StudyBuddyBot

Authors: Jeffrey Jehng, Mahlet Mesfin, Shannon Yang
Department of Electrical and Computer Engineering,
Carnegie Mellon University

***Abstract*—Many students struggle to be efficient with their time when studying. Students may find it difficult to stay on task or retain the information they have spent hours studying. The StudyBuddyBot is a robot designed to facilitate studying with features such as timed studying sessions and text-to-speech functionality while also providing relaxation through interactive features, all of which are synced via a companion website.**

For the scope of our project, we aim to design and integrate a robotic system that can be tested on its ability to improve the studying performance of students. Our implementation goal can offer real-time feedback on the study sessions and provide features to aid their studying such as interactive engagement and entertainment during breaks.

***Index Terms*—Design, Robot, Web Application, Human-Robot Interaction**

1. Introduction

In our modern society, students are more stressed than ever as they struggle to achieve a good work-life balance. Many find it difficult to focus on studying and achieve good results due to the various constant distractions. Often, this issue is compounded by the lack of motivation to study, leading to demoralization and burnout. Furthermore, some students also struggle with finding time to take a break, resulting in reduced productivity and excessive fatigue. Subsequently, we saw the need for a solution that not only promotes good studying habits, but also ensures students take the time to relax. Our goal is to develop StudyBuddyBot (SBB), a robot that can motivate students to study, track their studying habits, and help them relax during study breaks.

To encourage students to study, we went with a user-friendly design and a greeting feature upon initial power-on for our robot to provide a welcoming environment that students would look forward to studying in. Additionally, to support the student as they study, our SBB will have studying features in the form of studying sessions, and Text-to-Speech (TTS) functionality. Studying sessions can help develop structured studying habits while the TTS feature can further support audio learners, enhancing the efficiency of their studying. Lastly, to facilitate a good balance between work and play, we have interaction features in the form of a Rock-Paper-Scissors game and Hear-and-Respond feature that the student can partake in with the robot. Rock-Paper-Scissors is a well-known game with simple instructions that students enjoy playing, providing them with some mental relief from their work. Meanwhile, the Hear-and-Respond feature processes a specific audio cue (two consecutive claps), and allows the robot head to turn towards the direction of an audio cue that the student creates for the robot to “look” at them, simulating responsiveness and fostering a more dynamic interaction. These features all come together to create an engaging and productive environment for students to thrive academically while also taking care of their well-being.

There are some existing solutions that try to solve this problem, such as productivity websites (Quizlet, Pomodoro Timers, etc.) and robot desktop pets (EMO, Eilik, etc.), but these typically focus on just one aspect—either improving productivity or providing companionship. SBB combines both important aspects into one product, providing a dual-approach to resolving the challenging task of optimal work-life balance.

2. Use-Case Requirements

SBB's body is built using a quarter inch thick hard board box of dimension 7x6x7 inches. This gives us a total volume of 294 cubic inches. This allows it to house the RaspberryPi , the speaker, and the servo motors. SBB’s head is built using 5x5x5 inches, giving a total of 125 cubic inches.The display screen will be placed on the head, and the joints to allow for the rotation and the translational motion of the robot’s head will be housed in the head. The material is biodegradable and environmentally safe.

For the DCI (Display Control Interface), we require low enough latency to ensure that the initializing of the robot face and displaying necessary elements stay below 3 seconds. One key tradeoff of the design was considering the DCI display vs the standard LCD displays. We decided on a DCI display as it offers faster response time and more advanced interactive features. However, this comes at a slightly higher cost and power consumption, limited by the 120V wall outlet. LCD displays latency may potentially exceed the 3-second requirement, which would make it not ideal for our use case of powering on within 3 seconds. We will work on simplifying the display to prevent this from happening.

When the StudyBuddyBot powers on, the display screen should light up to display a friendly robot face that greets the user within 3 seconds. Our goal is to provide an engaging welcome interaction for the user without compromising system stability or power efficiency. Studies on user experience has shown that a quick response time is essential to maintaining engagement with the user, especially in educational settings, while allowing the robot’s hardware to power up sufficiently. Furthermore, for the DCI display, we plan to have the robot power on and display the interactive face 95% (2 standard deviations from the mean) of the time to ensure a reliable experience for the user. By maintaining this success rate, the system demonstrates consistent performance in normal conditions

and provides the user with an interface that is ready to interact, which can increase usability and user comfort with the robot.

Our next use case is for the power requirement, which aims to keep the stationary robot powered at the students desk for durations of up to three hours. We want to create a system that is energy-efficient while also able to support core functionalities such as communicating with the web app via web sockets or running the rock-paper-scissors game. It takes roughly 5 volts to power the servo motors [14], along with another 5 volts for the Raspberry Pi [15]. Since we are using multiple motors, we aim to keep the total power consumption below 15V. This low power approach makes the robot eco-friendly by reducing energy cost while supporting an engaging studying environment for the students.

For the use case of a Study Session (SS) it involves communication between the WebApp (WA) and the robot. Thus, the corresponding use-case requirements involve how responsive the robot should be. There are 6 such situations. When starting a SS, the user clicks start on the WebApp (WA), the robot should respond with a Timer display in <250ms. Similarly, when the user presses pause/unpause button on the robot, the break in progress/studying in progress screen should appear on the WA in < 250ms. When a session duration is up, the user receives a pop-up alert on WA and if they choose to continue the session, the paused Timer display on the robot should resume in <250ms from when user clicks continue on WA. During a Pomodoro SS (SS with build-in breaks), when transitioning in and out of Break Time, the time between the Timer display restarting on the robot and the Break in Progress/Studying in progress screen appearing on WA should be <250ms. Also during a Pomodoro SS, when transitioning to Break Time, time between Break in Progress screen appearing on WA and audio output from robot should occur <500ms. The additional 250ms for audio outputs is to account for TTS processing time (more in design requirements). Lastly, when the user clicks End Session on the WA, the robot Timer display reverts to its default smiley face in < 250ms. The reason for most of these values being 250ms stems from the natural reaction time for humans and anything beyond 250ms starts to feel "slow" to humans. [1]

The robot's Text-to-Speech (TTS) feature has a use case requirement of outputting the audio in <2.5s from when the input was submitted on the WebApp (WA). This is based on the default time of 2s Siri pauses on iphones to respond to command prompts, and simulates natural response time, [2]. Since this is a lot longer than human reaction time, to prevent the user from being confused about if their prompt was received, we have another use case requirement of the loading screen while the audio output is being computed to show up in <250s, which is once again based on human reaction time. [1]

The SBB is designed to offer engaging study breaks to users via the rock-papers-scissors game available to the user. The game is very simple which makes it easy for anyone to understand the rules, but can also help students relax during a study break. The use-case requirements relies on the SBB and web app being able to communicate with each other with latency of under 250ms. Users are able to start a game on the web app, and input the number of rounds that they wish to play. After, the robot will display on the screen that it is ready to play rock paper scissors, and waits for the user to input a button press to start a round. Our goal is for the start round button to be processed in under 250ms, and the DCI display will flash rock, paper, scissors, for 250ms each. In that window, we want to be able to take in an user input of an option between rock, paper, scissors, that can be inputted before the robot says "shoot", so the user input would also need to be processed in under 250ms. Once the user input is taken in, the robot will declare a winner, and have a happy or sad face displaying on the DCI based on whether the user won or lost. This response should occur within 250ms after the results are revealed. Furthermore, there will need to be communication between the SBB and web app via web sockets, to update the number of rounds the user has along with the results of that round. This way, the user will be able to track their aggregate record across multiple study breaks, which adds to the interactivity the SBB can provide.

SBB must be able to respond to a double clap sound by turning towards the direction of the sound within 3 seconds after receiving the cue from 3 feet radius away. The robot should be able to identify the double clap by accuracy of 90%, as the double clap must happen within 1 second interval. The accuracy is low because it accounts for human error when clapping. If someone's claps are on the edge of the one second interval the audio is "recorded", it could potentially lose data allowing for less recognition. Based on surveyed clap intervals (we conducted) most double claps happen within 0.8 to 1.05 seconds. If the claps happen within public spaces, it might be disruptive to the surrounding, however, since our project assumes improving a user's solitary study experience, this would not be a major concern.

3. Architecture and/or Principle of Operation

The block diagram provided in the Appendix, Figure 8, provides a detailed overview of all the individual components of our solution. It includes the interconnections, all of the software and hardware components, and is color-coded based on which subsystem it belongs to, what kind of component it is, and computation processes in each subsystem. Our main computations for features such as our RPS Game and audio cue response will be done on the RPi, while the WA will be in charge of the other two features, the SSs and the TTS function. The WA

and robot will communicate through WebSockets, which will convey pieces of information such as the text data for TTS, the pause button for SS on the robot, and the RPS game start from the WA. The provided legend also gives more details on each individual component of the diagram, color-coded based on what category they fall under.

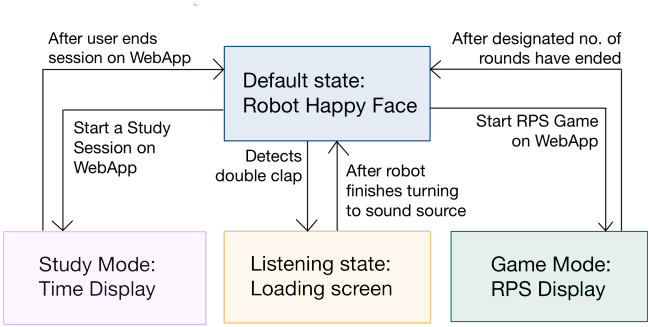

Figure 1: Flow chart of. overall robot display

The flow chart for overall robot display as shown in Fig 1, illustrates how the robot transitions between different displays. When the robot is first powered on, it is in the default state of a happy face. It leaves the default state when the user starts a study session on the WA to display time, and returns to default state when the session is over. When it hears an audio cue (two consecutive claps), it displays a loading screen to indicate it has heard the audio, and then turns to face the source of sound before displaying the default happy face. When the user starts a few RPS game rounds on the WA, the robot transitions to game mode display, (which will be elaborated more with Fig 1. (c)), and goes back to the default state when the RPS game rounds have ended. Note that our fourth feature (TTS) has no display mode because it is a simple process of just submitting input text on the WA and having the robot output audio, there is no other user interaction needed.

4. Design Requirements

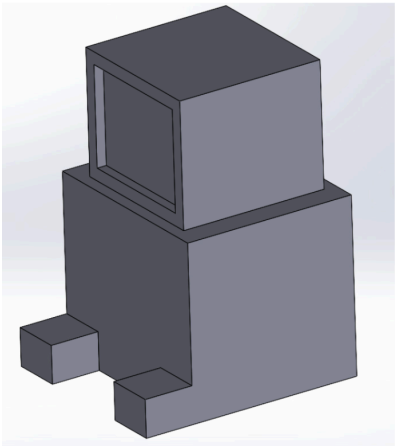

Figure 2: Robot CAD Design

Our first design requirement focuses on our robot base. The robot base design is intended to be spacious to allow the RPi to easily cool down and prevent overheating. Since the Robot is estimated to be on for a prolonged period of time, the internal components can heat up. To reduce damage to any components, we plan on having a spacious bottom, allowing components to sit separately. The RPi's size is 3.75x2.5x1.25 inches. To account for cable length, the length would increase by about 1.75in, considering the HDMI cable(the biggest cable we will have). This gives us an updated dimension of 5.5x2.5x1.25 inches. This can be placed at least 0.5 inches apart from the speaker. The speaker has dimensions of 3.14x1.2x1.78 inches. The 7x6 inches surface dimension allows both components to sit safely. In the body, there will be a shelf 3 inches above the surface. Here, we mount the two servo motors each with dimensions 1.6x0.8x1.6 inches. SBB's head will have the display screen of size 4.7x3.1 inches mounted externally.

Our next requirement is the DCI display latency for robot-only computation features. This relates to our use case requirements in terms of initial power up and display as well as the RPS Game. Our goal is for the robot's DCI display to refresh and show study timers/game prompts with minimal latency. The design requires the display to respond within 250ms of user inputs (in the case of direct button inputs for the RPS game), or under a second in response to audio cues or inputs from the web app. Since the user starts a game and enters their inputs via button press, the GPIO pins for these buttons must have latency of under 250ms to ensure that the game feels responsive. The reason why we chose the RPi 5 is to have hardware that can quickly process the game logic to determine a winner and loser in under 500ms to provide rapid feedback to users. Furthermore, as the number of rounds to be played is inputted from the web app, there should be a communication delay of under one second between the user's input and the game displaying on the DCI screen of the SBB, asking the user if they are ready to start the game. The robot was designed without using a FPGA, as we are only relying on the RPi for game logic and state transitions to maximize the hardware's response time. The buttons that we install on the robot should make it easier for the user to interact with the DCI display of the robot, and also reduce the overall latency as we are relying on the rapid GPIO pins to process button inputs.

Another design requirement we have is for the subsystem of Study Sessions, the main requirements surround the WebSockets latency and DCI/laptop display latency. We have the use case requirement that the WebApp must communicate that a study session has started or is continuing, and the robot must display accordingly in 250ms. This is crucial because users expect immediate feedback when they initiate study sessions. This is a direct requirement based on human reaction times, where delays beyond 250ms become noticeable and disrupts the user experience. Thus, we have the equation

$$T_{WebSocket} + T_{processing} + T_{display} \leq T_{comm}$$

We want the WebSocket transmission will have a nominal latency of around 50ms to support fast data transmission [3], and $T_{display}$ will have a latency of 50ms from before so this gives us $T_{processing}$ (processing time on the robot) to have a latency of ≤100ms to fulfill the use case requirement. A similar design requirement applies to when the robot sends information back to the web app, except $T_{display}$ in this case would refer to the user's laptop display time, which should be slightly faster since this comes from client-side processing time and browser rendering time which should be under 50ms in total considering that we are performing a lightweight operation with minimal DOM operation [4]. As such, $T_{processing}$ (processing time on the WebApp) will then have to have a latency of ≤150ms.

The audio detection and response subsystem has a design requirement that is based on how long it takes to detect the double clap cue, concurrently from the four microphones. Each microphone is going to be placed at the four corners of the robot's body. Depending on the location of the double clap, this sound reaches each microphone at different times. Based on data we collected, it is required that the double clap happens within one second. After taking in this data, each microphone will keep track of the audio input from the last one second, using a standard sampling rate of 44.1kHz. We take in the last second's audio input and split it by half to correlate and detect the double clap. Using this method, we aim for an accuracy result of 90%. This value accounts for multiple double clap detection within the limited 3 feet radius area. Each microphone will be keeping record of the data from the past second and if the audio is detected, this data will be stored and processed until the robot turns its face in the direction of the audio source. After this, we will take one second to compute location, and another second to set the servo to the source of audio cue. This will ensure the process happens within 3 seconds of audio detection.

5. Design Trade Studies

5.1 Hardware

5.1.1 Computation Device

For data processing and running various tasks simultaneously, we looked at several different options. One of the options were the STM-32 Nucleo boards. While the STM32 is a powerful microcontroller for simpler tasks such as running the servo motors, it lacks an operating system to allow efficient multitasking. Multitasking would have to be manually implemented through interrupt handlers or RTOS. This makes the design much more complex. RPi has a linux operating system allowing for an effective multitasking environment, by managing the servo motors, the display interface, playing Rock-Paper-Scissors,and running the text-to-speech system simultaneously. In addition to this, The RPi has a built-in WiFi and Ethernet, making it ideal for the communication between SBB and the WebApp.

5.1.2 Robot neck motion

When implementing the rotational and translational motions of the robot's neck, we compared and contrasted stepper motors and servo motors. A stepper motor divides a full rotation into multiple discrete steps allowing precise movement increments. This is often used when a device needs extremely accurate control of the angular position. However, for our purposes, stepper motors were excessive options as we are seeking for the general direction of the double-clap source. The motor would draw high current especially as it tries to maintain a position, since it is constantly energized to maintain torque. Stepper motors also tend to be bulkier than servo motors. We chose to use a servo motor since it provides simpler, smoother and more efficient control over the motion. The neck location has a margin of error of 5 degrees, and since the neck doesn't require high precision, the servo's feedback control offers sufficient accuracy. In addition, servo motors also have lower power consumption which is ideal for SBB, who will run continuously for the study session.

5.1.3 Input Microphones

To implement the sound localization and audio triangulation functionality, we compared and contrasted MEMS microphone (MM) and Cardioid Condenser microphone (CCM). The CCM uses a polar pattern to capture sound from the front while minimizing ambient sound from the sides and the rear. Using four CCMs, arranged in the corners of the robot, we can achieve more precise sound localization in the horizontal 2D plane. The cardioid pattern of each mic will help isolate the sound coming from specific directions at high accuracy. This will help in reducing interfering audio. Omnidirectional MM would require more complex signal processing to achieve the same level of directional accuracy as of the CCMs. The MM would require a more elaborate microphone array arrangement or additional software for accurate triangulation. In addition, the CCM is faster and more efficient as it focuses on a specific area allowing us to easily filter out irrelevant audio. MM would require additional signal processing to filter out irrelevant audio since it picks up a wider range of audio signals. This will slow down SBB's reaction time. As a result, we decided to proceed with the CCM.

5.1.4 Display screen

For the display screen, we considered alternatives among OLED, LCD, and DCI. We want the display subsystem to be cheap yet durable, while allowing for clear visuals to users. The most important criteria we were looking for was latency, as the display has to be able to sync with the Raspberry Pi or the web app in a low latency manner. When choosing among the displays, we considered several factors such as the refresh rate, display lifespan, and cost.

Ultimately, we selected the DCI display as it offers optimized refresh rates (not a very high rate which would require more power and heat up the RPi) and very fast response times with regards to latency, which is critical for interactive features like the rock-paper-scissors game. Furthermore, the low power consumption of DCI technology makes it an ideal choice for a stationary desk robot that needs to run for extended durations. Additionally, we design with durability in mind and the DCI interface should be able to withstand constant user interaction combined with the servo motors moving the screen while offering quality visual performance.

5.1.5 Speaker for TTS

We have decided to use a mini speaker computer speaker because it contains a USB cable that can feed both audio and electrical power to the speaker. This allows for an easier connection to the RPi. The speaker runs with an input voltage of 5 Volts and has a maximum output power of 0.5 Watts. This can easily be supplied by the RPi. In addition, the speaker is lightweight, and perfectly fits within our desired body dimension. This will give the user the experience that the robot is speaking to it without seeing the speaker, making the human-computer interaction smooth. It has a 2 Millihertz frequency response which is fast enough for the TTS generated.

5.2 Software

5.2.1 WebApp

We settled on using a WebApp as the companion app to the robot because a WA can run on any device with a web browser, making it widely accessible. This allows users to access their study sessions from various devices without the need for installation, which aligns with our design goal of facilitating study sessions easily. However, the WA may not be able to provide the same level of user experience as mobile apps, particularly in terms of interactions and animations. Furthermore, native phone applications typically perform better due to optimized usage and direct access to device hardware. This could help us achieve our strict latency requirements more effectively than a WA, but we ultimately decided on a WA because of its cross-platform compatibility, and its greater ease of development. For mobile apps, performance tuning can be more challenging due to hardware limitations and the need for efficient memory usage. There are also other factors to consider such as battery consumption, and local storage optimization that are more serious issues on phones than computers.

5.2.2 WebSocket

Using WebSockets (WS) for communication between the web app and the robot offers some substantial advantages, particularly in meeting the strict latency requirements outlined in our use case. WS enable real-time, full-duplex communication, this allows us to provide immediate updates such as starting a stopwatch on the robot or reflecting a "Break in Progress" status in under 250 milliseconds on the WA. In comparison, our other possible option of using REST APIs operates on a request-response model, which can lead to higher latency due to the overhead of establishing new connections for each interaction. While WebSockets may require a more complex implementation to handle stateful connections, they ultimately provide a more efficient and responsive solution for applications demanding frequent updates like ours. This complexity may pose integration challenges, but our WA design is simple overall and thus this should not pose a significant problem. We made the final decision to use WebSockets because of its better alignment with our design goals of maintaining responsiveness for study session management and we will handle the trade-offs associated with development complexity carefully.

5.3 Build Material

To build SBB's body with considerations of aesthetics, reliability and sustainability in mind, we chose between acrylic and hardboard wood. Both of these are aesthetically pleasing depending on the object design, however, one is more environmentally friendly than the other. Hardboard wood is a biodegradable material made of compressed wood fibers, and acrylic is a plastic derived from petroleum. In addition to this, hardboard wood is more resistant to impact compared to acrylic which is prone to cracking or shattering under stress. SBB' use case involves interactions with students and potential knocks during the study sessions. As a result, durability is crucial. Hardboard's higher impact resistance is a better option as a result. Acrylic's brittle nature makes it a less ideal solution. We chose to proceed with hardboard wood, for the reasons mentioned above and so that we get a warmer and more aesthetically pleasing study buddy.

6. System Implementation

6.1 Audio cue response

To detect the double clap audio cue, each microphone will continuously record audio from the past 1.5 seconds. The system is designed to parse this audio in real-time to identify the specific sound of a double clap. For each microphone, the process begins by analyzing the most recent 1-second audio segment. This segment is divided into two halves, and a correlation is performed between the first and second halves. The reasoning behind this is that the two claps are expected to have similar intensities, allowing us to effectively differentiate between a single clap and the double clap we are looking for.

Before performing the correlation, we apply a bandpass filter within the range of 2.2 kHz to 2.8 kHz to focus on the frequency range where the clap sound is most prominent.

This filtering process helps to remove background noise and unnecessary frequencies, improving the system's ability to focus solely on the clap. To ensure no clap is missed, the system processes audio in 100ms intervals, consistently analyzing each segment for the double clap pattern.

When a signal matching the filtered frequency range is detected, we use threshold correlation to confirm that the two halves of the audio contain the distinct sounds of two claps, rather than just one. Once the double clap is identified by one microphone, we begin cross-correlation between the audio inputs from all four microphones. This allows us to calculate the time difference of arrival (TDOA) for the sound at each microphone. Since sound travels at approximately 343 m/s in air at room temperature, we can use the small differences in time that it takes for the sound to reach each microphone to determine the direction of the sound source.

The TDOA is calculated by measuring the time difference between when the sound reaches different pairs of microphones. This is done using the formula:

$$\Delta t_{ij} = t_i - t_j$$

where Δt_{ij} represents the time difference between microphones i and j, and t_i and t_j are the times when the sound was detected at each microphone. With four microphones, we can calculate six time differences, which allows us to accurately triangulate the direction of the sound.

To ensure the robot only responds to nearby sounds, we limit the audio detection range to a 3-foot radius. By measuring the TDOA, we can localize the sound's origin within this radius, and the robot will turn its head to face the direction of the detected clap. Since all microphones are connected to the same Raspberry Pi and share a synchronized clock, there are no issues with synchronization, ensuring accurate TDOA calculations.

This approach provides a reliable way to detect the direction of the double clap sound, enabling the robot to respond dynamically and turn its head in the correct direction.

6.2 Study Sessions (SS)

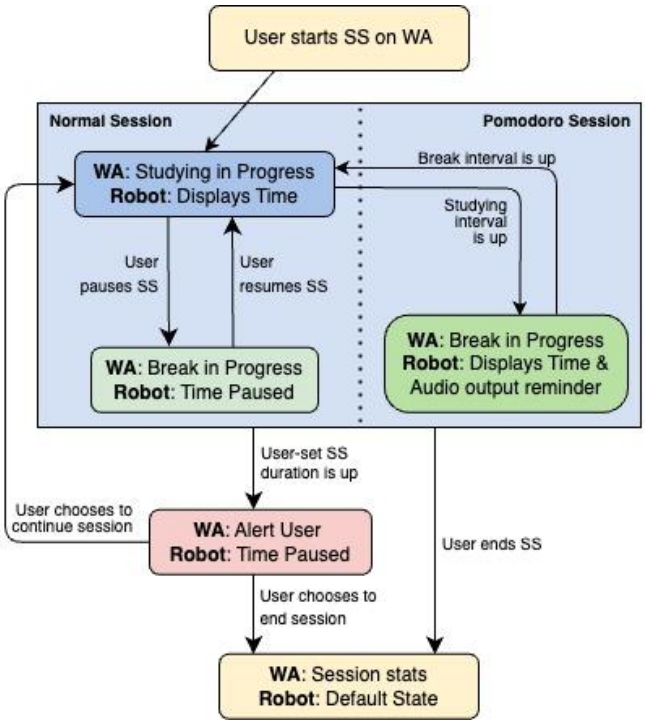

Figure 3: Study Session FlowChart

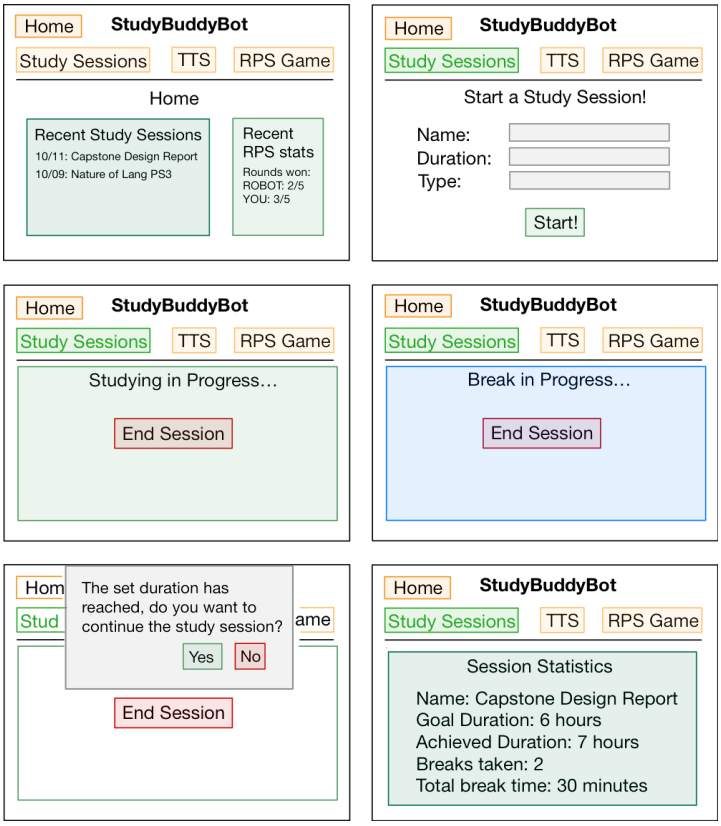

Figure 4: WebApp Mock-ups for Study Sessions

The flow chart for Study Sessions, as seen in Fig 3, shows the overall process of a user starting and ending a

study session. When the user starts the Study Session on the WA, they set the duration and type (Normal or Pomodoro). If the user starts a Normal SS, then it brings them to a Studying in Progress state where the WA displays "Studying in Progress…" and the robot displays the elapsed time for the study session. If the user pauses the session by pressing a pause button on the robot's foot, then it transitions to a Break in Progress state, where the WA will display "Break in Progress…" and the robot pauses the time display. When the user resumes the SS by pressing the same button again, it transitions back to the Studying in Progress state. At any point during a SS, the user can choose to end it by clicking "End Session" on the WA (both Studying in Progress and Break in Progress pages will have this button), which will cause the robot to stop the time display and revert to the default state while the WA displays the relevant statistics for the SS (Goal duration, Achieved duration, No. of breaks, Total break time). When the set SS duration is up, the WA will have an alert pop-up notifying the user and asking them if they wish to continue with the SS. If the user chooses to continue, they will click "Yes" on the WA and that will bring them back to the Studying in Progress page, and the paused time display will continue again. If they choose "No", then the SS ends in a similar fashion to if the user clicks "End Session".

For a Pomodoro SS, on top of the user-triggered Break in Progress state, there is an extra state of Break in Progress triggered by the study interval ending. The WA and robot should automatically transition from the Studying in Progress state to Break in Progress state if the set studying interval is up, together with audio output (that will come from the WA sending a text-to-speech input "Please take a break, you've been studying for _ hours!" to the robot) from the speaker to remind the user to take a break. When the set break interval is up, the WA and robot should also accordingly transition back to the Studying in Progress state.

This will be achieved through frontend pages that will be written with HTML and CSS that will define the layout of the pages that the user will interact with. On the backend, we will leverage Django's built-in features to handle the core functionality. Study Sessions will be stored using Django models and forms, which will allow the app to keep track of every past SS and its associated statistics. WebSockets will be used for communication between the robot and the WA, which will be kept simple and never contain more information than an updated state.

6.3 Text-to-Speech (TTS)

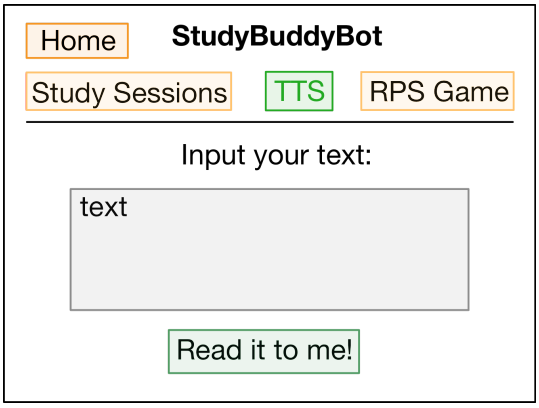

Figure 5: WebApp Mock-up of TTS page

To support students that are audio learners, our text-to-speech functionality involves them navigating to the TTS page on the WA, inputting the text into the provided text field, and clicking send. The robot should then output the text as audio through its speaker. This process is achieved through the WA sending the input text via WebSockets to the RPi, which utilizes the python library pyttsx3 to process the input text, and outputs the appropriate audio on the speaker. pyttsx3 is a Python library that provides a simple interface for text-to-speech conversion, and can convert any text into spoken language while allowing customization for speech rate and volume [8]. This is the core of our TTS feature, and we will focus on implementing it on the RPi with the connected speaker as audio output and fine-tune any problems related to volume and speech rate.

6.4 Rock-Paper-Scissors Game (RPS Game)

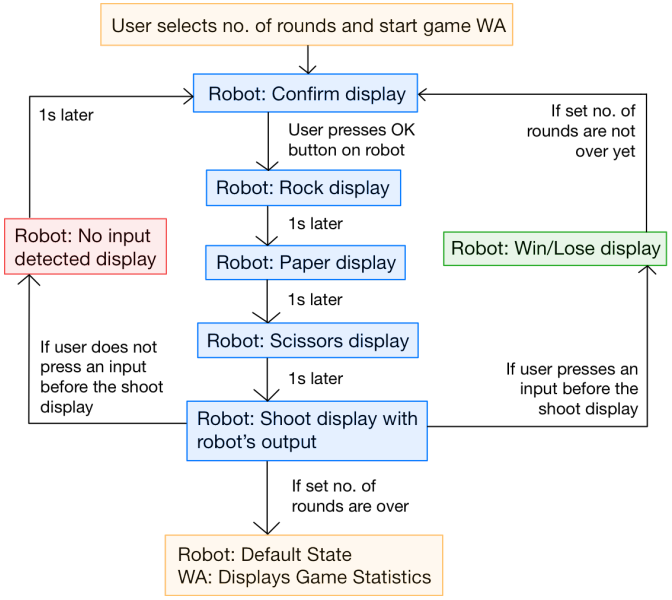

Figure 6: RPS Game FlowChart

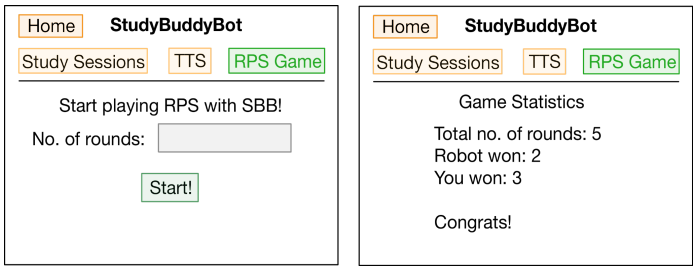

Figure 7: WebApp Mock-up of RPS Game page

With Fig 7, we can see the flowchart illustrating an overview of how the user can start the RPS Game on the WA and how the different states can flow from one to the other. The user starts on the WA, inputting the number of rounds of RPS they would like to play, which sends this information to the robot via WebSockets. The robot DCI Display then shows a confirmation message ("Let's play Rock-Paper-Scissors! Press OK to start") and when the user presses the OK button on the foot of the robot, the Robot displays the word "ROCK". One second later, the display changes to display the word "PAPER", and then "SCISSORS" one second later. After the "SCISSORS" screen is displayed, the user has to press one of the RPS buttons on the robot's foot. A second later, the robot will display the "SHOOT!" message, along with an image of the robot's choice (rock, paper, or scissors). If the user did not manage to press a button before the "SHOOT!" message is displayed, the DCI display should display the message "No input received. Let's try this round again!", and one second later, the confirm display for playing a game should appear again. If the user does press the input button in time, the robot display should transition to show either "You Win!", "You Lose!", or "Tie!" messages, and if the set number of rounds is up (for e.g. if the user selected to play 3 rounds of RPS and the third round is over), then the robot transitions back to default state (happy face), and the WA should receive the game results from the robot and display them as seen in Fig 7.

Initially, the WA sends the RPS game information over to the robot via WebSockets. Then, most of the RPS game logic occurs on the RPi, whereupon receiving the number of rounds, will generate a list of RPS outputs with the number of elements in the list corresponding to the number of rounds. Then, at the shoot display page, the list would be indexed based on the round number and the robot output will be retrieved to be displayed. Based on the user input received via the GPIO pins, the appropriate win/loss/tie game state will be calculated and based on the current round number the RPi will determine whether or not the game should continue. The RPi also keeps track of the stats for each round and if the game ends, the RPi then sends the information back to the WA via WebSockets which allows the WA to display RPS Game Statistics. After sending this information, the robot then goes back to its default state.

7. Test, Verification and Validation

7.1 Unit Testing

7.1.1 Testing for Audio Cue Response

To test the audio input, we will start by placing the microphones in a quiet room. We will try pinpointing the audio direction from the closest possible distance. After this, we will proceed to wider radii from the robot. We will test audio from 8 different directions following the standard compass directions. This should give us a very reasonable and exhaustive testing option. A successful test outcome would consist of getting an angular accuracy of ± 5 degrees from the audio source in a maximum of 3 seconds. Once we confirm it works in a quiet environment, we will proceed with a slightly more noisier environment to make sure we are filtering out unnecessary noise appropriately approximately 90%. We will be using a bandpass filter to get rid of the sound outside of the 2.2kHz and 2.8kHz[6]. The noise we aim to reduce are that of conversational sounds. The product is intended for English language speakers. Given that English is a non-tonal language, research suggests that a man's voice frequency is between 100 to 120Hz while a woman's average range is between 200-220 Hz[5][7]. As a result, the bandpass filter should isolate the ambient conversational noise by 100% accuracy. In addition, we will test double clap detection versus other clap frequencies to make sure we detect the double clap only, in an interval of 4 seconds at a time. Therefore, once a double clap is detected, we will not be registering another clap until the robot's head has turned to the audio source. We will also make sure to calibrate each microphone to ensure uniform audio sensitivity across all mics. This way, the computation will be accurate and no individual microphone will be disproportionately affecting the localization accuracy.

7.1.2 Testing for Study Session

To test our Study Session functions on the WA, we will start with small unit tests to ensure that the overall functions work. This will be done by adding a Pause button on the Studying/Break in Progress pages to simulate the user pausing the SS on the robot and seeing if the page correctly transitions from Studying to Break and vice versa. We will also add a Duration Reached button on the Studying/Break in Progress pages to simulate the robot informing the WA that the SS duration has been reached to test the alert system. We will of course test the buttons and fields that should already be there like the duration and type on the start page by and the end session button on the Studying/Break in progress pages by making the browser console display the appropriate information. For Pomodoro SS, we will test the break reminder function by having an additional Stud7/ying time is up/Break time is up button on the relevant pages to simulate the studying/break interval being reached, and check that the WA responds accordingly

by printing the break reminder text in the browser console.

7.1.3 Testing for TTS

To test our TTS function, we will write a simple Python testing function on our RPi that takes in a test input string and see if the speaker outputs the correct audio. We can also test the rate of speech and the volume, which are options available to tweak with the pyttsx3 - Text-to-speech x-platform library we plan on using to provide the most optimal experience for the user. Based on our current research, there seems to be no upfront latency associated with length of texts, as pyttsx3 seems to process the text in segments and starts converting the text to audio almost immediately, but we will observe this as we test to find any latency issues for longer texts and set a word limit if necessary.

7.1.4 Testing for RPS Game

To test the RPS game, we will start with verifying the game logic on our local machines to make sure that they work exactly as expected, following the game rules of RPS (We want to be able to determine if the user wins, loses, or ties). After this, we will integrate it with the RPi, and associate the inputs of the user with the RPi GPIO pins through the buttons. We will test to make sure that any button presses the user inputs during the “Rock Paper Scissors Shoot” countdown will be registered. However, the only value we take in and process is the first pressed value. The RPi will then take the user input choice to compute the winner, while also keeping track of the game score to later be sent via web sockets to the WA. If the user doesn’t input any value, we make sure that SBB gives the user the option to continue the game or not. We can test this functionality by loading in 1+ rounds and breaking down the test cases into fully finishing multiple rounds of RPS, pausing midway through multiple rounds, and finishing just one round, and record if the behavior observed matches our system implementation diagram and expected outputs.

7.2 Integration Testing

7.2.1 Testing for WebSocket Communication Latency

To verify that our WebSocket communication can meet the design requirement of ≤250ms latency, we will conduct latency tests for the study session feature. Specifically, we will measure the time from when the user starts a study session on the web app to when the time display appears on the robot. This will be done through WebSocket timing logs on the browser console to timestamp each communication point: initiation, transmission, and robot display. We will also try to simulate various network conditions (e.g., local V.S. public networks, low V.S. high bandwidths), so that we can determine if our system consistently meets the latency requirement. The test output will indicate whether total latency (including WebSocket transmission and robot processing) is under the 250ms threshold, therefore aligning with our design specification requirement.

7.2.2 Testing for Hardware Threading

RPi uses a linux operating system allowing for seamless hardware threading experience. To ensure all our systems run smoothly, we will start by powering on the robot and starting a study session(SS). During the SS, we will input a text for TTS output. We will make sure that the TTS works while the SS is in progress. Then, we will make sure that if a pause timer input to the robot is pressed, the text-to-speech and the timer pause in this order within 100ms. Then when the timer is restarted, the SS clock should continue and text-to-speech should continue in this particular order. The latency when restarting is not a concern since we will not be losing time or TTS data. After this, we will do a double clap while a SS is happening. The robot should only turn to the user while a SS break is in progress. If a TTS is in progress during the SS, the robot should not be able to turn as there is a high chance of interference from the speaker output. If a game is in progress, SBB shouldn’t be able to perform other processes such as TTS and SS since it is an exclusively SS break activity.

8. Project Management

8.1 Schedule

Please refer to Figure 10 for the full details regarding our schedule.

8.2 Member Responsibilities

Jeffrey is responsible for working on our robot’s power on and greeting user feature, the DCI display, as well as the speakers. He will also be building the robot’s base with Mahlet and working on WebSockets with Shannon.

Mahlet is responsible for the robot’s audio cue response feature with the microphones and the servo motors, as well as building CAD models for our robot. She will be in charge of audio triangulation and robot neck movement. She will also be working on TTS with Shannon.

Shannon is responsible for all WA-related functions (both frontend and backend), and will be in charge of the SS feature, and the TTS feature with Mahlet.

All members are involved in the RPS game feature, with Shannon working on the WA handling the starting and number of rounds and the RPi computation and display done by Mahlet and Jeffrey.

8.3 Bill of Materials and Budget

Please refer to Figure 9 for the thorough breakdown of the bill of materials along with budget information and details about components used. We have the remaining budget amount and the used amount in the table.

8.4 Risk Mitigation Plans

For our WebSockets implementation, there is a risk of communications not working out as smoothly as we would like and the actual latency being higher than what we expect. In this case, we have a back-up of directly hosting the WA on the RPi itself, so that if WebSockets prove to be a challenge, we can quickly pivot to using the RPi to host. It should have a lower network latency because all communication between the WA and the robot can occur locally but is not ideal as it could significantly slow down the system and affect our other features that require RPi computation from functioning as smoothly.

The audio input through the microphone presents the risk of not being accurately detected. Since the microphones are going to be very close to each other(placed at the four corners of SBB's body), the time delay between each microphone will be very small. To reduce this risk, we will fine-tune the sampling rate of the microphone from 44.1kHz and iteratively make it better depending on testing outputs. This will improve the precision of time difference measurements between each microphone. Another risk mitigation is placing the microphones in a slightly wider rectangle to increase the time delay difference between each mic.

The audio input through the microphone present the risk of not being

Our TTS feature has a risk of significant lag/delay between inputting text from the WA and hearing the audio over the speaker on the robot. To mitigate this, if the latency proves to be higher than ideal or the speaker on the robot seems to have issues relaying the audio information, we will aim to implement the TTS feature directly on the user's laptop with the WA such that the WA will run the pyttsx3 library and produce audio output from the user's computer instead of the robot's speaker, which might help in terms of latency issues, but would make the robot slightly less interactive.

Overall, our project also includes some parts that no one in the team has prior experience with, such as the python text-to-speech library we plan on using (pyttsx3), but this risk will be mitigated by doing research of our own based on the extensive documentation on the library, and possible reaching out to

9. Related Work

There is a plethora of literature on personal assistant robots that are designed to enhance productivity and focus in a work or study environment. One example is Stubie, which is a desktop robot designed to provide emotional support to encourage good studying habits by acting as an agent that can monitor student studying. Its main purpose is to use facial expressions on an OLED screen to "provide comfort and happiness to students so studying does not have to be lonely" [10]. We expand on that idea within our design, while focusing more on positively reinforcing structured study sessions with features such as timers, break reminders, and engagement via rock-paper-scissors. We want to design a social robot experience that can be more interactive with the user, emphasizing rapid user feedback to reinforce the user interaction experience.

For related works with our interactive feature of rock paper scissors, there is a humanoid robot called RASA that was trained on a Markov Chain model to play rock-paper-scissors against humans [11]. The robot's goal was to predict the player's next hand gesture based on prior information, then the robot will make the hand gesture via kinematics and also display on a screen what they play against the user, for transparency. Our design simply makes a decision among rock, paper, or scissors, as opposed to having an algorithm to make the optimal choice.

In our studying features, SBB incorporates a timer system similar to the Pomodoro Technique, which is a popular time management method that typically calls for 25 minutes of focused work followed by a short break [12]. With our design, users can customize their study and break durations based on their preferences from the web app. This flexibility allows for personalized study sessions, where the robot tracks time, notifies the user when to take a break, and helps maintain focus by alternating between work and rest periods, allowing for better concentration and productivity.

10. Summary

The StudyBuddyBot aims to provide support to students of all ages and help them achieve a good work-life balance by promoting healthy studying habits and helping to relieve stress, thus promoting public health and safety. By focusing on having both studying features and interaction features, we want our users to feel that our robot is truly like a friend that they can not only rely on to help foster good studying habits but can also depend on for relaxing together while taking a break.

Moving ahead, our team expects to develop the four subsystems of the robot to ensure that it can perform all of the features that will be provided to the user and have them seamlessly operate with one another. Although there may be difficulties encountered when trying to integrate the subsystems together and we may have to debug unpredictable challenges, we have confidence that our risk mitigation plans will help us to resolve any issues. In conclusion, we hope the StudyBuddyBot becomes a promising prototype that can better the lives of all students.

Glossary of Acronyms

SBB - Study Buddy Bot
WA - WebApp/Web Application
DCI Display - Display Control Interface Display
RPi – Raspberry Pi
TTS - Text-to-speech
SS - Study sessions
RPS - Rock-Paper-Scissors

References

[1] Carson, Oliver. “How Fast Is Real-Time?” PubNub, 8 Mar. 2024, www.pubnub.com/blog/how-fast-is-realtime-human-perception-and-technology/.

[2] Intriago, Rhett. “How to Change Siri Voice Response Time.” Www.Iphonelife.Com, 23 Sept. 2024, www.iphonelife.com/content/how-to-change-siri-voice-recognition-response-time#:~:text=Scroll%20down%20and%20tap%20Siri,roughly%20four%20seconds%20before%20responding.

[3] Chuprin, Barys. “WebSocket Data Feed Latency.” EmberDocs, ember.deltixlab.com/docs/performance/ws-market-data/. Accessed 10 Oct. 2024.

[4] GeeksforGeeks. (2024, January 18). How to measure time taken by a function to execute using JavaScript ? GeeksforGeeks. https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-function-to-execute-using-javascript/

[5] DPA Microphones. “Facts about Speech Intelligibility.” *DPA Microphones University*, https://www.dpamicrophones.com/mic-university/background-knowledge/facts-about-speech-intelligibility/

[6] National Arts Education. “Mapping Clapping: Five Months of Claps.” *NAE*, https://www.nae.org.uk/news/blog/mapping-clapping-five-months-of-claps/

[7] Science direct. “ The Speaking Voice in the General Population: Normative Data and Associations to Sociodemographic and Lifestyle Factors” https://www.sciencedirect.com/science/article/pii/S0892199716300558

[8] Srivastava, P. (2023, October 23). Text-to-Speech with Python: A Guide to Using pyttsx3. Medium. https://prateeksrivastav598.medium.com/text-to-speech-with-python-a-guide-to-using-pyttsx3-417f5787bb0f

[9] Raspberry Pi. "Documentation." *Raspberry Pi Foundation*, https://www.raspberrypi.com/documentation/

[10] Lee, J., Clouse, C., & Kazi, A. (2024). Interactive Desktop Study Buddy Robot: Stubie. Indiana University

[11] Ahmadi, Ehsan & Pour, Ali & Siamy, Alireza & Taheri, Alireza & Meghdari, Ali. (2019). Playing Rock-Paper-Scissors with RASA: A Case Study on Intention Prediction in Human-Robot Interactive Games. 10.1007/978-3-030-35888-4_32..

[12] F. Cirillo. 2009. The Pomodoro Technique. Creative Commons.

[13] Harris, Sarah L, and David Money Harris. Digital Design and Computer Architecture. Amsterdam ; Paris, Elsevier, Cop, 2016.

[14] “Servo Motor.” *Servo Motor - an Overview | ScienceDirect Topics*

[15] “How Do I Power My Raspberry Pi?” *The Pi Hut*

Please refer to this url to get an easily zoomed in view of the block diagram: link

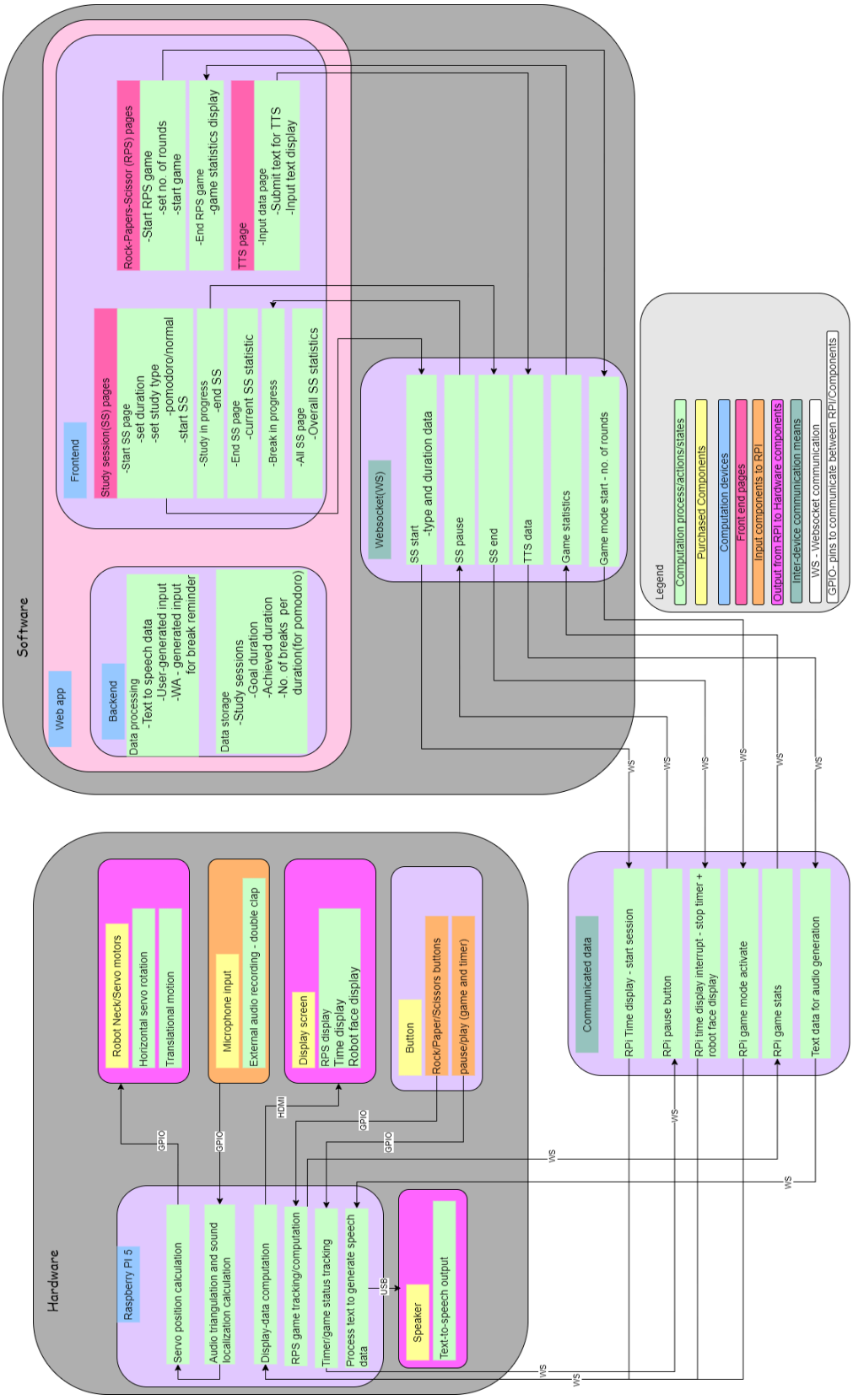

Figure 8: Block diagram of SBB

Part	Model Number/Name	Manufacturer	Vendor	Quantity	System spec(if needed)	Link	Cost per piece	Total cost	Purchase or Borrow?
RPi 5 8GB	ECE inventory num = S2404	-	CMU	1		-	0.00	0.00	Borrow
360 Degree Servo motor	Yosoo Health Gear3gbuaqxd2h	Yosoo Health Gear	Amazon	4		https://www.amaz	9.83	39.32	Purchase
Mini 4-Key Keypad	Sayodevice Arrow Keyboard		Amazon	2		https://www.amaz	19.99	39.98	Purchase
Rode M5 Matched Pair Microphone	M5	-	Amazon	1		https://www.amaz	199.00	199.00	Purchase
Rode M5 Matched Pair Microphone	ECE inventory num = F22083	-	CMU	1		-	0.00	0.00	Borrow
5 Inch Touchscreen IPS MIPI DSI Display	B0CXTFN8K9	Hosyond	Amazon	1		https://www.amaz	26.39	26.39	Purchase
USB Mini Speaker Computer Speaker Powered Stereo Multimedia Speaker	H002	HONKYOB	Amazon	2		https://www.am	10.95	21.90	Purchase
Hardboard wood	N/A	N/A	TechSpark	2	Thick: 1/4 in, Dim: 12x24 in		4.00	8.00	Purchase
							Total price:	334.59	
							Budget:	600.00	
							Remaining:	265.41	

Figure 9: Bill of materials and budget updates

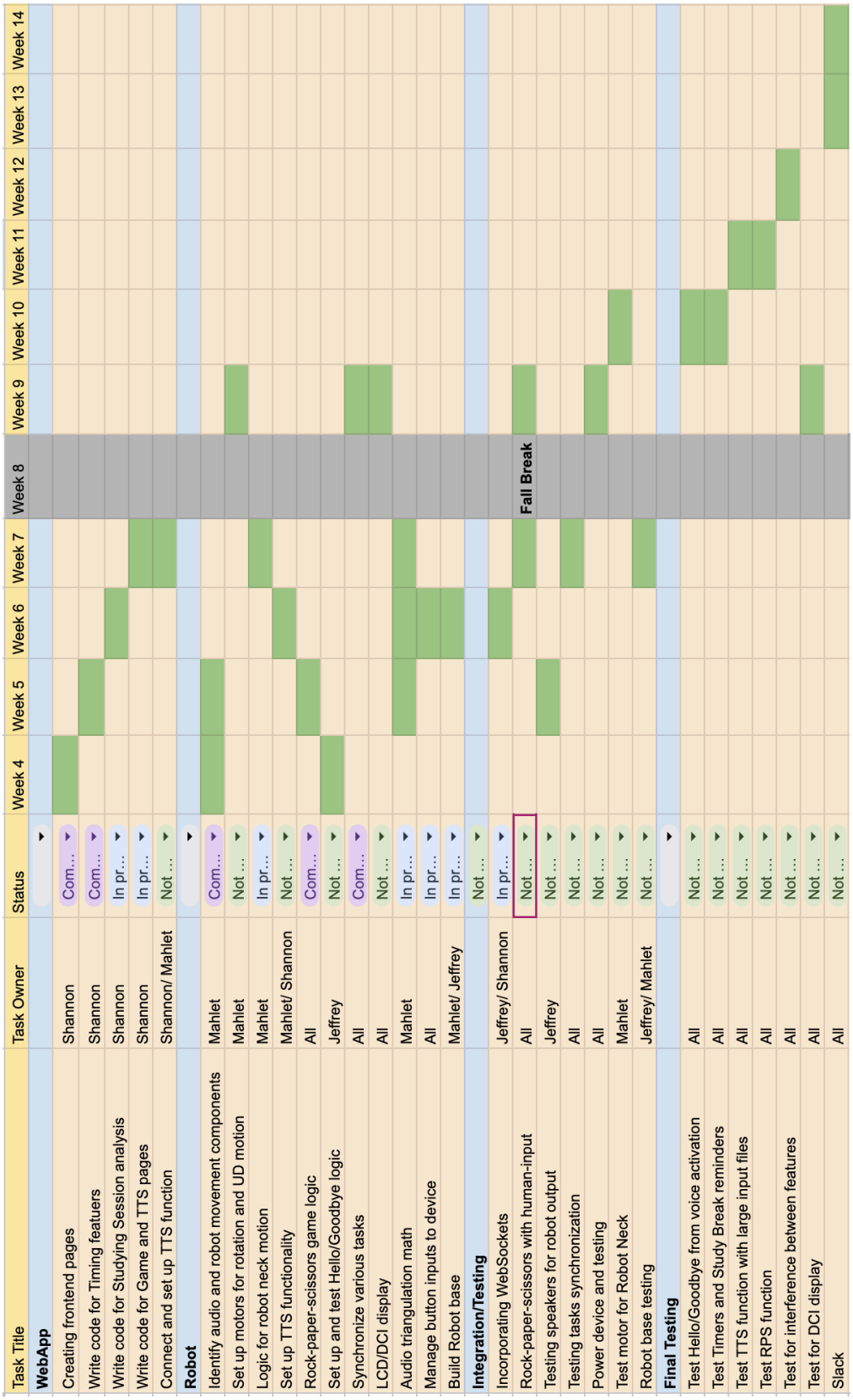

Figure 10: Gantt Chart