Northeastern University
Electrical & Computer Engineering Department
ECE Capstone 2 - Fall 2017

S.A.V.I.

(**S**martphone **A**ssistant for the **V**isually **I**mpaired)

Faculty Advisor

Professor Charles DiMarzio

Group Members

Bruno Costa, David Frymoyer, Matthew O'Brien, Frederik Petersen, Maher Shaikh, Rohan Verma

Table of Contents

Table of Figures

Introduction

Abstract

The SAVI team has designed a device which provides a tactile interface to assist the visually impaired in interacting with their smartphone. Current assistive braille devices function either as expensive full operating systems or as simple text readers. SAVI is the first tangible braille writer that allows you to navigate and interact with your Android smartphone. Our team has sought to create a modular device that is affordable, private, and simple to use. SAVI has two primary components: the android application and the device hardware. As SAVI is a proof of concept, the user can currently only interact with phone's messaging app. SAVI's modular system encourages future developments in communicating with a variety of IoT technologies and an expanded collection of native Android applications.

SAVI houses an eight-character braille display, a braille keyboard, six navigation buttons, a microcontroller, a printed circuit board, and four lithium batteries. SAVI has three main functions: set-up, read, and write. The user must download the SAVI app and establish a bluetooth connection with the external device. A voice assistant is provided. Once the initial connection is made, all future bluetooth pairings are automated. The user can read and write text messages while seamlessly navigating with the braille display. When a message is typed, SAVI writes the message on the braille display so the user can verify and spellcheck the text.

When a message is written on SAVI's display, the message data is converted to braille and is then passed either from the bluetooth module or the keyboard onto an Arduino Mega. The microcontroller sends a signal to the driver circuit, which consists of 6 demultiplexers, each one responsible for signaling 8 motors. The motors are controlled by a set of three H-bridges. Each DC motor has a resin-printed fin attached to its shaft. This fin is responsible for refreshing the braille display by pushing up a corresponding pin to form a single dot of the braille character.

The SAVI device was rigorously tested through a variety of quality assurance strategies. Software unit tests were written for a wide variety of failure scenarios. The full software stack was tested with a virtual version of the device with a Bluetooth connection. Lastly, the device was end-to-end tested through the course of different designs. Ultimately the SAVI team was able to deliver an affordable, personal, and simple SAVI product.

Background & Context

Today we live in a world dominated by tech. Take a walk through Boston and you might find a homeless person with a smartphone or an average joe with his smartwatch, smartphone, laptop and tablet, that is of course if you take the time to look up from your own screen to see it. Tech is everywhere and it will only keep on coming and keep on spreading, and most people will continue to take advantage of it each day. People wake up when the alarm on their smartphone goes off, they look at updates on social media, follow the news, chat with friends, and take pictures. It enriches their lives and allows them get to know people all over the world without ever having to meet them. What a wonderful world to be part of.

Now, we want you to imagine that you are not part of this world, for reasons outside of your control… Imagine that this world exists, that you know people who have access to this world. That you know that mostly everyone around you can use it, but not you. You might be able to use some of it, but ultimately you know and feel that this massive wave of tech that is supposed to bring you closer to everyone, is instead pushing you further away. Imagine being blind in the age of touchscreens.

For someone that is visually impaired, using a smartphone can be an impossible task. It has no texture, and the majority of tasks it can perform requires a user that is able to see the screen. Most existing solutions involve text-to-speech which provides some accessibility in navigating a smartphone without having to see the screen, but even then the voice is robotic, slow and cumbersome to use, taking away many advantages that a smartphone provides to those able to see. It is a tool which has been optimized for those with sight. Later, as an afterthought, minimal assistance was built into the device to assist the visually impaired. This is a massive issue with almost all technology. It relies heavily on sight and it seems that now with touch screens and virtual reality, we are moving further and further away from devices that are user friendly to the visually impaired. We do not believe that the solution to this problem will be to create special devices for every type of technology out there, a braille laptop, a braille phone, a braille watch, etc. Rather we believe in creating one universal device to assist the visually impaired with all sorts of modern technology and software applications.

Enter S.A.V.I

Problem Formulation

The goal of S.A.V.I is to make everyday technology easily available to the visually impaired.

In the nineties and early two-thousands, the visually impaired were fortunate enough to have a keyboard to write on, where the buttons were distinct and designed to be manipulated without looking. Today, with the invention of the touchscreen (which has made science-fiction into reality) the final life line for the visually impaired has been all but removed from the world of high-tech. Now, as an afterthought, voice-over has been implemented to assist the visually impaired with the many challenges they face with technology, but still the experience they have is nothing like that of a visually-abled. Everything is read in a slow robotic voice, only some applications are accessible, listening and writing is slow, navigation is slow, and ultimately you are left feeling like an afterthought.

Current and Competing Products

VoiceOver for iOS [1]

Although we primarily focus on VoiceOver for iOS here, we will note that certain third party apps do exist for Android, which perform some of the same features as VoiceOver does for iOS.

VoiceOver is Apple's native tool for the visually impaired. When it is activated the control of the iPhone is altered to adjust for use by a visually impaired person. The user is able to navigate the phony by using Siri or by sliding a finger across the screen. Each time one slides ones finger, the iPhone reads out the title of an app. Swipe again and it will read out the title of the next app. Text, and mostly everything else functions in pretty much the same way. You swipe and get a verbal response. This is the cheapest way to interface with an iPhone, as it does not require any new hardware or software to be purchased separately. It's downfall is that it is slow and not very private. Having to listen to each option is much slower than how we visually scan our screens to find the app we want. Voice is not private, and makes interacting with your phone, more uncomfortable in public spaces.

HumanWare Brailliant (IOS) [2]

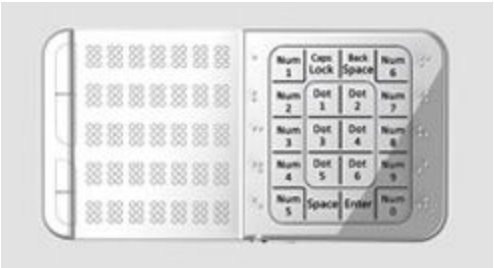
Here is probably our closest competitor, the HumanWare Brailliant device. This device is designed to help the visually impaired use their iOS devices more easily and in private. It is basically a braille keyboard that is wirelessly, or serially connected to an iOS device. From it you

can navigate your phone, read text in braille, read and write messages, etc. This device costs $2595, which is more than three times as much as the phone itself. This high price, makes it inaccessible to visually impaired persons all over the world, making it mostly only accessible to the select countries in the west, which have welfare strong enough to pay for them. As this device also doesn't connect to android devices and its software is not open-sourced, the user base is even more greatly limited and the company is not taking advantage of the millions of developers out there, who might improve on their code, or add new features. This device is still hardware heavy and is therefore difficult to lower in price.

HIMS Braille Sense U2 [3]

The HIMS Braille Sense U2 and the HIMS Braille Sense U2 QWERTY are two versions of a virtually complete computer for the visually impaired. Instead of a screen, it has one line of Braille pins where the user can see what they read or write. It comes packed with Microsoft Office, access to the internet, and many more features of a modern day computer. The price for such a device is $7195, an incredibly steep price, that once again leaves the majority of the worlds impoverished visually impaired persons behind. The device practically works in parallel with modern day computers. By this we mean that it does not take very much advantage of what has come before it. Instead of being a device that allows the visually impaired to share in modern day technology, it reinvents it instead. Given the very small market for devices for the visually impaired, this limits the access they have to great engineers, blows up the cost of the device, and makes the problem more difficult than it should be.

DrawBraille Phone [4]



The DrawBraille phone was a device that gained in popularity in 2012, where it was declared the first phone for the visually impaired. It was at the time a concept and no prototype or proof of concept has been built. Today it has still not been created and little to no new information since 2012 exists relating to the device. It is fairly safe to claim that the DrawBraille phone project likely has been dropped or is at least on hold.

Design

Through looking at the problem, as well as existing products, the needed solution was clear: A small interactive device that receives smartphone content and converts it into a user-friendly format for the visually impaired.

The device includes a refreshable braille display composed of pins, and a keyboard; the device interfaces with a corresponding smartphone application via bluetooth. The app gives permission to read and write data to and from apps on the smartphone, such as sending and receiving text messages. The app then transmits the data to and from the smartphone to the S.A.V.I device, which displays the content on the refreshable pin display.

In our ideal device (see Figure 1) we have imagined a pocket-sized device much like an enlarged version of an older slider mobile phone. The main difference between such a phone and our device would be the matrix of adjustable pins that would replace the screen on a regular phone. The reason this is our ideal concept device is because the matrix of pins allows us to not only represent braille, but also provides the opportunity to become more creative and perhaps create textured images or other interesting things that a regular braille character array could not. With our current design for braille pins, this idea is not practical and would be too expensive for the scope of capstone. For that reason we devised a second design as seen in Figure 2, which contains only an array of braille pins, making it a cheaper solution but also limits the flexibility of the device. With the array instead of a matrix, we can fit both the keyboard and the display onto one surface.

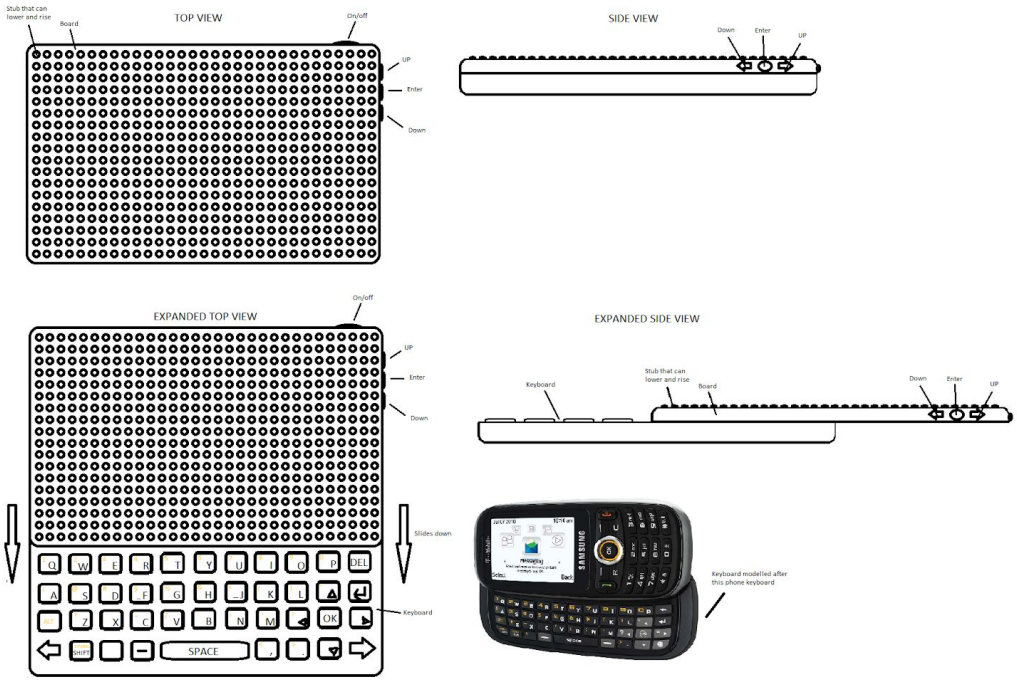


Figure 1 - Ideal S.A.V.I design

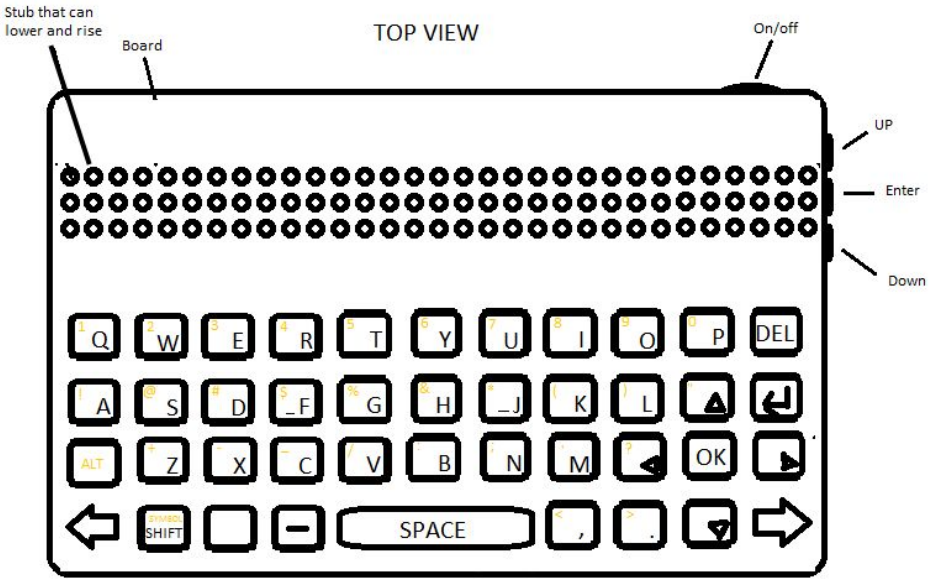


Figure 2 - S.A.V.I Second Revision

We continued to develop the device and realized some potential problems we would face, we ultimately ended up with the design shown in Figure 3. This design features a braille keyboard, which is both easier to adapt for the device as well as more convenient for the target user base. In addition to this, the device better compensates for the limitations faced during the prototyping phase. Lastly Figure 4 shows our ideal device. This device would be a realistic implementation of a device with the advantages that would be brought with creating this product in scale and with size in mind.

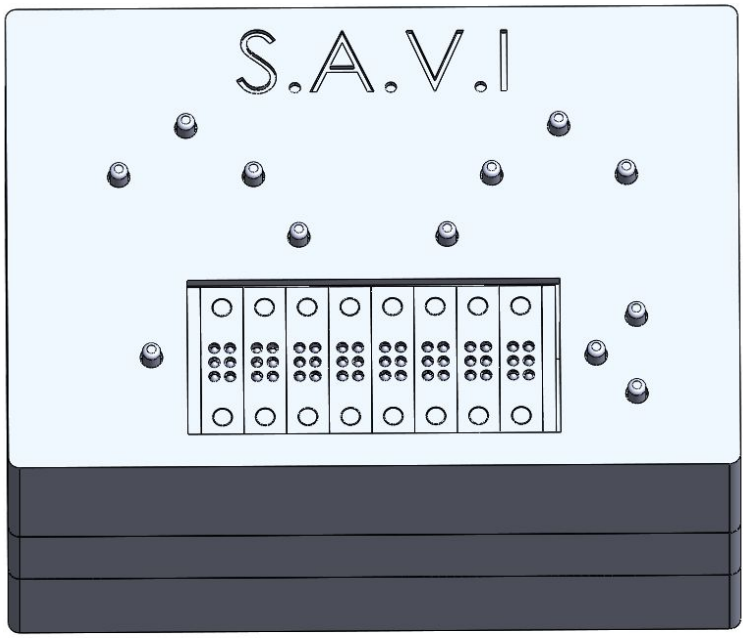


Figure 3 - S.A.V.I Current Revision

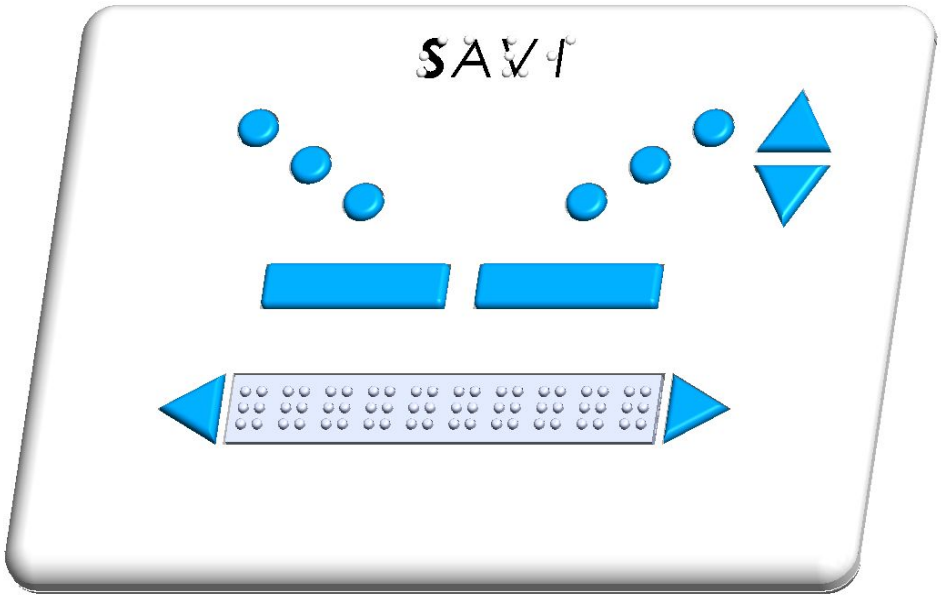


Figure 4 - S.A.V.I Future Revision

The design in Figure 4 is accomplishable assuming the success of our proposed improved solution, which involves muscle wire to move the braille pins. We managed to design and test a single pin using muscle wire to confirm that the idea is possible, and believe that given more time, we could design a version of SAVI, that only slightly larger than the standard android smartphone. We believe the motors to be the bottleneck, for size simply because most other hardware components have been proven shrinkable. Replacing the arduino mega with a custom designed circuit controller and the batteries with the type of lithium ion battery in phones would already drastically decrease the size of our design.

Parts and Implementation

Hardware

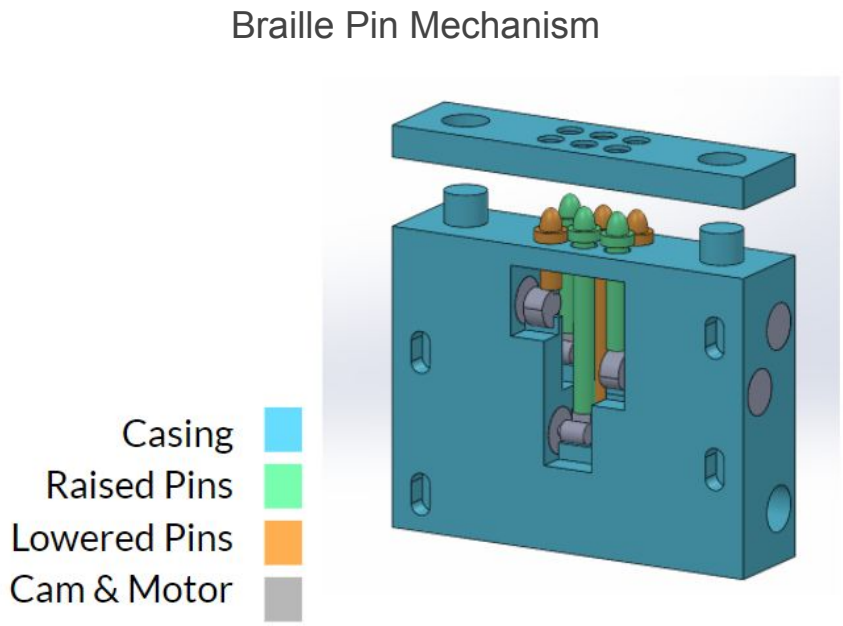


Figure 5 - SolidWorks Pin Design

The linear movement of the braille pins was a big design milestone for us. From an initial array of high risk, high reward to low risk, low reward designs ideas, we decided to implement a design that was likely to succeed and work at the end of Capstone and which, would still allow us to create a relatively small device. Our solution was to work with tiny DC motors to move and operate each pin independently. We designed a casing, cam and pins to go around the motors, as illustrated in Figure 5, which would ultimately function to convert the DC motors rotational spin, into the linear movement of the pins.

Braille Keyboard

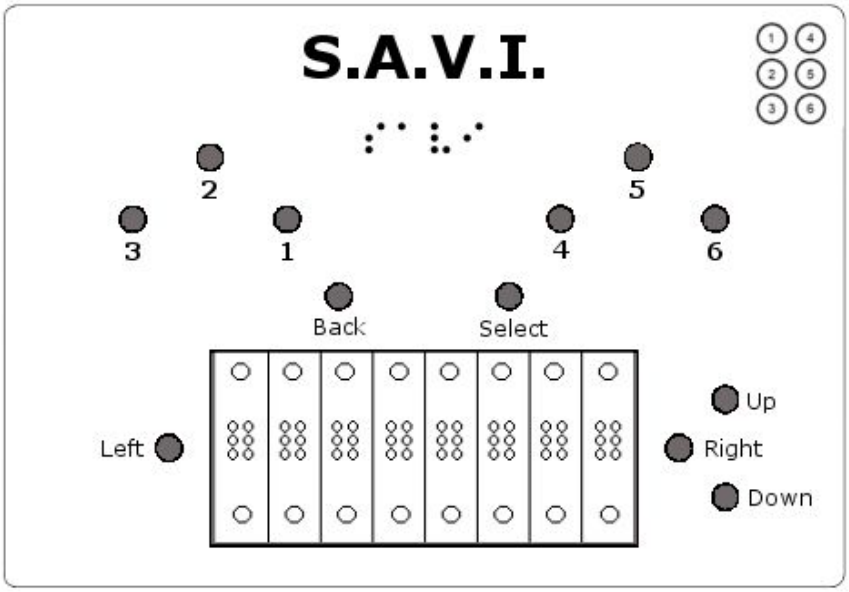


Figure 6 - S.A.V.I.'s Braille Keyboard

The device uses a custom braille keyboard that mimics the common braille keyboard for typing in a character, which is done by using buttons 1 through 6 and select in Figure 6 above. It has an additional 5 buttons to allow easy navigation through a smartphone. Up, down, left and right allow you to navigate texts and menus. Back and select help you enter or exit applications.

In order to ensure quality braille reading, the braille pins will meet the official braille measurements [5]:

Dot base diameter	1.5mm ~ 1.6mm
Distance between two dots in same cell	2.3mm ~ 2.5mm
Distance between corresponding dots in adjacent cells	6.1mm ~ 7.6mm
Dot height	0.6mm ~ 0.9mm
Distance between corresponding dots from one cell directly below	10mm ~ 10.2mm

Figure 7 - Official Braille Dimensions

Logic Controller

In order to control and regulate the embedded device a programmable logic controller was needed. As this was a prototype, it was decided that an Arduino Mega would suffice. The Arduino Mega has sufficient inputs and outputs to allow for the control of a large number motors in the device. The input pins on the Arduino also have pull-up resistors installed on board, making it is simple to create an input keyboard for the user with only the use of buttons and without requiring additional resistors. In addition to this, documentation and additional components are both easy to find for the Arduino Mega which minimizes the chance of facing a problem on the PLC side. Although an Arduino Mega would be replaced by a custom circuit controller in the final product, it was a great choice for the early portions of the design cycle which the majority of this project was within.

Driver Circuit

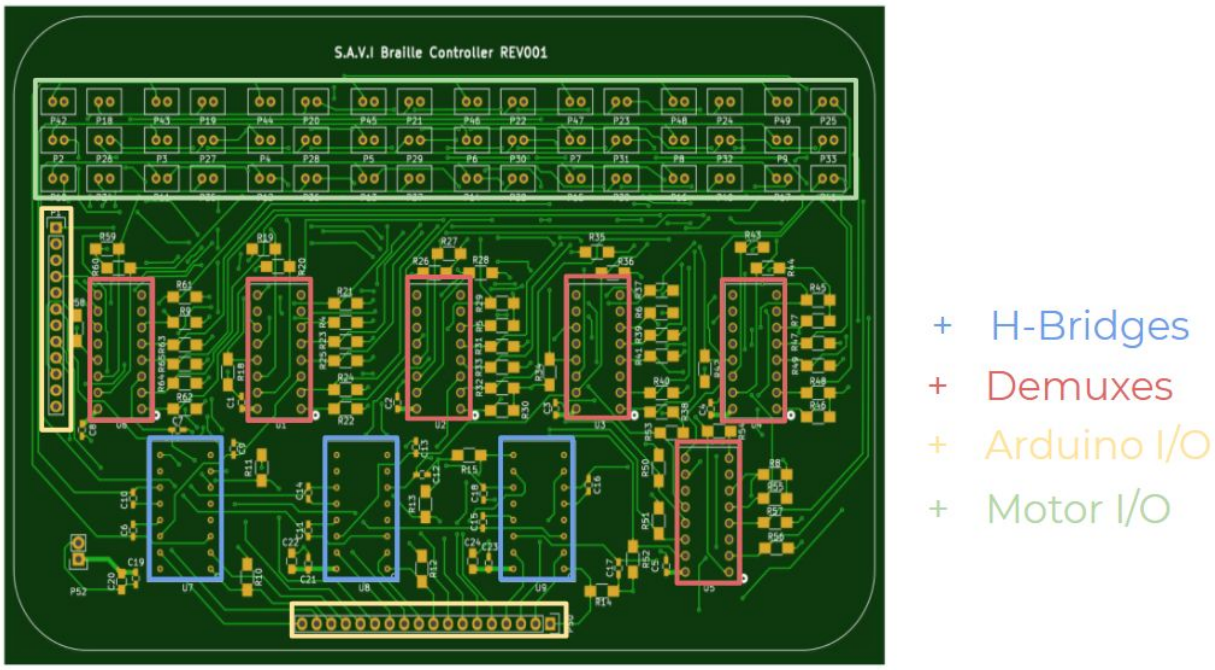


Figure 8 - 4 Layer Driver Circuit

The braille pins are controlled by a dedicated driver circuit controlled by digital signals from the logic controller. The driver circuit is custom designed in order to physically fit the device and be capable of handling the switching of state for the number of braille pins used. It includes demultiplexers which are able to take a combination of 3 digital inputs and which are used to select the appropriate output respective to the pin that needs to be moved. This is beneficial since it drastically reduces the number of output pins required on the logic controller allowing the device to be as small and efficient as possible. The motor control signals themselves are generated using a design similar to that of a solenoid/motor driving circuit which will be switched

on and off momentarily using metal-oxide-semiconductor field-effect-transistors, or MOSFETs [10, 11]. This was done by using 3 dual h-bridge integrated circuits where each one is capable of providing the signals for 2 motors. Since there are three of these integrated circuits, the board can provide the control signals for a total of 6 motors which are further demultiplexed to control the 48 motors used individually, and 6 motors simultaneously. The H-Bridge integrated circuits are capable of providing control for the speed, direction, and on time for the motors. This ensures that current is only being passed through the braille pin for a short duration of time which prevents overheating and any safety risks while switching the position of the braille pins.

Power Management

The device is powered by a single 18650 lithium-ion high discharge cell. This provides a nominal 3.7V for the power regulators while providing 2.5 Amp-hours of capacity [12]. The device could be designed with multiple cells in parallel to provide more capacity, and thus more usage time for the device [13]. However, a single cell is chosen so as to fit all the parts neatly in the enclosure. The device is rechargeable and is charged using a micro USB cable. The runtime of the device is affected by the number of pins on the device as well as the pin mechanism selected. Lithium-ion cells were selected due to their high capacity and relatively small size along with features such as high discharge rates. In a commercial device, the battery would be replaced by the lithium-ion style battery such as those found in smartphones as they have a lower profile allowing to stack the battery with the other electronics.

With a single cell, the nominal output voltage is 3.7V with the voltage ranging from 3.3V(discharged) to 4.2V(fully charged). Each cell has a capacity of 2.5 Amp-hour providing a total nominal power of 9.25 Watt-hour [12, 14]. The device should be able to provide 2.5 amps of current at 3.7V for an hour continuously, however, the device uses current mainly to switch the pins up/down. This means that the batteries should last relatively long since the pins require very short durations of power to switch their position.

Total current for pins:

$$I = I\,pin * N$$

Where **I** represents the total current, $\mathbf{I_{Pin}}$ the switching current per pin, **N** the number of pins.

Total power for pins:

$$P = I * V$$

Where **P** represents the total power, **I** the total current, **V** the voltage needed to switch each pin.

Total time for device:

$$T = \frac{9.25\ Watt{-}hour}{P}$$

Where **P** represents the total power, **T** the runtime of the device.

The battery is connected to a battery management system which ensures that the batteries are safely charged to the required levels. In addition to this, it also provides safety features such as short circuit protection and over-discharge protection. The battery management system is an off the shelf breakout circuit board designed specifically for 18650 lithium ion cells.

The battery is connected to switching voltage regulators providing the required voltage level of 9V for the Arduino Mega and 5V for the driver circuit as well as the bluetooth module. The voltage regulator breakout boards have a high current output rating and are capable of handling the current requirements of the system. The number of voltage regulators is affected by the current requirements of the system and is designed to maintain a desirable operating temperature so as to ensure safety while using the device.

In a commercial manufactured device, both the battery charging circuitry, as well as the voltage regulation circuitry would be placed on a common circuit board. This would likely be a single circuit board that would be combined with the driver circuit.

The device could be designed to plug into the cell phone in order to supply its power but this is not ideal as most cell phones are not designed for this. This could harm the cell phone, and it would drain the cell phone's battery very quickly which would result in inconveniences for the user. Thus, rechargeable batteries were chosen as the ideal power source.

Bluetooth Hardware

The bluetooth module is an off-the-shelf breakout Bluetooth Low Energy (BLE) board that uses bluetooth 4.0 and communicates with the smartphone. It handles receiving data from the smartphone and transferring it to the Arduino where the information is processed for controlling the braille pin array. The bluetooth module is also responsible for relaying information back to the smartphone from inputs such as the braille keypad and the push buttons on the device.

Encasement

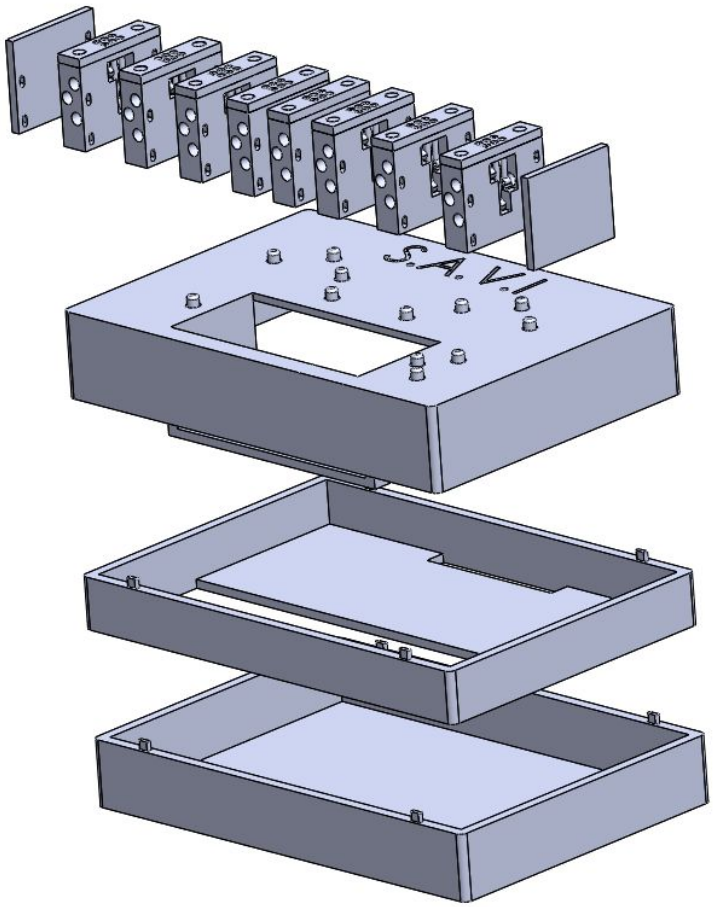


Figure 9 - Prototype Device Encasement

The prototype device encasement was designed to safely and securely hold all of the components of SAVI together. The 4 layer circuit board shown in Figure 9 would sit on the bottom layer of the case, on the middle layer the Arduino Mega would rest, and at the top of the device the braille display mechanism as well as the buttons would be located.

Software

S.A.V.I. inherently needed a sizeable amount of code in order to meet the functionality requirements which our design demanded. The software which was written can be broken up into two primary components; the Android application, and the embedded software which runs on the Arduino.

Android Application

The Android application associated with the S.A.V.I device is tasked with handling the majority of the logic because Android smartphones are powerful small computers and we want to take advantage of that processing power. Doing this allows us to minimize costs for hardware, which ultimately makes our product both smaller and cheaper than if we were to build the logic into the S.A.V.I device. We also take advantage of the smartphones access to WIFI and it's infrastructure for updating software. Having our software primarily based on the smartphone allows us to push updates the moment we make them. This would not be possible on the S.A.V.I device, as it does not have access to the internet.

More importantly than reducing price or making updates easy; by moving the logic onto the android smartphone, we allow our S.A.V.I device to become more like a universal display. What is meant by this is that in the future we could design a PC application or applications for IoT which could also utilize the S.A.V.I display. Furthermore, the open-source smartphone community is large; by building our logic into our app and making it open-source, we are able to tap into a large, willing, enthusiastic workforce. Not only will this allow us to take advantage of free labor, it might also become the beginning of a community of programmers dedicated to assist in developing new solutions for bringing tech to the physically and mentally impaired.

Logic

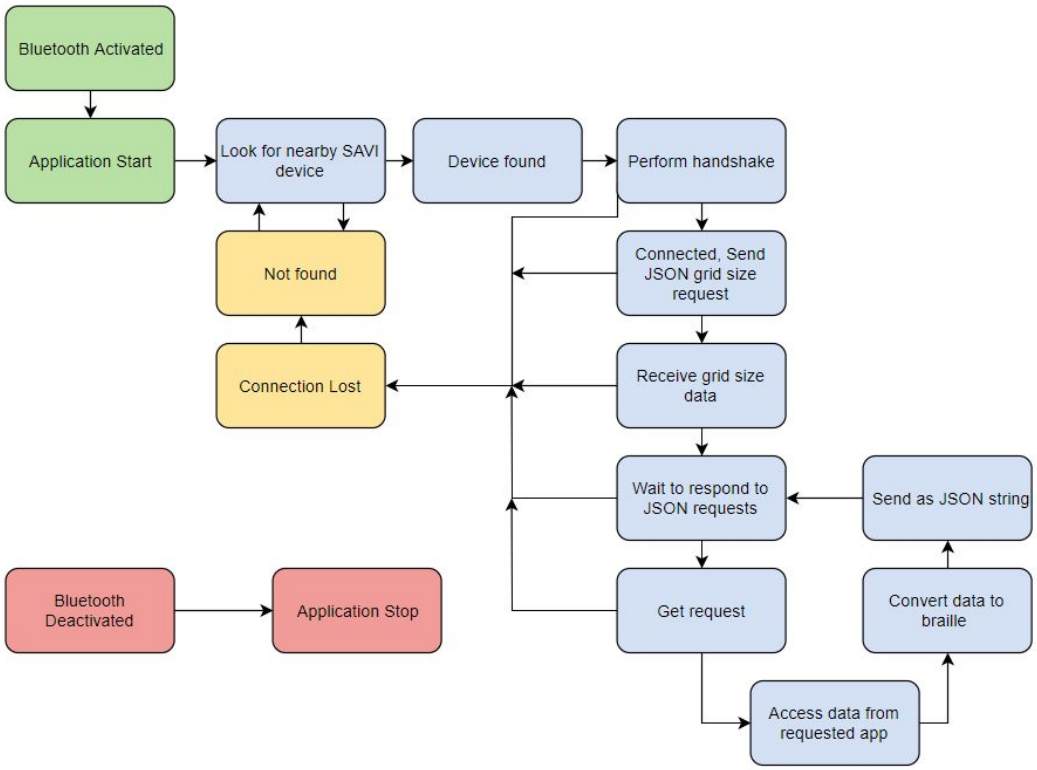


Figure 10 - Android App Logic Flow Chart

To visualize the logic for the android application see Figure 10. The first task of the android application is to establish and maintain a Bluetooth connection to the S.A.V.I device. It does this by looking for a specific name assigned to the S.A.V.I bluetooth device. Once it identifies the device, the app will request to establish a connection. The S.A.V.I device will then confirm the identity of the android phone and acknowledge the connection.

Once the Bluetooth connection has been established, the application will simultaneously request a grid size from the S.A.V.I. device, which it uses to format the braille text for any braille screen size, and prompt the user for permission to access all current apps that it can connect to (for capstone this is limited to messages and contacts) and given that the user gives permission to the app, it will be able to access the various databases to read and write information such as contacts and text messages.

After the S.A.V.I app pulls the data from another app, the data is passed through a function that converts the content into braille. The braille is then converted into a byte array that represents the matrix of pins on the S.A.V.I device. The byte array is then passed over bluetooth to the S.A.V.I device.

If the S.A.V.I device passes data back to the phone, the app reads the byte array and converts it to the correct format for the current app in use. It then transmits the data to the local app and that app responds as it normally would, by for example sending a text message or creating a new contact.

API

We have a messaging structure in place where the app sends messages to the embedded device as text over Bluetooth in a JSON format. The embedded device the reacts to each of those messages. The embedded device will also be sending back various forms of status messages. Some of our known message types are the following (from the perspective of the app):

Setup data

The braille display grid size is sent to the android app, which the app uses to format the braille text.

Sent data

A byte array of “ ”’s and “x”’s is sent from the smartphone to the device to control the pin display.

Received data

Keyboard input and device status information is passed back to the smartphone from the device.

UI

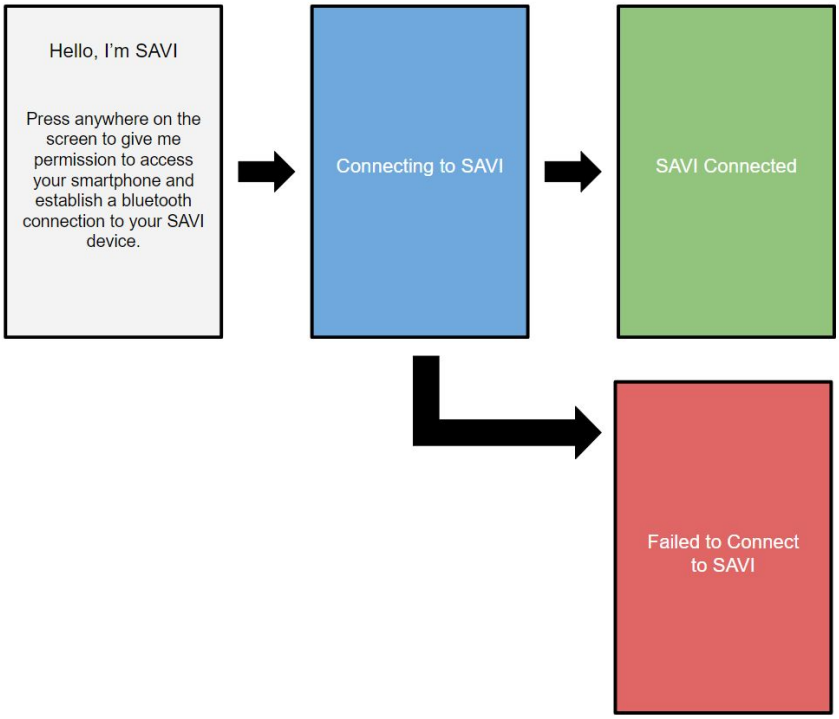


Figure 11 - Android App UI

Our android application was designed as an android service, which allows it to run in the background of the phone without needing a UI interface. That being said, the application does have a UI to help the user setup the app initially and as a guide to indicate the current state of the app's connection to the S.A.V.I. device. An illustration of the apps display can be seen in Figure 11. The UI consist of a brief intro page and 3 connection pages, which indicate whether the device is connecting, connected, or failed to connect. The screens consists of text, and a one color background. Given that some legally blind persons can see faint colors, the mono colored background will be useful in setting up the device initially.

Future Improvements

Post Capstone we hope to expand the S.A.V.I app to connect to more local apps such as; calendar, contacts, maps, etc. and ultimately to make the entire phone accessible through the device. Going even further we would develop our own flavor of android, which would allow us to make a device to replace the smartphone, but which still has access to all the power of android and its developers.

Another improvement that could be made, would be to make similar applications as the android app but for other Bluetooth enabled devices such as a PC or some IoT devices. This would make even more of the world of tech accessible to the visually impaired.

Arduino Embedded Software

The embedded software is the code which runs on the Arduino device. This code operates in one continuous loop that repeatedly iterates as the device is running. The code itself has three main components; the main loop, key event handling, and the display driver function.

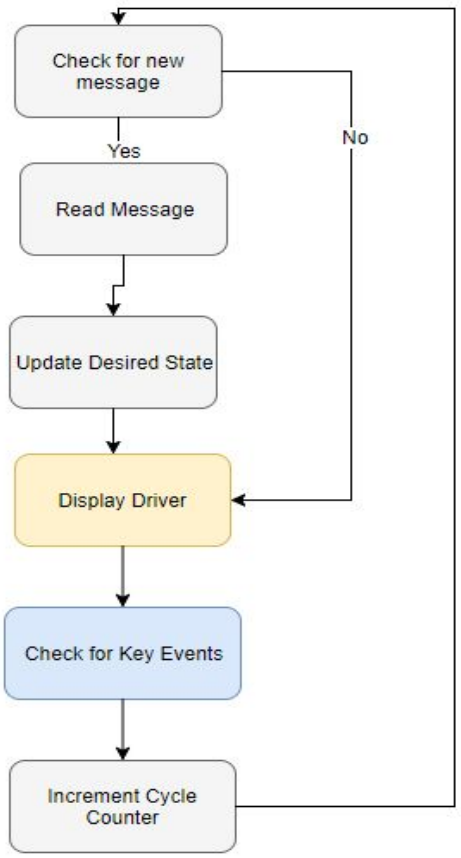


Figure 12 - Embedded Software Main Loop

The main loop of the embedded software, as shown above in Figure 12, is what drives the software of the device. This loop takes in inputs from both the user and the application, and delivers those inputs as outputs to the other. The logical start point of the loop is when the arduino checks for any messages from the Android application. If there are not any messages, the loop jumps ahead to the display driver function which is described further below. If there are any messages from the application, the arduino then deciphers the message and updates the desired state of the device before entering the display driver function. After the display driver function completes, the key event handler runs before finally incrementing the cycle count and going back to the beginning of the loop.

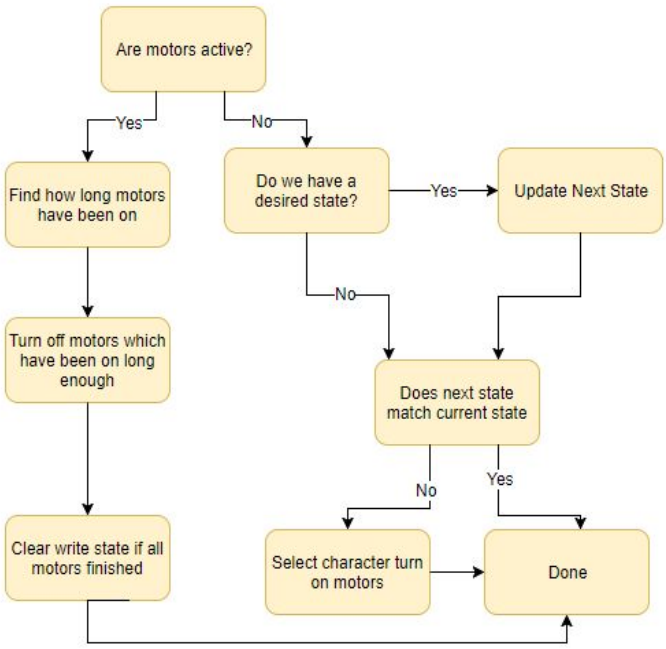


Figure 13 - Display Driver Function

The display driver function, shown in Figure 13 above, drives the 48 motors which drive the 8 braille characters. Along with the logic above, the function cares about three states; Current State, Next State, and Desired State. The Current State is the state that is currently being represented by the braille display. The Next State is the state which will next be appearing on the driver, and the desired state is the state which the display will eventually show. This function maintains those states as well as controls the motors to verify that they are in the correct state, and transition to the pending states.

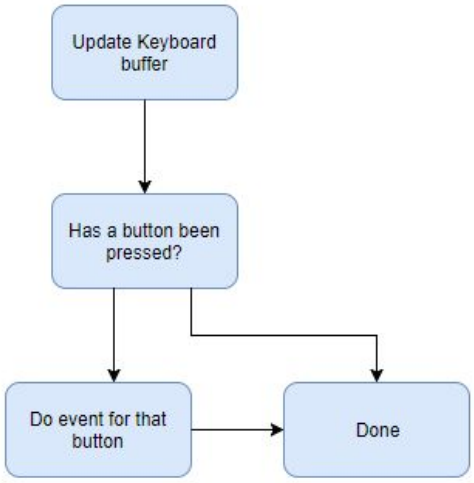


Figure 14 - Key Event Handler

The key event handler function, depicted in Figure 14 above, occurs whenever a key is pressed on the device. The handler then performs the event depending on which button was pressed. Many of the events are simply handed off to the application, while others are handled locally on the device. Once all needed functions are complete, the software loop resumes.

Results and Discussion

Final Prototype

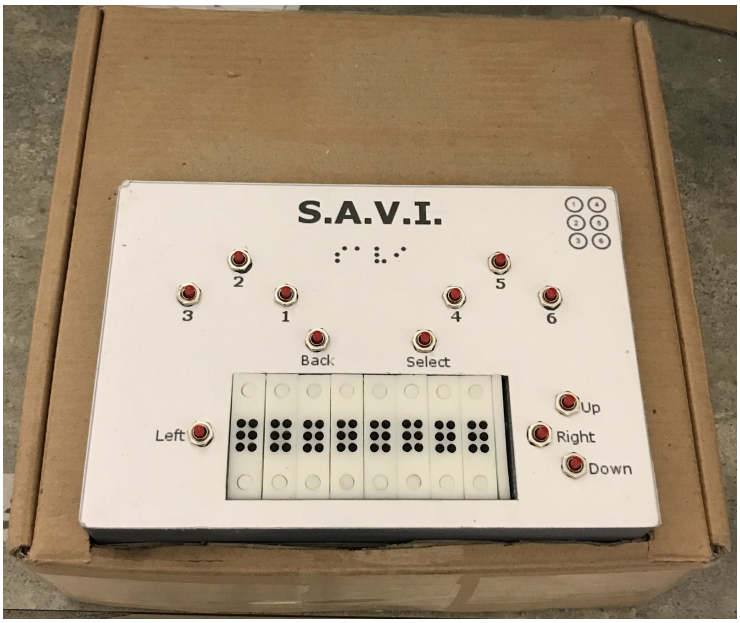


Figure 15 - S.A.V.I. Final Prototype

From Figure 16 it is clear to see that the final braille design came out slightly bigger than anticipated. The primary reason for this was a lack of time to accurately place all parts of the device in the smaller casing and trim wires. Additionally we came to the conclusion that ultimately the box would allow us to open our device and and present our hardware. Also it would make it easier for us to perform last minute fixes, given something should go wrong with the device on the day of presentation.

Ultimately the device performed as it expected, in order for easier displaying and working with the device, we designed a sticker to apply to its casing to more easily use it to navigate. In the end we decided to develop our prototype with 8 braille characters as we saw this as sufficient for illustrating the functionality of the device.

The final product managed to be mostly functional, as seen in Figure 16 where two of our braille characters are displayed together, we had some last minute stability issues with our pins, which we managed to fix with laser cut support frames. Additionally we proved that both software and hardware worked together, however due to some incidents with gluing the cams to the motors, some motors we rendered immovable. Despite this, enough motors were fully functioning to

confirm that our product was functional. In the future, more caution will need to be taken in respects to gluing these miniature parts together.

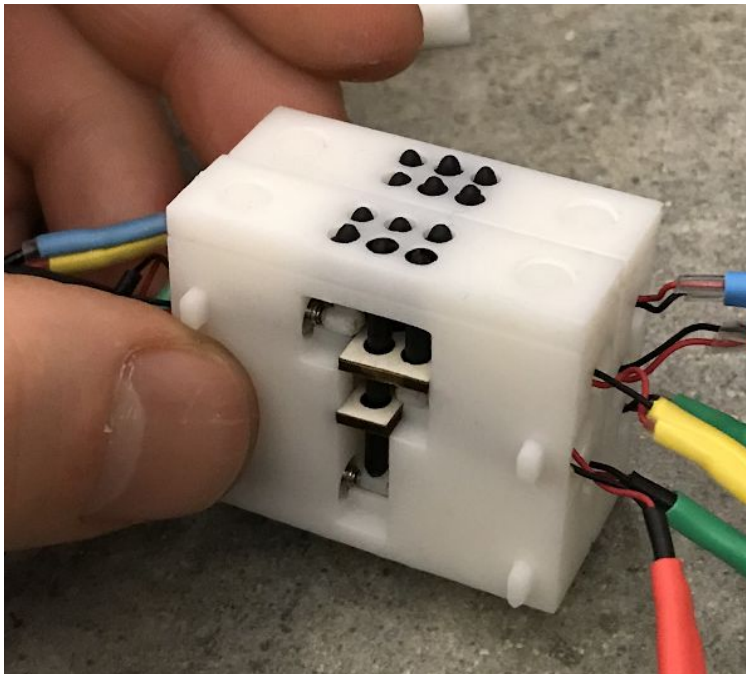


Figure 16 - 2 Refreshable Braille Characters

Testing

Android

In order to test the S.A.V.I application, we designed a virtual model of the S.A.V.I device, which we pass data to over Bluetooth. We pass data with a known expected outcome to check visually that the data that we expect to be passed, is passed. We also test how the app responds to data being sent from the device. The test was successful, proving the functionality of our application.

Embedded

We used standard unit testing to test our embedded software. We also have a virtual device that we used to simulate hardware so we could test the rest of our code without the actual hardware. This enabled us to test much of our software before the hardware side was ready for it, helping us identify bugs earlier in the development process. We also tested our code with an LED matrix display. As the pins have two states (up or down), the LEDs in the matrix also have two states (on or off) this allowed us to test our software with some amount of hardware being involved as well.

Fullstack

Fullstack tests were run in a similar fashion as for testing with the virtual device. Basically we tried to move through the application in a live demo and confirmed that how the device and the phone interacted with one another, was correct.

Costs and Cost Analysis

Braille Pin Housing*:

Part	Qty	Cost
Pin 0, 5	16	$4.48
Pin 1	8	$4.64
Pin 2, 3	16	$6.72
Pin 4	8	$5.92
Motor Housing	8	$129.28
Top Cover	8	$23.04
Side Covers	2	$21.01
Total:	66	$162.91

*cost of 3D prints with high-resin material at 25 micrometer precision.

General*:

Part	Qty	Cost
Printed Circuit Board	1	$62
SAVI Housing	1	$50
Braille Pin Housing	1	$162.91
DC Motors	48	$8
Bluetooth Module	1	$11
Circuit Components	N/A	$40
Battery Components	N/A	$15
Total:	N/A	348.91$

*Cost reflects parts used, not parts purchased

Challenges Faced

Although the general concept of the project may seem simple, numerous challenges hindered progress throughout the design and implementation processes. The first problem faced was determining the optimal method to use for pin motion. Magnets, motors, muscle wire, forced air, and a few other propulsion methods were considered before ultimately deciding upon motors for the initial prototype. Once motors were chosen, a different issue was realized, being able to independently control 48 motors in parallel. Eventually this problem was overcome only to face another problem. Although the ability to control 48 motors individually was there, a mechanical design to turn the rotational motion into vertical motion was needed. As this was primarily a mechanical engineering problem, an area outside of this teams' expertise, it proved to be one of the most difficult problems to solve. Even after reaching a well thought out solution, issues with the stability of the design required a work around to make the solution function properly on a consistent basis. Ultimately many of the challenges faced when designing the mechanical components stemmed mostly from the delay between ordering parts, which generally took weeks, and the limited time we had to get the designs right. With only time to make 1 or 2 iterations of our mechanics, designing the mechanics of the device became a greater challenge than it had to be.

Conclusion

Throughout this report and ultimately the entire process of Capstone, we have proven that our design is capable of dramatically lowering the price of a braille device, that it is fully capable of interacting with any facet of a smartphone, and that increased accessibility to technology for the visually impaired is within reach. All though our design was developed to function as a prototype, it is clear that most of the sizing hurdles of the device are related to wiring and the current choice for batteries, which are both components that are known to be shrinkable. Furthermore, by transitioning from motors to muscle wire, we could move from making one of the smallest braille computers in the market to making a device that competes in size with actual mobile phones.

Acknowledgements

Throughout this capstone, numerous individuals outside of the team went above and beyond in helping our team in creating SAVI. First and foremost, Professor Charles Dimarzio advised our group throughout the entire design process. Professor Dimarzio's guidance was invaluable throughout the process, and his experience led our team to create a much better product as a result. Secondly, the National Braille Press provided us with a recognized resource to better understand the problem we sought to solve, as well as the successes and failures that similar solutions encountered. Professor Stephen Intille of Northeastern University assisted our team in the implementation details concerning the Bluetooth and Android application components of this project. Doug from 3D Hubs helped verify the robustness of our mechanical designs as well as the verified the integrity of those designs after manufacturing, but before shipping. Throughout the project, Jonathan Sohnis and Victor Pesak were a great help in discussing ideas and with messing around with all kinds of ideas for developing braille pins. Again to everyone that helped us reach the finish line, thank you.

References

1. Apple. “Apple Accessibility: VoiceOver Getting Started Guide.” *Apple Help Library*, Apple, 2016, www.help.apple.com/voiceover/info/guide/10.12/

2. “HumanWare Brailliant BI 32 Braille Display.” *Apple*, www.apple.com/shop/product/HJB42VC/A/humanware-brailliant-bi-32-braille-display

3. “HIMS Braille Sense U2 QWERTY.” *Braille Notetakers - Blindness - Vision Impairment*, www.enablemart.com/braille-sense-u2-qwerty?gclid=EAIaIQobChMI9sqvmbbY1QIVEFmGCh2pcQSMEAkYAyABEgIfkfD_BwE

4. Seth, Radhika. “The Ultimate Braille Phone.” *Yanko Design*, Yanko Design, 19 Feb. 2012, www.yankodesign.com/2012/02/20/the-ultimate-braille-phone/.

5. “Size and Spacing of Braille Characters.” Braille Authority, http://www.brailleauthority.org/sizespacingofbraille/sizespacingofbraille.pdf

6. R. Nave, “Solenoid Field from Ampere’s Law.”, http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/solenoid.html

7. G. L.Pollack and D. R. Stump, “Magnetic Field of a Bar Magnet.”, Electromagnetism, http://www.pa.msu.edu/people/stump/EM/chap9/9ex1.pdf

8. R. Nave, “Magnetic Force.”, http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/magfor.html

9. “Technical Characteristics of Flexinol.” Dynalloy, Inc. Makers of Dynamic Alloys, https://cdn.sparkfun.com/datasheets/Components/General/TCF1140.pdf

10. “MOSFET as a Switch - Using Power MOSFET Switching.” *Basic Electronics Tutorials*, 18 Apr. 2017, www.electronics-tutorials.ws/transistor/tran_7.html.

11. Swagatam. “Simple Mosfet Switch Circuit with Delay Timer.” *Electronic Circuit Projects*, 19 Aug. 2017, www.homemade-circuits.com/2013/04/simple-mosfet-switch-circuit-with-delay.html.

12. “18650 Samsung INR18650-25R R5 2500mAh (GREEN) High Discharge FLAT Top.” *Illumn*, www.illumn.com/18650-samsung-inr18650-25r-r5-2500mah-green-high-discharge-flat-top.html.

13. “Battery Bank Tutorial: Joining Batteries Via Series or Parallel for Increased Power.”*BatteryStuff.com: We Have The Stuff*, www.batterystuff.com/kb/articles/battery-articles/battery-bank-tutorial.html.

14. “Clearing up the Question of Battery Capacity in Electronics –...” *Solar Life Blog*, 28 May 2013, www.goalzero.com/solarlife/2013/05/28/clearing-up-the-question-of-battery-capacity-in-electronics/