Brailliant

Yujun Lee, Ziyu Li, Samay Sahoo

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—**Brailliant aims to provide a system capable of translating English text to braille and displaying braille on a pad of actuators controlled by an Arduino. It is integrated with a web-application to transfer text input to the pad with ease. It explores innovative ways to actuate multiple pins and utilizes 3D printing to achieve such designs. It is more than 8 times cheaper than typical commercial braille displays and designed to offer a widely accessible braille translation/education tool to the visually impaired.

***Index Terms*—**Stepper motor, Braille, Database, Web Application, Arduino, Python, HTML, JavaScript, CAD

I. Introduction

In 2015, it was revealed that globally 36 million people suffer from blindness, not to mention the 217 million that are suffering with severe visual impairment. While the cases of visual impairment has been showing a decreasing trend over the last 30 years, the challenges are more prominent than ever due to the growing and aging population [1]. Despite the growing need of assistive technology, most of them today are accessible through auditory features, failing to provide sufficient help in loud public settings, as well as to provide a complete command over written language.

The current standard of reading for the blind is through Braille, a patterned cell of 6 protruded dots representing each alphabetic character. Yet, public schools including those for blind students have few teachers who are able to read braille. Such circumstances explain the low literacy rate among the visually impaired, merely capping at a low 10% [2].

With the continuous technological breakthroughs over the recent years, there has been numerous devices in the market designed to combat the literacy issue for the blind such as portable braille displays. However, these products continue to be inaccessible to the general blind population due to their high prices averaging around $5000. Aside from the pricing, there are also battery life and portability problems as the large number of intricate solenoids and actuators in them require a large amount of energy.

To address these short comings, Brailliant proposes an affordable, highly portable, and low energy consuming braille pad with a built in text to braille translator which allows the user to easily learn braille and read texts at their own speed. A key component in Brailliant is its innovative use of sliders and stepper motors to replace pre-existing solenoid/actuator architecture which reduces the pricing as well as the energy consumption of displaying braille patterns. Thus, it is Brailliant's goal to provide a highly accessible way to learn and read braille anywhere, at any time.

II. Use-Case Requirements

Our product aims to provide a learning tool for the visually impaired that can help bridge this gap in braille literacy, at an accessible price. As such, the device must be able to dynamically refresh an array of braille pins to sequentially display words from any text input. It must be portable, easy-to-use for visually impaired users, and have battery capacity to last at least a day of learning. These use-case requirements were developed with our users–visually impaired students–especially in mind.

Regarding the physical device, we identified a form factor of 30cm x 20cm, resembling that of an e-reader, for the device to accordingly function as a portable and accessible educational device. The device should be able to display braille one cell at a time (where each cell is 3pins x 2pins) to represent English word inputs. Finally, since the product should function as a reader, we identified that an acceptable learning reading speed is >10wpm. This is based on the average beginner braille reading speed of 6 wpm [5]. We aim to be able to actuate patterns of words quickly enough to achieve this speed.

Finally, the device must have greater than 95% accuracy for our text-to-braille (T2B) algorithm, including error handling for any unrecognizable characters in the text. After we produce the pattern encodings, the device must have greater than 90% accuracy in physically actuating the correct braille pin patterns. This ensures that our learning device does not become an inhibitor with misinformation. The battery life should last a day of learning, so we aim for 6 hours running time. This ensures that the device is a viable learning tool with our users in mind.

III. Architecture and/or Principle of Operation

Figure 1 outlines our full system diagram. Our device can be divided into two major components: the hardware braille pattern actuator array, and an accompanying software for user customization of the array. The braille pattern actuator will be enclosed in a 3D printed 12" x 8" box. As illustrated by the CAD rendering in Figure 2a, we split up each individual 3x2 braille cell by column, and the resulting three braille dots can be actuated at the same time by a slider. The curvature changes along the slider allows it to push a different subset of the dot pins upwards at different given discrete locations. A detailed design of the slider is shown in Figure 2b, and since there are only eight unique configurations for a group of three dots, the reader can verify for themselves that this design indeed can

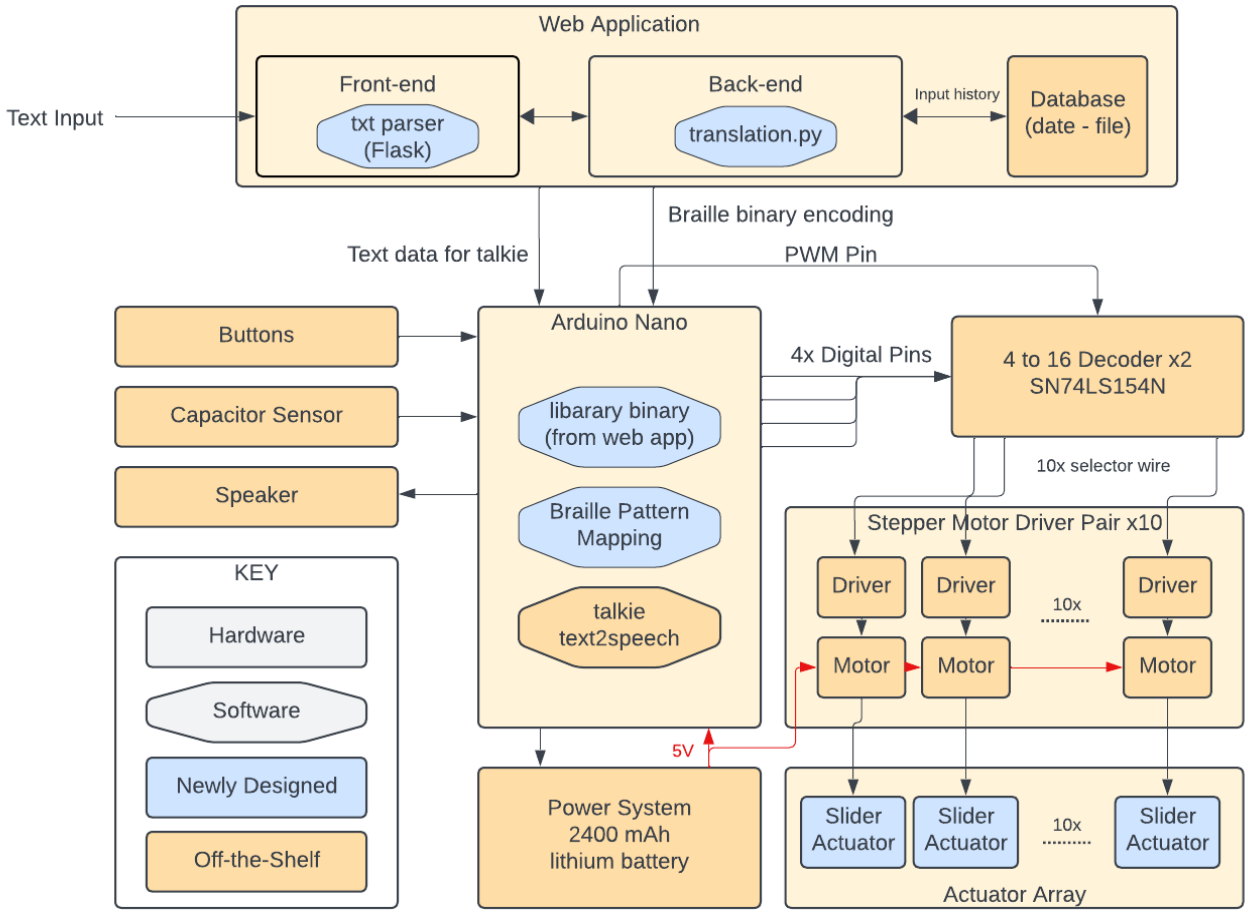


Figure 1: System Block Diagram

cover all of them. The linear actuation of the slider is achieved by driving a stepper motor through a gear and rack mechanism. Further, a single braille cell can be formed by joining two of these sliding actuators face-to-face, and an alternating motor mounting position is designed to allow a linear layout of ten of these braille cells to be mounted inside the box enclosure, as shown in Figure 2c. This actuation scheme is inspired by previous project by Ulmas Zoirov [6]. Contrary to most off-the-shelf solutions, where each braille dot is actuated by a separate piezo or solenoid component, this design is much cheaper to manufacture, while retaining reasonable actuation speed as outlined in section II.

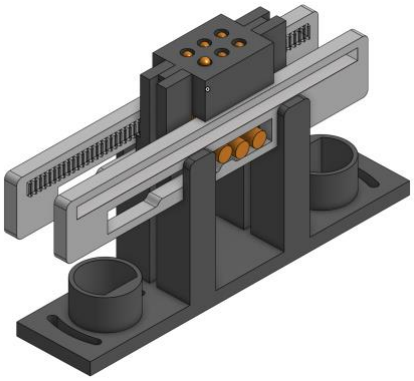
Each motor is controlled by a driver, which in turn receives its signal from a PWM pin from the Arduino Nano controller. The Arduino will only control one pair of drivers that correspond to a single braille cell at any given time. The PWM signal is sent to two 4-to-16 decoders, which reroute the signal to one of ten driver pairs. Four digital pins of the Arduino also connect to the decoder as selector wires.

The mechanical design of the actuator is conducted using the online computer aided design (CAD) software OnShape, and parameterized modeling technique is used to ensure the CAD model can be flexibly adjusted so that we can test various dimensions of the braille cell. Agile development principle is also utilized to rapidly fabricate the design using a 3D printer, so that we can quickly test the gap between the ideal design space, and the actual manufacturing deviation.

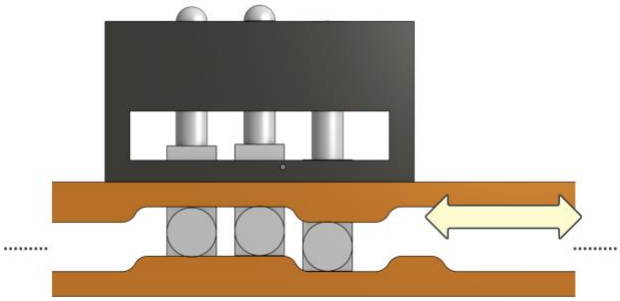
On the software side, our translation process and text input will be handled by the web app. While the initial proposal was to drop a .txt file to be parsed by the web app, we concluded use a different method to increase accessibility and usability. Now with a simpler prompt input method, the user could simply type in a word on a prompt to be immediately added to the pad's data storage.

The web app will contain a hash map of character key to braille pattern value where each braille cell column is a 4-bit binary value encoding with each bit marking a pin actuation among the 3 pins located on the braille cell column (Figure 3). The one remaining bit would be used as an end-of-word marker for the Arduino to pick up when to stop actuating. Once this is done the encoded data would be sent to the Arduino via a serial port connecting the web accessing device and the braille pad.

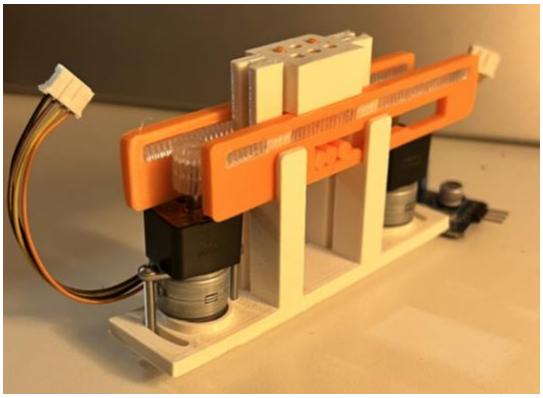
The web app will also be connected to its web-app database containing all the inputted words. All words on the database would be displayed on the web-app such that the user be aware of the sequence of words and delete inputs or alter the sequence. The words will be sent to the Arduino once a button is pressed on the web-app.



(a)



(b)



(C)

Fig 2. Braille pin actuation mechanic. (a) Top view of design. (b) Side-view of design. (C) Manufactured actuator

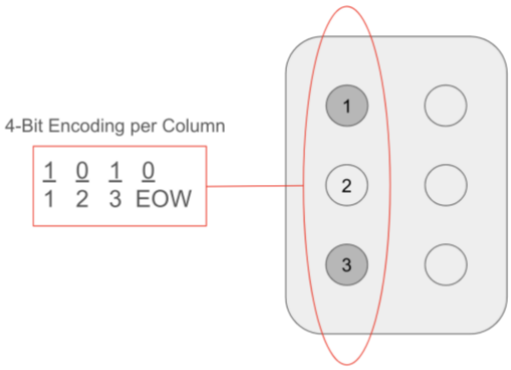


Fig 3. Braille pin encoding scheme.

The braille pad will handle the incoming data through its Arduino where it will step through encoded data to map each binary data to a stepper motor activation until the end-of-word bit is read. Furthermore, a next word indicating button could be easily implemented to the hardware pad to read the next word received until the next end-of-word marker is read.

IV. Design Requirements

In order to satisfy the above use-case requirements and considering our working implementation plan, we have identified the following technical design requirements.

To achieve the identified use-case requirement for battery life, we need to consider the power consumption of our entire system. The micro stepper motors we have selected consume 0.6W and one motor per column per braille cell. As a result, the 20 motors running for 6 hours will at most require the following:

$$\frac{0.6W}{motor} \times 20\,motor = 12W$$
$$12W \times 6\,hr = 72\,W \cdot h = 14400mAh$$

Thus, the above equation shows us that our device must have a rechargeable battery capacity of at least 14400 mAh in order to operate the required number of motors for the required length of time.

Next, our device's usability and accessibility as a learning device depends greatly on the physical layout. Specifically, each braille pin must adhere to standard braille sizing requirements for physical braille. According to the Braille Authority of North America, the distance between two braille dots within the same cell may be between 2-3mm [4]. Further, the distance between corresponding cells can be between 6-8mm. The layout of our 3D-printed braille cells and sliders must adhere to these spacing requirements. Finally, each braille pin should protrude 1mm up from our device's "screen" in order to be properly recognized by our users' hands.

The accuracy requirements beget functionality to reset and correct any mechanical jams we are bound to experience given the small scale and mechanical stress. As such, the stepper motors may lose tracking, especially considering that stepper motors do not have a feedback system to control and remember position. The device must recognize and re-try pattern actuation in these situations.

The beginner reading speed of 12wpm requires our design to be able to actuate the correct pins for any given word quickly enough to be able to display words on the device to users at this speed.

$$\frac{12\,wpm}{60s} = 0.2\,word/s = 5s/word$$

Thus, the derivation above shows us the timing requirements for our motor-slider actuation subsystem is 5s per word; in other words, the design requirements is 0.5s per cell. This signifies that each motor should spin to the correct position within this amount of time.

V. Design Trade Studies

A. Software Platform

The choice of Arduino as the hardware/software electronic platform was made in consideration of the low power usage, and low price range for the overall product. Table 1 details the difference between the two products. In terms of its alternatives, a valid alternative would be a Raspberry Pi. An advantage of using an RPi would be it's higher processing power (1.6GHz vs. Arduino's 16MHz). However, the processing speed only proves to be beneficial in the case of complex software computations which would not be necessary as Brailliant's translation process would be done on the web app prior to inputting the text data into the hardware device. In other words, our project deals with sending the predetermined instructions to the motors and hardware components to the device through the application and thus would not require the processing power and flexibility that RPi provides. Furthermore, considering the portability of the product, RPi's power usage stands around 1.7W which is comparable to Arduino's 0.14W which much suitable to our battery driven product.

Table I. Comparison of Board Specs

Table Head	Type of Board	
	Arduino Uno	***Raspberry Pi***
CPU Type	8-bit Microcontroller	64-bit Microcontroller
Operating System	None	Linux
Storage	32KB Flash	Depends on SD Card
Memory	2KB	1GB RAM
GPU	None	Built-in
Networking	None	Wifi, Bluetooth, Ethernet
Price	$20-30	Up to $80
Power Consumption	Less than 0.25W	Up to 1.8 W

B. User Interface

The trade-offs that we considered when choosing the user interface included the choice between a web-app, mobile application, and a design where all the data management and translation are all incorporated into the braille display device. Ultimately, we decided a mobile application would not be suitable due to the higher accessibility of web-based applications. Additionally, we decided to separate the user interface into the physical implementation of the device and the web application. To step through the words in the text file, we decided to use a button on the actual device considering the fact that the device is to be portable and to be used in outside environment. For inputting the list of words, we decided to use the web app since the texts are likely to be added by a third party, and as a result we needed to incorporate the input process using the most widely accessible method of interface which is the web. Furthermore, by using a web application we could handle the translation process and dictionary data within the web prior to transferring the data to the device which could alleviate the memory usage and processing done in the Arduino. This is more suitable for the project since it would be unreasonable for the Arduino's limited 2KB RAM and 32KB flash to hold the translation algorithm as well as the dictionary data which is to be accessed continuously throughout the translation process.

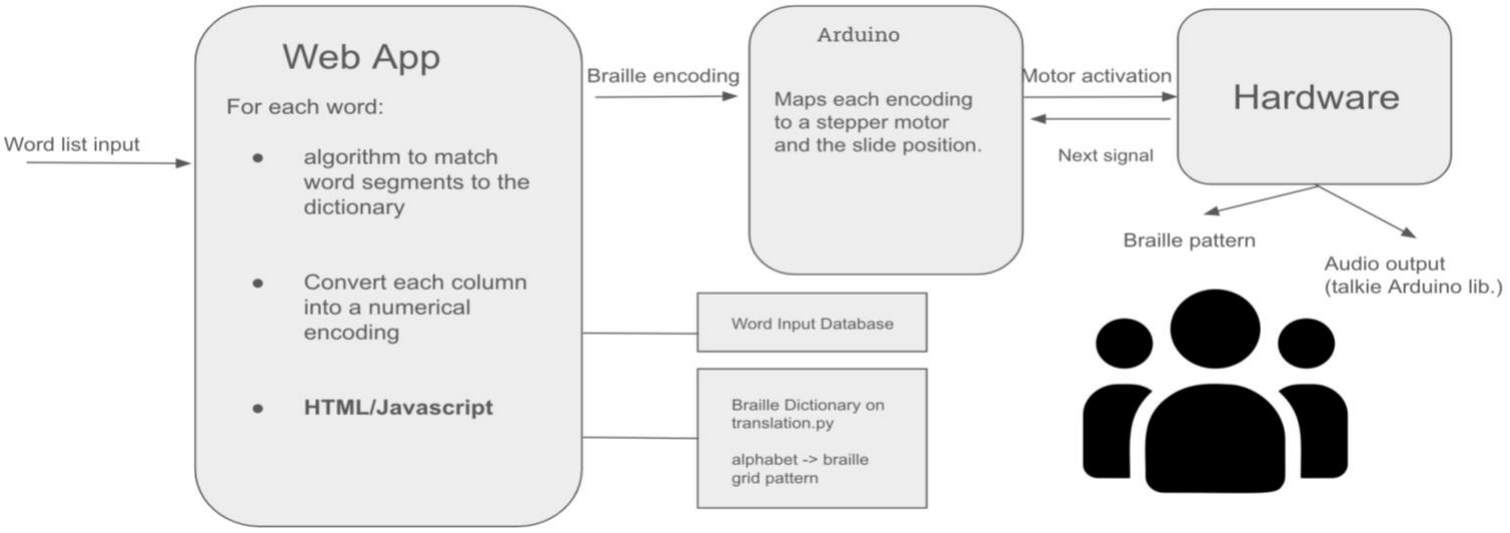
C. Translation Algorithm

Pre-existing translation algorithms include a basic hash map dictionary based algorithm and a machine learning classification algorithm. While contemporary ML is extremely powerful, we decided that it would not be necessary as there is no variability in the translation for braille and no grammatical changes depending on the context of the word. Thus, we decided to use hash maps organized in key-value pairs in C++. Most braille translation algorithm in the market uses one map that maps each alphabetic pattern to a braille arrangement. However, in our project we decided to use two maps (one for whole words, and one for prefixes, suffixes, and braille contractible alphabet patterns such as "aa", "ment", or "pre". The difference in the number of dictionary maps used creates a large difference in the time complexity of the algorithm. With each access of the dictionary being O(1), the usage of one hash map would lead to a complexity of O(n^2) as for each of the examined characters, the rest of the characters need to be checked for a match in the dictionary. On the other hand, the two map dictionary model we are using take advantage of the fact that all contractible character sequences are capped at a length of 4, and the rest are whole word abbreviations. This leads to a time complexity of O(4n) since each character needs to be checked with its 4 following characters at most to determine their contractibility.

D. Actuation Method

The choice to utilize micro stepper motors to actuate braille pins was developed based on our requirements for size, speed, and cost. We first considered more straightforward actuation with solenoids, planning to lay out an array of solenoids with one solenoid per braille pin. However, this led to significant design challenges as the available solenoid components are not nearly small enough to fit our sizing design requirements. Furthermore, we found out that solenoids require power to remain in the actuated position. This led to significant power consumption concerns since our requirements are to actuate each pin and remain in this state until the user has finished reading the entire word. We also considered piezoelectric actuators, however this led to cost challenges as we required a great number of actuators and the cost per part exceeded our budget.

We finally reached a solution with an entirely distinct approach to actuating pins—actuate patterns of pins using a 3D printed sliding component and stepper motors. Each slider with engravings will actuate all 8 different patterns that each column



of 3 pins can achieve. This design has significant advantages over solenoids or piezoelectric actuators due to the low power consumption, smaller component size, and reduced number of components. We were able to find small stepper motors that operated at 0.5W-1.2W. Further, this reduced the number of "actuators" by a factor of three—2 motors per cell compared to the previous 6 actuators per cell.

VI. System Implementation

We will start the section by discussing fabrication method. This section is then split up by hardware and software implementation. The hardware portion is further split into actuator mechanics (motor, driver, and decoders), and miscellaneous (power system, speaker, capacitor sensor, etc.).

A. Fabrication

We used off-the-shelf products for all electronic components. Critically, we will fabricate the device enclosing and all individual slider actuators ourselves using fused deposition modeling (FDM) 3D printers. This choice is primarily motivated by our design goal of making this device as easy to manufacture by DIYers in a garage as possible, and other fabrication equipment, like laser cutters and CNC routers, are not as readily available to the public as 3D printers. The printer of our choice is ANYCUBIC MEGA-S since this printer is widely available on the secondhand market for very cheap prices. The printer settings, however, may be altered for manufacturing on different printers.

During our fabrication and testing process, we discovered that 3D printing using PLA plastic cannot produce our miniature gear and rack components with our desired accuracy and endurance. The small tip on gears proves to be too compliant for relatively high torque use cases. Thus, this is the only part in our design that must use laser cutting as a workaround. Laser cut gear can provide the additional benefit of reducing jamming, since the laser beam will slightly melt away and shrink the design dimensions, conveniently providing tolerance to the power delivery train.

B. Web-application

The web application will be built using JavaScript, and HTML. Fig. 4 displays the role of the web app in consideration to the whole software system. The web application will be linked to a built-in SQL database to store the history of user input. Once the user clicks done, the translation.py python script would be run as a thread in the JavaScript front-end code and would translate each word to a Grade-2 standard braille, mapping each sequence of characters to a pattern from left to right.

The braille dictionary would be incorporated into two different key-value hash maps: one for whole word contractions and one for 4 letter max character sequence contractions which are the two types of word trimming used in grade 2 standard braille. The translation process starts from left to right with each letter and its 3 following letters being checked with the dictionary hash map to output a matching braille encoding. After the translation, the data would be transferred to the Arduino and the user inputted text data would be stored to the web database for reuse in the future. Fig. 5 depicts the final web-app with each step of the word input process.

C. Device Software Implementation

The programming for the braille display would be done in the Arduino IDE in C++. With the input from the web app, the Arduino would map each braille encoding to a stepper motor through a 4 to 16 decoder. The stepper motor was controlled via its built-in library. The velocity setting was set to max to increase hardware response speed and the steps needed to reach a certain pattern on the braille cell was defined. In cases of jams and motor stalling, the motor code would step left and right in small margins multiple times until no jamming is detected by the motor's built-in jam detector.

D. Microcontroller

The Arduino Uno microcontroller will be responsible for interfacing with the motor drivers, battery, and Python translation program. These other subsystems will connect to the

Arduino via its analog and digital pins. The Arduino code uploaded to the microcontroller.

E. Power

The motor drivers are supplied with 12V DC power and thus the power subsystem will be integrated onto our main breadboards. It is delivered from a 120V AC to 12V DC converter from a wall outlet. The Arduino and VIO (logic) voltage on motor drivers are powered by 5V via USB. The initial plan to have the system battery powered did not work as we found out that the stepper drivers worked much better at higher power.

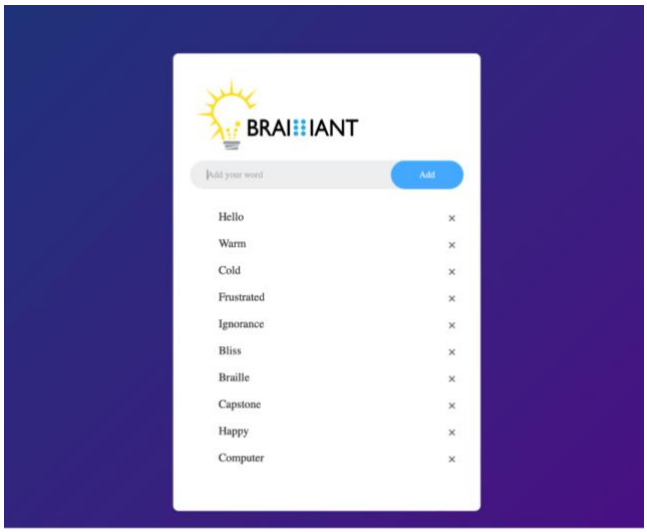


Fig. 5 Web-app implementation with system's software

VII. Test, Verification and Validation

Testing for Brailliant was mainly composed of testing the user-case requirement for correctness (braille display and translation algorithm), responsiveness, portability, and accessibility. These requirements meant a design requirement of a fast actuator response speed, accurate actuation, accurate text-to-braille translation, and an easy-to-use and accessible web-application. A summary of the test results, methods, and the passing parameters are displayed on Table 2.

A. Tests for Web Application

For web-app validation, we collected 5 subjects from CMU to perform user-testing on the difficulty of the text-inputting process, data transfer between the web-app and the braille display, retrieval of input history, and the basic end-to-end performance of the site. Each subjects was told to input at least 20 words to test the performance. To obtain a quantitative measure on overall difficulty, usability, and responsivity, the subjects were asked to give a rating of the web-app for each criteria in a scale of 1 to 10. As aforementioned in sections II and III, accessibility is a crucial requirement for this device, so user-testing was done extensively in order to take feedbacks from users to ensure our ease of navigation through the site. The first iteration of the test yielded an average score of 7.25 out of 10 and one of the biggest flaw to our web-app was text data input method which used to be implemented with a drag and drop of a .txt file and a .txt file parser. Such method, restricted the users to a single file format and created barriers to users unfamiliar with text editing. As a result, the user prompt method where words can be simply typed into the web-app was implemented and was able to obtain a score of 8.75 on the next iteration of user-testing.

B. Tests for Translation Algorithm

As mentioned in section IV, the correctness of our translation is crucial to our project. In order to allow the previously stated 100% accuracy in text to braille translation, we tested the translation algorithm implemented in our web-app with numerous sequences of words. To do this, we collected more than 100 words of varying length and complexity to achieve 100% accuracy. The 100 words were further categorized to 50 contractible words and 50 non-contractible words as lengthy words in braille are often contracted to shorter versions to reduce the space needed for display. To solely test the software before integration with the hardware, we formed a testing function which displayed the encoding to a visualized braille pattern to compare with the reference solution. The test results successfully validated the accuracy level of the translation algorithm, reaching the required 100%.

C. Tests for Hardware Pin Actuation

Other than the algorithm itself, it is also significant for the array of stepper motors to be accurate by itself. On all the possible patterns (8 patterns), we tested the accuracy of actuation. The user-case requirement for display accuracy translated to an accuracy larger than 80% for design requirement. The result showed an accuracy average slightly above 80% over multiple iterations with motor jamming due to 3D printer accuracy and motor power being the major obstacle toward reaching higher accuracy.

D. Tests for Actuator Response Speed

As stated in our use-case requirement, it is important for the Braille cells in our device to display each braille cell at the average braille learner reading speed which is at 12WPH (0.5 second per braille cell). Thus, we tested the actuation speed for each pattern through a slow-motion capture video, ensuring that the actuation of a pattern takes less than 0.5 seconds. At our first iteration of testing we achieved an average of 4 seconds for a pattern actuation, significantly exceeding our set limit. One of the biggest setback was the motor jamming resolution which needed a greater torque on the motor to resolve. Thus, we ran the test with different power sources for the motor driver (5V vs. 12V) exploring the tradeoff between motor speed and torque against power usage. The detailed result in shown on Fig. 6. In conclusion, an average of around 1 second was reached with a higher voltage power supply, marking an improvement from our previous iteration. However, further study on stepper motors is required to reach the desired spec on the design requirement.

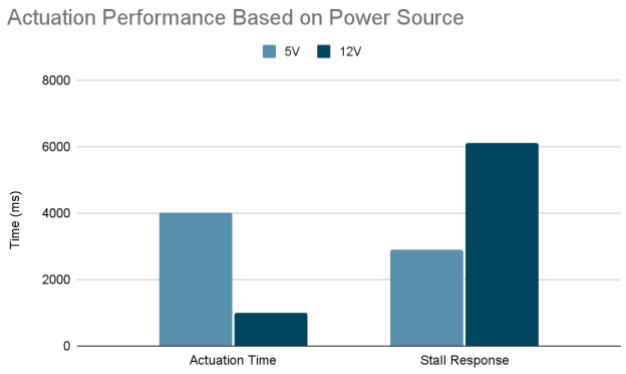


Fig. 6. Actuator response speed based on power source

E. *Tests for Portability*

The reason why most braille devices are struggling with implementation is due to the large amount of power consumption. In section II and IV we derived a necessary battery life of 6 hours for the use of our product during a normal school day. To test the battery consumption of our device we tested our device throughout the day, occasionally switching through different braille patterns. After going through more than 2 iterations our braille pad stayed operational more than 6 hours. Sizing of the 3D printed braille cell was also an important component to our design requirement which ties to our user-case requirement associated with portability. Our final product was able to stay 2 times the size of the unified braille standard which allowed our product to be stay within the 12 in. by 6 in. size limit we set for 10 braille cells.

TABLE 2. TEST AND VERIFICATION SUMMARY

Test	Testing Method	Requirement	Result
Actuator Response Speed	Record response time per column on 10 words	< 500ms actuation time	~1000ms average actuation time + 3s jamming resolution
Actuator Accuracy	Test grid on all possible braille pattern inputs	>80% accuracy	78% accuracy
Text-to-braille Algorithm Accuracy	Test on 100 words and compare with online translator	100% accuracy	100% accuracy
Web-app User-testing	User-test on 5 students (feedback on 1-10)	>8/10 average	8.75/10 average

VIII. PROJECT MANAGEMENT

A. *Schedule*

See Figure 7 at end of document. The schedule has changed from the design document to account for new tasks, abandoned tasks, and adjustments to testing and assembly over the last month. For example, we added laser cutting for the small gears and linear racks that we ended up designing due to lack of readily available parts.

B. *Team Member Responsibilities*

The project was divided into two primary components: Software and Hardware. Due to the mechanical challenges presented in the hardware implementations of the project, two members were allocated for hardware and one was assigned to deal with the software.

Ziyu was responsible for the CAD design of the project including the 3D printed sliders and their connection to the stepper motors. The pins and the casing design were also handled by Ziyu. He also assisted the schematic design handled by Samay to allow a smooth integration between the mechanical components and the electrical components.

Samay is primarily responsible for the schematic for the motors. He also dealt with the power management, microcontrollers, as well as the code for stepper motors. It was his primary goal to seamlessly connect the hardware components all together and transfer the necessary signals for a successful actuation of pins

Yujun's was responsible for the software implementation. Aside from handling the text-to-braille translation algorithm, he created encoding scheme for braille patterns to transfer all the necessary information while minimizing data usage. Furthermore, he designed the web-application to take user input in a user-friendly manner.

The testing for hardware was done by Ziyu and Samay, while software testing was handled by Yujun. Ziyu focused primarily on testing the precision of the 3D printer such that the printed product is accurate to the expected design.

C. *Bill of Materials and Budget*

See Table 3 at end of document.

IX. ETHICAL ISSUES

When it comes to ethical concerns, it is paramount to identify who the ideal users of our product is. In the case of Brailliant, it is the visually impaired who lack access to written texts. Since our users are a distinct group of people within the population, It's crucial to think of ethics within their perspective. Furthermore, when there is a failure in the technology it could create a bigger divide between these distinct group of people, namely the visually impaired, and those who are not as these groups of people would be solely affected by whatever impact our product causes.

A. *Public Health, Safety, and Welfare*

When giving the blind an affordable reading tool to use, it could increase their public health as they can gain more access to information, and not be separated from the general population. As stated by the National Library of Medicine, patients with low literacy turned out to have poorer health outcomes and uses of health resources [7]. Thus, our design allows to lower the health risks for the blind which is already higher for the blind due to their lack of visual awareness.

In terms of welfare, there is an obvious gap in the welfare between those with healthy vision and those who are visually impaired. Other than the fact that they lack one of the six senses, this also causes higher illiteracy rates which is hard to

counter as educational establishments often lack braille-reading teachers to teach these people the command over written language. Thus, a flaw in the accuracy of our product could possibly fail to affect such pre-existing gaps or in fact worsen it if users were to become reliant on the product. As a result, the accuracy requirement of the text-to-braille translation of our product was increased from 90% to 100%. However, there is still a need to increase the accuracy of the actuation technique with a possible integration of a higher-spec motor to ensure the users' welfare is improved as intended.

B. Cultural Concerns

One cultural issue that could arise from our system design is the possibility of a welfare gap between English speaking and non-English speaking people as our product only translates English. In other words, the product contains a cultural barrier to non-English-speaking users, barring them from accessing the product. To counter such effects, we decided to make our algorithm as open source as possible and be based on a map of the braille dictionary that is easily replaceable. Consequently, it is fairly simple to replace the dictionary map with another language dictionary and have the braille display work as intended mitigating the possible discrimination the product could engender in the market.

C. Economic and Social Concerns

As previously discussed, our product aims to increase the welfare and health of the visually impaired to close the gap between them and the visually abled. However, by doing so with a marketable product could lead to inequalities between more privileged group of people and the less privileged depending on the price of the product which could be a societal issue. To mitigate such effects, we designed the technology to be very affordable with the price more than 8 times cheaper than the typical price of braille displays on the market. Therefore, even though it is inevitable that economic concerns will follow any beneficial product unless it's free, our project still maintains to keep this inevitable harm as small as possible with an affordable price.

Other than economic barriers to accessing the product there could also be other societal traits coming into play. For instance, as Brailliant is heavily based on technological innovations to achieve its goals, the benefits could be taken away from those who are unfamiliar with technology, especially those who are unfamiliar with computer usage. Our initial design included an input method which utilized a .txt file input. This could've increased the gap between those who are familiar with technology and those who are not (elderly and the young) considering the fact that file formats are not commonly understood among regular people and file format changes are only supported in certain devices or through third party applications. Thus, we tried to mitigate creating such societal barriers by introducing a newly integrated design where inputs are easily and directly typed into the web-app through a prompt given on the screen instead.

D. Environmental Concerns

The environmental issue that arises from our Braille display is tied to our use of 3D printing. While 3D printing offers countless benefits including rapid prototyping, customization, and cost reduction from decentralized manufacturing, it also has environmental consequences. Traditional 3D printing such as the method our project used involves plastic filaments. As our product is largely made of these filaments, it contributes to plastic waste if not managed properly. Considering the fact that plastic waste is the main perpetrator of water pollution and corresponding habitat loss for marine animals, it is crucial to ensure that environmentally friendly materials or at least the waste is managed properly. While these possible solutions were not within the scope of our project, with further research and development of our product we could counter the possibility of pollution with the use of bioplastics, or recycled plastic filaments.

X. Related Work

Our product seeks to be able to dynamically display braille words, which has and remains a challenge. Other products that currently exist include braille readers such as: . They have the capability to produce braille translations of inputted words and include many physical buttons to interface with the device. However, these products are much larger in form factor and 5-10 times as expensive as our product aims to be.

The product DotPad is more aligned with our proposed device with its own custom electromagnetic actuator technology. This product actuates each braille dot on a large tablet-sized "screen" (array of dots). However, this product is designed specifically to convert non-text media such as images and charts into a physical pattern for the visually impaired.

There are also numerous research projects that have developed innovative piezoelectric materials or other proprietary MEMS based actuators.

XI. Summary

Our goal with the Brailliant product is to be able to address the incredibly concerning low literacy rates in braille among the visually impaired. We want to provide a device that is universally able to take any text input and both produce the braille patterns and provide audio pronunciation output. This will be achieved with our custom sliding actuation method. There remain challenges in physically organizing the motors and corresponding motor controllers within the 3D printed case. Another area of concern that must be addressed is the reliability, especially given the reliance on mechanical moving parts of our design.

Glossary of Acronyms

CAD – Computed Aided Design
DIY – Do It Yourself
MQTT – Message Queuing Telemetry Transport
OBD – On-Board Diagnostics
PCB – Printed Circuit Board
PWM – Pulse Width Modulation
RPi – Raspberry Pi

WPM – Words Per Minute

REFERENCES

[1] Ackland, Peter, et al. “World Blindness and Visual Impairment: Despite Many Successes, the Problem Is Growing.” *Community Eye Health*, U.S. National Library of Medicine, 2017, www.ncbi.nlm.nih.gov/pmc/articles/PMC5820628/

[2] Amato, Sheila. “Standards for Competence in Braille Literacy Skills in Teacher Preparation Programs.” Journal of Visual Impairment & Blindness 96, no. 3 (2002): 143-153.

[3] “Human Interface Guidelines.” Apple Developer Documentation, developer.apple.com/design/human-interface-guidelines/. Accessed 12 Oct. 2023.

[4] National Library Service for the Blind and Physically Handicapped, Library of Congress. Specification 800: Braille Books and Pamphlets. (www.loc.gov/nls/specs/800_march5_2008.pdf)

[5] Perkins School for the Blind, *Perkins School for the Blind: Reading Rates*, Accessed on Feb 24, 2024, [Online]. Available: https://www.perkins.org/resource/reading-rates/

[6] Zoirov, Ulmas. Development of the low-cost solution for Braille display based on linear actuators with 3D printed mechanisms and servomotors. Rel. Marcello Chiaberge. Politecnico di Torino, Corso di laurea magistrale in Mechatronic Engineering, 2021

[7] https://www.nnlm.gov/guides/intro-health-literacy

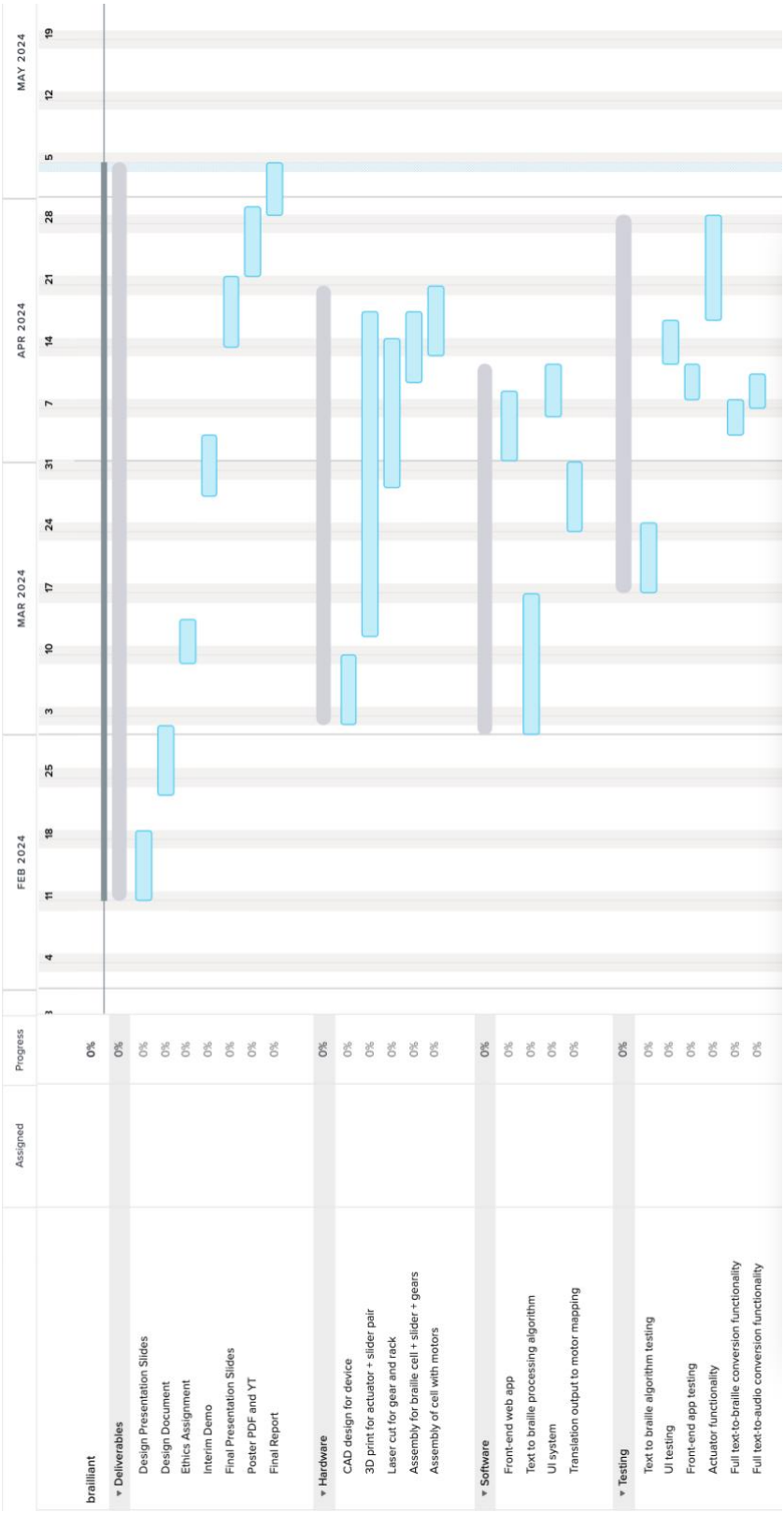


Figure 3. Schedule

Description	Model Number	Manufacturer	Quantity	Unit Cost	Cost
Micro Stepper Motors	N/A	Abovehill	2	$10.99	$21.98
Micro Stepper Motors	N/A	QINIZX	6	$12.59	$75.54
PLA Filament	N/A	ELEGOO	2	$17.99	$35.98
Screws	N/A	uxcell	1	$7.79	$7.79
JST Connectors	N/A	N/A	1	$15.00	$15.00
					$156.29

Table 3: BOM

Fig 8. Final Web-app design with user inputs