

Garage Door Opener with Authentication

Final Project

ECE 631

Aidan Harries

Andy Freshnock

May 10th, 2024

Introduction

Project Scope and Goals

The objective of our project was to design and implement a sophisticated garage door opener system equipped with authentication capabilities using a combination of hardware and software components. This system was intended to enhance security and functionality by allowing door access only to authenticated users, and providing real-time updates on the status of the door and the proximity of a vehicle to the wall within the garage.

Hardware Used

- **Two ESP32 Development Boards:** One interfaced with an ultrasonic sensor and the other with a hall effect sensor to manage different functionalities of the garage door system.
- **Raspberry Pi 4:** Served as the control unit for the NFC card reader and to output sound notifications through a speaker.
- **Digital Ocean MQTT Broker:** Facilitated message communication between hardware components, chosen for its ease of access from different locations.
- **PN532 NFC Card Reader:** Connected to the Raspberry Pi, it authenticated user access based on NFC card scans.
- **49E Hall Effect Sensor and HC-SR04 Ultrasonic Sensor:** These sensors detected the door's status and measured the distance of a vehicle to the wall, respectively.
- **Miscellaneous Components:** Wires for connections, a speaker with an Aux chord for audio feedback, a laptop for observation/programming, and an oscilloscope for testing.








Wiring Tables

ESP32	Name	Ultrasonic Sensor
Pin 1	3.3V	Pin 1 – VCC
Pin 37	GPIO 23	Pin 2 – Trigger
Pin 36	GPIO 22	Pin 3 – Echo
Pin 38 or 32	GND	Pin 4 – GND

Ultrasonic Sensor Wiring Table

ESP32	Name	Hall Effect Sensor
Pin 1	3.3V	Pin 1 – VCC
Pin 38	GND	Pin 2 – GND
Pin 3	GPIO 36	Pin 3 – Vout

Hall Effect Sensor Wiring Table

RPI4	Name		NFC
Pin 15	GPOI 22 Reset		Pin 8 – RST0
Pin 17	3.3V		Pin 5 – Vcc
Pin 19	MOSI – Master Out Slave In		Pin 3 – MOSI
Pin 20	GND		Pin 6 – GND
Pin 21	MISO – Master In Slave Out		Pin 2 – MISO
Pin 23	SCLK – Serial Clock		Pin 1 – SCLK
Pin 24	CE0 - Chip Select		Pin 4 – SS

NFC Wiring Table

Software Used

- **C Language:** Utilized to program the ESP32 microcontrollers, handling sensor data and controlling device functionalities.
- **Python:** Used to program the Raspberry Pi, mainly for handling NFC authentication and sound playback.
- **Arduino Development Environment:** This platform was used for writing, debugging, and uploading the C code to the ESP32 microcontrollers.
- **MQTT Library:** Enabled the ESP32s to communicate with the Digital Ocean MQTT broker.
- **Freeboard Dashboard:** A web-based dashboard that displayed real-time information from the various components of our system.
- **NFC Library (PN532):** Integrated into our software to support the NFC card reading processes.
- **PuTTY and HiveMQ:** Used for secure SSH communication with the Raspberry Pi and initial MQTT communication tests, respectively.

This system was designed to meet specific tasks, including authenticating users via NFC, communicating authentication results, indicating door status, measuring proximity, and controlling door operations, all managed through a remotely accessible dashboard.

Design Partitioning

Our project design was strategically partitioned into hardware, software, and functional sections, each addressing specific needs of the garage door opener system.

Hardware Partitions:

1. **ESP32 Development Boards:** Two boards were utilized where one interfaced with the ultrasonic sensor to measure the distance and the other with the hall effect sensor to detect the door's status.
2. **Raspberry Pi 4:** Managed the NFC authentication and sound notifications using a connected speaker.
3. **Sensors and Additional Components:** Included a PN532 NFC card reader, 49E Hall Effect Sensor, HC-SR04 Ultrasonic Sensor, speaker, wires, and an oscilloscope for setup and testing.

Software Partitions:

1. **Setup:** Initialization of WiFi, MQTT connections, and pin configurations for each sensor and the ESP32s.
2. **Main Loop:**
 - **Ultrasonic Sensor:** Managed the distance calculations using a moving average filter and controlled the timing and intensity of an LED based on the distance.
 - **Hall Effect Sensor:** Monitored the door's status and communicated changes using MQTT messages.
 - **NFC Reader (Raspberry Pi):** Executed the authentication process and handled access control, with sound notifications and MQTT message updates regarding the authentication status.
3. **Interrupt Service Routine (ISR):** For the ultrasonic sensor to handle echo pin changes, calculate echo times, and update the moving average for distance calculations.

Functional Partitions:

- **Authentication via NFC:** Validates user access with the NFC card reader.
- **Distance Measurement:** Uses the ultrasonic sensor to determine the distance of a vehicle to the wall.

- **Door Status Monitoring:** Utilizes the hall effect sensor to provide real-time door status.
- **Communication and Notification:** Involves sending data over MQTT and updating the Freeboard dashboard. It includes the following panels:
 - **NFC Authentication Feedback:** Displays images and messages indicating access granted or denied. For example, a positive authentication shows a welcoming message with a friendly image, while a denial displays a warning.
 - **Distance Measurement Visualization:** A panel shows traffic light colors (green, yellow, red) to represent the distance of the car from the wall, aiding in parking.
 - **Actual Distance Readout:** Provides numerical distance readings from the ultrasonic sensor in inches, offering precise feedback.
 - **Door Status Indicators:** Shows the current status of the door (open, closed, opening, closing) in real-time.

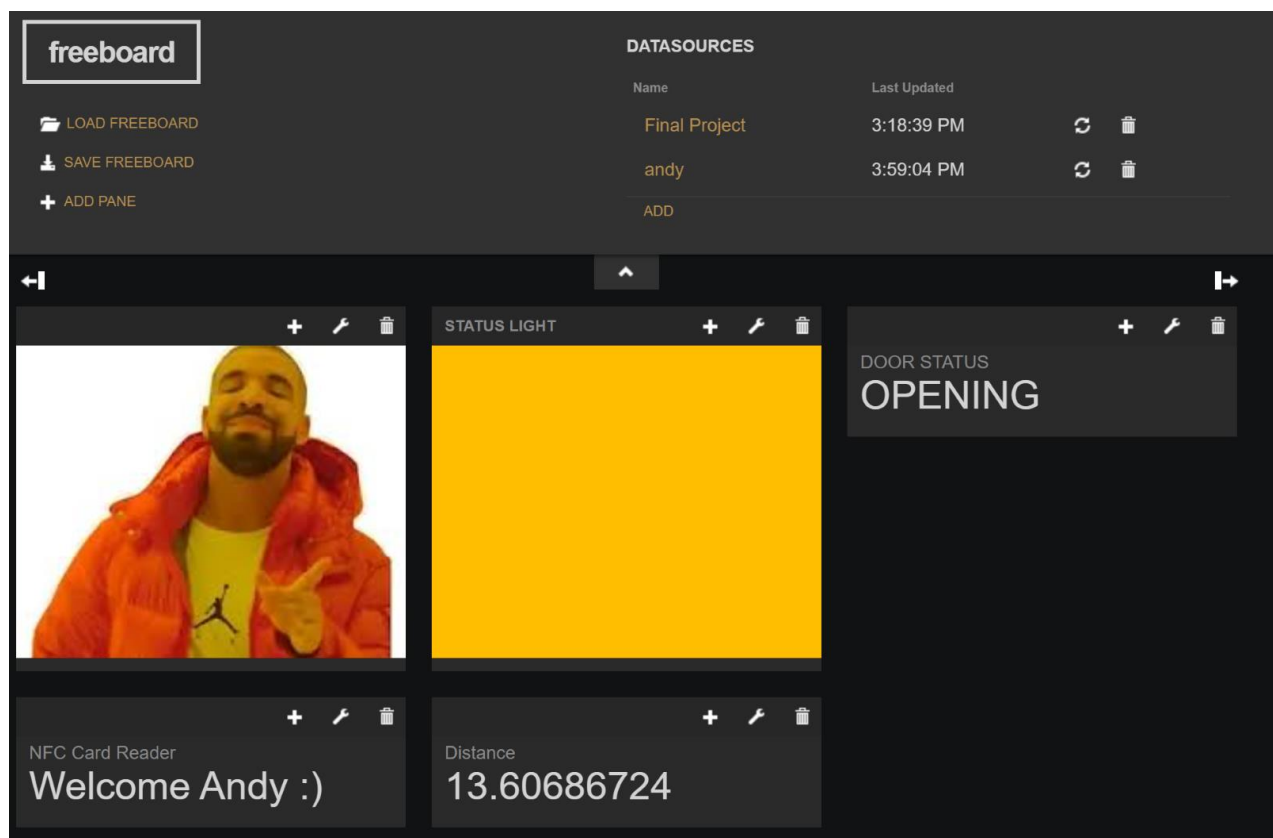


Figure 1 – Freeboard Layout

Rationale for Partitioning

The design of our garage door opener system was partitioned to maximize each component's strengths, ensuring reliable operation, optimal resource use, and easier maintenance. Here's a explanation of our rationale:

- **Separate ESP32s for Distinct Tasks:** We utilized two ESP32 microcontrollers, each handling specific tasks—distance measurement and door status monitoring. This division minimizes task interference, enhances performance by allocating full computational resources to each task, and simplifies maintenance by isolating functionalities, making troubleshooting more straightforward.
- **Utilizing Raspberry Pi for NFC Authentication and Sound Management:** The Raspberry Pi was chosen mainly for its auxiliary port, allowing us to connect a speaker for audio feedback during NFC authentication. This feature enhances user interaction by providing audio cues based on authentication results. Our familiarity with the Raspberry Pi from previous labs also influenced this decision, leveraging our prior experience for efficient implementation.
- **Software Partitioning:** We aligned our software setup with the hardware configuration to optimize functionality:
 - **Dedicated Setup Routines:** Each microcontroller runs setup routines specific to the peripherals it manages, streamlining initialization and reducing setup conflicts.
 - **Specialized Main Loops and Interrupts:** By tailoring main loops and interrupts to specific tasks, each microcontroller operates efficiently with minimal delay, directly supporting their dedicated functionalities.
- **Functional Specialization for Communication:** Using MQTT for communication between controllers and the dashboard allows for reliable data handling and system scalability. MQTT's lightweight protocol is ideal for real-time updates, important for maintaining system responsiveness and facilitating possible future integration with other IoT systems.

This approach leverages the technical capabilities of our selected hardware and adheres to best practices for creating a robust, efficient, and maintainable system.

Discussion on Design Decisions

In our project, we made several decisions to enhance the functionality and reliability of our garage door opener system:

- **Moving Average Filter:** The moving average filter is applied to the distance readings stored in the circular buffer. This filter smooths out the sensor data, reducing the impact of anomalies and short-term fluctuations. It's particularly useful in environments where the sensor might pick up erroneous signals due to environmental factors, making sure the system reacts only to genuine changes in distance.

```
distanceOutput = (windowSum / (float)WINDOW_SIZE) * 0.0135039; // calc avg and inches
```

- **PWM (Pulse Width Modulation):** We used PWM to control the timing and operation of the ultrasonic sensor. By adjusting the duty cycle, we can control the sensor's pulse emission, which is necessary for initiating the distance measurement process. This method provides us with control over the sensor's operational parameters, allowing for reasonably consistent and reliable readings.

```
void setup() {  
  //mqtt  
  setup_wifi();  
  client.setServer(mqtt_server, 1883);  
  // put your setup code here, to run once:  
  ledcSetup(PWM_CHANNEL, PWM_FREQ, PWM_RES);  
  ledcAttachPin(TRIGPIN, PWM_CHANNEL);  
  ledcWrite(PWM_CHANNEL, PWM_DUTY);  
  attachInterrupt(digitalPinToInterrupt(ECHOPIN), EchoInterrupt, CHANGE);  
  Serial.begin(115200);  
  
  pinMode(LEDBLUE, OUTPUT);  
  digitalWrite(LEDBLUE, HIGH);  
}
```

- **Use of a Circular Buffer:** We implemented a circular buffer to manage the stream of distance data from the ultrasonic sensor. This approach allows us to continuously update the data while using a fixed amount of memory. It helps keep the most recent readings and discards the oldest ones automatically, which is important for the moving average calculations that follow.

- **ISR (Interrupt Service Routine):** We configured an ISR to handle the echo responses from the ultrasonic sensor efficiently. The ISR is triggered by the change in the echo pin state, capturing the precise time when the echo is detected. This setup allows for immediate processing, which is critical for calculating accurate distances based on the time-of-flight of the ultrasonic pulses.

```
void EchoInterrupt(){
    //do things
    uint64_t now = micros();
    int trigger = digitalRead(ECHOPIN);
    if(trigger == 1 ){
        OldMikes = now %TIMECLAMP;
    }else{ // trigger = 0
        CurrentMikes = now % TIMECLAMP;
        uint64_t echoTime = ( CurrentMikes - OldMikes + TIMECLAMP) % TIMECLAMP;
        windowSum -= windowAvg>windowIndex]; // subtract old value
        windowSum += echoTime; // add new value
        windowAvg>windowIndex] = echoTime; // add new value to array
        windowIndex= (windowIndex+1) % WINDOW_SIZE;
    }
}
```

These decisions were driven by the need to balance system performance with the hardware and software capabilities available to us, making sure the garage door opener operates reliably and accurately in real-world conditions.

Design Issues / Limitations

During the development of our garage door opener system, we encountered a few challenges, particularly with the audio output functionality. Here is an overview of the issues we faced and the solutions we implemented:

Bluetooth Connectivity Problems: Initially, we planned to use Bluetooth for playing sounds through a connected speaker to provide audio feedback during NFC authentication. We successfully established a connection between the Raspberry Pi and the speaker, but the connection was unstable and would drop frequently. Despite extensive troubleshooting and consulting various online forums for potential solutions, we could not determine the missing libraries or configurations needed to stabilize the Bluetooth connection.

Solution - Switch to AUX Connection: Due to the persistent issues with Bluetooth, we decided to switch to using the Raspberry Pi's auxiliary port for audio output. This change proved to be more reliable for our needs. The AUX connection was straightforward to set up and provided a stable link to the speaker without the connectivity issues we experienced with Bluetooth.

Limited Speaker Output Power: After switching to the AUX connection, we noticed that the audio output from the speaker was quieter than expected. It appears that the Raspberry Pi does not supply sufficient power to fully drive the speaker at higher volumes. Even at maximum volume settings, the sound output remains somewhat muted, which is less than ideal for clear audio feedback in noisier environments.

Workaround: While the lower volume is a limitation, the audio output is still functional and meets the basic requirements of our project. We made sure that the system's critical audio cues could still be heard in a typical garage setting. For future improvements, we might consider integrating an external amplifier to boost the audio output or selecting a speaker with lower power requirements that is more compatible with the Raspberry Pi's output capabilities.

Testing

In our project, we conducted a series of tests to verify the functionality and reliability of each component. These tests were designed to ensure that the system performs as intended and to identify any areas for improvement. Here's a comprehensive overview of the tests we performed:

PWM Test

Procedure: We connected the ESP32 that controls the ultrasonic sensor to an oscilloscope to monitor the PWM signal. This test aimed to verify the waveform characteristics and measure the width of the PWM output used to trigger the ultrasonic sensor.

Results: The oscilloscope confirmed that the PWM signal was stable and matched the expected frequency and duty cycle set in our code. The width of the PWM signal correlates to the time measurement used to calculate the distance by the ultrasonic sensor, confirming the correct operation of our triggering mechanism.

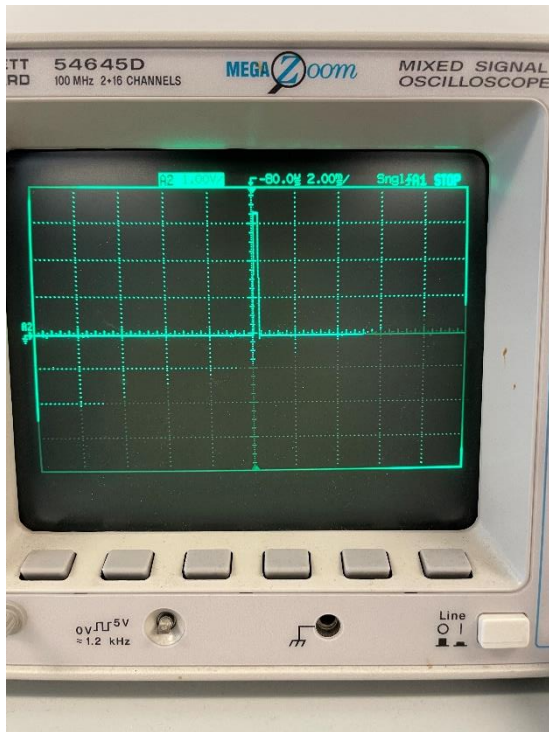


Figure 2 – Input PWM

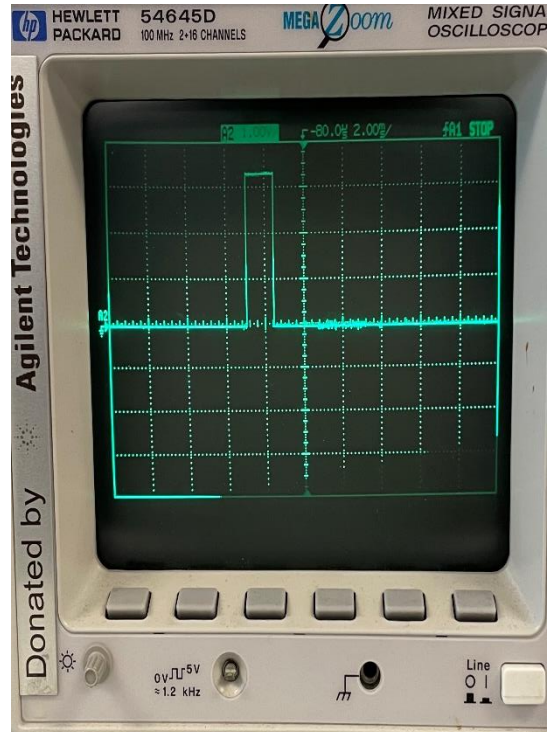


Figure 3 – Output Signal

Ping Sensor Distance Test

Procedure: We placed the ultrasonic sensor at a fixed position and tested it against a box placed at marked distances on a table (0", 5", 10", 15", ... , 35"). Each distance was measured twice to check for consistency.

Results:

Test 1:

0 in: 1.33 in
 5 in: 10.45 in
 10 in: 19.42 in
 15 in: 28.87 in
 20 in+: No stable reading

Test 2:

0 in: 1.32 in
 5 in: 9.98 in
 10 in: 18.9 in
 15 in+: No stable reading

The results showed consistent discrepancies, with measurements nearly doubling the actual distances. This issue indicates potential calibration or software errors affecting sensor accuracy.

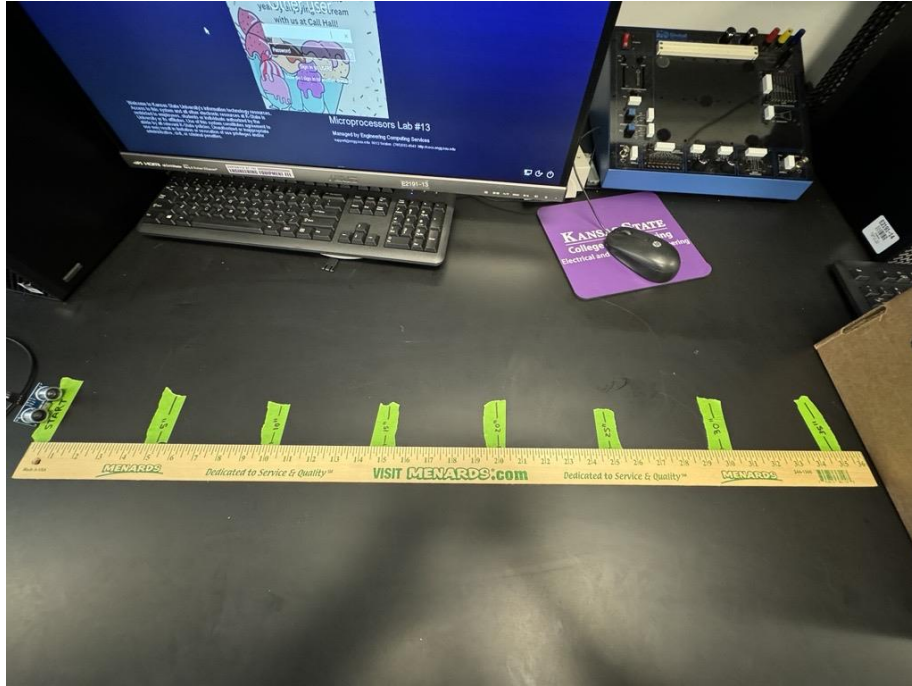


Figure 4 – Ping Sensor Distance Testing Setup

NFC Authentication Test

Procedure: We validated the NFC system using different NFC cards. Andy's card was set as the "authorized" card, triggering positive feedback through the speaker and dashboard, while Aidan's and other classmate's cards were used to test unauthorized access.

Results: The system reliably recognized Andy's card and granted access, while correctly denying access with other cards. The audio and dashboard notifications corresponded appropriately to each card's authorization status.

Hall Effect Sensor Test

Procedure: We connected the Hall Effect sensor to an ESP32 and tested it by altering the magnetic field to simulate different door statuses: open, closed, opening, and closing.

Results: The sensor outputs matched the expected states based on the magnetic field changes, confirming that the sensor could accurately detect and report the door's status.

Blinking LED Test

Procedure: The LED on the ESP32 with the ultrasonic sensor was set up to blink at varying speeds based on the distance measurement. The closer the object, the faster the LED would blink. We tested this by approaching the sensor with an object and observing the change in blink rate.

Results: The LED blinked faster as the object got closer to the sensor, demonstrating that the LED's blink rate accurately reflects changes in distance.

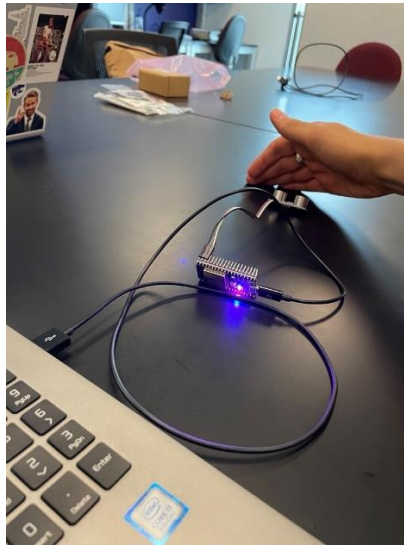


Figure 5 – Blinking LED Test

Analysis and Future Improvements

The PWM and NFC tests met expectations, demonstrating that these components function correctly. However, the distance measurement tests revealed significant accuracy issues. Future improvements will focus on recalibrating the sensor or adjusting the software algorithm to enhance measurement accuracy. Potential changes could include adjusting the number of samples used for averaging or reviewing the conversion formula from time to distance.

This comprehensive testing approach has verified that while some components perform well, others, particularly the ultrasonic sensor, require adjustments to meet the project's accuracy requirements. This testing not only confirms functionality but also highlights areas for enhancement, showing the system can be refined for better performance.

Conclusion

Throughout the course of our final project for ECE 631, we designed and implemented a sophisticated garage door opener system with authentication features, using a variety of hardware and software components. This system was made to enhance security by allowing door access only to authenticated users and providing real-time updates on door status and vehicle proximity.

Our project successfully integrated two ESP32 development boards, a Raspberry Pi 4, various sensors, and communication protocols to create a functional and interactive system. The Raspberry Pi handled NFC authentication and sound notifications effectively, while the ESP32s managed distance measurements and door status monitoring. We also implemented a Freeboard dashboard that displayed system statuses clearly and in real-time, improving user interaction and control.

One significant challenge was the initial attempt to use Bluetooth for audio feedback, which faced stability issues. We switched to a more reliable AUX connection, although it presented limitations in sound volume. Additionally, our distance measurement tests highlighted inaccuracies in sensor output, suggesting the need for recalibration or software adjustments.

Our testing process covered the PWM signal integrity, ping sensor distance measurements, NFC authentication effectiveness, door status accuracy, and the functionality of the blinking LED indicator. Each component was tested to make sure it performed as expected under various conditions. These tests confirmed the reliability of most system parts, though they also revealed areas needing refinement, particularly in distance measurement accuracy.

To improve our project, we would focus on enhancing the ultrasonic sensor's accuracy by adjusting the sampling method or revising the distance calculation algorithm. Additionally, integrating an external amplifier could resolve the audio output limitation, allowing for clearer feedback in various environments.

In conclusion, this project gave us good hands-on experience in designing and troubleshooting a comprehensive IoT system. The challenges encountered and the knowledge gained from this project will be beneficial in our future endeavors in electronics and system integration.

Appendix

Code for this project can be found here:

<https://gitlab.beocat.ksu.edu/s24.the-a-team/finalproject>