FIRE AND GAS DETECTION SYSTEM WITH REAL TIME LED NAVIAGTION

By

Abel Garcia

Alex Parafinczuk

Jainam Shah

Final Report for ECE 445, Senior Design, Spring 2025

TA: Surya Vasanth

07 May 2025

Project No. 81

Abstract

Our project presents a Fire and Gas Detection System integrated with a real-time LED-based navigation mechanism, engineered to identify the safest exit route within a one-story residential house. The system incorporates temperature sensors, a microcontroller, a digitally actuated servo-driven vent, and a web application for remote data visualization and storage using Firebase. Sensor data is transmitted to Firebase, where it dynamically updates LED indicators to guide evacuation paths and commands vent closures upon fire detection to inhibit fire propagation. System testing demonstrated high accuracy in both exit determination and vent actuation. With optimized power efficiency and reliable performance, this system provides an effective solution for enhancing residential fire safety.

Contents

1. Introduction

1.1 Problem

Fires, whether accidental or natural, can cause severe damage to homes and the inhabitants. Today, fire and gas systems only trigger alarms and notify emergency services when a hazard is detected. While this is good, the emergency response time isn't immediate, and a lot can happen during this time. Another factor that we must consider is that residents can panic or become disoriented during these hazardous emergencies leading to confusion during precious seconds which could be used for getting outside to safety. So, a need exists for a system that could both mitigate the damage done as well as guide individuals to the safest exit to minimize confusion.

1.2 Solution

Our solution to this problem is a Fire and Gas Detection System with Real-Time LED Navigation. Our project aims to detect fires using temperature sensors and a microcontroller. This means that we will have one main board in a central area of the house, then, there will be mini boards which will only house a temperature sensor across all the other rooms. Using Firebase-based web applications as the main communication method with ESP32 via wi-fi, we will guide users to the nearest exit with LED indicators on each board all controlled by an algorithm that will give individuals the best exit to take for safety in the presence of a fire. All of this is done by users inputting their floorplan manually and providing exits, doors, and rooms for the software to understand the general locations of all the sensors to correctly light up when the time arrives. In the case of a fire, we will use a motorized vent cover which will automatically close when a fire has started, this is to suffocate the oxygen flow for the fire and buy time for first responders to arrive and handle the situation. It has been shown that the main growth of fire is due to a source, or the mere presence of oxygen accelerating the speed and growth of the fire. This project is designed for low power, quick response, and ease of use for all homeowners to be able to use.

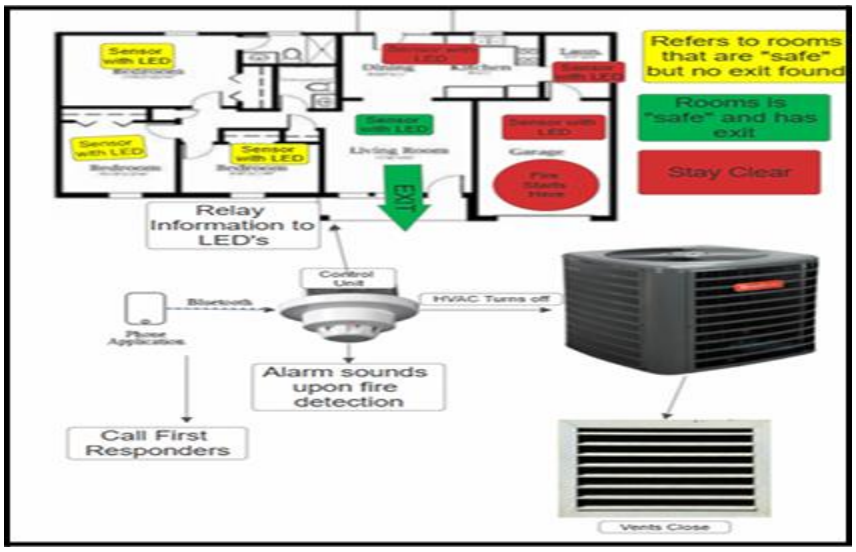

Figure 1: Visual Representation of the System

1.3 High Level Requirements

For this project to be considered successful, we have set some high-level criteria that our Fire and Gas Detection system needs to meet. These are:

1. The web-based application will consist of a place where homeowners can lay out their floor plan by designating rooms with connections such as doors, and this will lead to other areas in the house along with the location of sensors and exits. The algorithm will isolate rooms that are affected by the fire and designate an exit that is furthest from the fire, and this will be visible on the LEDs with guaranteed accuracy.
2. Once a threshold of 90°C is detected on the temperature sensors, a fire is present, and a signal is sent to the control unit which will trigger the algorithm to run and shine LED's along with an alarm sounding. The gas sensor will sound the alarm when a Carbon Monoxide (CO) level greater than 20 ppm is present.
3. When a fire is detected, the HVAC will turn off, following this, the vents will receive the signal to close, limiting the oxygen flow to the fire. Upon clearance of fire, operation of the HVAC and vents will function as normal.

These high-level requirements collectively serve as our goal for our design. Following this, we aim to show each subsystem and the design choices and considerations that were involved in the process of reaching the goals mentioned above. After analyzing the design of the whole project and the subsystems, we will give a summary of the costs and schedule that were involved in the making of the project, and finally, we will wrap up the paper by talking about uncertainties and what could be done in the future with this project. Overall, we believe this project to be innovative in what it attempts to do and believe that the project has a lot of potential use in replacing modern Fire and Gas Detection systems.

2 Design

2.1 Subsystem Overview

As shown in our block diagram below, Figure 2, our project consists of multiple subsystems. These subsystems are as follows: Power Supply, Control system, Vent/Motor, Sensing, and application.

The power supply system consists of three 9V batteries as the source, one is regulated to 7V, the other is regulated to 5V, and the last one is regulated to 3.3V which is then regulated to 1.5V. The sensing subsystem is the system in charge of collecting environmental data for us, in particular, temperature data and gas data. The sensing subsystem also houses our LEDs which only shine in the presence of a fire. The vent subsystem consists of an HVAC (fan), a motor, and a vent. The ESP32 will control the motor with a PWM signal to indicate whether it should open or close. The application subsystem is where users will be able to manually input their floorplan to the system, the application subsystem is also in charge of running the algorithm to then give the information on the relevant LEDs which should light up to the ESP32. Finally, we have the control subsystem, the ESP32 is the main key in this system as it oversees sending all the relevant signals in the case of a fire, as well as constantly receiving data from all the sensors. These subsystems combine to form a unit that enables efficient monitoring and a safer alternative for homeowners whenever a fire is present.

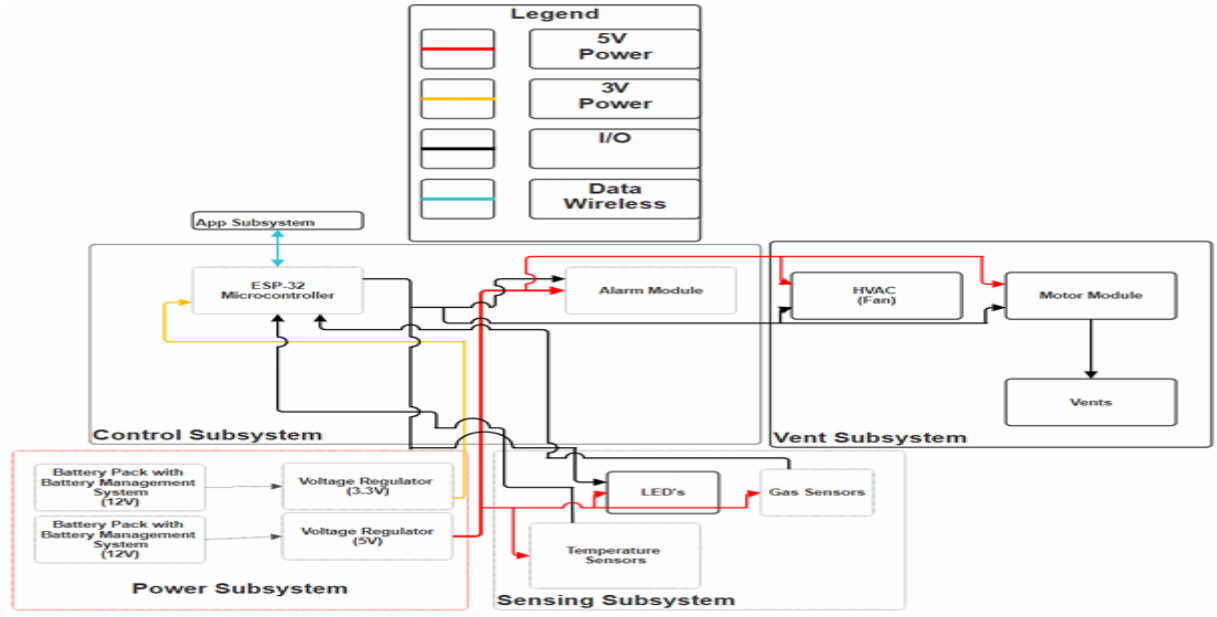

Figure 2: High-Level Block Diagram of the Project

2.2 Physical Design

To test the project, we designed a 3D-floor layout using blender. The reason for this is because the application expects you to have a floor plan, so, to test the algorithm in real-world application, we replicated a mini floor plan of a basic single-story house. The layout is simple; it consists of a main room in the center which will house our main PCB board. The other rooms will house our temperature boards, and there is a cutout for each board to place snuggly into the house. There is a total of 2 exits in this design to test the algorithm with as shown in Figure 3.

Figure 3: The design of the housing used for each PCB board

We also had the ECE Machine design a motorized vent for us. We ordered a vent and a $25kg$ digital motor to serve as the arm which will rotate the vent cover to close and open depending on the given situation. In Figure 4 below, you can see that the design is simple and would integrate well into homes with no issues, the only connection required is the motor connections.

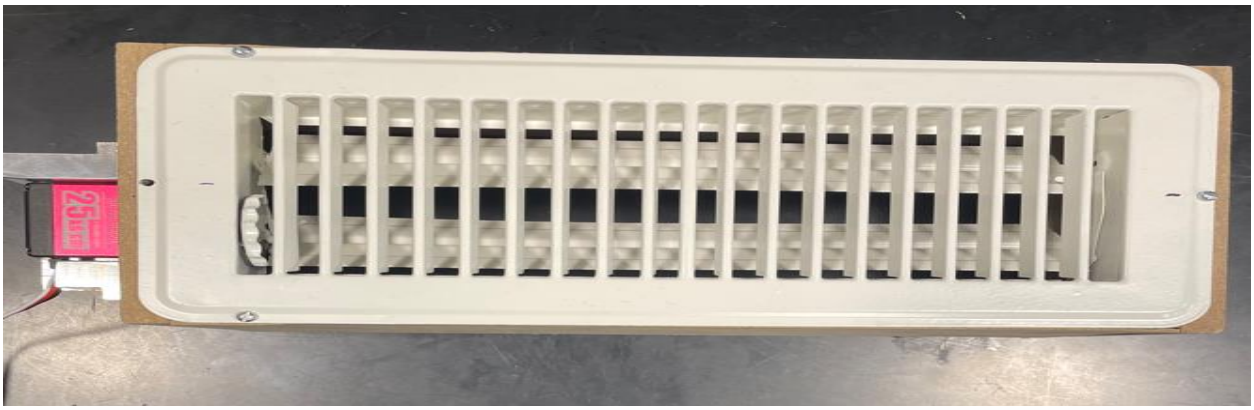

Figure 4: Motorized Vent Cover

2.3 PCB Layout

When designing the PCB, the main objective is to provide a compact, modular, and testable board. Having a compact board is useful in reducing noise, especially in cases like our design where we have sensor data that we want to read accurately. In our layout, we tried to keep subsystems together so that we knew the general area of all our subsystems. We created two designs for our PCB boards, one was our main PCB board, and this board consisted of every subsystem. The other PCB board is the temperature board which only consists of the sensing subsystem. For the main PCB board, we wanted to make sure that we could test make this board first and test its functionality. The reason for this is because if we know the temperature sensor works for the main board, then it would work for the temperature board, the only thing we must add are several connectors to ensure that we have

connections with all the temperature boards to the main board. Test points are crucial and are scattered all around the boards, you can see these test points in Figure 4 in areas such as our power subsystem on the bottom left, with test points such as 5V, 3.3V, etc., and this is important in troubleshooting the board as regulators can stop regulating if there is too much of a load present. The schematics and a 3D model of our PCB boards can be found in the appendix, Figures B.1 Figures B.2, Figures B.3, Figures B.4 respectively.

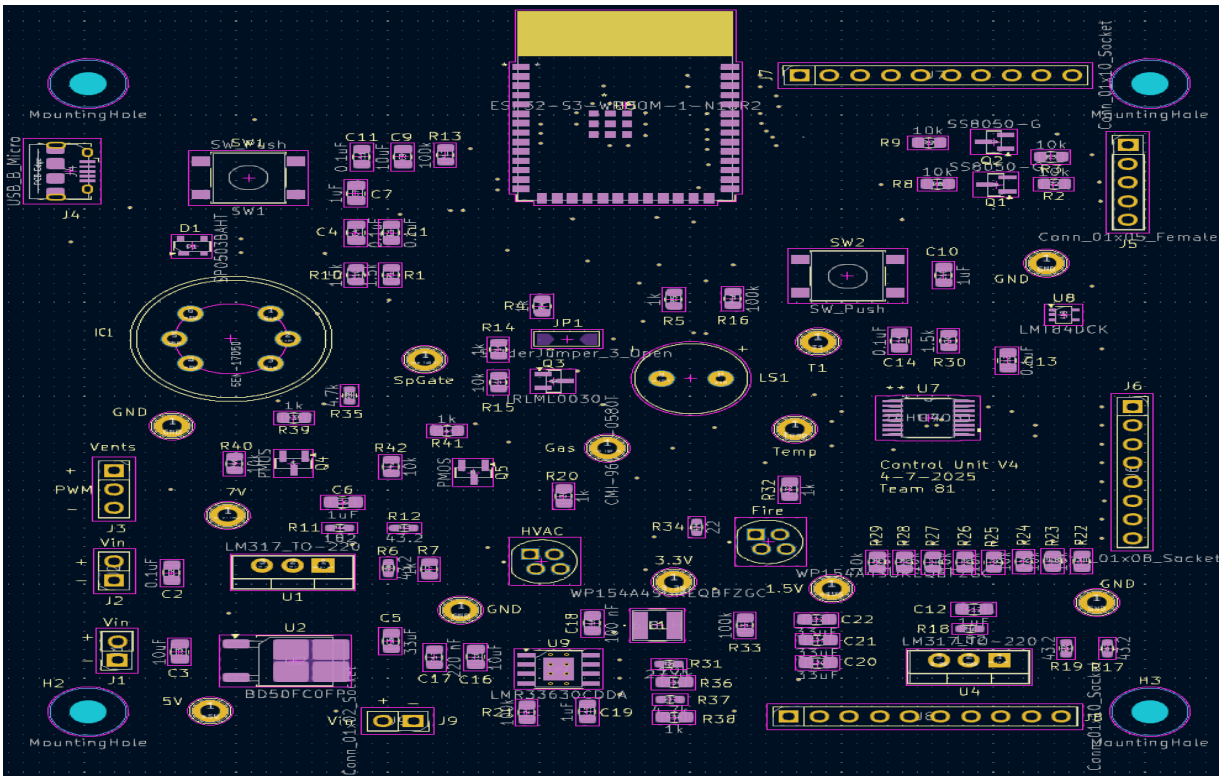

Figure 5: Main PCB Board Layout

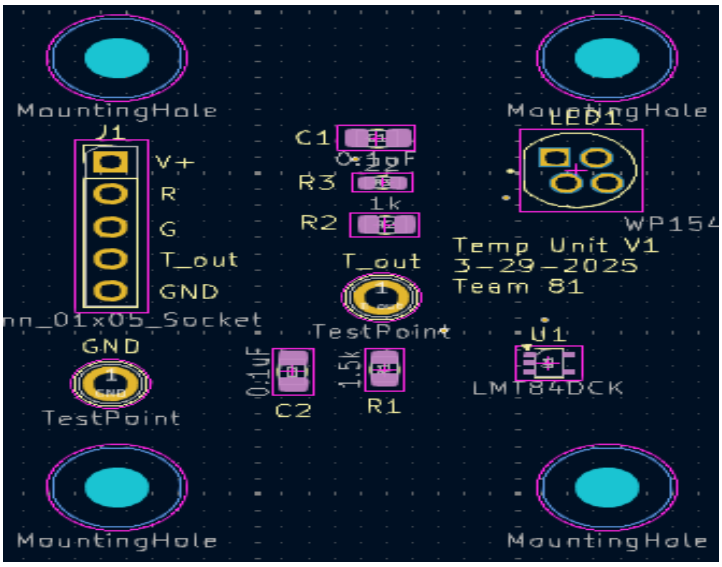

Figure 6: Temperature PCB Board Layout

2.4 Power Supply Subsystem

2.4.1 Design Decisions

The power supply subsystem is responsible for converting a 9V input from a battery to four different distinct voltage levels. These levels correspond to 7V, 5V, 3.3V, and 1.5V. This requirement is fulfilled by using three 9V, 1500mAh Lithium-Ion battery with two LD1085 Low Dropout (LDO) regulators for our 7V and 1.5V rails. This LDO features a low dropout voltage compared to the LM317 which has a much higher dropout voltage. The BD50FC0FP was chosen as our 5V voltage regulator, and this is a fixed output regulator which is guaranteed to regulate loads under 3A. For the 3.3V rail, we decided to go for a buck converter design for more efficiency and allow more current to be drawn.

2.4.2 Design Details

For our battery, the maximum current draw is 500mA, so this means the batteries we have chosen can sustain our project for 3 hours. Regularly, this amount of current is not common as this only appears once the motor opens the vent, so this ensures consistent power delivery to all the components.

In our design, we have our buck converter in series with our LDO, this will be the 3.3V rail in series with the 1.5V. The fixed regulator and the last LDO have their own battery, we decided that the 3.3V has the buck converter, so we can afford to place an LDO in series with it to preserve battery usage. Initially, we used a LM317 to output 1.5V, but the dropout voltage is usually high, typical values of 1.5V-2.5V, and this value only increases when there is more load present in the circuit. So, when we tested with no load, it ran fine, but then, we switched to 100mA load, and it failed to regulate a constant 1.5V. Our switch to the LD1085 proved to be the best move as it has the same footprint as the LM317 but has a maximum dropout voltage of 1.5V at 3A which guarantees that it will always regulate our system. All the other regulators and buck converters used in the project can also handle a max load of 3A, so, because our current demand is never going to reach that high, we can guarantee functionality. The maximum current found during operation was 500mA.

To ensure stability, capacitors are placed on both the input and output of all the regulators and buck converters used in the project board. These capacitors help smooth voltage fluctuations and maintain consistent operation during periods of high current draw (ESP32 during wi-fi operation). They also help reduce noise and ripple which is critical for the proper functioning of sensitive components like the ESP32. And with this design, we will provide a reliable, and safe power supply for the entire system.

2.4.3 Design Verification

Verification of the power supply subsystem was conducted using a multimeter to measure the output voltage produced by our regulators and buck converter, and the input of the voltage making sure the batteries were at 9V as designed. Then, powering on the board, we observed with no load, the voltage outputs across our regulators and converters were all as specified, 7V, 5V, 3.3V, and 1.5V. Finally, we did a load test, and the results can be seen below in Table 1 confirming that our power supply subsystem is fully functional:

Table 1: Verification Test of Power Supply

Current (Load)	7V LDO	5V Fixed Regulator	3.3V Buck	1.5V LDO

0mA	7.056V	5.041V	3.309V	1.51V
100mA	7.034V	5.018V	3.305V	1.503
200mA	7.020V	5.012V	3.302V	1.494V

2.5 Control Subsystem

2.5.1 Design Decisions

Our project aimed to develop a smart application that allows users to store a digital floor plan and, in the event of a fire, execute an algorithm that identifies the safest evacuation route using real-time LED indicators. To support this functionality, we required a microcontroller capable of Wi-Fi communication to enable bidirectional data exchange with a remote server, as well as onboard control capabilities such as PWM signal generation for managing vent actuation.

After evaluating several options, we selected the ESP32-S3-WROOM module, which had been introduced to us during lecture and was readily available at the university. This module proved to be a strong match for our system requirements. We chose Wi-Fi over Bluetooth due to its superior range, higher bandwidth, and better integration with web services such as Firebase. With realistic data transfer rates between 4–10 Mbps, Wi-Fi allowed us to reliably transmit sensor data to the cloud and maintain consistent, responsive communication.

The ESP32 also offers extensive library support, including WiFi.h and Firebase_ESP_Client, which significantly simplified development and integration. These resources enabled us to efficiently implement the system's core functionalities with reliability and scalability in mind.

For the alarm system, we opted for a straightforward yet effective solution: a 5V piezoelectric buzzer. This component activates when supplied with 5V and emits an 80 dB sound, ensuring the alarm is clearly audible throughout the household in emergency scenarios. While not a sensor, the buzzer functions as a key component of the alert subsystem, providing immediate feedback in the event of fire detection.

2.5.2 Design Details

To ensure the buzzer alarm remains inactive during normal operation and only activates in emergency scenarios, we implemented control via an N-Channel MOSFET (IRLZ34NPBF-ND). In our design, the MOSFET's drain is connected to the ground pin of the buzzer, while the gate is driven by the ESP32's GPIO. When the ESP32 outputs a HIGH signal, the MOSFET enters saturation mode, completing the circuit and allowing 5V to flow through the buzzer, thereby activating the alarm.

For the ESP32 hardware configuration, we followed the reference design provided on our course website's wiki. This design supported two programming interfaces: USB and UART. We opted for UART programming using a USB-to-UART bridge, which allowed us to program the ESP32 without incorporating a dedicated programming circuit. The flashing process involved manually placing the ESP32 in boot mode by first holding down the GPIO0 (boot) button, then pressing the EN (reset) button. Once in boot mode, we could release GPIO0 and upload our code via the UART interface.

The ESP32 serves as the central controller for multiple subsystems, performing the following critical tasks:

1. Alarm Control: Activates the 5V buzzer when a fire is detected by any temperature sensor.
2. Temperature Sensor Multiplexer: Interfaces with an 8:1 analog multiplexer to sequentially read values from eight temperature sensors which are distributed across the main board and peripheral sensor boards. The ESP32 can cycle through all sensors in under 5 milliseconds, a limit set by the ESP32's ADC sampling, not the multiplexer.
3. Heater Control for Gas Sensor: Drives a PMOS-based switching circuit to periodically power the heater element of the gas sensor, enabling accurate gas detection.
4. Gas Sensor Reading: Reads the analog gas concentration output after heater activation.
5. Ventilation Control: Sends PWM signals to a digital servo motor to rotate its arm by 180°, opening or closing the ventilation duct based on fire presence.
6. Data Transmission: Sends all collected sensor data to Firebase for storage and real-time monitoring.
7. LED Navigation Feedback: Receives LED control commands from Firebase, including which LEDs to illuminate and what color they should display, to indicate safe evacuation routes.

Each of these tasks was developed and tested incrementally. We followed a modular development approach, writing and validating code for one subsystem at a time. Only after successful testing would we integrate additional components, ensuring the overall reliability and maintainability of the control subsystem.

2.5.3 Design Verification

To verify our control subsystem, we must verify the functionality of the ESP32 and the Buzzer Alarm. The verification of the buzzer alarm is straightforward, we grabbed a breadboard, connected the positive leg to a power source of 5V, and grounded the other side, as soon as this connection is made, the alarm will sound. Then, using the apple watch built-in Noise app, or any other form of detecting noise, we measured the sound and found it to be 80dB as intended.

For ESP32, there are more steps to follow. First, you must program the circuit, then, you must connect the ESP32 to Wi-Fi to allow it to interact with Firebase. The following image below will show the following: First, ESP32 is able to read data from sensors and upload it to Firebase to store it. Next, you will also see that there are signals labeled "False" which indicate that a fire has not been detected yet by our temperature sensors, so there is no need to close the vents or trigger the alarm. For a picture showing functionality of the ESP32 giving signals to LEDs, you can refer to Figure B.7 which shows the functionality of our algorithm as well as our LEDs.

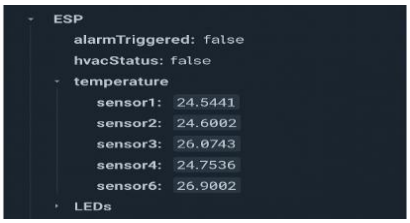

Figure 7: Verification of ESP32

2.6 Sensor Subsystem

2.6.1 Design Decisions

The sensor subsystem is responsible for collecting temperature readings from every room in the user's house as well as carbon monoxide concentrations from the main control board and sending these readings to the ESP32 in the control subsystem. The temperature sensor used is the LMT84DCKR, which has an operating range of -50 ℃ to 150 ℃ and an inverse relationship between the output voltage and detected temperature. This covers the expected range of temperatures for our project from room temperature to fire detection at 90 ℃ as well as ensuring protection for our ESP32 as the voltage output from the sensor will always be below 3.3 V. Since the temperature sensors are to be placed in every room in the user's house, the 8:1 MUX CD74HC4051PWR was used to cycle the multiple temperature readings into the ESP32 to minimize GPIO usage. As for the gas sensor used, the MQ-9B was selected as it can detect carbon monoxide concentrations anywhere from 10 to 500 ppm, which was low enough to meet our project's high-level requirements. Finally, the WP154A4SUREQBFZGC RGB LED was included in our sensor subsystem as one is placed in every room of the user's house and would take control signals from the ESP32 to visually indicate the designated path for escape to the user in the event of a fire.

2.6.2 Design Details

For the LMT84DCKR temperature sensors, they are supplied by the 5 V line from the power subsystem and have low pass filters at their output to give smooth temperature readings. According to the datasheet, the output temperature T is calculated using Eq. 1 in which V_{TEMP} is the voltage measured at the output of the sensor.

$$T = \frac{5.506 - \sqrt{(-5.506)^2 + 4(0.00176)(870.6 - V_{TEMP}(mV))}}{2(-0.00176)} + 30 \quad (1)$$

This equation gives an output voltage 0.898 V at 25 ℃, which we had to calibrate in the application subsystem to ensure an accurate temperature is being read. These temperature readings are then sent to the 8:1 MUX, which allows the use of only four pins of the ESP32, three GPIO pins for selector signals and one ADC for sensor data, to read the temperature data from up to eight different rooms, thus minimizing the need for each temperature sensor to have its own respective input ADC.

The MQ-9B gas sensor also operates from the 5 V supply line, but it has an additional heater element, as seen between pins 2 and 5 of the equivalent circuit in Fig. B.6 of Appendix B, that requires a switching voltage source between 5 V for 60 seconds and 1.5 V for 90 seconds in order for the sensor to accurately detect carbon monoxide concentrations. From the output of the gas sensor, a parallel sensing resistance of 4.7 KΩ is placed so that the concentration sensitivity can detect 20 ppm of CO. The output is finally sent through a low pass filter and read by an ADC on the ESP32. For the switching voltage requirements of the heater circuit, the initial design utilized the TS5A23157DGSR, a 2:1 MUX, that would switch between 5 V and 1.5 V given a selector signal from the ESP 32. The 2:1 MUX was only rated to carry a maximum of 50 mA, however, which was a problem as the heater circuit in the MQ-9B can draw up to 156 mA at 5 V. The solution was to replace the MUX with a PMOS switching circuit using two DMP2045U-7 p-type MOSFETs as well as two gate signals from the ESP32.

For the RGB LEDs, series resistances of 1KΩ and 22 Ω are placed on the green and red LEDs respectively, which ensures both colors shine with comparable intensity off the 3.3 V signal they receive from two ESP32 GPIO pins.

2.6.3 Design Verification

For verifying the functionality of the MQ-9B, we were not able to get the heating circuit fully operational by the time of the project deadline, but the issue was found and corrected afterwards without testing being able to be performed.

To verify the correct operation of the LEDs, when a fire was detected and LED data was sent from the ESP32, we visually confirmed that each LED lit up the correct color and that they were all equally bright.

When verifying the temperature sensor functionality, initial measurements were showing higher temperatures than expected for room temperature, which correlates to a lower output voltage than expected. The solution is to add an offset into the app subsystem equation for calculating temperature to ensure each temperature reading was accurate. Table 2 below shows example readings at room temperature once the offsets were placed, and each reading gives a reasonable temperature in the range of 24°C to 26°C for room temperature.

Table 2: Verification of Temperature Sensor

	T_1	T_2	T_3	T_4	T_5
Temperature	24.544°C	24.600°C	26.074°C	24.754°C	26.900°C

Additional tests were performed to determine how the temperature sensors responded to increases in temperature. These included using heat produced in our fingers to warm the temperature sensors and we saw an average reading of 28°C for each sensor as well as using a soldering iron to quickly heat the sensor, and we observed a sharp increase in temperature that quickly rose past 30°C and kept increasing.

2.7 Vent Subsystem

2.7.1 Design Decisions

To ensure the user would have as much time to escape as possible, the vent subsystem closes all the vents in the user's home as well as turn off the HVAC unit when a fire is detected to both isolate and try and suffocate the fire. When the fire is cleared, the vents open back up and the HVAC unit returns to normal operation. For the automated vents, a custom enclosure was designed to connect the Hiwonder HPS-2518 digital servo motor to the opening and closing mechanism of a vent cover. The HPS-2518 has a torque of 25 Kg cm, which is enough to operate the vent closure, and it can operate with a 7 V supply along with a PWM control signal that comes from the control subsystem. For the HVAC operation, a WP154A4SUREQBFZGC LED is used as a visual indicator instead of connecting an HVAC unit to our design to visualize when the unit will turn on and off.

2.7.2 Design Details

To operate the digital servo motor, a PWM signal is sent from the ESP32 at a frequency of 50 Hz where the duty cycle of the pulses determines the angle the motor turns through. When connected to the vent enclosure, a duty cycle of 5% was found to correlate with the vent being open and a duty cycle of 10% was enough to close the vent fully. To minimize the current the motor draws from the 7 V line, the duty cycle is stepped by 0.3% over a range of about 2 seconds to meet the design requirements while lengthening the lifespan of the 9 V batteries and protecting the power subsystem from over currents.

For the HVAC indicator, the RGB LED was connected to shine green whenever a fire was detected to signal that the HVAC unit would have turned off and turn off when the fire was cleared to indicate the HVAC unit would have returned to normal operation.

2.7.3 Design Verification

To verify the operation of the HVAC indicator, when a fire was detected, we visually confirmed that the LED indicator shines green and that it turns off once the temperature sensor cooled down and no fire was detected.

For the motorized vent, we again visually confirmed that the vent would close and then open when a fire was detected and then cleared and confirmed that the opening and closing would each take around 2 seconds to finish. From Appendix B, Fig. B.7 shows what the vent looks like when a fire is detected and it is closed, and Fig. B.8 shows what the vent looks like when a fire is cleared, and it is open. Over the operation of the vent opening and closing, a power supply was also used to measure the current draw of the motor to ensure it doesn't exceed limits in the power subsystem. It was found that the current draw reached a maximum of 500 mA during opening, which falls within the ratings of the voltage regulator used to produce the 7 V line.

2.8 Application Subsystem

2.8.1 Design Decisions

The Application Subsystem plays a vital role in the functionality of our project as LED navigation relies entirely on feedback from the application. Key decisions for the application subsystem revolved around ensuring a responsive and efficient user experience. The application needed to dynamically update with the latest data from the sensors and pathfinding algorithm while providing actionable insights to the user. We chose to use React for its component-based architecture and efficiency in building interactive UIs. Firebase was chosen as the backend service due to its real-time capabilities, allowing the app to update in real-time without requiring constant polling, which ensures the responsiveness of the system.

As our goal is to provide the optimal escape route to the user as quickly as possible, we needed a way to store their floorplan for pathfinding. The simplest solution was to create a web application for the user to design their own floor plan including the sensors and LEDs corresponding to the numbered temperature boards. We chose a grid layout with drag-and-drop components for rooms, doors, sensors, LEDs, stairs, and exits using React-DnD, as it allowed the floorplan to be easily created and modified.

To use A-star to find the optimal path, we first needed to create a graph of the floorplan. Using the grid spaces, we were able to convert the user's floor plan into nodes consisting of rooms and exits, with edges consisting of doors and stairs. Nodes contain name, position information, size information, sensor/LED number, and hazard information. The hazard information contains temperature, gas levels, and a hazard score. These are all initialized as 0 and only updated when the sensors are above the threshold.

We chose to use A-star as our path-finding algorithm as it would efficiently provide us with the optimal path using weighted nodes. We first updated the hazard information, then ran A-star to find the optimal route to every exit. The optimal route after this is returned in terms of nodes, which contain LED numbers. The LEDs are assigned by making the optimal path green, hazardous areas red, and suboptimal locations as yellow. We chose to make the LEDs all yellow in the event of no possible exit as it would be a way to inform the user that they need to find an alternate solution.

2.8.2 Design Details

The Application Subsystem is built around several critical components, each of which serves a specific role in the system's overall functionality. These components ensure that the user experience is seamless, interactive, and real-time, while enabling pathfinding and evacuation updates based on sensor data.

Floorplan UI:

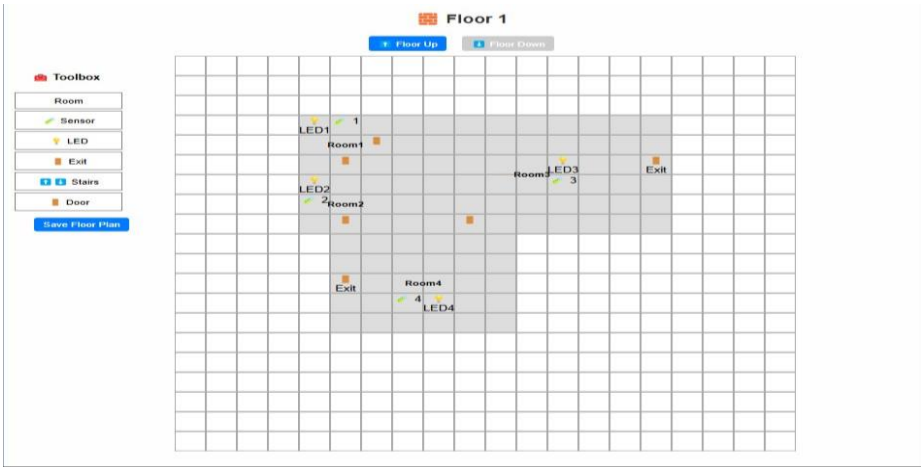

Figure 8: Floorplan Interface

The Floorplan UI is a visual representation of the building layout, which is displayed as a grid to the user. This grid allows the user to interactively place rooms, sensors, LEDs, doors, stairs, and exits, creating a customizable floor plan. The grid system provides users with the ability to drag and drop various components onto the grid to design their own floorplan. Each grid cell represents a portion of the floor, and each component (e.g., room, sensor, LED) is represented by specific markers or icons within those cells. Sensors and LEDs are assigned numbers in this UI which correspond to numbered temperature boards, allowing us to identify which room a sensor is in. When creating a

room, the Room Creation modal shown below appears, allowing the user to size and name the room.

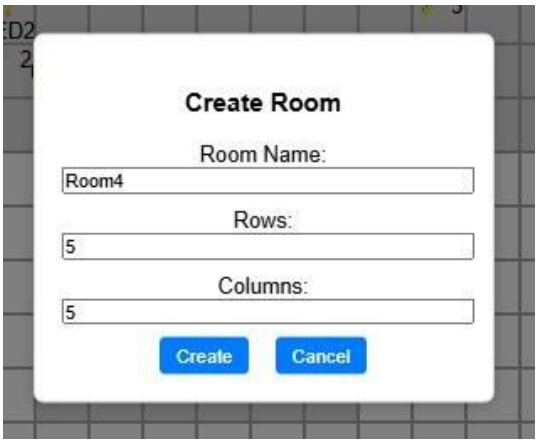

Figure 9: How to Develop Rooms

Graph Builder:

- The Graph Builder is responsible for converting the floorplan grid into a graph that can be used for pathfinding. In the grid, rooms and exits are treated as nodes, while connections between rooms, such as doors and stairs, serve as edges. Each room's data, including its position, size, and hazard information (e.g., temperature and gas levels), is stored within the nodes.
- The graph builder traverses the grid, creating nodes for rooms and exits and adding edges between adjacent rooms that are connected by doors or stairs. Once the graph is built, it is uploaded to Firebase, where it can be accessed by the path-finding algorithm for further calculations. This graph is used by the path-finding algorithm to find the optimal evacuation path, considering hazard levels and room connectivity.

LED Assignment Algorithm:

- The LED Assignment Algorithm is crucial for providing users with real-time evacuation guidance based on hazard levels and path finding results. After the A* algorithm calculates the optimal path, the system assigns LED colors based on the following rules:
    - Green LEDs: These represent the safest rooms as they are in the optimal path.
    - Red LEDs: These represent high-risk rooms that are hazardous, based on sensor data.
    - Yellow LEDs: These are used to indicate rooms that are not optimal but are not hazardous.
- The LED colors are updated dynamically based on real-time hazard information from sensors and the path-finding algorithm's output. The system ensures that the LEDs provide clear, actionable feedback to users during an emergency.

Firebase Integration:

- Firebase serves as the backbone of the Application Subsystem, enabling real-time data storage and synchronization between the front-end UI and the back end. Firebase stores various pieces of critical data, including:
    - Floorplan Data: The grid layout, room configurations, and the associated data (e.g., sensor numbers, LED labels, hazard scores) are stored in Firebase. This ensures that any changes to the floorplan UI are reflected immediately in the database, allowing for easy updates and retrieval of the floorplan configuration.
    - Sensor Data: Firebase continuously updates with the latest readings from temperature and gas sensors. These readings are used to dynamically update the hazard status of rooms. This data is essential for hazard scoring and the LED assignment algorithm.
    - LED Feedback: Firebase stores and controls the LED configuration for each room, which is dynamically updated based on the pathfinding results and the hazard levels. This ensures that the user sees the correct evacuation path and hazard levels in real-time.

2.8.3 Design Verification

To verify the functionality of the application subsystem we need to test the Firebase integration, UI, and pathfinding algorithm. For our Firebase integration we manually verified that data was stored correctly by viewing the database. To ensure that the UI allowed the user to create and store floorplans, we created and stored various floorplans and inspected Firebase to ensure the graph representation was correct. This can be seen below, where the graph storage of a single room and door are shown. Verifying the pathfinding algorithm revolved around modifying sensor values in Firebase manually and ensuring that LEDs changed to the expected output as shown in the image below.

Figure 10: Verification of the Application Subsystem

3 Cost Analysis

There are some factors that will go into the cost analysis for this project. Firstly, here's a list of all the parts that were required for our project.

Table 3: Bill of Materials

Component	Part #	Quantity	Cost	Total Price
MQ-9 Gas Sensor	B07KP4F9FF	2	$4.00	$7.99
MQ-9B Gas Sensor	SEN-17050	1	$6.50	$6.50
9V Li Battery 1200mAh	B09Q2Q3XCG	4	$5.098	$20.39
Vent Cover	RMGFR-4X10	1	$8.95	$8.95
Digital Servo Motor	B0DDKN9RS6	1	$13.99	$13.99
Buzzer 5V	2223-CMI-9605IC-0580T-ND	2	$1.03	$2.06
N-Channel Mosfet Switch	IRLZ34NPBF	5	$1.45	$7.25
01x03 Socket Connector	61300311821	2	$0.32	$0.64
01x02 Socket Connector	61300211821	4	$0.36	$1.44
01x05 Socket Connector	732-61300511821-ND	12	$0.325	$3.90
01x08 Socket Connector	61300811821	2	$0.38	$0.76
01x10 Socket Connector	732-2859-ND	4	$0.57	$2.28
LED RGB	WP154A4SUREQBFZGC	10	$1.253	$12.53
Analog Temperature Sensor	LMT84DCKR	10	$0.386	$3.86
2:1 Dual Mux	296-27120-1-ND	2	$0.50	$1.00
8:1 Mux	CD74HC4051PWR	2	$0.32	$0.64
Fixed 5V Voltage Regulator	BD50FC0FP-E2	2	$1.63	$3.26
Fixed 3.3V Voltage Regulator	AZ1117CD-3.3TRG1	2	$0.64	$1.28
Mosfet	IRLML0030TRPBF	2	$0.39	$0.78
6.8μH Inductor	NRS4018T6R8MDGJ	2	$0.17	$0.34
Buck Converter	LMR33630	2	$2.94	$5.88
Linear Adjustable Regulator	497-1482-5-ND	2	$1.43	$2.86
Capacitor .1μF (0805)	C0805C104K5RACTU	20	$0.03	$0.60
Capacitor 33μF (0805)	C0805X5R0J336M125AC	14	$0.474	$6.636
Capacitor 10μF (0805)	C0805X106J8RAC7210	25	$0.653	$16.325
Capacitor 1μF (0805)	CL21B105KAFNNNE	14	$0.021	$0.294
BJT	SS8050-G	4	$0.24	$0.96
Resistor 10kΩ (0805)	CRCW080510K0FKEAC	20	$0.04	$0.80
Resistor 1.5kΩ (0805)	RC0805JR-071K5L	10	$0.01	$0.10
Resistor 22Ω (0603)	ERJ-U01F0805C	13	$0.093	$1.209
Resistor 43.2Ω (0603)	CRCW080543R2FKEA	13	$0.031	$0.403
Resistor 4.7kΩ (0603)	CRCW0805K70FKEA	6	$0.10	$0.60
Resistor 1kΩ (0805)	ERA-6AED102V	20	$0.187	$3.74
Resistor 182Ω (0603)	CRCW0805RFKEAC	8	$0.11	$0.88
Resistor 100kΩ (0805)	RC0805JR-07100KL	8	$0.1	$0.80
Switch Tactile	1825910-6	6	$0.13	$0.78
Test Points	5005	35	$0.2624	$9.184
Microcontroller	ESP32-S3-WROOM	3	$5.49	$16.47
				$163.5

As seen above, the total cost for our parts is $163.5. Adding a 10% shipping cost and a 7.25% sales tax increase, we get a new total cost of $191.7.

We now must consider labor costs that go into our project. The average Graduate Electrical Engineer salary is $82,575 a year, the average for a Graduate Computer Engineer is $121,515 a year. So, taking hourly rates, we see that on average Electrical Engineers get paid $41.29 per hour, and Computer Engineers get paid $58.42 per hour. We have two electrical engineers in the group, and one computer

engineer. We estimated working 9 hours a week, which is around 9 weeks. As a result, we can estimate the total labor cost for our project as follows:

$$2(\$41.29 * 9 * 9) + (\$58.42 * 9 * 9) = \$11{,}421$$

So, in total, the estimated cost for this project is $11584.5.

4. Conclusion

By the day of our final demo, we were successful in achieving what we had set out to accomplish in our project, except for the gas sensor which we will talk about in more detail in the next section. We were still able to reach all our three high-level requirements, which was an achievement for our group. This Achievement shows our perseverance and diligence in working on this project and overcoming new challenges and adapting on the fly. During the demo, we showcased our project running and displayed a scenario in which the most optimal path is shown, and all this information was also available on the Firebase web page for everyone to see. And with this, this demonstration highlighted the functionality and reliability of our design if it were to be integrated into a practical application as it has a lot of merit.

4.1 Uncertainties

The gas sensor proved to be the component that we will struggle to get to work. The reason for this wasn't a complex one, it was more of a timing issue. We had ordered our third round PCB board, which we expected would work with the gas sensor, but we forgot to account for the fact that the 2:1 Mux we were using was rated for 50mA, and the gas sensor demands 200mA. So, we then changed the design to PMOS logic switching which essentially will switch between the voltage levels needed for the MQ-9B heater pins to detect gases, 5V and 1.5V respectively. This change was only present on the fourth round PCB boards which came four days before the final demo. Upon testing the gas sensor, we noticed that all our voltage levels would be off, indicating that our implementation of the PMOS was causing excessive current draw which could affect the whole system if left on for long. The issue of this was simply a footprint issue on Ki-Cad, the model we had chosen had the incorrect footprint to what the actual PMOS was. So, given enough time, we are confident that we would be able to correct this issue as it would just require getting a new board that corrects this issue or find a way to twist the legs of the PMOS to fit the right ordering of source, drain, and gate as it corresponds to the chosen footprint.

4.2 Future Work

When it comes to the future of our project, we would of course like to get full functionality of our project, this means getting our gas sensor to work. Outside of this, future plans would be to replace the LEDs with something more practical for real life applications. For example, we can have LED strips on the floor (embedded), and with these strips, instead of having colored lights, we can just have a bright light on the floor that shines the given path to safety. The reason for the floor is because smoke can reduce visibility, so, the area in which you will be able to see more clearly would be the floor.

In addition to this newly reinvented LED model, we would also like to add more than just a single-story layout. As of now, our model will only work for single-story homes, so, in future work, we will make it so there can be multilayered houses, i.e., the second floor, and basement house models will work. Then,

once this works, we can then make it work for hotels as well where it would most likely be the most useful as it would be beneficial to have these guides in hotels.

4.3 Ethics

In designing our fire and gas detection system, we adhere closely to the IEEE and ACM Codes of Ethics to prioritize the well-being and privacy of our users. Our foremost ethical responsibility is to protect public health and safety, in alignment with IEEE Section I.1, which includes disclosing any safety risks associated with our project. Furthermore, our system's use of personal data such as user-input floor plans raises ethical considerations under ACM Section 1.6. We address this by ensuring that all personal information is securely handled, accessed only by the control unit, and limited strictly to what is essential for safe navigation and system operation.

4.4 Safety

Safety is integral to every aspect of our project, from power systems to environmental testing. We mitigate the risks of using 9V batteries by implementing fault protection to prevent harm to users and hardware. During development, we will test sensors in controlled, ventilated environments, especially when simulating hazardous conditions like carbon monoxide exposure to uphold lab safety standards. Additionally, moving components like the automated vent system will be safely enclosed to prevent injury, and precautions will be taken to ensure the HVAC system shuts off correctly during emergencies to safeguard both the equipment and the user.

5 References

[1] "Ventilation-Limited Fires and the Influence of Oxygen," 2017. [Online]. Available: https://www.fireengineering.com/firefighting/ian-bolton-ventilation-limited-fires-and-the-influence-of-oxygen/

[2] H. Naidoo, "Oxygen Enrichment and Fire Hazards," 2024. [Online]. Available: https://www.co2meter.com/blogs/news/oxygen-enrichment-hazards?srsltid=AfmBOoqvc_zfwub4eY8siFmFGKAD62cOa29HnswcrraIdGanyEZ2TTF1

[3] *Magnetic Buzzer Indicator*, Same Sky, 2024, rev. 1.02. [Online]. Available: CMI-9605IC-0580T Datasheet - Audio Indicators | Buzzers | Same Sky

[4] *IRLZ34NPbf Power Mosfet*, International Rectifier, rev. 5. [Online]. Available: IRLZ34NPbF.pmd

[5] *Full Color LED Lamp*, Kingbright, 2021, rev. V.9A. [Online]. Available: WP154A4SUREQBFZGC(Ver.9A)

[6] *Analog Temperature Sensors*, Texas Instruments, 2017, rev. 2.0. [Online]. Available: LMT84 1.5-V, SC70/TO-92/TO-92S, Analog Temperature Sensors datasheet (Rev. E)

[7] *High-Speed CMOS Logic Analog Multiplexer and Demultiplexer*, Texas Instruments, 2024, rev. 4.0. [Online]. Available: CDx4HC405x, CD4HCT405x High-Speed CMOS Logic Analog Multiplexer and Demultiplexer datasheet (Rev. N)

[8] *Toxic Gas Sensor*, Winsen, 2015, rev. 1.4. [Online]. Available: MQ-9B_Ver1.4__-_Manual.pdf

[11] IEEE, "IEEE Code of Ethics," ieee.org, Jun. 2020. https://www.ieee.org/about/corporate/governance/p7-8.html

[12] ACM, "ACM Code of Ethics," ACM, [Online]. Available: https://www.acm.org/code-of-ethics

[13] Espressif Systems, *ESP32 Series Datasheet*, 2022. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Accessed: 1-Oct-2024].

Appendix A: Requirement and Verification Table

Table 4: Sensor Subsystem RV Table

Requirement	Verification	Verification status (Y or N)
1. The Temperature Sensor will be able to measure temperature from -50°C to 150°C with an accuracy of ±0.4°C.	1. Power on the Temperature Sensor (LMT84DCKR) with 5V. With a Voltmeter, measure the voltage at ambient temperature, and use the transfer function table given in the document to read out the corresponding temperature. Compare this with the room temperature. a. Using a heat source, measure the voltage and if it decreases, it shows proper functionality b. Using a cold source, measure the voltage and if it increases, it shows proper functionality.	Y
2. The gas sensor will be able to detect Carbon Monoxide ranges from 10~500 ppm, Methane ranges from 300~10000ppm	2. Power on the Gas Sensor (MQ-9B) by using a 5V DC source across pins 3 and 6. Next, you need a separate power supply that can cycle between 5V and 1.5V for 60s and 90s respectively across pins 2 and 5. Measure the output as it will read a voltage value, use the table provided in the document to convert this value into ppm and see if it's within reasonable ranges in ambient environment. a. Find a way to produce Carbon Monoxide (burning paper), then bring this close to the gas sensor and note if the voltage increases, if so, the functionality is verified.	N
3. The LEDs will display yellow, green, and red colors above 300mcd which we deemed to be bright for our demonstration.	3. Power on the LED with 3.3V across pins 1 and 4. These pins should also have a current limiting resistor, a value of 100Ω should be fine in testing the LED. Ground the Cathode, pin 2, and once the power is on you should verify if yellow is seen, you can use a phone app to measure the brightness. a. Next, we will disconnect power from pin 4 and only power on pin 1. You should now see red, verify the brightness. b. Finally, we will disconnect power from pin 1 and give power to pin 4. You should now see green, very brightness.	Y

Table 5: Vent Subsystem RV Table

Requirements	Verification	Verification status (Y or N)
1. The Motor should positionally be able to rotate its arms from 0-180°	1. Power on the motor with a 7V input. Using a signal generator, adjust the duty cycle to verify that the motor arm can rotate from 0-180°.	Y
2. The motor should be able to open and close the vent	2. Power on the servo motor with a 7V input and apply a PWM signal from the ESP32 with positional commands which will rotate the arm to 0° which is our open position. a. Next, to verify that the vent can close, send a command from the ESP32 which will close the vent which occurs at 180°.	Y
3. The motor should at maximum draw 500mA during it's opening and closing operation	3. Power on the servo motor with a 7V input and apply a PWM signal from the ESP32 which will rotate the motor arm from 0-180°. With the power supply, measure the amount of current that is drawn when you send the command for the motor to close and open. a. Next, if the current draw is peaking at 500mA, then, raise the current limit to 600mA, ensure that the range is close enough to 500mA to protect the batteries from discharging fast.	Y

Table 6: Application Subsystem RV Table

Requirements	Verification	Verification status (Y or N)
1. The Application must correctly display and store the user-defined floor plan.	1. Test drag and drop functionality and create example floor plans. a. After constructing graph representations, use visualization tools to print the graph and manually ensure it is correct. b. Store to Firebase and manually ensure the graph is stored correctly.	Y
2. The app must determine and communicate the escape route to the control unit within 5	2. Use logging within the application to determine the time taken to calculate the optimal route and the time taken to store the LED configuration.	Y

seconds of hazard detection.	a. Run the algorithm with dummy sensor data and an example floor plan to ensure the time taken to calculate the route and store the LED configuration is less than 5 seconds.	
3. The app must consistently receive sensor data while connected to Wi-Fi in real time.	3. Display sensor data in the application read directly from Firebase. a. Connect the ESP32 to Firebase to store sensor data. b. Manually change sensor values by increasing temperature and verify real-time changes in the application.	Y

Table 7: Power Subsystem RV Table

Requirements	Verification	Verification status (Y or N)
1. The LDO Regulators must be able to sustain 7V and 1.5V ±0.2V at a load of 500mA	1. Power on the PCB board with 9V, in this case, you can use the motor to give you 500mA load or find another way to get a 500mA. Next, using a multimeter, you can probe the output of the regulator, and then, send a command to the motor to turn on and draw current. a. Monitor the voltage across the regulator, you should see that 7V and 1.5V is maintained so regulation is guaranteed for our project.	Y
2. The Buck Converter should output 3.3V ±0.2V at a load of 500mA.	2. Power on the PCB board with 9V, find a load that can draw around 500mA. Next, probe the output of the buck converter, and observe that the buck converter can maintain 3.3V at 500mA load.	Y
3. The fixed voltage regulator should output 5V ±0.2V at a load of 500mA.	3. Power on the PCB board with 9V, find a load that can draw around 500mA. Next, probe the output of the regulator and observe that the regulator can maintain 5V at a 500mA load.	Y
4. The battery must be able to provide 9V during the 1200mAh operation.	4. Set up a no-load test to measure the voltage across the battery, a fresh battery should be 9V or higher. a. Connect a 1kΩ load resistance to the battery and see if the battery maintains a voltage of around 9V with this light load. b. If an extra battery is available, use a resistor that draws 300mA. The battery should then last 4 hours at a 300mA load.	Y

Table 8: Control Subsystem RV Table

Requirements	Verification	Verification status (Y or N)
1. ESP32 should be able to send PWM signals for the motor	1. Power on ESP32 with 3.3V and program a PWM capable GPIO pin on the ESP32 which will send a square wave at a given duty cycle in a period. a. Connect an LED with some current limiting resistor to the output of the PWM pin. b. You should now note that the LED will go bright, and after a while will dim, and continue this cycle indicating successful PWM signal.	Y
2. ESP32 should be able to connect to the Wi-Fi to ensure connection to Firebase	1. Power on ESP32, use the Wifi.h library and follow the rules on how to connect to the Wi-Fi. Once you run the code, you should now see in the console that the ESP32 has connected to the Wi-Fi. a. You now must ensure that ESP32 will send the information to Firebase, this can be done by following the Firebase library available in Arduino IDE. Upon doing this, you will note that on the Firebase page, ESP32 is now connected.	Y
3. ESP32 will read temperature sensor data from an 8:1 multiplexer in milliseconds. a. When a sensor reaches the threshold temperature (90°) indicating a fire, an alarm will sound. b. The alarm should be turned off when the fire is cleared.	3. Power on the ESP32 as well as the temperature sensors and the alarm. Without any hazard, verify that the ESP32 is cycling through the channels and reading accurate data from the sensors (room temperature) in milliseconds. a. Upon verification of the previous step, take a heat source, or lower the threshold of when the alarm triggers, but you now want to heat one of the temperature sensors up. b. Notice that only one temperature sensor starts increasing in temperature, if you do get it to 90°C, you should read .543V using a multimeter on the temperature sensor output. c. Upon reaching this level, the ESP32 should trigger the alarm. Once the temperature sensor falls below the	Y

	threshold, the alarm should shut down.	
4. ESP32 correctly stores sensor values to Firebase	4. Power on the ESP32, set up the sensors with the ESP32 connected to Wi-Fi. Then program the ESP32 to store sensor data in Firebase. a. Manually change sensor values by increasing temperature and verify real-time changes in the database.	Y
5. ESP32 correctly reads LED configuration from Firebase and signals the right LEDs to turn on	5. Power on the ESP32 and connect to the Wi-Fi and set up the LEDs. Program the ESP32 to store sensor data in Firebase. a. Store an example LED configuration that you manually make in Firebase and check to ensure that the LEDs that turn on match the configuration.	Y
6. The buzzer should be able to play a sound of 80dB at 5V.	6. Using your board or a power supply, connect the buzzer to a 5V source. A sound should immediately sound, and the range should be around 80dB. a. Use a phone app or a decibel meter to measure how close the sound being played is to 80dB. b. If the sound is within ±4dB of 80dB, then the alarm is functioning properly, if not, use a voltmeter to measure the voltage across the buzzer alarm ensuring that is indeed 5V.	Y
7. ESP32 can read values from the gas sensor and trigger an alarm once the threshold is passed (CO > 20ppm) as well as turning off the alarm once the concentration returns to normal levels.	7. Power on the ESP32 as well as the gas sensor with their respective voltages that are required. Upon ensuring that the ESP32 is reading values from the gas sensor by looking at the console and reading that the values are of normal concentration (clean air). a. Grab a carbon monoxide source, burning paper works, and bring it close to the gas sensor. Once the threshold is met, the alarm should start ringing. b. Once the hazard clears, the alarm should stop playing the sound and you will be able to visually confirm on the console that the gas concentration is indeed decreasing.	N

Appendix B: Figures

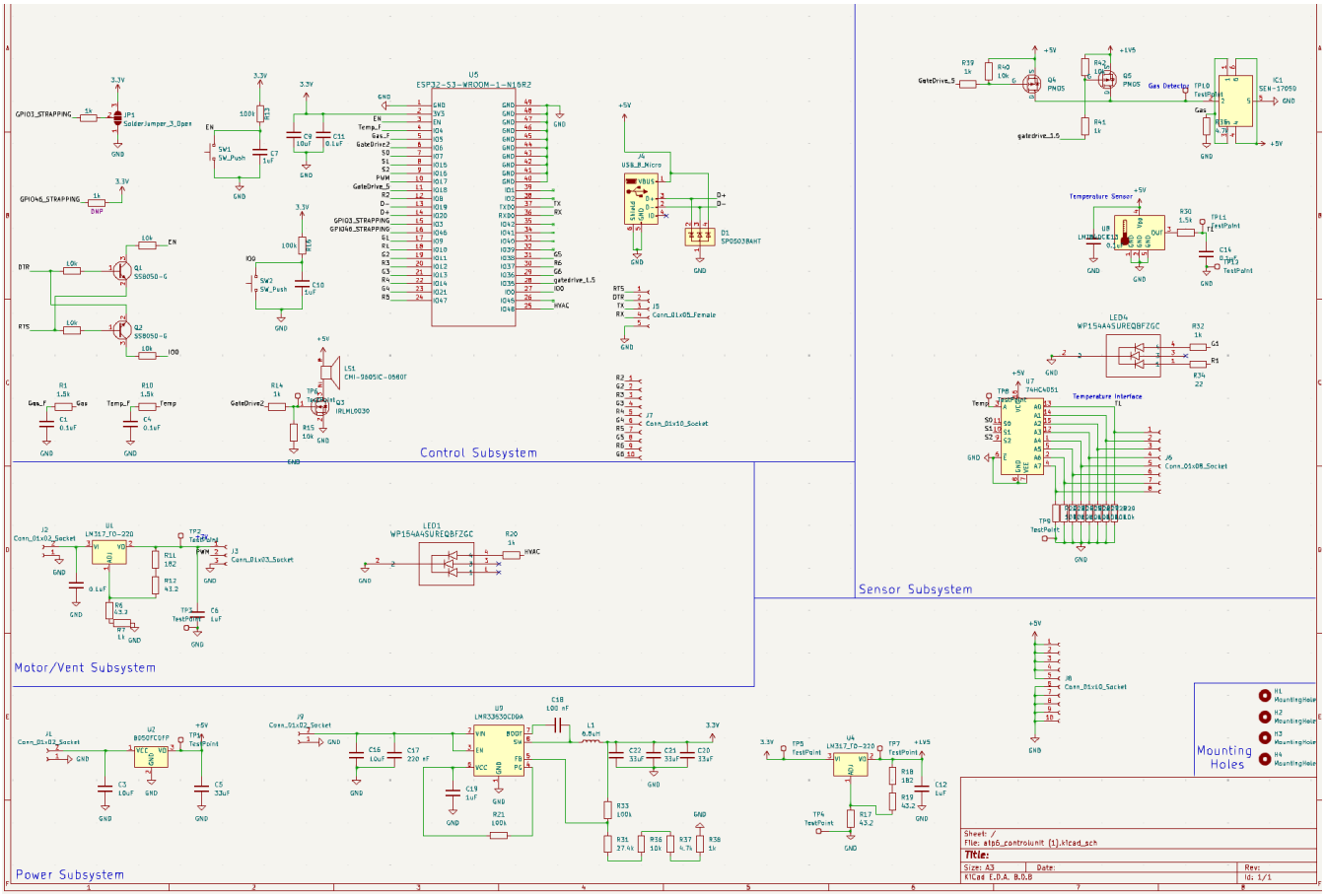

Figure B.1: Fire and Gas PCB Schematic

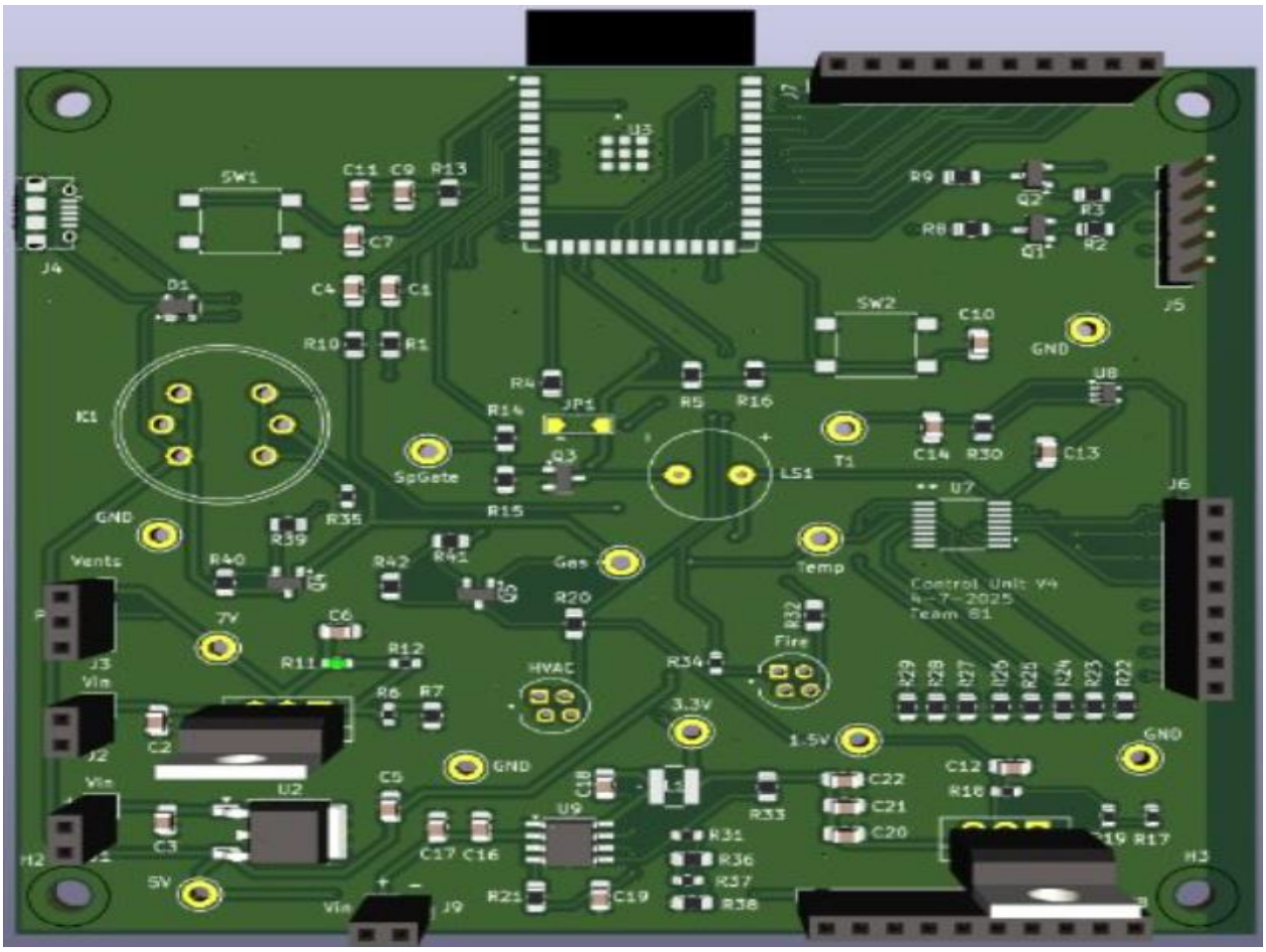

Figure B.2: 3D View of Main PCB Board

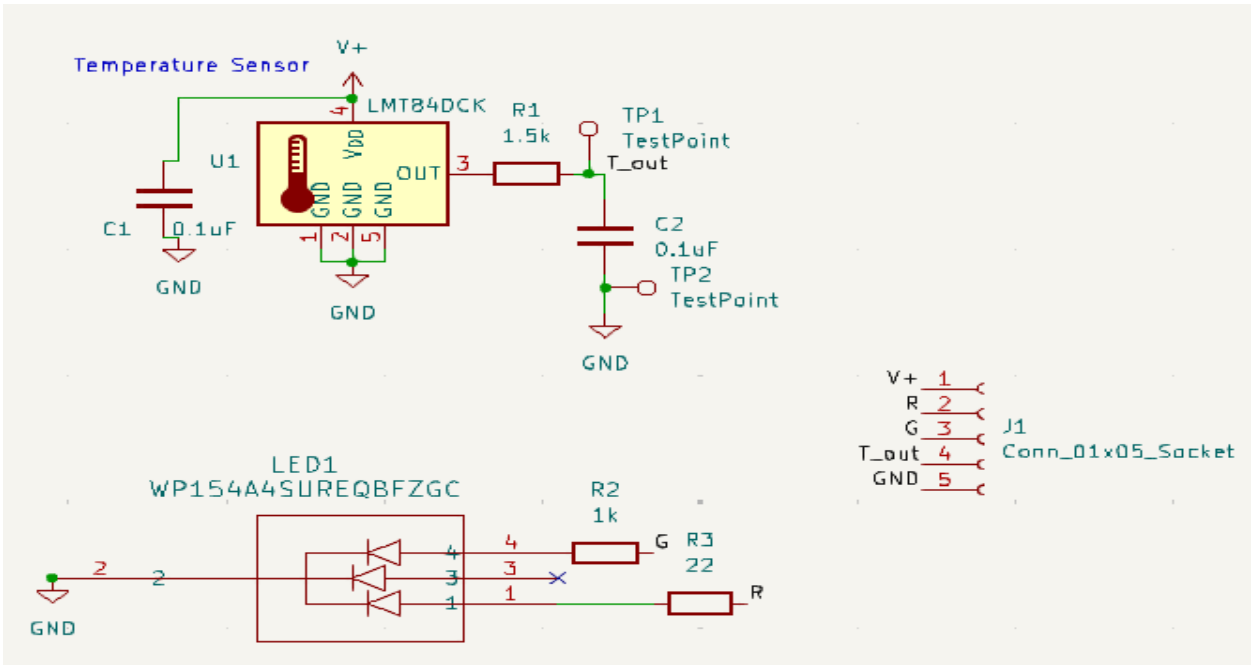

Figure B.3: Temperature Sensor PCB Schematic

Figure B.4: 3D View of Temperature PCB Board

Table 3. LMT84 Transfer Table

TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)
–50	1299	-10	1088	30	871	70	647	110	419
–49	1294	-9	1082	31	865	71	642	111	413
–48	1289	-8	1077	32	860	72	636	112	407
–47	1284	-7	1072	33	854	73	630	113	401
–46	1278	-6	1066	34	849	74	625	114	396
–45	1273	-5	1061	35	843	75	619	115	390
–44	1268	-4	1055	36	838	76	613	116	384
–43	1263	-3	1050	37	832	77	608	117	378
–42	1257	-2	1044	38	827	78	602	118	372
–41	1252	-1	1039	39	821	79	596	119	367
–40	1247	0	1034	40	816	80	591	120	361
–39	1242	1	1028	41	810	81	585	121	355
–38	1236	2	1023	42	804	82	579	122	349

Feature Description (continued)

Table 3. LMT84 Transfer Table (continued)

TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)	TEMP (°C)	V_{OUT} (mV)
–37	1231	3	1017	43	799	83	574	123	343
–36	1226	4	1012	44	793	84	568	124	337
–35	1221	5	1007	45	788	85	562	125	332
–34	1215	6	1001	46	782	86	557	126	326
–33	1210	7	996	47	777	87	551	127	320
–32	1205	8	990	48	771	88	545	128	314
–31	1200	9	985	49	766	89	539	129	308
–30	1194	10	980	50	760	90	534	130	302
–29	1189	11	974	51	754	91	528	131	296
–28	1184	12	969	52	749	92	522	132	291
–27	1178	13	963	53	743	93	517	133	285
–26	1173	14	958	54	738	94	511	134	279
–25	1168	15	952	55	732	95	505	135	273
–24	1162	16	947	56	726	96	499	136	267
–23	1157	17	941	57	721	97	494	137	261
–22	1152	18	936	58	715	98	488	138	255
–21	1146	19	931	59	710	99	482	139	249
–20	1141	20	925	60	704	100	476	140	243
–19	1136	21	920	61	698	101	471	141	237
–18	1130	22	914	62	693	102	465	142	231
–17	1125	23	909	63	687	103	459	143	225
–16	1120	24	903	64	681	104	453	144	219
–15	1114	25	898	65	676	105	448	145	213
–14	1109	26	892	66	670	106	442	146	207
–13	1104	27	887	67	664	107	436	147	201
–12	1098	28	882	68	659	108	430	148	195
–11	1093	29	876	69	653	109	425	149	189
								150	183

Figure B.5: Temperature Sensor Readout Values

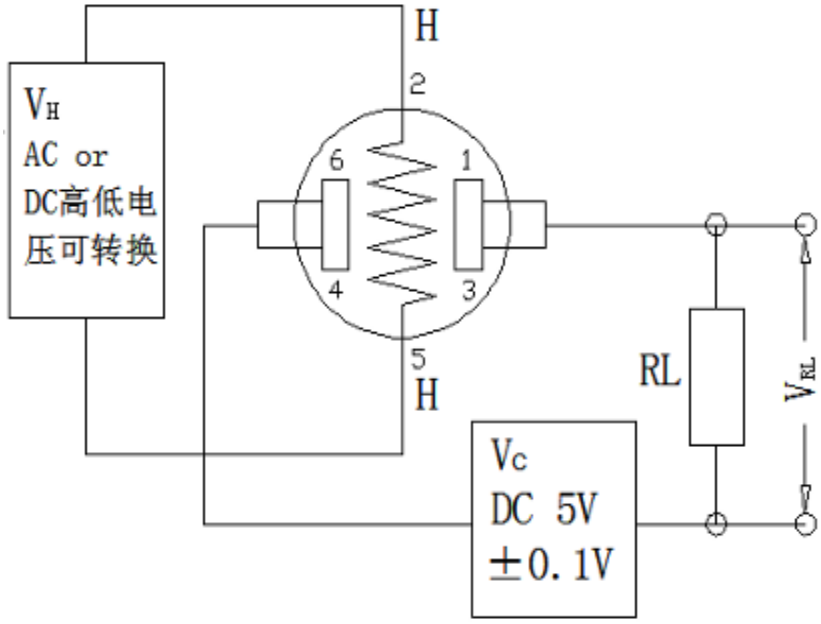

Figure B.6: Gas Sensor Circuit

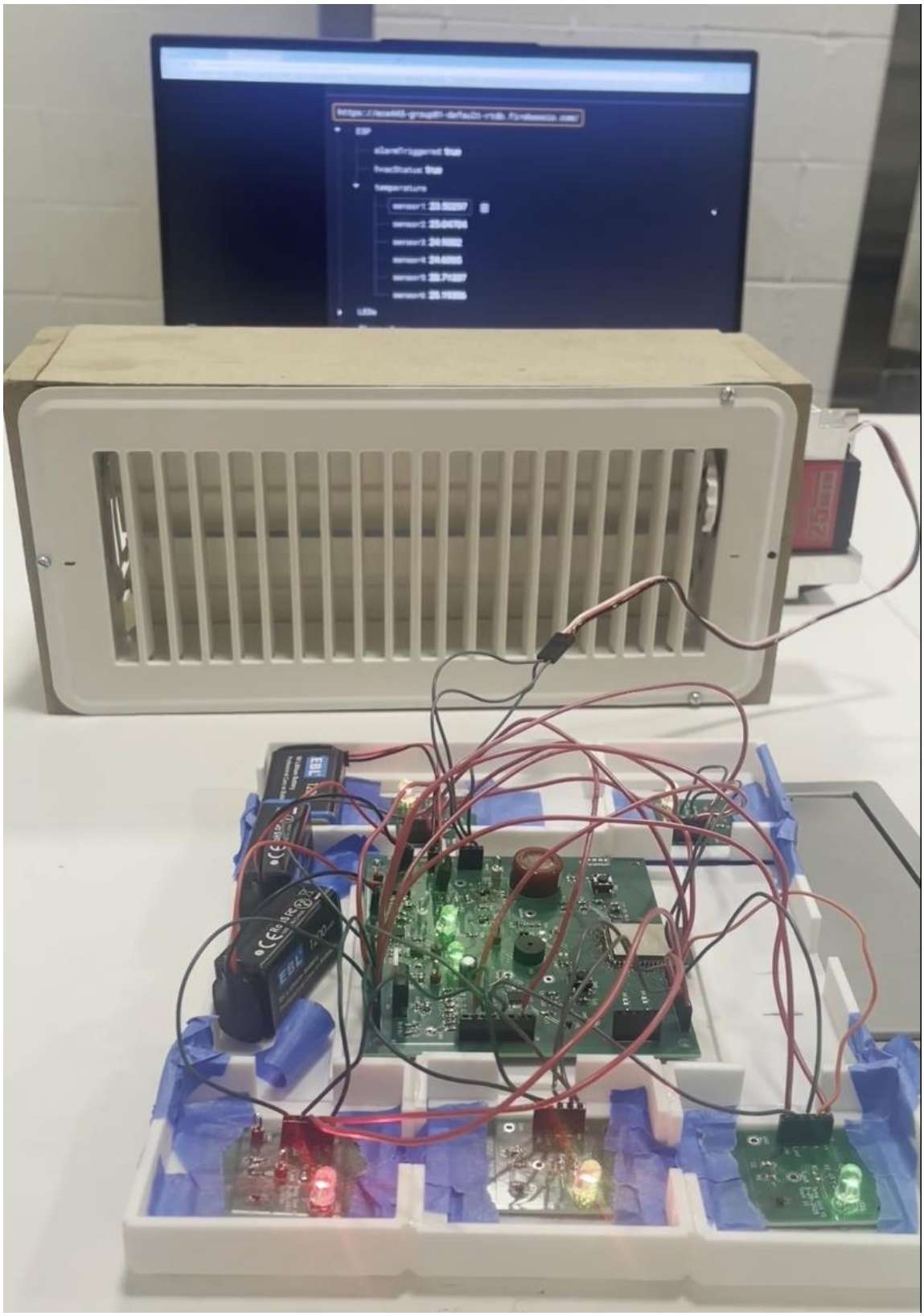

Figure B.7: Project Functionality in the Case of Fire

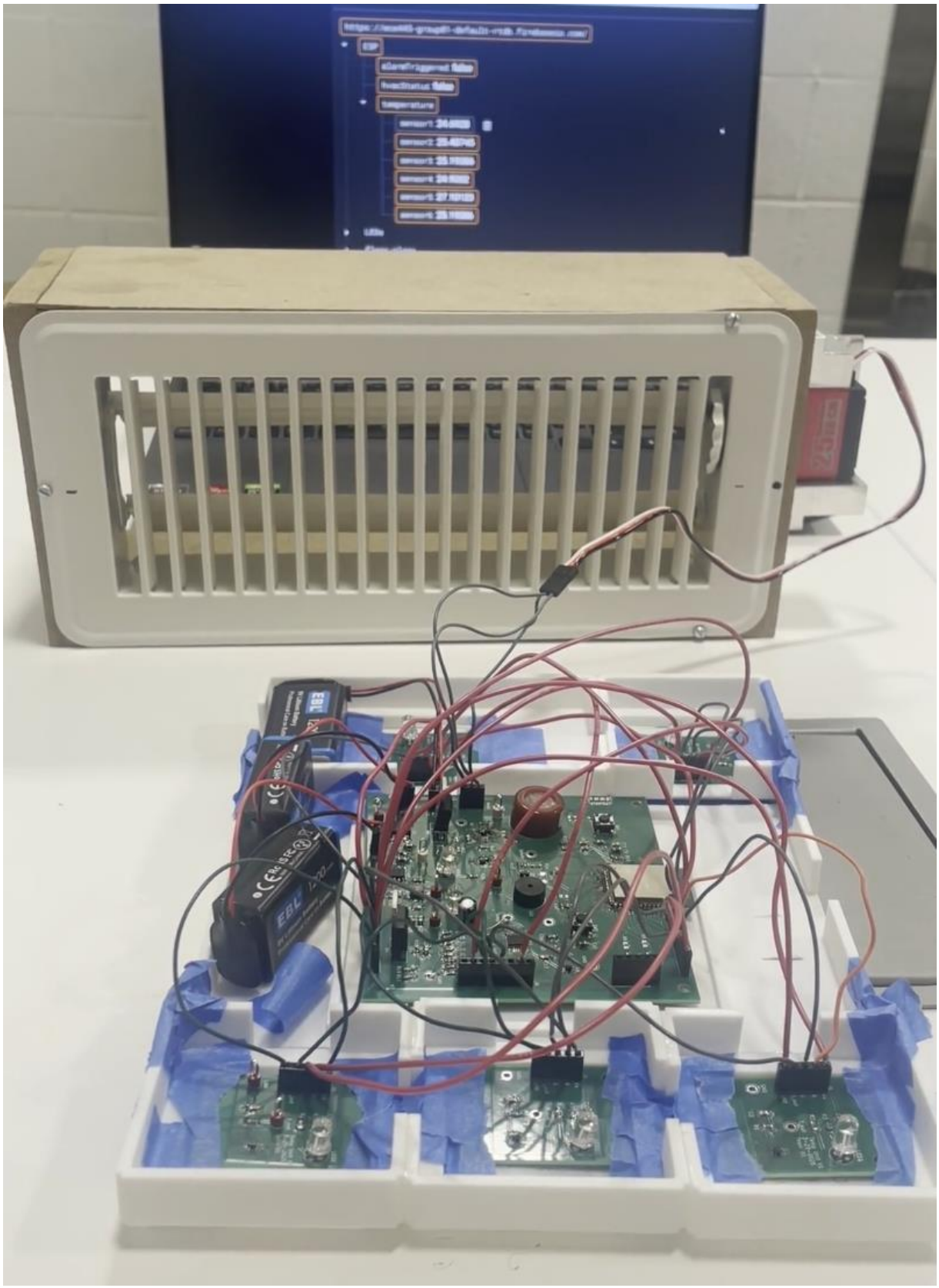

Figure B.8: Project Functionality in the Case of no Hazard

Appendix C: Schedule

Table 9: Schedule of Project

Week	Task	Person
March 2nd - March 9th	Finish 1st round PCB orders	Alex
	Finish Design Document	Everyone
	Start Breadboard Testing	
	Teamwork Evaluation 1	
March 9th - March 16th	**2nd round PCB orders (3/13/25)**	Everyone
	Breadboard Demo (3/11/25)	
	Start Developing Application	Jainam
March 16th - March 23rd	Spring Break	Everyone
	Start the Individual Progress Report	
March 23rd - March 30th	Start Soldering/Debugging	Abel & Alex
	Start Testing Application	Jainam
March 30th - April 6th	**Third Round PCBway orders (3/31/25)**	Everyone
	Individual Progress Report (4/2/25)	
	Start to integrate both Software and Hardware	

April 6th - April 13th	**Final Round PCBway Orders (4/7/25)**	Everyone
	3D print enclosure for the PCB boards	
	Have the Model of the floorplan ready	
	Continue Testing the System under different circumstances	
April 13th - April 20th	Start working on Final Paper	Abel
	Continue Testing with full modelled system	Everyone
	Prepare for Mock Demo	
April 20th - April 27th	**Mock Demo (4/22/25)**	Everyone
	Finalize Testing for the Final Demo	
	Start work on Final Presentation	
April 27th - May 4th	Week of the Final Demo	Everyone
	Finalize Presentation	
May 4th - 11th	Week of the Presentation	Everyone
	Finish the Final Paper (5/7/25)	
	Lab Notebook Due (5/8/25)	