HelpMeRecall

By

Michael Jiang

Sravya Davuluri

William Li

Final Report for ECE 445, Senior Design, Spring 2026

TA: Hossein Ataee

06 May 2026

Project No. 33

Abstract

HelpMeRecall serves as an assistive memory device that is placed on the wrist and is intended to allow users to log their recent activities through sensor-gated voice interaction. The device uses an ESP32-S3 microcontroller, capacitive touch sensor, accelerometer, microphone, speaker, LED, and non-volatile flash storage via the ESP32. Voice processing is enabled only when both the touch and accelerometer sensors detect valid user interaction, reducing accidental activation and unnecessary power drainage. When an activity keyword is detected, the device records it with a timestamp attached. When the user says “query,” the device outputs stored logs through the speaker. The final prototype has successfully implemented and integrated the various subsystems, keyword detection, timestamped logging, log deletion, LED feedback, and speaker response. Verification showed fast state transitions, working sensor gating, audible query responses, and stable 3.3 V power operation. Though we implemented the final integrated system on a breadboard rather than the PCB, the prototype demonstrated the intended core functionality.

Contents

1. Introduction

1.1 Purpose

Short-term memory loss is a problem in people, particularly as people age. Many individuals have difficulty remembering recent activities throughout the day and in some cases people may forget routine tasks like eating or taking medication. According to the Alzheimer's Association, 7.2 million Americans aged 65 and older suffer from Alzheimer's disease in 2025 and this number is expected to grow in the coming years [1].

Although the younger population generally don't have Alzheimer's disease, students and professors can be forgetful. Under the pressures of academia and constant deadlines, it's easy to forget self-care. However, these issues can begin to be resolved with a person who can also care for others or more simply, a device that can log your activities as you go on with your day.

1.2 Functionality

Users will be able to interact with an accelerometer, touch sensor, microphone, LED, button, and speaker. Before using the device, users can see whether the device is powered with the LED. During use, the LED will also act as a status indicator. The accelerometer detects motion from the user, where upon sensing motion the user can touch the touch sensor. This activates the microphone to start listening for keywords. If the user speaks the keyword "query", the device will output all the recorded logs with their timestamp via the speaker. Users will also be able to delete the latest log by pressing the button once, or delete all logs by holding the button down for 3 seconds.

1.3 Subsystem Overview

The final device consists of 4 subsystems as seen in Fig. 1 below.

1) Microcontroller unit subsystem consists of only the ESP32, but handles all the device logic
2) Input subsystem consists of the touch sensor, accelerometer, microphone, and button
3) Output and Feedback subsystem consists of speaker, amplifier, vibration motor, and LED
4) Power subsystem consists of a 5 V battery to power the entire system

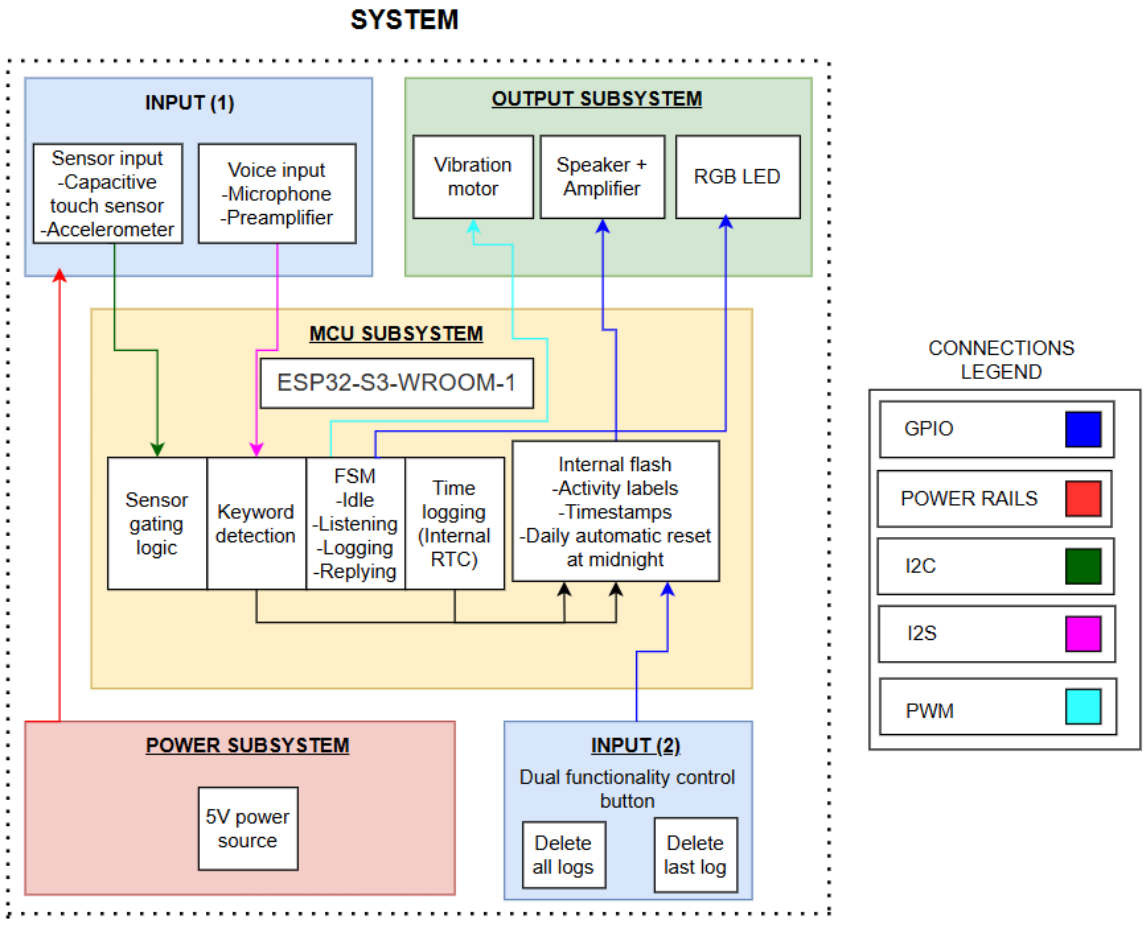

Figure 1: Block diagram

2. Design

2.1 Design Description & Justification

The HelpMeRecall device is designed to prioritize reliable operation and low-power, intentional user interaction. This is achieved through a sensor-gated interface, where voice input is only processed when both touch and motion conditions are satisfied. By avoiding continuous audio processing, this approach significantly reduces power consumption while also minimizing unintended activations.

To further improve efficiency, the system enforces a listening timeout of 10 seconds when no valid keyword is detected. This prevents prolonged operation of the microphone and associated processing, which are among the more energy-intensive components. Together, these mechanisms ensure that system resources are only used when necessary.

User feedback is designed to be immediate and intuitive through a combination of visual and haptic outputs. The LED provides real-time indication of system state, while the vibration motor confirms successful activity logging. This multi-modal feedback improves usability without increasing system complexity.

System behavior is controlled by a Finite State Machine (FSM), shown in Figure 2, consisting of four states: idle, listening, logging, and replying. This structure ensures that only the required components are active in each state, reducing unnecessary power draw. The FSM also provides a clear and deterministic control flow, simplifying implementation and debugging.

The ESP32-S3-WROOM-1 module was selected as the microcontroller due to its higher memory capacity and support for lightweight machine learning applications. These features are necessary for implementing on-device keyword detection using a machine learning model while maintaining real-time performance. Integrating all processing on a single microcontroller also reduces system latency and hardware complexity.

Overall, the design emphasizes energy efficiency, robustness, and ease of use, ensuring consistent performance under real-world operating conditions.

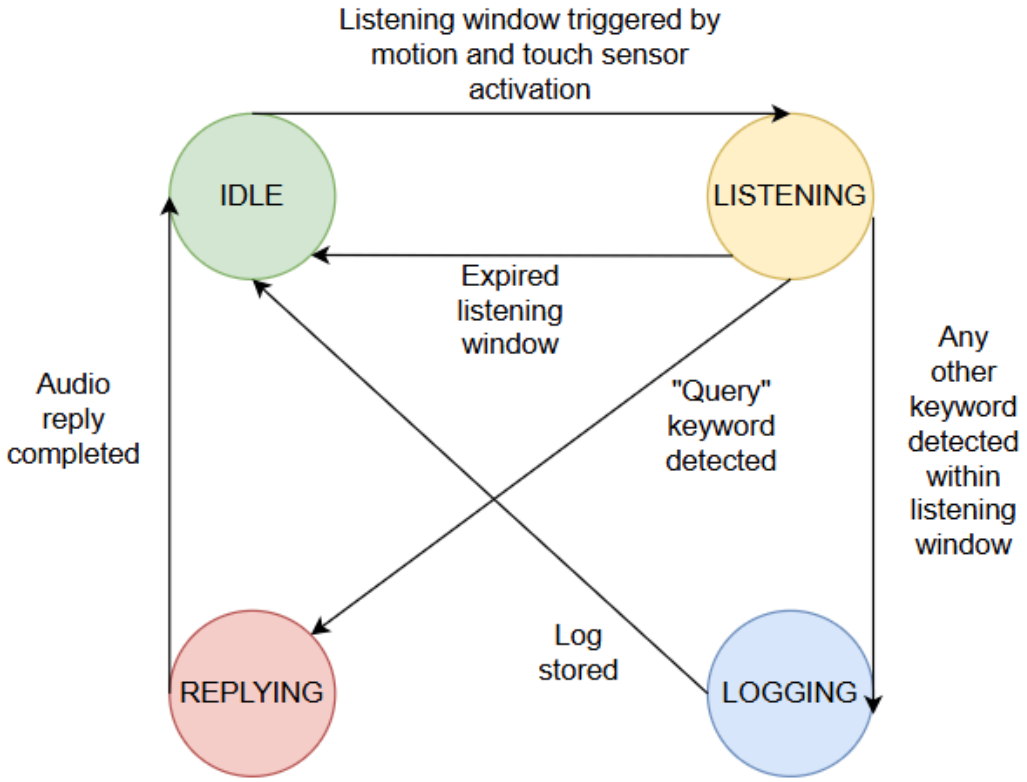

Figure 2: Finite State Machine Diagram

2.2 Equations and Simulations

In the demo, we used a power pack that supplied a 5 V supply while the ESP32 provided a regulated 3.3 V rail for the sensors and supporting components. Because the device is not continuously recording audio, the average current depends strongly on how long the system remains in each state. The main current-consuming components are the ESP32-S3, microphone, accelerometer, LED, speaker/amplifier, and vibration motor.

The estimated battery life can be calculated using:

$$t = C / I\_avg$$

where *t* is battery life in hours, *C i*s battery capacity in mAh, and I_avg is the average current draw in mA.

As an alternative to a rechargeable battery pack, we are also able to use four AA batteries in series to power the system. This would supply 6 volts. Assuming the AA batteries supply 2000 mAh similar to Duracell batteries:

$$Power\ consumption\ of\ device \sim 0.363\ W$$

$$Battery\ energy\ = \ 6\ V \times 2\ Ah\ = \ 12\ Wh$$

$$Battery\ life\ = \ 12\ Wh\ /\ 0.363\ W \approx 33.1\ hours$$

This estimate shows that the device can operate for several hours under normal usage. The actual battery life depends on how often the user activates listening mode, logs keywords, or triggers the speaker response. Since the microphone and keyword detection are only enabled after touch and motion validation, the device avoids the higher current draw of continuous audio processing. This supports the design decision to use sensor gating and a listening timeout.

The 3.3 V rail was also verified under load in the power subsystem verification. The measured voltage remained within the required 3.3 V ± 5% range, meaning the acceptable voltage range was:

$$3.3(0.95)\ = \ 3.135\ V$$

$$3.3(1.05)\ = \ 3.465\ V$$

Since the measured output stayed within 3.135 V to 3.465 V, the power subsystem was considered stable enough to support the ESP32 and attached components during operation.

2.3 Subsystem Diagrams & Schematics

The overall system architecture is shown in Figure 1, which illustrates the interconnection between all four subsystems, and the signal flow through the device. The microcontroller unit has centralized logic and coordinates all input processing, state transitions, device logic, and outputs. The system is powered using a 5 V battery source. Communication between subsystems is implemented using multiple communication protocols for different components. Overall, the subsystem design emphasizes clear signal flow, efficient communication protocols, and electrical compatibility, enabling reliable integration of all system components.

2.3.1 MCU Subsystem

Figure 3 shows the MCU subsystem. This subsystem is responsible for primary functionality logic and coordination of all other subsystems. It has several logical components:

1) Sensor gating: Sensor gating is implemented in software using inputs from the touch sensor (GPIO) and accelerometer (I2C). The MCU evaluates these signals within a defined time window to determine whether valid user interaction has occurred before enabling audio processing.

2) Keyword detection: Keyword detection is performed on audio data received through the I2S interface from the microphone. The sampled audio is processed using a lightweight model optimized for real-time execution on the microcontroller.

   The keyword detection model was developed and trained using the Edge Impulse platform [2], which enables efficient deployment of embedded machine learning models. This approach allows on-device inference without requiring external computation, ensuring low latency and preserving user privacy. We exported the model as a library and integrated it into the code.

3) Finite State Machine: Figure 2 describes the Finite State Machine transitions between the four defined states. The idle state is the starting state. The listening state is activated through sensor gating, and while the listening window is active, the microphone is listening to detect the keyword in the user's speech. The logging state records the audio of the user speaking the keyword it detected, and stores it in flash memory along with the timestamp. In the replying state which is triggered when the keyword detected in the listening state is "query", the speaker outputs all stored logs in the format of user's voice recording saying an activity keyword + timestamp.

4) Time logging: Time logging is handled using internal timing resources, allowing each activity to be associated with a timestamp.

5) Internal memory (flash): Logged data is stored in internal flash memory, providing persistent storage without requiring external components.

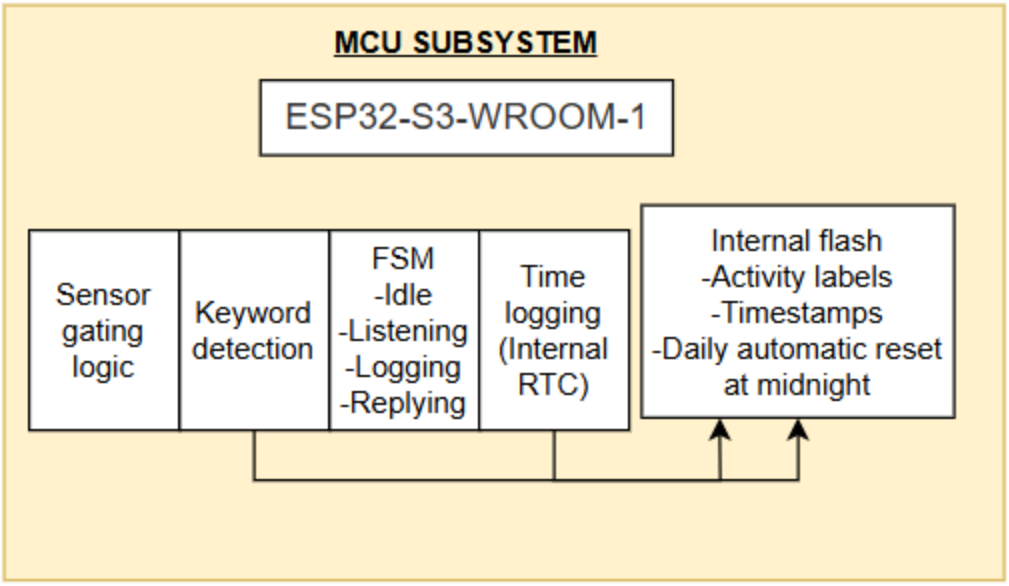

Figure 3: MCU Subsystem

2.3.2 Input Subsystem

The input subsystem (Figure 4) is responsible for capturing and validating user interaction before initiating system actions. It consists of three different kinds of input to the device.

1) Sensors: Sensor inputs include the capacitive touch sensor and accelerometer. The touch sensor provides a digital signal through GPIO, while the accelerometer communicates motion data via I2C. These signals are combined to implement sensor gating. This ensures that the system only responds to intentional user actions, and times out after 10 seconds of no keyword detection after activation through these sensors.

2) Voice input: Voice input is captured using a digital microphone (INMP441), which transmits audio data to the MCU (via I2S protocol).

3) Dual functionality button: Short presses and long presses (3 seconds) are detected through GPIO input and interpreted in software to perform different actions. The short press deletes the latest stored activity log while the long press deletes all stored activity logs.

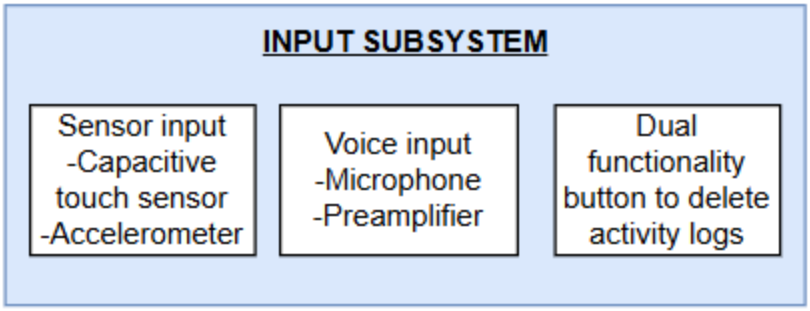

Figure 4: Input Subsystem

2.3.3 Output Subsystem

The output subsystem (Figure 5) provides three types of output from the device to the user:

1) Speaker: Audio data output to the speaker uses I2S protocol during the replying state.
2) Vibration motor: Haptic feedback through the vibration motor is controlled using a PWM signal. The vibration motor activates once an activity is successfully stored in memory, and before and after the speaker replies to the user.
3) RGB LED: Visual feedback through the RGB LED is handled through GPIO pins. This LED shows the different states that the system is in to aid usability. The LED is green in the idle state, yellow in the listening state, flashes red while logging an activity and is blue in the replying state.

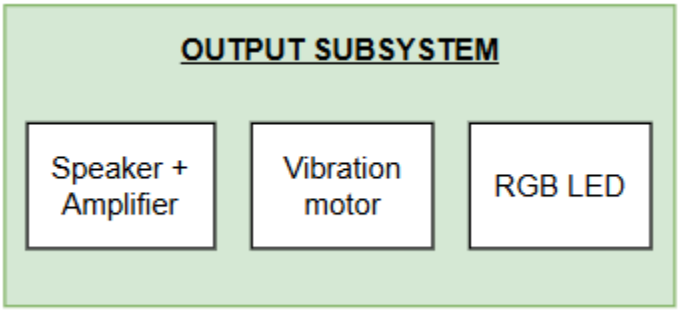

Figure 5: Output Subsystem

2.3.4 Power Subsystem

The power subsystem (Figure 6) consists of a 5 V power supply. 3.3 V output from the ESP32 to power all the components attached to the ESP32.

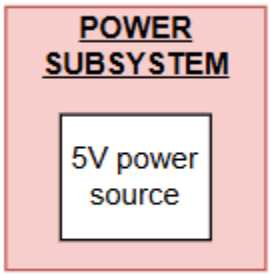

Figure 6: Power Subsystem

2.4 Design Alternatives

Several design alternatives were considered throughout the project. One major design choice was whether to use continuous listening or sensor-gated listening. Continuous listening would make the device easier to activate, but it would increase power consumption and raise privacy concerns because the microphone would be active more often. We chose sensor-gated listening instead, where the microphone only becomes active after both touch and motion are detected. This reduced accidental activations and helped extend battery life.

We also made design changes during implementation. Our original PCB design had issues that prevented the final device from working fully on the PCB, so the final prototype was verified on a breadboard. Our PCB had the USB-C port facing in a direction towards resistors, and certain devices plugged to the wrong voltages. We decided to implement it on small breadboards that fit in the wrist box, so it was able to hold the same major components as the PCB design, including the ESP32, microphone, accelerometer, touch sensor, LED, speaker, and amplifier, which mimics the intended original design.

3. Cost & Schedule

3.1 Cost

Table 1 lists the components used, along with their costs, in the making of our project.

The labor cost estimate for each partner calculates using the formula ideal salary (hourly rate) x actual hours spent x 2.5 assuming the hourly rate as $40 and hours spent as 45 hours, we get:

Labor cost estimate = 40 x 45 x 2.5 = $4500

The total cost of all our parts before shipping and tax is $94.89. With a 5% shipping cost and a 10% sales tax, the cost of our parts becomes $109.13. Thus, for our team of 3 members, the total labor cost is $13,500. Therefore, the cost estimate for our entire project is $13609.13.

Table 1: Itemized list of Components and Costs

Description	**Manufacturer**	**Quantity**	**Extended Price**	**Link/Location**
ESP32 Processor	Espressif	2	$0.00	ECE Shop
Omnidirectional Microphone	InvenSense	3	$8.99	Link
Capacitive Touch Sensor	Microchip Technology	3	$10.35	Link
Gyro Accelerometer	InvenSense	2	$13.98	Link
Amplifier	Maxim Integrated	3	$10.86	Link
Voltage Regulator	Microchip Technology	3	$1.50	Link
Stereo Speaker	CQRobot	2	$15.98	Link
Li-Po Rechargeable Battery - 3.7v 1200 mAh	Pkcell Battery	2	$19.90	Link

USB-C Header	G-Switch	5	$0.40	Link
Diode 17V 5A	STMicroelectronics	3	$1.08	Link
Coin Vibration Motor	DFRobot	3	$3.00	Link
Diode 50 V 1A	onsemi	3	$8.85	Link
Capacitor 4.7µF / 6.3V	Kemet	3	$0.00	ECE Shop
Capacitor 10µF / 20% / 10V	Cal-Chip	3	$0.00	
Capacitor 1µF / 6.3V	Kemet	6	$0.00	
Resistor 100kΩ 5% (1/8W)	Stackpole Electronics	3	$0.00	
Resistor 2.2kΩ 5% (1/8W)	Stackpole Electronics	3	$0.00	
Resistor 22Ω 1% (1/10W)	Yageo Group	6	$0.00	
Resistor 5.1kΩ 5% (1/8W)	Stackpole Electronics	6	$0.00	
Resistor 10kΩ	—	6	$0.00	
Resistor 240Ω 1% (1/10W)	Automotive AEC-Q200 Thick Film	3	$0.00	
MOSFET	Infineon Technologies	3	$0.00	ECE Shop
RGB LED	Optoelectronics	1	$0.00	
Tactile Switch	TE Connectivity	2	$0.00	
4 Pin Header	Sullins Connector Solutions	1	$0.00	
3 Pin Header	Sullins Connector Solutions	2	$0.00	

8 Pin Connector	TE Connectivity AMP Connectors	1	$0.00	
6 Pin Connector	Würth Elektronik	1	$0.00	

3.2 Schedule

Table 2 shows the schedule our team followed during the semester while working on this project.

Table 2: Schedule for Project Progression

Week	**Task**	**Person**
March 9th to March 16th	Breadboard testing, machine shop	Everyone
	Hardware verification	Michael
	Design review	Sravya
	PCB revision	William
March 16th to March 23rd	Ensure final and complete PCB design	Everyone
	Establish communication to accelerometer	Michael
	Code FSM logic	Sravya
	Microphone programming and filtering	William
March 23rd to March 30th	Final PCB ordering	Everyone
	Calibrate vibration motor feedback	Michael
	Program features	Sravya
	Verify touch sensor	William

March 30th to April 16th	Overall FSM logic complete, verify circuitry, and verify physical casing	Everyone
April 6th to April 13th	Complete touch sensor, accelerometer, LED, microphone	Everyone
April 13th to April 20th	Complete amplifier, speaker, timestamp code	Everyone
April 20th to April 27th	Complete assembly and integration Verify all project features	Everyone
April 27th	Demo	Everyone

4. Verifications and Results

We provide explanations of our verification techniques in this section. Tables 3, 4, 5, and 6 in Appendix A show the detailed Requirements and Verification table along with their verification status.

4.1 MCU subsystem

We used an oscilloscope to aid the verification process of our MCU subsystem. For example, Figure 7 shows a screenshot of an oscilloscope reading. We measure the time between when the green LED becomes HIGH to when the yellow LED becomes LOW (for this LED, a LOW signal indicates that the LED is on). The green LED indicates the idle state while the yellow LED indicates the listening state. The time between Green LED turning off and Yellow LED turning on is 950 µs, indicating that the state transition took 950 µs. We took 5 trials for this, resulting in an average transition time of 588.36 µs.

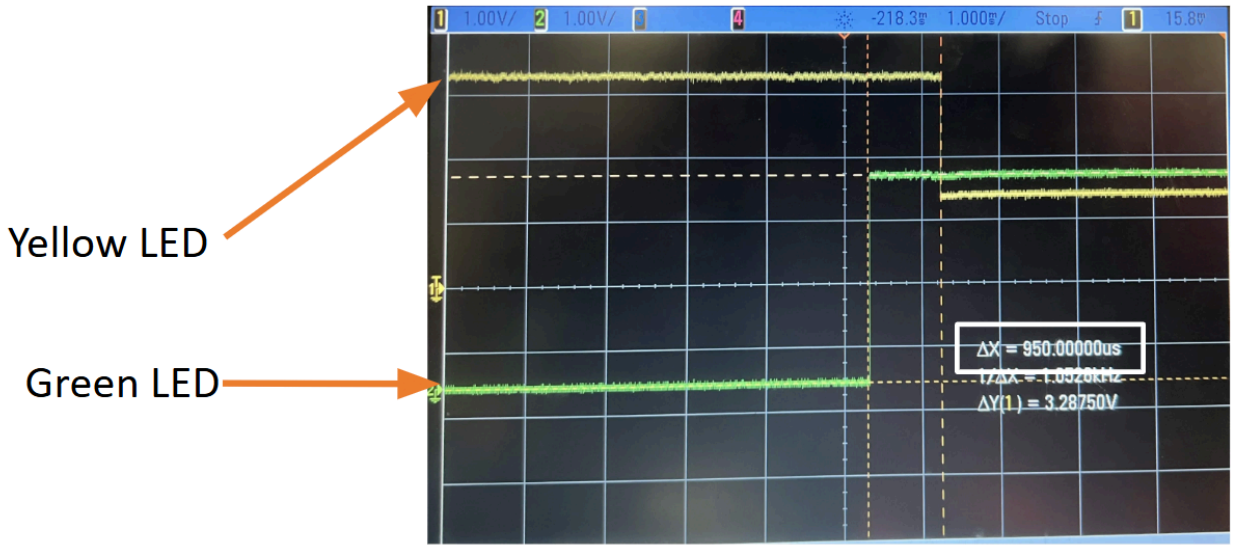

Figure 7: Oscilloscope figure 1

To test keyword detection functionality, we did 30 trials for each keyword the model was trained to recognize. Table 7 in Appendix B of the report shows the detailed data recorded in this session, where "1" means that the device logged the keyword correctly and "0" means that the device failed to log the keyword correctly. The word "water" performed as per the expected accuracy threshold, while keywords like "food" and "dinner" performed poorly. This inconsistency may be due to human variability in voice, and an insufficient number of training samples for complex words with multiple syllables or that are phonetically hard.

4.2 Input processing subsystem

We used an oscilloscope to aid the verification process of checking functionality of our Input subsystem as well. As an example, Figure 8 demonstrates that when the touch sensor is pressed while the accelerometer is not detecting motion, the listening state is not activated. The green LED indicates the idle state while the yellow LED indicates the listening state.

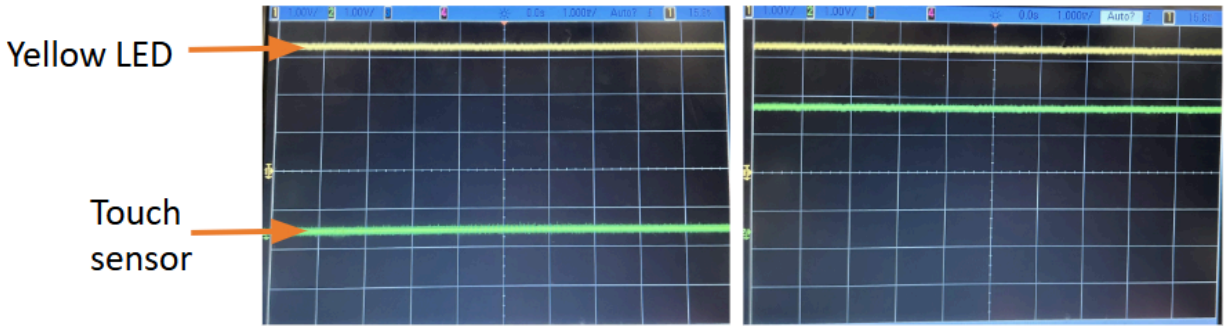

Figure 8: Oscilloscope figure 2

4.3 Output processing subsystem

Figure 7 also verifies the LED reflecting the FSM state without any perceptible delay (950 µs). The response output by the speaker was audible and began within 3 seconds of the user finishing their query action. This behavior is consistent with the goals we set for our output subsystem.

4.4 Power subsystem

The power subsystem was verified using a USB digital power tester while the device was operating, as shown in Figure 9. The tester measured an input voltage of 5.097 V, current draw of 0.071 A, and total power consumption of 0.363 W. This shows that the full system consumed approximately 0.36 W during normal operation. Since the device operated reliably at about 5 V input power, the power subsystem satisfied the requirement of providing stable power to the ESP32 and connected components during testing.

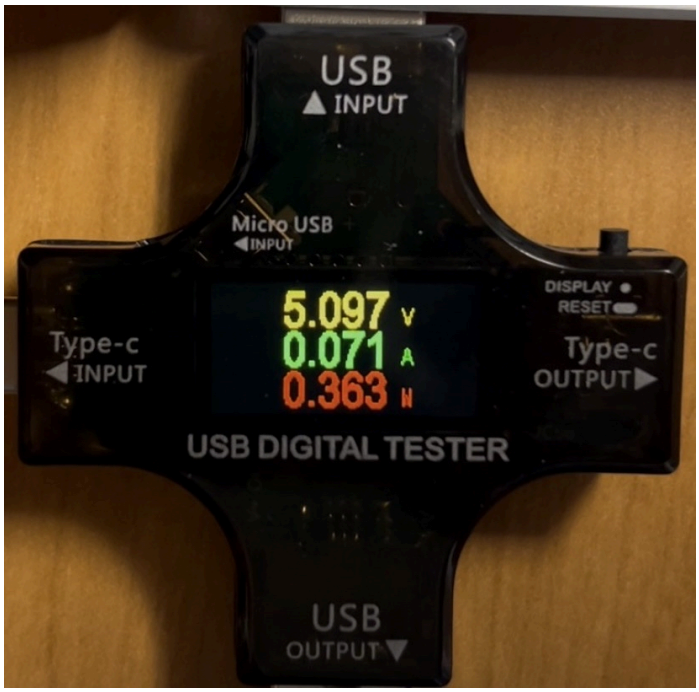

Figure 9: Multimeter measurements of power supply

5. Conclusion

5.1 Accomplishments

We successfully implemented our project idea into a device. We successfully created a keyword detection library and implemented various features, such as timestamps, midnight reset, and more. Although we weren't able to get it working on a PCB due to design errors, the breadboard uses the same devices that would go on the PCB, which means with the correct USB-C orientation on the PCB and a few jumper wires, we could be able to run the code successfully on the PCB for a working a product.

5.2 Future work

Due to time constraints with our projects, we were not able to train and test with as many keywords thoroughly as we wanted. One of the most important things to do in the future would be recording more samples of the keywords and also compacting everything, which would be easier done with a PCB and a smaller box.

5.3 Ethical considerations

Our project prioritizes safety for the user. However, due to the nature of our project involving a microphone, users may be concerned that the device is listening to them continuously, even though it only listens to them while the device is in the listening state. The device casing also has several holes in it. Even though it's large to be worn on the wrist, any water that gets into the device would turn into a hazard for the user. If the casing were to be made smaller, it would be very important to also make the device better sealed with fewer holes for water. Another concern is that users could attempt to rely purely on our device for remembering all their tasks. In a nursing home this would be a big problem since nurses do much more than just feed or remind patients to take their medication. But in a college setting where the user may be alone without a caretaker, solely relying on the device could be dangerous as it's meant to be assistive and not a replacement for remembering tasks.

6. References

[1] “Alzheimer’s Disease Facts and Figures.” *Alzheimer’s Association*, www.alz.org/alzheimers-dementia/facts-figures.

[2] “Keyword spotting - Edge Impulse Documentation,” *Edgeimpulse.com*, 2026. https://docs.edgeimpulse.com/tutorials/end-to-end/keyword-spotting

Appendix A Requirements and Verifications Table

Table 3: System Requirements and Verifications- MCU Subsystem

Requirement	Verification	Verification status (Y or N)
1) Transition between FSM states occurs within 200 ms when a valid state-triggering event occurs under 3.3 V operating voltage.	1. Use an oscilloscope to detect the toggled “debug” GPIO pin that we set high in the code when FSM changes state. 2. Measure the time between input event detection and GPIO state toggle. 3. Repeat for 5 trials. 4. Record mean time for these 5 trials.	Y
2) When capacitive touch is active and the listening window is active, keyword detection should happen within 3 ± 0.5 seconds of receiving the speech input from the user.	1. Activate touch sensor 2. Activate listening window by moving the device 3. Play a pre-recorded sentence with a supported keyword at 60dB. 4. Use an oscilloscope to measure the time from end of speech to the activation of the vibration motor. 5. Repeat for 5 trials. 6. Record mean time for these 5 trials.	Y
3) Store at least 20 activity logs in non-volatile memory. Each log has an activity ID and a timestamp.	1. Sequentially log 20 activities 2. Retrieve logs 3. Confirm timestamps	Y
4) Retrieve and respond to a user query within 3 ± 0.5 seconds.	1. Query the device for an activity performed. 2. Measure time from end-of-speech to first audio output. 3. Repeat for 5 trials. 4. Record mean time.	Y

Table 4: System Requirements and Verifications- Input Subsystem

Requirement	Verification	Verification status (Y or N)
1) Activate listening mode only when the touch sensor output is high and accelerometer detects motion within a validation window	1. Connect a multimeter to touch sensor output 2. While sensor is not touched, check that output is logic low 3. Put finger on touch sensor pad 4. Speak and try to trigger a keyword while accelerometer is laying on a table 5. Check that MCU does not transition to listening stage 6. Speak and trigger a keyword while waving the accelerometer 7. Check that MCU transitions to listening stage	Y
2) Disable microphone after 10 seconds of being in “listening” state without successful logging	1. Activate touch sensor and accelerometer 2. Wait 10 seconds 3. Attempt to trigger keyword while microphone is locked out 4. Check log history that logs were not recorded	Y

Table 5: System Requirements and Verifications- Output Subsystem

Requirement	Verification	Verification status (Y or N)
1) Activate vibration within 2 ± 0.5 seconds of log confirmation.	1. Connect oscilloscope to the vibration motor 2. Trigger a valid log event 3. Toggle GPIO upon log confirmation 4. Measure time between the toggle and first PWM signal on motor driver 5. Repeat for 5 trials 6. Record mean time	Y
2) Provide audible response (≥60 dB) to the user when in a quiet indoor environment.	1. Position a sound level meter about 30 cm away from the speaker 2. Trigger a response from the device by querying 3. Measure SPL during response 4. Record if passes or fails	Y
3) Respond within 3 ± 0.5 seconds of the query.	1. Connect an oscilloscope to the debug GPIO going high on start of replying state 2. Query the device 3. Record time from end of query to first response waveform (I2S) 4. Repeat 5 times 5. Record mean time	Y
4) LED shall reflect the current FSM state without perceptible delay.	1. Connect oscilloscope to LED control 2. Trigger state change 3. Measure the time from the debug GPIO toggle to LED signal change 4. Confirm there is no perceptible delay	Y

Table 6: System Requirements and Verifications- Power Subsystem

Requirement	Verification	Verification status (Y or N)
1) Output voltage of 3.3 V ± 5% under full load.	1. Connect load to 3.3 V rail 2. Use a digital multimeter to measure voltage 3. Set load to different currents (100mA to 500mA) and record measured voltage 4. Ensure all within required output voltage	Y

Appendix B Keyword detection trails and associated data

Table 7: Keyword detection data

Keyword	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27	T28	T29	T30	No. of successful logs	No. of trials	Accuracy
Water	1	1	0	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	24	30	80%
Snack	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	1	0	0	1	11	30	36.66%
Food	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	6	30	20%
Medicine	1	1	0	1	0	1	0	1	0	0	1	1	1	0	0	0	0	1	0	0	1	0	1	1	0	0	0	1	0	0	13	30	43.33%
Dinner	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	9	30	30%
Query	0	0	0	1	1	1	1	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	1	1	14	30	46.67%