TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

ABSTRACT

This report presents the design and evaluation of a smart home system tailored for individuals with limited or no limb mobility. With the growing need for independent living among people with disabilities and the elderly, smart homes offer a transformative solution through automation and remote control of daily household tasks. However, most existing systems lack inclusive designs that address the specific needs of users with impaired motor function. To bridge this gap, we developed a minimalist yet scalable smart home framework incorporating a custom adaptive remote control device that uses directional input to interact with various smart appliances. The system enables users to perform essential and semi-essential tasks such as lighting, entertainment, and door control, with minimal physical effort.

CHAPTER 1

Introduction

Independent living involves managing a set of day-to-day activities within a living environment, such as a home. This involves multiple environmental, social, and safety dimensions, which are shaped by the interaction between the individual, their surroundings, and the tasks they perform [1, 2]. A 'smart home' is a concept that enables its resident to accomplish independent living. It is a residential building equipped with a design to facilitate automation, promote independence, and monitor inhabitants in terms of health and safety [3, 4, 5]. Smart homes enhance quality of life by providing automated appliance control and assistive services, improving user comfort through context-aware systems and predefined rules tailored to the home environment's conditions [6].

The development of Internet of Things (IoT) technology has enabled the development of smart devices that can connect to the internet and thus be controlled remotely [7]. This has revolutionized the way some of these appliances work, because a device can be more useful if it is interconnected with other devices. However, the IoT is not just a collection of devices and sensors linked through wired or wireless networks – it is a complex integration of the physical and digital worlds, facilitating seamless communication between people and devices. It can be viewed as an interwoven system of networks of varying scales [8], collectively forming a vast global infrastructure. The development of IoT technology has fundamentally revolutionized smart homes, making them more intelligent, responsive, and capable of enhancing the quality of everyday life [7, 8].

Today, because of IoT, smart homes also support various functions like remote health monitoring, which is becoming critical due to a rapidly aging population, the high cost of formal healthcare, and the strong desire within individuals to remain independent within their own homes [4]. Additionally, the smart home concept also has the potential to offer vital support to people with disabilities and those with chronic illnesses who live alone. These systems not only help the residents accomplish their daily tasks but also provide a medium for people to stay connected with healthcare providers and caregivers when needed, allowing early detection of health risks and timely intervention [8].

This is important because, according to the World Health Organization, almost 15% of the world's total population, which amounts to around 785 million, suffers from some form of physical or cognitive disability [9]. Some of them have disabilities caused by age, while others have disabilities due to various factors like accidents, stroke, mental shock, genetic disorders, etc. According to the Center for Disease Control and Prevention (CDC) in the United States, 35.2 million adults experience difficulties with physical functioning, representing 13.5% of the adult population. Additionally, 75.4 million adults struggle to perform at least one physical activity without difficulty, accounting for approximately 32.9% of the adult population [9]. These statistics highlight a significant national and global need for assistive technologies [9].

Additionally, navigating a home presents numerous challenges for individuals with disabilities, often limiting their independence and safety. Those with mobility impairments may need help with tasks like turning off lights, adjusting thermostats, or opening blinds. Cognitive disabilities can make managing household routines, remembering schedules, or operating appliances difficult, increasing reliance on caregivers. Similarly, people with visual or hearing impairments may struggle with traditional home interfaces, such as reading displays or hearing alerts, which disrupt daily living. Safety concerns like locking doors, responding to emergencies, or monitoring security further heighten vulnerability, with specific challenges varying by disability type [10, 11].

Existing homes are inadequate since they tend to be very niche and only support a few individuals based on certain designs. Many smart home technologies are designed with the general population in mind, neglecting the unique requirements of users with disabilities. This leads to a lack of adaptability and personalization for many users. Similarly, the design of many current home systems and their user interface fails to consider individuals with visual impairments, many of whom may struggle with interfaces that rely on visual cues and feedback. There is also a gap in integrating assistive technologies, which can hinder the usability of these systems for users who rely on them for daily functions. Another often overlooked aspect is the high cost, which can be a prohibitive burden for many individuals [12].

This report discussed a smart home concept for users who experience mobility limitations and impaired limb function. This includes individuals with conditions such as cerebral palsy, multiple sclerosis, spinal cord injuries, arthritis, or stroke-related paralysis, many of whom may have weakened hand control, limited dexterity, or loss of function in their legs. The report explores a concept of a directional control device (remote), network architecture, and certain smart devices and interfaces that enable independent living for people with little to no limb mobility. While the list of smart devices can be extended as needed, this report shall take a minimalist approach to exploring how directional control, through an adaptive remote,

can enable control of a wide variety of assistive devices with numerous functions. At the end, the report also reports the performance and user experience of the adaptive control device and explains how this concept might be incorporated with other assistive input interfaces.

CHAPTER 2

Previous Work

A recent study by Peresa et al. [13] proposed a user-centric, data-driven approach for developing smart homes for people with disabilities. The authors centered the smart home architecture into four layers: perception and actuation, network/transmission, preprocessing, and applications and devices. The authors claimed that, based on this framework, it is possible to create a smart home for people with disabilities, depending on the type and level of impairment. However, the study still remains very conceptual in nature, and not much research has been done on the human-centric design for each type of disability. Additionally, the study does not address the ethical implications of collecting, processing, and storing personal real-time health data of individuals. The study also assumes that the users will want to engage with predictive information systems, however, it does not discuss situations where trust, technology fatigue, and system automation might be intrusive and confusing.

An important consideration in designing smart homes for people with disabilities is the development of adaptive control systems. The study by Wang et. al [14] discusses an approach that uses various sensors to collect data on user behavior, which is then analyzed through activity recognition and machine learning to enable autonomous control of smart devices. While sensor-based activity recognition may support effective automation, the study does not thoroughly address the decision-making mechanisms or the role of ontological mappings in determining device behavior. Moreover, it remains unclear how users can regulate the system, especially when their behavior or activity patterns change suddenly or over time. Additionally, there might be concerns about how this system might be used for more complex homes, housing multiple people with disabilities.

Mtshali and Khubisa [15] proposed a smart home appliance control system using voice-command systems like Amazon Alexa, Google Home, etc., in conjunction with camera-based surveillance, to control various devices. While voice recognition is an effective and popular method for developing smart homes for people with limb mobility disorders, relying solely on it comes with disadvantages. It is because, over time, it is encouraged that the residents

use their residual limb function to facilitate rehabilitation, rather than completely relying on a method that does not encourage limb mobility.

Bacchin et al. [16] also conducted research on co-housing systems like DOHMO to support caregivers and improve the quality of life for individuals with disabilities. In their study, caregivers were asked to perform specific automation tasks using the DOHMO system and were later interviewed about their willingness to delegate such tasks to home automation. While most participants, primarily caregivers, acknowledged that systems like DOHMO are generally effective, they also raised concerns about the importance of providing alternative controls. These included manual overrides for unexpected situations and the ability to operate the system without relying on an internet connection. However, the study did not explore how such systems could offer alternative input methods specifically for people with disabilities, nor did it address how patients themselves might independently control devices or request assistance from caregivers when needed.

CHAPTER 3

Concept

3.1 System Architecture

The diagram showcases the high-level connections between all the devices and hardware. At the central focal point is the Hub. The arrows indicate a form of communication or connections, which for most of the devices here is handled via a local WiFi network using MQTT as the communication protocol. The exceptions here include the Smart TV and the Smart Speaker, as the Pico serves as a courier of HTTP requests/responses, and the ESP32 uses HTTP requests respectively. The Smart Mobile Device (SMD) communicates with the Hub using REST API calls, structured HTTP web requests that tell the system which action to perform. The Kick buttons (SALE-Remote) communicate with the SMD over Bluetooth Low-Energy (BLE). The Remote Support Station is represented by any computer with access to the Home Assistant website that communicates via HTTPS.

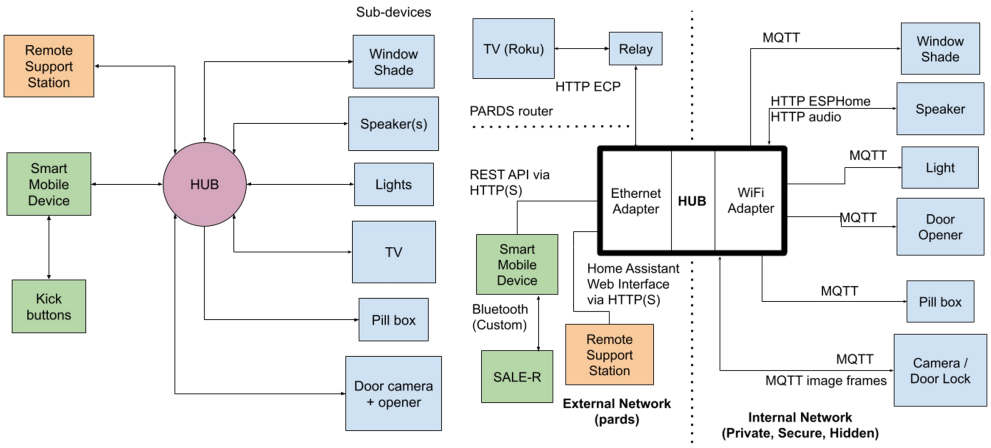

Figure 3.1: System Configuration for Project SALE

3.2 Wireless Network Protocols

3.2.1 Bluetooth Low-Energy

Bluetooth Low Energy is a protocol for wireless communication that operates on the same 2.4GHz bandwidth as Bluetooth. [17] However, unlike Bluetooth, the protocol enters a sleep mode when not in use. [18] It saves much more power than having a constant Bluetooth Classic protocol running at full power. [17] Bluetooth Classic is better for larger amounts of data or when minimizing latency is an important factor while power consumption is not. [17] However, when the application suits it, Bluetooth Low Energy often has lower cost of development and wider device support than similar competitors. [18]

3.2.2 MQTT

MQTT is a lightweight connectivity protocol designed around a publish/subscribe method of message transportation. [19] One device sends out, or publishes, a message to a broker. [20] The broker then distributes those messages to devices that are subscribers to that topic. [21] It is important to note that all MQTT clients can both send and receive, and are only publishers or subscribers relative to a message transfer. [20] Because the broker handles most of the load of the connection, the software on the clients is lightweight and clients are able to have very low minimum hardware requirements, which is great for small microcontrollers. [21]

3.3 Target Population

The smart home design may cater to four categories of population, categorized based on how they most comfortably issue commands:

- Aging population: Older people who have limited physical strength and mobility, and have resultant difficulties managing their daily tasks, like turning off appliances, managing medications, etc. They may manage the smart home through the SALE Android app.
- People with limited forelimb (hand) mobility: This includes individuals who have congenital conditions, injuries, or neurological disorders that limit the hand mobility and fine motor skills, like using fingers to use a touch screen interface. The adaptive device enables gross hand motions (like fist pounding) to control the smart home.

- People with no hand mobility and hind limb mobility: This includes people who are not able to use their hands due to paralysis, amputation, or serious neuromuscular diseases. The adaptive control device can use leg motions (kicks) to control the smart home.

- People with no limb mobility: This includes individuals who have lost or were born without both upper and lower limbs face significant challenges in interacting with their environment. For them, our smart home system can provide comprehensive limbs-free control through voice recognition.

All access methods are enabled by default, so as a person moves between classes or discovers their preferences for controlling the home, they may use multiple methods.

CHAPTER 4

Centralized Control

4.1 Smart Hub

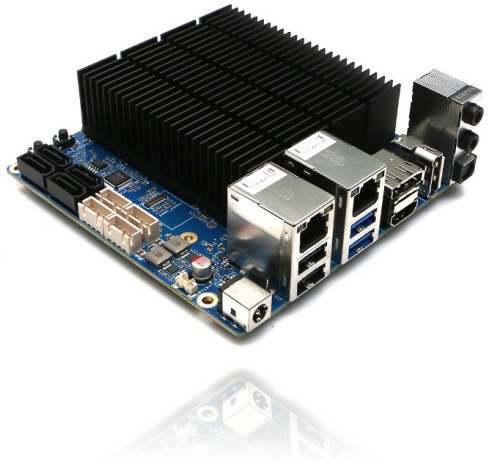

Figure 4.1: Smart Hub Hardware: ODroid H4

4.1.1 Objective and Design Philosophy

The Smart Hub is the central computing device that manages and coordinates all smart home operations. It is built around the ODroid H4, a full-featured x86 desktop-class machine[22]. We chose to run Debian 13 ("trixie") as our operating system. The primary program is the Home Assistant, a customizable and open-source smart home integration platform. The

system is configured to stay on the stable release branch of Debian with automatic updates enabled, reducing the risk of major failures from system changes while still receiving security patches. By using Home Assistant, the Smart Hub provides interoperability between commercial and custom devices, enabling the Remote Support Station (RSS), Smart Mobile Device (SMD), and other components to interact seamlessly. Its open-source ecosystem ensures future scalability and easy adaptation to new technologies.

4.1.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- ODroid H4: Hosts all core smart home software and provides local compute resources
- Debian 13 ("trixie"): Operating system running on the hub
- Home Assistant: Central smart home integration and automation platform
- IEEE 802.11g Wi-Fi Network: Local-only network hosted via hostapd
- Mosquitto MQTT Broker: Manages messaging between smart devices
- HTTP Server and REST API: Facilitates command or control from the SMD
- Log Storage System: Retains 30 days of historical device data

4.1.3 Implementation Strategy

Upon startup, the Smart Hub initializes a Wi-Fi network through hostapd, establishing an isolated 802.11g access point with restricted routing policies. All smart devices, except the Smart Speaker and Roku Relay, connect through this network using the MQTT protocol. Each device publishes or subscribes to specific MQTT topics managed by the Mosquitto server hosted on the hub. MQTT was selected due to its lightweight, reliable messaging capabilities and its compatibility with Home Assistant's auto-configuration APIs. This enables smart devices to register themselves upon connection, significantly reducing manual setup. In addition to MQTT communication, the hub runs an HTTP server that exposes a REST API used by the Smart Mobile Device to trigger device actions such as turning on lights or initiating video calls. Commands are received as HTTP POST requests and interpreted by Home Assistant to activate the appropriate devices.

4.1.4 Engineering Challenges

4.1.5 Performance

The Smart Hub has shown strong performance across all expected functionalities. MQTT communication is reliable and has low-latency, enabling responsive interaction between devices. The REST API receives and processes commands from the Smart Mobile Device without noticeable delay.

4.2 Smart Mobile Device

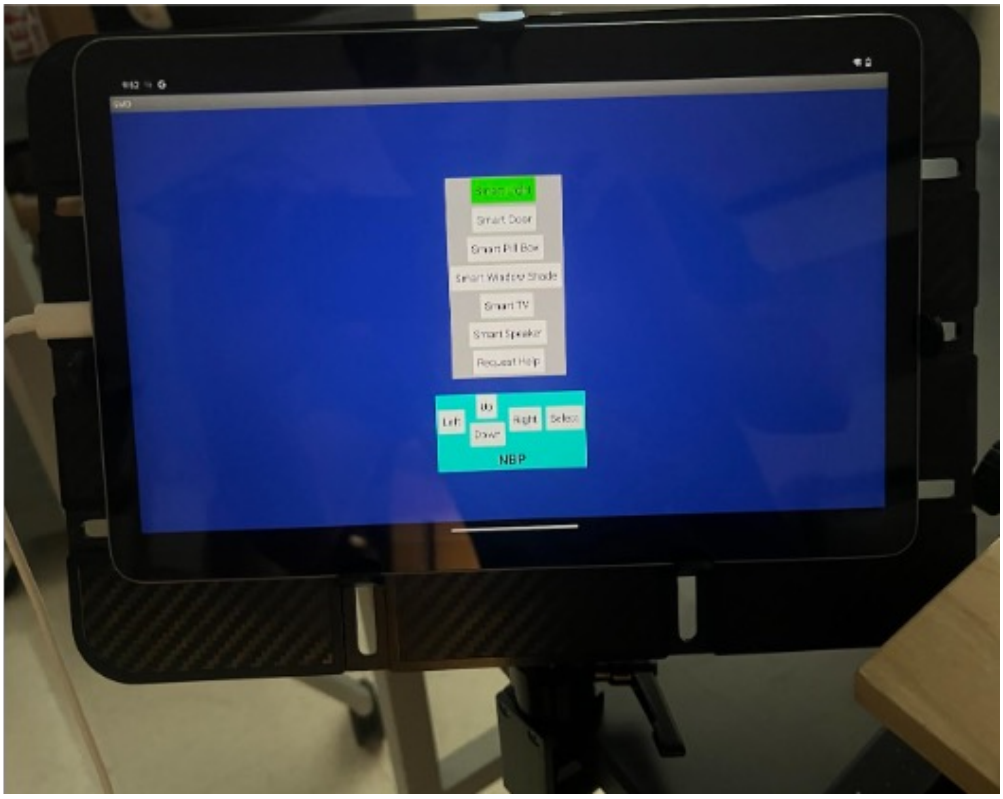

Figure 4.2: Smart Mobile Device

4.2.1 Objective and Design Philosophy

The Smart Mobile Device (SMD) was designed to give users an easy way to control everything in the smart home from a user-friendly and intuitive graphical user interface. It's a custom Android application housed on an Android device, such as a Google Pixel Tablet. The app was built using MIT App Inventor, a web-based development platform that uses block-based

coding and is useful for rapid prototyping and proof-of-concept design. The app lets users navigate through various menu options using either the touchscreen or SALE-R inputs. The kick buttons connect to the app over Bluetooth Low-Energy (BLE), allowing the resident to move through the interface without using their hands. Once a command is selected (such as turning on a light or opening a window shade), the app sends that command to the Smart Hub using a web request. The app emphasizes accessibility, ease of navigation, and reliable wireless communication with both input and output devices in the smart home environment.

4.2.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- Android Tablet: Runs the SMD app and displays the interface
- MIT App Inventor: Used to build the app using block-based programming [23]
- Bluetooth Low Energy (BLE): Handles wireless input connection from peripherals [24, 25]
- REST API over HTTPS: Used by the app to send commands to the Smart Hub [26, 27]

4.2.3 Implementation Strategy

After startup, the SMD runs an initialization function that sets up the interface and establishes a Bluetooth Low Energy (BLE) connection to the kick buttons by connecting to the correct GATT service and characteristic using predefined UUIDs. Once connected, the resident can begin interacting with the system using either the touchscreen or directional kick button inputs.

The app interface is implemented through various different menu states, with each state tied to a specific device (or the main menu for device selection). These states are managed using nested if-else logic that determines what the current menu is and how inputs should be interpreted. Each menu corresponds to a specific device or function that can map directly to kick button inputs, with the exception of the Smart Speaker and the Smart TV, as both devices have multiple inputs and require their own menus (and, in the instance of the Smart TV, submenus). As the resident moves through the interface, inputs update the current state, controlling what actions are available and what screen is displayed.

Once an action (such as toggling a light, adjusting volume, or requesting assistance) is made, the app sends a REST API call via an HTTP request to the Smart Hub to carry out

the selected command. The Smart Hub works with Home Assistant to interpret the request and activate the appropriate functionality in the target device.

To ensure smooth interaction, kick button inputs received over BLE are debounced through custom logic built into the app, while touchscreen inputs are automatically debounced by MIT App Inventor.

To implement the Smart Mobile Device application from scratch, a new user should first visit the public GitHub repository at https://github.com/ECE-492-SALE/SALE_ECE491-02/tree/main. From the main directory, navigate to the `Smart Mobile Device` folder and download the most recent version of the application, which has been exported as a `.aia` project file.

Next, open MIT App Inventor (available for free at https://appinventor.mit.edu/) and create a new account if needed. Import the downloaded `.aia` file to load the project. Once opened, use App Inventor to build and download the corresponding `.apk` file. This `.apk` can then be transferred to an Android tablet with both Bluetooth and internet capability (e.g., the Google Pixel Tablet) either via USB file transfer or email (typically within a ZIP archive to avoid attachment restrictions). Once on the tablet, the file can be run directly from a file manager to install the app.

Before building and transferring the application, ensure that the correct Bluetooth UUID values are set within the project. Specifically, the two global variables `ServiceUUID` and `btnUUID` must be updated to reflect the BLE service and characteristic UUIDs used by the current iteration of the SALE-R microcontroller responsible for kick button input.

For proper web request functionality, the tablet should be connected to the secure Wi-Fi network associated with the current system build. As long as the Home Assistant instance's IP address matches the value used within the project's Web component blocks, the REST API calls will execute successfully.

4.2.4 Engineering Challenges

- BLE connections required significant development time. The app was originally planned in Android Studio using `Java` and `XML`, but poor documentation and deprecated functions made progress difficult. Development was shifted to MIT App Inventor, but even then, importing the correct BLE extension, selecting the right version, and properly configuring BLE notifications made the initial setup frustrating. However, we ultimately achieved full functionality and established a stable connection between the kick buttons and the SMD.

- As the number of supported devices increased, maintaining a clear and functional user

interaction model became challenging. A nested if-else structure was used to manage control states across menus and submenus, and this logic had to be revised frequently as new devices were added and usability improvements were made. Submenus were especially difficult to implement, particularly when they required refactoring already-established logic.

- Directional input from kick buttons sometimes triggered repeated signals. A logic-based debouncing system was added to filter out rapid or duplicate BLE inputs and ensure stable navigation.

4.2.5 Performance

The SMD demonstrated reliable performance during normal operation. BLE inputs were recognized with a negligible delay and consistency, and touchscreen responses were immediate. Users were able to navigate menus and trigger smart home commands with ease.

4.3 Remote Support Station

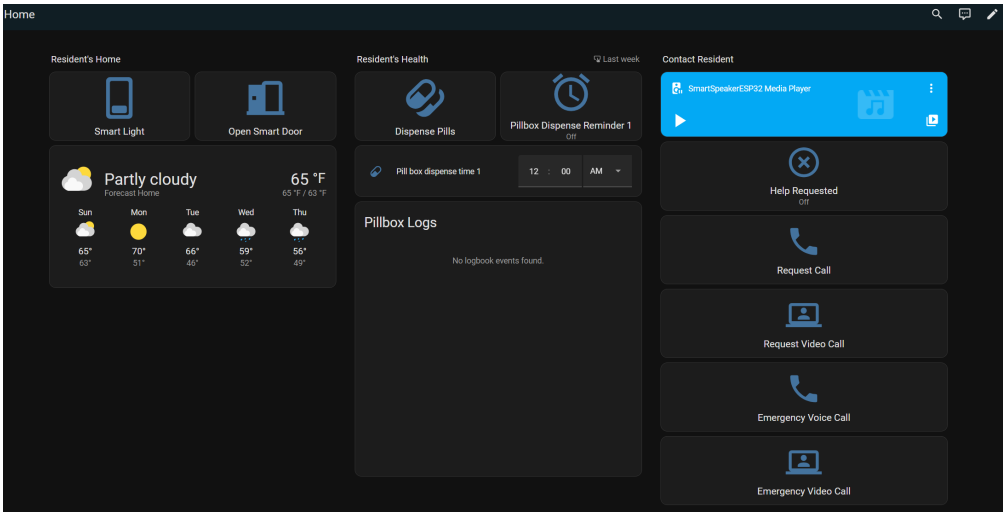

Figure 4.3: Remote Support Station

4.3.1 Objective and Design Philosophy

The Remote Support Station (RSS) is a specialized website that allows caregivers or family members to communicate with senior and/or disabled residents of the smart home and support them without compromising their privacy. The RSS provides remote support staff

access to information from certain smart home devices: the Smart Pillbox, the Smart Door, the Smart Light, and the Smart Mobile Device (SMD).

4.3.2 System Architecture and Core Components

Home Assistant: Home Assistant is an open-source home automation platform designed to run on devices like a Raspberry Pi or a local server. It comes with customization for dashboards, users, and automations.

4.3.3 Implementation Strategy

To set up the RSS, log into the owner account on Home Assistant. git pull the Hub files from the github, or copy configuration.yaml and automations.yaml into your Home Assistant file editor. The File editor is accessible via the navigation menu on the left of the dashboard.

The system is built around the Home Assistant software. The Remote Support station uses a custom dashboard [28] and non-admin user account to handle access limitations [29]. It also uses a custom input boolean to see when the resident requires help. It uses a helper input boolean to turn medication reminders on/off. To set the medication reminder time, we installed the datetime integration [30] and set up an automation that triggered at the set time. When the set time arrives, the automation checks if medication reminders are on. If medication reminders are on, the speaker plays a reminder and the Smart Pill Box dispenses pills.

4.3.4 Engineering Challenges and Solutions

- The original design involved the ability to send calls to/from the RSS and a custom app on the SMD. Due to issues with programming the SMD app, we did not have enough time to implement calls.

- The original design planned to use the Home Assistant Authentication system to implement limited permissions for the RSS user. However, permissions are not yet enabled on Home Assistant. This means that the RSS user is only limited by the configuration file for their dashboard.

- The original design had the RSS user creating automations in order to set reminders for the Smart Pill Box. In order to allow the RSS user to access Home Assistant automations, they had to be an admin user. This caused issues with implementing

limited permissions. We resolved this by adding the datetime integration to allow the RSSuser to schedule reminders without directly accessing automations.

4.3.5 Performance

The Remote Support Station is able to see the logs and control the devices that it should have access to. The call functionality to and from the SMD has not been programmed yet, and is currently represented by audio feedback from the Smart Speaker.

4.4 Home Assistant Website

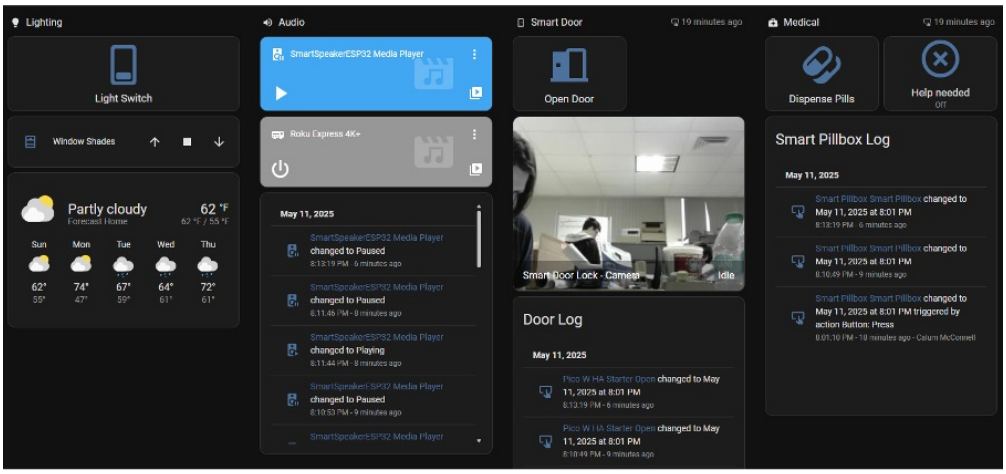

Figure 4.4: Home Assistant General Website

4.4.1 Objective and Design Philosophy

The Home Assistant Website is a specialized website that allows residents to control their homes.

4.4.2 System Architecture and Core Components

Home Assistant: Home Assistant is an open-source home automation platform designed to run on devices like a Raspberry Pi or a local server.

4.4.3 Implementation Strategy

Set up the Home Assistant following the instructions for the Smart Hub and Remote Support Station. You can add additional dashboards and automations using the Home Assistant GUI.

4.4.4 Engineering Challenges and Solutions

- This was relatively simple to implement. The challenges mainly resulted from selecting the wrong device/entity when we set up the dashboard and automations.

4.4.5 Performance

The Home Assistant Website is able to see the logs and control the devices that it should have access to. It is able to request help and alert the RSS staff.

CHAPTER 5

Assistive Control

5.1 SALE-R

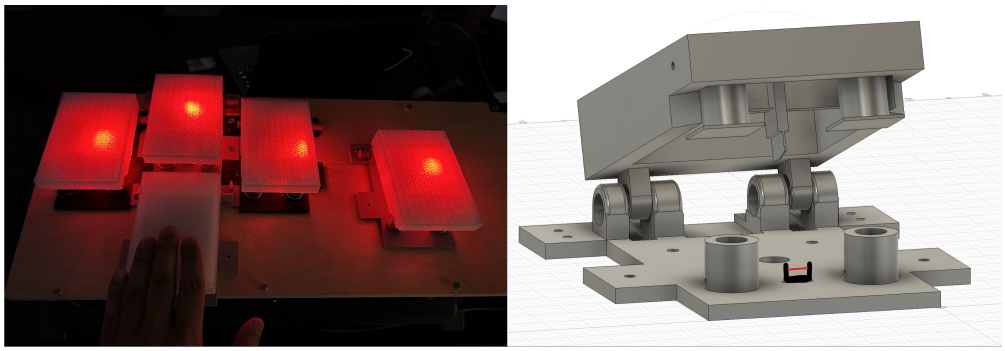

Figure 5.1: SALE Remote (SALE-R)

5.1.1 Design Philosophy

The SALE-R (SAY-lor) buttons are designed to offer directional control to the resident, allowing them to navigate the smart home interface using up, down, left, right, and select commands. These buttons serve both assistive and rehabilitative purposes. While they enable the user to control the entire smart home system through the Smart Motion Device (SMD), they also promote the use of residual limb mobility. Each time a button is physically pressed, the corresponding User Interface (UI) button in the bottom window of the Smart Mobile Device app lights up green.

5.1.2 Core Components

- Optical Sensor: Each button is equipped with an optical sensor. The button mount includes a spring mechanism, and when the button is pressed, an internal flag interrupts

the laser beam inside the optical switch. This interruption signals the sensor that the button has been activated, thereby registering a button press.

- LED-circuit: Each button has an Light Emitting Diode (LED) circuit, comprising two LEDs, which light up red when the buttons are not pressed. Every time the buttons are pressed, the LEDs turn off.

5.1.3 Implementation

The primary circuitry of the SALE-R consists of a printed circuit board that detects button press signals, processes them, and sends them to the Smart Mobile Device using Bluetooth Low Energy (through the Pico W). The main board is called SALE-R Board (Figure 5.2). The kick buttons were custom-designed and 3-D printed, with the physical descriptors shown in Figure 5.1. Additionally, each of the individual buttons have an LED board (Figure 5.3), that lights up two LEDs when the button is not pressed, and turns the LEDs off when the buttons are pressed. The schematics for these circuits are included in Appendix C and GitHUB, and the code to run the BLE modules is included in SALE-R Bluetooth Folder

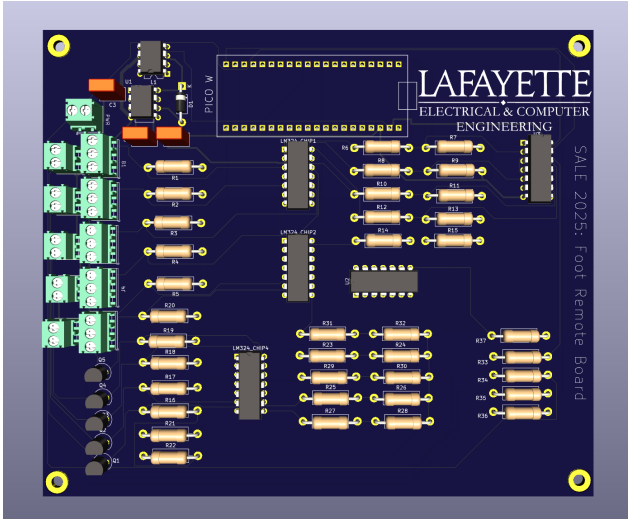

Figure 5.2: SALE Remote (SALE-R) Main Board

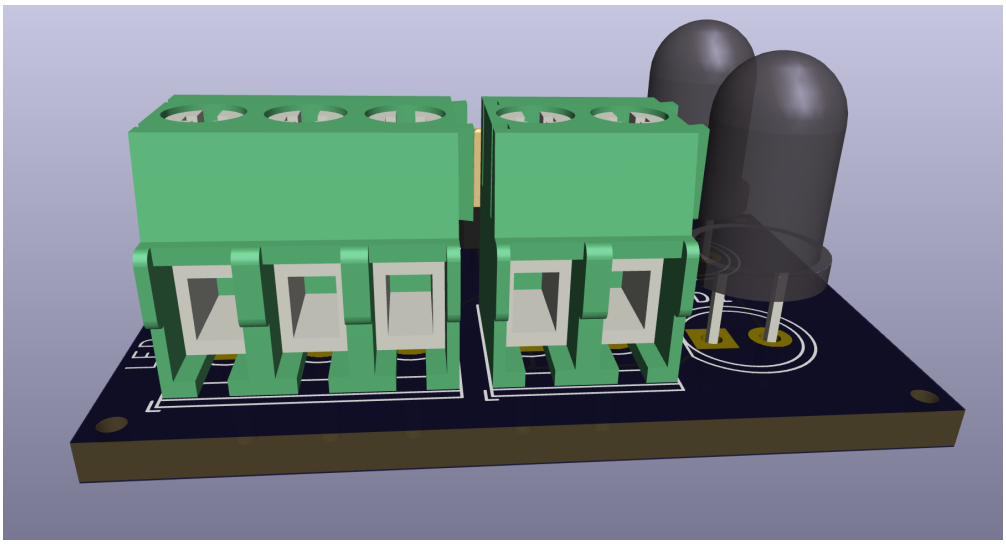

Figure 5.3: SALE Remote LED Board for Individual Buttons

5.1.4 Engineering Challenges and Solutions

- The main challenge the enabling wireless communication to the smart mobile device. While the SALE-R could connect to the HUB's private wifi network, relaying the button press signals from the SALE-R to the HUB and back to the smart mobile device would induce delays. Hence, the way to solve this problem was to enable direct communication between the SALE-R and the Smart Mobile Device. This issue was solved using enabling communication over Bluetooth Low Energy.

- Another challenge was designing a layout that would be ergonomically intuitive for the user. This was addressed by testing multiple configurations and ultimately selecting the one that offered the best combination of usability and compactness.

5.1.5 Performance

The SALE-R circuit can detect button presses almost instantly and transmit the results without delay. At present, the buttons are programmed such that the user has to hold the buttons for at least 0.2 seconds for it to register a button press. However, this value can be custom-programmed based on the needs of the user.

5.2 Voice Control

The home assistant ecosystem dedicated a year of development to implementing and integrating voice assistant functionality. We leverage this in our design to provide a voice assistant, which can control devices around the home. We set up the system to use all-local processing, allowing it to work offline and without exposing user data to easily-hacked third parties. While the system can support many different "assist satellites", devices which provide microphone streams for processing by the hub, only the smart speaker was appropriately configured.

A wakeword engine, openwakeword, continuously monitors the audio stream for the configured word. When it is received, the engine begins recording and transmits the recording to OpenAI's Whisper speech-to-text (STT) engine[31]. Whisper transcribes the recording, and then Home Assistant's internal engine interprets the meaning of the words[32]. Then, a response is generated, which is sent to Piper, a text-to-speech (TTS) program[33]. This generates an audio clip of the response being read, which is played on the speaker.

CHAPTER 6

Smart Devices

6.1 Smart Speaker

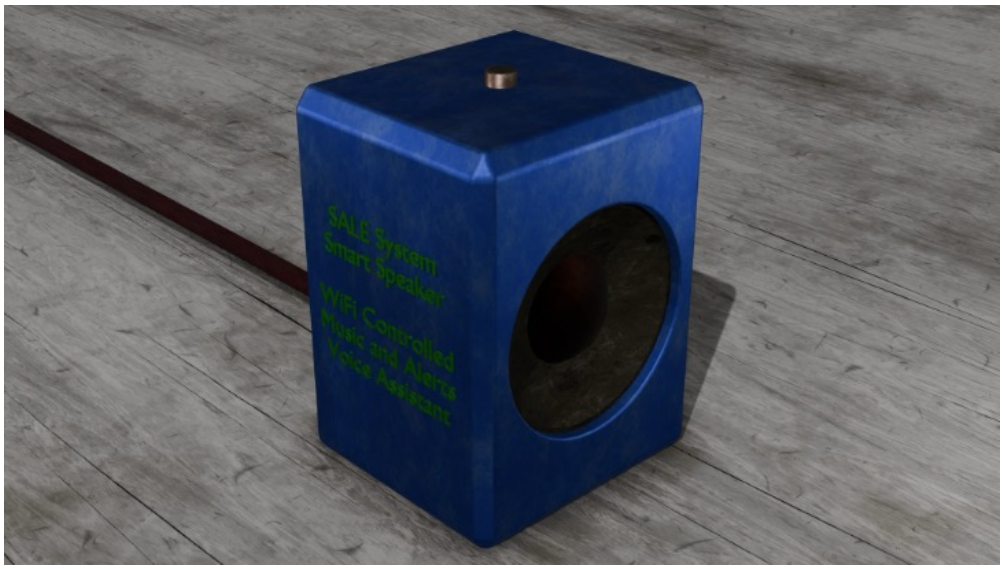

Figure 6.1: Smart Speaker Hardware Rendering

6.1.1 Objective and Design Philosophy

The smart speaker is intended as one of the primary elements of the SALE system. The Smart TV is one of two components that provide entertainment to the user, and with the Smart Mobile Device, it is also one of two that provides a large number of options to the user to interact with. The speaker is intended to provide high-quality audio playback, and is carefully designed to achieve this goal. It also interfaces with the Smart Hub to provide a voice assistant functionality, with an inbuilt microphone that feeds the speech recognition functions on the hub.

6.1.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- Microcontroller (Originally Raspberry Pi Pico W (RP2040 core), later Arduino Nano ESP32 (ESP32-S3 core).
- External DAC: LTC1655LCN8
- Audio Amplifier: LM384N
- Microphone with integrated preamp and compressor: Adafruit 1713
- Voltage Converter: LM2675N-5 and associated components.

6.1.3 Implementation Strategy

The smart speaker is built on a custom single-board PCB design, which integrates several distinct subsystems. It includes a from-scratch 5V power supply circuit, a microphone, a discrete DAC and audio amplifier, and a microcontroller. The board design was originally made for a Raspberry Pi Pico WH, but due to time constraints, an Arduino Nano ESP32 was substituted in its pace. This substitution allowed the use of the pre-existing Voice Assistant module of the open-source ESPHome software suite. The suite includes many existing software modules for microcontrollers to driver various pieces of hardware. However, as I discovered along the way, the primary hardware components I used in my design were not supported directly by the ESPHome software suite. This included the external DAC, which provides the speaker output, and the ADC, integrated into both microcontrollers, which provides the speaker data input. Without these components, the speaker would not work; so we began writing drivers for them in the ESPHome ecosystem.

The ESP32 includes support for driving DAC components via I2S, a protocol where a stream of signed integers are sent to compatible chips. This support has a driver written for it in ESPHome. Unfortunately, the chip I selected does not provide a I2S interface, which meant I had to improvise. I2S has a few variants, one of which is almost identical to the SPI interface that my DAC provides. As the ESPHome doesn't have any existing drivers for continuously sending SPI data, I decided to build on the existing I2S driver instead. After a number of false starts, I determined the signed/unsigned problem and wrote an appropriate driver. The driver furthermore intercepts and implements volume changes on its own, since multiplication on signed integers yields very different results than on unsigned.

In addition to the custom DAC driver to perform these conversions, as well as carefully configuring the I2S driver to behave in an SPI-compatible way, I also wrote a custom ADC

driver for the ADC built into the ESP32. This was surprisingly not yet supported by ESPHome, and was also significantly more complicated to write effectively. The code needed to run this was significantly more complicated, requiring me to understand the interface of the DMA and ADC modules of the ESP32. However, I successfully wrote this custom code, and the data will stream successfully. From here, I used the ESPHome interface to configure and connect the voice assistant, based on other implementations. ESPHome supports over-the-air updates, as well as many other features, which I configured on the device.

The smart speaker functions as a media player, capable of playing any audio saved to the hub. It allows volume control, as well as pausing and playing. It functions as a satellite for the voice assistant, streaming microphone audio to the hub for voice processing and playing back the responses. It has a wide dynamic range, with carefully selected speaker drivers to maximize the audio quality.

6.1.4 Engineering Challenges and Solutions

- Potentiometer having too high of a total resistance, adding noise; swapped potentiometer, though this damaged the PCB and required a second board
- Mis-routed power pins (VCC not connected); fixed with a jumper
- Complexity of home assistant streaming; fixed with ESPHome
- Inadequate ESPHome support for RP2040; migrated to ESP32-S3
- Missing ADC Driver; wrote one
- Signed/Unsigned confusion; wrote custom driver Low speaker impedance limiting undistorted sound output; added a second speaker in series

6.1.5 Performance

Below, Figure 6.2 shows how a sine wave is distorted by simply ignoring the signed-to-unsigned conversion that was needed to allow both the ADC and DAC drivers to work; while there is still an audible tone with the correct fundamental frequency, it is massively distorted.

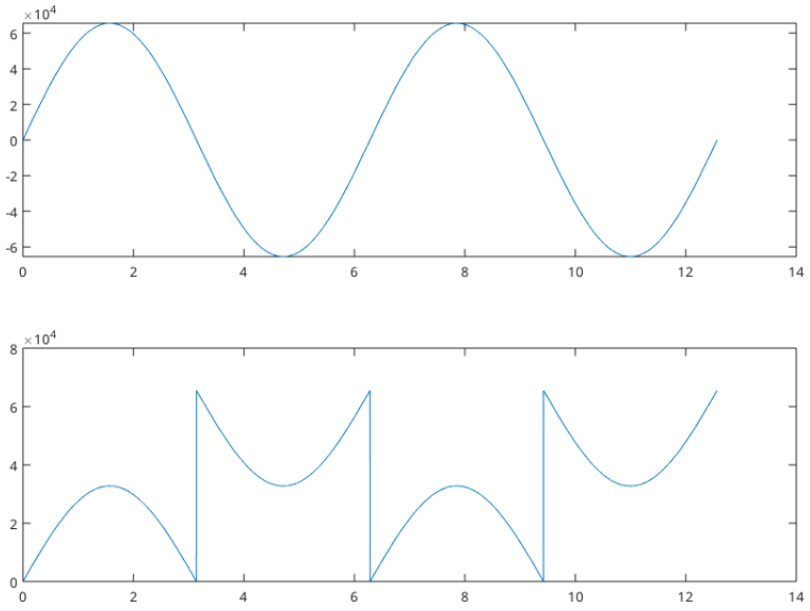

Figure 6.2: Smart Speaker Error: treating signed numbers as unsigned

Figure 6.3 shows the same sine wave, where the conversion to unsigned values is done properly, but where the volume is done as though the values were still signed. In other words, the result of multiplying the wave by a constant as though it were a signed integer, when it has already been converted to be unsigned.

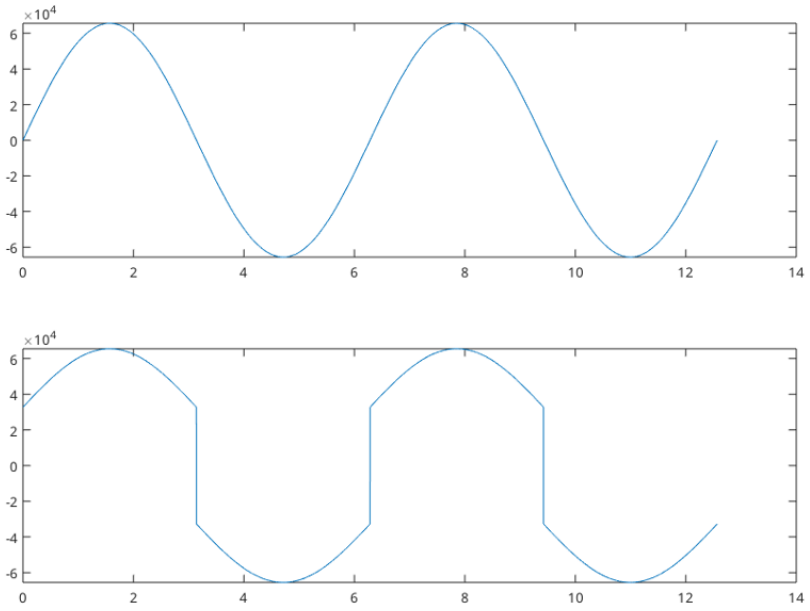

Figure 6.3: Smart Speaker Performance 2

6.2 Smart Light

Figure 6.4: Smart Light

6.2.1 Objective and Design Philosophy

The goal of this device is to develop a smart lighting system able to be easily controlled by a disabled resident or their caregiver. The system is designed to function as a reliable and safety regulation-compliant lighting system for the smart home. Emphasis is placed on ease of use and safety in case of a power outage.

6.2.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- LM2675N Buck Converter: Converts power input to the correct voltage for use by the pico (5V)
- 7 Watt Replaceable LED Light Bulb: Emits light when toggled for the lighting system
- Backup Battery: Backup battery automatically powers the light during power cuts, for up to 90 minutes as per the National Fire Protection Association 101 Life Safety Code[34].

6.2.3 Implementation Strategy

This implementation uses a very simple commercial light bulb for cheap and easy replacement. Control of the light is achieved by using the PicoW_HomeAssistant_Starter project as a base to control a GPIO pin. This pin connects to a transistor which allows the Pico to toggle a higher voltage light.

6.2.4 Engineering Challenges and Solutions

- Concern about the high voltage of the light frying the Pico. An optoisolator was used to make sure the Pico did not receive too much power.

6.2.5 Performance

The lights are able to be toggled using the Kick Buttons, directly through the Smart Mobile Device interface, and from the Remote Support Station website. The light bulb is bright, but not too hot, and turns on in a timely fashion after being wirelessly prompted.

6.3 Smart Pill Box

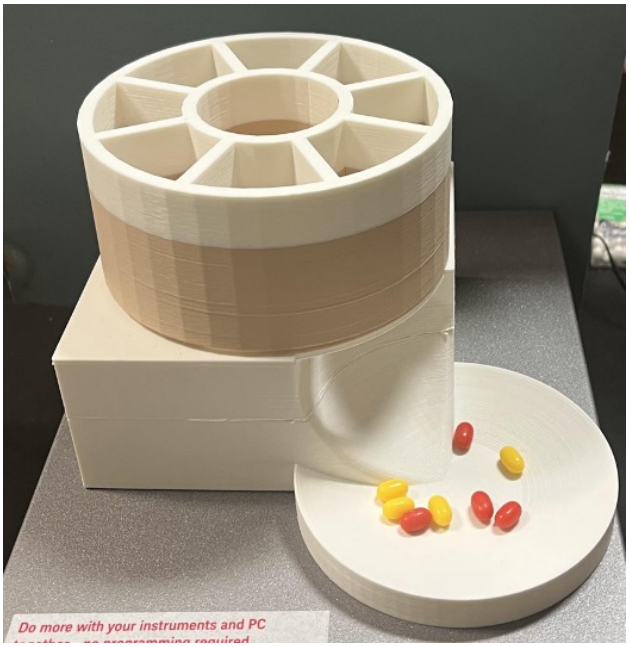

Figure 6.5: Smart Pill Box

6.3.1 Objective and Design Philosophy

The goal of this device is to develop an easy way to dispense medication to patients that might struggle with memory or motor function. The system is designed to be easy to use and access for as many client profiles as possible.

6.3.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- Raspberry Pi Pico W: Communicates with motor driver to spin spindle
- 28BYJ-48 Stepper Motor: Turns the Pill Box Spindle a set number of degrees
- ULN2003 Motor Driver: Acts as a bridge between the Raspberry Pi Pico W and the Stepper Motor
- AccelStepper Library for Arduino[35]: Helps code the Pico to communicate with the motor driver

- Wi-Fi & MQTT: The Pico W uses Wi-Fi to connect to the Smart Hub (Home Assistant) and publishes messages via MQTT.
- Home Assistant: Open-source smart home hub that receives MQTT messages.

6.3.3 Implementation Strategy

The main method of dispensing pills is the premade stepper motor and motor driver kit. The driver is able to be directly hooked into the Raspberry Pi Pico W GPIO pins. The Raspberry Pi Pico W GPIO pins are then controlled using code based on the PicoW_HomeAssistant_Starter project.

6.3.4 Engineering Challenges and Solutions

- The pills would initially fall into the tray with too much velocity and go flying out the side. We had to add a barrier to the tray.

6.3.5 Performance

The device is able to dispense pills when remotely prompted. It can either be commanded to immediately dispense from the Smart Mobile Device, Kick Buttons, and Remote Support Station. It can also be scheduled to dispense pills at a given time from the Remote Support Station. When dispensed, pills land in the tray to be consumed. The spindle is able to be easily removed for cleaning of the device.

6.4 Smart TV

6.4.1 Objective and Design Philosophy

The goal of the Smart TV is to allow control of a TV and a Roku home entertainment system through the Home Assistant and thus through the Smart Mobile Device. This allows the resident to have control over their day-to-day entertainment.

6.4.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- Pi Pico W (bridge): Connects to both the Hub and the Roku to enable the Roku integration on Home Assistant to function.

- Roku Express: A streaming media player that offers many free channels and handles streaming services. It can be controlled through HTTP commands from a device on the same network.

6.4.3 Implementation Strategy

Set up the Roku Express following the instructions that come with it. Follow the steps in "Getting started with your Raspberry Pi Pico W" to install MicroPython onto the Pi Pico W and set up the Thonny Python IDE. Open main.py in Thonny and save the file to your computer. Edit the section labeled "Internet Credentials" to the Wi-Fi credentials and Roku IP address of your own system. Save this modified program to your pico by selecting File, Save As, Raspberry Pi Pico. Before you click OK, ensure that the file name is "main.py", which may require overwriting an existing main.py file. This will allow the program to start up as soon as the Pico receives power.

This implementation uses a Raspberry Pi Pico W to bridge communication between Home Assistant and a Roku Express located on separate networks. Home Assistant already has a Roku integration that communicates to Roku devices using HTTP [36]. For security reasons, the Hub and thus Home Assistant are located on a local private network, the Roku Express is not able to use HTTP to communicate with devices on separate networks.

The Pico connects to Wi-Fi [37] and runs a local server on port 8060 [38], mimicking the Roku Express's API endpoint so Home Assistant can send commands to it. Upon receiving a request, the Pico rewrites the destination IP to the Roku's IP address and forwards the data to the actual Roku, then relays the response back to Home Assistant. The data forwarding is done in cycles and garbage collection [39] is initiated after a complete request is forwarded to avoid memory errors. This allows Home Assistant to interact with the Roku as if it were on the same network, bypassing network segmentation restrictions. The setup includes connection diagnostics, error handling, and non-blocking sockets for efficient data forwarding.

6.4.4 Engineering Challenges and Solutions

- Before we figured out how to get the system to automatically disconnect, we would have to wait a while after each activation or get an address in use error.

- We had issues with sending longer requests, and discovered that it was either sending incomplete requests or ran out of memory. We fixed this by cycling through the request and forwarding it in chunks.

- We had memory issues after running the code for a period of time. This was resolved by adding garbage collection after sockets closed.

6.4.5 Performance

The Pico proxy works to bridge communication between the Home Assistant and the Roku. There are slight delays, although it is unclear how much of it is due to the Pico and how much is built into the pre-existing system. Human participants were able to use the Home Assistant dashboard, the SMD UI, and the SALE-R to control the TV and watch various media.

6.5 Smart Door Opener

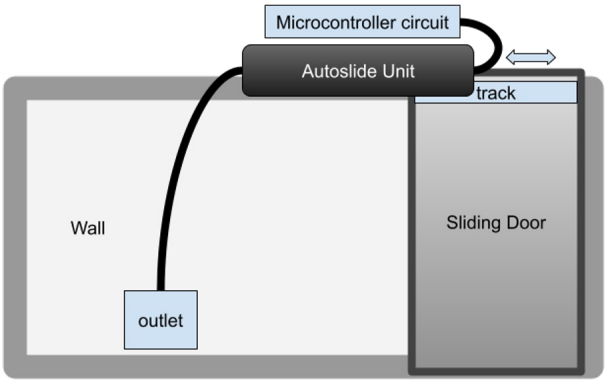

Figure 6.6: Smart Door Opener

6.5.1 Objective and Design Philosophy

The Smart Door Opener is designed to allow the resident more control over their door. When the resident or support staff chooses to open or close the door, the Smart Hub takes that command and sends it to a microcontroller to open/close the door. The power supply for the door opener comes from a wired connection to a power outlet, and the door opener powers the microcontroller that connects to the Hub.

6.5.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- Autoslide Sliding Door Opener: Opens and closes the door in response to a relay closing. Includes collision detection.
- Pi Pico W: Connects to the Hub to enable wireless control via the MQTT protocol. Also includes code to integrate with the Home Assistant.
- G5LE-1 DC12 12V Relay: Allows the Pi Pico W to control the Door opener
- TLV2217-33KCSE3 Liner Regulator: Converts wall power to a lower voltage for the Pi Pico W

6.5.3 Implementation Strategy

The Autoslide commercial sliding door opener is the actuation mechanism due to its built-in safety features, such as collision detection. The Autoslide system includes relay cable ports that allow for external input control, enabling integration with third-party hardware. Smart control is achieved using a modification of the PicoW_HomeAssistant_Starter project to control a GPIO pin. The GPIO pin connects to a simple circuit that toggles a relay, which in turn activates the door opener via these ports.

6.5.4 Engineering Challenges and Solutions

- We initially planned to connect the GPIO pins to the fob remote that came with the Autoslide system. The chip inside the fob was not meant to be soldered and easily broke the connections. In addition, it did not reliably trigger the inside sensor on the Autoslide Computer Unit, so we decided to go with a relay instead.
- Mechanical/installation issues with smooth sliding of the door. This was resolved by re-installing the door.

6.5.5 Performance

The device operates with the HA website, through the UI of the SMD, through the SALE-R buttons, and through manual control. Latency is negligible.

6.6 Smart Security Door Camera

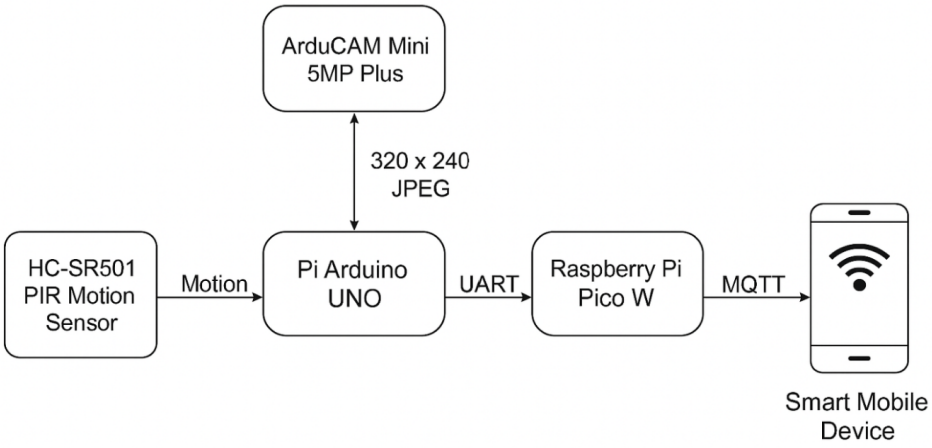

Figure 6.7: Smart Security Door Camera System Level Diagram

6.6.1 Objective and Design Philosophy

The goal of this device is to develop a smart security system capable of detecting motion at an entry point, capturing and processing image data, and securely communicating visual alerts via wireless protocols. The system is designed to function as a reliable and low-power video monitoring device for the smart home system. Emphasis is placed on reliable wireless communication and latency optimization for real-time streaming.

6.6.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- ArduCAM Mini 5MP Plus (V5642): Used to capture 320 x 240 JPEG images at 10 frames per second and compress them on board to reduce transmission burden
- Pi Arduino UNO: Controls the ArduCAM and transmits JPEG image data line-by-line over UART
- Raspberry Pi Pico W: Reconstructs the JPEG and conditionally transmits the image via MQTT
- HC-SR501 PIR Motion Sensor: Connected to the Arduino to detect motion events. It triggers a notification on the Smart Mobile Device to prompt the user to open the live feed.

6.6.3 Implementation Strategy

The system architecture separates camera capture and wireless communication into two distinct microcontroller units: the Arduino Uno, which manages image acquisition and motion sensing, and the Raspberry Pi Pico W, which handles image reconstruction and MQTT publishing over Wi-Fi. This modular approach allows each subsystem to operate within its strengths. The Arduino offloads all time-critical sensor and camera communication, while the Pico handles all network-related tasks and acts as the gateway to the smart hub.

The ArduCAM Mini 5MP Plus (OV5642) is interfaced with the Arduino via SPI, using dedicated chip select, MISO, MOSI, and SCK lines. Upon motion detection using the HC-SR501 PIR sensor, the Arduino initiates continuous image capture in JPEG format at a resolution of 320×240. There is no fixed frame count limit; the Arduino continues to stream frames at a rate of approximately 10 frames per second for the duration of the motion event. Each frame is buffered internally by the ArduCAM and then read byte-by-byte over SPI. The Arduino transmits the JPEG data to the Pico via UART, enclosing each image between custom markers: IMG_START:¡frame_number¿ and END. These markers allow the Pico to detect and parse each complete image reliably, even if some markers are split across multiple reads.

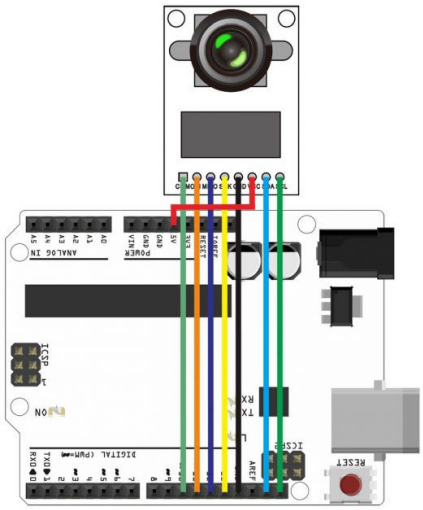

Figure 6.8: Arducam-Arduino System Level Diagram

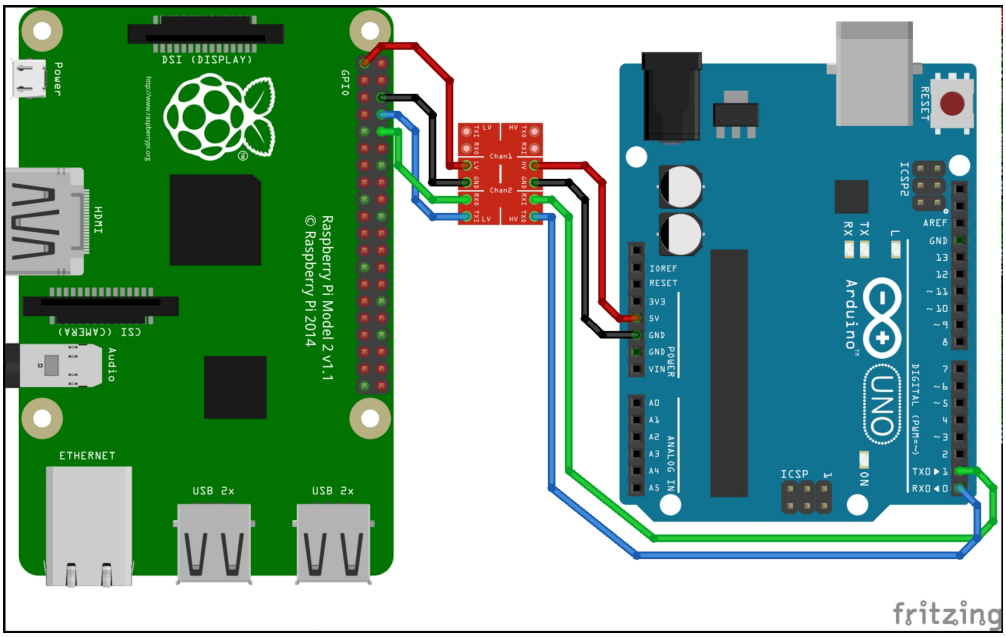

Figure 6.9: Arduino-Pico UART System Level Diagram

To ensure electrical compatibility between the two boards, a 2kΩ–1kΩ voltage divider is implemented on the Arduino TX line to safely shift its 5V UART signal to the Pico's 3.3V logic level. The UART connection operates at a baud rate of 57600, which was experimentally determined to be the highest stable rate for full JPEG frame transfers without packet loss or corruption.

On the receiving end, the Pico W continuously reads from its UART buffer, monitoring for the IMG_START marker. Once this is detected, the Pico collects image data into a byte array until it sees the END marker. Afterward, it verifies the data by locating the standard JPEG start (0xFFD8) and end (0xFFD9) markers, and strips any garbage bytes before or after the actual image. The cleaned JPEG is then base64-encoded and published to the MQTT topic camera/frames/latest, where it can be rendered by Home Assistant or any other subscriber.

In parallel, the Arduino also transmits motion state messages (MOTION:1 or MOTION:0) when the PIR sensor is triggered or reset. These messages are processed by the Pico and published to the MQTT topic camera/motion, which allows the Home Assistant hub to distinguish between idle and active states, log motion events, and trigger user notifications. Because image capture is tied directly to the PIR signal and operates in real-time during motion, the system provides a continuously updated visual feed to the user while minimizing unnecessary image transmission during periods of inactivity.

This implementation plan has been validated in testing and is robust under real-world

usage. With appropriate attention to wiring, baud rate configuration, and message formatting, the system can be replicated using the provided hardware components and firmware structure.

6.6.4 Engineering Challenges and Solutions

- The Arduino operates at a higher voltage than the Pico. To reduce the voltage to safe levels for the Pico, we implemented a voltage divider on the TX line.
- JPEG data transfer was originally too slow for real-time video (9600 baud rate). We increased the baud rate to 57200, which was the maximum value we achieved for successful image transfer. Additionally, the UART buffering was optimized to accommodate full-frame streaming.
- JPEGs were either unreadable or misaligned due to incomplete parsing. We added stream parsing logic to detect JPEG start (0xFFD8) and end (0xFFD9) markers and discard all pre/post garbage.
- The image start and end markers were occasionally split across multiple reads, so we introduced a rolling window buffer and character-by-character UART parsing to catch markers regardless of alignment.
- Because images failed to transfer after a soft reboot, we added UART buffer flush routines and introduced handshake logic between the Arduino and Pico to re-align the stream.

6.6.5 Performance

System performance was evaluated in terms of image capture latency, transmission delay, and data consistency. With the ArduCAM configured to capture images at a reduced resolution of 320 x 240 pixels, average JPEG file sizes ranged between 8-15 KB depending on scene complexity. Simpler, low-texture scenes yielded smaller files and faster transmission times, while high-contrast or detailed scenes increased JPEG size and introduced transmission lag.

The total end-to-end latency from image capture to image availability in Home Assistant was measured to be approximately 1 second under worst case conditions. The ArduCAM completes image capture in a consistent 100ms, while the remaining 900ms is split between UART transmission and Wi-Fi/MQTT publishing. Since image transfer over UART accounts for the bulk of this delay, the residual 200-400 ms was conservatively attributed to

JPEG parsing, base64 encoding, MQTT transmission, broker processing, and Home Assistant rendering. While precise timing of each software layer was not instrumented, these estimates align with expected performance from similar MQTT-Wi-Fi pipelines. Transmission speed was strongly influenced by image complexity, as more detailed images required more bytes over UART and Wi-Fi. This resulted in occasional latency variability of ±0.5 seconds.

While the system is not intended for high frame-rate streaming, it is well-optimized for periodic or event driven image transmission. Further optimizations such as having event-driven image capture with motion detection for a set number of frames could improve power performance. Additionally, the circuit could be implemented on a PCB for a better UART connection and a higher baud rate ceiling. For practical purposes, this latency is acceptable for a security camera that allows the user to monitor feed outside their room.

6.7 Smart Window Shades

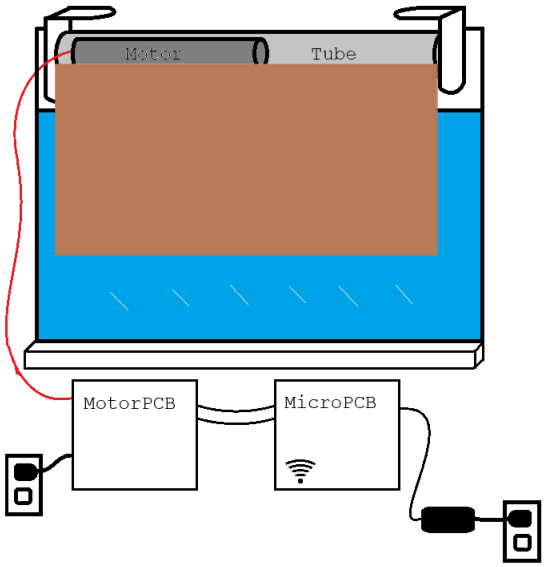

Figure 6.10: Smart Window Shades Sketch

6.7.1 Objective and Design Philosophy

The goal of this project is to develop an embedded, remotely controlled window shades using wireless protocols. The system is designed to serve as a reliable and straightforward add-on to commonly available window shade motors, integrating easily with the hardware of

our smart assistive living environment. It enables users to control window lighting without requiring significant physical effort.

6.7.2 System Architecture and Core Components

The system is built around the following hardware and software elements:

- 120V Electric Tubular Motor to control tubing and attached shades.
- Pi DSP2A-DC12V, DPST-NO (2 Form A) Relay to control power to the motor.
- G5LE-1 DC12, SPDT(1 Form C) Relay to control the direction of the motor.

6.7.3 Implementation Strategy

Follow the steps in Daniloc's Pico W HA Starter, utilizing PlatformIO and VS Code. Configure the code based on the available code from the Window Shade's Github code in the 'src' folder. Make sure to edit the "Credentials.h" file to your intended home Wi-Fi's credentials. In VS Code, build and upload the project built from the existing files on the GitHub onto a Pico W.

The system design is divided across two separate PCBs: one dedicated to power distribution and the other to the microcontroller. This separation provides electrical isolation between the DC power and AC power domains, enabling safe and effective motor control. The Raspberry Pi Pico W microcontroller communicates with the central hub using the MQTT protocol, facilitated by the ArduinoHA library.

6.7.4 Engineering Challenges and Solutions

- Since the window shades directly use wall power, power isolation was essential to prevent any components from malfunctioning or burning up. To solve this, the design was revised with optoisolators, fuses, and a circuit breaker to follow safety precautions.
- To prevent exposure to live wires, the PCBs were enclosed in 3d printed housings.

6.7.5 Performance

System performance was evaluated through a set of tests where the functionality was performed. The device operated with the HA website, through the UI of the SMD, through the SALE-R buttons, and through manual control. Latency was negligible, and the time

the shades took to close from being fully up was around 12.5 seconds. The length to which shades drop down can be adjusted as per the user's preference via the motor's adjustable key holes.

CHAPTER 7

Testing

7.1 Individual Device Testing

7.1.1 SALE-R

The SALE-R was first validated by testing the functionality of an optical switch circuit, after which the optical switch was mounted inside the button. In the second stage, the LED circuit was added, and it was observed whether the LEDs turned off when the buttons were pressed. The buttons were then placed on a platform in their intended layout, and the Raspberry Pi Pico was used to detect button presses from the set of five buttons. Functionality was verified by initially printing button press messages via a serial communication port to the console in the Thonny IDE. Finally, Bluetooth communication was tested by programming the smart mobile device UI to display button-press detection messages. The detailed test report can be found in Appendix B, section B.1.

7.1.2 Smart HUB

The Smart Hub was extensively tested as a natural result of integrating it with other devices, which was done from the beginning. Individual tests were done on it as part of the process of this integration. The detailed test report can be found in Appendix B, section B.2.

7.1.3 Smart Mobile Device

Testing for the SMD began by validating Bluetooth Low Energy (BLE) connectivity between the tablet and the SALE-R kick button interface. Once the BLE connection was successfully established using the correct service and characteristic UUIDs, directional inputs from the kick buttons were tested to ensure they were accurately detected by the app. The interface was then tested by navigating through all device menus and submenus using both touchscreen

and kick button inputs. Each action was checked to make sure it correctly updated the internal menu state and triggered the intended behavior. To verify communication with the rest of the system, we sent commands from the SMD and confirmed that the corresponding actions were executed through the Smart Hub and visible on the Home Assistant interface. All device control screens were tested for consistent functionality. The detailed test report can be found in Appendix B, section B.3.

7.1.4 Remote Support Station

The Remote Support Station's ability to see the correct logs is validated through viewing of the RSS after devices have been used. The call functionality to and from the SMD has not been validated yet. The detailed test report can be found in Appendix B, section B.4.

7.1.5 Home Assistant Website

The Home Assistant Website is able to see the logs and control the devices that it should have access to. It does not require further testing as it is built into the Home Assistant system and was used in tests for all other devices.

7.1.6 Smart Pill Box

We first verified the dispensing functionality of the Smart Pill Box by powering the device, loading it with pill substitutes, and setting it to dispense in five minutes. We confirmed that the pills were dispensed on time and correctly landed in the tray. Next, we tested the emptying function by triggering the pill box seven times through the Pi Pico W and verified that notifications were sent for each dispensing and once when the device was empty. We then connected the Smart Pill Box and Remote Support Station to the Smart Hub, navigated the menu to schedule a one-minute dispense, and confirmed that a notification was received and medical information was accessible through the Remote Support Station. Finally, we had a third-party tester attempt to open the pill box, remove the spindle, and extract pill substitutes, confirming that all actions could be completed without additional instruction. The detailed test report can be found in Appendix B, section B.5.

7.1.7 Smart Light

We first conducted a run test on the Smart Light by powering the device, placing a piece of paper on the bulb, and operating it continuously for 30 minutes, confirming that the paper did not burn and the light remained stable. Next, we tested wireless control by

repeatedly turning the Smart Light on and off five times using the Kick Buttons, seeing that the light was responsive each time. We then tested bulb replaceability by powering the light, confirming that illuminated, unscrewing the original bulb, installing a new one, and checking that the new bulb also lit up properly. Finally, we looked closely at the Smart Light's physical appearance to confirm that no wires or electronic components were visible. The detailed test report can be found in Appendix B, section B.6.

7.1.8 Smart Door Opener

We first verified that the inside sensor trigger properly opened the door by pressing the inside sensor button on the Autoslide Sliding Door Opener. We then plugged in the relay cable and connected the ends of the yellow and black wires to validate that when connected, the door would open. We separately tested that the microcontroller could control the relay by controlling it via the Home Assistant website and checking the voltage between the pins that would be connected to the yellow and black wires. Finally, we connected the system together and verified the functionality by clicking the "Open Door" button on the Home Assistant website. The detailed test report can be found in Appendix B, section B.7.

7.1.9 Smart Security Door Camera

We verified the functionality of our Smart Security Door Camera through a series of targeted tests. First, we confirmed that the PIR sensor correctly detected motion by observing the motion state update in the terminal and MQTT logs. Then, we verified that an image was captured upon motion and transmitted via MQTT by checking that a valid base64 JPEG appeared in the camera/frames/latest topic. In the next test, we validated the integrity of the UART image transfer by confirming that each JPEG included proper start and end markers and displayed correctly. Afterwards, we confirmed that the Pico W reconnected to the network and resumed publishing motion and image data after reboot, and timed the delay between motion detection and image appearance in Home Assistant. We found this to remain consistently under 1 second. Finally, we validated that the system could handle multiple image captures in quick succession without data loss or misordering. All tests were confirmed visually through terminal output, MQTT logs, and correct image rendering in Home Assistant. The detailed test report can be found in Appendix B, section B.8.

7.1.10 Smart TV

The http functionality of the Smart TV was first validated by sending hard-coded http commands to the Roku device in a loop, and seeing that the commands were implemented on the display. Then, we used the Hub command line to control the TV display. This validated the ability of the Hub to control the Smart TV despite being on separate networks. Finally, overall functionality was tested by navigating to various menus and controlling aspects (like volume, speed, time, etc.) of various media.

7.1.11 Smart Window Shades

The functionality of the window shades device was validated first by ensuring communication with the Hub, and then testing the actual performance of the motor through a set of tests. The detailed test report can be found in Appendix B.

7.2 System Integration Testing

The system was integrated by performing a series of steps involving network connectivity, user interface operations, and peripheral control mechanisms.

7.2.1 Connecting to HUB

The HUB generates a private WiFi network. Each individual device's microcontroller (Pi Pico W) connects to the HUB network by adding credentials to the "Credentials.h" file in the PicoW_HomeAssistant_Starter repo[40], which includes the HUB IP, network password, MQTT password, and other details. The Arduino Home Assistant starter pack helps declare the number of buttons or switches that each device contains. A successful device is tested with its appearance on the Home Assistant UI. On the HUB logs, confirmation of a proper connection is noted with the message "RSN Handshake Successful," indicating that the Pairwise Key Handshake has completed. Once connected, devices can automatically reconnect to the HUB network upon being powered.

7.2.2 Operation Through Home Assistant

Once devices were connected to the HUB, they were tested through the Home Assistant web interface. If the devices responded to input commands via the virtual buttons and switches displayed on the Home Assistant website, it confirmed their functional integration with the system.

7.2.3 Operation Through Smart Mobile Device

The Smart Mobile Device was then connected to the same private WiFi network of the HUB (test123), using a designated user ID and password. The mobile app used HTTP calls to control the individual buttons and switches assigned to each smart device. After verifying device functionality through the Home Assistant UI, the same operations were validated using the Smart Mobile Device interface to ensure consistent performance.

7.2.4 Operation Through SALE-R

Finally, the SALE-R was integrated with the Smart Mobile Device using Bluetooth Low Energy (BLE). BLE messages were used to control the Smart Tablet UI through serial-peripheral command transfers sent to the central device (Tablet). When the devices could be controlled properly through the SALE-R button inputs, it was verified that the full system integration was successful.

7.3 Usability Verification on Human Subjects

This study included a total of 8 participants. The participants were asked to fill up an informed consent form before taking the test. After that, the participants tried out a subset of smart home devices in order. The goal was to let participants navigate using directional controls to use a range of smart home devices with different numbers of operations.

7.3.1 Stage 1

To familiarize the participants with the SALE User Interface, they were asked to operate the smart mobile device using the UI. The participants tried out the following smart home devices, in increasing order of the number of functions each of them can perform, chronologically:

- Smart Door Unit: For this device, the participants were asked to slide the door using the UI.
- Smart Pill Box: For this device, participants were asked to dispense a slot of pills using the dispense buttons using the UI. This was followed by the participants dispensing another two slots of pills, one after another.
- Smart Light: The participants turned the light on and then turned it off.

- Smart Blinds: The experiments started with the blinds being completely up. The participants were asked to put the blinds down completely. Then they were asked to put the blinds completely up. At last, they were asked to stop the blinds midway, using the user interface.

- Smart Speaker: The participants were asked to select one song from a list of 4 pre-selected songs that appeared on the UI. They were then asked to pause and unpause the song. This was followed by the participants selecting another song from the list of smart devices. At the end, participants were asked to activate the voice recognition feature to give voice commands to the smart speaker.

- Smart TV: The participants were given a formal explanation about where the different submenus corresponding to the various buttons on the Roku are located. An explanation about how the Roku interface works was also given. After that, the participants were asked to navigate to Disney+ and select any movie they liked on the front screen. This was followed by participants performing actions like rewinding/fast forwarding the movie, increasing or decreasing the volume, and pausing and unpausing the movie unit.

- RSS Request Help: At the end, participants were asked to press the request help button on the main menu of the smart home user interface. This button sent a notification to the caregiver monitoring the remote support station, asking for assistance.

7.3.2 Stage 2

This was followed by the participants using the SALE-R and hand motions to control the user interface. However, before the users started using the SALE-R to control the smart home appliances, they were given a brief demo on how to interact with the user interface using the SALE-R. The participants were not allowed to touch the user interface with their fingers. This helped simulate the category of target second category of users mentioned in section 2.2. The participants were asked to operate the smart devices in the order mentioned in Stage 1.

7.3.3 Stage 3

In the final stage of testing, participants were asked not to use their hands at all. The SALE-R was placed on the ground, closer to the feet of the participants. The participants were asked to operate the smart devices in order, as mentioned in stage 1, without using their

hands or touching the touchscreen interface. This allowed us to simulate the third category of users mentioned in section 2.2.

7.3.4 Stage 4

In this stage, the participants were asked to check the status of if smart devices were connected to the HUB, and were asked to operate the devices through the remote support station. They also inspected the feedback mechanism when the patient requested help through the remote support station, requesting assistance or intervention.

7.4 Questionnaires

This was followed by participants completing multiple questionnaires about smart devices, in which they rated the "ease of use" for the smart home appliances using the UI through a touch screen interface, and through the SALE-R, using their hands and feet. This was followed by a question about each device's efficiency or reliability in response to their action, through all three methods identified in stages 1, 2, and 3. The final questionnaire was about the whole smart home system, in which participants were asked to rate their experience using SALE-R, voice control, home assistant interface, remote support station interface, and if the feedback mechanism allowed the caregiver to collaborate with a patient-centric feedback system in case of errors or emergencies.

7.5 Participant Responses

Participants noted that devices with fewer operations (up to seven) were generally easy to control using the user interface (UI). However, interfaces like Roku posed challenges, as they required users to navigate through multiple smaller steps. While most users found simpler devices easy to use, many described the smart TV interface as moderately to highly difficult due to its complexity. A common point of feedback was that experience and training significantly improve usability for complex devices like smart TVs. Some participants also mentioned unfamiliarity with the Roku interface or remote, which added to their learning curve.

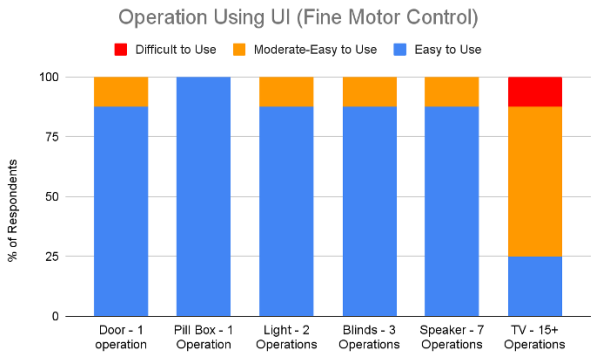

Figure 7.1: Participant response to smart home operation using the touch screen device

Regarding the directional control device (kick buttons), participants generally found them easy to use for devices with fewer operations. However, confusion arose when navigating interfaces with many sub-menus. Users had difficulty understanding how the directional buttons on the remote corresponded to the controls on the smart mobile device UI. Some participants preferred a linear list of options for smart TV navigation, while others were comfortable with sub-menus, as long as clear instructions and training were provided. This was seen as essential to prevent technology fatigue among users.

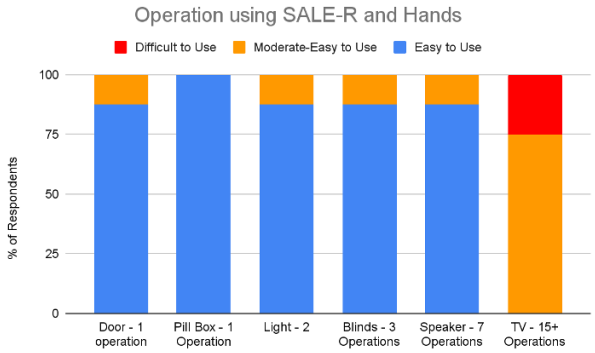

Figure 7.2: Participant response to smart home operation using SALE-R and hands

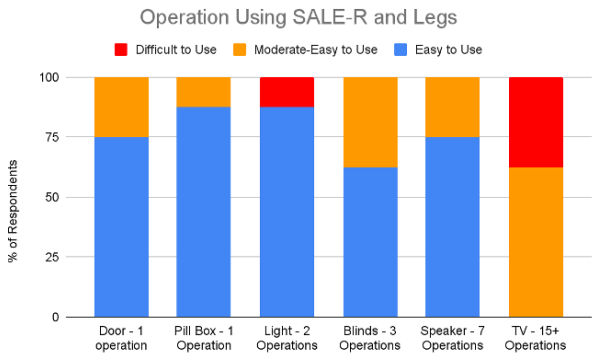

Figure 7.3: Participant response to smart home operation using SALE-R and legs

In terms of the reporting efficiency and reliability of the three methods and the functionality of the smart devices, the majority of participants said that these methods are highly efficient. The participants reported positively about the devices reported properly, however, main concerns included the complexities of navigating multiple buttons and sub-menus, alongside navigating challenges of internet latency in case of devices like smart TV.

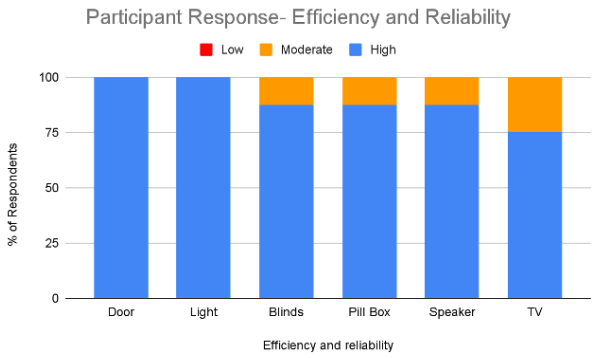

Figure 7.4: Participant response to the efficiency and accuracy of smart devices in relation to three modes of operation

CHAPTER 8

Conclusion

The final product of the senior design project is the Smart Assistive Living Environment (SALE) system. This system is designed to support residents in a smart home by helping them manage daily tasks, maintain connections with friends and caregivers, and live more independently. The project achieved positive results, with all proposed devices completed on schedule. Throughout the development process, we gained valuable insights into the needs of individuals with disabilities, deepening our understanding of how to design more effective assistive technologies. In user testing, most devices controlled through the SMD were rated as "easy to use," with only occasional ratings of "moderately easy to use." The only exception was the Smart TV, which required more complex navigation due to its large number of functions. To accommodate this, the Smart TV interface was divided into submenus. While functional, these submenus were less intuitive, especially for first-time users (which is why we recommend training). For example, one submenu mapped kick button inputs directly to Roku-style directional arrows, but the TV's select button was located in a different menu. This meant that tasks like entering a search term on a streaming service required switching menus to confirm each letter. While not ideal, the Smart TV was still rated as "moderately easy to use" overall, despite the numerous different functionalities.

While the results were encouraging, there remains significant room for improvement, an expected aspect of any engineering project. Each device presents opportunities for refinement and enhancement. It's also important to recognize that the scope of this project was relatively narrow, especially when compared to the wide range of assistive technologies currently being explored in smart home environments.

The project, while mostly complete, remains conceptually oriented and lacks polish in its current state. Several of the device's initial requirements still require refinement to reach full functionality. In addition to finalizing the existing components, future work should explore the integration of alternative interface methods to enhance usability and accessibility. Promising approaches include Brain-Computer Interfaces (BCIs), eye tracking, Virtual or

Augmented Reality (VR/AR), and wearable or ambient sensors. The BCI could be implemented to use on lights, door locks, or window blinds according to a study [41]. Eye-tracking could be implemented to allow users to control lamps, TVs, and fans by gazing at specific points or locations on a screen [42]. VR and AR technologies are gaining popularity and have shown potential in empowering older adults to interact with smart home systems, thereby promoting independence, safety, and overall quality of life [43]. Although somewhat intrusive, wearable technology can help monitor the well-being of the user and assist with navigating interfaces [44].

APPENDIX A

Code, Mechanical, and Schematic Designs

For the full software and hardware design implementation of the project, please visit the GitHub repository: https://github.com/ECE-492-SALE/SALE_ECE491-02

APPENDIX B

Test Report

B.1 SALE-R

Mechanical Design Requirements

- **Test T1: Interaction with Optical Switch Surface**
  Method: Visually inspect and measure the clearance between the metal strap and the surface of the optical switch in all button positions using a caliper or other precision measuring tool.
  Acceptance Criteria: There should be a minimum clearance of 1 mm between the metal strap and the optical switch surface in both pressed states.

- **Test T2: Laser Interruption upon Button Press**
  Method: Press the button and observe the optical switch output using a voltmeter to confirm that the laser is interrupted by the metal strap.
  Acceptance Criteria: The laser should be fully interrupted once the button is pressed.

- **Test T3: Laser Continuity in Equilibrium Position**
  Method: Place the button in its equilibrium position and monitor the optical switch output using a voltmeter.
  Acceptance Criteria: The laser should remain uninterrupted in the unpressed state.

- **Test T4: Button Return to Equilibrium Position**
  Method: Manually press the button multiple times and observe whether it reliably returns to its original position. Use a caliper if needed to confirm alignment.
  Acceptance Criteria: The button must return to its equilibrium position after every press.

- **Test T5: LED Circuitry Accommodation**
  Method: Insert the LED driving circuitry into its designated compartment and check

for secure fit and unobstructed placement.
Acceptance Criteria: The circuitry must fit securely within the button housing with no movement or interference.

Electrical Functionality

- **Test T6: Optical Switch Voltage Regulation**
  Method: Use a multimeter or oscilloscope to measure voltage supplied by the LM7805 regulator to the optical switch in both idle and active states, under various load conditions.
  Acceptance Criteria: Voltage must not exceed 5.5 V.

- **Test T7: LED Driver Circuit Current Limitation**
  Method: Power the LED circuit and measure current across the 300-ohm source resistor using a multimeter. Test under standard and maximum base current conditions.
  Acceptance Criteria: Current must remain less than or equal to 40 mA in all tested conditions.

- **Test T8: GPIO Input Voltage Limitation to Microcontroller**
  Method: Measure voltage at the GPIO input pin during idle and active states using a multimeter or oscilloscope.
  Acceptance Criteria: Voltage must not exceed 3.8 V.

Individual Button Functionality

- **Test T9: Microcontroller Readings**
  Method: Monitor GPIO pin on the microcontroller while pressing the button. The onboard LED should be programmed to turn on when a high signal is received.
  Acceptance Criteria: The microcontroller must detect a logic high when the button is pressed.

- **Test T10: LED Illumination on Button Press**
  Method: Press the button and visually verify that the button's LED lights up and remains on while pressed.
  Acceptance Criteria: LEDs must illuminate immediately when the button is pressed.

- **Test T11: Bluetooth Message Transmission on Button Press**
  Method: Pair the microcontroller with a tablet or computer. Monitor the receiving application or terminal to confirm transmission of "ON_buttonNumber" and default

"OFF_buttonNumber" messages.
Acceptance Criteria: An "ON_buttonNumber" message must be sent when the button is pressed. "OFF_buttonNumber" must be sent when no button is pressed.

System Level Integration (5 Buttons)

Setup: Power the button system (5 buttons and the circuitry) using a 12 V supply. Place the 5 buttons on a custom mount/footrest that is comfortable to use when a person is in a wheelchair.

- **Test T12: Button State Detection and String Composition**
  Test Method: Use a USB connection to print serial output from the microcontroller while pressing each button individually and in combinations; confirm that the generated string accurately reflects the state of each button.
  Acceptance Criteria: The generated string accurately reflects the current state of each button in the format "ON_1 OFF_2..." etc.

- **Test T13: Bluetooth Transmission of Button States**
  Test Method: Pair a tablet or computer with the microcontroller via Bluetooth, then press buttons individually and in combinations while monitoring the received string in a logging application; verify that each transmission matches the current button states and test for consistency under rapid presses.
  Acceptance Criteria: The tablet consistently receives the correct button state string for every combination of presses, in the specified format (e.g., Left pressed, Right pressed, etc.).

- **Test T14: Stability and Responsiveness Testing**
  Test Method: Use logging software to monitor transmitted strings while rapidly pressing buttons in different sequences, including simultaneous inputs; observe whether the system consistently updates the string without missing or misreporting button states.
  Acceptance Criteria: The system should not miss any button press events or provide incorrect button states.

Test Procedure	**Acceptable Range**	**Actual Measurements**	**P/F**
T1: Interaction with Optical Switch Surface	Minimum 1 mm clearance in all button positions	2 mm clearance observed in compressed state	Pass
T2: Laser Interruption upon Button Press	Laser must be fully interrupted on press	Voltage dropped to 0V when button pressed	Pass
T3: Laser Continuity in Equilibrium Position	Laser should remain uninterrupted when unpressed	Voltage unchanged when button unpressed	Pass
T4: Button Return to Equilibrium Position	Must return to original position after every press	Button reliably returned to original position	Pass
T5: LED Circuitry Accommodation	Circuitry must fit securely with no interference	LED board fit securely; 2cm x 3cm dimensions confirmed	Pass
T6: Optical Switch Voltage Regulation	Voltage $\leq$ 5.5 V under all load conditions	Voltage measured at 5.2 V	Pass
T7: LED Driver Circuit Current Limitation	Current $\leq$ 40 mA under all conditions	Max current observed was 38 mA	Pass
T8: GPIO Input Voltage Limitation to Microcontroller	Voltage $\leq$ 3.8 V	Voltage measured at 3.29 V	Pass
T9: Microcontroller Readings	Detect logic high on press; onboard LED turns on	Logic high detected and LED turned on on press	Pass
T10: LED Illumination on Button Press	LED must light immediately upon press	LED illuminated immediately on press	Pass
T11: Bluetooth Message Transmission on Button Press	Transmit "Pressed" and "!Pressed" messages correctly	Proper messages transmitted and verified on mobile device	Pass
T12: Button State Detection and String Composition	String must reflect current button states correctly	Accurate string observed	Pass
T13: Bluetooth Transmission of Button States	Correct state string received for all combinations	Correct states received for all press combinations, even rapid	Pass
T14: Stability and Responsiveness Testing	No missed or incorrect button events; must respond within 0.25 s	System consistently accurate for press sequences within 0.25 s	Pass

Table B.1: Acceptance Test Procedure – Kick Buttons

B.2 Smart Hub

Design Requirement: Smart Device can connect to Hub WiFi, communicate with the hub, and cannot be reached from outside the network.

- **Test T1: WiFi Connection**
  Equipment: Pi Pico W, Hub, WiFi
  Method: Attempt to connect a smart mobile device to the Hub WiFi
  Pass/Fail Criteria: Device connects to Hub WiFi, as confirmed from the logs.

- **Test T2: WiFi Data Transmission**
  Equipment: Pi Pico W, Hub, WiFi, `WebServer.py`
  Method: Load a basic web server onto the Pico W, and attempt to read it from the Hub
  Pass/Fail Criteria: Device claims an IP address and serves the website when requested from that IP.

- **Test T3: WiFi 2-Way Communication**
  Equipment: Pi Pico W, Hub, WiFi, `WebServer.py`
  Method: Load a dynamic web server onto the Pico W, and read and modify it from the Hub
  Pass/Fail Criteria: The device serves a website allowing dynamic control of the board LED.

Design Requirement: Smart Device integrates with MQTT and Home Assistant.

- **Test T4: MQTT Posting Data**
  Equipment: Mosquitto MQTT Server
  Method: Connect the appropriately configured Smart Device to the network
  Pass/Fail Criteria: The device posts its configuration data and status to a MQTT topic.

- **Test T5: Home Assistant Reading MQTT Data**
  Equipment: Mosquitto MQTT Server, Home Assistant
  Method: Clear retained MQTT messages. Connect Smart Device to the network.
  Pass/Fail Criteria: The device appears in Home Assistant.

- **Test T6: Home Assistant Sending Commands**
  Equipment: Mosquitto MQTT Server, Home Assistant

Method: Connect Smart Device to the network. Attempt to send a command (e.g., toggle switch).
Pass/Fail Criteria: Command MQTT topic receives command; device updates state and performs action.

Design Requirement: Smart Hub can obey voice commands and provide responses.

- **Test T7: Microphone Voice Command Test**
  Equipment: Smart Hub, USB Microphone
  Method: Connect microphone, enable addon and debug logs. Say wake word and a command.
  Pass/Fail Criteria: Logs record wake word, correct transcription, and valid action/response.

- **Test T8: Speech to Text Test**
  Equipment: Smart Hub, Media player
  Method: Play test phrase using local speech-to-text engine.
  Pass/Fail Criteria: Player outputs phrase. Minor mispronunciations acceptable.

- **Test T9: Smart Speaker Audio Feedback**
  Equipment: Smart Hub, Smart Speaker
  Method: Stream audio, enable voice recording, and monitor CPU usage.
  Pass/Fail Criteria: CPU load increases during streaming; debug file contains speech.

Design Requirement: Smart Hub can carry out automated actions on devices.

- **Test T10: Home Assistant Action Menu**
  Equipment: Home Assistant, device with action
  Method: Use Developer Tools to trigger an action.
  Pass/Fail Criteria: Device performs action and updates state; Home Assistant reflects change.

- **Test T11: Home Assistant Automation**
  Equipment: Home Assistant, device with action
  Method: Set up automation to perform an action at a set time.
  Pass/Fail Criteria: Action is executed at specified time.

Design Requirement: Smart Hub provides data logging for 30 days.

- **Test T12: History Visibility**
  Equipment: Home Assistant, various devices
  Method: Use History menu to review past data.
  Pass/Fail Criteria: Data from up to 30 days ago is available; older data is not.

Design Requirement: Smart Hub is secure.

- **Test T13: WiFi No-Forwarding**
  Equipment: Pi Pico W, Hub, WiFi, `WebServer.py`
  Method: Connect Pico to hub. Confirm web server via Test T2. Try external access.
  Pass/Fail Criteria: Pico is not accessible from external device.

- **Test T14: Red-Team Testing**
  Equipment: Smart Hub, hacker friend
  Method: Share Hub IP with external, savvy user. Attempt penetration test.
  Pass/Fail Criteria: Test fails if friend successfully breaches system.

Test Procedure	**Acceptable Range**	**Actual Measurements**	**P/F**
T1: WiFi Connection	Device connects to Hub WiFi as confirmed by logs	Pass	Pass
T2: WiFi Data Transmission	Device claims IP address and serves web page on request	Pass	Pass
T3: WiFi 2-Way Communication	Web interface allows dynamic control (e.g., LED)	Pass	Pass
T4: MQTT Posting Data	Device posts configuration/status to MQTT topic	Pass	Pass
T5: Home Assistant Reading MQTT Data	Device appears in Home Assistant after connecting	Pass	Pass
T6: Home Assistant Sending Commands	Device receives command and updates state via MQTT	Pass	Pass
T7: Microphone Voice Command Test	Wake word, transcription, and valid response appear in logs	Pass	Pass

Test Procedure	Acceptable Range	Actual Measurements	P/F
T8: Speech to Text Test	Playback yields intelligible phrase; minor errors acceptable	Pass	Pass
T9: Smart Speaker Audio Feedback	CPU usage rises; debug file includes speech	Pass	Pass
T10: Home Assistant Action Menu	Device performs action and reflects updated state	Pass	Pass
T11: Home Assistant Automation	Action executes automatically at set time	Pass	Pass
T12: History Visibility	Data from up to 30 days ago viewable in UI	Pass	Pass
T13: WiFi No-Forwarding	Pico is inaccessible from external devices	Pass	Pass
T14: Red-Team Testing	External attacker fails to breach Smart Hub	External hacker was able to breach hub only when we stored our passwords incorrectly	Pass

Table B.2: Acceptance Test Procedure – Smart Hub Integration

B.3 Smart Mobile Device

Design Requirement: Smart Mobile Device can respond to kick button inputs.

- **Test T1: Command Handling from Kick Buttons**
  Equipment: Smart Mobile Device, Kick Button Module
  Method: Send various commands to the SMD using kick button inputs and observe function execution.
  Pass/Fail Criteria: The SMD successfully recognizes and executes each command sent via the kick buttons.

Design Requirement: Smart Mobile Device has a user-friendly graphical interface.

- **Test T2: User Interface Usability Assessment**
  Equipment: Smart Mobile Device

Method: Conduct a user experience review focused on navigation flow, screen clarity, and interface accessibility.
Pass/Fail Criteria: The interface is rated as intuitive and easy to navigate based on user feedback.

Design Requirement: Smart Mobile Device can run custom applications.

- **Test T3: Custom App Installation and Operation**
  Equipment: Smart Mobile Device, Custom App Installer
  Method: Install the custom SMD application and test its functionality by navigating menus and triggering commands.
  Pass/Fail Criteria: The app installs successfully and performs all intended functions without error.

Design Requirement: Smart Mobile Device operates on battery power.

- **Test T4: Battery-Powered Operation**
  Equipment: Smart Mobile Device, Power Meter (optional)
  Method: Fully charge the SMD, then disconnect it from external power and observe system operation during normal use.
  Pass/Fail Criteria: The SMD functions normally while running solely on battery power.

Design Requirement: Smart Mobile Device is easy to use for non-technical users.

- **Test T5: General Ease of Use Feedback**
  Equipment: Smart Mobile Device, Survey Form
  Method: Have test users interact with the SMD and complete common tasks, then collect feedback through a brief survey.
  Pass/Fail Criteria: A majority of test users (e.g., ¿80%) report that the SMD is easy to use.

Design Requirement: Smart Mobile Device camera captures still and video images.

- **Test T6: Camera Functionality**
  Equipment: Smart Mobile Device
  Method: Open the SMD camera and attempt to capture both still images and live video; verify output through the display.
  Pass/Fail Criteria: The camera produces clear image and video output and functions without lag or error.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Command Handling from Kick Buttons	Commands are correctly executed through kick button inputs	All inputs triggered correct actions; occasional double press needed	Pass
T2: User Interface Usability Assessment	Interface is intuitive and easy to navigate	Interface rated easy to use for simple tasks; more complex menus (e.g., TV) posed some difficulty	Pass
T3: Custom App Installation and Operation	App installs successfully and performs all intended functions	Custom app installed and functioned as intended	Pass
T4: Battery-Powered Operation	Device functions normally while running solely on battery	Operated as expected when unplugged	Pass
T5: General Ease of Use Feedback	Majority of users report the device is easy to use	Test users provided positive feedback; TV menus resulted in more difficulty; See user feedback in the Testing and Validation chapter	Pass
T6: Camera Functionality	Camera captures clear still images and video without lag	Images captured clearly; video feed updates slowly (every 3–7 seconds) but functioned	Pass

Table B.3: Acceptance Test Procedure – Smart Mobile Device

B.4 Remote Support Station

Design Requirement: Remote Support Station access must require authentication.

- **Test T1: Authentication Requirement**
  Equipment: Computer with internet access, RSS website
  Method: Try to log onto the website without having set up user authentication on that

computer.
Pass/Fail Criteria: The test passes if the user must provide user authentication to access the support options.

Design Requirement: Remote Support Station can control the Smart Door Opener.

- **Test T2: Door Access**
  Equipment: RSS interface, Smart Door Opener
  Method: While logged in as the RSS user, click the "Open Door" button on the RSS dashboard.
  Pass/Fail Criteria: This test passes if the Smart Door Opener opens the door within 1 second of the button press.

Design Requirement: Remote Support Station can schedule and play audio reminders.

- **Test T3: Medication Reminders**
  Equipment: RSS interface, Smart Speaker
  Method: While logged in as the RSS user, use the dashboard to set the Smart Pillbox dispense time and turn the reminder on.
  Pass/Fail Criteria: This test passes if the speaker plays the expected audio reminder at the expected time, then dispenses pills.

Design Requirement: Remote Support Station can dispense pills through Smart Pillbox.

- **Test T4: Dispense Pills**
  Equipment: RSS interface, Smart Pillbox
  Method: While logged in as the RSS user, click the "Dispense Pills" button on the RSS dashboard.
  Pass/Fail Criteria: This test passes if the Smart Pillbox dispenses one dose of pills (1 segment of the pillbox) within 1 second of the button press.

Design Requirement: Remote Support Station can control and reflect Smart Light state.

- **Test T5: Smart Light**
  Equipment: RSS dashboard, Resident dashboard, Smart Light

Method: While logged in as the RSS user, click the "Smart Light" switch a couple times in rapid succession. Then, do not click it on the RSS dashboard and use the Resident user to control the Smart Light.
Pass/Fail Criteria: This test passes if the Smart Light correctly turns on/off in response to both dashboards, and the RSS dashboard displays the correct state of the Smart Light at all times.

Design Requirement: Remote Support Station can broadcast text-to-speech messages.

- **Test T6: Text to Speech**
  Equipment: RSS interface, Smart Speaker
  Method: While logged in as the RSS user, select the Smart Speaker and use the text-to-speech feature to play a test message.
  Pass/Fail Criteria: This test passes if the Smart Speaker plays the test message as a spoken message.

Design Requirement: Remote Support Station can call the Resident via Smart Mobile Device.

- **Test T7: Call the Resident**
  Equipment: RSS interface, Smart Mobile Device
  Method: While logged in as the RSS user, select the call button to initiate a call on the Smart Mobile Device (SMD). Do not pick up on the SMD. Initiate the call again, and this time pick up the call on the SMD.
  Pass/Fail Criteria: This test passes if the call shows up on the SMD but only goes through when the Resident gives permission.

Design Requirement: Remote Support Station can respond to help requests.

- **Test T8: Respond to Help**
  Equipment: RSS dashboard, Resident dashboard or SMD
  Method: While there is no pre-existing help request, have someone use the SMD or Resident dashboard to select the "Help Requested" button. Then, on the RSS dashboard, click the "Help Requested" button.
  Pass/Fail Criteria: This test passes if the resident help request caused the "Help Requested" button to light up on the RSS dashboard, and the second button press (from the RSS) was able to dismiss the help request.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Authentication Requirement	RSS requires user authentication before granting access to support functions	Authentication prompt appeared immediately; access granted upon valid login	Pass
T2: Door Access	Smart Door Opener opens within 1 second of "Open Door" button press	Delay of under 1 second	Pass
T3: Medication Reminders	Audio reminder plays and pills dispense at scheduled time	Audio reminder played at scheduled time; pills dispensed within 2 seconds	Pass
T4: Dispense Pills	Smart Pillbox dispenses one segment within 1 second of button press	Pill segment dispensed in 1 second of button press	Pass
T5: Smart Light	Light responds correctly to both dashboards; state always accurate on both	Both dashboards updated the light and saw the light state correctly	Pass
T6: Text to Speech	Smart Speaker plays text as speech upon command	Speech played clearly within 1 second of command	Pass
T7: Call the Resident	Call displays on SMD; only connects when permission is granted	Not Measured	
T8: Respond to Help	RSS dashboard lights up on request; clears request on second press	Button on the dashboard lit up instantly and a message played on the speakers; cleared after second press and played second message as expected	Pass

Table B.4: Acceptance Test Procedure – Remote Support Station

B.5 Smart Pill Box

- **Test T1: Dispensing**
  Method: Provide power to the Smart Pill Box. Load up the spindle with pill substitutes. Set the pill box to dispense in 5 minutes. Set a timer on a stopwatch for 5 minutes.

Pass/Fail Criteria: The pills are dispensed at the designated time and land in the tray.

- **Test T2: Emptying**
  Method: Provide power to the Smart Pill Box. Load up the spindle with pill substitutes. Send a command to trigger the pill box. Repeat six more times.
  Pass/Fail Criteria: The pill box sends a notification each time a pill is dispensed and one notification when the pillbox is empty.

- **Test T3: Information**
  Method: Provide power to the Smart Pill Box. Connect the Smart Pill Box to the Smart Hub. Provide power to the Remote Support Station. Connect the Remote Support Station to the Smart Hub. Navigate the menu of the Remote Support Station to the Smart Pill Box menu. Set a pill to be dispensed in one minute. Go to the main menu and wait for the notification. After it is received, navigate back to the Smart Pill Box menu and look for the medical information. Pass/Fail Criteria: The health information can be accessed from the Remote Support Station.

- **Test T4: Openable**
  Method: Place two pill substitutes in the first two slots of the Smart Pill Box spindle. Close the lid of the pill box. Bring a third party product tester into the room. Instruct them to attempt to open the pill box. If they succeed, instruct them to remove the spindle. If they succeed in removing the spindle, instruct them to remove the pill substitutes.
  Pass/Fail Criteria: The pill substitutes are able to be removed without additional instruction.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Dispensing	Pills are dispensed at the designated time and land in the tray	Pass	Pass
T2: Emptying	Notification is sent with each dispense; additional notification when empty	Pass	Pass
T3: Information	Health information accessible from the RSS interface	Pillbox log, and pillbox scheduling information is accessible from RSS interface	Pass
T4: Openable	A user can remove pill substitutes without additional instruction	Pass	Pass

Table B.5: Acceptance Test Procedure – Smart Pill Box

B.6 Smart Light

- **Test T1: Run test**
  Method: Provide power to the Smart Light. Place a piece of paper on top of the light bulb. Turn on the light bulb. Time 30 minutes using a stopwatch.
  Pass/Fail Criteria: The paper is not burned and the light does not fall over.

- **Test T2: Connection**
  Method: Provide power to the Smart Light. Kick the Kick Buttons to wirelessly turn the Smart Light on. Kick the Kick Buttons to wirelessly turn the Smart Light off. Repeat five times.
  Pass/Fail Criteria: The light is responsive.

- **Test T3: Replaceable**
  Method: Provide power to the Smart Light. Turn the Smart Light on. Turn the Smart Light off. Unscrew the light bulb from the socket. Screw a new light bulb into the socket. Turn on the Smart Light. Pass/Fail Criteria: The light illuminates using both light bulbs.

- **Test T4: Presentation**

Method: Visually inspect the device.
Pass/Fail Criteria: No wires or electronic components are visible.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Run Test	Paper does not burn; light does not fall over within 30 minutes of being on	Pass	Pass
T2: Connection	Smart Light responds to on/off commands from Kick Buttons, repeated 5 times	Pass	Pass
T3: Replaceable	Light illuminates using both original and replacement light bulbs	Pass	Pass
T4: Presentation	No wires or electronic components visible upon visual inspection	Wires and electrical components are mostly contained and hidden by the box	Pass

Table B.6: Acceptance Test Procedure – Smart Light

B.7 Smart Door Opener

Safety and Function Requirements

- **Test T1: Initial Function**
  Equipment: Autoslide computer unit
  Method: Install the Autoslide as described in the installation manual [45]. Press the "Inside Sensor" button on the Autoslide computer unit.
  Pass/Fail Criteria: The door opens and closes smoothly.

- **Test T2: Sensor Ports**
  Equipment: Relay cable, Autoslide computer unit
  Method: Plug the relay cable into the sensor ports on the Autoslide unit, and ensure the wires are not touching on the other end. Then, connect the yellow and black wires on the cable.

Pass/Fail Criteria: The inside sensor LED lights up and the door opens when the yellow and black wires are connected.

- **Test T3: Collision Sensing**
  Equipment: Autoslide unit
  Method: Place a hand in the path of the door and press the inside sensor button. Hold the hand firm.
  Pass/Fail Criteria: The Smart Door Opener automatically stops and retracts after collision with the hand.

- **Test T4: Manual Opening During Power Outage**
  Equipment: Smart Door Opener (disconnected from outlet)
  Method: Disconnect the Smart Door Opener from the outlet. Manually attempt to slide the door.
  Pass/Fail Criteria: A student is able to open/close the door with only one hand and without excessive force.

Home Assistant Integration

- **Test T5: Relay Control**
  Equipment: Pico W, relay circuit, voltmeter, Home Assistant interface
  Method: Upload the program to the Pico W and connect it to the relay circuit and power. Connect a voltmeter to the relay pins that correspond to the yellow and black wires. Select the "Open Door" button on the Home Assistant website.
  Pass/Fail Criteria: The voltage between the two pins quickly changes from the original voltage to ¡100 mV and back within 1 second of the button press.

- **Test T6: Whole System**
  Equipment: Relay circuit, Autoslide unit, Home Assistant interface
  Method: Connect the yellow and black wires to their respective pins in the relay circuit. Select the "Open Door" button on the Home Assistant website.
  Pass/Fail Criteria: The door starts to open within 1 second of the button being pressed.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Initial Function	Door opens and closes smoothly after pressing "Inside Sensor" button	There is some jolting when the door first starts/stops, but it is otherwise smooth	Pass
T2: Sensor Ports	Inside sensor LED lights and door opens when yellow and black wires connected	Pass	Pass
T3: Collision Sensing	Door automatically stops and retracts after collision with hand	Pass	Pass
T4: Manual Opening During Power Outage	Door can be opened/closed manually with one hand and without excessive force	Pass, felt like a normal sliding door	Pass
T5: Relay Control	Voltage changes to ¡100 mV and back within 1 second of "Open Door" button press	Voltage dropped to 10 mV then returned to nominal within 0.6 seconds	Pass
T6: Whole System	Door starts to opens within 1 second of "Open Door" button press on Home Assistant	Delay of under 1 second	Pass

Table B.7: Acceptance Test Procedure – Smart Door Opener

B.8 Smart Security Door Camera

Design Requirement: The system captures images when motion is detected and transmits them via MQTT to a smart device.

- **Test T1: Motion Detection Trigger**
  Equipment: HC-SR501 PIR Sensor, Raspberry Pi Pico W
  Method: Walk into sensor range and monitor terminal output and MQTT topic `camera/motion`
  Pass/Fail Criteria: Sensor detects motion and the Pico publishes a motion event within 1 second.

- **Test T2: Image Capture & Transmission**
  Equipment: ArduCAM Mini 5MP Plus, Arduino UNO, Pico W, MQTT Broker, Home Assistant
  Method: Trigger motion; check `camera/frames/latest` for a valid base64 JPEG
  Pass/Fail Criteria: JPEG is published via MQTT within 1 second of motion trigger; image appears in Home Assistant.

- **Test T3: UART JPEG Data Integrity**
  Equipment: Arduino UNO, Pico W
  Method: Trigger image transfer and inspect for correct 0xFFD8/0xFFD9 markers
  Pass/Fail Criteria: JPEG reconstructs without byte corruption and renders correctly.

- **Test T4: MQTT Connection Stability**
  Equipment: Pico W, MQTT Broker, Smart Mobile Device
  Method: Reboot Pico and monitor if motion/image data resumes correctly
  Pass/Fail Criteria: Pico reconnects and publishes events and images within 5 seconds of reboot.

- **Test T5: Latency Measurement**
  Equipment: Full system with timestamped logs
  Method: Time delay between motion event and image display in Home Assistant
  Pass/Fail Criteria: End-to-end latency ≤ 1 second under normal operation.

- **Test T6: Overload Handling**
  Equipment: ArduCAM, Arduino, Pico W, MQTT Broker
  Method: Trigger 5+ motion events within 10 seconds
  Pass/Fail Criteria: System handles rapid capture and transfer without frame loss or order errors.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: Motion Detection Trigger	Motion detected and published to `camera/motion` within 1 second	Pico published motion state to MQTT within 0.5s of movement	Pass
T2: Image Capture & Transmission	JPEG published to `camera/frames/latest` within 1 second; image appears in Home Assistant	Valid base64 JPEG rendered correctly in Home Assistant within 0.8s	Pass
T3: UART JPEG Data Integrity	JPEG has correct markers (0xFFD8/0xFFD9) and no byte corruption	Image verified with header/footer intact; no transmission corruption	Pass
T4: MQTT Connection Stability	Pico reconnects and resumes MQTT publishing within 5 seconds of reboot	Reconnection confirmed via MQTT log; data published after reboot consistently	Pass
T5: Latency Measurement	End-to-end delay $\leq$ 1 second	Average observed delay was 0.73s across 10 trials	Pass
T6: Overload Handling	5 captures in 10 seconds; no dropped/misordered images	All frames received correctly and displayed in sequence in Home Assistant	Pass

Table B.8: Acceptance Test Procedure – Smart Security Door Camera

B.9 Smart Speaker

Design Requirement: Speaker is powered from the wall

- **Test T1: Power Level Check**
  Equipment: Smart Speaker, Multimeter
  Method: Connect the device to power. Verify the 20 V and 5 V voltages. After the 5 V voltage is verified, close the jumper to the Pico and verify the 3.3 V voltage.
  Pass/Fail Criteria: All voltages are accurate to within 1% or multimeter precision, whichever is worse.

- **Test T2: Microcontroller Check**
  Equipment: Microcontroller
  Method: Program the microcontroller with a test program that blinks the LED.
  Pass/Fail Criteria: The LED blinks and continues blinking while the microcontroller is powered from either the board or the wall power.

Design Requirement: Speaker communicates with the hub

- **Test T3: Home Assistant Check**
  Equipment: Microcontroller
  Method: Program the microcontroller with a test program that allows the RGB LED to be controlled by the hub. Change the colors via the interface.
  Pass/Fail Criteria: Colors selected on the hub interface appear on the board LED as expected.

Design Requirement: Speaker provides a microphone for voice recognition

- **Test T4: ADC Check**
  Equipment: Microcontroller
  Method: Program the microcontroller with a test program that sends the average of the received ADC data to the hub. Touch the ADC pin with a finger, and release.
  Pass/Fail Criteria: ADC output is near 2^{11} (midpoint of 12-bit precision) when finger is present; likely pulled near zero otherwise.

- **Test T5: Smart Speaker Microphone Streaming**
  Equipment: Smart Hub, Smart Speaker
  Method: Enable an audio stream from the smart speaker, as well as debug saving of voice recordings. Monitor CPU usage using the `htop` or task manager.
  Pass/Fail Criteria: Open Wake Word consumes significant CPU when streaming, and none otherwise. A debug recording containing recognizable speech is produced in the specified folder on the hub.

- **Test T6: Smart Speaker Voice Commands**
  Equipment: Smart Speaker, Smart Hub
  Method: Set up the smart speaker and speak voice commands with wake words.
  Pass/Fail Criteria: The hub responds to the command, performs the requested action, and plays back confirmation.

Design Requirement: Speaker can play audio streamed from the hub

- **Test T7: Audio Playback**
  Equipment: Smart Speaker, Smart Hub
  Method: Set up the smart speaker and play audio via the media player menu on the hub.
  Pass/Fail Criteria: Audio plays from the speaker and sounds clear and intelligible.

Test Procedure	**Acceptable Range**	**Actual Measurements**	**P/F**
T1: Power Level Check	All voltages (20 V, 5 V, 3.3 V) within 1% or multimeter precision	Pass	Pass
T2: Microcontroller Check	LED blinks continuously when powered from board or wall power	Pass	Pass
T3: Home Assistant Check	Colors selected on hub interface appear on RGB LED as expected	Not Measured	
T4: ADC Check	ADC output near 2^{11} with finger touch; near zero otherwise	Pass	Pass
T5: Smart Speaker Microphone Streaming	CPU usage high when streaming wake word; debug recording with speech produced	Pass	Pass
T6: Smart Speaker Voice Commands	Hub responds, performs actions, and plays confirmation on voice commands	Some issues with understanding our speech, but it worked when the speech was parsed correctly	Pass
T7: Audio Playback	Audio plays clearly and intelligibly from speaker via hub media player	Pass	Pass

Table B.9: Acceptance Test Procedure – Smart Speaker

B.10 Smart TV

Design Requirements

- **Test T1: TV Power On/Off Control**
  Test Method: Use SALE-R and the Smart Mobile Device to power the TV on and off.
  Acceptance Criteria: The TV turns on and off promptly with each control input.

- **Test T2: TV Volume Control**
  Test Method: Use SALE-R and the Smart Mobile Device to adjust the TV volume up and down; verify the change via on-screen indicators or audible volume shift.
  Acceptance Criteria: The volume adjusts accurately and consistently with each control input.

- **Test T3: Pause and Unpause Media**
  Test Method: While media is playing on the Roku, use SALE-R and the Smart Mobile Device to pause and unpause playback.
  Acceptance Criteria: Playback pauses and resumes immediately in response to each input.

- **Test T4: Rewind and Fast Forward Media**
  Test Method: During video playback, use SALE-R and the Smart Mobile Device to rewind and fast forward content; confirm movement within the video.
  Acceptance Criteria: The media player correctly rewinds and fast forwards in response to user input.

- **Test T5: Application Shortcut Control**
  Test Method: Use dedicated shortcut buttons on the SALE-R or Smart Mobile Device to open Roku applications such as YouTube, Hulu, and Disney+ directly, bypassing menu navigation.
  Acceptance Criteria: Each shortcut correctly launches the intended application without delay or misrouting.

Test Procedure	Acceptable Range	Actual Measurements	P/F
T1: TV Power On/Off Control	TV turns on and off promptly with each control input	TV powered off within 0.5 seconds of input; unable to turn TV on using Home Assistant	Fail
T2: TV Volume Control	Volume adjusts accurately and consistently; confirmed via on-screen or audible feedback	Volume changed smoothly; on-screen indicator and audio confirmed response	Pass
T3: Pause and Unpause Media	Playback pauses and resumes immediately on each input	Playback paused and resumed within 0.5 seconds of command	Pass
T4: Rewind and Fast Forward Media	Media rewinds and fast forwards correctly in response to user input	Video rewound/fast forwarded correctly within 0.5 seconds of input	Pass
T5: Application Shortcut Control	Shortcut buttons launch intended applications without delay or misrouting	Applications launched within 1 seconds with no errors using website, occasional lack of response from SALE-R likely due to debouncing from SALE-R	Pass

Table B.10: Acceptance Test Procedure – Smart TV

B.11 Window Shade

Mechanical Design Requirements

- **Test T1: Motor Interaction**
  Test Method: Apply voltage to the relays using a wall outlet, power supply, and connected motor; confirm that the motor rotates when the relays are activated.
  Expected Outcome: Motor rotates once the voltage applied to the relays is high.
  Actual Outcome: Motor rotates once the voltage applied to the relays is high.

- **Test T2: Breaker Component**
  Test Method: Use the built PCB and breaker to disable power to the motor; confirm that the motor does not rotate when the breaker is engaged.

Expected Outcome: The breaker disables power and prevents motor rotation.
Actual Outcome: The breaker successfully disables power and prevents motor rotation.

- **Test T3: Window Shades Reach**
  Test Method: Connect the motor and PCBs to the window shades and activate the system; observe whether the shades extend fully to the bottom of the window. Expected Outcome: The shades shall reach the bottom of the window.
  Actual Outcome: The shades reach the bottom of the window in 12.5 seconds.

- **Test T4: Accommodation of PCBs**
  Test Method: Insert PCBs into their designated housing using 3D Fusion printed parts; check fit and stability using a ruler or manual inspection. Expected Outcome: The PCBs shall fit securely within their housing with no movement or obstruction.
  Actual Outcome: The PCBs fit securely within their housing.

Electrical Functionality Requirements

- **Test T5: Voltage Converter Voltage Regulation**
  Test Method: Use a multimeter or oscilloscope to measure the output voltage from the voltage converter to the microcontroller under normal operation. Expected Outcome: The voltage supplied to the microcontroller output side should not exceed 5.1 V.
  Actual Outcome: The voltage supplied to the microcontroller by the voltage converter can be adjusted and results in 4.91V.

- **Test T6: Relay Circuit Current Limitation**
  Test Method: Measure current through each relay coil under standard operating conditions using a digital multimeter; evaluate both DSP2A-DC12V and G5LE-1 DC12 relays.
  Expected Outcome: Coil current should be $\leq$ 25 mA for the DSP2A-DC12V and $\leq$ 33.3 mA for the G5LE-1 DC12.
  Actual Outcome: Coil current satisfies the $\leq$ 25 mA for the DSP2A-DC12V and $\leq$ 33.3 mA for the G5LE-1 DC12. When enabled, the currents measured 14mA and 25mA, respectively.

- **Test T7: GPIO Output Voltage Limitation to Microcontroller**
  Test Method: Use a multimeter or oscilloscope to monitor voltage at the GPIO input of the microcontroller during normal operation. Expected Outcome: Voltage at the GPIO output should not exceed 3.8V.

Actual Outcome: Voltage of the GPIO output measured a high of 3.2V

Wi-Fi Integration Requirements

- **Test T8: Microcontroller Readings**
  Test Method: Use a computer and oscilloscope to observe GPIO signals from the microcontroller while pressing the up, down, and stop controls. Expected Outcome: The microcontroller shall register logic high or low signals corresponding to each input.
  Actual Outcome: The microcontroller correctly outputs a logic high for up, and logic low for down.

System Level Integration Requirements

- **Test T9: Control from Home Assistant Website**
  Test Method: With the motor and tube mounted to the ceiling and PCBs wired together and powered, send up/down/stop commands from the Home Assistant dashboard and listen for relay activation. Expected Outcome: The window shade responds accordingly.
  Actual Outcome: The window shades respond to the website. Audio feedback from the relay confirms activation.

- **Test T10: Control from Smart Mobile Device**
  Test Method: Use the SMD interface to send up/down/stop commands to the shades and verify correct movement with audio feedback from the relay. Expected Outcome: The window shades respond accordingly.
  Actual Outcome: The window shades respond to the SMD input. Audio feedback from the relay confirms activation.

- **Test T11: Control from SALE-R Kick Buttons**
  Test Method: Use the SALE-R kick buttons to send commands to the shades and monitor their response along with audio confirmation from the relay. Expected Outcome: The window shades respond accordingly.
  Actual Outcome: The window shades respond to the SALE-R input. Audio feedback from the relay confirms activation.

Test Procedure	**Acceptable Range**	**Actual Measurements**	**P/F**
T1: Motor Interaction	Motor rotates once the voltage applied to the relays is high.	Motor rotates once the voltage applied to the relays is high.	Pass
T2: Breaker Component	The breaker disables power and prevents motor rotation.	The breaker successfully disables power and prevents motor rotation.	Pass
T3: Window Shades Reach	The shades shall reach the bottom of the window.	The shades reach the bottom of the window in 12.5 seconds.	Pass
T4: Accommodation of PCBs	The PCBs shall fit securely within their housing with no movement or obstruction.	The PCBs fit securely within their housing.	Pass
T5: Voltage Converter Voltage Regulation	The voltage supplied to the microcontroller output side should not exceed 5.1 V.	The voltage supplied to the microcontroller by the voltage converter can be adjusted and results in 4.91V.	Pass
T6: Relay Circuit Current Limitation	Coil current should be $\leq$ 25 mA for the DSP2A-DC12V and $\leq$ 33.3 mA for the G5LE-1 DC12.	When enabled, the currents measured 14mA and 25mA, respectively.	Pass
T7: GPIO Output Voltage Limitation to Microcontroller	Voltage at the GPIO output should not exceed 3.8V.	Voltage of the GPIO output measured a high of 3.2V	Pass
T8: Microcontroller Readings	The microcontroller shall register logic high or low signals corresponding to each input.	The microcontroller correctly outputs a logic high for up, and logic low for down.	Pass
T9: Control from Home Assistant Website	The window shade responds accordingly.	The window shades respond to the website. Audio feedback from the relay confirms activation.	Pass
T10: Control from Smart Mobile Device	The window shades respond accordingly.	The window shades respond to the SMD input. Audio feedback from the relay confirms activation.	Pass
T11: Control from SALE-R Kick Buttons	The window shades respond accordingly.	The window shades respond to the SALE-R input. Audio feedback from the relay confirms activation.	Pass

Table B.11: Acceptance Test Procedure – Smart Window Shades

APPENDIX C

Schematics

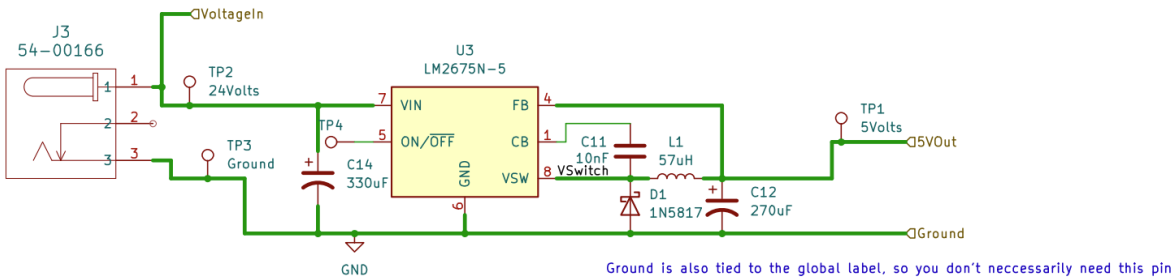

Figure C.1: Standard 5V Converter Circuit Schematic

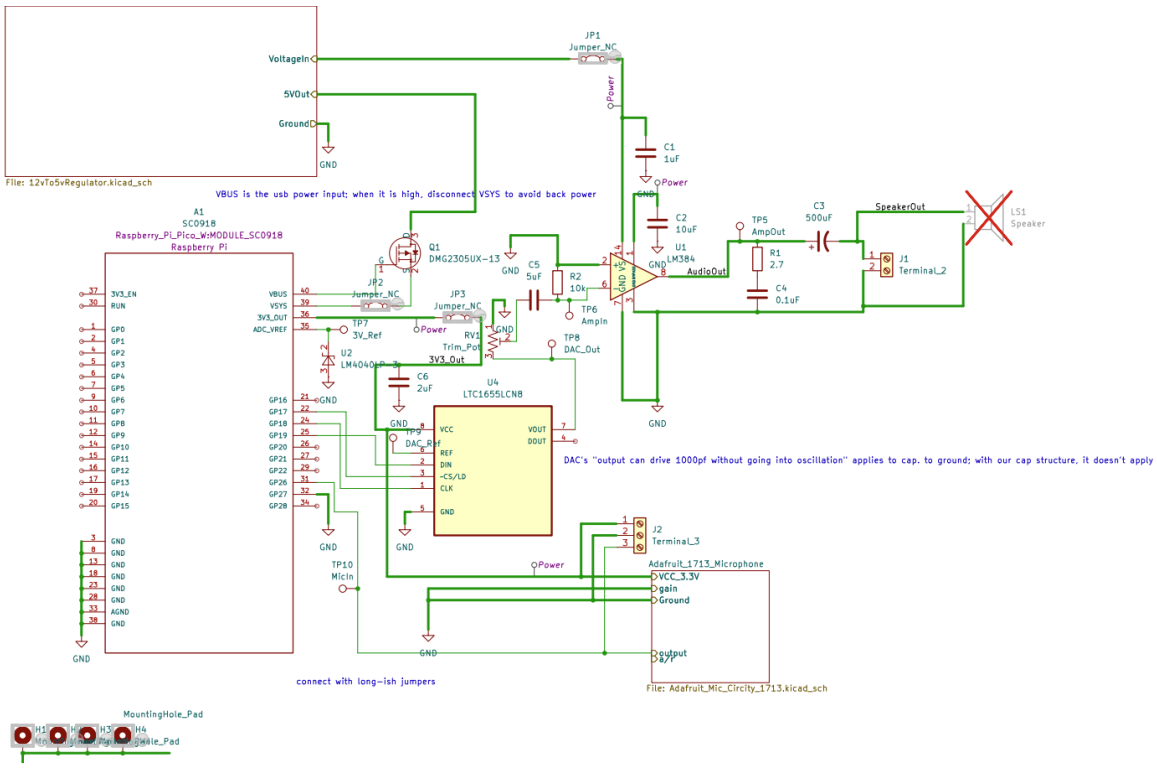

Figure C.2: Smart Speaker Schematic

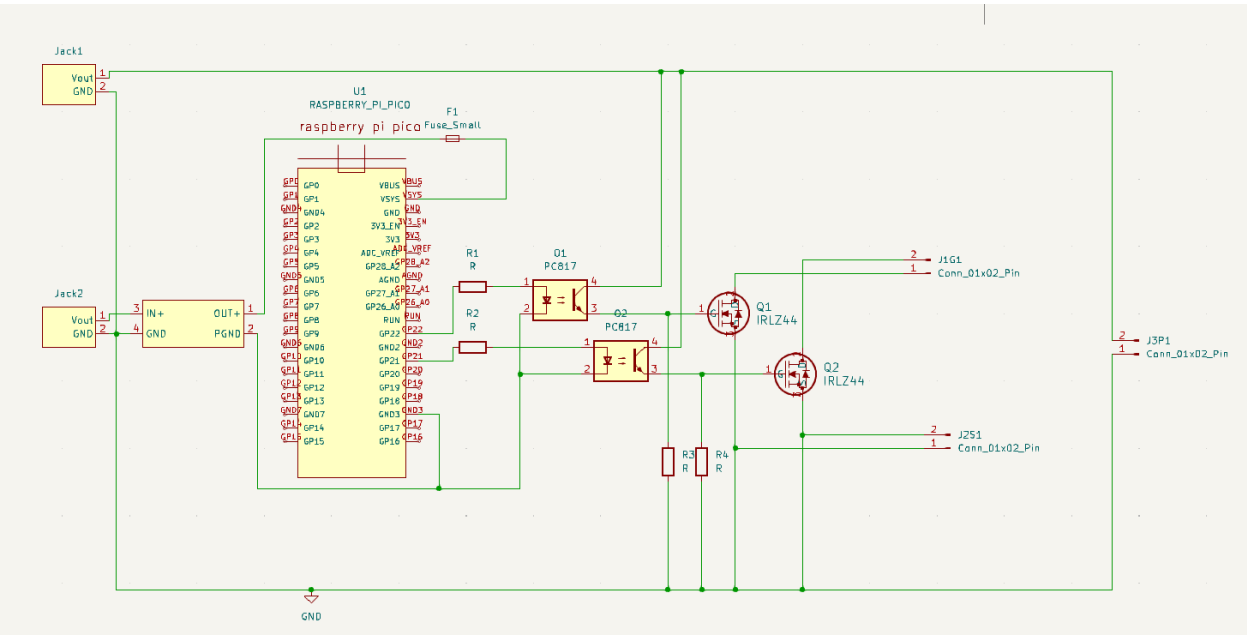

Figure C.3: Micro PCB Schematic of the Window Shades

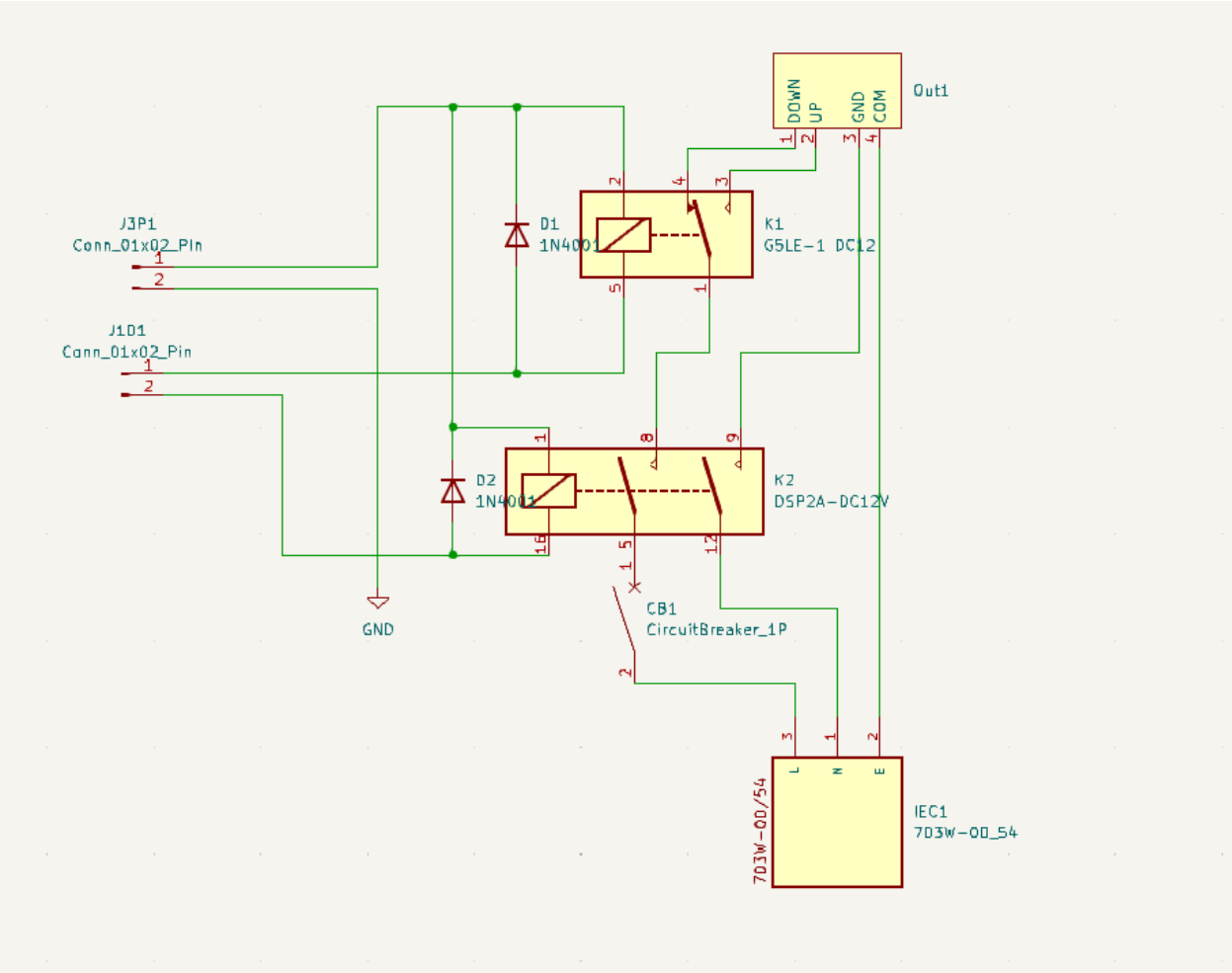

Figure C.4: Motor PCB Schematic of the Window Shades

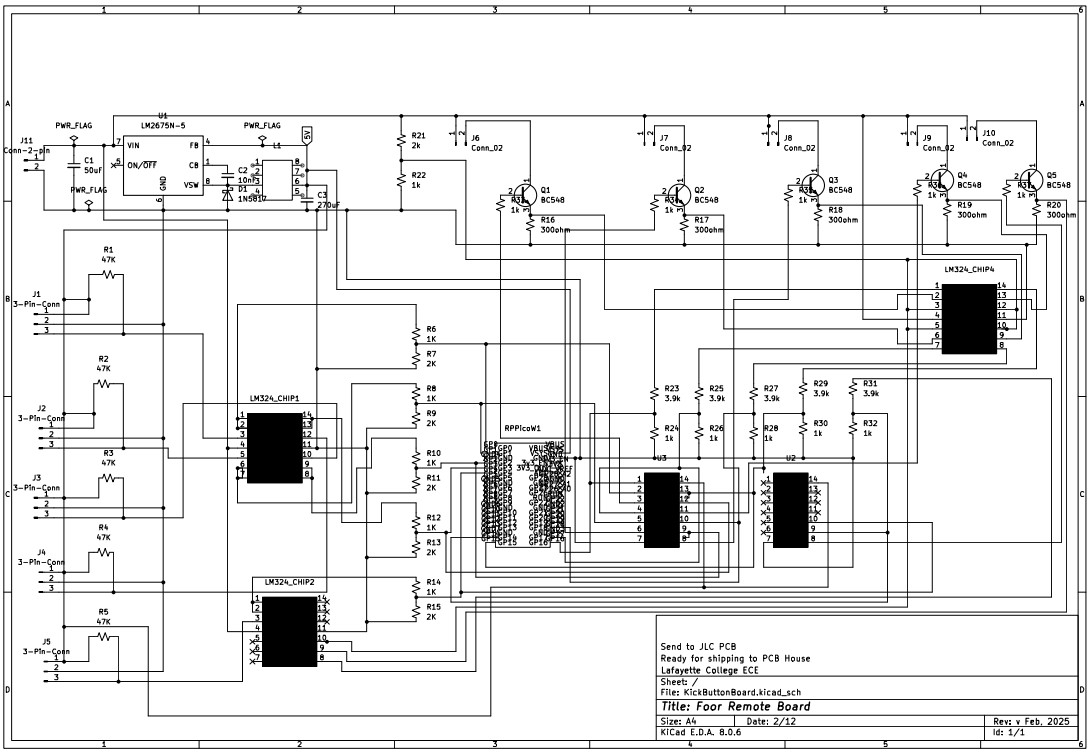

Figure C.5: SALE-R Board

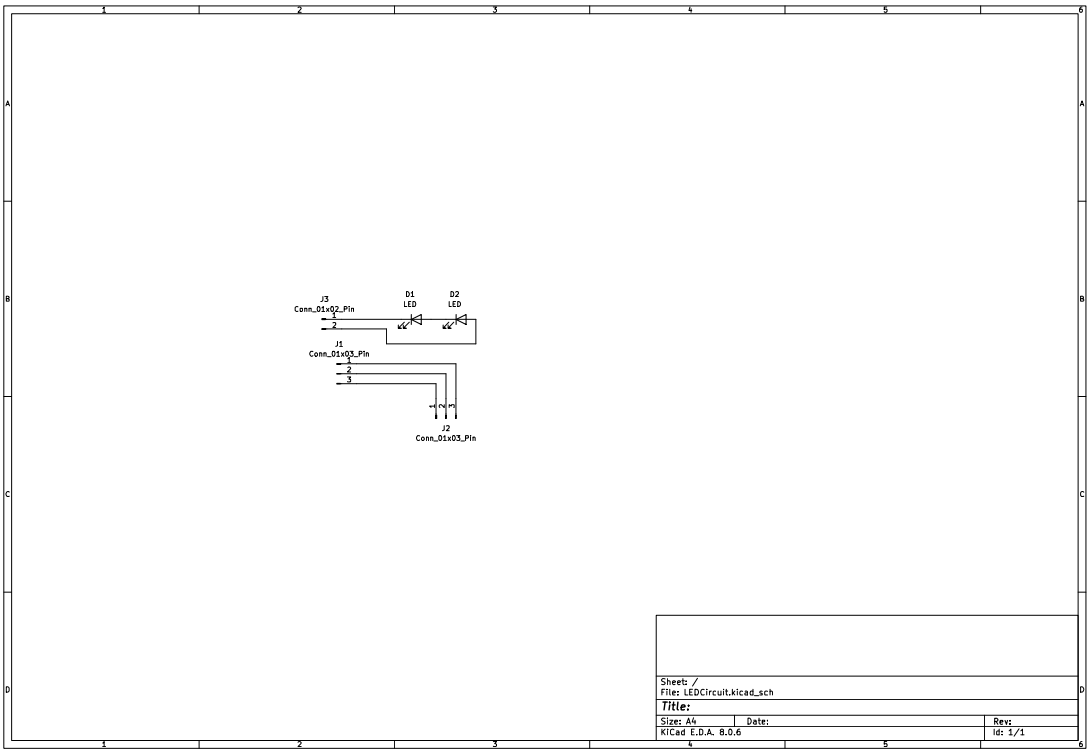

Figure C.6: SALE-R LED Circuit

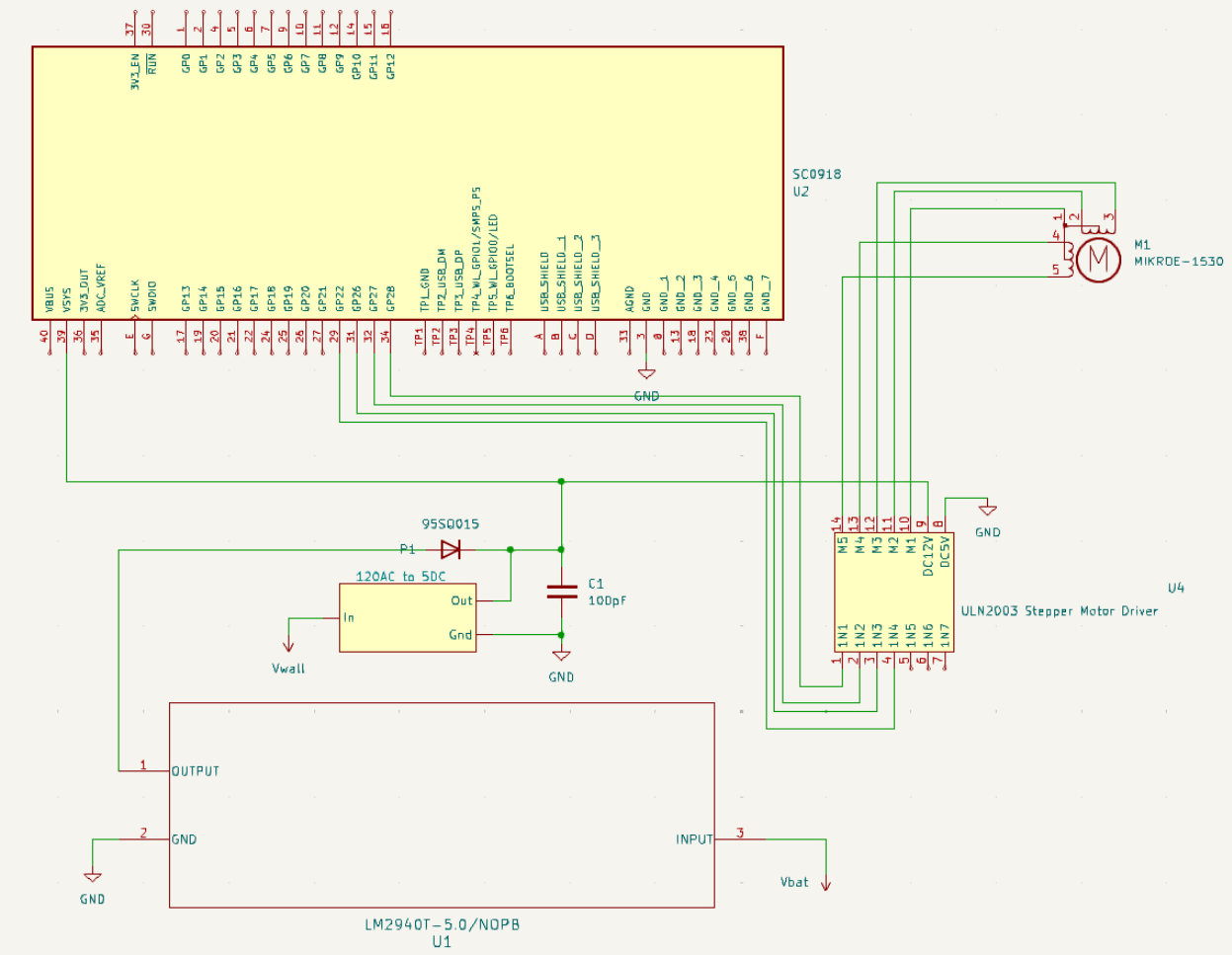

Figure C.7: Smart Pillbox Schematic

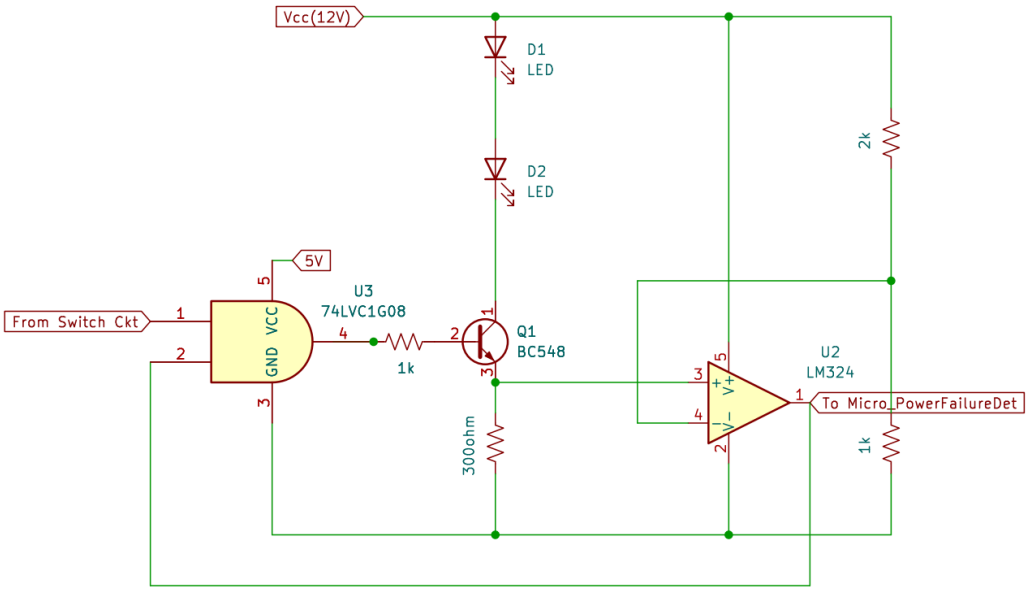

Figure C.8: SALE-R LED Circuit for 1 Button

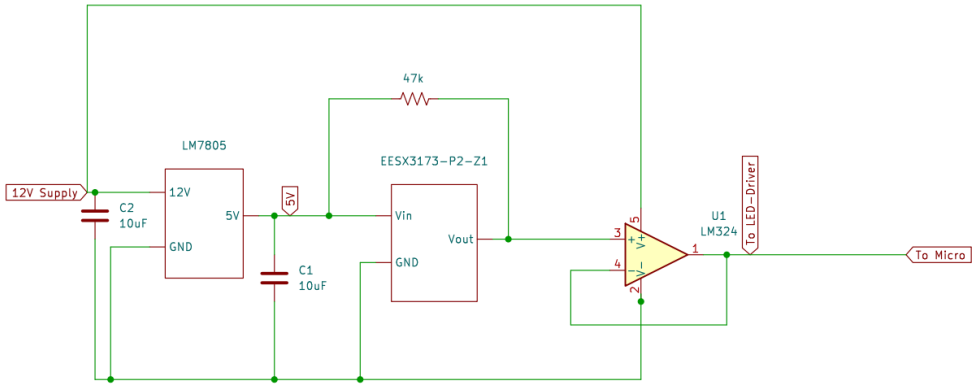

Figure C.9: SALE-R Detection Circuit Schematic for 1 Button

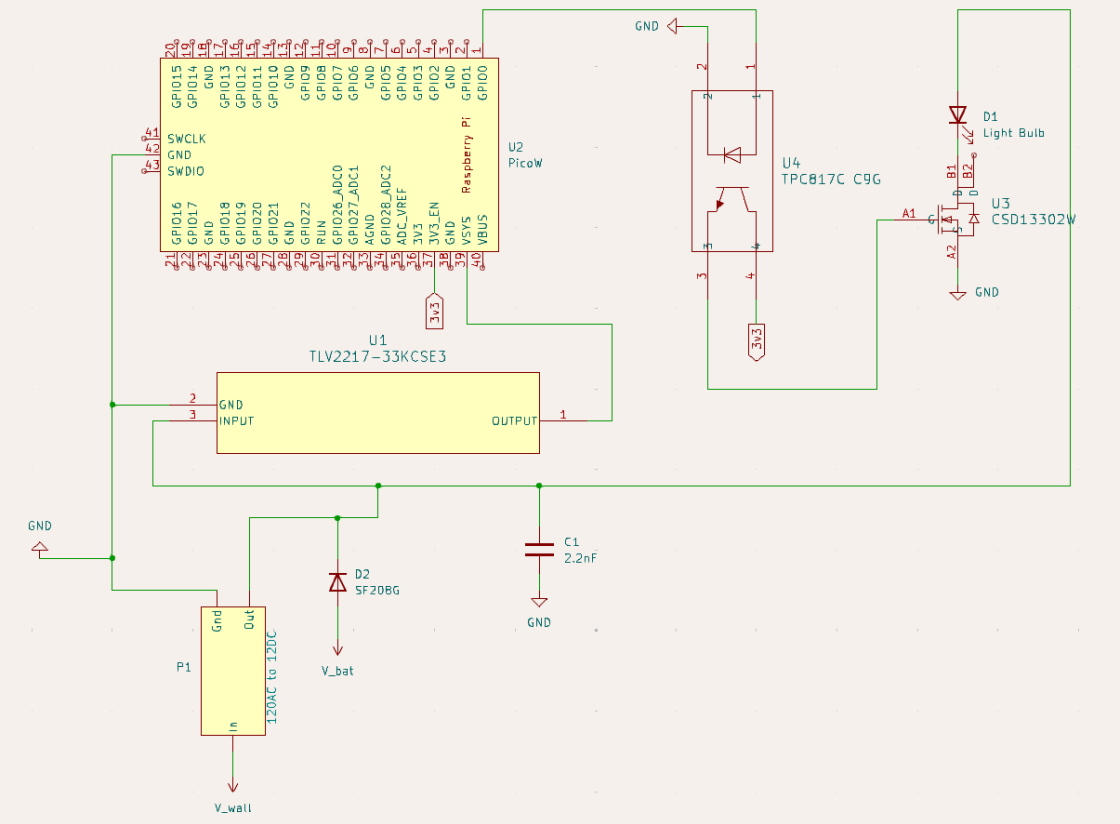

Figure C.10: Smart Light Schematic

APPENDIX D

Bill of Materials

Detailed device breakdown available at SALE-Budget.xlsx. The costs listed include prototyping costs.

D.1 Overall Budget

Device/Component	**Estimated Cost**
Smart Hub	$185.37
Smart Mobile Device (SMD)	$481.80
SALE-R	$244.73
Remote Support Station (RSS)	$0.00
Smart Light	$128.00
Smart Door Lock	$272.41
Smart Door Opener	$713.24
Smart Pill Box	$121.72
Smart Speaker	$239.93
Smart TV	$51.23
Smart Window Shade	$345.15
Sub-total	$2,783.58
Shared costs/components, shipping, etc	$61.92
Total	**$2,845.50**

Table D.1: Final Cost Breakdown of Devices and Components

Device/Component	**Estimated Cost**
Smart Hub	$495.37
Smart Mobile Device (SMD)	$451.81
SALE-R	$614.69
Remote Support Station (RSS)	$0.00
Smart Light	$375.07
Smart Door Lock	$276.36
Smart Door Opener	$676.08
Smart Pill Box	$204.29
Smart Speaker	$428.60
Smart TV	$75.69
Smart Window Shade	$341.04
Sub-total	$3,939.00
10% Additional Buffer – shipping and redesigns	$393.90
Total	**$4,332.90**

Table D.2: Initial Budget Proposal

BIBLIOGRAPHY

[1] C. G. Willes, "Smart homes," *Technology and Disability*, vol. 15, pp. 163–164, 2003.

[2] P. Fougeyrollas, L. N. Bergeron, R. Cloutier, S. Dion, and G. S. Michel, "Social consequences of long term impairments and disabilities: Conceptual approach and assessment of handicap," *International Journal of Rehabilitation Research*, vol. 21, pp. 127–141, 1998.

[3] M. Chan, E. Campo, D. Estève, J.-Y. Foumiols, and J-Y, "Smart homes - current features and future perspectives," *Maturitas*, vol. 64, pp. 90–97, 2009.

[4] P. Rashidi, D. Cook, L. Holder, M. Schmitter, and M. Edgecombe, "Discovering activities to recognize and track in a smart environment," *IEEE Transactions on Knowledge and Data Engineering*, 2010, http://www.eecs.wsu.edu/~cook/pubs/tkde10.pdf.

[5] R. J. Robles and T.-H. Kim, "Review: Context aware tools for smart home development," *International Journal of Smart Home*, vol. 4, pp. 1–12, Jan. 2010.

[6] M. R. Alam, M. B. I. Reaz, and M. A. M. Ali, "A review of smart homes—past, present, and future," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1190–1203, Nov. 2012.

[7] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: Challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, Jan. 2017.

[8] D. Mocrii, Y. Chen, and P. Musilek, "Iot-based smart homes: A review of system architecture, software, communications, privacy and security," *Internet of Things*, vol. 1–2, pp. 81–98, Sep. 2018.

[9] E. Ahmed, A. Islam, F. Sarker, M. N. Huda, and K. Abdullah-Al-Mamun, "A road to independent living with smart homes for people with disabilities," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, Dhaka, Bangladesh, 2016, pp. 472–477.

[10] admin, "Empowering Independence: How Smart Home Devices Transform Accessibility for People with Disabilities," Nov. 2024. [Online]. Available: https://know-the-ada.com/how-smart-home-devices-aid-people-with-disabilities/

[11] P. Purohit, P. Khanpara, U. Patel, and P. Kathiria, "Iot based ambient assisted living technologies for healthcare: Concepts and design challenges," in *2022 Sixth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Nov. 2022, pp. 111–116.

[12] L. Varriale, P. Briganti, and S. Mele, "Disability and home automation: Insights and challenges within organizational settings," in *Exploring Digital Ecosystems*, A. Lazazzara, F. Ricciardi, and S. Za, Eds. Cham: Springer International Publishing, 2020, pp. 47–66.

[13] M. Periša, P. Teskera, I. Cvitić, and I. Grgurević, "Empowering people with disabilities in smart homes using predictive informing," *Sensors*, vol. 25, no. 1, p. 284, Jan. 2025, https://doi.org/10.3390/s25010284.

[14] Z. Wang, S. Dogan, M. Stolikj, and A. M. Kondoz, "Towards adaptive control in smart homes: Overall system design and initial evaluation of activity recognition," in *2017 Intelligent Systems Conference (IntelliSys)*, London, UK, 2017, pp. 129–136.

[15] P. Mtshali and F. Khubisa, "A smart home appliance control system for physically disabled people," in *2019 Conference on Information Communications Technology and Society (ICTAS)*, Durban, South Africa, 2019, pp. 1–5.

[16] D. Bacchin, P. Pluchino, A. Z. Grippaldi, D. Mapelli, A. Spagnolli, A. Zanella, and L. Gamberini, "Smart co-housing for people with disabilities: A preliminary assessment of caregivers' interaction with the domho system," *Frontiers in Psychology*, vol. 12, Sep. 2021, https://doi.org/10.3389/fpsyg.2021.734180.

[17] Link Labs, "Bluetooth vs. bluetooth low energy (ble): What's the difference?" 2023, accessed: 2025-05-18. [Online]. Available: https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy

[18] M. Afaneh, "Bluetooth low energy (ble) – a complete guide," 2022, accessed: 2025-05-18. [Online]. Available: https://novelbits.io/bluetooth-low-energy-ble-complete-guide/

[19] Home Assistant, "Mqtt integration," n.d., accessed: 2025-05-18. [Online]. Available: https://www.home-assistant.io/integrations/mqtt/

[20] S. Cope, "How mqtt works," 2021, accessed: 2025-05-18. [Online]. Available: http://www.steves-internet-guide.com/mqtt-works/

[21] MQTT Org, "Mqtt: The standard for iot messaging," n.d., accessed: 2025-05-18. [Online]. Available: https://mqtt.org/

[22] Hardkernel, "ODROID-H4 – ODROID." [Online]. Available: https://www.hardkernel.com/shop/odroid-h4/

[23] MIT App Inventor, "Mit app inventor," 2025, accessed: 2025-05-18. [Online]. Available: https://ai2.appinventor.mit.edu/

[24] Android Developers, "Bluetooth low energy overview," 2025, accessed: 2025-05-18. [Online]. Available: https://developer.android.com/develop/connectivity/bluetooth/ble/ble-overview

[25] MIT App Inventor, "Bluetoothle component documentation," 2025, accessed: 2025-05-18. [Online]. Available: https://iot.appinventor.mit.edu/iot/reference/bluetoothle

[26] Home Assistant Developers, "Rest api," 2025, accessed: 2025-05-18. [Online]. Available: https://developers.home-assistant.io/docs/api/rest/

[27] MIT App Inventor, "Web component - connectivity documentation," 2025, accessed: 2025-05-18. [Online]. Available: https://ai2.appinventor.mit.edu/reference/components/connectivity.html#Web

[28] Home Assistant Developers, *Dashboards*, Home Assistant, 2025, available: https://www.home-assistant.io/dashboards/, Accessed: May 18, 2025.

[29] ——, *Authentication*, Home Assistant, 2025, available: https://developers.home-assistant.io/docs/auth_index, Accessed: May 18, 2025.

[30] ——, *Core Entity - DateTime*, Home Assistant, 2025, available: https://developers.home-assistant.io/docs/core/entity/datetime, Accessed: May 18, 2025.

[31] J. Wook Kim, "whisper," May 2025, original-date: 2022-09-16T20:02:54Z. [Online]. Available: https://github.com/openai/whisper

[32] H. Assistant, "Getting started - Local." [Online]. Available: https://www.home-assistant.io/voice_control/voice_remote_local_assistant/

[33] M. Hansen, "rhasspy/piper," May 2025, original-date: 2023-01-10T22:25:40Z. [Online]. Available: https://github.com/rhasspy/piper

[34] The Exit Light Company, "NFPA 101 Life Safety Code," n.d., accessed: 2025-05-18. [Online]. Available: https://www.exitlightco.com/NFPA-101-Life-Safety-Code.html

[35] M. McCauley, "Accelstepper arduino library," n.d., accessed: 2025-05-18. [Online]. Available: https://www.airspayce.com/mikem/arduino/AccelStepper/

[36] Home Assistant Community, *Roku Integration*, Home Assistant Documentation, 2025, available: https://www.home-assistant.io/integrations/roku/.

[37] Raspberry Pi Ltd., *Connecting to the Internet with Raspberry Pi Pico W*, Raspberry Pi Documentation, 2024, available: https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf.

[38] Python Software Foundation, *socket — Low-level networking interface*, Python Documentation, 2025, available: https://docs.python.org/3/library/socket.html.

[39] ——, *gc — Garbage Collector interface*, Python Documentation, 2025, available: https://docs.python.org/3/library/gc.html.

[40] D. Campos, "daniloc/PicoW_homeassistant_starter," May 2025, original-date: 2022-08-28T01:13:53Z. [Online]. Available: https://github.com/daniloc/PicoW_HomeAssistant_Starter

[41] M.-V. Drăgoi and et al., "Real-time home automation system using bci technology," *Biomimetics (Basel)*, vol. 9, no. 10, p. 594, Oct. 2024.

[42] A. Bissoli, D. Lavino-Junior, M. Sime, L. Encarnação, and T. Bastos-Filho, "A human–machine interface based on eye tracking for controlling and monitoring a smart home using the internet of things," *Sensors (Basel)*, vol. 19, no. 4, p. 859, Feb. 2019.

[43] R. A. J. de Belen, T. Bednarz, and D. D. Favero, "Integrating mixed reality and internet of things as an assistive technology for elderly people living in a smart home," in *Proceedings of the 17th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry (VRCAI '19)*. Association for Computing Machinery, Nov. 2019, pp. 1–2.

[44] R. K. Nath and H. Thapliyal, "Wearable health monitoring system for older adults in a smart home environment," Jun. 2021, arXiv:2107.09509.

[45] AutoSlide LLC, *AutoSlide Installation Manual*, AutoSlide, 2022, available: https://autoslide.com/wp-content/uploads/2022/11/AutoSlide-Installation-Manual-102122.pdf, Accessed: May 18, 2025.