# Braille translator

Team 48
Ashmita Chatterjee (ashmita2), Matthew Price (mjprice2), Aayush Raj (aayushr2)
TA: Shuai Tang
Professor: Rakesh Kumar

# 1 Introduction

## 1.1 Objective

Being visually impaired in a world where the majority of tasks are completed by visually perceiving different objects, can often hamper the day to day functioning of the visually impaired. Technology has made great strides to make their lives easier through the use of products such as braille electronic note-pads, braille watches, etc. However, there is still a long way to go for technology to help them feel more normalized in public settings where they require to read menus, books, sheets of paper, etc.

In public places like libraries, restaurants or even grocery stores, the visually impaired face difficulties if there aren't braille versions available for these texts. Braille is a tactile writing system used by people who are visually impaired. Most public use libraries have a very limited selection of braille texts for users (highlighted in the article linked below [1]) and this limits people's potential and causes a dependency on other people. Since braille menus are very rare (as highlighted in this article linked below [2]), they often feel like they have to depend on others to order food or read a novel. In an attempt to empower the visually impaired further, we are building a hand-held real-time braille translator device which will give back some of their freedom by allowing them to read menus, sheets with normal text simply using this device.

## 1.2 Background

We plan to create a device that will be able to read text that visually impaired people can't and then produce that same text in a format that is perceivable to them. We plan to do this by having a scanner at the bottom of the rectangular device that will be able to scan any English text when placed on a surface with some legible text printed on it. This machine will then process the image data and produce a braille print on the braille display made using servo motors placed at the top of the device. This device should be controllable using buttons placed on the side of the braille display.

This device is different from other products available on the market such as the optacon because it has certain key components that make it a much more user-friendly product than some other devices. The optacon, for example, is a more bulky product and requires the user to move a tiny scanner across the page with one hand while the "read" vibrations produced on a big machine from the other hand. Our device takes a better approach to this problem by using micro servo motors that have a very small footprint and allow the device to be portable without making it too heavy.

Considering the use cases mentioned above, this device addresses certain key issues users face in scenarios like reading books at the library and menus at restaurants. The portability of a device for such uses is important to users as has been correctly identified by our colleague Abhijoy Nandi in his research when building the concept design for Samanaya[3],

which is the basis for our design. Another interesting issue that this device addresses is that users are presented with a format for a text that is a lot more familiar to them - braille.

## 1.3 Visual Aid

The following is the pictorial representation of how we have conceptualized our final product to be working. One purpose is to use the device in a public place like a restaurant to read items on the menu like portrayed in the picture below.

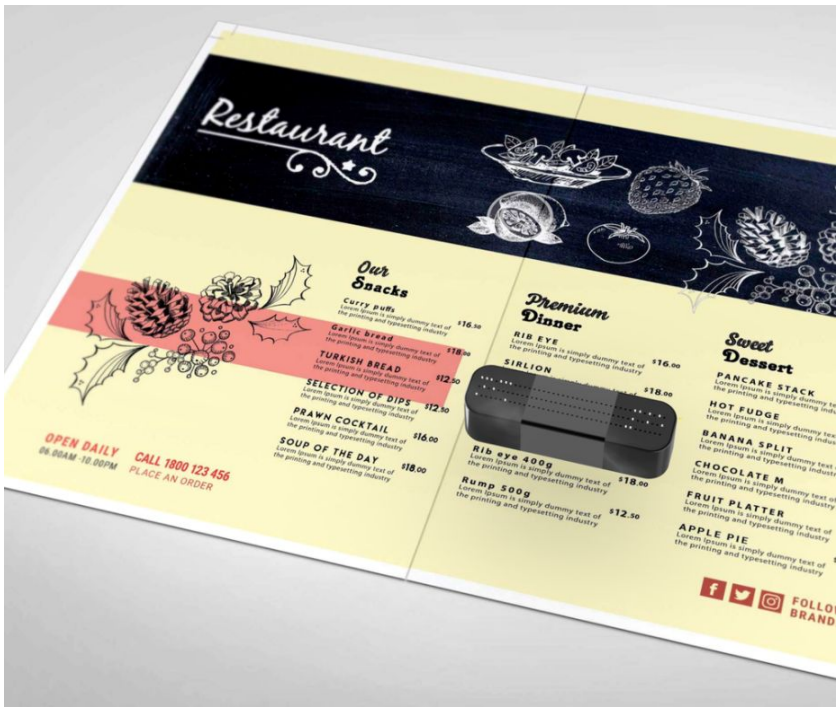


*Figure 1: Concept design showcasing restaurant use [3]*

## 1.4 High Level Requirements

- The device should be able to scan a selected section of text in under 0.5 seconds.
- The device should be able to interpret relevant text with an accuracy of 75%. (interpret relevant text refers to the device being able to scan given text and convert it to a string of characters in the correct order)
- The device should be able to display scanned text in the correct order using a refreshable braille display with an accuracy of 90%.

# 2 Design

## 2.1 Physical Design and Block Diagram

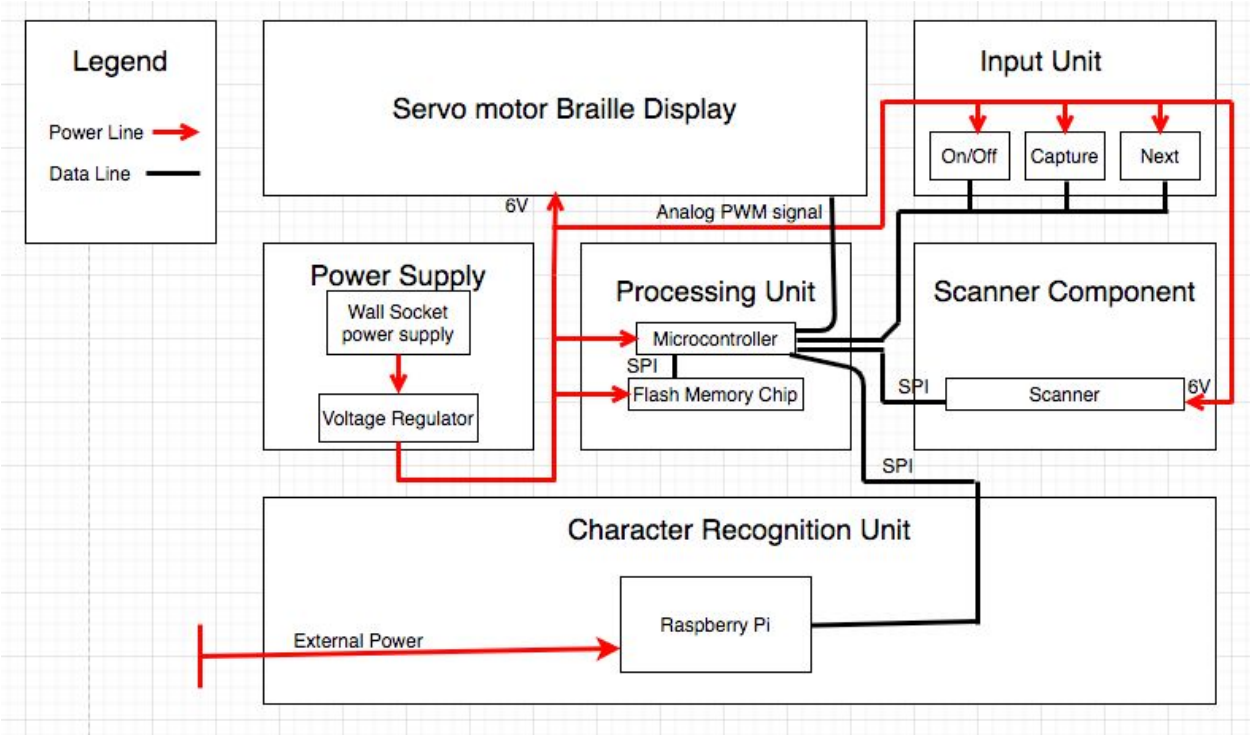


*Figure 2: Block diagram for project*

*Figure 3 : The physical design of the braille translator*

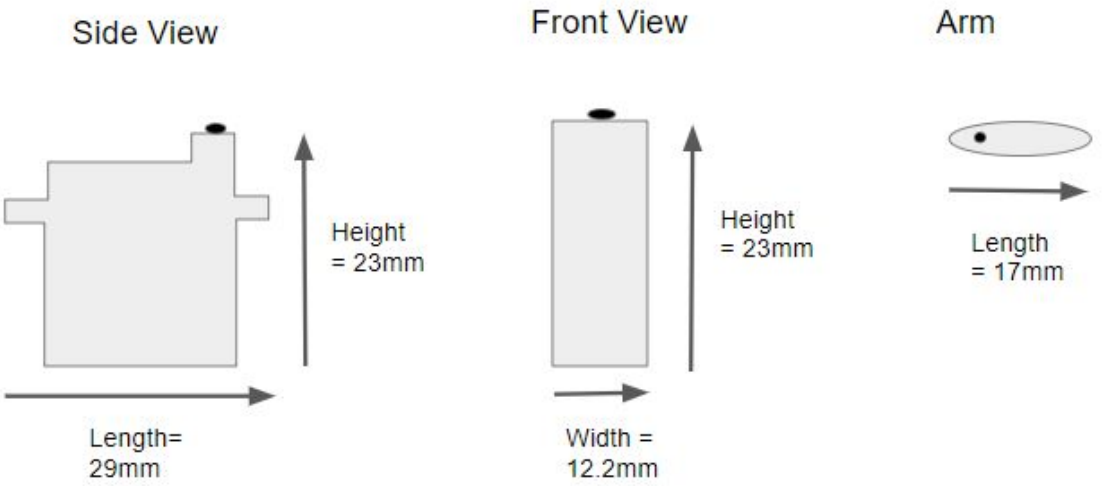


*Figure 4: Dimensions of a single servo motor*

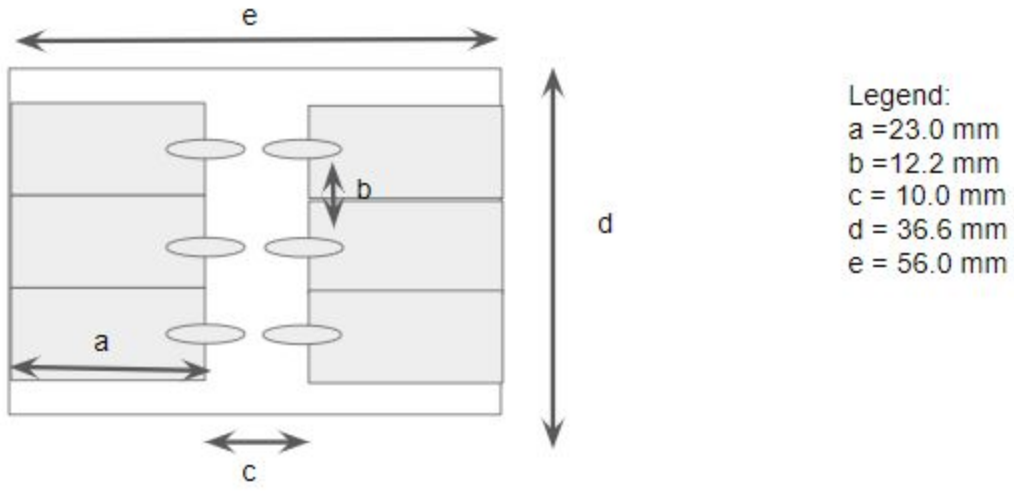


*Figure 5: Layout of motors for a single character*

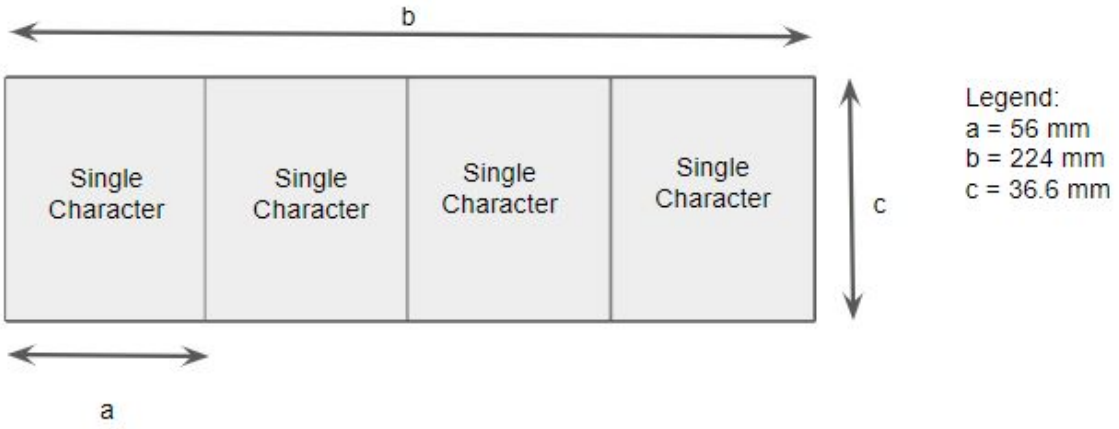


*Figure 6: Layout of motors for the entire device*

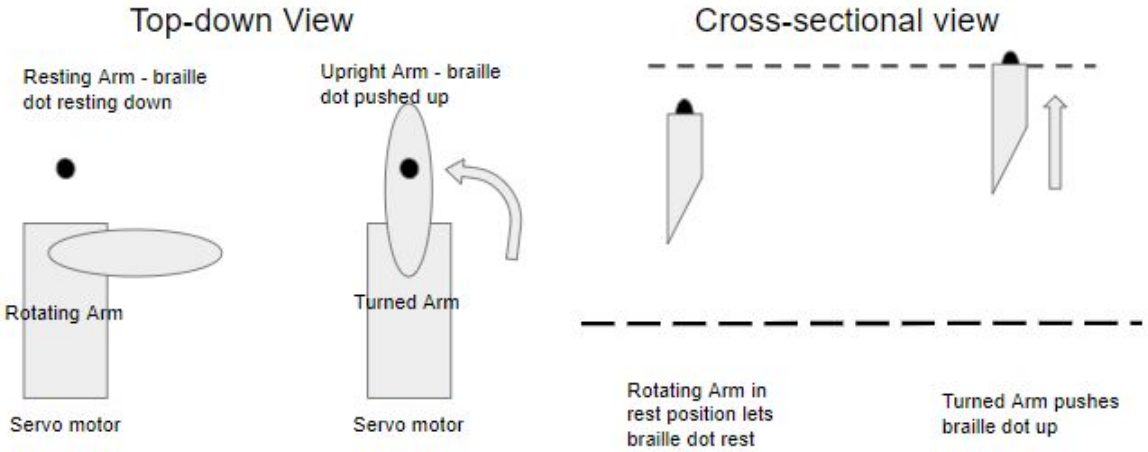


*Figure 7: Diagram of Motor Rotation and Arm Movement*

## 2.2 Functional Overview

The functionality of the device is as described below:

- The device is powered on using the power button.
- The microcontroller receives various inputs from the input component like the power on/off button press, text capture button press, and the next button press. Once power on button press is received, the microcontroller will activate the scanner and the raspberry pi for any bootup procedures needed.
- The device can be placed over any surface with readable text and the capture button is pressed to allow the scanner to scan the text underneath the device.
- On receiving a capture button press, the microcontroller will send signals to the scanner to capture a block of text and will wait for the scanner to finish processing.
- Once the scanner returns an image, the microcontroller will send image data to the raspberry pi connected as part of the character recognition component for further processing and will wait for a return output.
- Once the character recognition component returns a string of characters to the microcontroller, the microcontroller will convert each character to its braille equivalent and send part of the selected text (up to 6 characters) to the braille display. The microcontroller will then wait for more inputs from the user.
- The braille display unit receives a digital signal from the microcontroller containing the string of characters of the deciphered text in braille. The digital signal is then supplied to the appropriate servo motors which are pushed up to form braille characters.
- On pressing the next button, the microcontroller will send the next set of characters to the braille display and will continue to wait for inputs.
- On receiving the power off button press, the microcontroller will power down the scanner, the raspberry pi, and all the other components.

## 2.3 Scanner Component

This component will be responsible for scanning displayed text on which the device is placed. This component should scan text and produce an image as an output that will then be sent over to the microcontroller component for further processing. We are using a scanner because we simply want the digital format of a 2D previously captured image and so a camera would be an overkill. We plan on using a 2D hand-held scanner that will be connected to the Raspberry pi via USB. A typical 2D-hand-held scanner should be around ~$35 which should be fine in our total budget.

Our device should be able to scan a small block of text (~32 characters on a single line) within a reasonable amount of time (0.5 seconds) so that the delay between user input (capture button press) and device output is minimized.

| Requirements | Verification |
|---|---|
| The resolution standard for images that contain text to be interpreted, will be 400 ppi | 1. Use both visual and ISO standard 16067 targets to verify scanner resolution to be at least 400 ppi. |
| Component is able to scan a string of up to 32 characters in a single line within 0.5 seconds | 1. Boot up Brailler device using power on button<br>2. Once boot up completes, place a sheet of paper with 32 characters under the braille device<br>3. Press scan button and start timer<br>4. After 0.5 seconds, boot off the device and extract memory card<br>5. Look for a buffer of 32 characters stored in the memory card to verify the task was completed within specified time. |

## 2.4 Control Component

This component will be responsible for the correct operation of the entire device. It will handle the state machine for the entire device and ensure that each button press is handled correctly. The control component will mainly comprise of the microcontroller unit and the flash memory storage unit.

| Requirements | Verification |
|---|---|
| Correctly send the signals corresponding to the right character. | 1. Send a known character into the microcontroller.<br>2. Monitor the output signals.<br>3. Compare the output signals to the actual braille character |
| Power up or down the device when the power button is pressed. | 1. Toggle the power input.<br>2. Observe whether or not the device turns on.<br>3. Toggle the power again.<br>4. Observe whether or not the device turns off. |

### 2.4.1 Microcontroller Component

The microcontroller will activate the scanner and raspberry pi on booting via USB and after it has detected a button capture the microcontroller will then send over the captured image to the Raspberry Pi for processing and deciphering the text in the image. Once the raspberry pi has deciphered the text using  it will send a string of characters to the microcontroller which will then convert it into a digital signal and with the use of the servo motors the string of characters will be displayed in braille. The microcontroller that we have decided to use will be the 32 bit STM32F427AIH6 and will communicate with the scanner component via USB.

### 2.4.2 Flash Memory Component

The flash memory component will comprise of a SD storage card that will be able to store the text in the captured image in a buffer as we will not be able to display all of the deciphered text in the braille display unit at once due to budget limitations. To overcome this difficulty we decided to display only 4 characters and store the remaining characters in a memory buffer which the user can swipe through using the next button. We will be using a 256kB program storage however requirements may change if we require more storage.

## 2.5 Character Recognition Component

This component is responsible for converting an image to a string of characters. This component should include a raspberry pi connected to the microcontroller. This component will take an image as an input and convert it to a string of characters and return that string back to the microcontroller.

The exact technology we plan on using is OCR (Optical Character Recognition) on a Raspberry pi which can be used via the Tesseract OCR engine on the pi. The ability to recognize the full text in an image is what OCR does. We will be connecting the scanner to the input of the pi and the pi with the already downloaded Tesseract OCR engine should be successfully able to interpret the text in the image captured by the scanner.

| Requirements | Verification |
|---|---|
| Raspberry pi is able to boot up according to microcontroller signal correctly | 1. Connect raspberry pi to external power<br>2. Setup circuit on breadboard to emulate the microcontroller sending out a boot up signal to raspberry pi<br>3. Connect circuit to raspberry pi<br>4. Send signal to boot up<br>5. Send a test command to raspberry pi<br>6. Observe returned output in the form of an LED sequence |
| Raspberry pi is able to receive data from | 1. Connect microcontroller to raspberry pi |

| | |
|---|---|
| microcontroller in the form of an image | 2. Connect raspberry pi to monitor<br>3. Send test image from microcontroller to raspberry pi<br>4. Visualize the image on the monitor to verify functionality |
| Raspberry pi is able to extract text from image (with reasonable quality) and send that text back to microcontroller through data lines within 2 seconds of transmitting image data | 1. Connect raspberry pi to monitor and keyboard<br>2. Run OCR function (using OpenCV) and pass in a test image<br>3. Once Raspberry Pi completes processing the image, it should produce a string of characters which should represent the text shown in the image input. |

## 2.6 Display Component

This component will be responsible for displaying the braille characters for the user to be able to read in real-time. It will be able to take the control signals from the microcontroller and turn those into the correct characters. We will be using servo motors that can be switched on easily, which will be able to push the characters up and down, creating the braille characters.

| Requirements | Verification |
|---|---|
| Each servo motor is controlled independently. | 1. Attach each motor to the breadboard .<br>2. Set up the correct circuit for each motor.<br>3. Supply the 6 volts to the breadboard and across each motor.<br>4. One by one supply each motor with a positive signal and verify that only that one motor moves. |
| The motor will be able to push the characters up and down within 0.2 seconds of receiving a signal. | 1. Attach a motor to the breadboard with the correct supply voltage.<br>2. Supply a positive signal to the motor.<br>3. Record time for the motor to properly move up.<br>4. Supply another positive signal to the motor.<br>5. Record time for the motor to move down. |
| Each servo motor will be controlled to rotate 80-90 degrees, to accurately display braille | 1. Attach a motor to the breadboard with the correct supply voltage. |

| | |
|---|---|
| dot. | 2. Supply a positive signal to the motor.<br>3. Observe for the motor to properly move from the down position to having the braille dot fully pushed up. |

### 2.6.1 Braille Display Plate

This part will be where the motors are attached together. For each individual character, we will need six different motors, because each character is made up of six dots. We will also use a 3D printed armature, in order to assure that the dots are close enough together for someone to recognize them as a single character. The size of each motor will be the constraint for the number of characters we wish to display because we want the device to be handheld, so we want it to be small enough for a person to carry around.

### 2.6.2 Status Indicator

We will use one extra servo motor to push up when the device is powered on, and to be down when it is powered off. This is necessary, because the visually impaired users would not be able to see a light or any other indicator that could signal it being powered on.

## 2.7 Input Component

This component is responsible for correctly passing user input to the microcontroller. It consists of three buttons, “power on/off” button, “image capture” button and “next characters” button. For each button press, the input component is expected to pass a signal to the microcontroller that is then perceived as an interrupt by the system and is handled by the microcontroller.

| Requirements | Verification |
|---|---|
| When the buttons get pushed, a signal is sent to the microcontroller. | 1. Power the device on.<br>2. Press the button of interest.<br>3. Check if the signal is sent to the microcontroller. |

### 2.7.1 Power Button

This button is necessary to toggle the power of our device. This will allow us to not drain the battery pack, considering it is a limited power supply.

### 2.7.2 Capture Button

This button will send a signal to the processor telling it to scan what is currently underneath the device, and then begin the process of converting the characters.

### 2.7.3 Next Characters Button

This button is necessary, because the amount of characters recognized by the device may end up being more that our device can display at once. If this is the case, the next character button will display the next characters that are left in the string.

## 2.8 Power Supply Component

This component is responsible for supplying the correct voltage to each of the different components of the design. The power supply needs to be able to supply different voltages to different components.

| Requirements | Verification |
|---|---|
| Must regulate the 9 V source to provide the required voltages, with minimal noise to ensure our components are safe | 1. Build an example circuit with the voltage regulator and resistors.<br>2. Power the circuit using a 9 volt battery<br>3. Measure the voltage at the output pin using a multimeter. |
| Must supply 3.3 V for the processing unit, scanner component, input unit and character recognition unit. | 1. Power the device on.<br>2. Measure the voltage at the scanner component. |
| Must supply 6 V to the servo motors so that they rotate to the proper position. | 1. Power the device on.<br>2. Measure the voltage across the display unit using a multimeter.<br>3. Check to make sure the motors turn when powered on. |

### 2.8.1 Battery Pack

Our device is to be designed as a portable device, making it easier for the users to bring it to the desired locations. This means that we are going to use a battery pack that can supply the necessary voltage. The motors will require up to 6 volts. We should use a 9 Volt standard battery to provide the necessary voltages.

### 2.8.2 Voltage Regulator

We will need a different voltage for the display component, the scanner, and then everything else. In order to ensure that each component is receiving the correct voltage, we will need three voltage rectifiers to take the input voltage from the batteries and turn it to the correct voltages for each component

## 2.7 Risk Analysis

There are some risks associated with the development of this device however the main few pressure points are the scanner, power supply unit and the servo motors. A few factors that come into play while deciding on a scanner for our device is the size of the scanner and it's compatibility with our selected microcontroller. If we are unable to find a scanner which is portable or small enough to fit in our hands then this device may be at a risk of being too bulky. The scanner should also be able to receive and transmit information to the microcontroller as this is crucial for the completion of our project.

We plan on using a battery pack to power our device however a few risks associated with that is the battery pack may not be able to supply enough power to the servo motors which should individually take up to 6 V, the scanner which should be operable at 9 V and the raspberry pi. We plan on mitigating these risks through trial and error.

## 2.10 Tolerance Analysis

Our device is to be used as a way to convert written text to a braille display in real time in order to help the visually impaired read. The key here is in real time, we obviously want our device to function quickly, so that the user doesn't have a hard time using it. The key timing components we need to be aware of are the time it takes for the servo motor to rotate after receiving the signal and the time it takes for the scanner to scan the entire area of interest.

For the time for the servo motor to spin, it can depend on the total angular distance you want the motor to spin. For the specific servo motor that we chose to use, we know that the speed of rotation is about 0.1 second/60 degrees. Ideally we want our servo motors to spin 85 degrees. So to find the time it will take to rotate we can just use the equation below to find it should take seconds about 0.142 second

$$.0.1/60 = time/85$$

# 3 Project Differences

## 3.1 Overview

The previous solution we had was to build a device that captured a close-up image using a camera (placed 15cms of the surface), processed the image using a raspberry pi and produced output text in the form of braille using solenoids to represent each dot on a braille character.

Our current solution changes the two main parts of the device. We plan to use a scanner to capture images of text printed on a flat surface, then process that image using a raspberry pi and print characters in braille using servo motors to represent dots.

The biggest differences between our previous project and the current one are as follows:

- We will be using a scanner instead of a camera to capture images of the underlying text
- We will be using six servo motors instead of six solenoids to represent a braille character

## 3.2 Analysis

Using solenoids for the braille display allowed us to make a braille display that was simplistic and relatively cheap. When we were searching for how to produce the braille characters, we were limited by the size constraints, as braille characters are very small. These solenoids were small and had a rounded shape, so that we couldn't directly use it as the actual braille dots. This would help simplify the design, keep the design smaller and reduce the number of potential problems. After continuing to design our product, we realized that powering 25 solenoids would be a major problem. This is because, to power a solenoid, you have to continuously be supplying the rated voltage and current, and doing that for up to 25 solenoids at a time adds up. Supplying that much power would eliminate the idea of being portable using batteries, and would also require more mechanical parts to help alleviate some of the demand that the wall power supply would have to supply. More mechanical parts would give our product more areas where we could have problems. When we were redesigning, our main goal was to find a new mechanism that would drastically reduce the power demand. To do this, we landed on servo motors, which only need to be powered when they are turning, not continuously like the solenoids. The drawback to the servo motors is that they are larger, making our device larger than before. Also, we have to be aware that instead of vertical movement, the motors spin, so we have to add an arm that can be moved up and down to display the characters. Given all of the pros and cons, ultimately the servo motors make our design better because reducing the power demand will help us to actually make our product portable as we were hoping for.

Additionally, we changed from a camera module in the first design, to now use a scanner. Our design has either the camera or scanner, as the closest thing to the desired object. This would hopefully be as close as possible, considering we want our device to be small and portable. Originally, when we were using a camera module, we ran into the problem that

most cameras cannot focus and take clear images at a short distance away. Also, having a camera close to the desired image, means that it would not be able to read as many characters as we want. Using the camera effectively meant we may have to change our original design, but using a scanner can help us solve this issue.

# 4 Cost and Schedule

## 4.1 Cost Analysis

When looking to determine the cost of our labor, we settled on the idea that each engineer working on this project should make $50 per hour. Also, we determined that this would take around 100 hours for us to complete. Therefore we were able to see that the cost of our individual labor would be $37,500. In order for us to have our project actually manufactured, we will need the machine shops help to create the physical parts. We estimate that it would take them about 15 hours in order to research and build what we need. At $25 per hour, this means the cost of the machine shop's labor would be $375. The last main cost we have is the total cost of our parts. Below we have a table with our main parts and their cost. The parts total come out to $. In the end, we see the total cost to be $$38,051.36.

| Description | Manufacturer | Part # | Qty | Unit Cost | Total Cost |
|---|---|---|---|---|---|
| Servo Motor | SparkFun | ROB-09065 | 25 | $19.00 (10 pack) | $57.00 |
| Scanner | Konica Minolta | 4037-0015-02 | 1 | $49.00 | $49.00 |
| MicroController | STMicroelectronics | STM32F427AIH6 | 1 | $13.45 | $13.45 |
| Raspberry Pi | Raspberry Pi | Raspberry Pi 4 | 1 | $35.00 | $35.00 |
| Battery | Ultralife | UV9LJPBK | 1 | $17.03 | $17.03 |
| SD Card Port | Samtec Inc. | HSEC8-130-01-S-DV-A | 1 | $4.88 | $4.88 |

Total Cost for R&D: $38,051.36

Engineering Labor Cost: $37,500

Machine Shop Cost: $375

Parts Cost: $176.36

## 4.2 Schedule

The following schedule has been proposed assuming the start date to be the week of spring break and going up to the last working week of Spring semester 2020.

| Date range | Aayush Raj | Matthew Price | Ashmita Chatterjee |
|---|---|---|---|
| 03/16 - 03/22 (Spring | Research alternative | Research | Research servo |

| | | | |
|---|---|---|---|
| break) | parts for pricing and place orders for them. Send email to the machine shop with details on servo motor purchased | microcontroller program loading methods. Work with other members to write test programs for a microcontroller | motor functionality and make approximations regarding power needed to turn the level appropriately |
| 03/23 - 03/29 | Test servo motor functionality and the amount of power needed to rotate the level by 90 degrees (based on calculations done by Ashmita) | Deliver a sample servo motor to the machine shop and figure out if we will need more parts of changes to the physical design of the product. | Research alternative options to a camera. Look into the scanner as a possible replacement and evaluate the changes that have been considered so far. |
| 03/30 - 04/05 | Load microcontroller with state machine instructions and test using basic signal I/O | Design the eagle schematic for the braille display unit, camera, and microprocessor | Load OCR software on Raspberry Pi to test out how OCR works with Raspberry Pi |
| 04/06 - 04/12 | Finalize the PCB design and place the order for the PCB on mypcbway. | Starting to develop the display component, specifically design the metal plate and send in order to the machine shop. | Connect the scanner to the PI and test functionality of the scanner and capability of PI to use OCR to detect text. |
| 04/13 - 04/19 | Ensure OCR works successfully on Pi and then continue to test with an image captured using an optical scanner. | Start testing the braille display unit by connecting the servo motors to power and figuring out voltage requirements. | Building the body of the product to contain the pi, battery pack, and all other components of the device. |
| 04/20 - 04/26 | Start testing individual components with the microcontroller. Begin testing with the scanner and ensure the scanner is able to successfully send the scanned image to the microcontroller. | Test the microcontroller with the servo motors to ensure that the microcontroller can power the servo motors and move them up and down based on a digital signal. | Test the Pi with a microcontroller and ensure the Pi is able to use OCR on an image scanned and sent by the microcontroller and then send a string of characters to the microcontroller. |

| | | | |
|---|---|---|---|
| 04/27 - 05/03 | Continue testing on individual components and ensure that the device is able to successfully translate to braille. | Continue testing on individual components and debug to increase the success rate of translation into braille. | Finish the testing phase and prepare to get the device ready with it satisfying all the high-level requirements mentioned. |
| 05/04 - 05/11 | Ensure that all the components are working separately and together and prepare for the demo. | Prepare for the final presentation and finish up the final paper. | Prepare for the final presentation and the final paper. |

## 5 Ethics and Safety

There are quite a few safety hazards that our project could potentially present. Our device might make use of a battery pack to power the device which is dangerous as batteries can leak. Leakage from batteries can cause hazardous liquid to damage other electronic components present in the device and can also cause the battery to explode. Lithium-ion batteries are known to explode if overheated. Some other potential dangers are electrocution or burning from overheating of the battery or its components. While it may not be possible to avoid these hazards completely, there are precautions that can be taken by us to possibly avoid some of these hazards. We can always make sure that the device is powered down before we make any changes in the circuitry and also remember not to touch any of the components of the device while testing as some of the components may be carrying more current than possible causing it to overheat and cause burns[4].

Our device is also meant to be used in public places like restaurants, libraries, grocery stores, etc. One potential hazard of using a device in places like restaurants or grocery stores are spills. It is extremely easy to spill water or any other liquid in restaurants and if our device was exposed to such a spill there could be possible short circuitry of the device causing it to malfunction or even electrocute its user. To prevent such mishaps we could look into making some of the components of the device waterproof however that may increase the budget of the device. As a temporary fix, we think that a waterproof case for the device may be enough to prevent damage[5].

Another potential hazard of this device as with any other electronic device, it should be kept out of reach for infants. Children using this device should be under adult supervision simply because children often misuse electronic devices as they are not aware of the precautions that one should take before operating such a device.

While there might be quite a few ethical issues with the project one major ethical issue could be with the copyrights to this project. There are other companies or organizations that strive to churn out products aimed towards the visually impaired and hence have already developed similar products. We aim to design and develop a unique product that will cater to the needs of the visually impaired while flaunting a new proposed design compared to our previous project.

## 6 References


[1] *Oh Where, Oh Where, Are the Braille Books?* [Online]. Available: https://www.nfb.org/sites/www.nfb.org/files/images/nfb/publications/bm/bm13/bm1303/bm130308.htm. [Accessed: 29-Mar-2020].

[2] K. Shah, "How American restaurants fail to accommodate blind diners," *Mic*, 09-Sep-2016. [Online]. Available: https://www.mic.com/articles/153739/how-american-restaurants-fail-to-accommodate-blind-diners. [Accessed: 8-Apr-2020].

[3] "Samanya," *Portfolio*. [Online]. Available: https://www.abhijoynandidesigns.com/samanya. [Accessed: 8-Apr-2020].

[4] "IEEE Code of Ethics," *IEEE*. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 4-Apr-2020].

[5] K. CarreroKara, "Stress free ways to protect electronics from rowdy kids and water spills," *Extremely Good Parenting*, 14-Feb-2018. [Online]. Available: https://karacarrero.com/protect-laptop-water-spill-kids/. [Accessed: 14-Apr-2020].

[6] "Konica Minolta Bizhub C450 Optical Scanner Pwb-Ic 4037-0115-02," *eBay*. [Online]. Available: https://www.ebay.com/itm/Konica-Minolta-Bizhub-C450-Optical-Scanner-Pwb-Ic-4037-0115-02/301343535976?hash=item4629797368:g:s~gAAOSw1x1UMtDg&fbclid=IwAR0T_c8JpqhRV8YuBi5gOTtQLqNVsOe306nweP6ba62Z5NNxDrsgfp8Zark. [Accessed: 18-Apr-2020].

[7] R. Squirrel, "Servo - Generic (Sub-Micro Size)," *ROB-09065 - SparkFun Electronics*. [Online]. Available: https://www.sparkfun.com/products/9065. [Accessed: 1-Apr-2020].

[8] "Optical Microscanners and Microspectrometers using Thermal Bimorph Actuators," *Google Books*. [Online]. Available: https://books.google.com/books?id=m2mMT-472qgC&pg=PA212&lpg=PA212&dq=power+consumption+of+optical+scanner&source=bl&ots=ocZZP6zdsb&sig=ACfU3U1Kwl-nKh--b-TwSt0i4sRUNVoTGQ&hl=en&sa=X&ved=2ahUKEwjpyLXO7vDoAhVBV80KHfNFCiEQ6AEwCHoECAwQNQ#v=onepage&q=power consumption of optical scanner&f=false. [Accessed: 13-Apr-2020].

[9] M. Wei-Haas, “This Device Translates Text To Braille in Real Time,” *Smithsonian.com*, 08-May-2017. [Online]. Available: https://www.smithsonianmag.com/innovation/device-translates-text-braille-real-time-180963171/. [Accessed: 14-Apr-2020].

[10] Haque and M. M. Asaduzzaman, “Low Cost Wireless Electronic Braille Reader,” *Academia.edu*. [Online]. Available: https://www.academia.edu/25527839/Low_Cost_Wireless_Electronic_Braille_Reader. [Accessed: 17-Apr-2020].