Accessibility Pi/O

Ji Chang, Carlos Armendariz, and Jorge Tamayo
Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—Our goal is to create an inexpensive, open-source keyboard and mouse set accessible to those with cerebral palsy. Most dedicated accessibility I/O tech currently on the market requires the user to assemble it themselves, and pre-built one-handed keyboards are much more expensive than standard keyboards.**

***Index Terms*—Accessibility, Cerebral Palsy, Keyboard, Mouse**

1 INTRODUCTION

Cerebral palsy (CP) is an umbrella term for a group of diseases that impair a person's motor functions. Spastic hemiplegia is one such form of CP, in which one side of the body is impacted much more severely than the other. Since modern computer use assumes the user has full mobility of both hands, those who can only reliably use one side of the body are forced to either use tech not designed for them, assemble their own, or purchase expensive specialized keyboards.

Our goal is to help those with Cerebral Palsy by designing inexpensive I/O hardware that is operable with one arm and one leg. Current solutions are either fairly expensive or require additional hardware to achieve full use, and many current accessibility input devices for computers are not designed to allow simultaneous mouse and keyboard use. We aim to create a design that will not be as broadly applicable to all motor disabilities, but will be a practical tool for those with spastic hemiplegia specifically. This would consist of a one-handed keyboard and a mouse operable by foot, both of which connect to a programmed Raspberry Pi that translates the inputs for a host computer.

To keep the design accessible to everyone, the project will be open source and use inexpensive, ubiquitous components. This allows people to easily and cheaply reproduce our results, or improve upon them if need be.

2 USE-CASE REQUIREMENTS

The keyboard should allow a baseline level of proficiency in typing. If users cannot type quickly on it, then it is not worth using. So, users of our keyboard should be able to type at about 30 WPM once acclimated to it. The keyboard should allow a baseline level of proficiency in typing; it is only worth using if someone can type quickly on it. Our goal is for users to type about 30 WPM once acclimated.

Also, to be usable for most tasks, a certain number of characters must be supported. A total of 74 characters is the baseline we will aim to meet, as this will allow most tasks to be performed on a computer. This includes 52 upper/lowercase letters, 10 digits, space, enter, and punctuation (‘.,”;()[]?!). Also, to be usable, other supporting keys need to be present. The non-character keys we plan on adding are: spacebar, left/right mouse click, backspace, enter, shift, ctrl, alt, a toggle key for hold key presses (i.e. if you press and hold a key, it will only read one press), and a toggle key to type digits.

We also aim to make a comfortable design. The keyboard should be easy to use without much strain. The design itself also needs to be durable; if it breaks within a week of use, then it is not a very useful design. So for our purposes, the switches used need to function after a large number of presses, and the design itself should be able to withstand some abuse.

The keyboard should also be usable to interface with a text to speech program, in addition to standard text entry. This is because people with cerebral palsy can sometimes have issues with the fine motor control required for speech. Also, the keyboard and mouse should be able to be used simultaneously. This is a key requirement for anyone who wants to play games or use certain CAD tools.

The quantitative metrics that we aim to test are how well it reduces the rate of common errors while typing. The most common type of error found in people with disabilities (at a rate of about 10%) was a long key press error [2]. This occurs when the user holds a key for longer than intended after pressing it, resulting in extra characters being input. Long key press errors were also found to be a non-issue for people without disabilities. Other common errors in people with disabilities were additional key press errors, missing key errors, dropping errors, and bounce errors. We aim to create a design that implements hardware and software features that can practically reduce the rate of these errors to match the occurrence in people who have full motor functionality.

The latency on the keyboard and mouse are also important. Too much delay can render an input device unusable, so any solution that introduces excessive latency should be avoided.

Our final requirement is that the production cost for the final devices should be below $200. This is to make it more accessible to people with disabilities; most purpose-built devices are significantly more expensive than this, and the cheaper variants usually require additional hardware and assembly for full functionality.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

Our system works first by subdividing the tasks into the keyboard, mouse, and Raspberry Pi.

The GPIO pins on the Raspberry Pi will be used to receive the key-press signals from the keyboard. Since a large number of keys are required, we will employ a technique called matrix scanning, which is commonly used for other keyboards [1]. Matrix scanning allows keys to be detected in a row and column format as if they were laid out in a grid. Usually, the physical layout is a grid, but this is not a requirement for matrix scanning. So, a small number of GPIO pins can be used to detect a large number of keys. For example, if eight pins were used, 16 keys could be distinguished from each other (four rows and four columns).

So, using matrix scanning, the Raspberry Pi will be able to determine when and which keys are pressed. Then, it will send the corresponding key over USB to the host computer. The mouse will use standard USB input as well.

Our keyboard also contains four keys that will change how the Pi reads the input: two shift keys, a num lock, and a hold press lock. The first three are meant to allow the same key to be mapped to multiple characters; shift turns lowercase into uppercase and num turns certain letters into numbers. The two shifts operate differently; one works like a standard keyboard and will be activated for as long as it is pressed, and the other works like a phone keyboard and will capitalize the first letter only. The num lock will work like a normal lock key, where pressing it on will turn all the letters into numbers until it is pressed off. Num lock will override shift.

The last key, the hold press lock, is meant to help with the long key press error (when the user presses a key longer than they mean to, resulting in a longer string of letters). While the hold press key is off, pressing and holding a key will only count for one press, and while the hold press key is on, holding the key will be treated as multiple presses. We give the user the option of turning the hold press on and off because there are some cases where repeated presses are desirable, such as playing a video game. Like num lock, pressing the hold press lock once will turn it on until pressed again.

4 DESIGN REQUIREMENTS

For ergonomic purposes, the individual keys on the keyboard can be increased in size relative to that of a standard keyboard. Increasing the size of and distance between keys is a common fix for people with dexterity issues. However, the key size is limited by how fast we expect people to be able to type; if the keys are too big or are spaced too far apart, then typing speed can suffer. A size that appears to be a good balance is a minimum of 20mm on both sides of the key cap, approximately 30% larger than usual. We will also be using linear key switches because they are easy to make and comfortable to press.

Due to the limited number of GPIO pins on a Raspberry Pi, and the total number of characters that we want to support (74 at minimum), the keyboard circuit is a matrix circuit. This determines which key is pressed by actively scanning the columns and seeing which row “reacts”.

The diodes are necessary to prevent unwanted current

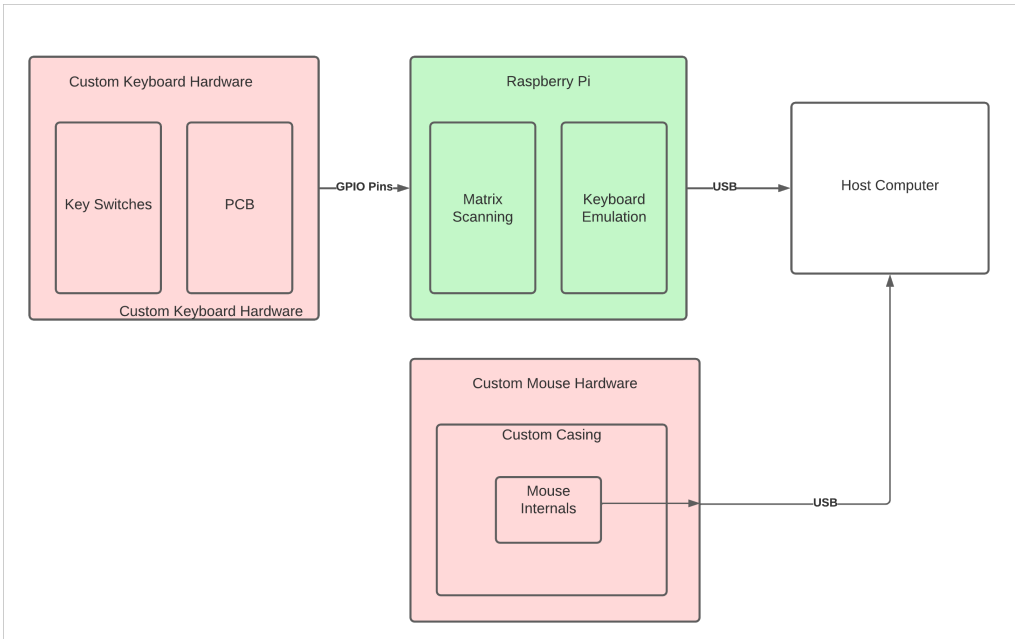

Figure 1: System block diagram.

paths. The forward voltage on the diodes used in the matrix circuit are required to have a forward voltage that is less than 1.8V. This is because the Raspberry Pi would interpret any voltage above this threshold as a high digital signal. The circuit works by pulling the row pins low when a key is pressed. This means that if the voltage across the diode is more than 1.8V, it would not read any change at all in the rows, even when keys are pressed. The total number of diodes that we will need will match the total number of key switches that are on the board.

Since the computer will see our device as a standard keyboard, the requirements for being able to interface with a text-to-speech program will come for free. Any keyboard is usable for these programs, so ours will be no different in this area.

In order to reduce the rate of common errors while using the keyboard, a combination of hardware and software solutions will be employed. There will likely need to be software that recognizes when a key is being held unintentionally in order to avoid long key press errors. Also, a removable keyguard will likely be required to prevent missed key errors.

To keep latency low, there are a limited number of actions we can perform in software between detecting a key press through matrix scanning and sending the key press to the computer. The exact limit here will require testing.

5 DESIGN TRADE STUDIES

5.1 Designing a Mouse Alternative

For our implementation of mouse functionality, we had many considerations to make. Most current solutions involve a mouse alternative such as a ball mouse or a roller mouse, or for the Microsoft adaptive controller, a large joystick to control mouse functionality. We went over many iterations of ideas but ultimately decided that these solutions were not in line with what we wanted to implement.

All of these solutions have the added downside that the control of a cursor would be very complicated and would require very difficult movements to use. We wanted someone to be able to preserve the same precision as a typical mouse. This compounded with our second criteria, which was that we wanted the mouse and the keyboard to be usable at the same time. It is a simple feature but one that is not typically seen.

We opted then to do a foot controlled mouse, which we felt was a good alternative since we maintain the finer controller of a typical mouse, without sacrificing the one hand use we planned to implement for the keyboard.

5.2 Designing a Keyboard for a One Handed User

The general layout of our design was going to be a one handed keyboard. We felt this was an easier and more practical decision than designing a keyboard for two handed use in which one hand's mobility is limited.

Designing a one handed keyboard from scratch is likely impractical, as one handed layouts already exist and have likely not only been established for much longer, but also been more thoroughly tested than any completely new layout we could come up with. Considering this, we decided to base our custom layout on a pre-existing design, but modifying certain aspects to make it more useful for our particular user and use-case.

By designing a keyboard that is meant to be used with one hand, we also felt we create the potential for someone to gain good typing speed with the device, despite having SH.

5.3 A Keyboard for CP and SH

With the general design of the one handed keyboard already established, we needed to find a way to attend to the particular difficulties of someone with CP. We went through sources of people with CP to try and gauge what the typical difficulties with CP, and more specifically SH, were. Among these issues were keys being too small or being placed too closely together. While these might seem like relatively minor issues, for someone with SH these issues are very real and are the most common cause for accidental error input.

These are the particular considerations that were kept in mind and should be addressed when designing the ergonomics of our device. These are the primary inspirations for our choice to use 20mm keycaps and including a keyguard in our design. Our feedback from Professor Carrington also confirmed the importance of these elements in our design.

We also decided on linear key switches as they are the easiest kind of key switch to press down on. Other key switches include tactile and click switches. We opted to go with linear as opposed to these because linear key switches have smooth travel as they are pushed down, while the others have a kind of push-back part way through the motion.

5.4 Possible Further Considerations

One thing we did not consider when making our design, but may be useful when considering economics in the future, are the parts of the body which we are not designing for. Our feedback from Professor Carrington gave us much insight into the kinds of details we might have overlooked.

For one, although our design is intended to be used for the side of the body that is high functioning, that does not necessarily mitigate the problems present from the other side. The less functional side of the body is prone to spastic behavior which can affect the comfort and ease of use of the side of the body that is high functioning through indirect means. We can resolve this issue likely through having a more considerate picture of the kind of person we are trying to create this device for and stresses the importance of finding a person with CP to consult.

6 SYSTEM IMPLEMENTATION

6.1 Keyboard PCB Specification

In Figure 2, the matrix scanning circuit is shown. To determine which keys are pressed, an individual column is set to low by the Raspberry Pi. Then the value on each row is read. If the row has a value of 0 (less than 1.8V across the diode), then the key in the column that is set low and that row is pressed.

This is the primary mechanism with which the keyboard operates, with each column we are not scanning set to high, and the one we are scanning set to low. Quickly scanning through each column is virtually undetectable to people and is a typical mechanism for any keyboard. This means that for n pins we get $(\frac{n}{2})^2$ possible inputs.

The diodes on the keyboard help to stop current from flowing in a direction that isn't desirable. While technically they are optional, most if not all standard keyboards include them because it makes them less prone to error.

The PCB is a fairly standard matrix scanning circuit as mentioned prior, with spacing between the switches allowing for our non-standard 20mm key caps. The outputs of the keyboard are then fed to the Pi by way of male header pins. The Pi also includes male header pins and we plan to have a female-female cable connecting the Pi to the Keyboard PCB.

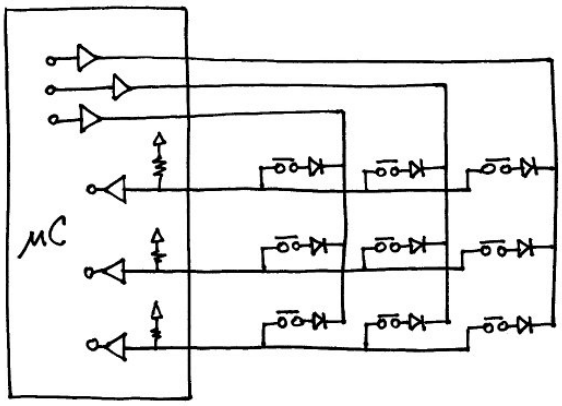

Figure 2: Matrix scanning circuit diagram[1].

6.2 Typing Behavior

The keyboard operates in a slightly non-standard way due to it being a one-handed keyboard, as well as having some particular features which makes typing for people with disabilities much easier. The typical combination keys (num, shift, fn, ctrl, etc.) are modified by our software to behave in a way that is atypical for a keyboard.

As an example: when the shift key is pressed, it will toggle on, if pressed again, it will toggle off. If the shift key is toggled and a letter is pressed, it will be typed as a capital letter and then shift will be automatically de-toggled.

This allows for more comfortable one handed use of a keyboard while also being a particular benefit to those with SH. The keyboard also includes keys to replace left and right click, which is also typical for one-handed keyboards. This feature complements our mouse alternative which only performs the cursor tracking function of a mouse.

6.3 Keyboard Ergonomics

The Ergonomics of our Keyboard is largely based on the typical complaints from people with SH. Although people with SH are capable of using both hands, the one that is affected is seldom used, and most are relegated to typing with one hand on a two handed keyboard.

We found that people with SH and CP tended to prefer keyboards with larger keycaps because it was much easier for them to type. A common error was also the accidental pressing of multiple keys, which people with SH often tolerate but dislike. The feature of a keyguard is not often seen in keyboards made for adults, and is something that people with CP often prefer. Our keyboard then, will have non-standard 20mm keycaps, as well as an additional keyguard which will be optional.

6.4 Mouse Implementation

The mouse will operate as a typical USB mouse but with less features. Since left and right click are relegated to the keyboard, the mouse only needs to perform cursor trying. We tried to come up with a way for the user to move the cursor while also being able to use the keyboard at the same time. As such, the mouse will be controlled with one foot.

The implementation of the mouse is fairly simple. We will purchase a mouse and replace the housing with our own custom design, made to be strapped to the underside of a flat shoe. While implementing an entirely custom mouse PCB was an option, it seemed impractical, especially since our mouse does not have any unique features apart from its unique ergonomic design

7 TEST, VERIFICATION & VALIDATION

When it comes to testing an input device there are some fairly common problems that arise. We hope to be attentive to these considerations and build a thorough testing plan for our device. We are limited by the availability of people with CP to test the device with, which could be a potential shortcoming in the future.

Regardless, we have tried to implement a survey based testing plan to try and gauge the quality of our device from comfort and ergonomics to ease of use and error prevention. Ideally, we will observe people with CP using both our keyboard, and their standard input device for a computer.

The verification testing in this context would be seeing if the keyboard and mouse combination is functional, and validation testing is checking if people with CP can use our keyboard to improve their input speed and/or accuracy.

7.1 Tests for Ease of Use/Ergonomics

In order to evaluate whether or not the ergonomic requirements of the design are met, we plan to provide a survey to those who have tried using our design. There will be two sections of text that we ask participants to type. One will have simple sentences, and another will have more complex sentences. The survey will ask the following questions:

1. Rate your usual typing ability from 1 to 10.
2. How comfortable would you say it was to type?
3. If you could/can only use one hand, how confident are you that you could get used to using this keyboard?
4. How much harder would you say the complex text was to type than the simpler text
5. Are there any mistakes you encountered while typing? (Example: Did you press a key you were not intending to?)
6. Were there any inputs in particular that you felt were too difficult to perform? For example, was it more difficult than necessary to press the comma key?
7. Do you have any suggestions for improving the layout?

Questions one through four are ranked on a scale of one to ten, and we aim to average a score of at least seven on all four of these. The remaining questions are meant to provide qualitative feedback on the practical use of our keyboard.

7.2 Tests for Determining Error Reduction

When we have users perform the typing task referenced in the previous testing section, we will also monitor the types of errors that they make. This, along with using external research that cites the most common types of errors (and their rate of occurrence), will allow us to determine how our design impacts error rates while typing.

7.3 Tests for Simultaneous Keyboard and Mouse Use

To test the practicality of this feature, we will also have participants play Minecraft (or some other software that they are familiar with that can utilize simultaneous mouse and keyboard use) before completing the survey.

7.4 Tests for Latency

Latency tests may require external tools to perform accurately. Measuring the latency of the software strictly within the Raspberry Pi can be done within the software itself. However, measuring key-press to on-screen character appearance may require other methods. The most accurate method is likely using a video to record both the keyboard and the screen. Then the time between both can be measured. However, the camera would need to be recording at a higher frame rate than normal for this to work. There can also be some testing by hand, where we try typing on the keyboard and see how the latency feels.

8 PROJECT MANAGEMENT

8.1 Schedule

Our plan for implementing our accessibility based input device begins mainly with the keyboard itself. In general, whatever hardware we make has a complementary piece of software which can be worked on in parallel. So the first step of implementing the software and hardware of the keyboard is the first step. What follows is a design phase, in which the practical aspects of our design have already been thought through and we need only to deal with the physical ergonomics of what we make.

At this stage we begin work on the mouse input device, which is less complex and requires only the design of a new housing for mouse internals. This is why it is part of the design phase and not the initial hardware and software implementation phase.

See figure 3 for a graphical representation of our planned schedule.

8.2 Team Member Responsibilities

There are mainly four different aspects to our division of labor, hardware, software, design work, and design implementation. For Hardware, Jorge and Ji are primarily responsible. The software has typically been Carlos' responsibility, but we are all capable of working on it. The design work is a shared collaboration between everyone and the actual implementation (3D printing, and construction) will also be a shared responsibility.

8.3 Bill of Materials and Budget

Our budget of six hundred dollars is more than enough to implement what we are planning to make. The highest cost item will likely be the custom PCB, which we plan to order from Oshpark. This board will likely cost a significant amount as it is quite large and there will likely be a minimum order number of at least three boards. The materials we will need are listed in table 1.

8.4 Risk Mitigation Plans

Our primary risk is implementing an input device which is unhelpful. We have mitigated this already by meeting with a specialist in accessibility devices and are making active plans to seek out and consult someone with the relevant target disability.

Table 1: Bill of materials

Description	Quantity	Cost Per Item	Total
Custom PCB	3	$50	$150
Diode - 1N4148	42(min)	n/a	$5
Mouse Internals	1	$20	$20
3D Printing	n/a	n/a	$50
			$14.00

The more logistical risks of a custom PCB not working, can be mostly mitigated by prototyping our keyboard circuit before committing to ordering the board. Other parts such as the linear key switches can have delays, but that risk does not seem very likely. All our components, even the Diodes are available through many sites, even amazon.

9 RELATED WORK

There are currently many related accessibility input devices on the market. Since we plan our use-case to be video games (Minecraft) it is important to note Microsoft's adaptive controller. Like many of the alternatives currently available, it is a generalized controller that can be adapted to multiple disabilities. Our design differs from this in that we are focusing on a very specific subgroup which could benefit from a more niche input device.

There are a number of pre-existing devices which are often used by people with CP but which are not tailored to them. Examples of these devices include ball mouses, roller mouses, and one handed keyboards. Our device has the benefits of a one handed keyboard but with particular considerations for limited mobility, and common user experience problems for people with spastic hemiplegia.

10 SUMMARY

Our accessibility input device is a one handed keyboard with particular considerations for people with SH. This includes larger key sizes (20mm) as well as keyguards which prevent unintentional input. We also include a mouse which can be controlled with one foot, allowing people to use the keyboard and mouse at the same time, a feature that is rare in the alternatives already available on the market.

Our challenges moving forward have to do with the particular challenges when designing for someone with SH. Despite the fact that our design focuses on the high functioning side of the body, the other side is prone to spastic behavior which can affect comfort and ease of use, even if our design is not intended to interact with that particular side.

It is important that we think of the device as a genuine improvement upon pre-existing alternatives and try to design something which is attentive to all the problems associated with CP and not just ignore the side of the body that is not in use.

Glossary of Acronyms

- CP - Cerebral Palsy
- RPi – Raspberry Pi
- SH - Spastic Hemiplegia

References

[1] *Input matrix scanning.* URL: `http://www.openmusiclabs.com/learning/digital/input-matrix-scanning/index.html`.

[2] Edmund F. LoPresti, Heidi Horstmann Koester, and Richard C. Simpson. "Measuring Keyboard Performance for People with Disabilities". In: 2006.

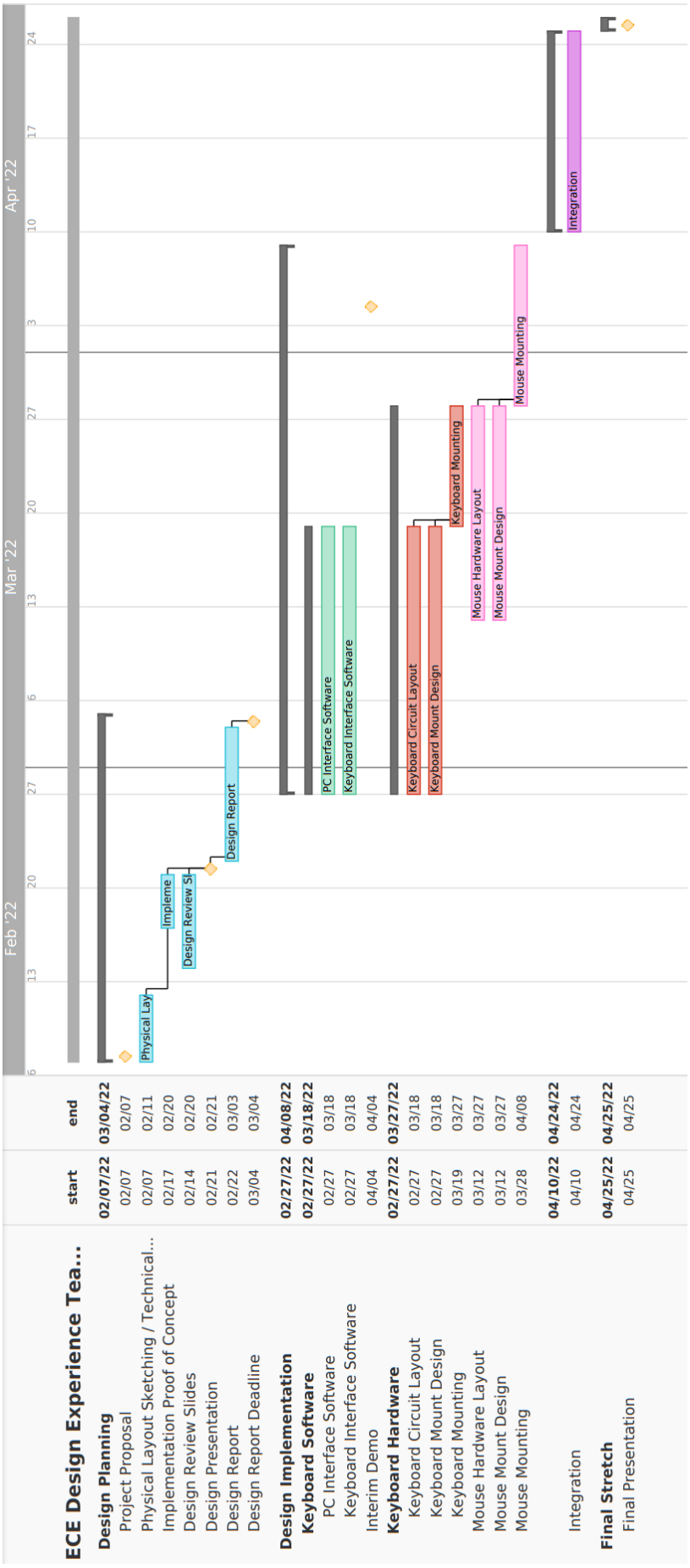

Figure 3: Gantt chart that details our planned schedule for this project.