User Manual

Accessible Switches

GNG 2101 [B]

Submitted by

B14: Accessible Switches

Ross McNamara, 300119731

Jake Brown, 300112518

Ethan Bowering 300116537

Daniel Holmes 300117647

Ethan Graves 300110377

December 3rd, 2020

University of Ottawa

Abstract

Accessibility switches, switches that allow certain people with disabilities to use computers, come in a standard 3.5 mm mono jack. Special software and hardware interfaces are required to use switches with a personal computer, creating unnecessary costs. Furthermore, Windows does not natively support switch scanning. In order to dramatically increase computer access for people with disabilities, the task of creating a USB accessibility switch and lightweight application to replicate switch scanning mouse was initiated. Customer needs, benchmarking and target specifications are developed to define and solve the problem at hand. Conceptual designs are created to form solutions, which later form prototypes. Extensive prototyping and testing is performed to produce a final product; the Press-Ability switch interface. The final product contains two main methods; the scrolling method and the bisection method. It consists of both hardware, modelled on Solidworks and 3D printed, and software, written in GO. The product's primary function is to allow people who suffer from disabilities or limitations to communicate effectively with added ease. Customization has been implemented for the product, so all users can have an optimal experience. Health and safety guidelines are listed and should be followed to ensure proper use of the product. An affordable product supporting 5 switches and high customizability will assist users with overcoming dexterity disabilities while using a computer.

Table of Contents

Abstract 1

Introduction 1

Conceptual Designs 2

- 2.1 Functional Decomposition 2
- 2.2 Designs 2
  - 2.2.1 Ethan Graves 2
  - 2.2.2 Ross McNamara 3
  - 2.2.3 Jacob Brown 4
  - 2.2.4 Ethan Bowering 6
  - 2.2.5 Daniel Holmes 7
- 2.3 Design Analysis 8

Prototype 1 9

- 3.1 Prototype Description 9
- 3.2 Visual Representation 10
- 3.3 Evaluation of Prototype 11

Prototype 2 11

- 4.1 Improvements 11
- 4.2 Prototype Description 12
- 4.3 Visual Representation 12
- 4.4 Testing and Evaluation 13

User Manual 15

- 5.1 Important Features and Customization 15
- 5.2 Functions and Capabilities 16
- 5.3 Making the Prototype 16
- 5.5 Health and Safety Guidelines 20

Conclusions and Recommendations for Future Work 21

Bibliography 22

APPENDICES 23

List of Figures

Figure 2.1.1: Functional Decomposition Chart 2
Figure 2.2.1.1: Cursor Scrolling 3
Figure 2.2.2.1: displays this screen for a row-column scan 3
Figure 2.2.2.2: represents the screen keyboard for this method. 4
Figure 2.2.2.3: Two Switch Scanning 4
Figure 2.2.3.1: Four Switch Interface 5
Figure 2.2.3.2: Bluetooth Version of Four Switch Interface 5
Figure 2.2.3.3: Predictive Speech 6
Figure 2.2.4.1: Arduino Nano 6
Figure 2.2.4.2: Breadboard and Schematic Views of Circuit 7
Figure 2.2.4.3: Macro Editor 7
Figure 2.3.1: End-Goal Concept 9
Figure 3.2.1: Prototype 1 10
Figure 4.3.1: Configuration Dashboard 12
Figure 4.3.2: Only 1 switch plugged in until we can solder to arduino nano (not shipped). 13
Figure 4.3.3: High waveform means unpressed, low means pressed. 13
Figure 5.3.1: Protective Case dimensions 18
Figure 5.3.2: Protective Case Lids 19
Figure 5.3.3: Protective Housing Rendering in Solidworks 19
Figure 5.3.4: Final Circuit **20**
Figure 5.3.4: Arduino Micro in Protective Housing 20

List of Tables

Table 2.3.1: Concept Analysis 8
Table 3.3.1: Evaluation of Product Metrics 11
Table 4.4.1: Metric Evaluation 13
Table 5.3.1: Protective Housing Dimensions 17
Table A.1: Customer Needs 23
Table A.2: PDM Matrix 23
Table A.3: Target Specifications 24
Table A.4: Bill of Materials and Parts 25
Table A.5: Github Repository for all the software files 25

List of Acronyms

Acronym	Definition
BOM	- Bill of Materials
CAD	- Computer Aided Design
OS	- Operating Systems
PDM	- Product Decision Matrix
FPS	- Frames Per Second
COM	- Communications

1 Introduction

As the world transitions into a digital era, an increasing number of people have resorted to using computers as a primary form of communication. However, using a computer can be a major challenge for individuals who suffer from medical conditions such as: cerebral palsy, multiple sclerosis, and Parkinson's, as well as for people who have poor hand-eye coordination. A solution that allows users with disabilities to operate a computer is a device containing switches, where the switches are buttons that act similarly to using a mouse. However, most computers only allow for one switch to be plugged in at any given time, and most people would need more than one switch. Therefore, a more effective solution would be a switch interface that could run multiple switches simultaneously.

Currently, the majority of market solutions that make computers more accessible for such individuals are generally expensive and not easily obtainable. Furthermore, the current switches on the market use 3.5mm mono-jacks that plug into a computer. However, those are incompatible and not recognized by the Windows operating system (OS).

When designing this product, certain criteria must be satisfied in order to meet the user requirements. Those criteria include: the product must be reliable, while being easy-to-use, affordable and customizable to individual users. After the third, and final, iteration of prototyping, the most important categories of our target specifications (A.3) were met and exceeded those of our competitors.

In the final prototype, we were able to manufacture the switch interface for $38, which is not only well below our $50 target, but also more than three times cheaper than most similar products currently available on the market. Moreover, our product is able to accommodate up to five switches, that can all be used simultaneously. We also made sure to maintain an open-source switch interface that allows for further individual user configurations.

2 Conceptual Designs

2.1 Functional Decomposition

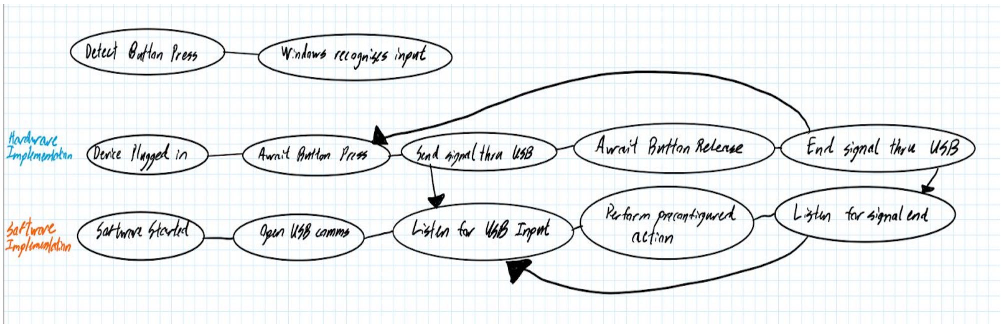


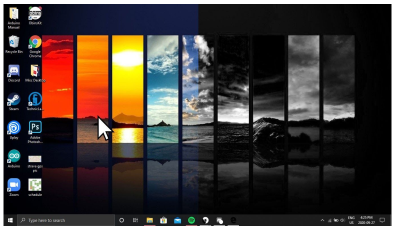
Figure 2.1.1: Functional Decomposition Chart

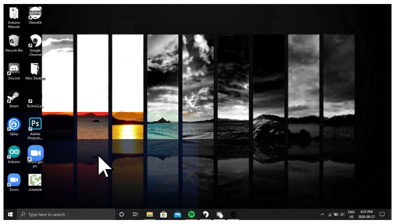
2.2 Designs

Three designs were created by each team member, visual representations follow most designs.

2.2.1 Ethan Graves

The first design is a software for controlling windows with a switch: The cursor starts at top left and scrolls right along the page then moves down a row and repeats. When a switch is pressed it activates a left or right click and resets the cursor to base position. The second design is hardwire switches 3.5mm mono jack to a usb to get a native accepted signal that you could make software for. The third design is a software for controlling windows with a switch: Use root bisection method on computer screen to find desired cursor location. Would require one switch to choose the side of the screen and another to select. In the image below the coloured area would be the selected side. Keep cutting down until the mouse is over the desired location.





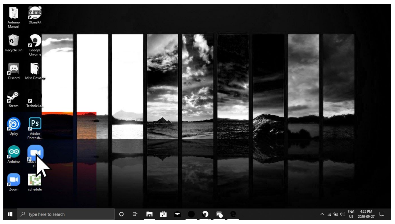


Figure 2.2.1.1: Cursor Scrolling

2.2.2 Ross McNamara

The first design is a row-column scan, it is a software for windows using 1 switch, and USB type A interface. First, a highlighted row starts at the top, and advances to the next after a fixed amount of time. When the row with the desired key is highlighted, the user hits the switch. Once a row is selected, each column is highlighted, which would be a single key on the row in this case. When the desired key becomes highlighted, the user hits the switch again selecting the key. This software can use word prediction, for example have the last row contain 3-5 words that start with the letter selected or a common word (ex The, I, you, etc). The speed of row/column passing can be increased or decreased to fit the users abilities. The switch can be placed on a desk, chair, as a foot pedal, and more. Figure __ displays this screen for a row-column scan

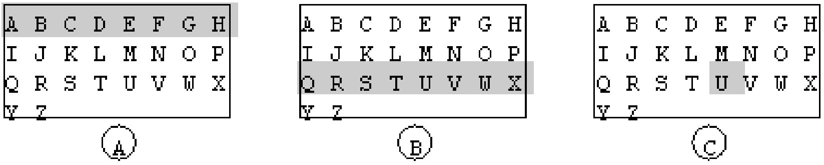


Figure 2.2.2.1: displays this screen for a row-column scan

The second design is keyboard halving switches, a software for windows, uses USB type A interface, 3 switches needed. This works like a bisection method, eliminating halves at each press. Pressing button one (B1) will eliminate half of the keyboard, or pressing B2 would eliminate the other half. Keep pressing B1 or B2 to continuously half the new segment of the keyboard, until the desired key is pressed. There can be a 3rd switch (B3) for backspace in case the wrong button is pressed (an undo button). If it takes too many clicks to get the desired key, users may lose patience. This is a very easy software to learn/use, and quick setup.

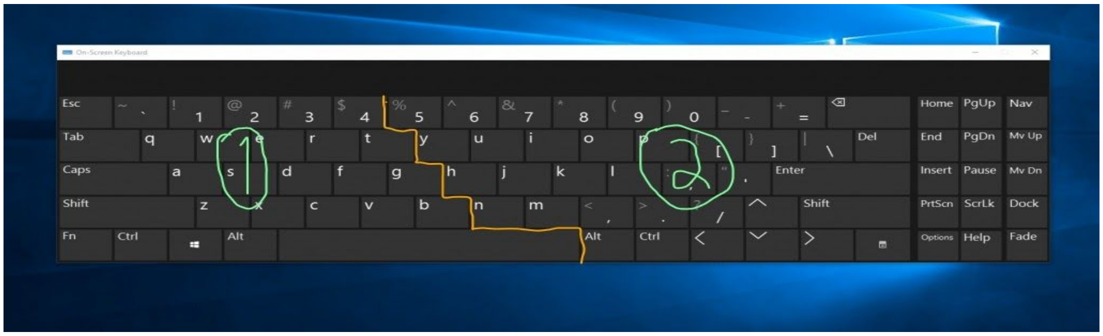


Figure 2.2.2.2: represents the screen keyboard for this method.

The third design is two=switch scanning, using USB type A interface, for windows. There are only 2 switches needed. Adjustable to perform left click, right click, tab, enter, backspace, space, arrow keys and more. The buttons are 3 inches in diameter. These switches are represented in Figure 2.2.2.3

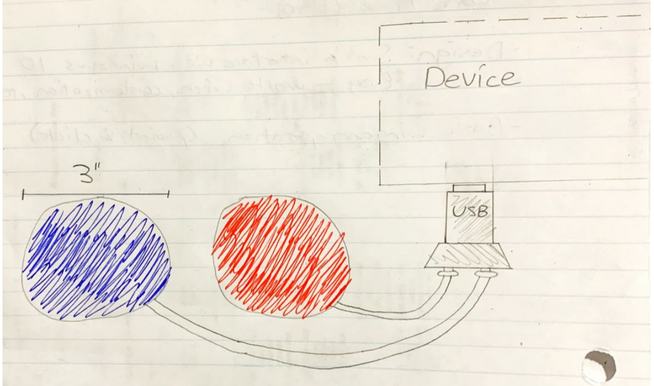


Figure 2.2.2.3: Two Switch Scanning

2.2.3 Jacob Brown

The first design develops a USB-A interface that works for Windows 10, capable of running two or more switches simultaneously. The button sequence would be less complicated using four switches compared to two or three. When using four switches, the user is not required to operate a keyboard or a mouse. The buttons can have an arrangement in any order depending on the needs of the user. Following is an example of an arrangement that could be implemented for the user as seen in figure #.

- The purple button would be the backspace/ back button and left-arrow
- The yellow button would be the tab button and right-arrow
- The red button from the left would be the left-click, enter, and up-arrow
- The blue button would be the right-click and down-arrow.

Each button controls ¼ of the screen. To move the cursor to the desired location, hold down the button(s) until the cursor is in the correct location, then double-click both buttons at the same time to click on the item.

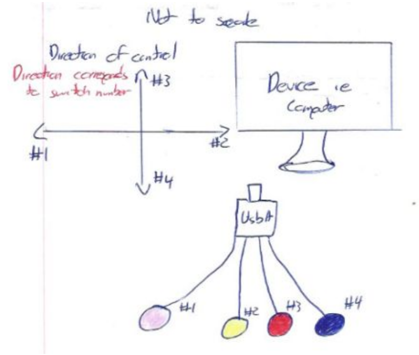


Figure 2.2.3.1: Four Switch Interface

The second design is a Bluetooth adapter interface for Windows would give users an alternative option to USB-A adapter interfaces. A customer designed Bluetooth adapter could accommodate up to four switches simultaneously. The switches could still be used as described in the idea above. A downside to the Bluetooth option would be that the adapter would need to be plugged into a wall outlet.

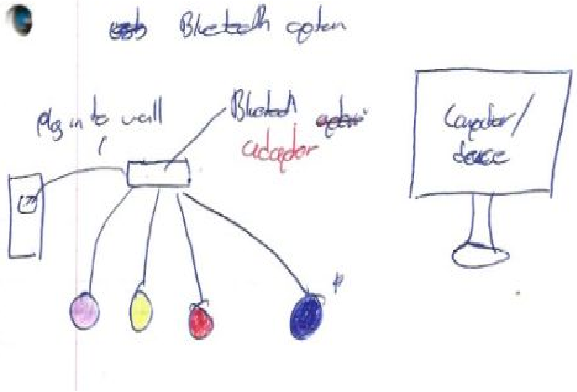
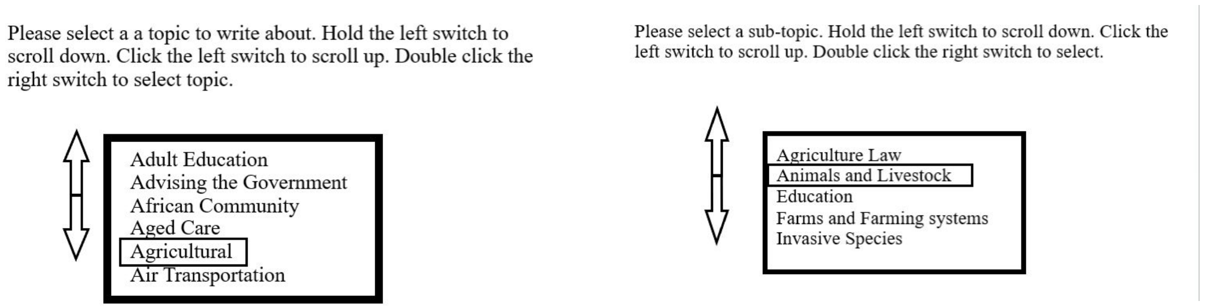


Figure 2.2.3.2: Bluetooth Version of Four Switch Interface

For the third design, in order to allow users with disabilities to write documents such as emails or word documents, a software would be created that would help users with such tasks. The USB-A interface would utilize two switches to scroll up and down and the other for left and right. The document writing software would be programmed to help the user communicate effectively and efficiently using predictive suggestions. The program would initially give the user a very general list of topics and then the user would scroll through the topics until they came across one that they wanted to write and discuss. Then, the program would give a more specific list of topics relating to the matter they just selected. The program will provide predictive text suggestions to the user to help write documents quickly and efficiently.


Please select a a topic to write about. Hold the left switch to scroll down. Click the left switch to scroll up. Double click the right switch to select topic.

Please select a sub-topic. Hold the left switch to scroll down. Click the left switch to scroll up. Double click the right switch to select.

Figure 2.2.3.3: Predictive Speech

2.2.4 Ethan Bowering

For the first design, use an Arduino Micro as the processor and I/O hub for the switch adapter. An Arduino Micro seems to be the best choice of arduino for this project as it is very small (48mm x 18mm), and is based off of the ATmega32U4 microcontroller. This microcontroller has onboard USB Communication support which would allow it to connect to a PC over USB and provide mouse and keyboard inputs to the PC, without the need of an external circuit to add this functionality (an Arduino Nano, and many other models would require this).

Figure 2.2.4.1: Arduino Nano

For the second design, a circuit can be built consisting of several 3.5mm mono jacks which can then each complete a circuit between the 5V line and one of the digital I/O pins on the Arduino Micro (which features 20 individual digital I/O pins). When the button is pressed, it completes a circuit which will connect the 5V line to the I/O pin, which the arduino can then read as “HIGH” using the “digitalRead” function, and then the software implementation will react accordingly to the button press. When the button is not being pressed, a pull-down resistor will connect the I/O pin to GND, which the “digitalRead” function can then read as “LOW”, and the software will act accordingly (In the figure below, the button switch would be replaced by a 3.5mm jack which then connects to an accessibility switch).

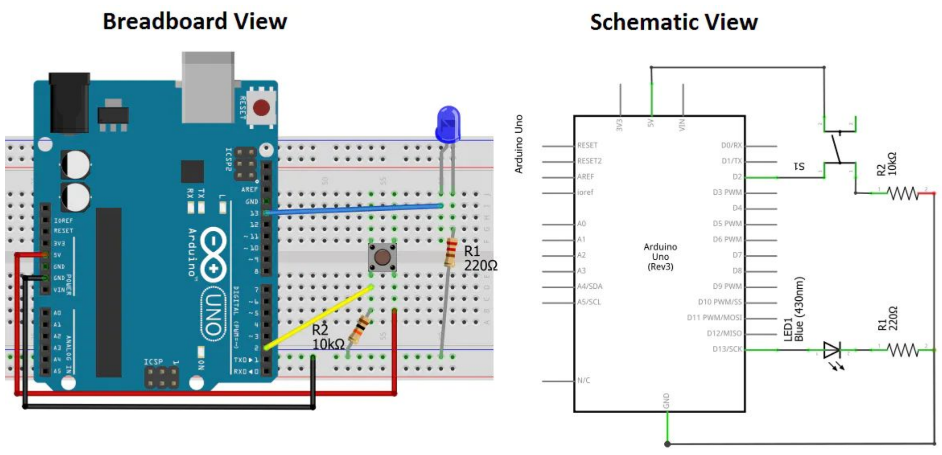


Figure 2.2.4.2: Breadboard and Schematic Views of Circuit

For the third design, in software, a macro editor could be created. This would allow the user to easily create their own macros (a pre-configured sequence of several keystrokes), which can then be bound to a certain button and activated when that button is pressed. Windows heavily uses keyboard shortcuts which can be difficult to physically use as they can require multiple hands, or very stretched out hands. For example, a button could be bound to a macro that simulates the “ctrl-alt-del” sequence of keystrokes to easily bring up the menu to open the task manager, or sign out of the computer. Another example is “alt-F4”, which is used to close a window. This macro could be set up and then bound to a certain button to easily close the current window.

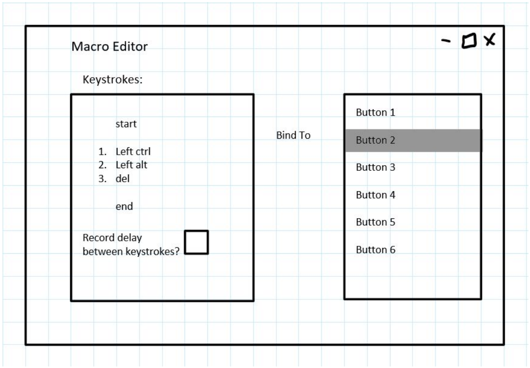


Figure 2.2.4.3: Macro Editor

2.2.5 Daniel Holmes

The first idea is creating a chrome extension is probably the best way to interact with more than one other piece of software using the input of our switches. Because manipulating applications running the browser would always be done through the HTML. Desktop applications are extremely difficult to interact with. Most desktop applications are probably impossible to interact with without writing a specific plugin for each application. An extra step required when trying to get input of the USBs could not be accessed directly from the browser. However running a node.js server locally that can connect via websocket to the chrome extension would allow the input from switches to be used by the chrome extension.

Another necessary part of the software is most likely a desktop application that would be needed to read input from a USB. It would also allow the function of the USBs to be configured, and perhaps interact with windows. The inputs from the USBs could be sent to a chrome extension from this application as well. It would likely be written in javascript using node.js / Electron for the user interface.

Thirdly, use a python or node.js program to interact with the mouse in windows. The program would read the input of switches and toggle between different operations such as click, drag and drop, double click. These 3 concepts can be combined together and be separate parts of the same program.

2.3 Design Analysis

Each member ranked their concepts by the following criteria: +1 for containing a target specification, 0 if N/A, and -1 for missing a specification.

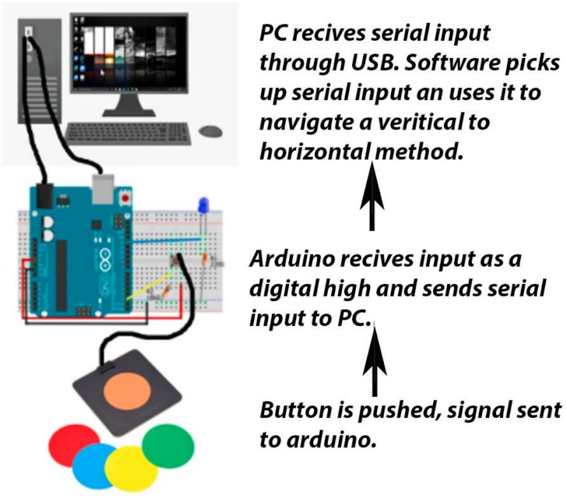
Table 2.3.1: Concept Analysis

Student	Idea #1	Idea #2	Idea #3
Ethan G	5	2	6
Ethan B	5	5	3
Daniel	4	3	5
Jake	4	3	3
Ross	5	4	2

From these concepts, our group has selected a few of the highest ranking to further develop. The solution will contain elements from Ethan G’s bisection method of finding the desired cursor idea, Ethan B’s Arduino Micro idea of transmitting switch signals, and Daniel’s idea for interacting with windows.

We have developed a group design concept, which is an integration of the concepts from the previous step. The final concept will use an arduino to collect a binary signal from a number of switches and transfer it as a digital signal to windows via USB A. Then an application on windows will interpret that signal and use a scanning method to operate the cursor.

A visual representation was created to show these concepts integrated.



PC recives serial input through USB. Software picks up serial input an uses it to navigate a veritical to horizontal method.

Arduino recives input as a digital high and sends serial input to PC.

Button is pushed, signal sent to arduino.

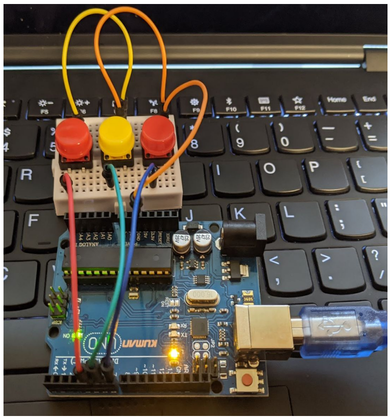
Figure 2.3.1: End-Goal Concept

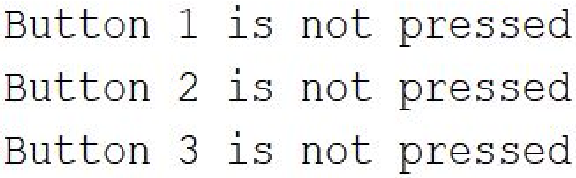
3 Prototype 1

3.1 Prototype Description

The first prototype was a focused one, on hardware. We created a prototype using components that a team member had in their house, which could also be found in the MakerStore. We had to test if the computer would recognize the button press. We used three buttons on the breadboard which were connected to the microcontroller, to replicate the switches that our client would use. So when the button(s) were pressed, the signal was sent to the arduino microcontroller, then sent to the computer screen. We did not use actual switches at this stage because they are far out of our budget for the first prototype. Every ¼ second, the computer should read the value of the button press, and indicate if button 1, 2, 3, or any combination was pressed. In our final design, we will increase the interval to a measure of nanoseconds for faster results.

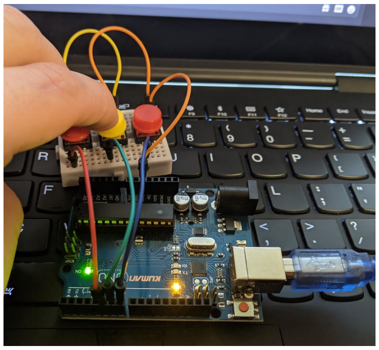
3.2 Visual Representation

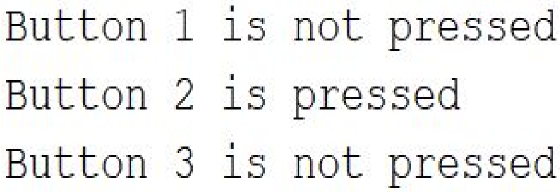




```
Button 1 is not pressed
Button 2 is not pressed
Button 3 is not pressed
```

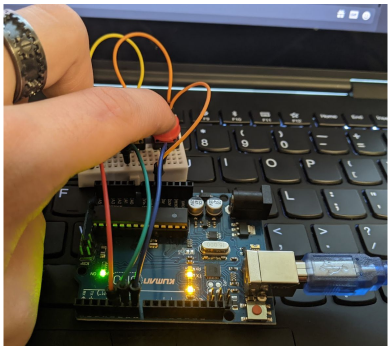
Arduino and Serial monitor 111

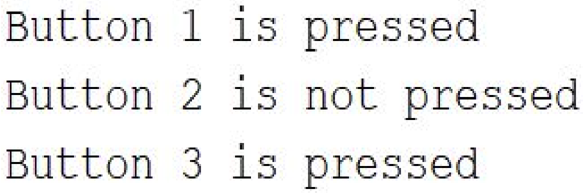




```
Button 1 is not pressed
Button 2 is pressed
Button 3 is not pressed
```

Arduino and Serial monitor 101





```
Button 1 is pressed
Button 2 is not pressed
Button 3 is pressed
```

Arduino and Serial monitor 010

Figure 3.2.1: Prototype 1

3.3 Evaluation of Prototype

We tested our first prototype to evaluate its performance. The first specification we noted was the number of switch inputs. We were expecting to use three to five switches, and our prototype used three, so we passed this criteria. The second specification we tested was the ability to connect to windows, as expected, our prototype connected to windows. We also noted the number of ports that our hardware used, and we met our criteria with one port. We tested if our software is configurable, since a signal was displayed on the computer screen it passes this criteria. We measured the cost of our prototype, which is tabulated in *Table 8.1: Bill of Parts and Materials*. We expected this prototype to cost under $50, and the actual cost came to $38, therefore passing this criteria. This $38 is if every component is bought from the MakerStore, or amazon. Our actual cost comes to $0 since a group member had all components at home.

Table 3.3.1: Evaluation of Product Metrics

Metric	Expected	Actual
# of switch input	=3 to 5	=3
Connects to windows	works	works
Cost	<$50	$38.5 ($0 for us)
# of ports	Up to 3	=1
Software is configurable	Yes	Yes

4 Prototype 2

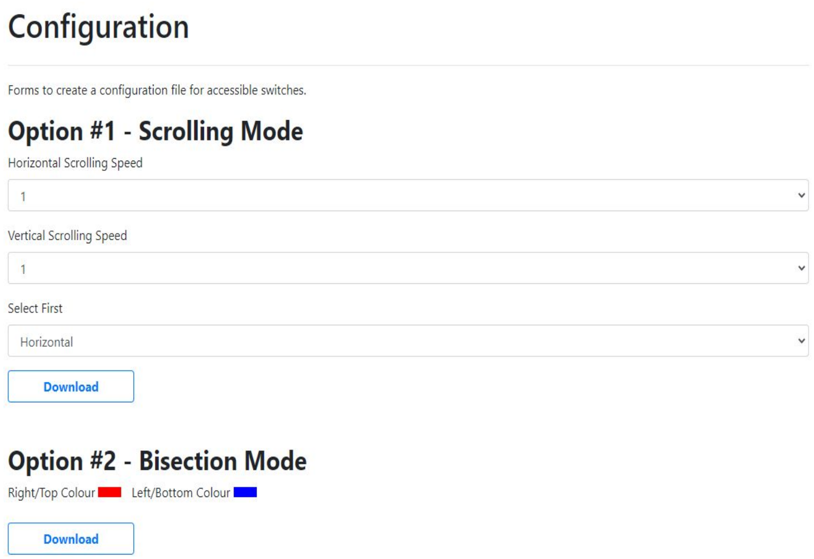
4.1 Improvements

Our group met with our client again, and based on our client's feedback, we needed to improve and modify our design slightly to improve our customer experience. Firstly, we need to add one extra mono jack to the existing four to accommodate five switches, allowing for more configurations simultaneously. A more straightforward configuration could be using those five switches, one for every direction and one for a clicker. As well, the software should contain an access bar to ease of changing configurations and settings. Furthermore, we need to continue to emphasize the smoothness of the software over speed. A slower and smoother interface is more straightforward for the customer to follow along and allows for a more comfortable user experience. To conclude, if we maintain keeping our product open source, we will allow the community to help develop low-cost switch interfaces to undercut expensive interfaces currently on the market.

4.2 Prototype Description

The second prototype is a comprehensive focused one including hardware and software. We used the components that were listed in BOM and purchased with the team budget. The first prototype hardware was improved by swapping out the three temporary buttons with 3.5mm mono jacks that will be used to plug existing switches into (figure 3.1.2). In terms of software the two main methods of controlling the screen: Bisection and Horizontal to Vertical scroll technique were successfully implemented as well as many customizable settings for each. Settings added included switching between the methods, scrolling speed and direction of scroll (Figure 3.1.1). The final element of this prototype was getting the software to read information. We achieved this by adding the library PiSerial (https://pypi.org/project/pyserial/) into Python and connecting the right COMM port.

4.3 Visual Representation



Configuration

Forms to create a configuration file for accessible switches.

Option #1 - Scrolling Mode

Horizontal Scrolling Speed: 1

Vertical Scrolling Speed: 1

Select First: Horizontal

Option #2 - Bisection Mode

Right/Top Colour 🟥 Left/Bottom Colour 🟦

Figure 4.3.1: Configuration Dashboard

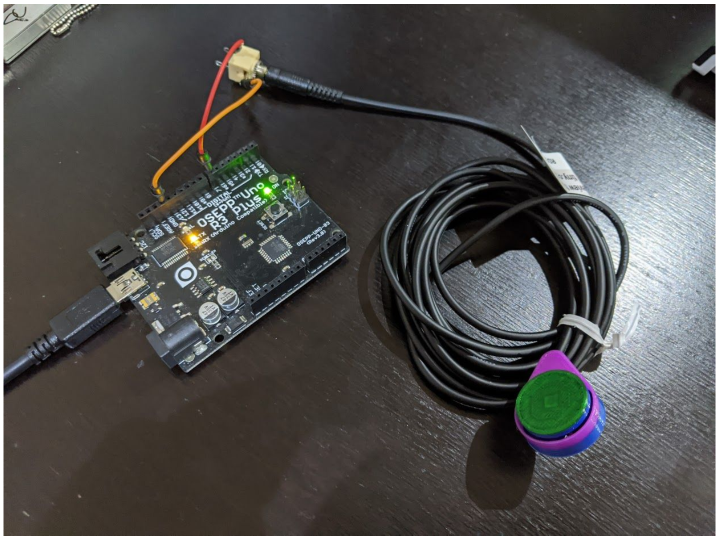


Figure 4.3.2: Only 1 switch plugged in until we can solder to arduino nano (not shipped).

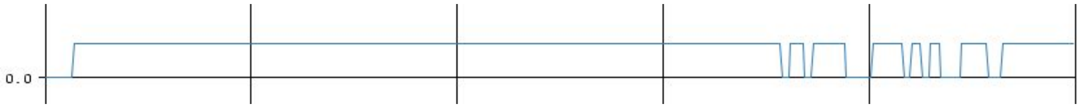


Figure 4.3.3: High waveform means unpressed, low means pressed.

4.4 Testing and Evaluation

Table 4.4.1: Metric Evaluation

Metric	Expected	Actual
# of switches	3.00 to 5.00	5.00
Cost of product ($)	<150.00	7.00
Storage required	< 1.00 GB	0.20 GB
Software is configurable	Yes	Yes

Uses USB A	Yes	Yes
Point	Yes	Yes
Click	Yes	No
Plugging switch into jack	Yes	Yes

The first metric we tested was the number of switches. We expected to use 3 to 5 switches in our target specifications, and we used 3 in our first prototype. For our second prototype, we expanded into 5 switches (which work), as wanted by the client. Next, we calculated the cost of our prototype. According to our target specifications, we have a budget of $150. Our first prototype cost $38.50, which ended being free because we already had the materials. Our second prototype is still well within budget, being $7.00, meeting our criteria. The third metric we measured was the required storage to run our software. We expected to need no more than 1 GB, and we met this metric by using 200 MB. This was checked by recording the storage that all of python uses for Daniel (2 GB), and subtracting 1.7 GB of non-project related files from this. We then subtracted another 0.1 GB for any error, regardless, it was still well under the 1 GB required. We did not test the storage required for our *first* prototype.

The next metric to test was if the software was configurable. Since our software runs and recognizes the button presses (Figure 4.1.3), we have met the specification. An easy target specification to check is if our prototype uses USB type A. Since this is the type of interface we have been testing with since the start, it is clear that we meet this criteria. The next two metrics we tested were very important. We needed to know if our prototype could successfully point and click. As seen in section 4.2, our software can scroll along the screen, proving we have the “point” metric met. We did not complete the clicking metric in time for our second prototype, therefore we did not meet this target specification yet. We also tested if the software would configure when the switch was plugged into the mono-jack, this was a success.

5 User Manual

5.1 Important Features and Customization

The final product contains two main methods; the scrolling method and the bisection method. Customization has been implemented for the product, so all users can have an optimal experience. For the switches, the minimum delay between button presses (input delay) can be customized to fit the user's reaction time. This delay is measured in milliseconds, with a default value of 50. If the user requires more time between button presses, they can increase this value. There is an option for the user to create shortcuts with the buttons, perhaps to instantly trigger a specific click. There are 2 shortcut slots available, and the most practical options would be left click for shortcut 1 and right click for shortcut 2. The access of these shortcuts is presented at the end of the important features section (6.1) in this report. The user can also pre-select the order of interval halving for the bisection method. For example the user could select horizontal halving as their first press, or vertical. Another required feature is that the COM port being used must be identified using the windows device manager.

For the scrolling method, the speed and directions can be adjusted to fit the user's capabilities. The speed (pixels per second) is determined by the FPS and the pixels per frame the by relation $\#of\ pixels/second = FPS \times Pixels/frame$. The user is able to change the FPS (the number of times the mouse will move per second) to 30, 60 or 144, with the default being 60 FPS. The pixels per frame represents the number of pixels the mouse will move in a second, and can be changed to any integer value (1,2,3 etc.). Therefore if the user increases the FPS and/or the pixels per frame, the scrolling speed will become faster, and vice versa. This is an important feature for this product since the range of users and user capabilities is so large. If needed, the user can also change the horizontal and/or vertical scrolling direction. This feature is more for the user's personal preference, some may want to look right to left instead of the traditional left to right. One last important feature for the scrolling method is the option to reset the scrolling position. This allows the user to restart the scrolling position from the current position of their mouse, if they made a mistake or just need a reset.

There exists basic controls for this product that the user's will learn quickly and with ease. For the scrolling method, pressing button 1 will stop the scrolling if the software is currently scrolling the screen, when it is not scrolling, the user can press button 1 to restart the scrolling. Button 2 is set to activate shortcut 1 (which can be a left click), and pressing button 3 will activate shortcut 2 (which can be right click). Button 5 allows the user to toggle the type of click they want for button 4. The process may seem confusing typed out, however the user will pick up on these features after little experience.

For the bisection method, the first click of button 4 will cycle the orientation, the second click activates "ready to click", and the third click controls the type of click. Pressing button 5 will toggle the type of click. If not ready to click, pressing button 1 will select the left side of the screen (or the top depending on the user's customization). If ready to click, the selection is restarted. Pressing button 2 will select the right side of the screen (or the bottom) if on not ready to click, or will activate the shortcut 1 if ready to click. Button 3 is optional for the bisection method, if on not ready to click, pressing button 3 will do nothing, and if on ready to click shortcut 2 will be activated.

5.2 Functions and Capabilities

Our switch interface's primary function is to allow people who suffer from disabilities or limitations to communicate effectively on a computer, with added ease. Commonly, these disabilities and limitations usually affect hand-eye dexterity and discourage users from using technology.

The second function is the implementation of a configurable scrolling method for the screen. The different user needs can be configured to increase their ability to use the computer effectively. The switch interface can scroll horizontal, vertical or use the bisection methods to narrow down the screen section that the user would like to select. The user can also determine the speed and colour of the scrolling method to help them.

5.3 Making the Prototype

The final product consists of both hardware and software components. To create the case for the switches, a 3D printer is needed, the one used in our prototype is an Ender 3 Pro. The box was printed using the environment friendly polylactic acid (PLA plastic); a thermoplastic material capable of being melted and reshaped. The box was modelled in solidworks, a 3D CAD software that is free for engineering students at the University of Ottawa. The box was initially measured with digital calipers in Solidworks. The dimensions for the outside box, lid outside, and lid inside are presented in Table 5.1.

Table 5.3.1: Protective Housing Dimensions

Dimensions			
Measurement	**Outside Box**	**Lid Outside**	**Lid Inside** (indented on all sides by 1 mm)
Length	48 mm	48 mm	46mm
Width	36 mm	36 mm	34 mm
Height	29 mm	1 mm	2 mm
Wall Thickness	1 mm		
Base Thickness	5 mm		
USB Cut out Feature			
Measurements	For USB cut out	(The fifth circle is the side closest to the USB side)	
Length From	6.05 mm	(distance from the side with the circle cut outs)	
Height	6 mm	(Height from bottom)	
Length	7.6 mm	(Width of cut out)	
Height	3.8 mm	(Height of cut out)	
Depth	1 mm	(Cut out is depth of the wall)	
Circle Cut Out Features			
Diameter	6 mm	(each of the five circles)	
Height	12.46 mm	(height of each circle from top)	
Distance C - C	9.05 mm	(distance between centers of two circles)	
Distance C - E	6.28 mm	(distance from centers of outside circles to the edge of the face (circle 5 is closest to USB side))	

Arduino Micro Support inside Housing		
Length	38.5 mm	(Starting from the USB End)
Width	17.7 mm	(Starting from Circle cut out side)
Height	-3 mm	(Into the base thickness)

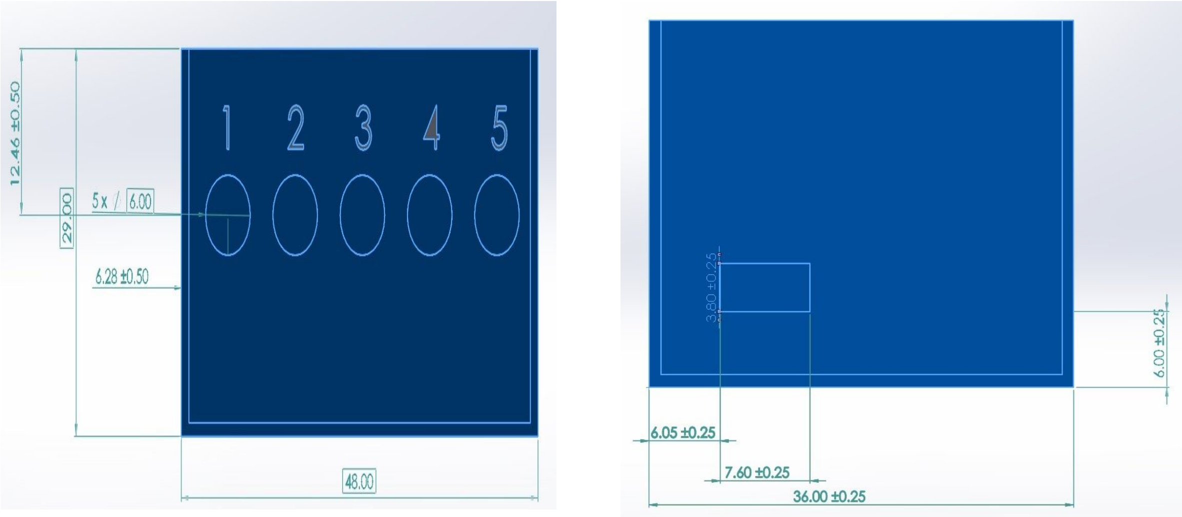
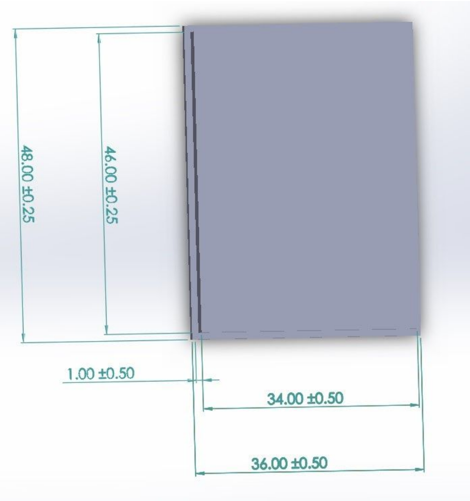


Figure 5.3.1: Protective Case dimensions



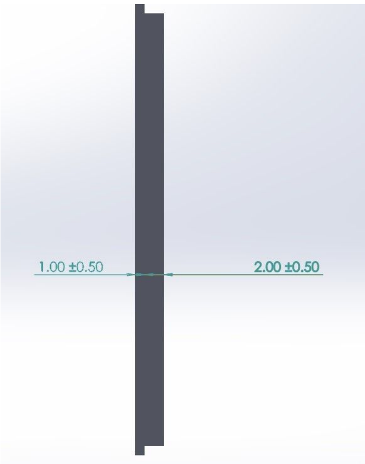


Figure 5.3.2: Protective Case Lids

After the protective housing and lid were designed in Solidworks, the files were printed in PLA [1] by an Ender 3 Pro, 3D printer.

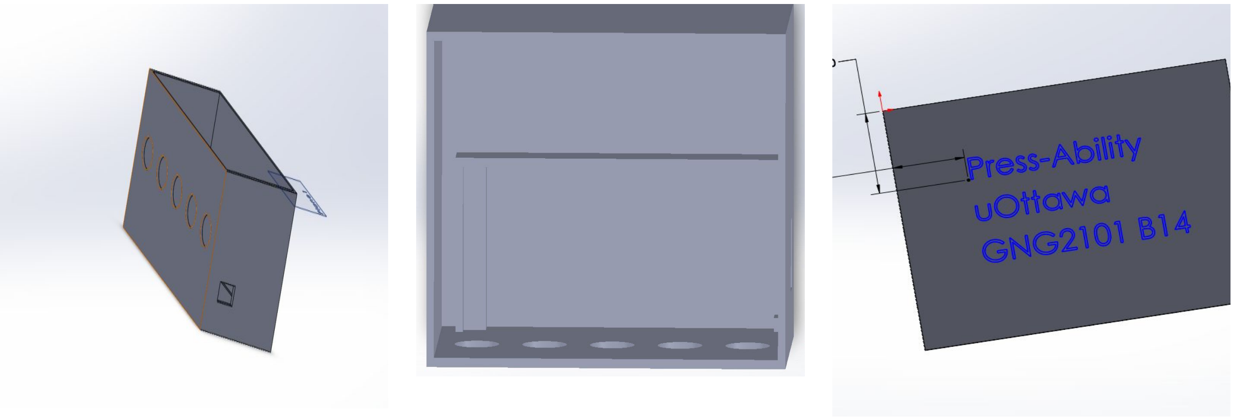


Figure 5.3.3: Protective Housing Rendering in Solidworks

Please refer to the github repository for all the software files (Table A.5).

For simplifying the circuit, we made a schematic in TinkerCad to show how the mono-jacks are connected to the Arduino Nano microcontroller. Each mono-jack has two pins that the wires are connected to, one for the ground and one for the voltage. The wires are soldered to complete the connections between the microcontroller and the mono-jacks on the actual circuit. It is important to note that the wiring and pinouts are the same.

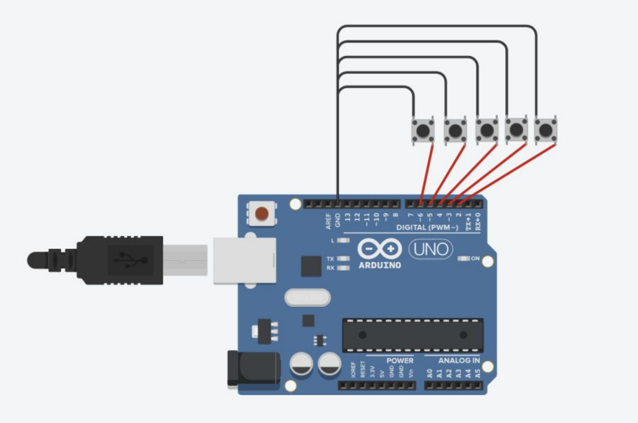
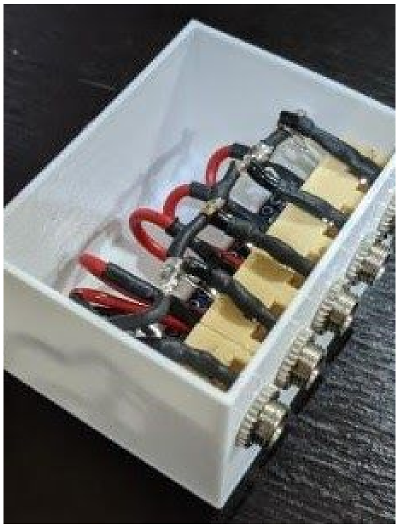


Figure 5.3.4: Final Circuit



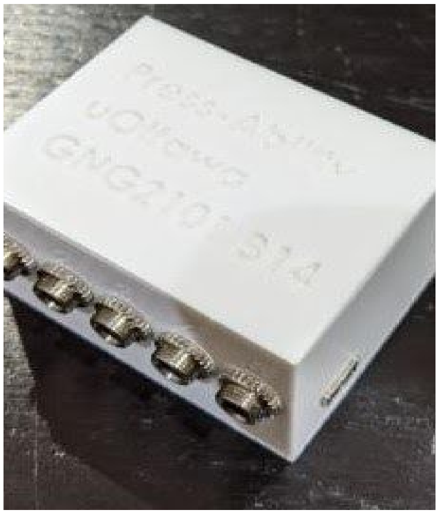


Figure 5.3.4: Arduino Micro in Protective Housing

5.5 Health and Safety Guidelines

The following health and safety guidelines should be followed to ensure proper use of the product:

- Do not expose to extreme temperatures, as it could damage the Arduino micro and all the electrical components inside the protective housing.
- Do not use the switch interface if the USB cable is frayed (wire exposed).
- If the protective housing is cracked or broken, refrain from further damaging the protective case as it will void the warranty. Immediately return switch interface to Press-Ability for a certified new replacement.
- Do not use excessive force when plugging in the USB cables or 3.5 mm mono jacks into the switch interface.
- Do not attempt to change or replace the protective housing components, as it will void all warranties.
- Refrain from using if the switch interface is undergoing an update.
- To prevent being shocked, plug the switches into the switch interface before plugging the USB cable from the computer into the interface.

- When handling the switch interface, ensure hands are dry to prevent electric shock.

6 Conclusions and Recommendations for Future Work

We learned many difficulties to overcome throughout this project when designing a product, especially during a pandemic. It is a challenge to have everyone access the hardware and test out the product's functionalities as a whole, mainly because many of the groups reside outside the Ottawa area and cannot commute and work in person with each other.

However, our product's capabilities can accommodate five switches synchronously, exceeding the four switches maximum that most competitor's products can handle. Using five switch configurations; community contributors can help develop more individual user configurations as our work is open-sourced.

The cost-savings will help people afford accessible products that assist with overcoming dexterity disabilities while using a computer. The cheaper we can make the product, the more people with dexterity disabilities or limitations can afford to use a computer with assistance.

The recommendations that we have to build off our final prototype involves much programming. Programmers should be familiar with coding in Python and other languages like Java and programming Arduinos. The next function that should be implemented is programming the predictive speech function. The user would then be able to write documents, emails, instant messaging or even programming. This implemented function would allow people with disabilities or limitations of dexterity in the hands to work in ever-growing technology-related fields such as business or engineering. For the protective housing, we figured out that a friction fit lid is still relatively loose. A recommendation would be to redesign the lid to be spring-loaded so that it will stay in place. The other function that could be added is integrating more button combinations of controls such as double-clicking, hold and drag. To conclude, expanding the switch interface to work with other OS like Linux or Apple's IOS would help people who suffer from dexterity type disabilities use our product.

7 Bibliography

[1] D. K. C. says: J. S. Says: E. H. says: and C. says: “What is PLA? Polylactic acid properties, uses, & melting point,” *3D Insider*, 10-Nov-2017. [Online]. Available: https://3dinsider.com/what-is-pla/.

[2] MakerStore, “Electronics, Materials, and Merch,” *MakerStore*. [Online]. Available: https://makerstore.ca/shop?olsPage=products%2Farduino-uno-r3.

[3] “SJ1-43502PM CUI Devices: Connectors, Interconnects,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/cui-devices/SJ1-43502PM/5130707?utm_adgroup=Barrel+-+Audio+Connectors.

[4] MakerStore, “Electronics, Materials, and Merch,” *MakerStore*. [Online]. Available: https://makerstore.ca/shop?keywords=Arduino%20Micro&olsPage=products%2Fbreadboard&page=3.

[5] MakerStore, “Electronics, Materials, and Merch,” *MakerStore*. [Online]. Available: https://makerstore.ca/shop?olsPage=products%2Fjumper-cables-per-10.

APPENDICES

Table A.1: Customer Needs

#	Need	Importance (1-5) 1 = Least 5 = Most
1	Product is intuitive	5
2	Product is customizable to the need of the individual	5
3	Product is reliable/won’t break	5
4	Product is inexpensive	4
5	The software can perform basic windows operation (point and click)	4
6	Product has an easy configuration for support/ family members to set up	4
7	The product uses USB-A (rather than USB-C)	3
8	The product can perform the text to speech	3
9	The product uses Bluetooth instead of traditional USBs or mono-Jacks	1

Table A.2: PDM Matrix

Metric #	Need #s	Metric	Imp (1-5) 1 = Least, 5 = Most	Units
1	1	Time for a new user to get accustomed to the software	3	Hours (h)
2	1,6	Steps to set up the software	3	# of steps
3	2,6,7	Number of switches/ ports that can be used simultaneously	3	#of
4	2,5,6	Software is configurable	4	# of settings
5	3	Required storage	2	GB
6	3	Ram	2	GB
7	3	CPU	2	GHz

8	4	Cost of product	5	USD
9	5	Average time to click on the right part of the screen	3	s_{avg}
10	8	Preform text to speech	3	Y/N
11	9	Bluetooth compatibility	1	Y/N

Table A.3: Target Specifications

#	Metric	Units	Marginal	Ideal	Reasoning
1	Time for a new user to get accustomed to the software	h	< 8	1	If the user is unable to learn how to use the product in an hour then it's probably too hard to use. For more than 8 hours they will give up.
2	Steps to set up the software	# of steps	< 10	3	There shouldn't be an excessive number of steps.
3	Number of switches/ ports that can be used simultaneously	#of	> 2	4	The average user will use 3 switches
4	Software is configurable	# of settings	< 20	10	We shouldn't need more than 20 settings. 10 Settings should allow for enough configuration
5	Required storage	GB	< 3	1	3 GB would be a lot. 1 GB would probably be more than enough
6	Ram required	GB	< 1	0.3	Using 1 GB of RAM on one program is a lot for most computers. 300 MB is not too bad
7	CPU required	GHz	< 0.3	0.1	For a 3GHz CPU with 2-4 cores, 0.3-0.1 GHZ is reasonable
8	Cost of product	USD	<150	0	Our closest software comparable is 150$. Our software would be free so we can try to stay as close to $0 as possible

9	Average time to click on the right part of the screen	s_{avg}	< 5	2	These would mean the solution is fast and easy to use
10	Perform text to speech	Y/N		Y	Optional ideally our software can perform the text to speech
11	Bluetooth compatibility	Y/N		Y	Optional ideally the switches could use Bluetooth

The following table is the Bill of Materials of the components to build our final prototype. Each component can be found at the University of Ottawa's MakerStore.

Table A.4: Bill of Materials and Parts

Item #	Part Name	Description	Quantity	Unit Cost	Extended Cost
1	Arduino Micro [2]	Contains programmable input/ output peripherals	1.00	$17.00	$17.00
2	3.5 mm mono jack [3]	Connects microcontroller to computer	1.00-3.00	$11.00	$11.00
3	Breadboard [4]	Solderless device for electronics testing	1.00	$10.00	$10.00
4	Wires [5]	Connects breadboard to the microcontroller	10.00	$0.05	$0.50
					Total:$38.50

Table A.5: Github Repository for all the software files

https://github.com/danielholmes839/GNG2101-Accessible-Switches