

CS/CE Senior Design Specification Report

Text to Braille Device

Kairui Hu | Christopher Jaramillo | Christopher Rios

Faculty Advisor Professor

Oren Segal

Fall 2023

December 28th, 2023

Abstract

Braille is a very important language, yet almost all the tools available for braille users are fairly dated and very expensive. Although in the modern day the necessity of braille seems to be low due to technology such as screen readers, braille displays, and speech recognition software, it is still a valuable language to learn. This is even more apparent in public schools, where the option of even learning Braille is rare if not nonexistent. This project aims to address these challenges by developing an accessible and cost-effective Braille Board coupled with an interactive application tailored for elementary, middle, and high school students. The design approach integrates the ESP32-S3 as the central component facilitating connectivity between the application and the Braille Board, utilizing the PCA9685 16 Channel 12-Bit PWM Servo Motor Driver to enable movement in Braille cells, and employing an HTML-based interface for the application. The prototype successfully demonstrates the functionality of the Braille Board and its seamless interaction with the application, accurately representing inputted letters through Braille cells. The affordability of the created board highlights its potential for widespread use, while the prototype's success indicates avenues for further enhancements. In conclusion, this project presents an effective solution for public schools, offering an accessible Braille learning tool that caters to both visually impaired individuals and those eager to learn Braille, emphasizing the viability and potential for broader implementation in educational environments."

Chapter 1: Project Description and Goals

1.1 Project Scope and Vision

There are many tools to assist visually impaired people with how to read braille, however those tools come with a mighty price tag. In order to teach blind students to read braille affordably, there must be a cheaper alternative. The concept is to make both an application that will take an input from a phone or website and a physical medium that will output the text into a physically touchable and readable version as this is a physically interactive language. Braille is the one of the most important methods that the blind have in navigating everyday life: from bathroom signs, restaurant menus, and in European Union countries it is mandatory for pharmaceutical packaging to include braille [1]. In terms of literacy and education, braille is essential for those with vision loss because of the Individuals with Disabilities Education Act, which states “In the case of a child who is blind or visually impaired, provide for instruction in braille and the use of braille unless the IEP Team determines, after an evaluation of the child’s reading and writing skills, needs, and appropriate reading and writing media (including an evaluation of the child’s future needs for instruction in braille or the use of braille), that instruction in braille or the use of braille is not appropriate for the child” [2]. This states that unless the IEP team determines otherwise, any child who is visually impaired or blind must be taught braille.

Braille is a very important language, yet almost all the tools available for braille users are fairly dated and very expensive. Although in the modern day the necessity of braille seems to be low due to technology such as screen readers, braille displays, and speech recognition software, it is still a valuable code to learn. Non-profit organizations such as The National Federation of The Blind Jernigan Institution, National policies such as the Individuals with Disabilities Education Act, and modern technology such as braille embossers have made braille more accessible, affordable, and acceptable to everyone who wants or needs to learn braille.

One important question that one might ask is, what other options do children with visual impairment have in terms of Literacy and education? The answer, if they are still visually enabled: enlarged print, a change in the standard print such as an increase in contrast, the use of optical devices to enhance the visibility of a document, are some options to help allow these children to have the same advantages that children with normal vision have in terms of reading books or documents. However, this does not mean that the children should not learn braille, as stated in Defining Literacy for Individuals Who are Blind or with Visual Impairments: A Qualitative Study of Stakeholders “There were students who had low vision and because of their etiology were gradually losing their

vision to the point of being legally blind but were still encouraged to use their vision at that point in time instead of being taught braille. This led to the 1990s low rates of literacy and many literacy barriers (Koenig & Holbrook, 2000)” [4]. On the other hand, if their residual vision is very low and standard print is not visible regardless of magnification or any other enhancements in visuality, then only braille or auditory materials are left as a choice for literacy.

Our Goal is to make a Braille Board and accompanying application for elementary, middle, and even high schools to better teach Braille to those who are blind as well as those who are interested in learning braille.

Society

By making Braille technology more accessible to public schools we hope to create a environment that encourages blind and visually impaired students to enjoy learning braille, and in doing so we hope to lower the 70% of unemployment rate in the blind community as 85% of employed blind individuals can read and write in braille, but currently only 10% of blind children even learn braille and this is mostly due to the inaccessibility of braille in public schools, and we hope to change this.

We chose to design the hardware to use easy to obtain cheap replaceable parts because we want the user to be able to easily fix any problems they might have by simply replacing the parts instead of purchasing a new machine. We chose servos as the mechanism for the vertical movements of the device because of the 16 channel 12 bit servo motor driver board which allows us to easily link multiple servos and servo drivers together and control their movements which is what we want for the braille board, we did attempt to use stepper motors, however the miniature stepper motors we purchased would heat up to high temperatures with long term exposure which is not what we want for a braille board what would be used for hours at a time. We 3D modeled and printed the base and case of the Braille Board as well as the small parts used to translate the circular motion of the servos to vertical movement, this is so that we can play around with the size and shape of the case and base and so that we have clear measurements of the entire hardware. If we were to simply use and cut the cover and base physically it would be difficult to see if there were any issues with redesigning the entire contraption if the parts we made were not of the right size, but by modeling it first we can minimize this issue. And lastly we chose to use WIFI instead of Bluetooth because it would be simpler for our target audience, schools, to connect the device to the wifi than it would be for bluetooth and wifi has a longer range which allows for more portability.

Results

The result of this is that the project came out as we planned, the servo and servo driver allowed us to manipulate the “Braille Dots” exactly as we wanted it to, the 3D model allowed us to test with the base and cover as we wanted, making putting the entire system together very simple. The only issue we had was with the power supply, but this issue can be solved by changing the current power supply

to one with a different current or voltage. But, all in all, our device does what is intended.

Chapter 2: Project Goals

In this chapter, we'll outline the primary project goals through a bullet-point presentation. These goals will serve as the guiding principles driving the creation and development of both the Braille board and accompanying application.

1. Build an application that will take an input in English, translate it into braille, display and communicate the translation to the microprocessor in the device.
2. Establish a wireless connection from the braille device to the device hosting the application via Wi-Fi.
3. Use a servo hub to connect the many servos needed to the microprocessor so it may send signals to protrude the necessary studs displaying the braille translation.
4. Create a battery powered source controlled by a switch in order to provide power to the system.
5. Wire two buttons to cycle between the next and previous braille letters.
6. Create custom housing that is efficient in containing all required components, visually appealing to the consumer, and portable.

Chapter 3: Requirements Specification

In designing any system, consideration for the entire system is crucial. This chapter navigates through the Requirement Specification of said system, delving into the intricacies of the system perspective, user stories, external interfaces, as well as functional and nonfunctional requirements. By creating a system perspective, we gain a comprehensive understanding of the system's scope, its user's needs, and the interaction with external entities. User stories illustrate the user-centric aspect of the system, detailing specific user interactions and expectations for said system, while the external interface details the connections between our system with external elements. Functional requirements discuss the specific functionalities and operations the system must perform, shaping its core capabilities. Simultaneously, nonfunctional requirements emphasize the system's qualities, including performance, security, and usability, outlining the criteria that governs its effectiveness beyond functionality.

3.1 System Perspective

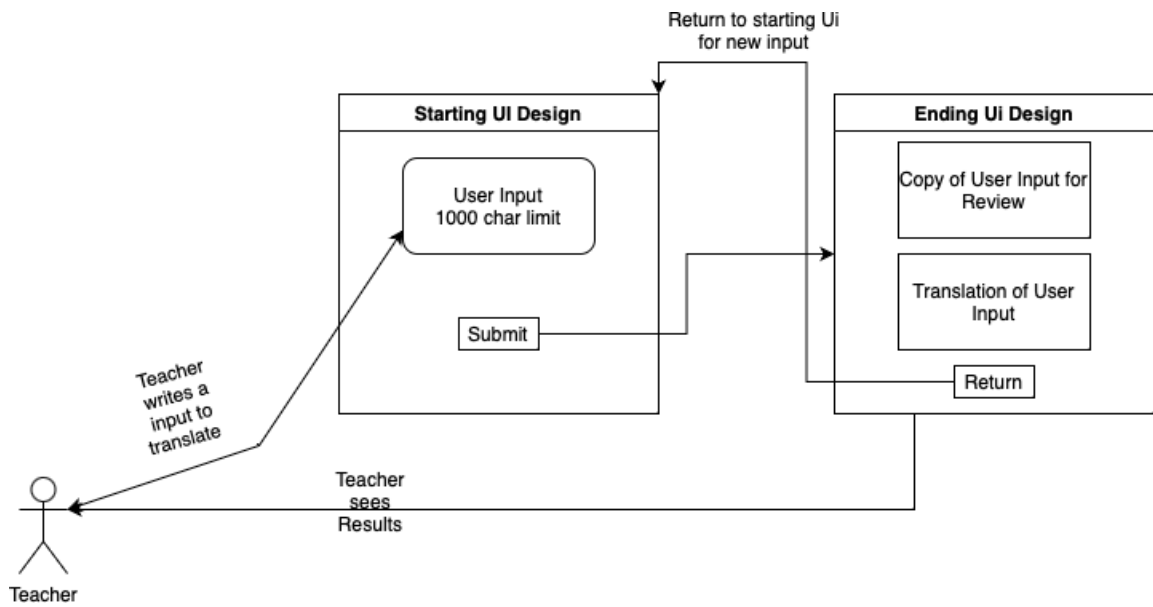


Figure 1: System design on the software side

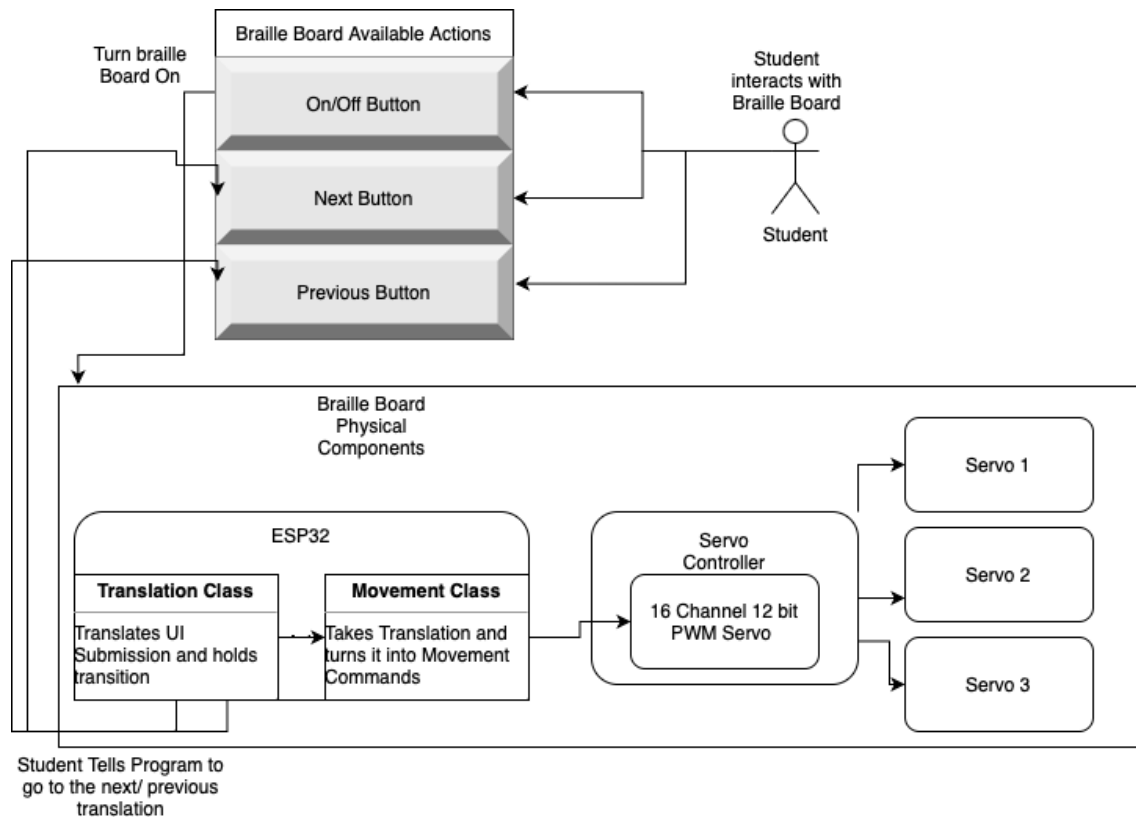


Figure 2: System design on the hardware side

Figure 1 is a depiction of the software element of the user experience that will often be used by the person who is teaching braille in order to test the learner who is on the hardware device. However, if there is a solo user, they will be using both aspects. Figure 2 is usually used by the student who is learning braille, as they will be tasked with using their finger to feel the protruding studs on the device.

The first step in using this system is to turn on the device which will activate the on-chip Wi-Fi. The user will then be prompted to establish a wireless connection or choose to straight hardwire into the device. Once a connection is established the application will prompt the user to input a prompt that they wish to be translated in English, up to 1000 characters. Upon entering the text and clicking on the submit button, the program will do the translation, display the original prompt, the braille translation, as well as sending the braille translation to the ESP32 chip in the device. The chip will then take that information and translate each character upon selection to the servo hub to communicate which stud needs to be raised. The user on the device side will then have the option to cycle between the prompt 2 characters at a time, each with real time translation.

3.2 User Stories

Name	US-1 Device
Description	Set Up Device For Initial Use
Rationale	To Start Up Device
User	Teacher/Student
Preconditions	The system is sitting idle, unpaired from any external computer.
Basic Course of Action	<ol style="list-style-type: none"> 1. Push button to initialize Bluetooth pairing 2. Device pairs with computer
Post Conditions	The device is paired via Bluetooth to a computer and the user can now use the device. LED light on device should be on to indicate device is working correctly

Table 1: User Story 1

Name	US- 2 Making A Translation
Description	Teacher can translate any English statement of length 1- 1000 characters into braille
Rationale	To translate teacher input into braille
Users	Teacher
Precondition	The System is turned on and successfully paired with the device
Basic Course of Action	<ol style="list-style-type: none"> 1. Teacher will open application (see Webpage Start slide to see sample of application) 2. Teacher will (in the available textbox) insert their desired input 3. Teacher will press start button (as seen on Webpage Start sample)
Post Conditions	The teacher's input is now in the system and the translation program will be able to begin translating. The Translations will be sent to the device for the student to use

Table 2: User Story 2

Name	US-3: Next button/previous button
Description	The student will view the next character of their input
Rational	The device can only translate 2 characters at a time, how will the student be able to view the next/previous character.
User	Student
Preconditions	The teacher has already written and submitted their input
Basic Course of Action	1) On the device there are two buttons on the front face the student can click on the left to go to the previous character, or the right to go to the next character
Post Condition	The device will read the next/previous character of the input. (If all the input is read the program will not loop to the start of the input when the next button is pressed and instead nothing will happen if the button is pressed)

Table 3: User Story 3

3.3 External Interfaces

3.3.1 User Interfaces

UI Name & Number	Short Description	Reference User Stories
UI-1 Wi-Fi	Connecting device to app host	US-1
UI-2 Translation	Webpage for receiving input	US-2, US-3
UI-3 Result	Webpage for outputting result	US-2, US-3
UI-4 Next/Previous	Transition from one character to the next	US-3

Table 4: User Interfaces

3.3.2 Inputs and Outputs

Input # and name	Source	Description	Units of measure	Data format	References to user stories, etc.

WEB1 – textbox	Textbox	The text box will hold user keyboard inputs	Text length	Text	US-2 & US-3, UI- 2 & UI-3
WEB2 – submit button	Button	The button will submit the textbox inputs to the program	N/A	button	US-2 & US-3, UI- 2 & UI-3
WEB3 - end process	Button	The button will push website back to user input screen	N/A	Button	UI-3
PHYB1- On/Off	Physical button	Will turn on or off the device	N/A	N/A	UI1
PHYB2 - Next Button	Physical Button	Will communicate with braille program to read next part of translation	N/A	N/A	UI1
PHYB3 – Previous Button	Physical Button	Will communicate with braille program to read previous part of translation	N/A	N/A	UI1
PHYB4 – Wi-Fi	Physical Button	Will tell device to allow Wi-Fi connection	N/A	N/A	UI1

Table 5: Description of Inputs

Output # and name	Destination	Valid range, accuracy, and/or tolerance	Units of measure	Data formats	References to user stories, etc.
WEB1OUT-textbox	User view	1 char to 1000 char	# char	text	US-2 & US3, UI-2 & UI-3
WEB2OUT-submit button	Braille translation program	Textbox valid	N/A	N/A	US-2 & US3, UI-2 & UI-3
B1O - On/Off	Led + powers device (led is for us to know that the device is on or off)	Buton must be pressed down past a specific point (to make sure it's an intentional press)	N/a	N/a	Design proposal figure 2

B2O- Next buton	Communicate with braille translation program to move to next part	Buton must be pressed down past a specific point (to make sure it's an intentional press)	N/a	N/a	Design proposal figure 2
B3O- previous	Communicate with braille translation	Button must be pressed down	N/a	N/a	Design proposal

Table 6: Description of Outputs

3.4 Functional Requirements

Name	FR-1 Starting application/connecting to device
Description	How the device turns on and connects to app
Rationale	The board will not function if not initialized and connected
System behavior	Switching the on position provides power to the device, which will start searching for a signal to connect to. In the application the device will be connected via Wi-Fi.
Cross-Reference	US-1

Table 7: Functional Requirement 1

Name	FR-2 Displaying Correct Translation
Description	After desired text is inputted to be translated, the studs on the device will protrude in the corresponding spots
Rationale	The main purpose of the device is to teach braille, it is essential to its functionality.
System behavior	The system will send signals to the servos activating the corresponding studs to display the correct translated letter.
Cross-Reference	US-2

Table 8: Functional Requirement 2

Name	FR-3 Next/Previous Buttons
Description	Buttons to be pressed to cycle through the characters on the input in braille format every two letters
Rationale	Without these buttons only two characters can be shown at a time where new input would have to be typed every two characters.
System behavior	Both buttons are connected to input terminals on the ESP32 microcontroller where every time an input is detected, it prompts the controller to send signals for the following or previous two characters in the English input to be outputted with their appropriate studs.
Cross-Reference	US-3

Table 9: Functional Requirement 3

3.5 Non-Functional Requirements

Name	FR-2 Servicing
Description	After a certain amount of usage, the device is bound to run out of battery which will need to be replaced/recharged.
Rationale	Without power the device will not function.
System behavior	Because of the portability factor, battery power is ideal for most situations. By utilizing either a screw system or pulling a plastic tab, the batteries may be switched out.
Cross-Reference	US-2

Table 10: Non-functional Requirement 1

Name	FR-4 Housing
Description	The motherboard and components are sealed inside a housing to protect its contents
Rationale	To protect its internals from user damage and appeal to the consumer with simplification
System behavior	A housing gets created, with methods such as 3D printing, to house the device's vitals as well as allowing all inputs and outputs to be visible and accessible. Loose wires and components have greater risk of damage from the user.
Cross-Reference	US-4

Table 11: Non-functional Requirement 2

Chapter 4: Engineering Standards and Constraints

Within system design, adherence to engineering standards and recognition of constraints are critical in shaping, developing, and ensuring the systems reliability and safety. This chapter will illuminate the core importance of engineering standards and constraints in shaping systems that not only meet but exceed user expectations while ensuring safety measures are integrated seamlessly. Throughout this chapter, we'll meticulously document the precise engineering standards and constraints adopted, offering both a detailed specification and a concise overview of each standard or constraint.

4.1 Engineering Standards

4.1.1 Bluetooth

- Bluetooth offers a vector of communication for the user to the device through a web interface or downloadable program.

4.1.2 Wi-Fi

- Wi-Fi offers a vector of communication for the user to the device through a web interface or downloadable program.

4.1.3 Size and Spacing of Braille Characters

- The actuating Braille components of the device will follow the Standards

for Braille Embossed on Paper. Following the standards for Braille on paper enables students to better transfer the knowledge they learn with the device to paper as another medium.

4.2 Engineering Constraints

4.2.1 Ethical

- The device and associate maintenance service must be focused on serving the blind learning braille and not on maximizing profit. Money isn't evil but the direction behind the device and service must not be about maximizing a number on a screen.

4.2.2 Economical

- The device must not cost more than approximately \$150 USD to manufacture.

4.2.3 Environmental

- No non-standard environmental constraints.

4.2.4 Legal

- No non-standard legal constraints.

4.2.5 Social

- The device must be built with interactive collaborative learning in mind between the teacher and student.

4.2.6 Political

- No non-standard political constraints.

4.2.7 Health

- Touchable surfaces of the device must not be made of materials that can or potentially contribute to illness such as but not limited to cancer.

4.2.8 Safety

- Habitual use of the device must not contribute to finger calluses, arthritis, or any form of finger injury. The actuating Braille components must be built with ergonomics and sustainable use in mind.

4.2.9 Manufacturing

- The device must fit in a 6x6x3 space (length, width, height).

4.2.10 Sustainability

- The device must be able to receive and install updates pushed from a company server for maintenance purposes.

4.2.11 Storage

- The device must be able to receive and install updates pushed from a company server for maintenance purposes.
- No vital data is required to be stored on the device.

Chapter 5: Project Design

In this chapter, we aim to present a holistic view of the system's components, offering an extensive overview of its schematic representation, structural elements, and interrelationships between parts. This comprehensive description includes detailed insights into each component, describing its functionality, and methodologies employed in its integration, and potential design alternatives that were considered. Furthermore, this chapter delves into the system's connections with other related products, providing a comprehensive resource list and skill set requirements necessary for seamless integration and operation.

5.1 Overview of System Components

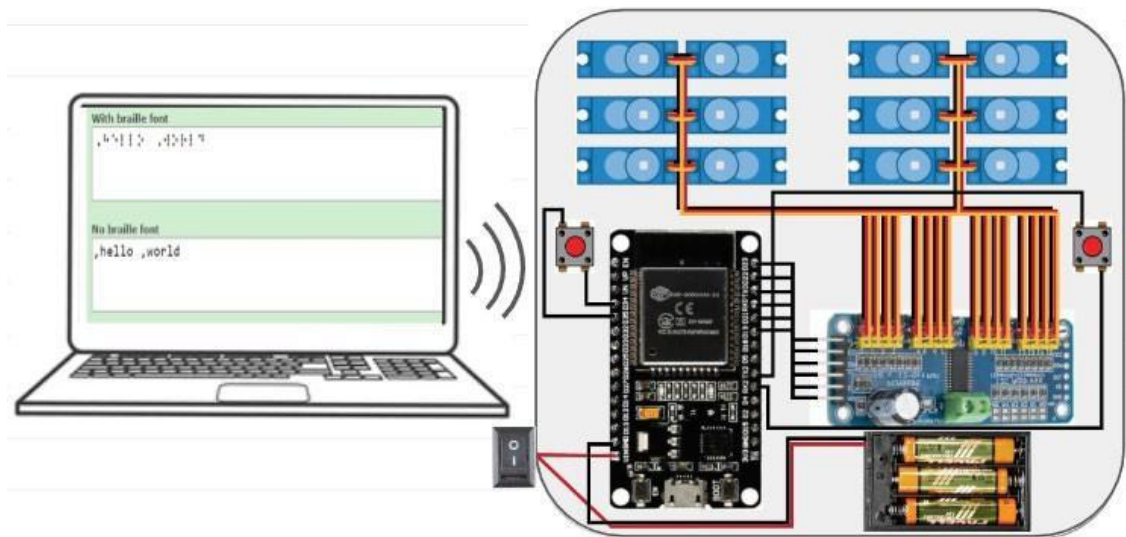


Figure 3: System Schematic

- **Application Interface:** Takes the teacher's/user input, translates it into braille in a way the microcontroller can understand and sends it to the microcontroller via Wi-Fi or hardware.
- **ESP32:** Brains of the device, takes in inputs on the device as well as from the program and does the necessary executions to perform desired tasks. Also responsible for communicating with servos.
- **Servos:** Does the physical act of raising and lowering studs to display whatever letter is being currently selected in braille.
- **Servo Motor Driver:** Since the ESP32 lacks the necessary number of ports for the number of servos required a driver hub is required to be able to make all the necessary connections.
- **Battery/Power Source:** Responsible for providing power to the system so it may run.
- **Previous/Next Buttons:** Allows the user to cycle between characters in translated input from the application.
- **Switch:** Saves battery power by being able to be switched off when not in use.

5.2 Structure and Relationship

5.2.1 Application Interface

- Once an input is submitted, an algorithm is run comparing an array of English characters along with an array of their equivalent braille characters where the selected characters will be stored in a brand-new array which will be sent to the microprocessor.

5.2.2 ESP3

- Once the ESP32 receives the input array, two indexes at a time, it will once again be translated into servo inputs which will activate the corresponding motors needed to display the correct translated letter. In addition, the microprocessor will also be constantly checking for inputs on the next/previous buttons and act accordingly

5.2.3 Servos

- Once input is received from the motor driver the motors with instructions to raise their stud will activate and rotate the set amount of degrees in order to display the studs, all aligned, on the protruding holes of the device.

5.2.4 Servo Motor Driver

- The 12 servos required to make the studs function will all be connected to this driver where the pins will then connect to a set of I/O pins of the ESP32 where a MUX will be used to take a limited number of inputs and have it output over 12 outputs.

5.2.5 Battery/Power Source

- The ESP32 requires a 5V source to function so 2 2.5V batteries will be used to power the system until they run out and need to be replaced.

5.2.6 Previous/Next Buttons

- With the ESP32 always checking for these buttons to be pressed, once they are, it will communicate with the application to receive the next two characters in the inputs or the previous two depending on which of the two are pressed.

5.2.7 Switch

- If the source is not connected to a switch, then the batteries will drain even when the device is not in use. This is not efficient as the device will need to have its batteries switched substantially more often.

5.3 Detailed Component Description

Identification	SC-1: FrontEnd (User View)
Type	JavaScript Class
Purpose	To display all UIs and receive User Input for desired translation
Inputs	User Desired Input for translation
Outputs	User Desired Translation
Data	User Friendly report
Internal Structure	Within a coding environment on the host device for the application
Processing	Start 1) User enters Application 2) View prompts to input and submit their desired translation 3) View sends translation to ESP32
Dependencies	ESP32, User computer
Resources	Memory
Cross-reference	US-1, US-2, US-3

Table 12: Component Description 1

Identification	SC-2: ESP32 Program
Type	C/C++ class
Purpose	To act as the middleman for the software and hardware
Inputs	Soft/Hardware commands (ex. Button presses, textbox submissions, etc)
Outputs	Backend responses
Data	Commands
Internal Structure	Microprocessor
Processing	<p>Start</p> <ol style="list-style-type: none"> 1) A command is issued from the software/hardware (button press, textbox submission, etc.) 2) ESP32 program takes the command and translates it to hardware movements via the ESP32 hardware.
Dependencies	Application's host device
Resources	On chip memory
Cross-reference	US-1, US-2, US-3

Table 13: Component Description 2

Identification	SC-3: Translation Program
Type	C/C++
Purpose	Placed as the software for the ESP32 to communicate between the User View and the Braille Board Hardware. Will do translations given the User View outputs. Will store current translation from User View.
Inputs	User View translations data, Braille Board Hardware next/previous data
Outputs	Servo commands
Data	Translated Text
Internal Structure	Within a coding environment on the host device for the application
Processing	1) Data is sent from the User View 2) Data is stored in a data structure 3) Data read and then translated to servo commands 4) Servo commands are outputted 5) Servos relay info on completion or failure of commands or 1) Data from physical board in sent 2) Data is read, and program checks currently stored translation data 3) Program determines Servo commands 4) Servo Commands are outputted 5) servos replay info on completion
Dependencies	ESP32, User computer
Resources	Memory
Cross-reference	US-1, US-2, US-3

Table 14: Component Description 3

Identification	SC-4: ESP-32
Type	Physical component. microprocessor
Purpose	Acts as the middleman for the User Interface and the physical components via Wi-Fi.
Inputs	1) Next/ Previous button presses 2) UI outputs for translation
Outputs	Commands for servo movement
Data	Arrays with translations
Internal Structure	Pins, internal memory
Processing	S1: the ESP32 receives input from either the physical braille board or the User View S2: The ESP32 uses an already installed software program to move servos to desired positions. S3: the ESP32 relays the info about either successful or failure to move components to the view
Dependencies	Needs to be connected to the User View and to the Servos, as well as have software programmed to work.
Resources	Power, on chip memory
Cross-reference	US-1, US-2, US-3

Table 15: Component Description 4

Identification	SC-5: Servos
Type	Physical component, Motors
Purpose	Up and down movement
Inputs	Commands from the ESP32 on movement
Outputs	Clockwise or counterclockwise movement, as success or failure of movement.
Data	Commands for degree of rotation
Internal Structure	Motors
Processing	1)The Servo is in initial state 2)the servo receives movement command 3)The servo moves 4)The Servo relays completion of movement back to ESP32
Dependencies	Needs to have connection to ESP 32
Resources	Power
Cross-	US-1, US-2, US-3

reference	
------------------	--

Table 16: Component Description 5

Identification	SC-6: Next and previous Buttons
Type	Physical Buttons
Purpose	Communication with ESP32 for Servo Movement
Inputs	Button press to indicate Iteration of next or previous letter in stored translation
Outputs	Moved servos
Data	Button On press
Internal Structure	Switch like, closes loop on press until released
Processing	1) Given that a translation is already stored in the Translation Program 2) The next/previous Button is pressed 3) Info of button press is sent to ESP32, who checks the stored translation for the next letter. 4) Then sends Commands to servos for movement
Dependencies	Connection with ESP32 through driver hub
Resources	Physical component
Cross-reference	US-1, US-2, US-3

Table 17: Component Description 6

5.4 Methodologies

5.4.1 Research New Tools and Methods

Components and Their Accompanying Inputs and Outputs:

1) Frontend (UI)

- a) Input: User input Text Translation, Submission Button On-click
- b) Output: request sent to Translation program, Translation of User Input to view

2) Translation Program

- a) Input: Request from Frontend with user input
- b) Output: Translation of user input sent to ESP32 program + UI

3) ESP32 Program

- a) Input: translation program request or User hardware button press
- b) Output: Hardware Movement (mostly servo movements)

4) ESP32

- a) Input: request from Translation Program or User button on-click
- b) Output: Hardware Movement

5) Servo

- a) Input: Request from ESP32 to move
- b) Output: move

6) Next/Previous Button

- a) Input: User On-Click
- b) Output: request to ESP32 for hardware movement

5.4.2 Identify New Tools and Methods

Frontend –

- 1) Built Via Flutter, Dart, or HTML depending on if we choose to do Webpage, Mobile Application, or Both.
- 2) Research includes Basic Webpage setup and front-end programming as well finding a method to test webpage.

Translation Program –

- 3) Built using C/C++ to better integrate with the ESP32 program
- 4) Research includes practice with C language

ESP32 Program –

- 5) Built using C/C++ because of previous experience
- 6) Research will include practice in C language and how it interacts with hardware

ESP32 –

- 7) Specific model of ESP32 will be the ESP32-Wrover-FreeNove
- 8) Research includes ESP32 “How-To” Tutorials on how to wire different components together as well as how the ESP32 Program will interact with said Components.
- 9) There are lots of online free tutorials on ESP32 Programming.
- 10) We chose the ESP32 specifically because of its WIFI connection capabilities and it's relatively cheap cost. Other options for compact development boards include Arduinos (issue: no WIFI connection abilities), Raspberry PI (Issue: Too Costly).

Servo –

- 11) Must create a method for Up and Down Movements
- 12) Research includes finding and testing ways to turn clockwise and counterclockwise movements into up and down movements

Next/Previous Buttons–

- 13) Must find some nice buttons with a good size and allow for firm presses.
- 14) Research: what buttons are good for the blind.

5.5 Design Alternatives

- 1) Dynamic Braille display using SMA coil actuator and magnetic latch
- 2) Refreshable Braille Display with Adjustable Cell Size for Learners with Different Tactile Sensitivity

The current design was chosen because of its simplicity and focus on developing a pleasant experience for the user rather than just a hardware piece. This allows the project to be completed within the allotted time, as well as provide experience with the frontend, backend, and hardware aspects.

5.6 Reuse and Relationships with other Products

No aspect of this design will have components, both software and hardware, that were used in other projects. Compared to other products on the market, this device will be substantially more affordable. Everything will be created from scratch, and we plan to own the rights to the design of the project.

5.7 Resource List

Name	Description	Availability	Cost
KaiRui	Student involved in project	15 hrs/Week	None
Chris J	Student involved in project	15 hrs/Week	None
Chris R	Student involved in project	15 hrs/Week	None
Prof. Segal	Advising Faculty	1 hr/Week	Tuition
3D printer	Making the shell to house all the components	Operating Hours at Hofstra	Free
Coding Environments	Create code for all the components to interact with one another	24/7	Free
Online Electronic Stores	Order said components	24/7	Price of Components Vary

Table 18: Resource List

5.8 Resource Skill List

Resource Skill	Skill Description	Where Skill will be Applied	References
Software Engineering	Ability to program in a team setting on a large scale	Communicating and working together on the software aspects	US-1, US-2, US-3
Logic Design and Digital Circuits	Design for circuits and how they interact with one another	Creating a circuit while having all the components correctly wired	US-1, US-2, US-3
Microprocessor Systems	How to write code for microprocessors and understand their inner workings	The ESP32 is a microprocessor so this certain programming language (assembly) will come in handy	US-1, US-2, US-3
Adv Data Structures, Object-Oriented Programming	How to organize data into structures that will enable algorithms to most efficiently execute programs	As this project deals with the English language as well as the braille equivalent code, it is necessary to implement data structures to house them.	US-1, US-2, US-3

Table 19: Resource Skill List

Chapter 6: Test Plan

This chapter encompasses the comprehensive test strategy, presenting both the overarching test plan and detailed test cases. Within this section, we outline our approach to testing, incorporating the test plan and detailing specific test cases that will be executed. The content covers a structured overview of our testing methodology and specific scenarios, ensuring a clear evaluation of the system's functionalities and performance.

Section 6.1 Test Plan

Since our device relies heavily on the integration of hardware and software, we have decided to center our testing plan around functional tests while including unit tests where appropriate. This approach enables us to effectively test our device and device components in a time efficient manner. Much of the code that is required for the operation of our device is not overly complex. The more challenging part of our project is the proper integration of software and hardware and ensuring that their integration produces an outcome that fulfills the functionality required for the user.

Our testing will be conducted manually using test scripts and visual verification. Each test requires that we flash our program onto the ESP32. So, to be efficient with our time, we combine unit and integration tests, when possible, into a single test.

Our test cases are arranged to test each individual component of the system through a mix of functional and unit tests. Our test cases are concluded with a final test simulating an end user using our device.

Section 6.2 Test Cases

Name	TC-1: Verify that ESP32 can properly actuate Braille Dots Assemblies
Type	Unit, Functional
Description	Test that ESP32 can actuate Braille Dots Assemblies
Preconditions	<ul style="list-style-type: none"> - ESP32 is powered on - ESP32 and Braille Dots Assemblies are properly wired together
Basic Course of Events	1. Each dot in the assembly raises and lowers sequentially (dots 1 through 6, inclusive, for each assembly)
Expected Results	Each dot in the assembly raises and lowers sequentially
Acceptance Criteria	Each dot in the assembly raises and lowers sequentially without hesitation
Cross-References	N/A

Table 20: Table for Test Case 1

Name	TC-2: Verify that ESP32 can display the next and previous set of braille characters when the user interacts with the next and previous buttons on the device
Type	Unit, Functional
Description	Test that ESP32 can display the next and previous set of braille characters when the user interacts with the next and previous buttons on the device

Preconditions	TC-1 completed successfully
Basic Course of Events	<ol style="list-style-type: none"> 1. Load Braille alphabet, numbers, and punctuation onto the device. 2. Cycle to the end of the text 3. Cycle to the beginning of the test
Expected Results	Braille content will be loaded to the device. Device will properly cycle through all the content

Acceptance Criteria	<ul style="list-style-type: none"> - All braille characters are represented accurately on the braille board. - Say our device has 6 braille cells, when the next button is pressed, device displays the next 6 characters in the translation. - Say our device has 6 braille cells, when previous button is pressed, device displays the previous 6 characters in the translation.
Cross-References	N/A

Table 21: Table for Test Case 2

Name	TC-3: Verify that ESP32 can translate English text to Braille and display the translation to the Braille board
Type	Unit, Functional
Description	Test that ESP32 can translate English text to Braille text. Specifically, English letters, numbers and punctuation.
Preconditions	TC-2 completed successfully
Basic Course of Events	<ol style="list-style-type: none"> 1. Load an English sentence to ESP32. 2. ESP32 translates sentences and displays them on the braille board. 3. Translation is verified manually
Expected Results	An accurate translation of the given English sentence is represented on the braille board
Acceptance Criteria	An accurate translation of the given English sentence is represented on the braille board
Cross-References	N/A

Table 22: Table for Test Case 3

Name	TC-4: ESP32 connects to local private Wi-Fi network
Type	Unit
Description	For a user to have access to the ESP32 through their private Wi-Fi network, they must have a way of connecting the device to their work
Preconditions	- ESP32 is powered on
Basic Course of Events	<ol style="list-style-type: none"> 1. ESP32 establishes its own network 2. User connect to ESP32 network and provides credentials to their Wi-Fi network 3. ESP32 joins user's Wi-Fi network

Expected Results	User joins the ESP32 network and provides Wi-Fi credentials. ESP32 joins Wi-Fi network
Acceptance Criteria	User joins the ESP32 network and provides Wi-Fi credentials. ESP32 joins Wi-Fi network
Cross-References	FR-1

Table 23: Table for Test Case 4

Name	TC-5: ESP32 serves web interface to user
Type	Unit
Description	ESP32 should serve web interface for the device when a user navigates to its IP address
Preconditions	<ul style="list-style-type: none"> - TC-4 - User's computer/phone and ESP32 are connected to the same network
Basic Course of Events	<ol style="list-style-type: none"> 1. User navigates to the IP address of ESP32 2. ESP32 serves web interface to user
Expected Results	<ol style="list-style-type: none"> 1. User navigates to the IP address of ESP32 2. ESP32 serves web interface to user
Acceptance Criteria	ESP32 serves web interface to user when user navigates to its IP address
Cross-References	US-1

Table 24: Table for Test Case 5

Name	TC-6: Web interface initiates English to Braille translation and translation navigation
Type	Functional, Integration
Description	The web interface can take English text, provide it to ESP32 for immediate translation and display/navigation on the braille board
Preconditions	TC-5
Basic Course of Events	<ol style="list-style-type: none"> 1. User submits English text to a text field 2. Text is received by ESP32, translated, and displayed on the braille board for navigation
Expected Results	Web interface initiates English to Braille translation
Acceptance Criteria	Web interface initiates English to Braille translation
Cross-References	US-1

Table 25: Table for Test Case 6

Name	TC-7: Web interface cancels an English to Braille translation and navigation
-------------	--

Type	Functional, Integration
Description	The web interface cancels/ends a running translation/navigation
Preconditions	TC-6

Basic Course of Events	1. User presses a button on the web interface to cancel the running translation/navigation
Expected Results	User presses a button on the web interface to cancel the running translation/navigation
Acceptance Criteria	When cancelation button is pressed, the web interface cancels the running translation/navigation
Cross-References	US-1

Table 26: Table for Test Case 7

Chapter 7: Project Plan

This chapter is dedicated to a detailed exploration of the implementation process conducted in iterative sprints. It will meticulously examine the sprint objectives, demo benchmarks, task breakdown, estimated timelines for task completion, and task assignees. Additionally, this section will encompass a comprehensive risk management plan aimed at identifying, assessing, and mitigating potential project risks. Furthermore, it will outline the teamwork plan, highlighting strategies and collaborative approaches adopted by the team to ensure effective coordination and synergy throughout the implementation process.

7.1 Sprints

Section 7.1 includes the sprint schedules for the team to plan the project implementation. There are three 4-week long sprints where after each sprint there is a working product.

Section 7.1.1 Sprint 1

Sprint goal: Assemble the braille device (ESP32 + 3D printed housing + custom actuating braille cells) and ensure that all hardware and software components operate and integrate with each other properly. As well as having a base program that will be the main way the user interacts with the device.

Sprint demo goal: Present the fully constructed braille device and demonstrate the translation algorithm on the software side.

Starting date and period: September

Sprint master: Chris Jaramillo

#	Task Description	Hardware/ Software Resources	People	Estimated Hours	Points Difficulty	Cross References
1	Construct custom actuating braille cells	- Small actuators - ESP32	KaiRui Rios Jaramillo	15	7	FR-2
2	Create device housing model	- Tinkercad	KaiRui	4	4	FR-2
3	3D print device housing	- Hofstra 3D printer	Jaramillo	4	4	FR-2
4	Assemble device	- Device components	Rios	4	3	FR-2

Table 27: Table for Sprint 1

Section 7.1.2 Sprint 2

Sprint goal: Program the device to translate English to braille, display the result through the devices braille board, and navigate through said translation (forward and back buttons)

Sprint demo goal: Present the device translating a given English sentence, displaying the translation on the braille board, and navigating through the translation by using the device's forward and back buttons.

Starting date and period: October

Sprint master: Chris Rios

#	Task Description	Hardware/ Software Resources	People	Estimated Hours	Points Difficulty	Cross References
1	Make web/app software to translate English to braille	- ESP32 docs	Rios	10	5	US-2
2	Write assembly code to display braille through servos	- ESP32 docs	KaiRui	10	5	US-2
3	Write assembly code that responds to button presses to navigate through displayed braille	- ESP32 docs	Jaramillo	10	5	US-2

Table 28: Table for Sprint 2

Section 7.1.3 Sprint 3

Sprint goal: Enable the device to connect to private Wi-Fi networks and build a web interface for the user to interact with the braille device. Interface the device and web interface so that device can be controlled from the web interface.

Sprint demo goal: Present the device being used in the way it would be used by an end user. Connect device to Wi-Fi network and while using the web interface, translate English to braille and see it appear on the braille board,

Starting date and period: November

Sprint master: KaiRui Hu

#	Task Description	Hardware/ Software Resources	People	Estimated Hours	Points Difficulty	Cross References
1	Make software to enable ESP32 to connect to private Wi-Fi network	- ESP32 docs	Rios	5	3	FR-1

2	Make Web UI more friendly to users unfamiliar with tech	- Online website building instructions (HTML/CSS)	KaiRui Rios Jaramillo	15	5	US-1
3	Integrate web interface with device	- ESP32 docs	KaiRui Rios Jaramillo	15	5	US-1

Table 29: Table for Sprint 3

7.2 Risk Plan

Risk #	Description	Probability	Impact	Alleviation
1	When assembling the components together, it is discovered that some are not compatible with one another	Medium - High	Medium	Find an outside source that allows them to communicate together or replace the part all together.
2	Device doesn't sustain a strong wireless connection	Low	Medium - High	Converting the device to be fully wired, changing the design schematic
3	Cost of assembly is higher than target price for making product affordable and accessible	Low	High	While prior research has been conducted to prevent this, there may be unforeseen expenses in the development process. Reevaluating design and condensing/eliminating redundant components is best course of action.

Table 30: Table for Risk Plan

7.3 Estimated Financial Budget

Item	Estimated Cost
ESP-32 Microprocessor	\$12
Servos	\$35
Servo Controller	\$10
Wires	\$8

Switch	\$3
Buttons	\$1
Batteries/Power Source	\$5
Total Cost	\$74

Table 31: Table for Estimated Costs

7.4 Teamwork Plan

By holding set meeting times where all group members are available and designating time to work on the design every week, as well as completing tasks outside of being together, the project should be completed in a timely manner. In addition to in person, online resources may be used to make collaboration easier. Zoom is best when group members are not available to meet in person, a code collaborator like Codeshare can be used to collectively work on the many code components of the assignment.

Chapter 8: Project Methods

This chapter delves into the methodologies, techniques, software tools, libraries, and new knowledge pivotal to our project's successful implementation. The discussion is structured into sections that directly relate to our project's development, encompassing Technical Methods, Tools, Programming Languages, Use of Open Source Code, and insights gained from adopting New Tools and Knowledge.

Section 8.1: Technical Methods

The Main algorithm that runs the braille board is the Loop() program, The Loop program is what is used to move the servos according to the user input and what makes the servos move with the buttons the program has a few steps:

- 1) String input = readFile(SPIFFS, "/inputString.txt");
 - a) Get the user input that is saved in the ESP32 SPIFFS and reference it as a string named input
- 2) char character = input[currentLetter];
 - a) currentLetter is an int that is defined outside the Loop function as to not reset the currentLetter index every time the program loops.
 - b) Character gets the current letter being translated from the user input
- 3) char *braille = getBraille(character)
 - a) getBraille is a function created to take a single character and iterate through a prewritten dictionary named EngToBrailleDict and returns the 6 binary braille representation of the character.
- 4) Next is the main algorithm for the servo movements for the first 6 dots as written in pseudocode below.

Pseudocode

```

    If the braille of the character is found on the table{
        Loop though the dot representation to move servos{
            Check for servo 1,3,and 6 because they move counter clockwise{
                Check if the current servo position needs to be changed{
                    Change it
                }
            }
            Check all other servos to move clockwise{
                Check if the current servo needs to be moved{
                    Change it
                }
            }
        }
    }
    Otherwise{
        print the braille representation does not exist
    }
}

```

Section 8.2: Tools

Libraries

```

//library used for wifi
#include <WiFi.h>
//library used to make a web server
#include <ESPAsyncWebSrv.h>
//Library used to load information into the esp32
#include <SPIFFS.h>
//library used to move the servo hub and servo motors
#include <Wire.h>
//library used by the servo driver
#include <Adafruit_PWMServoDriver.h>
//libraries used for strings
#include <string.h>
//build in header file for C
#include <stdio.h>

```

Section 8.3: Programming languages

C, HTML, CSS, Javascript

C, HTML, CSS, and JavaScript were intricately combined to craft a comprehensive interface for the Braille device. HTML provided the structural foundation, CSS styled the interface elements for a user-friendly experience, while JavaScript facilitated dynamic interactions and translation processing. JavaScript played a pivotal role in converting user inputs into instructions for the servo motors, ensuring accurate Braille output. Additionally, code was written specifically for the ESP32 microcontroller in C language, enabling it to communicate with the interface, process the instructions from JavaScript, and control the servo motors accordingly, forming the essential bridge between the user interface and the physical operations of the Braille device.

Section 8.4: Open Source Code

1. Freenove. “Chapter 30 WiFi Working Modes Starter Kit for ESP32S3.” YouTube, 7 June 2023, www.youtube.com/watch?v=6_JEtbyC0S8.
 - a. Used to start the Wifi for the ESP32.
2. “GitHub - Freenove/Freenove_ESP32_WROVER_Board: Apply to FNK0060.” GitHub, github.com/Freenove/Freenove_ESP32_WROVER_Board.
 - a. Served as basic tutorial on the ESP32 and Arduino IDE.
3. Control Multiple Servo Motors Using ESP32: Multiple Servo ESP-SERV3 - Robojax. robojax.com/learn/arduino/?vid=robojax_ESP32_Servo-3-Multiple.
 - a. Used to learn to move servos with the ESP32.
4. Control 32 Servo Motor Using PCA9685 Module and ESP32 V4 - Robojax. robojax.com/learn/arduino/?vid=robojax_PCA9685-V4.
 - a. Used to learn to connect servos to the Servo controller, as well learning to wire the servo controller to the ESP32, and the basic code to move all the servos at once.
5. Santos, Rui, and Rui Santos. “ESP32 Web Server - Arduino IDE | Random Nerd Tutorials.” Random Nerd Tutorials, 2 Apr. 2019, randomnerdtutorials.com/esp32-web-server-arduino-ide.
 - a. Used to learn how to add a webpage onto the ESP32.
6. SPIFFS in ESP32. www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_spiffs_storage.htm.
 - a. Used to learn how to add user inputs onto the ESP32.
7. Programming Electronics Academy. “Get Started With ESP32 Touch Buttons [Code and Wiring Diagram Included].” YouTube, 22 Sept. 2023, www.youtube.com/watch?v=acEs6YCWhJo.
 - a. Used to learn how to wire and code buttons onto the ESP32

Section 8.5: New Tools and Knowledge

Name	Type	Acquired By	Learned Through	Applied In
Arduino IDE	IDE	Download	Reading Documentation	ESP32
Servo Driver	Hardware	Purchase	Watching how to videos	Servo Movement
OnShape	Online program	Asking Professor Albers	Tutorials	3D Modeling
JavaScript	Programing	Downloading IDE	Tutorials	Making

Name	Type	Acquired By	Learned Through	Applied In
	Language			Webpage
HTML	Programing language	Downloading IDE	Tutorials	Making Webpage
ESP32	Hardware	Purchase	Watching videos, reading documents, trial and error	Micro controller that intermediates the hardware and software

Table 32: Table for tools used, its application, type, and how its acquired.

Chapter 9: Project Implementation and Results

This chapter revolves around our implementation strategies, detailing the team's approach to project execution, sprint planning, specific sprint goals, demo objectives, backlog breakdown, team members assigned to each backlog item, estimation versus actual time taken for each backlog item, and the testing outcomes.

Section 9.1: Implementation Strategy

In the first half of the project, while planning the project, we realized that there would be 3 main components to the project, the webpage, the hardware (this includes 3D modeling, creating different mechanisms for changing the circular movements of the servos to vertical movements, and wiring), as well as Programming the ESP32 with the translation program, servo movement program, and the middle man for the webpage and the hardware. We decided to split each of these into a different sprint and then we each chose one that we wanted to be the “Main” designer of, and we would have one other person, the “helper” to assist us in making that part of the project, while the third, the “secondary” would begin work on a different component, then when the sprint would finish we would move the “main” designer to the “helper” of the “secondary”, and the “secondary” would be the new “main” designer, while the “helper” would move to a new component and they would be the “secondary”. This method would help us evenly distribute the tasks and also allow us to stay on top of the assignments, while trying to maximize efficiency.

During the first sprint, Christopher Rios was leader and designed the webpage, the HTML, the Javascript, and the CSS, Kairui Hu was the assistant, and Christopher Jaramillo worked on developing the servo movement mechanism. During the second sprint, Kairui Hu lead the work on the ESP32 Programming (accessing wifi, adding the server with the HTML and JavaScript, and basic servo movements), Rios assisted Kairui Hu with the programming as we realized this task would take a lot longer than expected, and Jaramillo moved on to working on

welding and wiring the stepper motors (we ended up not using due to the motors overheating).

During the third sprint we realized we still had not finished with the ESP32 programming as we still needed to program a method move the servos with the servo driver, make the translation program (including the method to change a character into its braille representation and returning that representation to later be used to move the servos), as well as creating buttons and implementing said buttons. Because of the heavy workload Karui Hu and Rios continued to work on the programming while Jaramillo worked on 3D modeling the container, and other parts and pieces that would go into making the servo movements from rotational to vertical. And we also all worked on integration, at this point all of the program was fully integrated and there only needed to be adjustments made when the 3D model was complete.

During the entire process we communicated with each other via text, we would have zoom meetings once a week and meet with our adviser once or twice a month.

Project management on the other hand we did not have a well thought out plan, as such we used canvas to hold the files as canvas provided a group tab we all had access to, for version control we used different files for different sprints.

Section 9.2: Sprints

9.2.1: Sprint 1

- Duration: September - November
- Goal: Work to complete the Application, Start hardware development, Get acquainted to moving the parts with code
- Demo goal: Working and testable English to Braille translation program written in C, translate any English word into braille represented by an array of 1's and 0's.
- Accomplished work: Completed the English to Braille Translation Program, Made a basic UI, Estimated the size and Shape of the final product 30%, Formulate a plan on what parts to use
- Accomplished demo: HTML file demonstrating an English to braille translation simulating the web UI. Takes in user input and translates it into braille, character by character.
- Conclusion: very successful, we got done what we wanted to accomplish for the web page.

BackLog Item	Teammate	Completion date	Estimated number of days to complete	Actual days it took to complete
Complete the Program to Translate English to Braille	Christopher Rios, Kairui Hu	September 28th	10 days	15 days

Planning, Prototyping, and Testing different mechanisms for the physical component of the braille board, focusing on molding the braille dots using various parts and components to emulate real braille dots	Chrisopher Jaramillo	September 30th	30 days	30 days
---	----------------------	----------------	---------	---------

Table 33 : Sprint 1 Backlog & Completion

9.2.2: Sprint 2

- Duration: October 2023 - November 2023
- Goal: Have the Hardware complete and working, Moving Stepper Motors connected with the ESP32.
- Demo Goal: Have a Working machine that can move 12 braille "dots", the Demo will show the machine moving independently from one another as well as together in a group.
- Accomplished work: ESP32 Connecting to the webpage and connected to the Wifi for internet access, servo motors connected to the servo hub, and connected to the ESP32, Code for moving the servo motors with esp32 complete. Have a method to save input from webpage and saved to ESP32
- Accomplished demo: ESP32 Connecting to the webpage and connected to the Wifi for internet access, servo motors connected to the servo hub, and connected to the ESP32, Code for moving the servo motors with esp32 complete. Have a method to save input from webpage and saved to ESP32
- Conclusion: although we had procrastinated a bit early on, we got a good amount of stuff done completing a lot of the ESP32 tasks and beginning integration

BackLog Item	Teammate	Completion date	Estimated number of days to complete	Actual days it took to complete
3d Modeling (simple test)	Kairui Hu	N/A	7 days	30 days
Linking ESP32 with Arduino IDE	Kairui Hu	October 20th	1 day	1 day

Connect Servo Driver to Servos to ESP32 + Write Code to move servos using servo driver	Kairui Hu and Christopher Rios	October 30th	1 day	1 day
Write code to Activate the ESP32 WI-FI	Kairui Hu and Christopher Rios	October 30th	2 days	5 days
Write code to integrate ESP32 to webpage	Kairui Hu	October 30th	2 days	2 days
Soldering Motor parts together	Christopher Jaramillo & Christopher Rios	October 20th	1 day	1 day
Write code to test and move the Motors	Christopher Jaramillo	October 22nd	2 days	2 days
Misc Research(connecting ESP32 to wifi, how to move a stepper motor with esp32, etc)	Christopher Jaramillo	October 30th	7 days	7 days

Table 34: Sprint 2 Backlog & Completion

9.2.3: Sprint 3

- Duration: November 2023 - December 2023
- Goal: Finish the codes for moving the braille cells, make code for button movements, combining the separate codes for moving the cells and the buttons, combine the codes for the webpage and wifi with the movement code to finish the software side
- Demo Goal: Having a complete simple braille board that can connect to wifi, and allows the user to enter a website to input a word and having that word represented on the braille board and controlled with buttons
- Accomplished work: All the Above, except the 3d printed model due to scheduling issues and the printing taking longer than expected to complete.
- Accomplished demo: All the above except the 3d printed model
- Conclusion: We got everything but the 3d print complete because of printer issues, but all in all we got everything else complete.

Backlog Item	Teammate	Completion date	Estimated number of days to complete	Actual days it took to complete
Make simple code to get buttons to work with the esp32	Kairui Hu & Christopher Rios	November 15th	2 days	3 days
Write code to get the braille cells to move with buttons given a string	Kairui Hu & Christopher Rios	November 17th	1 day	1 day
Write code to integrate both the wifi code with the movement code	Kairui Hu & Christopher Rios	November 22nd	2 days	1 day
Finish the 3d model and printing	Christopher Jaramillo	December 10th	15 days	24 days
Integration of parts	All	November 30th	1 day	1 day

Table 35: Sprint 3 Backlog & Completion

Section 9.3: Testing Results

Sprint 1

- End to End and performance test on webpage
- Test Pass

Sprint 2

- Unit Test on ESP32 wifi, webpage, and servos
- Test Pass

Sprint 3

- Unit test, End to End test, and Performance test of translation program and all integrated parts, made sure the user interface could interact with the hardware (made sure when the user enters an input the braille device responds.)
- Test passed

Chapter 10: Problems and Changes

This chapter serves to outline the challenges encountered and subsequent changes made throughout the project lifecycle. It delves into the alterations made to project goals, design paradigms, and sprint methodologies in response to evolving circumstances. By documenting the encountered issues and subsequent modifications, this chapter offers a comprehensive view of the project's dynamic nature, highlighting the adaptive strategies employed to navigate challenges and refine project objectives, designs, and sprint structures

During the development phase, we opted to try transitioning from servos to micro stepper motors. It seemed promising but presented unexpected challenges due to the intricate soldering requirements. The stepper motor's soldering pads' size posed significant difficulties in establishing proper connections which lead us to encounter impediments in functionality. To streamline the process and ensure a more manageable setup, we reverted to utilizing servos. This decision facilitated simplicity by allowing direct connection to a PWM driver, enabling straightforward function calls in the code. This shift not only resolved the intricate wiring issues but also streamlined the operational complexity, ensuring smoother integration into the project.

Moreover, our initial plan involved implementing an on/off switch for convenient power management. However, due to unforeseen limitations in the originally intended battery source, which couldn't deliver adequate current to drive the motors effectively, we had to reevaluate our power supply strategy. Despite our initial intention for portability, the reliance on a large DC power supply became the alternative, though it wasn't ideal for our envisioned portable design. This deviation from the original plan impacted our project's mobility aspect, necessitating the use of a power supply that sacrificed portability for functionality. The adaptation to a stable, albeit less portable power source, became a pragmatic solution to ensure consistent power delivery to the servos, ensuring the device's operational reliability despite the compromise in portability.

Issue	Situation	Action(change backlog, change specification, change design, etc)	Result (solved, partially solved, moved later)
Migrating C code to ESP32	ESP32 has its own specific toolchain	No change	Not solved
Emulate braille dots	Original servos may not be optimal	Order different parts	Partially solved
Mechanical	Mechanisms to	Changed Design to	Partially

Systems	implement up & down movements	use stepper motors	Solved
Motors not pre soldered	Stepper motors were not pre soldered	Find a soldering station and solder the pieces together	Solved
The required wires were too large for soldering to the motor	The wires we need to solder to the stepper motors are smaller than expected	Purchase smaller wires	Solved
Soldered Motors don't seem to work, and instead just heats up	The Stepper motors don't seem to work	Attempt to lower the incoming power	Not Solved
O No, the Motors don't work	The Stepper motors don't seem to work	Changed design back to using the Servos instead	Solved
How to 3d printing	After getting the 3d modeling we needed to 3d print the resultant model	We pushed the 3D modeling finish date back.	We found a teacher, but due to both scheduling issues and 3d printing seemingly going to take a longer time than expected, we need to move the complete result to a later date.

Table 36: Project Issues and Changes

Section 10.1: Project goal changes

Please see section 9.2 for each Sprint' goal, demo goal, starting day, end day, backlog (list of tasks), team members assigned to each backlog item, completion date, estimated number of days to complete and actual days it took to complete each backlog item. There were no big changes to sprint tasks that were established prior to the start of each sprint but there were additions added as exploration for

possible changes to the project to lead it in a more efficient direction. In the end, the original plan was deemed to be most effective.

Section 10.2: Design Changes

The only major change to the project was the addition of an on/off switch since we were not able to establish a reliable power supply. The original 4 1.5V batteries were not able to supply efficient current to the motors as they were not able to process commands and work properly when connected. We opted to use a DC power supply which is not ideal for portability but was great for providing a temporary constant current flow that the 12 motors required.

Section 10.3: Sprint Changes

Priority was usually the main issue for changes in the sprints as parts of the project that were not as vital to its function were being spent too much time on. There were changes to the schedule multiple times in order to get vital components of the project in working order.

Chapter 11: Financial Budget

This chapter will provide a breakdown of the project's financial budget. The content will illustrate an examination of the budgetary components, outline expensive, and resource allocation.

Expenses		
Item	Estimated Cost	Actual Cost
ESP32 Kit	\$65	\$60
Servo motors	\$20	\$20
Stepper motors	\$50	\$50
Servo Motor Driver	\$14	\$14
Total Cost:	\$149	\$144

Table 37: Final Financial Budget Sample

Chapter 12: Conclusions

In conclusion, we accomplished our main goal of creating a cheap and simple alternative to the current braille boards on the market, costing around 100\$ to make at most. In this year-long experience we accomplished a lot, and In the process we learned about teamwork, hardware wiring, programming to more a physical device, we learned how to 3D model, and a lot about the integration process, we learned how to allow our program to interact with the user and respond in kind. We also learned more about the development process, from writing reports, to interacting with different departments to assisting in solving problems, we had a fun time experimenting with the project and felt the joy in seeing our work slowly come together. And lastly we would like to thank all faculty and professors who assisted us along the way in creating this braille device.

Appendix A: Sample UI Diagram

English Braille Translator

Enter text to see how it is represented in Grade 1 Braille.
Hover over a braille cell to see its english representation.

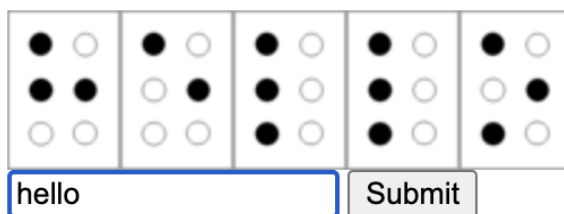


The diagram shows a user interface with a text input field containing the placeholder text "Enter text" and a "Submit" button to its right.

Figure 4: UI - Before User Input

English Braille Translator

Enter text to see how it is represented in Grade 1 Braille.
Hover over a braille cell to see its english representation.



The diagram shows the same user interface as Figure 4, but now the input field contains the text "hello". Above the input field, there are five Braille cells displayed in a row. Each cell is a 3x2 grid of dots. The first cell has dots 1, 2, and 3 filled, representing the letter 'h'. The second cell has dots 1, 2, and 4 filled, representing the letter 'e'. The third cell has dots 1, 2, and 5 filled, representing the letter 'l'. The fourth cell has dots 1, 2, and 6 filled, representing the letter 'l'. The fifth cell has dots 1, 2, and 3 filled, representing the letter 'o'. The "Submit" button is still present to the right of the input field.

Figure 5: UI - After User Input

Appendix B: Sample 3D Model

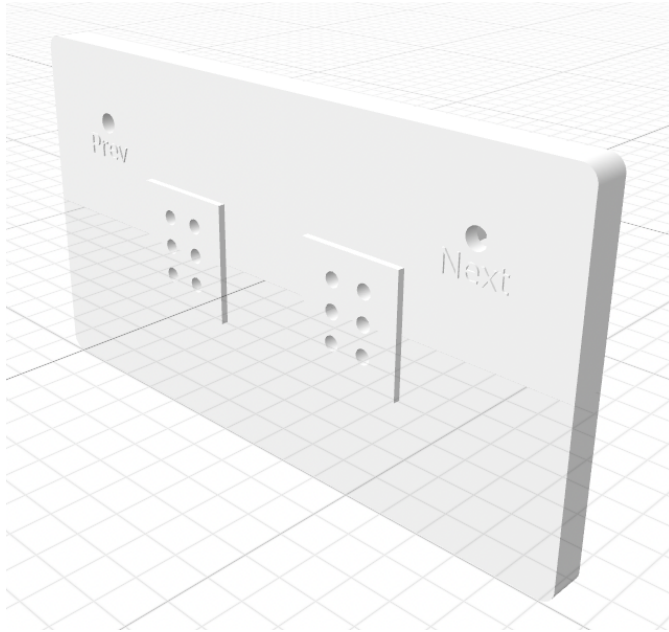


Figure 6: Top case - Perspective View

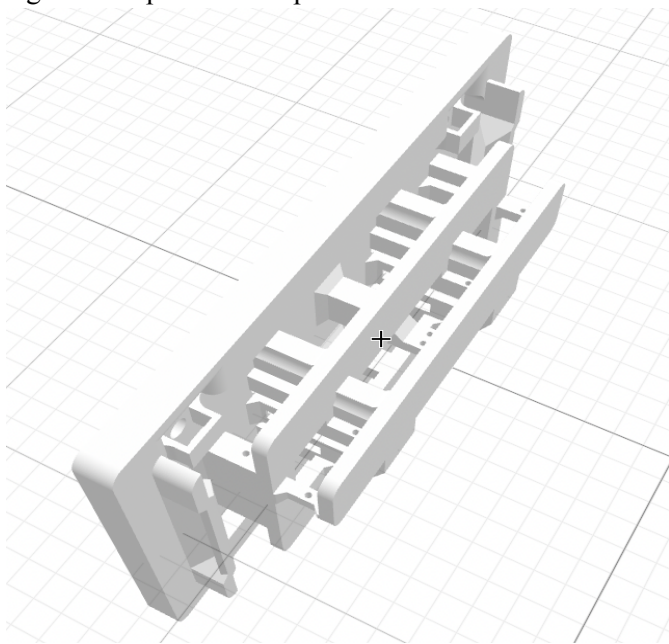


Figure 6: Top case - Top-down-side view

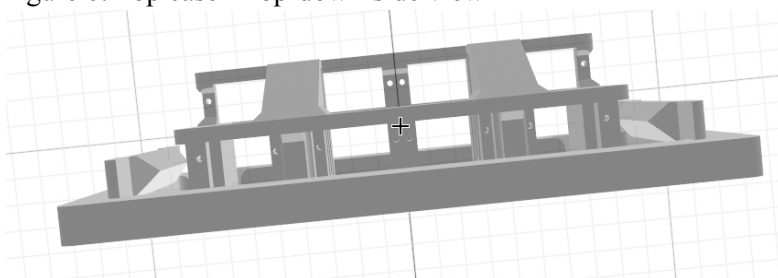


Figure 7: Top Case - Bottom View

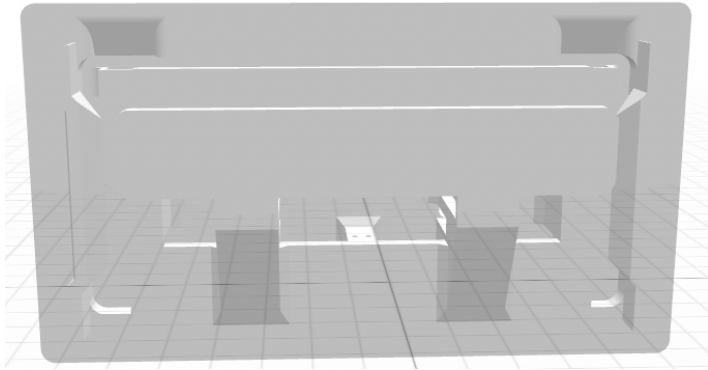


Figure 8: Top Case - Back-View

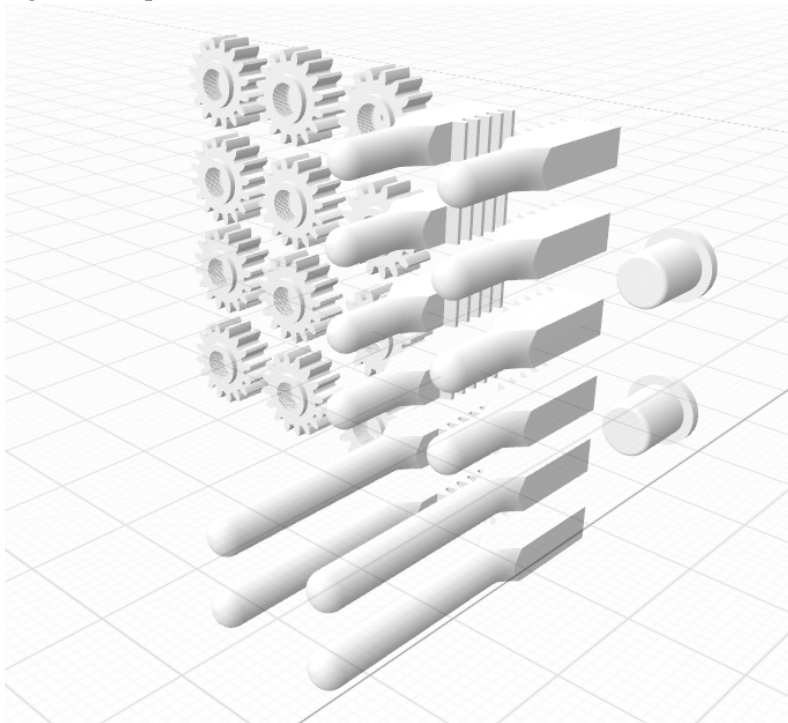


Figure 9: Servo Mechanism Parts

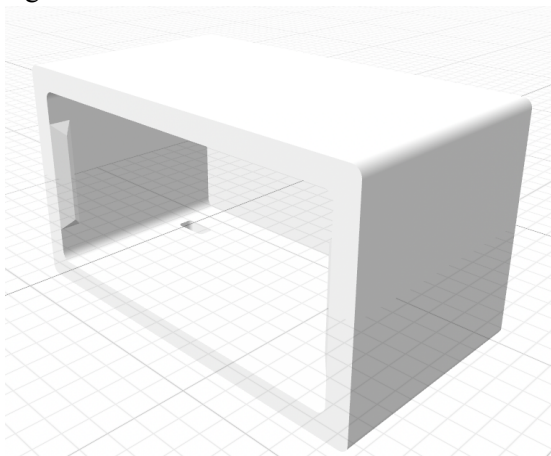


Figure 10: Bottom Case - Perspective View

References

- [1] Learning about braille: Botta, Mike. "Braille in Pharma: Differing Views in the U.S. and Europe." *Pharmaceutical Processing World*, 11 Oct. 2019, <https://www.pharmaceuticalprocessingworld.com/braille-in-pharma-differing-views-in-the-u-s-and-europe/>.
- [2] Learning about braille: Duarte-Barón, Katherin, et al. "Design and Construction of a Device for Facilitating the Learning of Braille Literacy System." *Ingeniería y Competitividad*, vol. 18, no. 1, 2016, p. 77., <https://doi.org/10.25100/iyc.v18i1.2179>.
- [3] Learning about braille: Ibarra, Manuel J., et al. "Ñawinchay: Low Cost System for Facilitating the Braille Literacy for Blind People." 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019, <https://doi.org/10.23919/cisti.2019.8760729>.
- [4] Learning about braille: In America - National Federation of the Blind. https://nfb.org/images/nfb/documents/pdf/braille_literacy_report_web.pdf.
- [5] Learning about braille: authors, All, and Brian W. Stone. "Teaching Visually Impaired College Students in Introductory Statistics." Taylor & Francis, <https://www.tandfonline.com/doi/full/10.1080/10691898.2019.1677199>.
- [6] Learning about braille: "Literacy for Blind and Visually Impaired School-Age Students." *Re:View*, vol. 22, no. 3, Fall 1990, p. 159. EBSCOhost, search.ebscohost.com/login.aspx?direct=true&db=asn&AN=9607211425&site=ehost-live.
- [7] Learning about braille: Ye Wang, and Shariffa Khalid Qais Al-Said. "Defining Literacy for Individuals Who Are Blind or with Visual Impairments: A Qualitative Study of Stakeholders." *Journal of Ethnographic & Qualitative Research*, vol. 8, no. 2, Winter 2014, pp. 99–112. EBSCOhost, search.ebscohost.com/login.aspx?direct=true&db=asn&AN=95101184&site=ehost-live.
- [8] Learning about braille: "Section 1414 (d) (3)." *Individuals with Disabilities Education Act*, 7 Nov. 2019, <https://sites.ed.gov/idea/statute-chapter-33/subchapter-ii/1414/d/3>.
- [9] ESP32 Wifi documentation: Freenove. "Chapter 30 WiFi

Working Modes Starter Kit for ESP32S3.” YouTube, 7 June 2023, www.youtube.com/watch?v=6_JEtbyC0S8.

[10] ESP32 Documentation: “GitHub-Freenove/Freenove_ESP32_WROVER_Board: Apply to FNK0060.” GitHub, github.com/Freenove/Freenove_ESP32_WROVER_Board.

[11] Servo Documentation: Control Multiple Servo Motors Using ESP32: Multiple Servo ESP-SERV3 - Robojax. robojax.com/learn/arduino/?vid=robojax_ESP32_Servo-3-Multiple.

[12] Servo Controller Documentation: Control 32 Servo Motor Using PCA9685 Module and ESP32 V4 - Robojax. robojax.com/learn/arduino/?vid=robojax_PCA9685-V4.

[13] ESP32 Web Server Documentation: Santos, Rui, and Rui Santos. “ESP32 Web Server - Arduino IDE | Random Nerd Tutorials.” Random Nerd Tutorials, 2 Apr. 2019, randomnerdtutorials.com/esp32-web-server-arduino-ide.

[14] ESP32 SPIFFS (Flash Memory) Documentation: SPIFFS in ESP32. www.tutorialspoint.com/esp32_for_iot/esp32_for_iot_spiffs_storage.htm.

[15] ESP32 Button Programming Documentation: Programming Electronics Academy. “Get Started With ESP32 Touch Buttons [Code and Wiring Diagram Included].” YouTube, 22 Sept. 2023, www.youtube.com/watch?v=acEs6YCWhJo.