**SANTA CLARA UNIVERSITY**

**DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Date: June 9, 2025

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

**Julius Gamboa**
**Muti Shuman**
**Tro Hovasapian**

ENTITLED

# The GRD Companion

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

**BACHELOR OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING**
**BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING**

Thesis Advisor: **Dr. Radhika Grover** Date

Thesis Advisor: **Dr. Ahmed Amer** Date

ECEN Department Chair: **Dr. Shoba Krishnan** Date

CSEN Department Chair: **Dr. Silvia Figueira** Date

# The GRD Companion

by

Julius Gamboa
Muti Shuman
Tro Hovasapian

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Electrical and Computer Engineering
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 9, 2025

# The GRD Companion

Julius Gamboa
Muti Shuman
Tro Hovasapian

Department of Electrical and Computer Engineering
Department of Computer Science and Engineering
Santa Clara University
June 9, 2025

ABSTRACT

This project introduces the design and development of the GRD Companion. This gesture-controlled reminder device aims to support individuals with special needs in their daily routines, in the hope of them being more independent. The system we designed integrates a Raspberry Pi Zero 2 W, PAJ7620U2 gesture sensor, and audio playback components, all housed in a custom 3D-printed case. Some of our key design priorities included accessibility by relying only on gesture recognition to control the device, eliminating screens and buttons, as well as portability, enabling all users to carry the device anywhere. The device is paired with our WaveLink App, which allows teachers and caregivers to record personalized audio tasks for the user and send them to the device when paired via Bluetooth or WIFI hotspot. We chose to code using Python for gesture recognition and a scheduling logic to manage the hardware interactions within the device. The project emphasizes affordability, usability, and adaptability to help support teachers and caregivers in aiding individuals with special needs. Human subject testing was conducted to gain feedback to help guide improvements and confirmed the functionality and reliability of our device. Future iterations include expanding on customization features such as custom gesture recognition, a reward system, and real-time task completion.

## Acknowledgements

We would like to express our gratitude to everyone who supported us throughout our senior design project. We first would like to thank our advisors, Dr. Radhika Grover and Dr. Ahmed Amer, for their guidance, feedback, and encouragement during the development of the GRD Companion. We would also like to thank Independence Network in Santa Clara, CA offering a facility to test our device and helping us find a participant to test our project to help us shape what GRD Companion is to truly meet user needs. Finally, we would like to thank the Department of Electrical and Computer Engineering and the Department of Computer Science and Engineering at Santa Clara University for providing the resources and facilities needed to complete this project.

# Table of Contents

# List of Figures and Tables

**Tables**

# Chapter 1
# Introduction

## 1.1 Problem Statement

Individuals with special needs often face challenges in completing daily task. These challenges require constant support from their teacher or caregiver. This becomes a problem because there aren't enough teachers or caregivers to help out every individual with special needs facing these challenges.

There is a need to have more accessible, simple, hands-free solution that would help individuals with special needs receive daily reminder tasks. Teachers and caregivers also need this as a reliable way to program and manage reminders without being physically present with the individual with special needs. This allows a supportive environment for individuals with special needs that will foster independence and reduce dependency on people helping them.

This project proposes we create a reminder device that provides a personalized voice reminders. We also need to design a program to go along with the device that the teacher and caregiver can use to program the personalized voice reminders. The device that we create should be designed to be portable, screen-free, and easy to use to ensure accessibility for the user and flexibility for the teacher and caregiver programming it.

## 1.2 Background

Many special needs adults may face daily challenges like communicating with others and focusing on tasks they must do for the day due to their intellectual disability and behavioral difficulties. Therefore, they need much more attention from a teacher or caregiver to receive help. One option that special needs adults can do is to attend a school catered towards helping special needs adults to help them gain an education, learn to do simple tasks, and hopefully become independent [1]. Teachers assist special needs adults at school, but there aren't enough teachers or time to help all the special needs adults simultaneously. According to the California Department of Education, there has been a consistent shortage of teachers in special education because of high turnover, a lack of qualified teachers, and credential issues in special education [2].

There have been many reminder products that are on the market that help out with special needs adults be reminded of daily tasks.

- Personal Voice Operated Reminder System [3]
- Reminder Actioning System [4]
- Location-Based Task Reminder [5]
- Multifunction Reminder System [6]
- Reminder Device Wearable by a User [7]

These reminder products aim to help people who require reading and interpreting text, small buttons, and screen-based prompts. For many special needs adults, it may be difficult to use these reminder products because of the tools being confusing and overwhelming making it unusable for them.

My group was motivated by a vision to create a device that would provide care support for special needs adults while keeping in mind helping the special needs adults around them to show care to one another [8]. We hope to end the project by creating a device we envision to help special needs adults do everyday tasks and become independent, in addition to helping the people around the special needs adults, such as their teachers and caregivers, take care of them.

## 1.3 Objectives

Based on our background and problem statement, we concluded that the device to be created should revolve around 4 objectives that need to be completed for the project.

| Objectives | Description |
|---|---|
| **Gesture Recognition** | ● Allows a different input method that reduces buttons or screens, which could be challenging for special needs.<br>● The device should be able to perform key functions by recognizing gestures to promote ease of use. |
| **Inputting and Outputting Tasks** | ● The device should be able to receive audio tasks set by the teacher and caregiver.<br>● Once the audio task is saved in the device, it should output tasks in the correct order it was inputted at the given time or when commanded. |
| **Affordability** | ● Design a device using low-cost components to keep the total build cost within a budget-friendly range.<br>● Ensure that our device can be assembled our way, that is, low-cost cost making it practical for schools and families to have. |
| **Portability** | ● The device needs to be compact and portable, allowing users to carry the device anywhere. |

***Table 1.3.1: Objectives Needed to be Accomplished***

## 1.4 Our Approach

As we look at our objectives, we have created a proposed solution to accomplish all of our objectives. Our proposed solution includes:

- Gesture Sensor

- Low-cost computer
- Software for the reminder application
- Speaker
- Battery
- Custom 3D-Printed Case

As we started the project, we defined what our goals were to make this project successful.

- Create a device to aid individuals with special needs with everyday life.
- The device's primary use is to have tasks be told to the user at a specific time or commanded by gesture recognition.
- Create software for teachers and caregivers to easily input tasks for the user to do and send those tasks to the device via WIFI or Bluetooth.
- The device must be compact to be portable everywhere the user goes.
- Test the device with 1-2 individuals with special needs and improve the device from the time of testing to the end of the project.

In the following sections of our paper, we will go over in detail the development and evaluation of our device, the GRD Companion. Chapter 2 outlines how we chose hardware components and software languages to make our device. Chapter 3 reviews existing products and explains how our solution differs, as well as testing and development of the GRD Companion and our WaveLink App. Chapters 4 and 5 discuss our testing protocols and the results of testing the GRD ourselves as well as with a human subject. Chapters 6 and 7 talk about the constraints, standards, and ethics considerations in the project. Finally, chapter 8 presents our project conclusion and discusses future iterations of the project.

Chapter 2

# Hardware and Software

In Chapter 2, we will discuss the hardware and software chosen for the project and how they contribute to achieving the goal of our project.

## 2.1 Hardware

This section explains the choices we made in choosing what hardware and components we would use to create our device. To help make decisions, we used a decision matrix table for the hardware and some components to show the factors when creating the device.

### 2.1.1 Raspberry Pi Zero 2 W

As stated in section 1.3, we need to have a computer to run the software we are creating, but we have to keep in mind that it needs to be low-cost. Since we couldn't use a normal desktop or laptop for our project, we started to look at microcomputers. The microcomputers would allow us to attach components necessary for our device, like receiving audio tasks from the application and recognizing different gestures detected. Below is our decision matrix on how we choose the computer for our device. We needed to factor in the cost, number of ports, size, and performance when choosing the right computer for our project.

| **Criteria**<br>(Scale 1 (Worst) - 5 (Best)) | **PYNQ-Z1** | **Arduino Nano** | **Raspberry Pi 5** | **Raspberry Pi Zero 2 W** |
|---|---|---|---|---|
| **Cost**<br>(To be affordable, materials need to be low-cost) | **1** | **5** | **2** | **5** |
| **Ports**<br>(The computer needs to have enough input ports or pins to attach necessary components) | **4** | **2** | **4** | **3** |

| | | | | |
|---|---|---|---|---|
| **Size**<br>(Needs to be small enough to be portable) | **2** | **5** | **2** | **4** |
| **Performance** (Must have the capability to run software smoothly) | **4** | **1** | **5** | **3** |
| **Total** | **11** | **13** | **13** | **15** |

*Table 2.1.1: Analysis of Alternatives for Choosing Our MicroComputer*

We decided to choose the Raspberry Pi Zero 2 W because we thought it would be the best small computer to fit our objectives. The Raspberry Pi Zero 2 W is low-cost and small, so it can be brought everywhere by the user. The main feature that we liked was that there were enough pins for us to attach all components, like our gesture sensor which allows us to run software that we created.

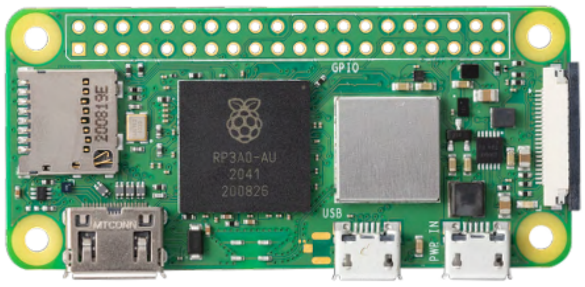


*Figure 2.1.2: Raspberry Pi Zero 2 W [9]*

## 2.1.2 PAJ7620U2 Gesture Sensor

Gesture recognition is one of the main features of our project, so we need to find a piece of hardware that works with the Raspberry Pi we have chosen. We had two different options to consider: a camera or a sensor. A camera would be the most optimal, as we would be able to incorporate machine learning to recognize different hand gestures, but we would run into a constraint. Privacy is a big concern for people as the camera would capture data for the device, and we as a team didn’t feel comfortable allowing a camera on the device. Therefore, we decided to go the sensor route when choosing a component for gesture recognition.

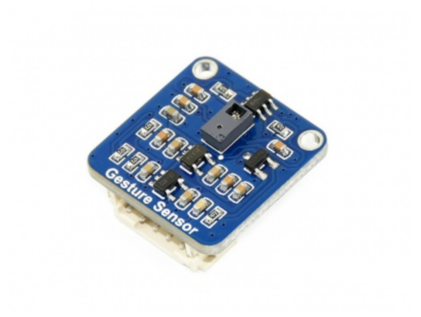


***Figure 2.1.3: PAJ7620U2 Gesture Sensor [10]***

The best option we found was the PAJ7620U2 Gesture Sensor. This specific gesture sensor allows simple gestures to be recognized like: Up, Down, Left, Right, Forward, Backward, Clockwise, AntiClockwise, and Shake. With this gesture sensor being able to recognize 9 hand gestures, we can utilize these gestures to make the device do different things, like moving through the task lists and connecting to Bluetooth.

### 2.1.3 Pisugar 2 Portable 1200 mAh UPS Lithium Battery

We want a battery that is compatible with the Raspberry Pi Zero 2 W and can run for a period without plugging the device into an external source. We looked at different batteries, and our decision matrix helped us determine the battery to use. Here are the factors we took into consideration when it comes to choosing our battery: size, voltage, battery life, and charge rate.

| **Criteria**<br>(Scale 1 (Worst) - 5 (Best)) | **UPS Hat Battery** | **PiSugar2 Portable Battery** | **Zero2Go Omini** |
|---|---|---|---|
| **Size**<br>(Needs to be small enough to be portable) | **5** | **4** | **2** |
| **Voltage**<br>(The voltage of the battery should be safe anywhere) | **4** | **4** | **5** |

| | | | |
|---|---|---|---|
| **Battery Life**<br>(Battery needs to last at least a day on a single charge) | **4** | **5** | **5** |
| **Charge Rate**<br>(Needs to charge at a significant rate) | **2** | **5** | **5** |
| **Total** | **15** | **18** | **17** |

*Table 2.1.5: Analysis of Alternatives for Choosing Our Battery*

We ultimately chose the PiSugar2 Battery because it meets the needs of the device the most. It also comes with a nice feature of low power mode, which allows the device to use less energy when it's inactive, which will come in handy when no task is outputted for a while. The Pisugar 2 Battery will make the device portable and can be used wherever it is taken.

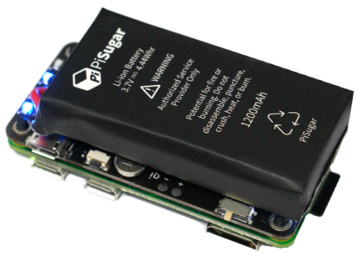


*Figure 2.1.6: PiSugar2 Portable 1200 mAh UPS Lithium Battery [11]*

## 2.1.4 Shutao 8 Ohm 2W Speaker

The speaker is used to output where the user can listen to the recorded tasks when available. When considering what speakers to use for the device, we had to look at two factors: small size and sound loud enough to hear audio. Therefore, we couldn't use a traditional speaker and needed to find a speaker that would best fit our device. We ultimately decided to go with the Shutao 8 Ohm 2W speaker because of its size and sound quality.

*Figure 2.1.7: Shutao 8 Ohm 2W Speaker [12]*

### 2.1.5 BiCool WM8960

For our device, we needed a way to attach the Shutao 8 Ohm 2W speaker chosen in section 2.1.4. The BiCool WM890 HAT was the best choice because it is small and has multiple ways to attach different speakers. It also doesn't limit access to the pins mentioned in section 2.1.2 to put different components, like our gesture sensor. Overall, we think that the BiCool WM890 HAT will add to our device's needs and more.

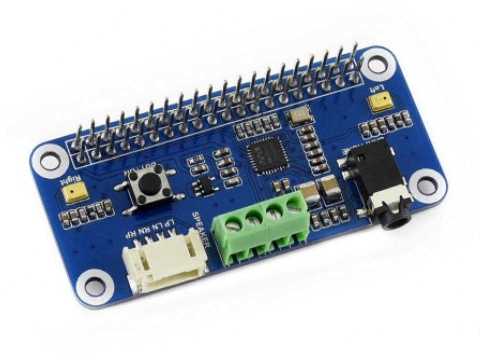


*Figure 2.1.8: BiCool WM8960 Audio HAT [13]*

Based on the proposed solution talked about in section 1.3, we have chosen all the components necessary for our device:

- Small Computer: **Raspberry Pi Zero 2 W**
- Gesture Sensor: **PAJ7620U2 Gesture Sensor**
- Battery: **PiSugar 2 Battery**
- Speakers: **Shutao 8 Ohm 2W Speaker**

- Audio: **BiCool WM8960 HAT**

The hardware and components chosen should work together to create our device and do what we envision it to do. In addition to the hardware, software is as important as it will be the way to communicate with the hardware, as well as create the application to go along with the device.

## 2.2 Software

In this section, we discussed what coding languages were chosen to create the software for the project.

### 2.2.1 Windows Applications: Developing Using C++

For the Windows applications, the programming language used was C++. This is because C++ offers great efficiency and simplicity when creating a program that will direct and control hardware.

**Objectives and Resources**

Installation and configuration of hardware components like the gesture sensor and battery.

Creating and managing tasks/reminders.

**Development Considerations**

*Programming language:* C++

- C++ has a fast execution speed, which is necessary for real-time applications that require communication between hardware components.
- Applications can easily manage communication protocols such as Bluetooth or Wi-Fi.

**Advantages:**

- Lightweight execution with minimal overhead, which is ideal for operations on resource-constrained systems.
- Direct hardware integration.

**Challenges:**

- GUI Complexity: Developing a user-friendly interface.
- Hardware Integration: Ensuring fast and precise communication between the application and the Raspberry Pi.

### 2.2.2 iPhone App: Developing using Swift

The iPhone app was developed to provide caregivers and teachers with a portable method to interact with the reminder device. The app prioritizes accessibility, allowing tasks and reminders to be managed conveniently and contingently.

**Purpose and Features**

- Users can manage reminders on their phones by adding, deleting, or editing the time of each reminder.
- The app connects seamlessly with the Raspberry Pi through Bluetooth.

**Key Features**

- User Interface:
    - Simple and easy-to-navigate design with clear labels and large buttons so that users can manage tasks with little effort
- Task Management:
    - Users can create new reminders and edit or delete existing ones.
- Real-Time Synchronization:
    - When a reminder/task is created, the app immediately uploads it to the Raspberry Pi. Data is automatically synced between the app and the device.

**Development Considerations**

- Real-Time Communication: Ensuring that data is updated instantly between the app and the device without delay or difficulty.
- Portability: Allowing users to manage their reminders and tasks wherever they are, providing flexibility and convenience.

**Challenges**

- Data Synchronization: Maintaining consistent communication between the app and the hardware required a lot of testing.
- Usability: Careful design to ensure functionality is prioritized while keeping the program simple.

These are the design choices that we believe will bring us to our goals outlined in Ch. 1.4. In the working product, we hope to have the hardware and software communicating with each other by connecting and sending information back and forth.

# Chapter 3
# Design and Rationale

Chapter 3 talks about our project design and the methodology used to make the overall device we created.

## 3.1 Existing Products vs. Our Product

As we were researching that most products today require a manual setup. For us to make the device simple for users, we wanted to do a remote setup where the device was programmed by the teacher and caregiver for the device to be used by the special needs user.

Existing applications for task reminders are often through a device and mainly delivered through text. This is helpful for most people, but for special needs, it may be challenging to read the text through a touchscreen or to not have access to the device. There was a study that talked about special needs listening to people they trust [14]. We found that special needs students listen to their teacher and caregiver from their voice and have a higher chance of doing a task if spoken to by a familiar voice. Therefore, we want to utilize this to help special needs in our product by allowing audio recorded by a teacher or caregiver and that being outputted for the user to do.

Most devices today have buttons and touchscreens, but that can be difficult for special needs to use those devices. We found that many special needs use gestures as a way to communicate with the people around them [15]. Knowing this information, we want our device to be interactive with gestures for the user as if they were communicating with their teacher or caregiver.

As we mentioned in Section 1.3, we envision the device to be portable and compact with similar size of a smartphone. Our goal was to make the device small enough to be easily carried to be brought anywhere or worn as an attachment to the belt for on the go use. These requirements were essential to making the device IoT wearable for users of our device.

All of the features we discussed above are accessible on most devices used today. During our research, we saw that many existing devices were too complex and distracting for special needs because they relied on screen-based interfaces and small buttons, which can be overwhelming and confusing for special needs. We specifically designed our product to be for special needs plans and simple.

| Feature Need | Existing Products | Our Product |
|---|---|---|
| **Task Input Method** | Manual Setup | Remote Setup |
| **Task Output for Users** | Text-based sent through notifications | Audio Reminders |
| **Control Interface** | Buttons and Touchscreens | Gesture Recognition |
| **Portability** | Smartphones and Tablets | Placed anywhere or clipped onto a belt |
| **Accessible for Special Needs** | Could be complex or distracting | Designed for Special Needs |

*Table 3.1.1: Existing Products vs. Our Device*

By understanding the existing products today and how our design can help support individuals with special needs, we began to test and integrate all the components mentioned in Chapter 2 to build the device we envision.

## 3.2 Hardware Design and Testing

### 3.2.1 Testing of Raspberry Pi Zero 2 W

This mini computer is the main component of our device and provides the functionality needed to achieve our project objectives. We first needed to install the Operating System of the Raspberry Pi Zero 2 W that was provided to us. After installing the operating system, we needed to understand how the Raspberry Pi Zero 2 W works by reading the documentation and testing the commands.

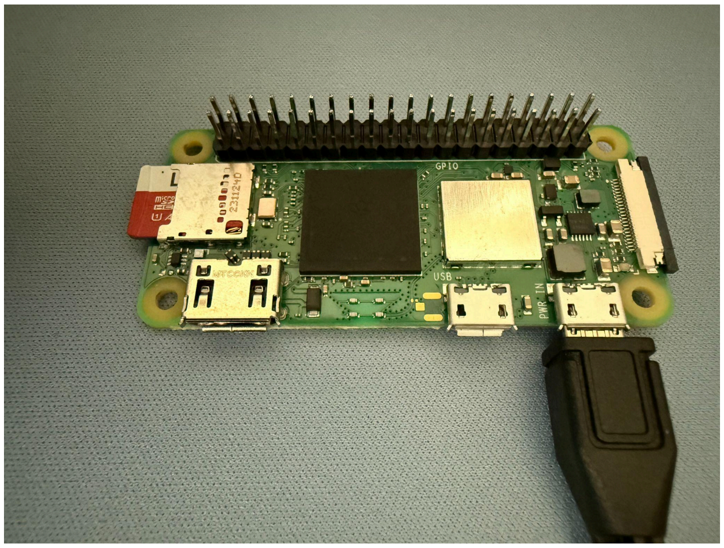


*Figure 3.2.1: Testing of the Raspberry Pi 2 Zero W*

As we were setting up the Raspberry Pi Zero 2 W, we noticed the Raspberry Pi heating up easily. We needed to design the device with enough airflow to compensate for the overheating or a cooling system to cool it down. We decided to go with more airflow because we are trying to get the device to be as small as possible, toward the end of the final build of the device.

### 3.2.2 Testing of Gesture Sensor

Our main objective is to make the device hand-gesture-controlled. To do so, we needed to check the effectiveness and accuracy of the gesture sensor we chose from Chapter 2. We first needed to check how to connect the gesture sensor to the Raspberry Pi and how the gesture sensor worked when connected, based on the datasheet.

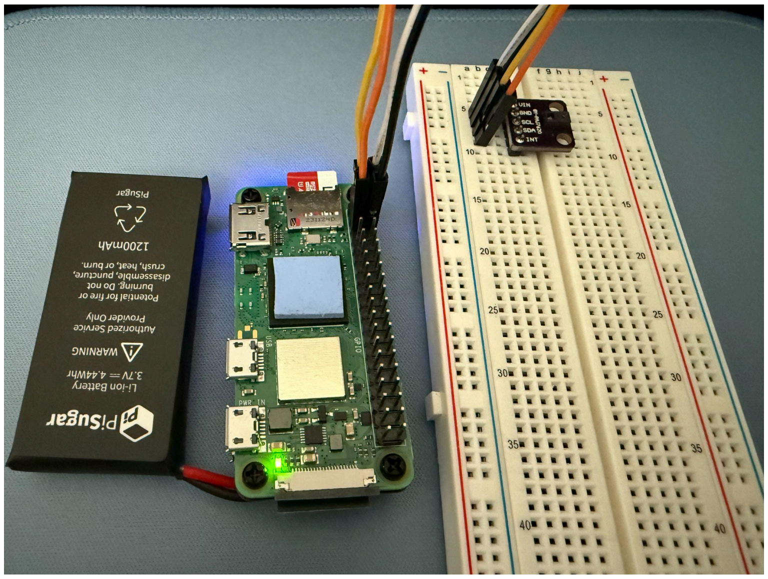


*Figure 3.2.2: Testing of the PAJ7620U2 Gesture Sensor*

As seen in Figure 3.2.2, we connected the gesture sensor as stated in the datasheet and started to test the hand gestures stated in the datasheet on what the sensor can detect. When we tested it for the first time, we saw that it was able to detect the hand gestures stated in the datasheet, but we noticed some inconsistencies. The sensor would detect the first gesture, but when we quickly switched hand gestures, it wouldn't register. As we saw the inconsistencies, we considered looking into the code provided by the datasheet and modifying it so we have enough delay to be able to read the right-hand gesture. Once we were able to correct these mistakes, we were able to use the gesture sensor to the best of our ability for the project.

### 3.2.3 Testing of Battery

We needed a battery to run the small computer without being plugged into the wall. To check if it works, we needed to follow the instructions on the datasheet for the connection to the Raspberry Pi Zero.

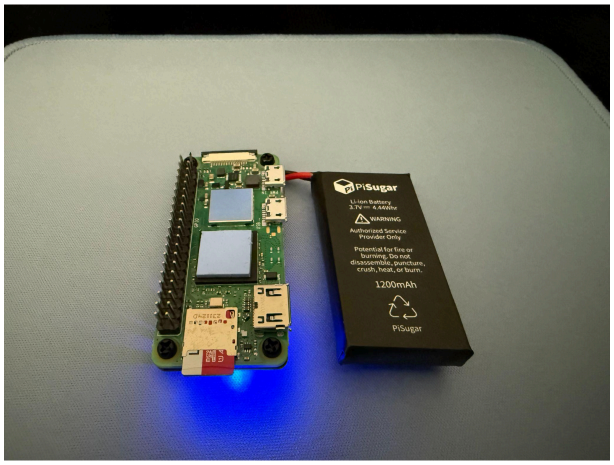

PiSugar
Li-ion Battery
3.7V ⎓ 4.44Whr
WARNING
Authorized Service Provider Only
Potential for fire or burning. Do not disassemble, puncture, crush, heat, or burn.
1200mAh
PiSugar

*Figure 3.2.3: Testing of the PiSugar 2 Battery*

One of the advantages of this battery is that it utilizes the underside pins of the Raspberry Pi Zero, which allows the top pins to be available for connecting additional components without interference, like our gesture sensor. When we powered the battery, we observed that all functionality of the Raspberry Pi Zero worked perfectly as if it was plugged to a power outlet. Additionally, the battery is rechargeable, eliminating the need of replacing batteries and allowing sustainability of the device.

## 3.2.4 Testing of Microphone HAT

We needed to test the microphone to see if it works and where it is being stored. After reading the datasheet, we needed to attach the HAT to the top of the Raspberry Pi Zero.

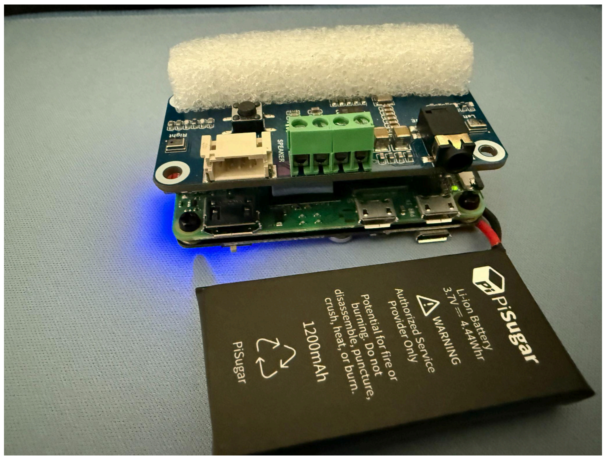

PiSugar
Li-ion Battery
3.7V ⎓ 4.44Whr
WARNING
Authorized Service Provider Only
Potential for fire or burning. Do not disassemble, puncture, crush, heat, or burn.
1200mAh
PiSugar

*Figure 3.2.4 Testing of the WM8960 HAT*

After attaching the HAT to the top of the Raspberry Pi Zero, we needed to be sure that the Raspberry Pi Zero knew it was connected to the HAT. As we were testing this out, we needed to be aware of how long it would record for and how much storage it would take up. After testing, we found that it worked and found that the HAT would record for as long as possible, but for this project, we wanted to have reminders 30 seconds or less, and usually, the max file size for that duration for an audio file is 1 megabyte (MB).

### 3.2.5 Testing of the Speaker

We needed to test the audio file to see if it would output for the speaker. First, we needed to test if the speakers we chose worked.

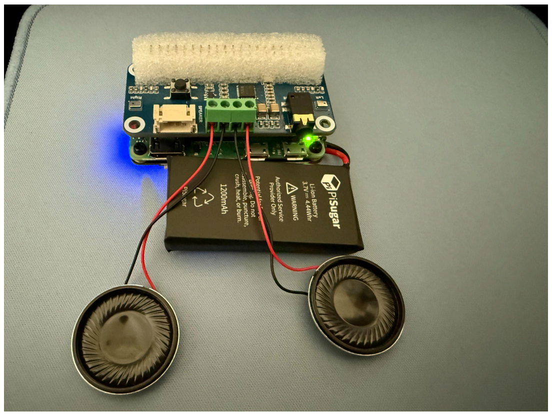


*Figure 3.2.5: Testing of the 8Ohm 2W Speaker*

We found that the audio was very low, and needed to correct the volume to be higher. From testing all of them, we wanted to incorporate everything together and put it into a 3D-printed case.

### 3.2.6 Design of 3D Printed Case

To save money, we wanted to design a 3D Printed Case to house our device. When we were designing the case, we had to keep in mind these two criteria:

- Choose a filament that is comfortable to wear but also protective against dropping the device.

- Design to have all open ports for ones that matter and have enough airflow so the device won't overheat.

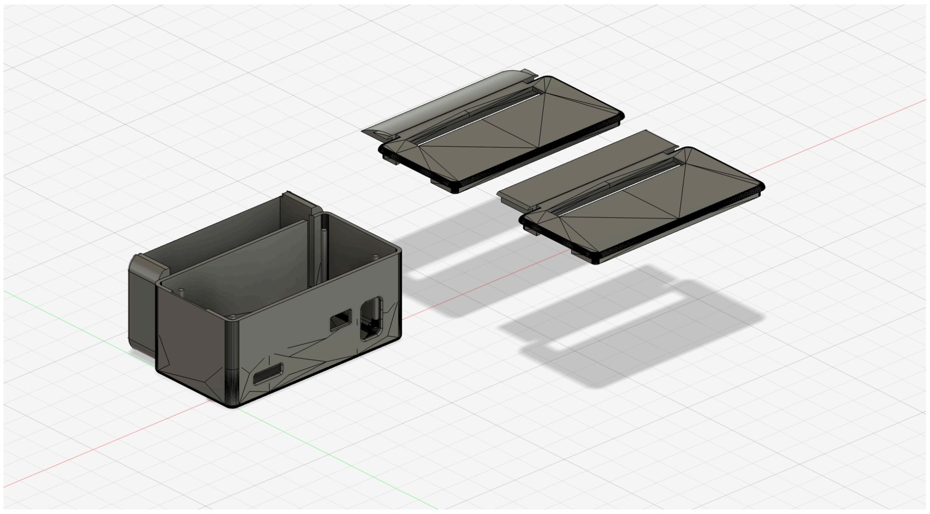


*Figure 3.2.6: Visual Concept of 3D Designed Case*

As shown in the figure above, we designed the case to have openings to plug in ports like the battery recharging and allow airflow so that the device doesn't overheat. Our research indicated that the TLA filament for 3D printing is strong and can protect the device from falls. Although other 3D printing materials such as PLA and PETG are more suitable for human contact, we decided to go with TLA because of the hard material and easier build with one filament.

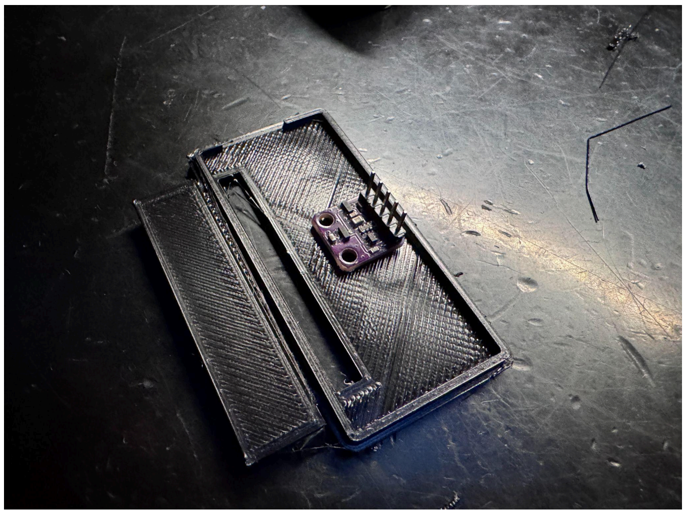


*Figure 3.2.7: 3D Printed Case (Lid)*

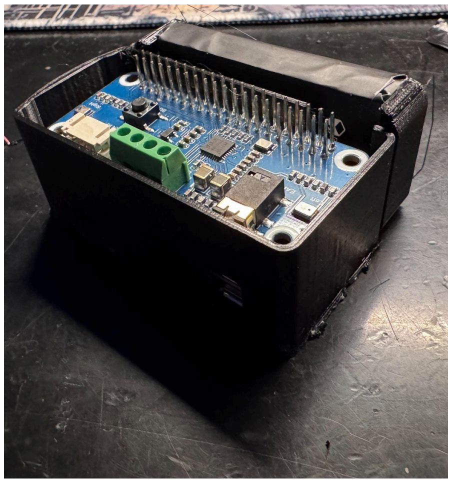


*Figure 3.2.8: 3D Printed Case (Base)*

Based on the two figures above, we can combine every component from this chapter into one overall device.

## 3.3 Software Design and Testing

When it comes to the software design of the project, we needed to create a state machine for the device to understand when a command is given.

### 3.3.1 Development of State Machine

The state machine of the device is used to help determine what the device should be doing when a command is given, and, in our case, when the user creates a gesture that the device can recognize.

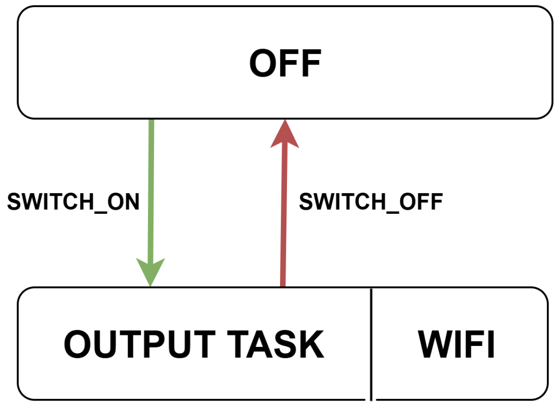


*Figure 3.3.1: Overview of the State Machine*

When developing the state machine, there was one main state that had two operations going on at the same time without interfering with each other, as shown in Figure 3.3.1. Those operations are:

- WIFI: Enable a wifi hotspot on the device to connect to the Wave Link Application to send audio tasks for the user to do.
- Output Task: When the reminder is triggered by the time, it will output the task to do, and will be confirmed by an upward hand gesture.

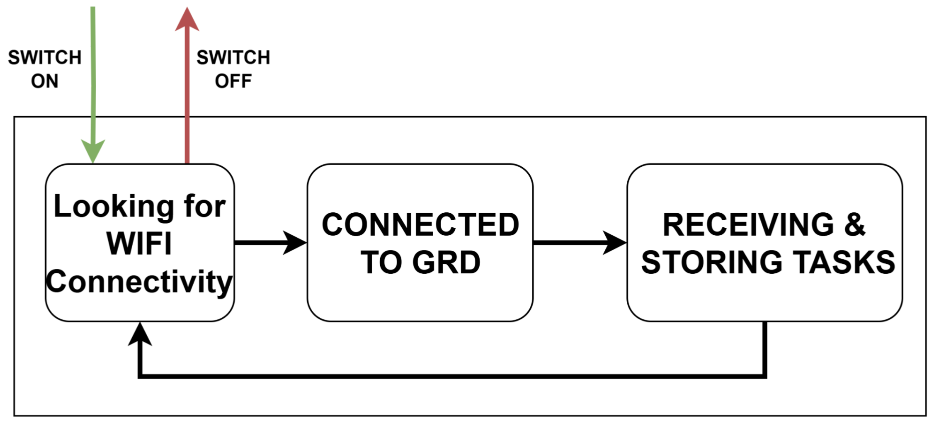


*Figure 3.3.2: WIFI State of State Machine*

When the GRD Companion is turned on, a wifi hotspot is enabled for the Wavelink app to connect to the device. In the setting of the teacher or caregiver's device, they will connect to the GRD companion. Once that connection is established, in the Wavelink app, record and edit tasks, and when ready, send the audio task files to the GRD, and it will store them in the device. The GRD will continue this process until the device is turned off.

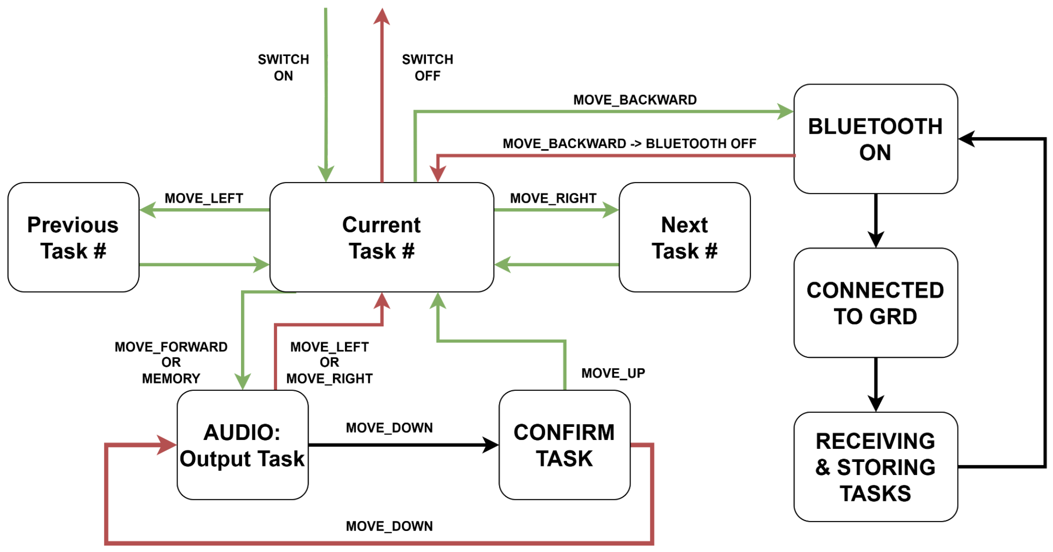


*Figure 3.3.3: Output State of State Machine*

The output state is the main feature of our device. Once the device is turned on, it enters the current task portion of the output state. If the task list is just given to the device from our Wavelink App, the default task number is 1 otherwise, it will go to the current task it left off on. If the user wants to traverse through the task list, they will make a gesture moving left to go back 1 task, or gesture moving right to go forward 1 task, and that will replace the current task. The current task will play the audio recorded by the teacher or caregiver if the time is set to output occurs or the gesture moves forward the sensor. If they don't want to do the task, they will make a gesture to change the current task, or if the user wants to confirm the task, they will make a downward motion. The device will be sure about it and confirm and erase the task when the confirmed task motion is done, and if not will go back to the audio task playing again.

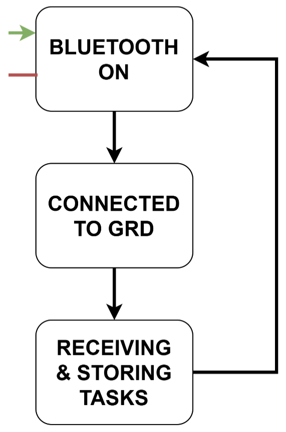


*Figure 3.3.3: Output State of State Machine*

We wanted to utilize the Bluetooth feature of being low power, so we needed to incorporate it. We decided to use Bluetooth Low Energy (BLE) and GATT. BLE devices broadcast short advertising packets at configurable intervals. The GRD periodically scans for these packets and only wakes its radio when it sees a device it's interested in, which in our case would be an iPhone.

GATT, which is the Generic Attribute Profile, is the structure of data exchange that BLE uses. It offers "services", which are containers, identified by a UUID, that have ***characteristics***, which also have their own UUID, that hold a single data value. The read and write operations allow a device to pull and push a characteristic's data.

For the device to turn on Bluetooth, the GRD must recognize a gesture moving away from the device. Acts the same as the WIFI operation we talked about earlier.

The state machine is a key map in understanding what the device should be doing at a given hand gesture. The struggle with this was knowing what hand gesture would lead to the next thing the device should do. This will be key in the software development of the applications we plan to design.

## 3.4 Integration of Hardware and Software

The success of the GRD Companion relies on tight coordination between its hardware components and the software modules. In this section, we describe how each software component maps onto the physical hardware, how communication protocols are configured, and how the state machine orchestrates overall device behavior.

### 3.4.1 GPIO and $I^2C$ Interfaces for Gesture Recognition

The PAJ7620U2 gesture sensor is connected to the Raspberry Pi Zero 2 W via the $I^2C$ bus. Within the firmware sensor.py, the $I^2C$ peripheral is initialized at boot to poll the sensor's address for gesture data. Whenever a hand movement happens, which can be either left, right, up, or down, the sensor.py decodes the register values into one of nine predefined gestures. These gestures are forwarded to the state machine module, state.py.

### 3.4.2 Audio Capture and Playback

Audio capture from teachers/caregivers is handled by the WM8960 audio HAT, which is attached to the Raspberry Pi's 40‑pin header. The software file audio_helpers.py uses the ALSA , Advanced Linux Sound Architecture, interface to configure the HAT in 16‑bit, 16 kHz mono mode, which matches our requirement of ≤1 MB per 30 s recording. When the user creates a new task through the WaveLink app, iPhone application sends the audio file over BLE to the Raspberry Pi. Once it has received the audio, task_manager.py saves the .wav file. To play the reminders at the scheduled time or on gesture command, audio_helpers.py opens the .wav file and routes PCM data through the audio HAT to the 8 Ohm, 2 Watt speaker.

### 3.4.3 Wi-Fi Hotspot and Bluetooth Low Energy (BLE) Pairing

At startup, a shell script, wifi_enable.sh, executes automatically to bring up the Pi's built-in Wi-Fi interface as an open hotspot. This allows the WaveLink desktop application, made with C++, to discover and connect over TCP/IP and upload task files. As for the iPhone application, ble_service_pairing.py initializes the BLE radio in peripheral mode. Using the Bleak library, it advertises a custom GATT service with two characteristics:

1. **Task Write Characteristic** for streaming new .wav reminders.
2. **Command Characteristic** for controlling playback, volume, and query operations.

The iPhone app scans for the GATT service and once it is found, it writes to the Task Write Characteristic to upload new tasks. Both the Wi-Fi and BLE systems run in the background so that gesture detection and state-machine transitions are never blocked by network I/O.

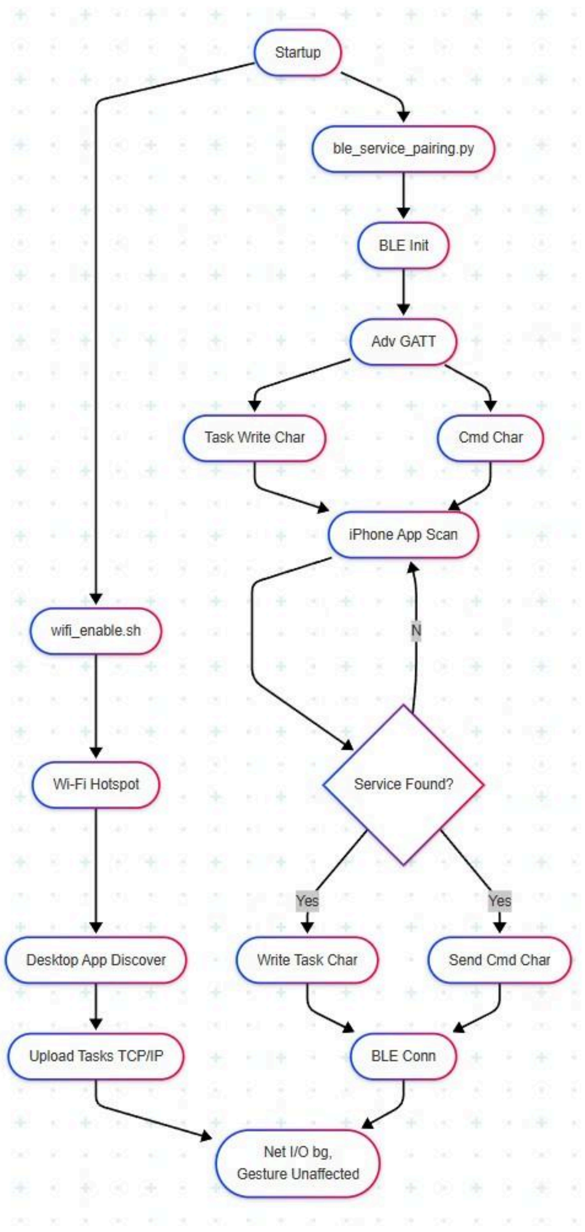


*Figure 3.4.1: Flowchart for Development of BLE/WIFI*

### 3.4.4 State Machine Coordination (state.py)

The core logic is in state.py, which implements a state machine mirroring Figures 3.3.1–3.3.3. At launch, the device enters the **OFF** state until a power-on gesture is completed. It then transitions to **WIFI**, where it invokes the hotspot and waits for tasks. Once the first task arrives, the state advances to **OUTPUT_TASK**. In this state, state.py queries task_manager.py for the current reminder's timestamp. If the system clock matches or a "play now" gesture is done, it invokes audio_helpers.py to play the .wav file. After playback, the module listens for a "confirm" gesture to delete the task from storage and move to the next reminder. Should a "next" or "previous" gesture, right or left swipe, occur at any time, the state machine updates an index pointer in task_manager.py without modifying the global schedule. Coming "back" from **OUTPUT_TASK** upon completion of all tasks or powering down returns the device to **OFF**, managing GPIO cleanup and safely dismounting any open file handles.

### 3.4.5 Cross‑Platform User Interfaces

While the Raspberry Pi firmware handles all hardware interactions, two separate frontend applications facilitate teacher/caregiver interaction:

1. **WaveLink Desktop (C++)**—Runs on Windows, listens for the "GRD_Companion" hotspot, and provides a GUI for recording, editing, and timestamping audio tasks. It connects via a TCP socket. Upon sending a new file, it waits for an ACK from the Raspberry Pi before marking the upload complete.
2. **WaveLink Mobile (Swift/iOS)**—Scans for the BLE advertisement. The GATT write operation streams audio in 20 KB chunks. On successful write, the mobile app updates its local view so caregivers can verify that the Pi has stored the correct file.

By mapping each software module to its corresponding hardware interface and embedding the state machine logic at the core, the GRD Companion achieves seamless integration. The $I^2C$ bus drives gesture input, the WM8960 HAT and speaker manage audio I/O, and the PiSugar 2 handles power management. At the same time, the combination of Wi‑Fi and BLE connectivity enables flexible task provisioning from either desktop or mobile devices. Collectively, these integration strategies ensure that our device remains responsive, energy‑efficient, and robust—meeting the needs of special needs adults as envisioned.

# Chapter 4
# Project Testing Protocols

## 4.1 Testing Protocols

As we finalized the device in Chapter 4, it was time to test the device. We had four points that we wanted to test:

- **Point 1:** The device should be able to receive audio reminder files straight from the Wavelink App via Bluetooth or WIFI.
- **Point 2:** The device should output the audio reminder at the given time it was set or commanded to output through the speakers.
- **Point 3:** The device should be mainly be controlled mainly through gesture recognition.
- **Point 4:** The device should be small and portable to be brought anywhere.

With each point, we had a certain number of questions that we needed to answer to be considered successful.

### 4.1.1 Audio File Transfer via Wireless Communication

We want to test the Bluetooth and WIFI connectivity between the teacher's or caregiver's computer and the device to see if data is sent correctly. Therefore, we need to answer the following questions:

1. Does the app work as designed?
2. Does the device connect via Bluetooth and WIFI?
3. Is the audio file stored correctly when transferred from the Wavelink App to the device?

### 4.1.2 Scheduled and Manually Playback of Tasks

We want to test if the audio reminder files can be output through the speakers at the appropriate time or when commanded by the user through a gesture. Therefore, we needed to answer the following questions:

1. Can we hear the reminder through the speakers?
2. Is the correct time that was set for the reminder to output correct?
3. Can we use a gesture to command the device to output the reminder?

### 4.1.3 Gesture Interaction between User and GRD

We wanted to test if the device was able to detect the appropriate hand gesture given the state machine for a certain task in Chapter 4. Therefore, we wanted to test our state machine to be sure that we can navigate through the device and do certain tasks only through gestures. We needed to answer the following questions:

1. Can the device detect hand gestures following the state machine in Chapter 4:
    - Navigating through the list of tasks?
    - Output Task?
    - Confirm a task was completed?
    - Turn on/off Bluetooth?

### 4.1.4 Portability

We wanted to be sure that the device was small and portable to be brought everywhere the user goes. We needed to answer the following questions:

1. Is the device small enough to fit in a small bag or backpack to be brought everywhere?
2. Can the device be used without being plugged into an external power source?
3. How long can the device be on till the battery dies?

As we test the device, we wanted to make sure that these questions are successfully answered before we go to human subject testing. In the next section, we will talk about the human subject testing overview.

## 4.2 Human Subject Testing

To see if our device works on real people applications, we wanted to get a Human Subject testing. At Santa Clara University, we needed to get approval for Human Subject Testing from an entity called the IRB. We needed to explain to the IRB that human subject testing is a huge factor in demonstrating that this project was successful. Therefore, we needed to fill out an IRB application to get approval for human subjects testing.

In the IRB application, we needed to be sure that we explained the following:

- Explain our project and why we must test with a human subject.
- The project is completely voluntary, and if the subject feels uncomfortable or doesn't want to participate, we won't hold anything against them.
- All data collected during testing will be confidential and protected, meaning the subject's name and face will not be revealed in any information we present.

By explaining these points, we submit the IRB application for review and wait for their response. In the meantime, we needed to find a tester of our device so that once we get approval, we can start right away.

We needed to find a special needs adult and their family to agree to help us show that our project can be successful. In connection with the Independence Network in Santa Clara, CA, we were able to meet with the head organizer, who was willing to help us set up a meeting for a potential user to help us with our project. In due time, the head organizer then appointed us a family whom they thought would benefit from our project. Once we communicated with the family about our projects and the benefits they could have, there were concerns at the initial meeting. They were able to reason and agree to help us with our project. During that time, we couldn’t test without IRB approval and therefore continued to work on the technical part of the project.

The IRB gave us approval on May 13th, 2025, to do testing of our device with individuals with special needs. As stated in section 4.1, we plan to follow the testing protocol with much emphasis on point 3, where we test the accuracy of our state machine of the device only using gesture recognition. It’s important to note that all data that will be collected during this testing will remain confidential between the members of the group and will share results given in the next chapter, based on the approval of the participant and family consent.

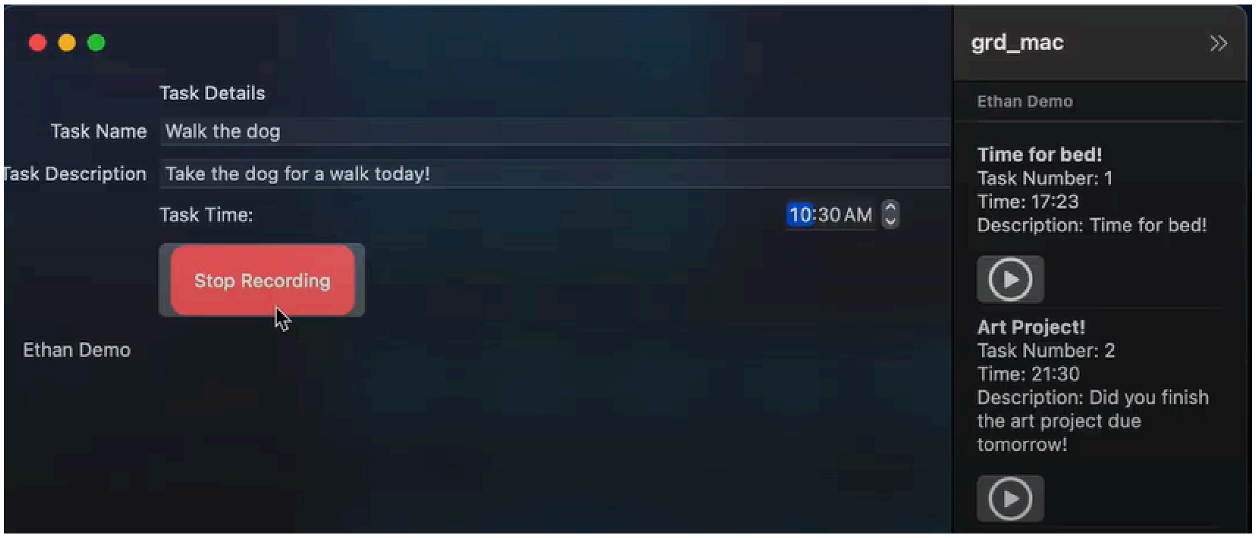
# Chapter 5
# Project Results

## 5.1 Member Testing Results

Based on the testing protocols mentioned in Chapter 5, we wanted to test the device ourselves before we brought it into human subject testing.

### 5.1.1 Testing of Wavelink App to GRD

As we discussed earlier, a test was carried out to determine if the recorded audio files could be recorded using our WaveLink App. The audio files that were recorded will be sent by connecting a computer that has the WaveLink App installed to the GRD Companion using Bluetooth or WIFI.



Task Name: Walk the dog
Task Description: Take the dog for a walk today!
Task Time: 10:30 AM

Ethan Demo

Ethan Demo

Time for bed!
Task Number: 1
Time: 17:23
Description: Time for bed!

Art Project!
Task Number: 2
Time: 21:30
Description: Did you finish the art project due tomorrow!

*Figure 5.1.1: Recording on the Wavelink App*

We first needed to test if we were able to use the Wavelink App to record audio tasks for the user to do. We see that it was successful, and we are also able to edit the tasks on the right hand side of the application.

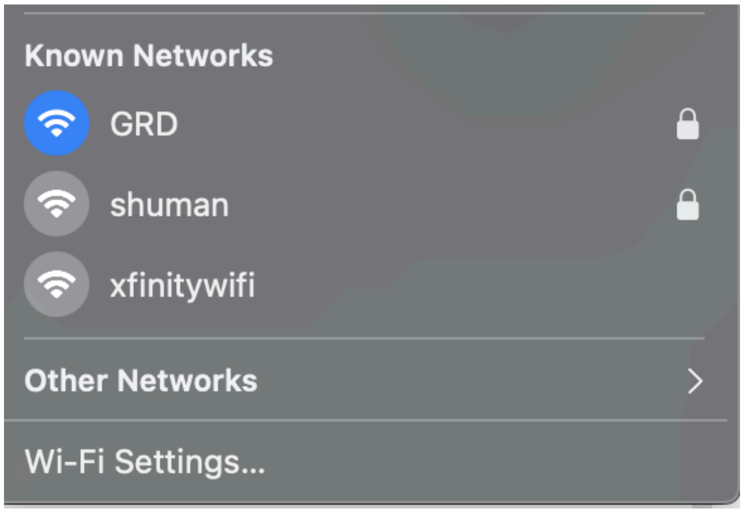


*Figure 5.1.2: Connect to the GRD via WIFI*

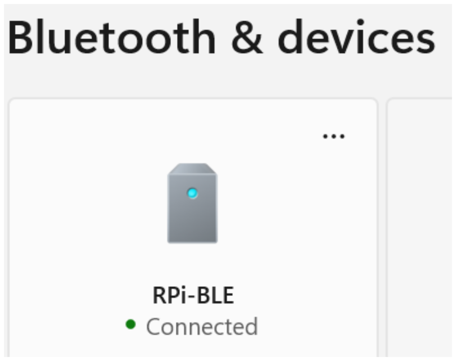
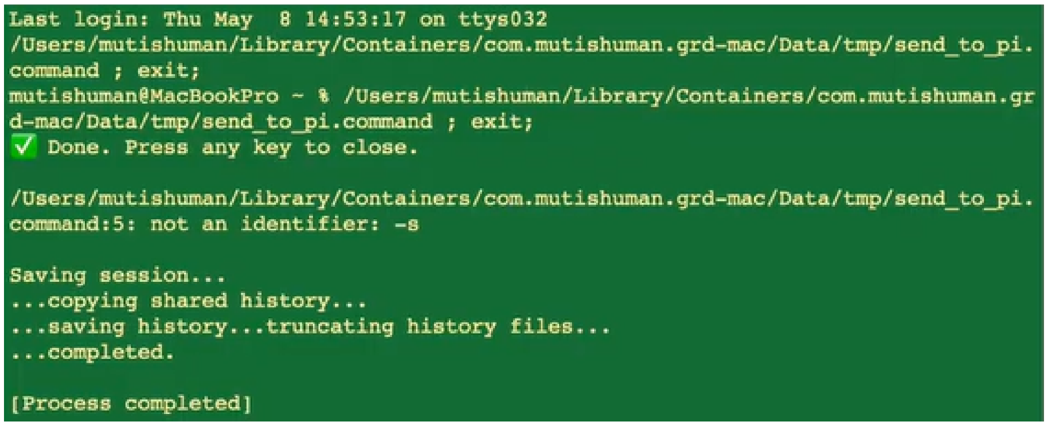


*Figure 5.1.3: Connect to the GRD via Bluetooth*

In the two figures above, we can connect to the GRD through enabling Bluetooth or connecting to the WIFI hotspot the GRD has. From this, we have a connection to send audio files from the Wavelink App to the GRD.



```
Last login: Thu May  8 14:53:17 on ttys032
/Users/mutishuman/Library/Containers/com.mutishuman.grd-mac/Data/tmp/send_to_pi.
command ; exit;
mutishuman@MacBookPro ~ % /Users/mutishuman/Library/Containers/com.mutishuman.gr
d-mac/Data/tmp/send_to_pi.command ; exit;
✅ Done. Press any key to close.

/Users/mutishuman/Library/Containers/com.mutishuman.grd-mac/Data/tmp/send_to_pi.
command:5: not an identifier: -s

Saving session...
...copying shared history...
...saving history...truncating history files...
...completed.

[Process completed]
```

*Figure 5.1.4: Confirmation Files Sent to GRD*

With this confirmation shown in the figure above, we can safely confirm that point 1 was a success, and we can send audio files from the Wavelink App to the GRD.

### 5.1.2 Testing of Task Output on GRD

In point 2, testing was required to verify that the GRD Companion would output a task that was saved from point 1. It was also important to see if we are allowed to move through the task list and output each task on the list. A key factor to test to see if the user doesn't want to command the task to output manually, then the task should automatically output at the time set from the Wavelink app.

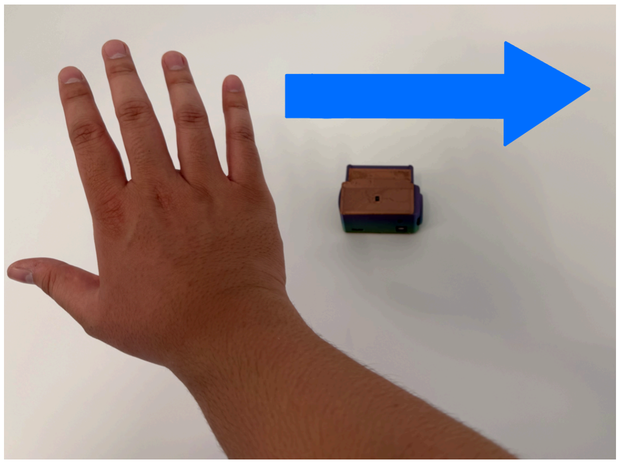


*Figure 5.1.5: Gesture to Navigate through Task List (Move Right)*

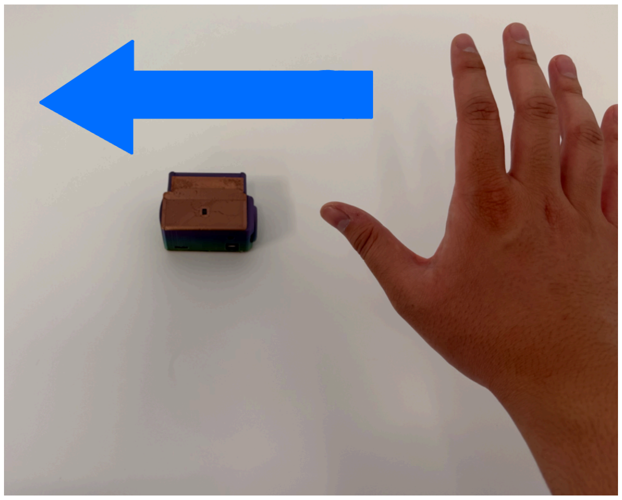


*Figure 5.1.6: Gesture to Navigate through Task List (Move Left)*

As shown in the figure above, users can navigate through the task list saved from our WaveLink App by performing right motion gestures to move up the list and left motion gestures to move down the list. We have designed this for the task list to be looped if the device reaches the beginning or end of the list and starts over again.

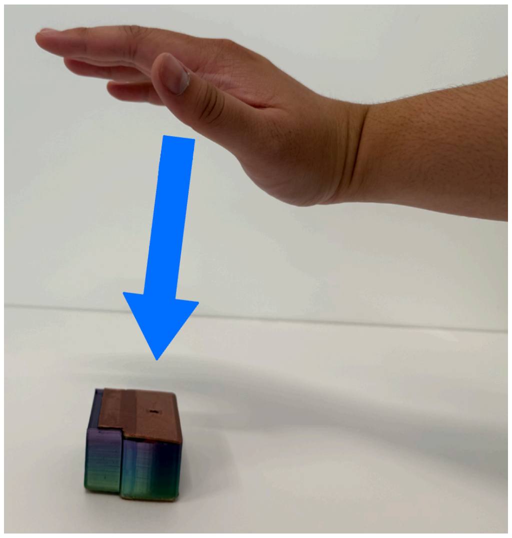


*Figure 5.1.7: Gesture to Manually Output Selected Task*

The figure above shows how to test if the forward motion gesture to output the task manually works. We were able to try this several times with different tasks in the task list and found that it output each task successfully.

As a part of point 2, another was necessary to see if the device is able to output a task given the time set to output. We put a task in the Wavelink app and set a time and waited for the task to output. It was safe to say that it worked both ways to output a task.

## 5.1.3 Testing of Gesture Recognition for GRD

In point 3, we wanted to test the gesture recognition that the GRD can do. This allows us to test the state machine that we came up with in the previous chapter. We were able to showcase how to navigate through different tasks and play them from the previous section. Therefore, we needed to only test the Bluetooth connectivity on how to turn it on and off.

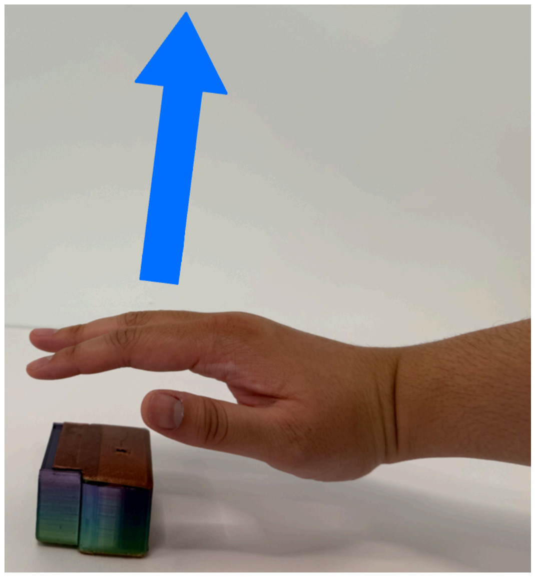


*Figure 5.1.8: Gesture to Turn ON and OFF Bluetooth*

As seen in the image above, we can use a gesture to turn on and off the Bluetooth of the GRD, and from section 6.1.1, we can successfully pair the Wavelink App to the GRD through Bluetooth. Because of the previous section and testing of this, we can tell this is true.

### 5.1.4 Testing of Size and Portability for GRD

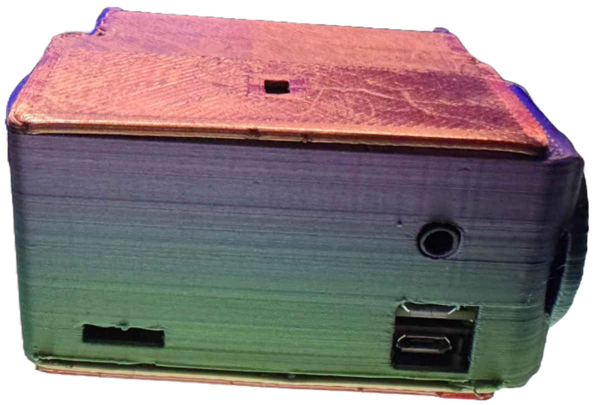


*Figure 5.1.9: The GRD Companion*

In the figure above, we can showcase the physical device, GRD Companion. The figure shows the size of the device, and we can put this device anywhere we go. This also shows the portability of the GRD, where it can be placed everywhere we go.

Now let's reflect on our points that we wanted to test, and if we were successful in each point:

- ☑ **Point 1:** The device should be able to receive audio reminder files straight from the Wavelink App via Bluetooth or WIFI.
- ☑ **Point 2:** The device should output the audio reminder at the given time it was set or commanded to output through the speakers.
- ☑ **Point 3:** The device should be mainly be controlled mainly through gesture recognition.
- ☑ **Point 4:** The device should be small and portable to be brought anywhere.

We can see that our testing was successful, with some issues that arose when testing. Luckily, we were able to identify and fix these issues, allowing us to proceed with human subject testing.

## 5.2 Human Testing Results and User Feedback

To test the usability and effectiveness of our device, we conducted initial human subject testing with an individual in special needs education. The participant was shown the device and asked to interact with it through simple gesture-based input. This allowed the device to navigate through the task list and output audio tasks set by the parent or teacher through our WaveLink App. Their feedback highlighted both strengths, areas of improvement, and suggestions for future iterations of the device.

The participant appreciated the device's simplicity and the use of a voice-based task given especially from a familiar voice like their teacher or caregiver. This shows that voice-based reminders can be more effective than text-based reminders for special needs who may struggle with text-based reminders. However, concerns were raised about the device size as it's big based on Figure 5.1.8. A suggestion given from this concern was to make the device smaller or adaptable to be worn on different body parts, like a necklace or a watch.

There was a strong interest in making the device more interactive and rewarding. This suggests that an implementation of a reward system is placed when a task or a certain number of tasks is completed, and the user is told a reward is granted for the user's hard work. The parent of the participant expressed that we should add a repetitive user motion detection to identify behavioral insights and a potential calm-down message to help stop the repetitive behavior detected.

As the participant was using the device, the parent mentioned that other users may need time to adjust to the device, where their engagement may depend on motivation or personal interest. To address this, it was recommended that the device offer consistent positive feedback and remain customizable to fit users' interests. Additionally, having a syncing feature where completed tasks with the WaveLink app are given automatically helps keep track of user progress and enables remote updates.

Overall, the feedback provided insight that the device worked as intended and revealed several promising directions for future iterations of the project.

Chapter 6

# Constraints and Standards

We will discuss the constraints we were aware of and the standards placed for the project.

## 6.1 Constraints

The development of the GRD Companion was shaped by several constraints that influenced both the hardware and software of our design. These constraints reflect limitations such as cost, time, available resources, and special needs users' needs. The team carefully considered economic, technical, environmental, and user-related factors in our design process, recognizing the importance of delivering a solution for special needs.

### 6.1.1 Economic Constraints

One of the primary limitations for our project was the budget. We wanted to use affordable components for our project to ensure the device could be taken by any school or family. Therefore, we wanted to be sure that the device we plan to build was under $60 to ensure that all were able to afford this.

### 6.1.2 Technical Constraints

The technical capabilities of the system were limited by the computer we chose to use. This will influence peripherals and require the use of efficient communication protocols like I2C and gesture recognition algorithms. Also, based on our research, we couldn’t find any software that suited our needs, therefore, we needed to create our application from scratch.

### 6.1.3 Environmental Constraints

This device was designed to be brought everywhere, which imposed constraints on size and durability. Therefore, we needed to be sure that we chose components that were small enough when put together to be brought everywhere, as well as have a 3D printed case that was nontoxic and protected the device from environmental causes.

### 6.1.4 User Constraints

Since the device is intended for individuals with special needs, it had to be simple, intuitive, and require minimal interaction. This drives the design toward having gesture-based input and audio feedback instead of button or screen interaction.

## 6.2 Standards

In addition to the constraints of the project, we want to consider engineering standards. These standards help us provide a framework that guides the ethical and technical design decisions. The device's hardware and software were selected and implemented following specifications to support performance and future adaptations of the project.

### 6.2.1 Communication Standards

Since we will be working with Inter-Integrated Circuit (I2C), we need to follow the protocol of I2C to ensure that no two components share the same address that will be on the same bus [16]. We also need to ensure that we follow the logic level compatibility for safe hardware interfacing. Therefore, we needed to follow the reusable I2C protocol structure proposed by Hu [17], which separates the I2C system into protocol, signal, and interface levels. This layered approach allows clear communication, allowing easy adaptation when integrating I2C-based components.

### 6.2.2 Electrical and Safety Standards

Since our device will be powered by a rechargeable battery, we need to be sure that it is a low-voltage design that will minimize the risk of electrical hazards to the users of our device. To meet this objective, we needed to follow the basic circuit protection, ensuring that there are no hazards when the device is turned on, used, and turned off. In alignment with safety practices outlined by Kumar and Raghuveer [18], we also considered low-voltage distribution standards to reduce the likelihood of overheating, electrical shock, or failure due to environmental constraints mentioned before.

### 6.2.3 Accessibility and Usability Standards

To ensure our device is accessible and user-friendly for individuals with special needs, we need to consider the principles from ADA standards for accessible design [19]. Therefore, we needed to consider the physical and cognitive accessibility, so we decided to avoid buttons and touchscreens and just use gesture-based interaction. Additionally, we designed the interface between the user and our device to be simple and consistent, reducing the cognitive load, which aligns with ADA guidelines for accessible communication systems.

### 6.2.4 Ethical Standards

Our project is guided by ethical standards where we prioritize safety, dignity, privacy, and inclusivity for individuals with special needs. During our human subject testing, we asked for informed consent from the participant and their family and ensured that all data collected would not have personal identifiable information and would be deleted when the project concludes. In our device design, we choose to pick components that allow simple gesture-based input while maintaining privacy concerns. Accessibility and safety were essential in our design to ensure that the device posed no physical or emotional harm to the user. Throughout the development of our device, we were committed to creating a respectful and secure tool that supports the user and the people around them.

# Chapter 7
# Societal Issues

An important aspect of this project is to reflect on the ethics of being an engineer and how our work impacts society, human rights, and other elements. In this section, we look into ethical justification, how we used ethics in our project to show the characteristics of a good engineer, considerations regarding safety and risks, and how we decided to go through with the project.

## 7.1 Ethical and Social

We specifically designed our device for special needs users, and we kept in mind privacy concerns for our project. When choosing the components for our device, we factored in privacy. This came specifically on how we can do gesture recognition. We had the choice of using a camera or a sensor. While a camera would allow for more accurate customization, it would cause privacy issues because the camera always records and keeps that data. With a sensor, we can respect the privacy of the user and allow more simplicity to the project. Therefore, we want to ensure that our ethical and social view of the project was designed to help the user.

## 7.2 Political and Economic

Our project addresses economic disparities that prevent everyone can affording expensive, high-tech products. Therefore, our solution to this was to allow an affordable product that can be used in schools and low-resource communities. By using low-cost hardware and open source tools, we can ensure that the device remains financially reasonable without sacrificing the functionality our device should have. Politically, we can say that we live up to the educational and accessibility standards that promote equal opportunities for individuals with special needs.

## 7.3 Health, Safety, and Manufacturability

We wanted to be sure our device is safe to use and easily replicated for all people in the special needs community. In designing our device, we wanted to be sure that low voltage was used so no electrical or thermal hazards would arise. In the case design, we wanted to use 3D printing material that was protective of the device as well as a safe, non-toxic material. We wanted to be sure that all the components of our device were off the shelf so that it was easily accessible and could be replicated easily.

## 7.4 Sustainability and Environmental Impact

As mentioned before, we wanted to be sure low power consumption was taken for safety as well as sustainability. To avoid e-waste, we decided to incorporate a rechargeable battery rather than a disposable battery. On top of that, we wanted to make sure that if we were to replace any component, it would be modular and easy to recycle.

## 7.5 Usability

The device was designed to help special needs users with limited cognitive abilities and utilizing gestures that they tend to do. It can be difficult for people to use devices with small buttons or read fine print on screens, which makes many user interfaces too complicated to use. Therefore, we wanted our device to allow simple audio output of someone the user trusts, as well as make sure the gesture interface improves accessibility and ease of use of the device.

## 7.6 Lifelong Learning

The project has taught us technical skills that can be used in the real industry. We learned how to program embedded systems and make sure that the design of our device was a human-centered design for special needs. We also wanted to promote lifelong learning for users by helping them stick to routines set by their teacher and caregiver, allowing the user to gain independence over time.

## 7.7 Compassion

The reminder device is designed to serve a community of individuals with special needs and those around them. We empathize with individuals with special needs and their families and want to support them through this device. Our goal is to create not just a tool, but a meaningful device that assists individuals with special needs and encourages independence, and strengthens the connection between individuals with special needs and their family and caregivers.

# Chapter 8
# Conclusion

## 8.1 Final Design Assessment

We began our senior design project with a vision to help special needs by designing a device that they can bring everywhere to help with their everyday life. Researching and understanding the special needs community and what they envision and what they need to achieve helped us understand what we needed to create for them. We were able to come up with the GRD Companion, a gesture-controlled device designed to help special needs with everyday life by giving audio reminders set by the user's teachers and caregiver. With a device goal in mind, we were able to choose components to incorporate and build a successful device. The final prototype showcased a successful demonstration of using gestures to control the device and enable task reminders, to support users becoming more independent. Overall, the GRD Companion met the expectations of our design and testing phase, and was received with positivity throughout our testing and human subject testing.

## 8.2 Future Work

While our prototype is successful with the goals we set, there are several areas that we can improve and expand on our project. Future iterations could include additional features such as real-time task completion tracking, a reward system for completing certain tasks or the entire task list, customizable hand gesture recognition, and voice recognition to detect repetitive behaviors using artificial intelligence (AI). Enhancing customization will improve user adaptability and allow more usability options for our users in using our device. We would also like to expand on our WaveLink App for mobile and desktop. We envision the WaveLink App to be accessed on any device. In doing so, we would like to publish the application across every platform and continue adding features to the application, like the real-time task completion tracking and reward system. Furthermore, more testing with a diverse user base would provide a deeper insight into improvements for the device and the mobile and desktop applications. Our vision for the GRD Companion is that it will continue to evolve and potentially become a device that every individual with special needs can buy at a low cost to help them throughout the day, with the intention of them incorporating the device in their daily lives that support them in becoming independent.

# Chapter 9

# References

[1] K. L. Robson, P. Anisef, R. S. Brown, and G. Parekh, “The intersectionality of postsecondary pathways: The case of high school students with special education needs,” Can. Rev. Sociol., vol. 51, no. 3, pp. 193–215, Jul. 2014, doi: 10.1111/cars.12044.

[2] Yam, “Addressing the significant shortage of special education teachers in California,” The Arc of California, Aug. 14, 2023. [Online]. Available: https://thearcca.org/addressing-the-significant-shortage-of-special-education-teachers-in-california/

[3] D. R. Bryant-Rich and D. E. Barshaw-Rich, “Personal voice operated reminder system,” U.S. Patent Application US20120265535A1, filed Sep. 6, 2010, published Oct. 18, 2012.

[4] A. J. Venner et al., “A reminder and actioning system,” European Patent Application EP1115076A1, filed Jan. 6, 2000, published Jul. 11, 2001.

[5] R. R. Dunton, “Location based task reminder,” U.S. Patent Application US20060061488A1, Mar. 23, 2006.

[6] E. Yoffie, “Device for reminding users,” U.S. Patent US7746730B2, issued Jun. 29, 2010.

[7] R. Grover and J. Tan, “Reminder device wearable by a user,” U.S. Patent Application US20160299572A1, filed Apr. 11, 2016, published Oct. 13, 2016. [Online]. Available: https://patents.google.com/patent/US20160299572A1

[8] V. Y. Sigounas, “Technologies of care and the engineering imaginary: Two approaches to assistive device design for the Global South,” Med. Anthropol. Q., Oct. 2023, doi: 10.1111/maq.12818.

[9] Raspberry Pi Ltd., Raspberry Pi Zero 2 W Product Brief, Apr. 2024. [Online]. Available: https://www.raspberrypi.com/documentation/

[10] Waveshare, “PAJ7620U2 gesture sensor,” Waveshare Wiki. [Online]. Available: https://www.waveshare.com/wiki/PAJ7620U2_Gesture_Sensor. [Accessed: Dec. 8, 2024].

[11] PiSugar Kitchen, “PiSugar 2 1200 mAh Raspberry Pi Zero battery,” PiSugar. [Online]. Available: https://www.pisugar.com/products/pisugar2-raspberry-pi-zero-battery. [Accessed: Apr. 7, 2025].

[12] Amazon, “Mini portable speaker, compatible with any device with 3.5mm jack,” Amazon. [Online]. Available: https://www.amazon.com/dp/B0C49TKSQQ. [Accessed: Apr. 7, 2025].

[13] Waveshare, “WM8960 Audio HAT,” Waveshare. [Online]. Available: https://www.waveshare.com/wiki/WM8960_Audio_HAT. [Accessed: Dec. 8, 2024].

[14] E. L. Hardman, “Three children with emotional and behavioral disorders tell why people do right,” Int. J. Spec. Educ., vol. 27, no. 1, pp. 160–176, 2012. [Online]. Available: https://files.eric.ed.gov/fulltext/EJ979722.pdf

[15] A. Attwood, U. Frith, and B. Hermelin, “The understanding and use of interpersonal gestures by autistic and Down’s syndrome children,” J. Autism Dev. Disord., vol. 18, no. 2, pp. 241–257, Jun. 1988, doi: 10.1007/bf02211950.

[16] J. Wu, “A basic guide to I2C,” Texas Instruments. [Online]. Available: https://www.ti.com/lit/pdf/sbaa565

[17] Z.-w. Hu, “I2C protocol design for reusability,” in Proc. 3rd Int. Symp. Inf. Process., Qingdao, China, Oct. 2010, pp. 83–86, doi: 10.1109/ISIP.2010.51.

[18] C. Kumar and S. Raghuveer, “Design guidelines for safer low voltage distribution systems,” Int. J. Sci. Eng. Res., vol. 4, no. 6, pp. 2027–2030, Jun. 2013.

[19] U.S. Department of Justice, 2010 ADA Standards for Accessible Design, Sep. 15, 2010. [Online]. Available: https://www.ada.gov/2010ADAstandards_index.htm

# Chapter 10

# Appendices

## 10.1 UML Diagram

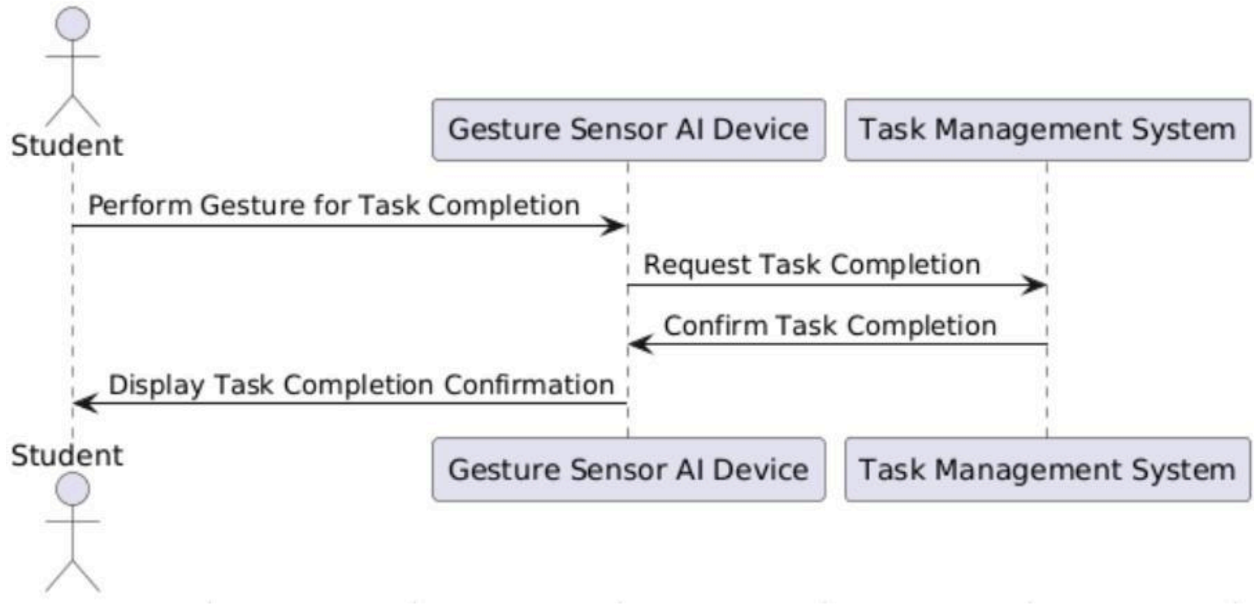
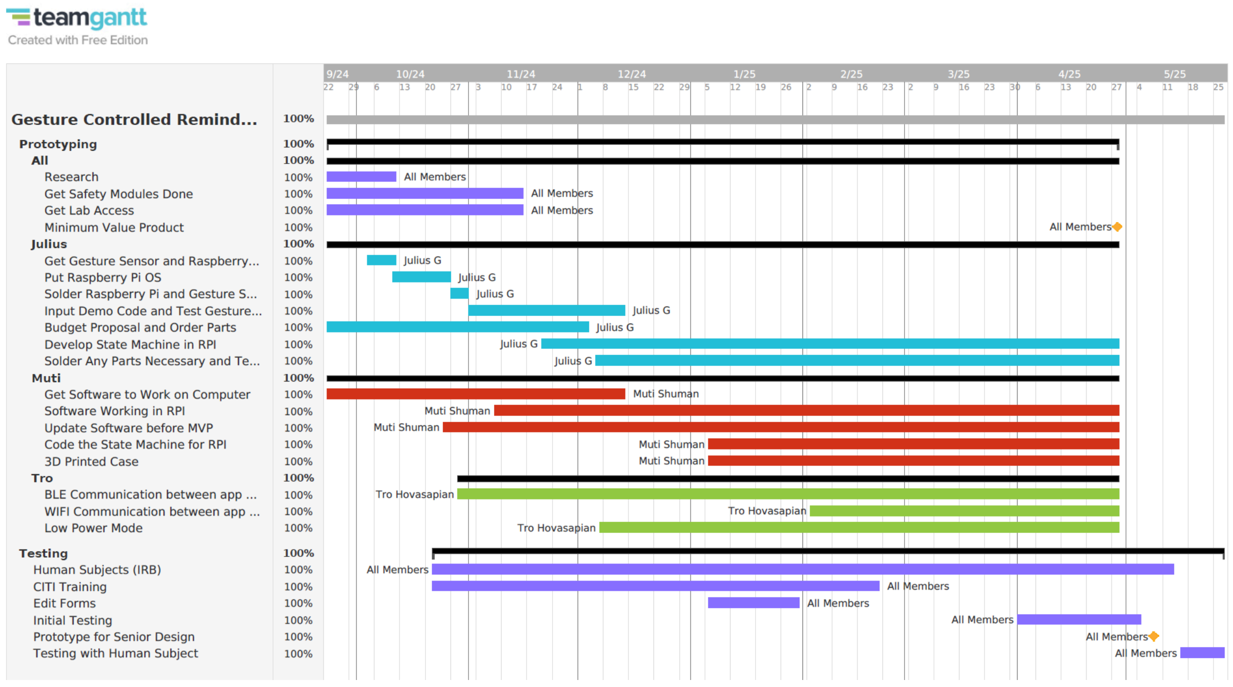


*Figure 10.1.1: UML Diagram*

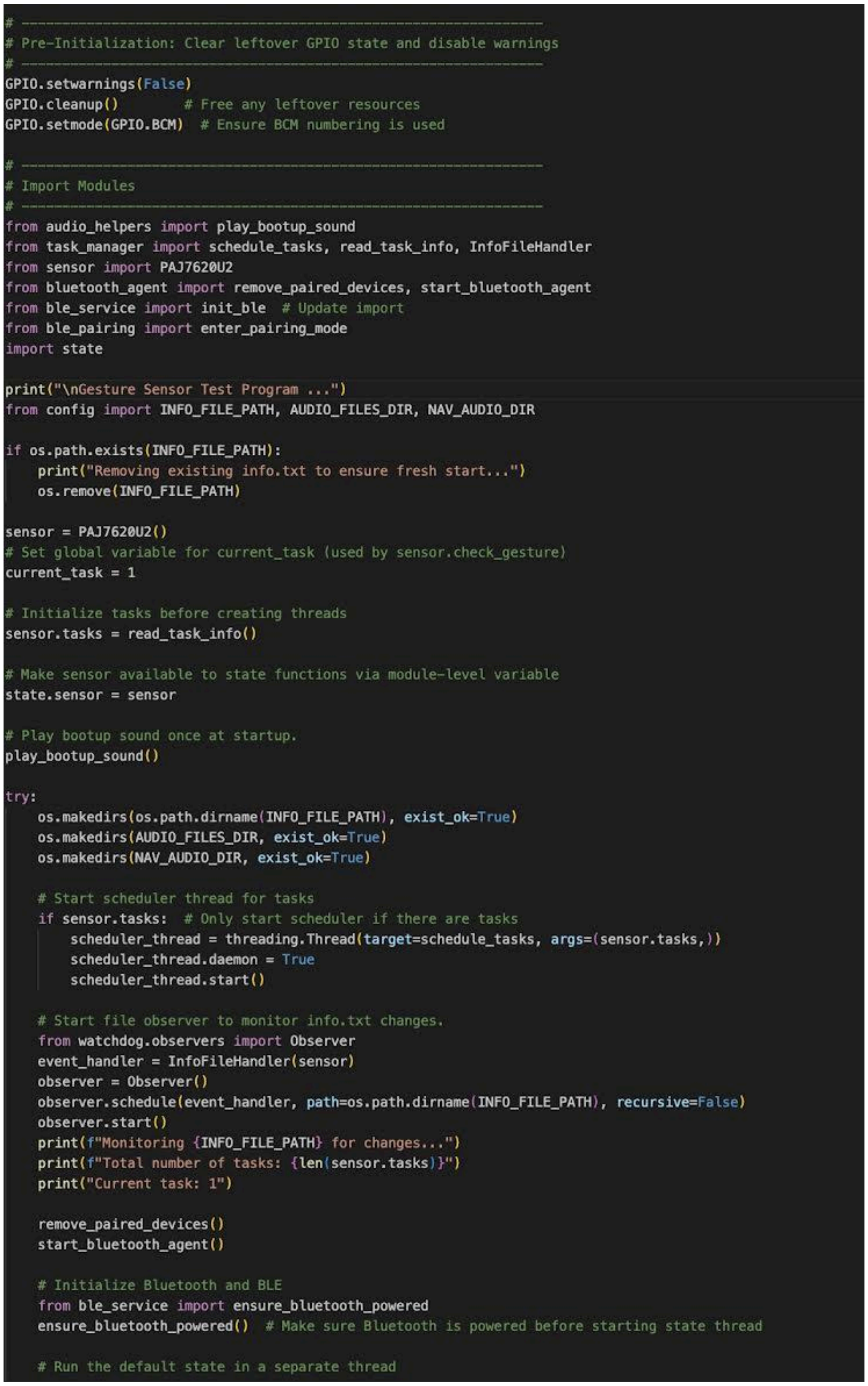
## 10.2 Timeline



| Task | Progress | Assigned |
|---|---|---|
| **Gesture Controlled Remind...** | 100% | |
| **Prototyping** | 100% | |
| **All** | 100% | |
| Research | 100% | All Members |
| Get Safety Modules Done | 100% | All Members |
| Get Lab Access | 100% | All Members |
| Minimum Value Product | 100% | All Members |
| **Julius** | 100% | |
| Get Gesture Sensor and Raspberry... | 100% | Julius G |
| Put Raspberry Pi OS | 100% | Julius G |
| Solder Raspberry Pi and Gesture S... | 100% | Julius G |
| Input Demo Code and Test Gesture... | 100% | Julius G |
| Budget Proposal and Order Parts | 100% | Julius G |
| Develop State Machine in RPI | 100% | Julius G |
| Solder Any Parts Necessary and Te... | 100% | Julius G |
| **Muti** | 100% | |
| Get Software to Work on Computer | 100% | Muti Shuman |
| Software Working in RPI | 100% | Muti Shuman |
| Update Software before MVP | 100% | Muti Shuman |
| Code the State Machine for RPI | 100% | Muti Shuman |
| 3D Printed Case | 100% | Muti Shuman |
| **Tro** | 100% | |
| BLE Communication between app ... | 100% | Tro Hovasapian |
| WIFI Communication between app ... | 100% | Tro Hovasapian |
| Low Power Mode | 100% | Tro Hovasapian |
| **Testing** | 100% | |
| Human Subjects (IRB) | 100% | All Members |
| CITI Training | 100% | All Members |
| Edit Forms | 100% | All Members |
| Initial Testing | 100% | All Members |
| Prototype for Senior Design | 100% | All Members |
| Testing with Human Subject | 100% | All Members |

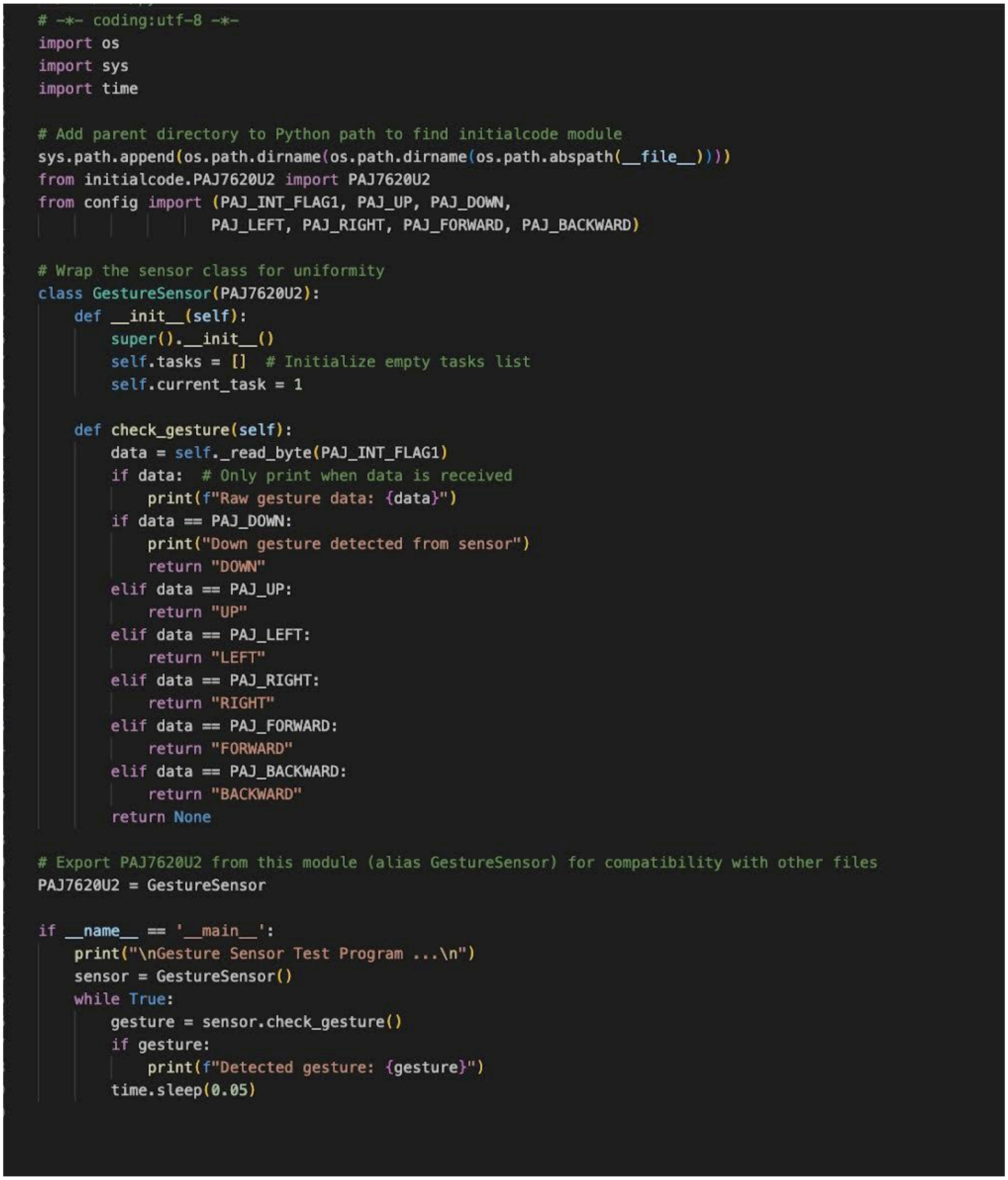
*Figure 10.2.1: Gantt Chart of Project*

## 10.3 Code for Gesture Sensor



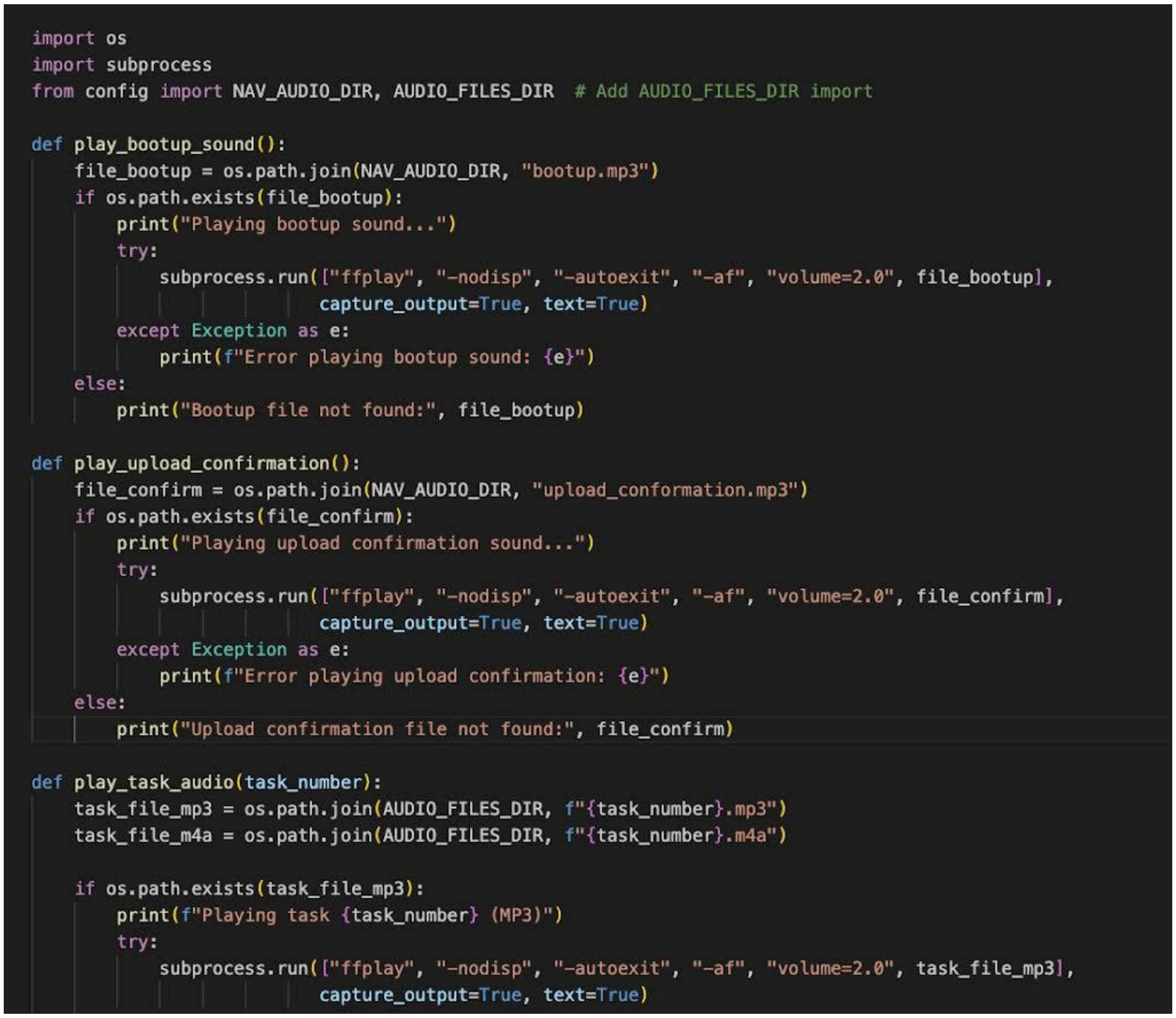
```python
# ---------------------------------------------------------------------
# Pre-Initialization: Clear leftover GPIO state and disable warnings
# ---------------------------------------------------------------------
GPIO.setwarnings(False)
GPIO.cleanup()        # Free any leftover resources
GPIO.setmode(GPIO.BCM)  # Ensure BCM numbering is used

# ---------------------------------------------------------------------
# Import Modules
# ---------------------------------------------------------------------
from audio_helpers import play_bootup_sound
from task_manager import schedule_tasks, read_task_info, InfoFileHandler
from sensor import PAJ7620U2
from bluetooth_agent import remove_paired_devices, start_bluetooth_agent
from ble_service import init_ble  # Update import
from ble_pairing import enter_pairing_mode
import state

print("\nGesture Sensor Test Program ...")
from config import INFO_FILE_PATH, AUDIO_FILES_DIR, NAV_AUDIO_DIR

if os.path.exists(INFO_FILE_PATH):
    print("Removing existing info.txt to ensure fresh start...")
    os.remove(INFO_FILE_PATH)

sensor = PAJ7620U2()
# Set global variable for current_task (used by sensor.check_gesture)
current_task = 1

# Initialize tasks before creating threads
sensor.tasks = read_task_info()

# Make sensor available to state functions via module-level variable
state.sensor = sensor

# Play bootup sound once at startup.
play_bootup_sound()

try:
    os.makedirs(os.path.dirname(INFO_FILE_PATH), exist_ok=True)
    os.makedirs(AUDIO_FILES_DIR, exist_ok=True)
    os.makedirs(NAV_AUDIO_DIR, exist_ok=True)

    # Start scheduler thread for tasks
    if sensor.tasks:  # Only start scheduler if there are tasks
        scheduler_thread = threading.Thread(target=schedule_tasks, args=(sensor.tasks,))
        scheduler_thread.daemon = True
        scheduler_thread.start()

    # Start file observer to monitor info.txt changes.
    from watchdog.observers import Observer
    event_handler = InfoFileHandler(sensor)
    observer = Observer()
    observer.schedule(event_handler, path=os.path.dirname(INFO_FILE_PATH), recursive=False)
    observer.start()
    print(f"Monitoring {INFO_FILE_PATH} for changes...")
    print(f"Total number of tasks: {len(sensor.tasks)}")
    print("Current task: 1")

    remove_paired_devices()
    start_bluetooth_agent()

    # Initialize Bluetooth and BLE
    from ble_service import ensure_bluetooth_powered
    ensure_bluetooth_powered()  # Make sure Bluetooth is powered before starting state thread

    # Run the default state in a separate thread
```

*Figure 10.3.1: main.py*



```python
# -*- coding:utf-8 -*-
import os
import sys
import time

# Add parent directory to Python path to find initialcode module
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from initialcode.PAJ7620U2 import PAJ7620U2
from config import (PAJ_INT_FLAG1, PAJ_UP, PAJ_DOWN,
                    PAJ_LEFT, PAJ_RIGHT, PAJ_FORWARD, PAJ_BACKWARD)

# Wrap the sensor class for uniformity
class GestureSensor(PAJ7620U2):
    def __init__(self):
        super().__init__()
        self.tasks = []  # Initialize empty tasks list
        self.current_task = 1

    def check_gesture(self):
        data = self._read_byte(PAJ_INT_FLAG1)
        if data:  # Only print when data is received
            print(f"Raw gesture data: {data}")
        if data == PAJ_DOWN:
            print("Down gesture detected from sensor")
            return "DOWN"
        elif data == PAJ_UP:
            return "UP"
        elif data == PAJ_LEFT:
            return "LEFT"
        elif data == PAJ_RIGHT:
            return "RIGHT"
        elif data == PAJ_FORWARD:
            return "FORWARD"
        elif data == PAJ_BACKWARD:
            return "BACKWARD"
        return None

# Export PAJ7620U2 from this module (alias GestureSensor) for compatibility with other files
PAJ7620U2 = GestureSensor

if __name__ == '__main__':
    print("\nGesture Sensor Test Program ...\n")
    sensor = GestureSensor()
    while True:
        gesture = sensor.check_gesture()
        if gesture:
            print(f"Detected gesture: {gesture}")
        time.sleep(0.05)
```

*Figure 10.3.2: task.py*



```python
import os
import subprocess
from config import NAV_AUDIO_DIR, AUDIO_FILES_DIR  # Add AUDIO_FILES_DIR import

def play_bootup_sound():
    file_bootup = os.path.join(NAV_AUDIO_DIR, "bootup.mp3")
    if os.path.exists(file_bootup):
        print("Playing bootup sound...")
        try:
            subprocess.run(["ffplay", "-nodisp", "-autoexit", "-af", "volume=2.0", file_bootup],
                           capture_output=True, text=True)
        except Exception as e:
            print(f"Error playing bootup sound: {e}")
    else:
        print("Bootup file not found:", file_bootup)

def play_upload_confirmation():
    file_confirm = os.path.join(NAV_AUDIO_DIR, "upload_conformation.mp3")
    if os.path.exists(file_confirm):
        print("Playing upload confirmation sound...")
        try:
            subprocess.run(["ffplay", "-nodisp", "-autoexit", "-af", "volume=2.0", file_confirm],
                           capture_output=True, text=True)
        except Exception as e:
            print(f"Error playing upload confirmation: {e}")
    else:
        print("Upload confirmation file not found:", file_confirm)

def play_task_audio(task_number):
    task_file_mp3 = os.path.join(AUDIO_FILES_DIR, f"{task_number}.mp3")
    task_file_m4a = os.path.join(AUDIO_FILES_DIR, f"{task_number}.m4a")

    if os.path.exists(task_file_mp3):
        print(f"Playing task {task_number} (MP3)")
        try:
            subprocess.run(["ffplay", "-nodisp", "-autoexit", "-af", "volume=2.0", task_file_mp3],
                           capture_output=True, text=True)
```

*Figure 10.3.3: audiohelper.py*

**<u>Missolanius Parts (NOT PART OF THESIS):</u>**

- Chapter 3.1

Before we do that, we wanted to look at existing products to help us design our device. One product we found that is coming up using gesture recognition technology is the Mudra Link. The Mudra Link is meant to act like a mouse that navigates your computer using only hand gestures [5]. The device detects a hand gesture the user makes and the device determines what the hand gesture is associated with the commands it's set to output in the computer. For example, if the user puts the thumb and index finger together, the Mudra Link detects that you are selecting something on the computer. This device inspired our project to be wearable and be able to detect a hand gesture, but instead of controlling a computer, our device outputs a task the user must do. Therefore, we needed to find software that would remind user to do their daily tasks.

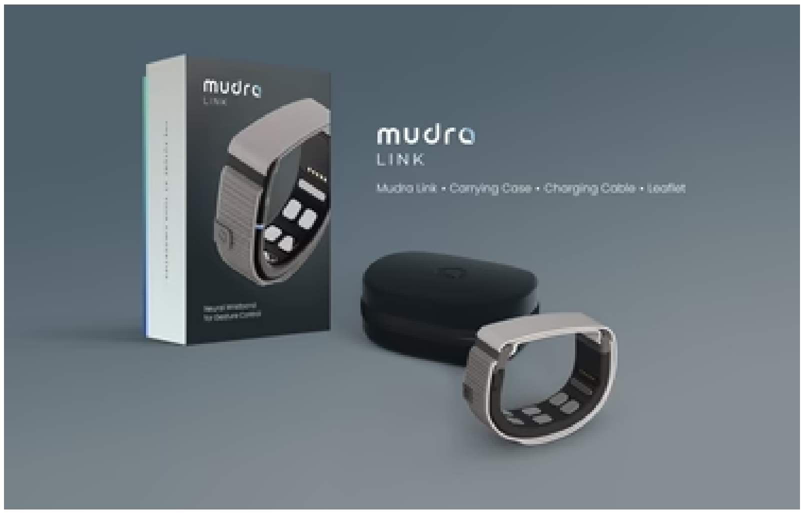


*Figure 3.1.1: Mudra Link [5]*

Another inspiration for our project was the reminder apps that many people in the world use today. Different reminder apps are used daily to help users be reminded of what to do for a certain day. Users will set reminders on tasks that need to be done that day and assign a specific time for a task to be done. It is shown in a study that people who use reminder apps on their smartphones improve their focus and memory in their lives therefore, we took this as inspiration for our project [6].

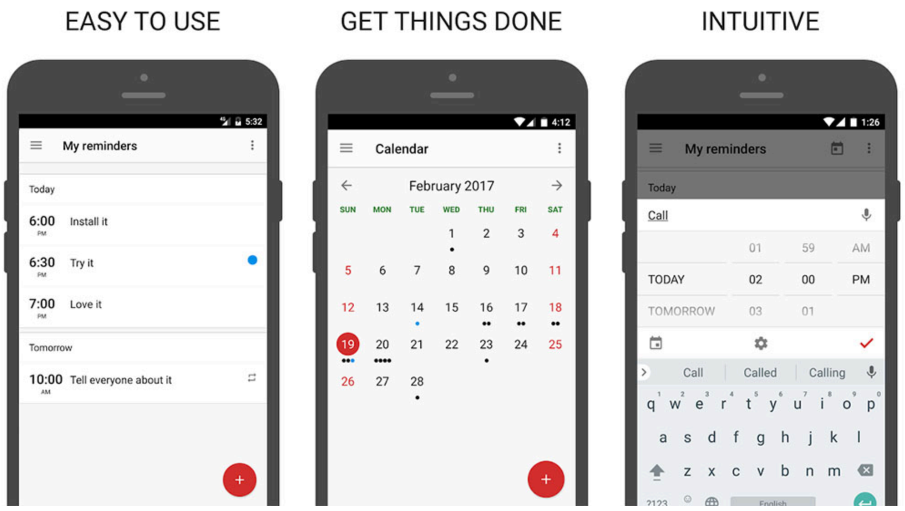
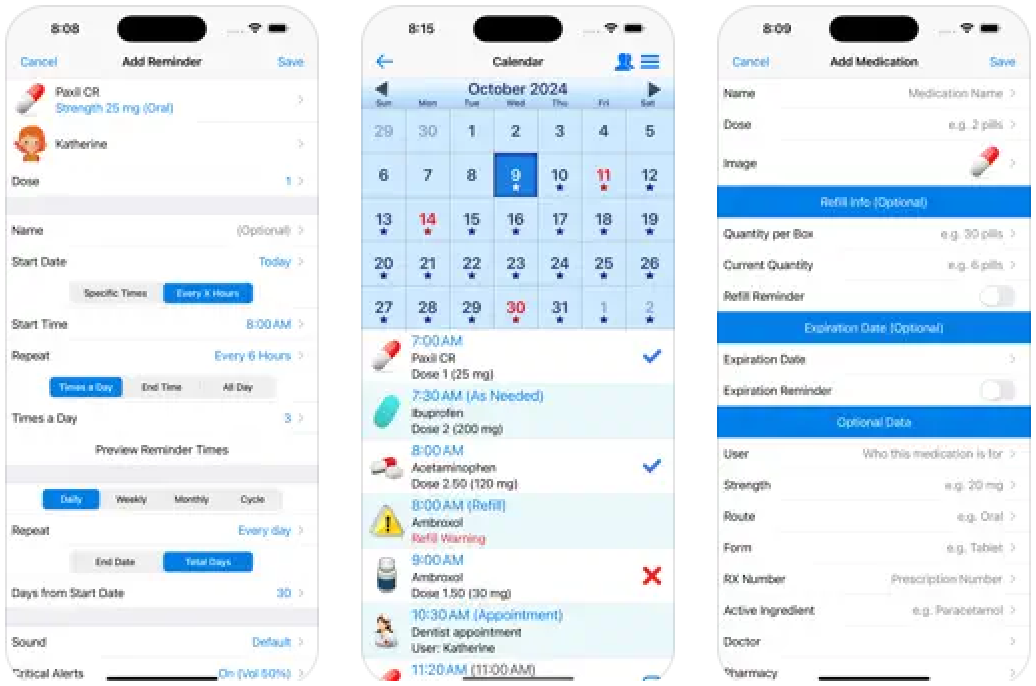


*Figure 3.1.2: Reminder App [7]*

The reminder app shown in Figure 2 is a generic one created by Apple that is text-based and will remind you of the task you added to be reminded at a given time. This is one of the many text-based apps people use today, but there are few audio-based reminder apps. Studies show that hearing someone's voice that they are comfortable with empowers them to do a task. [8]. That begs the question of why there aren't more audio-based reminder apps that empower someone to do tasks rather than a text-based reminder.


8:08

Cancel Add Reminder Save

Paxil CR
Strength 25 mg (Oral)

Katherine

Dose 1

Name (Optional)

Start Date Today

Specific Times | Every X Hours

Start Time 8:00 AM

Repeat Every 6 Hours

Times a Day | End Time | All Day

Times a Day 3

Preview Reminder Times

Daily | Weekly | Monthly | Cycle

Repeat Every day

End Date | Total Days

Days from Start Date 30

Sound Default

Critical Alerts On (Vol 50%)

8:15

Calendar

October 2024

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| 29 | 30 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |

7:00 AM
Paxil CR
Dose 1 (25 mg) ✓

7:30 AM (As Needed)
Ibuprofen
Dose 2 (200 mg)

8:00 AM
Acetaminophen
Dose 2.50 (120 mg) ✓

8:00 AM (Refill)
Ambroxol
Refill Warning

9:00 AM
Ambroxol
Dose 1.50 (30 mg) ✗

10:30 AM (Appointment)
Dentist appointment
User: Katherine

11:20 AM (11:00 AM)

8:09

Cancel Add Medication Save

Name Medication Name

Dose e.g. 2 pills

Image

Refill Info (Optional)

Quantity per Box e.g. 30 pills

Current Quantity e.g. 6 pills

Refill Reminder

Expiration Date (Optional)

Expiration Date

Expiration Reminder

Optional Data

User Who this medication is for

Strength e.g. 20 mg

Route e.g. Oral

Form e.g. Tablet

RX Number Prescription Number

Active Ingredient e.g. Paracetamol

Doctor

Pharmacy

*Figure 3.1.3: Pill Medication Reminder App [9]*

The pill reminder app shown in Figure 3 shows that reminders can be set by either the user or caregiver to take medication at a specific time and can be used audio for reminders. This is beneficial for families to help the user take daily medication with a voice they are comfortable with. Therefore, we see that there are a lot of text-based reminders but only a few audio-based reminder apps, but usually these audio-based reminder apps are more focused on medication, not task-based.  From these three products, we took inspiration from different reminder app concepts that were audio-based and Mudra Link technology to create a device that is cheap, easy to build, and catered toward helping special needs adults.

- Chapter 3.1

For an overview of the GRD Companion, we want to talk about what went into designing the hardware and software of the device. The GRD Companion aims to help special needs adults do everyday tasks and correct their repetitive behavior. To accomplish this, we need to make sure the hardware components are compatible, the software works as we intended, and the integration of both hardware and software is complete.

# - Chapter 7: Ethics

## 7.1 Ethical Justification

As we take a step back from the technical aspects of the project, we want to look at our project from a moral standpoint. From section 1, the main reason why we wanted to do this project to create a device was to help special needs adults. To dive deeper, we saw that special needs adults were having difficulty focusing on simple tasks like brushing their teeth or making lunch for themselves. With that in mind, we decided to create a device that would impact society and human well-being.

### 7.1.1 Societal Impact

The impact on society that our project will have is to open more doors for special needs adults to do activities that may be a challenge for them. Our device is meant to help special needs adults do everyday tasks that are set by the adult's parent or teacher. The hope is that special needs adults listen to these tasks, showing that they are capable of becoming independent. In doing so, this opens the opportunity for special needs adults in the future to have a job on their own, live for themselves, and do more that they can't do right now. The impact on society that our project could have is helping special needs adults with opportunities to help make an impact in the world.

### 7.1.2 Human Well-Being Impact

Special needs adults are looked at differently and need more care and assistance to go on with everyday life. The impact that we envision our product to have on human well-being is to empower special needs adults so that they can do any tasks given to them. Our device is meant to help special needs adults do everyday tasks independently as it is just a reminder from the parent or teacher. As special needs adults use our device, we hope that it empowers them to do things independently and shows that they can do all things. Not only does it empower the special needs adults themselves, but the people around them know that they are capable of doing tasks themselves and trust they will get done. Our project hopes to impact the human well-being of special needs adults and the people around them.

Not only do we look at the impact of our project, but we also use ethical principles like human dignity, justice, and rights to justify the work that we did for the project. Our work on the project takes a look into human dignity and justice, for it allows special needs adults to have equal opportunity to do things that people may overlook as hard or unmanageable. We also took into consideration human rights as we were designing the device to keep the privacy and safety of humans in mind. Our project is centered around showing ethical justification in showing why our project matters and how it will impact and show ethical principles.

## 7.2 Characteristics of Being a Good Engineer

As every engineer wants to show their work is the best, it is also right to know what it means to be a morally good engineer. Throughout the project, it taught us about being responsible engineers. During the design phase of the project, we used ethical engineering ethics to help choose components for the project. For example, we needed a component to help the device see what hand gesture the user is making. We took into consideration privacy and data retrieval as we chose to go with a sensor rather than a camera. Another aspect of being a good engineer is to have professional responsibility. As we talked with other stakeholders, they expressed some changes to us, adding more to the project than what was intended. We considered those comments and modified the project to their needs as we kept the responsibility of doing things to help them. In this project, we portray the growth of becoming good engineers in thinking about ethical engineering ethics in design and portrayal in professional experience.

We also wanted to reflect on the ethical virtues of our project. As mentioned in section 6.1, our work in the project may impact by providing equal opportunities for special needs adults. Now, we look at environmental impact as thought of different harms our device may have to the environment and when choosing components, we wanted to find the best solutions like no radioactivity to provide a safe product to the world. We also want to benefit the people beyond just a technical purpose, as it was expressed to us to help special needs adults to help them correct any repetitive behavior they may do. Based on all this, we see that ethical teamwork is influenced in the decision-making of this project as all agree on every aspect of the project, ensuring helping special needs adults. This is the reflection we had on our ethical virtues and what it meant to us to be a good engineer in this project.

## 7.3 Public Considerations

For our project, there were different considerations when it came to the design and testing phase. Throughout the project, we needed to consider ethical pitfalls when it came to safety, risks, and public considerations.

### 7.3.1 Risks

For our project, there were risks that we and our stakeholders thought of. Some things that may be risks were the risk of overheating and radiation. There is the potential that our device may overheat, whether it be the computer or the battery, causing discomfort. The potential risks of mass exposure to radiation like the Apple Watch have shown in the past. These risks were taken into consideration, and safety precautions were a priority for the project. Therefore, we communicated these risks to the user and asked them to sign any forms involving risks taken for our project.

### 7.3.2 Public Considerations

As we look to those who benefit from and are affected by our project, we look into our audience for the project. We created the device for special needs adults and would benefit them and the people around them. But as we look beyond that, the people who may be affected by our project are the people around them and, potentially, medical businesses. The people around them may feel affected; it is as if we are changing the course of the plan they may have for the special needs adults. Medical businesses can look at our device if successful and see that it leads to a potential threat to products; they could do the same thing our project offers. As it may feel odd that our project may have a negative impact, we stir our focus on the positive to help special needs adults as our device is intended to help them.

In summary, ethical considerations play a crucial role in ensuring the integrity and societal impact of this project. By recognizing ethical justification, addressing characteristics of what it means to be a good engineer, and understanding considerations from safety and risks, we have taken steps to uphold the highest standards of professional responsibility. Moving forward as engineers, we are committed to ethical decision-making, evaluating potential concerns, and adhering to best practices to ensure we are morally good as engineers. For our project, we want to foster trust among our stakeholders to ensure that we are responsible and have a positive impact on society.

- Chapter 5.2

(Once we get IRB approval and do human subjects testing, we will fill in this section with the results we get.)

Notes:

- Concern about the size
- Showcase how our device works today
- Text Vs Voice-based questions
- Time to adjust to the device
- Try to implement a behavior corrector
    - Gyroscope implementation
- Potential no interest if they don’t want to input
- Option to confirm the task depending on the person
- Add a reward function
- Try to add a check function
- Try to make accessible for different body parts
- Might have a microphone recording to see if repetitive words or sounds are repeated consistently
- Try to have the app always updating with the tasks done

Chapter 7:

A feature we thought about is that our device and application can send a task can be sent to the device anywhere, where it needs no connection to external sources.

Chapter 1:

Independence Network is a special needs adult school located in Santa Clara, CA, that furthers special needs adults' education based on their interests. Special needs adults and their teachers need a device that reminds students to do daily tasks set by the teacher. Both the teachers and special needs adults need this device now because it will help the adults overcome different challenges they may face daily.

- References

[1] K. L. Robson, P. Anisef, R. S. Brown, and G. Parekh, “The Intersectionality of Postsecondary Pathways: The Case of High School Students with Special Education Needs,” *Canadian Review of Sociology/Revue canadienne de sociologie*, vol. 51, no. 3, pp. 193–215, Jul. 2014, doi: https://doi.org/10.1111/cars.12044.

[2] Yam, “Addressing the Significant Shortage of Special Education Teachers in California,” *The Arc of California*, Aug. 14, 2023. https://thearcca.org/addressing-the-significant-shortage-of-special-education-teachers-in-california/

[3] D. R. Bryant-Rich and D. E. Barshaw-Rich, “Personal voice operated reminder system,” *U.S. Patent Application* US20120265535A1, filed Sep. 6, 2010, published Oct. 18, 2012.

[4] A. J. Venner et al., “A reminder and actioning system,” *European Patent Application* EP1115076A1, filed Jan. 6, 2000, published Jul. 11, 2001.

[5] R. R. Dunton, "Location based task reminder," *U.S. Patent Application* US2006/0061488A1, Mar. 23, 2006.

[6] E. Yoffie, “Device for reminding users,” *U.S. Patent* US7746730B2, issued Jun. 29, 2010.

[7] R. Grover and J. Tan, *Reminder device wearable by a user*, U.S. Patent Application US20160299572A1, Oct. 13, 2016. [Online]. Available: https://patents.google.com/patent/US20160299572A1

[8] V. Y. Sigounas, “Technologies of care and the engineering imaginary: Two approaches to assistive device design for the Global South,” *Medical Anthropology Quarterly*, Oct. 2023, doi: https://doi.org/10.1111/maq.12818.

[9] Raspberry Pi Ltd, *Raspberry Pi Zero 2 W Product Brief*, Apr. 2024. [Online]. Available: https://pip.raspberrypi.com

[10] Waveshare, "PAJ7620U2 Gesture Sensor," *Waveshare Wiki*, [Online]. Available: https://www.waveshare.com/wiki/PAJ7620U2_Gesture_Sensor. [Accessed: Dec 8, 2024].

[11] PiSugar Kitchen, "PiSugar 2 1200 mAh Raspberry Pi Zero Battery," *PiSugar*, [Online]. Available: https://www.pisugar.com/products/pisugar2-raspberry-pi-zero-battery. [Accessed: Apr. 07, 2025].

[12] Amazon, "Mini Portable Speaker, Compatible with Any Device with 3.5mm Jack," *Amazon.com*, [Online]. Available: https://www.amazon.com/dp/B0C49TKSQQ. [Accessed: Apr. 07, 2025].

[13] Waveshare, "WM8960 Audio HAT," *Waveshare*, [Online]. Available: https://www.waveshare.com/wiki/WM8960_Audio_HAT. [Accessed: Dec 8, 2024].

[14] E. L. Hardman, “THREE CHILDREN WITH EMOTIONAL AND BEHAVIORAL DISORDERS TELL WHY PEOPLE DO RIGHT,” *INTERNATIONAL JOURNAL OF SPECIAL EDUCATION*, vol. 27, no. 1, pp. 160–176, 2012, Available: https://files.eric.ed.gov/fulltext/EJ979722.pdf

[15] A. Attwood, U. Frith, and B. Hermelin, “The understanding and use of interpersonal gestures by autistic and Down’s syndrome children,” *Journal of Autism and Developmental Disorders*, vol. 18, no. 2, pp. 241–257, Jun. 1988, doi: https://doi.org/10.1007/bf02211950.

[16] J. Wu, “A Basic Guide to I 2 C.” Available: https://www.ti.com/lit/pdf/sbaa565

[17] Z. -w. Hu, "I2C Protocol Design for Reusability," 2010 Third International Symposium on Information Processing, Qingdao, China, 2010, pp. 83-86, doi: 10.1109/ISIP.2010.51. keywords: {Writing;Protocols;Design methodology;Clocks;Registers;Timing;Field programmable gate arrays;I2C protocol;reusability;light sensor;RTC;bio-logging},

[18] C. Kumar and S. Raghuveer, “Design guidelines for safer low voltage distribution systems,” *Int. J. Sci. Eng. Res.*, vol. 4, no. 6, pp. 2027–2030, Jun. 2013.

[19] U.S. Department of Justice, *2010 ADA Standards for Accessible Design*, Sep. 15, 2010. [Online]. Available: https://www.ada.gov/2010ADAstandards_index.htm