOT STARTED			
MIK Sprint 5 15 Nov – 27 Nov (1 issue)			1	0	0	Start sprint	...
<input checked="" type="checkbox"/>	MIK-60 Prototype pricing vs. alternative final product materials	MARKETING	1	NOT STARTED			

Below the overall timeline for the project is shown again, which lists the due dates of deliverables for each team member based on the current Sprint the tasks belong to.

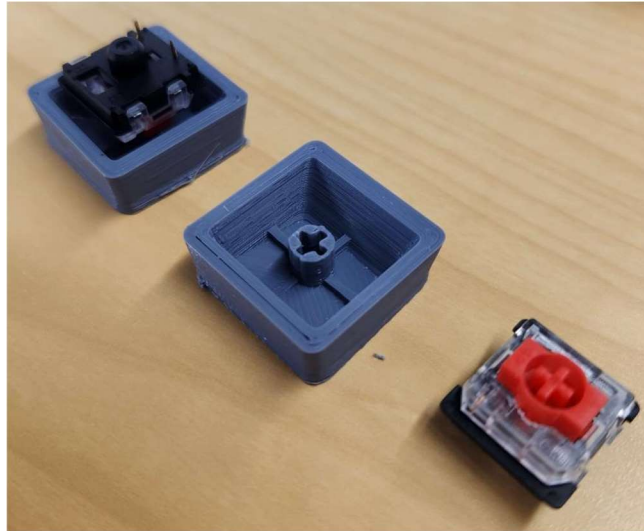
Sprint #	Dates	Status
Sprint 1	09/20/2022 - 10/04/2022	Complete
Sprint 2	10/04/2022 - 10/18/2022	In Progress
Sprint 3	10/18/2022 - 11/01/2022	Not Started
Sprint 4	11/01/2022 - 11/15/2022	Not Started
Sprint 5	11/15/2022 - 11/27/2022	Not Started
Final Reporting	11/27/2022 - 12/05/2022	Not Started

4 Preliminary results

3D Modeling for the keycaps has been rather successful through two iterations. The first iteration of the model included an inset key top to ensure that the user would be able to press on a key without sliding and hitting other keys. This needed supports which would be tedious to remove for eighty keys. This has become obsolete due to the case that the keys will be inset into. The model has changed to be a flat top which removes the need for supports when printing. The keycaps successfully interface with the switches for the keys.



The above image shows from left to right, the first version of the keycap with supports, the first version of the keycap with the supports removed, and the second version of the keycap.



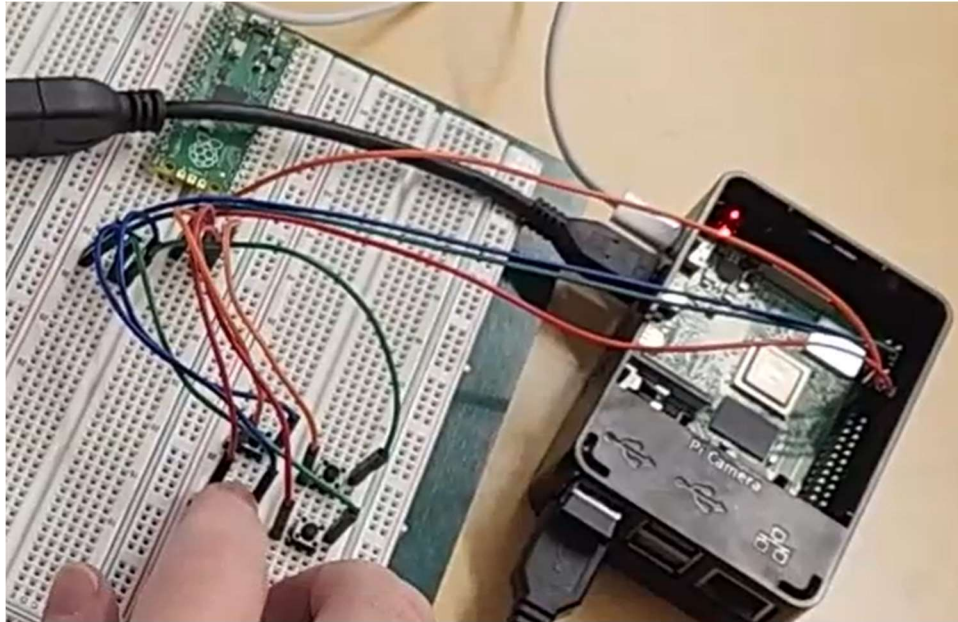
The above image shows from left to right, a keycap with a switch, the stem of the keycap, and the switch that interfaces with the stem of the keycap.

With the auto-complete algorithm complete, some preliminary tests of it have been performed to experimentally verify its utility and accuracy. Below are some sample test cases. As can generally be seen, the auto-complete algorithm finds candidates correctly and tends to prefer more common words (something verified by their placement in the dictionary file).

Auto-complete Testing

Input	Output
the	they their there
refe	reference references referred
introduce	introduced introduces NONE
int	into international internet
asdf	NONE NONE NONE

The team has constructed a hardware prototype utilizing the key scanning matrix that was successful. Each key was recognized as an individual key when using the circuitry from the diagram. This matrix was a simple 2x2 matrix that successfully interfaced with a Raspberry Pi 4 to show GPIO utilization. This was created using breadboard buttons rather than the switches for the keyboard due to the inability to place the switches on the breadboard.



[CSCE 483 Critical Design Review Demo](#)

