Introduction to Engineering Design with Professional Development 1

Final Report for The Smart Fire Alarm System

Team: SPACHZ
Section: 1
Instructors: Karthik Panneerselvam and Christine Allard

Version 1.0
August, 2024

Prepared by
Andy Huang (2026, CSE)
Josh Suber (2026, CSE/EE)
Pritom Paul (2026, CSE)
Omar Alsofi (2026, ME)
Shamik Chatterjee (2026, CSE)
Andrew Zheng (2026, CS/CSE)

Executive Summary

To Do: The Executive summary is a condensation of an entire report and must be short; try to keep it no more than one page. Focus on the objective(s) of project, the major points of your design and validation (what was done), the results and your recommendations. It is neither an introduction nor outline (table of contents) of the report. Hence, it should not contain phrases such as "This report presents …" and "Our main results are described in chapter 1".

A busy executive may **only** read this page to decide whether to pass your report along to a staff member! It has to grab their attention and make them want to read the rest.

Write this part **last** so that it accurately reflects the content of your completed report.

Table of Contents

Note on the Table of Contents – let Word automate it for you – the page numbers will be correct!

Revision History

Table 1 - Revisions

Version	Date	Name	Reason for Changes
0.0			Final Document

1 Introduction

In modern society, one of the most prevalent safety devices is the smoke detector. It is highly effective in preventing home fire related deaths, and is extremely easy to install. It is also low cost, making it extremely affordable. Most smoke detectors alert the homeowners with a loud noise through the built in speakers. This limits its effectiveness with those who are hearing impaired, leading to the invention of light based smoke detectors. In an effort to create an all encompassing accessible, the SMART Fire Alarm System project was created. By incorporating visual, tactile, and auditory signals, along with an app that allows for remote monitoring and alerts, it can address the needs of multiple disabled individuals and save more lives.

2 Project Objectives & Scope

Modern-day fire alarm systems are often inaccessible to deaf or hard-of-hearing individuals due to their reliance on solely auditory alarms. This issue is addressed by a new system designed to target multiple senses within the human body, creating an alarm that is accessible to a majority of people.

Problem areas of focus:

- Making fire alarm systems more accessible to disabled individuals
    - Specific focus on deaf/low hearing individuals
- Creating a system that can alert people who are currently asleep
- Developing a system that allows users to interact with it on their personal devices for easier access

2.1 Mission Statement

To create a user-friendly, accessible fire alarm system that meets the needs of disabled individuals and offers smart functionalities for broader user appeal in U.S. residence

2.2 Customer Requirements & 2.3 Technical Specifications

Customer Requirement	**Importance**	**Technical Specification**	**Target Value / Range of Values**
Capable of Alerting User	5	Sound Lights Vibrations	Alarm produces between 65 to 120 decibels 150 candelas 130-180Hz
Doesn’t hurt users	5	Safety	0 injuries
Non-toxic to the environment	5	Not Hazardous	Nearly 0 Radioactive Material

Customer Requirement	Importance	Technical Specification	Target Value / Range of Values
Capable of Alerting User	5	Sound Lights Vibrations	Alarm produces between 65 to 120 decibels 150 candelas 130-180Hz
Range of smoke detection	5	Range Radius	25m
Consistent	5	Time before failure arrives	5 years

3 Assessment of Relevant Existing Technologies

Table 2 - Competitive Benchmarking

Competitive Product	Title / Description	Relation to this project
Kidde Smoke Detector	Smoke Detector	Competing model with auditory and visual signaling capability
First Alert Smoke Alarm	Smoke Detector	Competing model with auditory and visual signaling capability

Two contemporary smoke detector products, as shown in Table 2, were used for benchmarking. The first product, the Kidde smoke detector, includes a photoelectric sensor, a carbon monoxide sensor, and is powered by a sealed battery with a 10-year lifespan. It signals the presence of a fire by emitting a high-pitched sound and activating a small LED (*Combination Smoke & Carbon Monoxide Detectors*, n.d.).

The second product, the BRK First Alert smoke alarm, includes an ionization sensor, a carbon monoxide sensor, and is also powered by a sealed battery with a 10-year lifespan. It signals the presence of a fire by emitting a high-pitched sound (Wholesale Home, n.d.). To measure up to contemporary smoke detectors on the market, the new product will need to be powered by a battery with a life of at least 10 years and include at least one of either a photoelectric or ionization sensor to detect smoke. Additionally, the product will need to include a carbon monoxide sensor. It will expand upon contemporary smoke detector designs by incorporating more robust visual and tactile signaling for users who are hard of hearing.

4 Professional and Societal Considerations

The team applied the engineering design process to produce solutions that meet the specified needs with consideration for the topics found in Table x - Engineering Solutions Impact.

Table 4 - Engineering Solutions Impact

Area of Impact	Impact	Description of Impact
Public Health and Safety	Yes	The standard smoke detector in most American homes greatly increases the chances of surviving a house fire. With the addition of extra sensors and alarm sources, the likelihood of the homeowner becoming aware of a fire further increases.
Global	No	Additional features on a smoke detector, however, are not significant enough to cause change on a global scale. Smoke detectors with smart features are already available on the market and have not yet created any global impact.
Cultural	No	The smoke detector and fire alarms are already considered standard safety protocol in most American homes. Creating a smart variant of such will not have a major cultural effect.
Societal	Yes	With the device being aimed at making the traditional smoke alarm more accessible, the device has the potential to bring more attention to the needs of the disabled. Demand for such devices could increase, and has the chance to push other technologies to adapt to those who are disabled.
Environmental	No	Smoke detectors in general do not have a direct impact on the environment, other than the small amount of radioactive material in ionization sensors. The product would only contribute to more electronic waste, which currently is not a major impact to the environment.
Economic	No	Since the device is not cheaper or easily mass produced compared to a normal smoke detector, it does not have a major economic impact. It is also only acquirable by those who can afford the increased price of the device due to its extra features.

5 System Concept Development and Selection

Early System Flowchart:

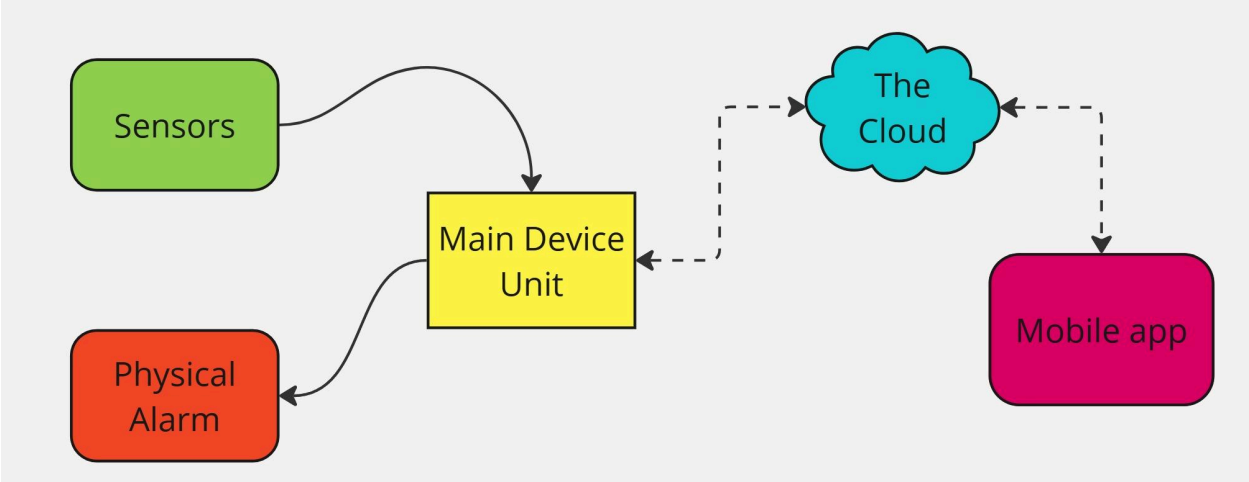


Figure X above depicts the initial design idea flowchart. The main device, which would be mounted on the ceiling, contains the physical alarms and sensors. While collecting information from the sensors, all data would be transmitted to the mobile app via the cloud. If a fire is detected by the sensors, the main device will activate the physical alarms and send data through the app to notify the user's phone.

Detection System Sketches:

Figure 5.#: Low-level diagram of the workflow

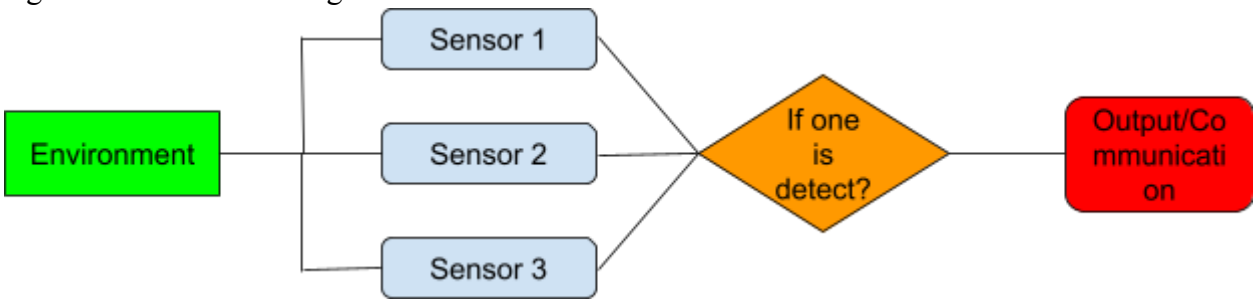


Figure 5.# presents the low-level diagram of the design process, illustrating how information is gathered from the environment and flows through the detection subsystem, leading to the final outcome in other subsystems. The environment serves as the external input, which is processed by one of the three sensors. A conditional statement (represented by the yellow diagram block) then determines the appropriate action based on the sensor input. The outcome of this decision-making process influences the physical design of the product and communicates with other subsystems.

Table 5.1: Initial Proof of Concept Selection Sensor Matrix

Criterion	**Infrared (IR) sensor**	**Ionization Sensor**	**Temperature Sensor**	**MQ-7 sensor**
Ease of Installation/Use	+	-	+	+
In the Market?	+	-	+	+
Cost-effectiveness	+	-	+	+
Response Time	+	+	+	+
Fire/Smoke Indicators	+	+	-	+
SUM	5	-1	4	5
Use?	Yes	Yes	Yes	Yes

Table 5.1 above presents the concept selection matrix used to determine the sensors for the initial product plan. As a result, the IR, temperature, and MQ-7 sensors will be utilized for proof of concept. Ideally, in the final product, the temperature sensor will be replaced with an ionization sensor, as the latter is more commonly used in real-world fire and smoke detection. Ionization sensors are generally more sensitive to small smoke particles from flaming fires, whereas temperature sensors detect heat buildup.

Table x: Concept Combination Table

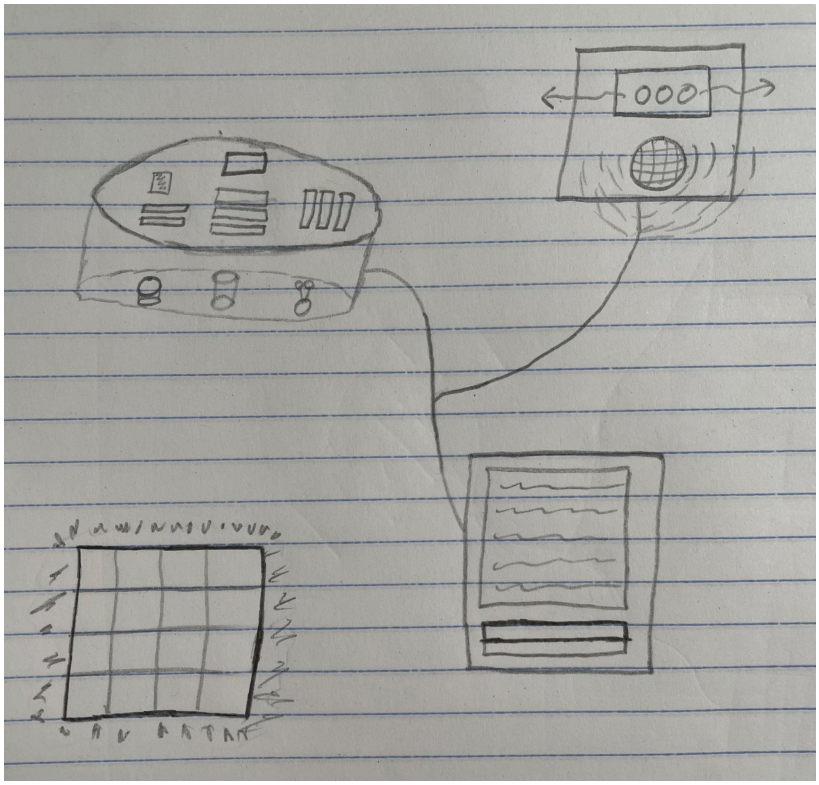
Methods for Signaling	**Methods for Detection**	**Power source**
LEDs	Photoelectric Sensor	Battery
Vibrational Device	Ionization sensor	Electrical socket
Scent Emitter	CO Sensor	

Table 3 above presents a concept combination table created during the design process, detailing three subsystems of the product and various ideas for implementing each subsystem. For the signaling subsystem, the options include LEDs, a vibrational device (such as a smartphone), and a scent emitter. The chosen implementation for this subsystem combines the LEDs and the vibrational device.

For the detection subsystem, the listed options are a photoelectric sensor, an ionization sensor, and a carbon monoxide sensor. The implementation for this subsystem will combine the ionization sensor and the carbon monoxide sensor.

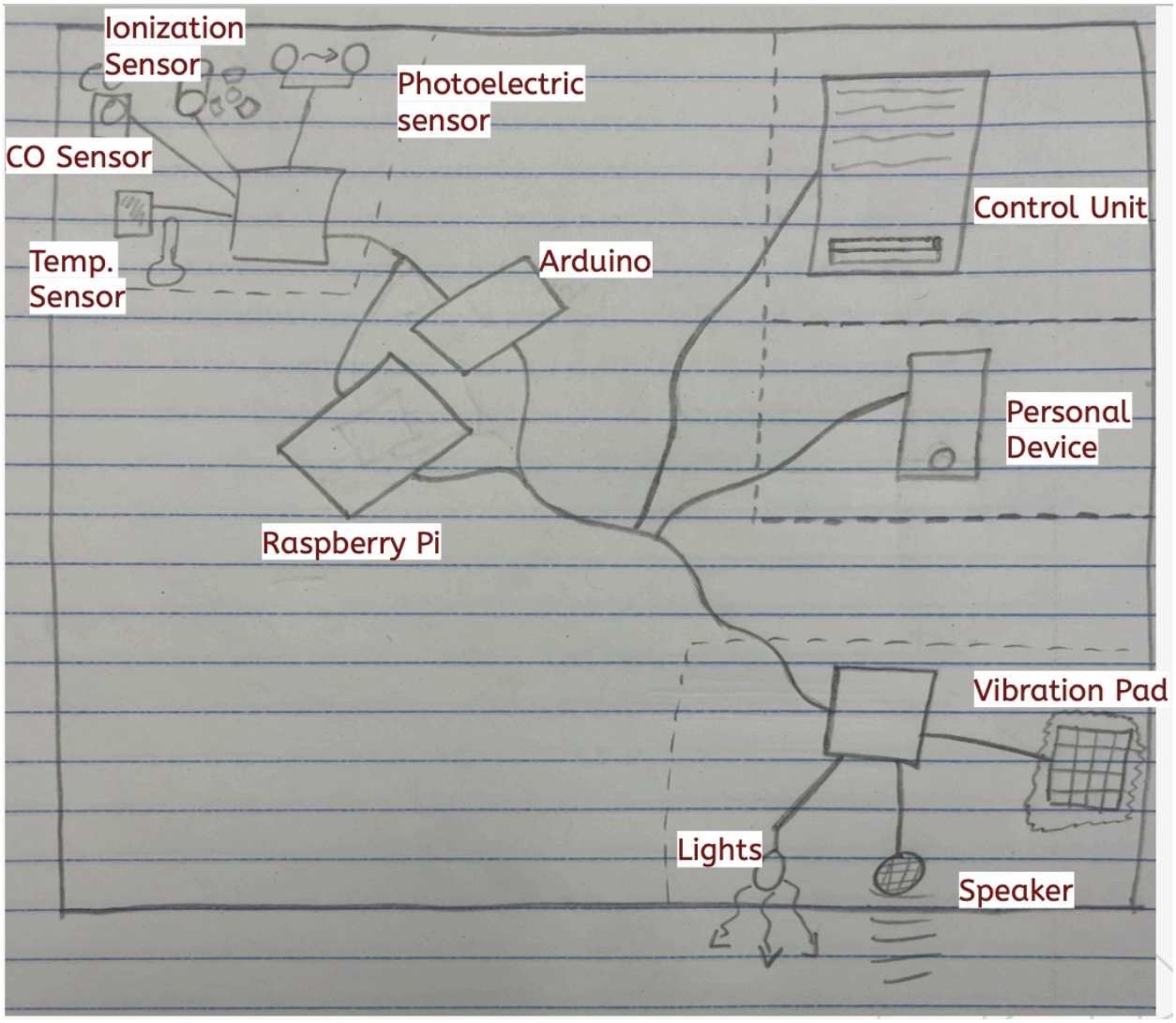
For the power subsystem, the options include a battery and an electrical socket. The battery was selected for the implementation of this subsystem.

Figure 5.1: Final Design Configuration (Hardware components)



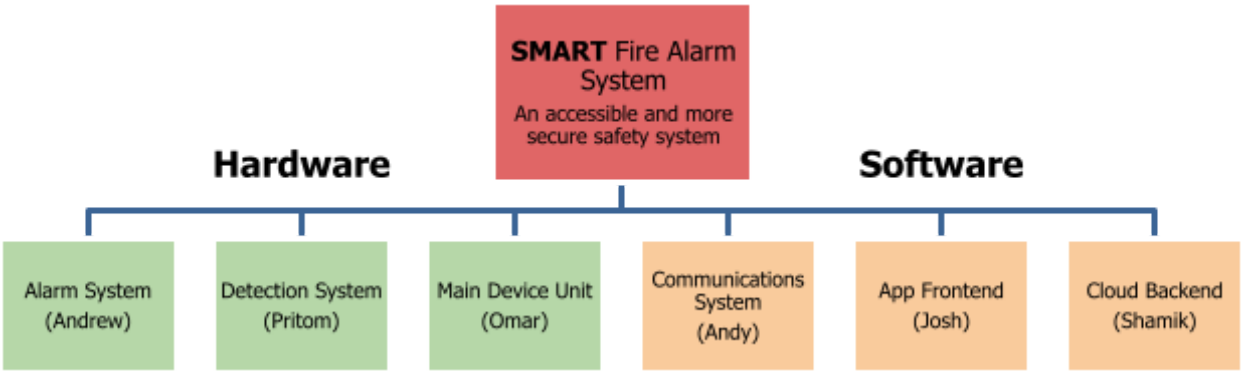
In Figure X, all the main components of the final design are shown. The top left illustrates the main device unit, which would be mounted on the ceiling of the user's home. The top right and bottom right show the alarm system and control unit, respectively. In the final design, these components would be integrated into the main unit but have been depicted separately for visualization purposes. The control unit, specifically, would be placed on the main device unit and used to turn on, turn off, or configure the main unit. Lastly, the bottom left shows the vibrational pad.

Figure 5.2: Final Design Configuration (All components)



In Figure X, all the small components of the design are listed, with each major stage or subsystem outlined for clarity. In the top left, the sensors are shown, followed by the Arduino and Raspberry Pi, which handle most of the computation for the main unit. In the top right is the control unit, and below it is the mobile device on which the app runs. Lastly, the bottom right displays all the physical alarms.

6 Subsystem Analysis and Design



In Figure X above, a diagram illustrates all the subsystems in the final design, which can be divided into two groups: hardware and software.

Hardware Side:

1. **Alarm System:**
    - **Objective:** Implement light, auditory, and vibrational alarms within the main device unit to ensure the customer is alerted to a fire regardless of their location in the home.
    - **Tasks:** Evaluate alarm components for energy efficiency.
2. **Detection System:**
    - **Components:** Photoelectric, ionization, and carbon monoxide sensors.
    - **Objective:** Calibrate sensors for accurate fire detection and ensure continuous functionality.
    - **Tasks:** Implement circuitry to process sensor signals and transmit data to the software end.
3. **Main Device Unit:**
    - **Objective:** Build the physical unit and integrate the alarm and detection systems into the design.
    - **Tasks:** Work closely with the other hardware systems, design CAD models, and ensure the internal layout accommodates all components.

Software Side:

1. **Communications System:**
    - **Purpose:** Manage data transfers between the main device unit, the app, and the cloud.
    - **Tasks:** Implement encryption to secure all device communications.
2. **App Frontend System:**
    - **Objective:** Design an intuitive and user-friendly interface for the mobile app.
    - **Tasks:** Capture user inputs for the backend, display real-time data, and update at consistent intervals.

3. **Cloud Backend System:**
    - **Purpose:** Process data and manage the logic behind the app and device communication.
    - **Tasks:** Parse sensor readings, store data in a database, and provide APIs for the app.

6.1 Subsystem 1: Alerting User[Andrew]

The output subsystem serves to alert the customer the device is triggered and signaling a fire. It does this by receiving input from Subsystems 2 and 3. Subsystem 2 is directly connected to the sound and light output giving a wired signal while the vibrational pad is connected via wifi through the communication between an Arduino and Raspberry Pi. The output devices all trigger at the same time, and will continue until instructed otherwise.

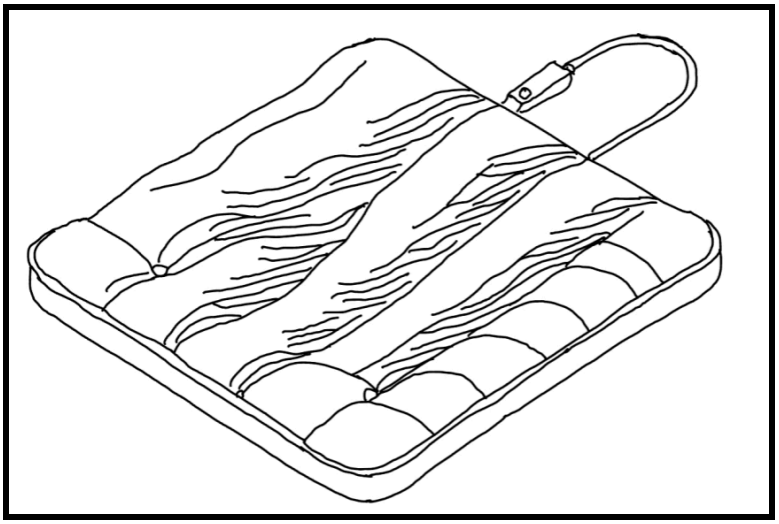
Design Process:

- Output Process:
    - Three different outputs that alert a user via three different senses. Sound and sight were directed using a high powered LED and high pitch speaker that took voltage as input which is placed in the main device. The vibrational pad is connected to an Arduino that receives digital data and sends voltage to the vibrating motors to turn them on.
- Wifi Connection:
    - The Arduino is connected to the main device via wifi with the Arduino device communicating with a Raspberry Pi. This ensures that the devices activate and generate all user alerting outputs at the same time. The output will not disable until receiving additional instructions to do so via App or from the physical main device.

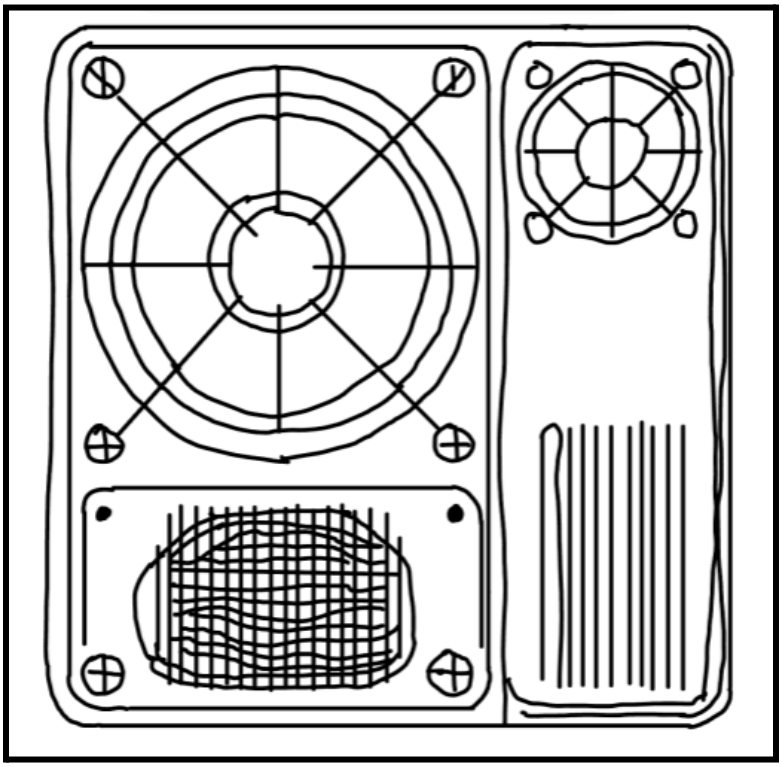
Testing:

- The outputs were tested by simulating inputs from the sensors using a battery pack. All the output devices simply need voltage to become enabled and instantly trigger their user alerting output.

Concept Sketches:



Vibrational Pad Sketch



Light and Speaker Sketch

6.2 Subsystem 2: Detection System[Pritom]

The Detection System utilizes photoelectric and ionization sensors to detect smoke and the CO sensors to monitor air quality, ensuring comprehensive fire detection.

Design Process:

- Sensors process:

    - Three sensors as previously mentioned above have to connected to the main device unit as the sensor is specifically place into/onto the shell of the physical design in order to measure the environment
- Arduino connection:
    - The sensor connects to the arduino uno to take digital/analog input as the data have to be a measure of gradient rather than immediate response. In other words, as the sensor gathers the data, the output response should not only give the data to the communication system but as well letting it know there's a long change in high response time of a fire in the area.
- Communication
    - Arduino uno should be able to communicate and integrate to the Raspberry Pi to allow data to flow through.

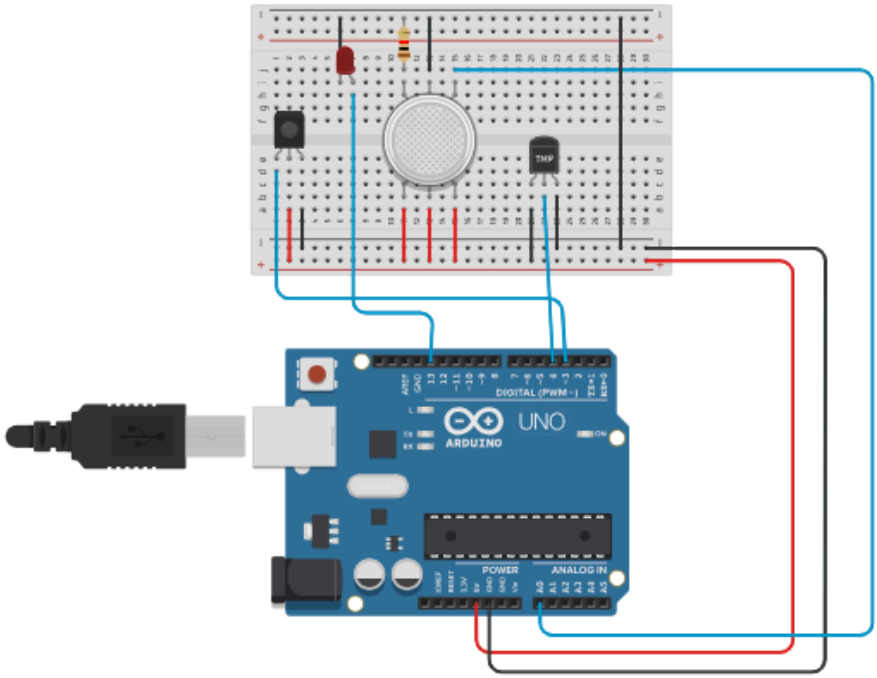


Figure 6.2.1 - Circuits View of Sensors

Figure 6.2.1 contains arduino uno, LED, IR sensor that acts as a photoelectric sensor, Gas sensor that acts as a carbon monoxide sensor, and temperature sensor. Red wire represents power, Blank wire represents ground, and blue wire represents the output signal.

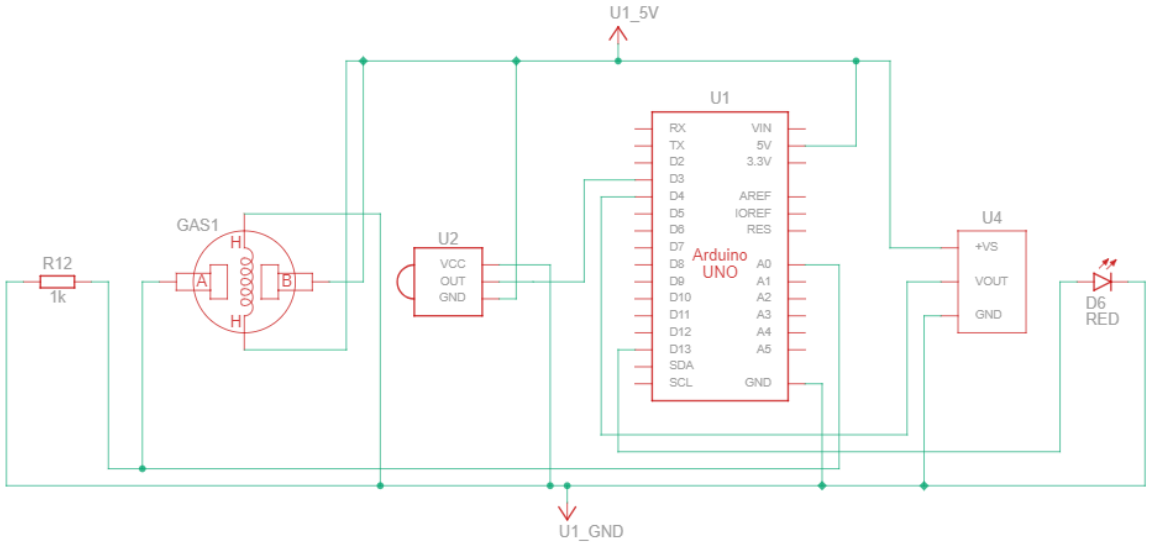
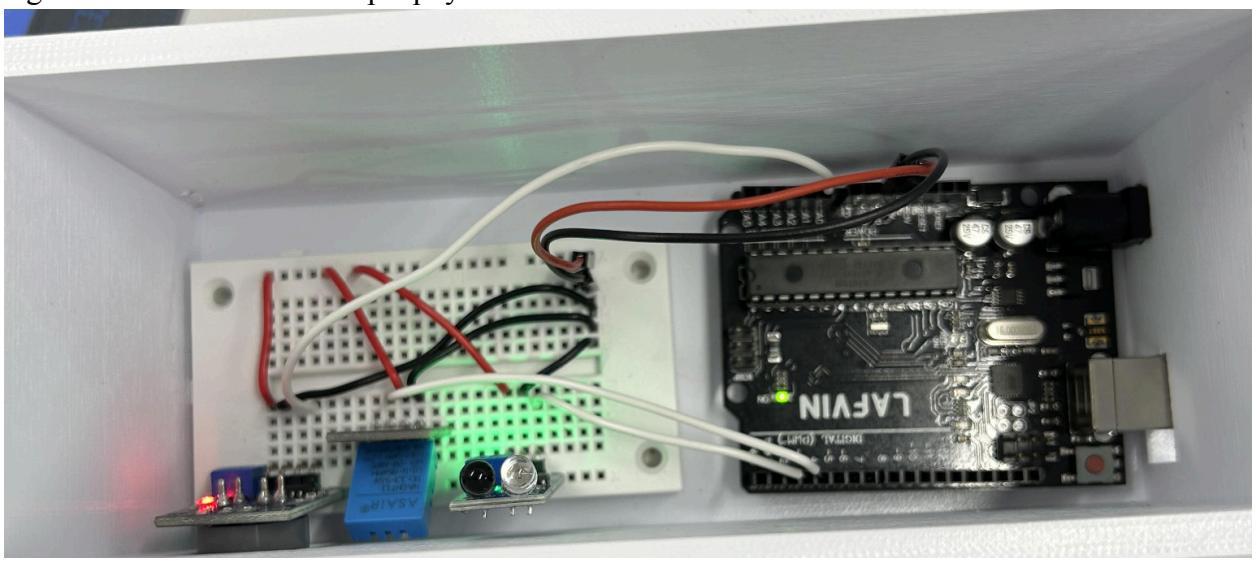


Figure 6.2.2 - Circuits Schematic for Sensors

Figure 6.2.2 is the initial circuit schematics that guide the design process of the input stage. In other words, the schematics of the sensors and the arduino uno serve as a guide of which pins should be connected.

Testing and Demonstration:
The input sensor was tested with the serial output in the Arduino IDE along with a LED lighting from the Arduino itself. In the Figure below, is the physical component that is used. The color of the wire is the same as Figure 6.2.1 with the only exception that blue wire is replaced with white wire.

Figure 6.2.3: Proof of Concepts physical circuit



All code shown in the appendix J for the “Arduino IDE code” as that was written by Pritom using the program in C, as that’s what all arduino code is programmed. In summary, the code will take the inputs from the sensors, check if the value is higher than the threshold, and the output to the json file for the Raspberry Pi to use as well as output the physical and serial monitor. If I’ve added the print statement to check the output during a testing phase, then output would generally look like this when a fire is detected in Figure below.

Figure 6.2.4:

```
Infrared Sensor: 1 | MQ-7 Sensor: 203 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 203 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 203 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 203 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 203 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 204 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 204 | Temperature: 117.86°F
Infrared Sensor: 1 | MQ-7 Sensor: 204 | Temperature: 117.86°F
```

The units would for these test are measure as the following in the respected order: 0/1 | analog output | Fahrenheit. The infrared is designed to be a pull-up resistor where 1 represents as nothing is detecting while 0 represents something is detecting. As such this output does verify our customer requirements as this data will be converted to simple turns to the Raspberry Pi to decode it.

In the early stage, the input stage would contain a photoelectric, ionization, and carbon monoxide sensor. A problem was then faced that there is no ionization sensor in the market, except if you buy a smoke alarm and take that component out of the circuit. However, modern fire alarms have the ionization sensor embedded in a pcb board which therefore, we can’t take the component and know the schematic of the pcb board since its company only, as shown in Table 5.1. As a result, we replace the ionization with a temperature sensor for the proof of concepts. Figure 6.2.1 shows the circuit diagram from a ThinkerCad to create a circuit with an arduino.

6.3 Subsystem 3: Communication System [Andy]

The communication system serves to transfer data between the sensors and the Cloud, as well as signaling the physical alarms to turn on when needed. It mainly consists of a Raspberry Pi 3B+, and an ESP32 Development Board. The Raspberry Pi served as the central communicating unit and was located in the main device, while the ESP32 was directly to the vibrational pad away from the main device.

Design Process:

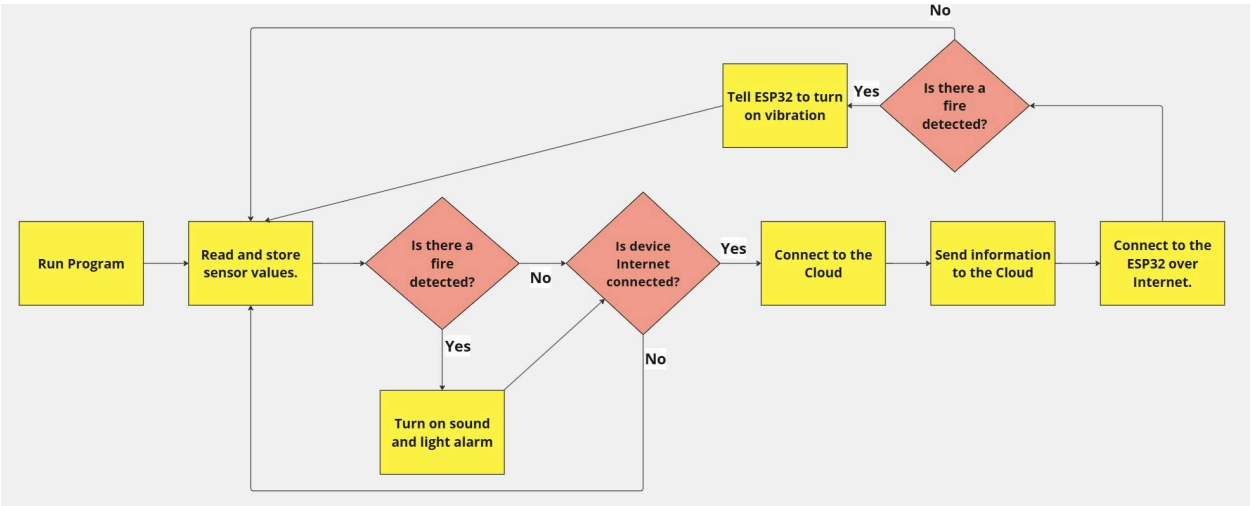
Before delving into the design of both the Raspberry Pi and ESP32 code, the method of communication must be decided upon. For this design, Message Queuing Telemetry Transport (MQTT) was used. It is a lightweight and efficient messaging protocol designed to decrease bandwidth consumption and power usage (*MQTT - the Standard for IoT Messaging*, n.d.).

Therefore, it is ideal for this project, as the vibrational pad needs to consume as little power as possible.

Essentially, MQTT has three main components: the broker, the publisher, and the subscriber. The broker is the central server that manages the message distribution between clients, and in this case it can be run from the Raspberry Pi. Then, the publisher is the client that sends messages to the broker, which is received by the subscriber. In this design, the Raspberry Pi will send a message once the sensors that indicate a fire is detected, and the ESP32 will receive this and activate its vibrational motor.

1. Raspberry Pi Logic:

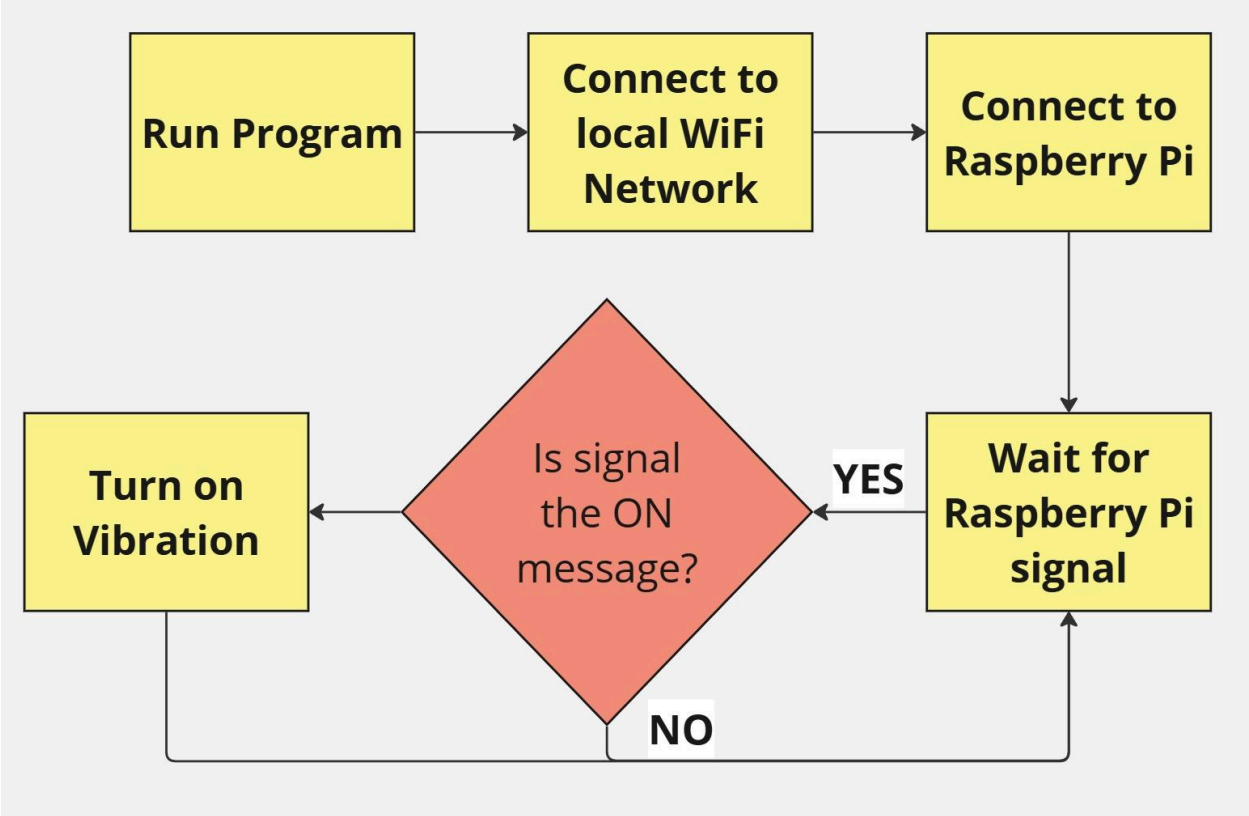
Below is Figure x, depicting the workflow of the program in the Raspberry Pi.



This program will be constantly running when the main device is plugged into power, as well as ethernet if the user wishes. Upon startup, the Raspberry Pi will read the sensor's output, and then check if the values indicate the presence of a fire. If there is a fire, the Raspberry Pi will alert all the physical alarms on the main unit. Then, the Raspberry Pi will check if the device is connected to the internet through either WiFi or Ethernet. If it is connected, all the sensor data will be sent to the Cloud, and the Raspberry Pi will also connect to the ESP32. If the sensor data indicates that a fire is detected, it will send a signal to the ESP32 to turn on its vibrational motor. If the Raspberry Pi is not internet connected, it will only perform the checking of sensor data for dangerous levels.

2. ESP32 Logic:

Below is Figure x, depicting the workflow of the program in the ESP32 board:



Once connected to a battery, the program above will run continuously. It will first connect to the WiFi network that is configured when setting up the Vibrational Pad. Then, it will connect to the Raspberry Pi, and wait for a specific "ON" signal. When that is received, the Vibrational Pad will turn on.

Testing:

To test and debug both programs, various print statements were sent to the terminal to verify if the code was functioning as intended.

Below is Figure x, which is an example command terminal output of the Raspberry Pi code if it has successfully connected to the internet, and has detected that a fire is present.

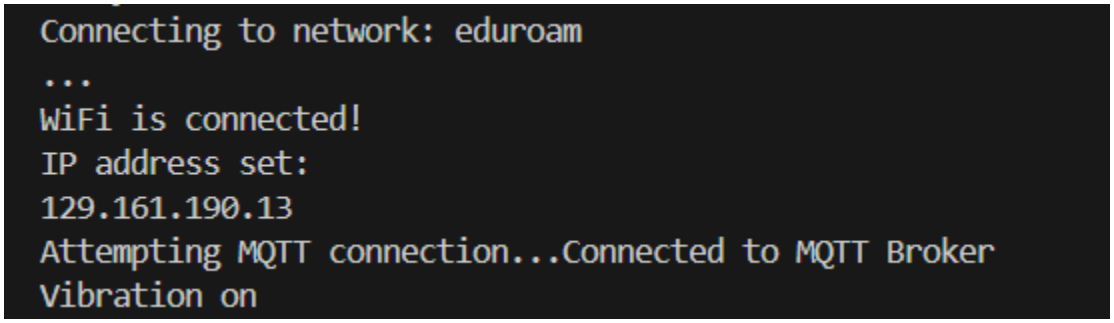
```
FIRE DETECTED, ALARMS ON
 ----------------

Connected to eduroam
Sending the following to AWS IOT:
[1, 199, 113]
Vibration on
```

As indicated by the flowchart, the program first checks if a fire is detected before doing anything else. Once that is done, it will check if the device is connected to the Internet, indicated by the

"Connected to eduroam" message. Once that is done, an array of all three sensor values is sent to the Cloud, which is printed. Lastly the program connects to the ESP32, and since a fire was detected in this instance, the vibration is turned on.

Below is Figure x, depicting an example terminal output of the ESP32 code that has successfully connected to the Raspberry Pi, and has turned on vibration.



As indicated by the flowchart, the ESP32 will first try to connect to the local WiFi network, which was eduroam in this case. Once that is done, it will confirm it, and verify by printing the WiFi network's IP address. Then, it will attempt to connect to the Raspberry Pi using the MQTT broker. Once connected, it will turn on the vibration once it is given the correct signal, indicated by the "Vibration on" text in the terminal.

6.4 Subsystem 4: App Backend[Shamik]

The application backend serves as an interface between the Raspberry Pi which receives the data from the sensors and the application which displays the data to the users. It comprises the server side of the application which handles incoming data from the Raspberry Pi. The Raspberry Pi sends the sensor data through the server to the client-side of the application so that it can be displayed.

Design Process:

Originally, the use of a Node.js server was considered to serve as an interface between the Raspberry Pi and the application. The Raspberry Pi would send the data to the Node.js server via a POST request. An SQL database would be used to store the incoming sensor data. This data would be requested by a GET request via a JavaScript fetch request in the application. This data would then be accessible and displayable by the application.

A serverless solution using the AWS IoT console was decided upon instead. This was decided to be a preferable course of action because the IoT Console could be easily set up on the Raspberry Pi, and the lack of need to manage a server would make testing easier. This solution consisted of three parts.

- Lambda functions
    - The Lambda functions are what get invoked by the Raspberry Pi and the JavaScript application for uploading and receiving the sensor data respectively.
    - There is a Lambda function that maps to the POST request in the API which is invoked by a Python script in the Raspberry Pi. This request uploads the
- DynamoDB Table

    - The AWS DynamoDB table is that to which the POST request writes the data, and from which the GET request retrieves the data.
- REST API
    - The REST API endpoint is how the Python script in the Raspberry Pi and the React application access the REST API to make the POST and GET requests respectively.

Testing

The subsystem was tested by invoking the Lambda function that calls the POST request and seeing if the DynamoDB table receives the inputted data. On invocation of the function, an entry appears in the DynamoDB table showing the infrared sensor reading, the temperature sensor reading, and the carbon monoxide sensor reading.

6.5 Subsystem 5: App Frontend [Josh]

The app frontend is served as the primary user interface for the SMART Fire Alarm System. The development of this subsystem was conducted using React Native, which allowed compatibility across both iOS and Android platforms.

Design Process:

- User Interface (UI) Design:
    - The design process was focused on creating an intuitive and accessible interface. Large buttons and clear fonts were implemented to ensure easy navigation by users of all ages and abilities.
    - The layout was designed to minimize the number of steps required for performing critical actions, such as deactivating the alarm or checking environmental statistics.
- Real-Time Data Display:
    - Data from the fire alarm system is continuously received by the app, with real-time updates being reflected to display the current status of the environment. This ensures that users are kept informed of any potential hazards.
- User Profile Management:
    - A comprehensive profile management system is included in the app, allowing personal information to be updated, multiple devices to be linked, and different languages to be selected. This feature ensures that the app is suited to the needs of diverse households.
- App Concept Sketches:
    - A mobile application was developed for the SMART Fire Alarm System using React Native. It is designed to function seamlessly on both iOS and Android

platforms, providing users with real-time updates on their environment's safety, allowing the monitoring of critical data such as air quality, CO levels, and gas levels from personal devices.

Fig 5.1 - App Color Palette

Fig 5.2 - App Sketches

Key Features of the App:

Home Screen: The current status of the home (e.g., "All Safe," "Caution," "Danger") is prominently displayed at the top of the screen. This allows the environment to be quickly assessed at a glance. A large central button is placed in the middle of the screen, allowing the alarm to be swiftly deactivated in case of a false alarm or emergency situation. A timer is also included on the screen, indicating how long the alarm has been active, providing context for the current alert.

Statistics Screen: Key environmental metrics such as Air Clarity, Carbon Monoxide (CO) levels, and Gas levels are presented on this screen. The data is displayed in large, easy-to-read fonts and updated in real-time to ensure the latest information is always available.

Profile Management: Profile settings, linked devices, language preferences, and profile details can be managed through the app. Multiple users are supported, allowing the system to be monitored by each household member.

Push Notifications: The app is equipped with a push notification system, alerting users even when the app is not actively in use. Notifications are sent to inform users of detected hazards or changes in the system's status.

Testing and Demonstration: The app's functionality was tested across both iOS and Android platforms to ensure consistent performance. Tests included responsiveness checks of the user interface, accuracy verification of real-time data display, and reliability assessments of push notifications.

Link to Code:
https://drive.google.com/file/d/1zhiLAP7hjk-UDS7NP-vysU93iQ1sxkEF/view

All code in the provided zip file was created by Josh Suber. The main file in which the app runs is *App.js*. The screens themselves are stored in the *screens* folder. The *aws-exports.js* file is used to connect the app to the backend. The *ColorContext.js* file manages screen color changes based on the timer, and *TimerContext.js* controls the timer. The *i18n.js* file handles language changes, while the *locales* folder contains JSON files with translations for the app. These files work in conjunction with *i18n.js* to change the app's language.

Screenshots App:

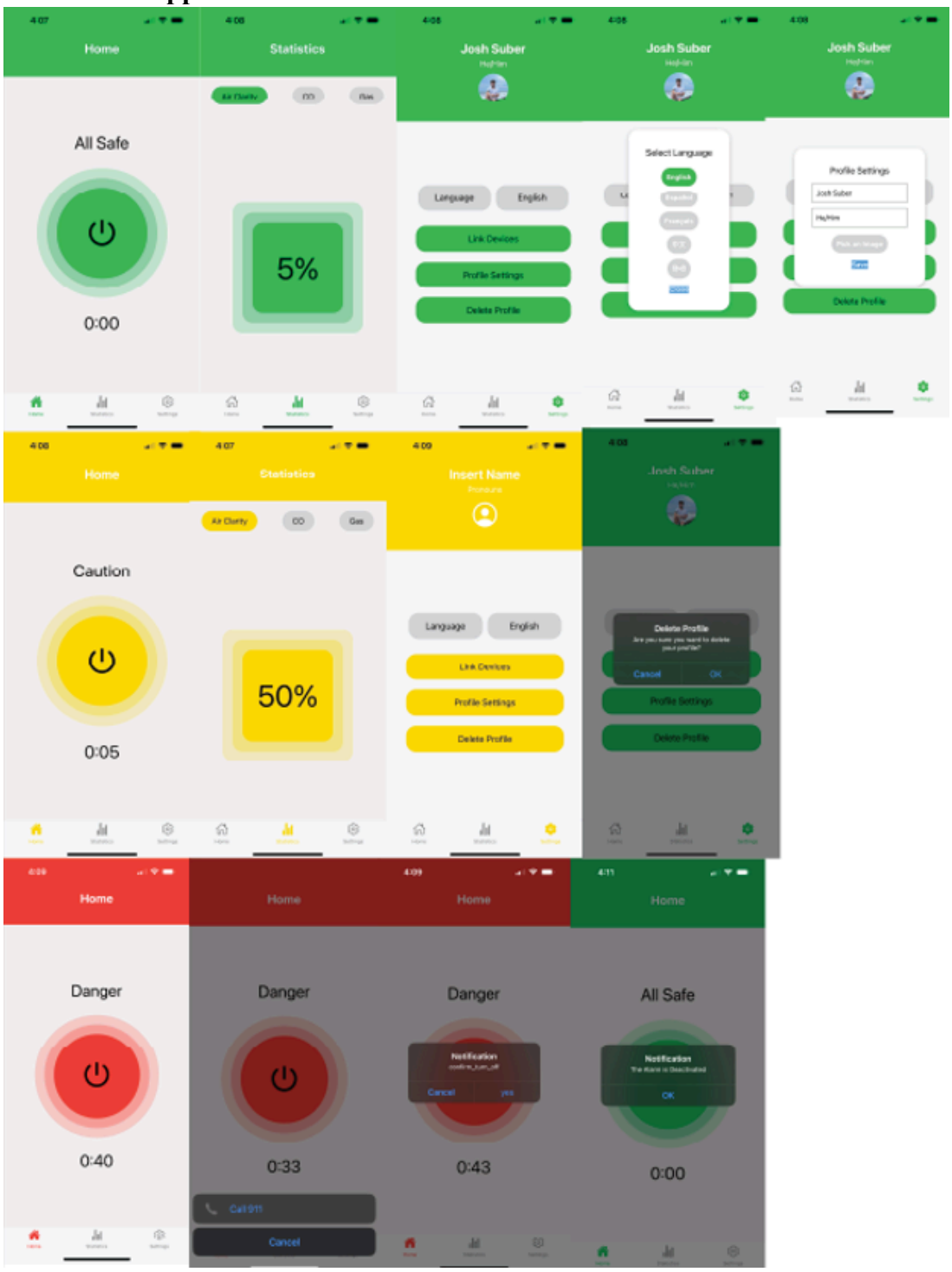


Fig 6.5 - App Screenshots

6.6 Subsystem 6: Outer Shell and Design [Omar]

This subsystem serves as the main containment unit for our physical hardware, as well as a collection for all our subsystems into one connected project. It consisted of a housing unit for our detector hardware/subsystem, a housing unit for our alarm subsystem (lights and sound specifically), and a central access unit/manual control device.

- Material (for all items): PVC
    - Lightweight
    - Sturdy
    - Resistant to Fire
    - Withstand high temperature

Detector Shell:

- Colored white
- Cylindrical Shell shape
    - 6.5 inch diameter
    - 1.8 inch thickness
- Vent holes to let smoke in
- Test button
- Warning light
- Attached to ceiling

Figure 6.6.1: Detector Final Design Sketch

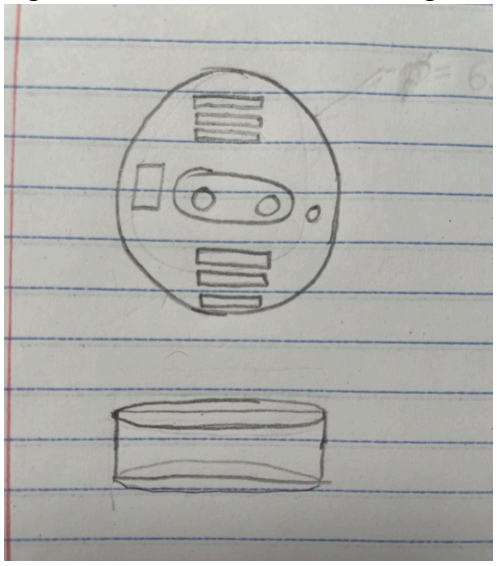


Figure 6.6.2: Detector Proof of Concept Model

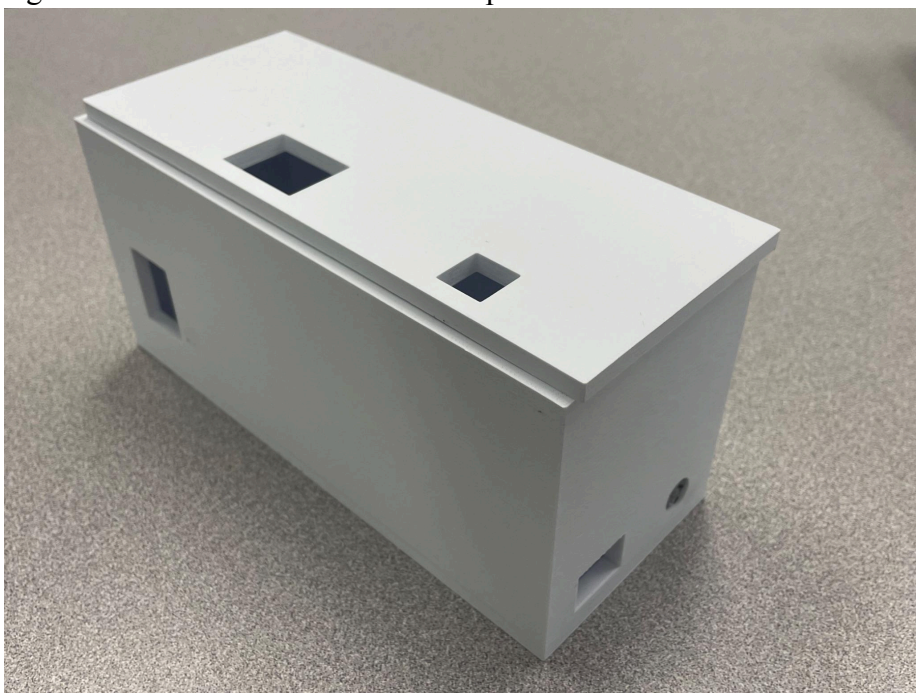


Figure 6.6.2 shows what has been demonstrated as the box has been 3D-printed using PLA filament. The holes have been measured to make connection between the input sensors as well as plugging in the arduino when using a battery. The holes at the top of the box are not only for one of the sensors to detect smoke but also for visual output of the flashing lights as an additional feature.

Alarm Shell:

- Small, compact device
- Material: PVC
- Attached near top of wall
- Colored red
- LED lights for visual alarm (flashing red)
- Speaker included for auditory alarm

Figure 6.6.3: Alarm Final Design Sketch

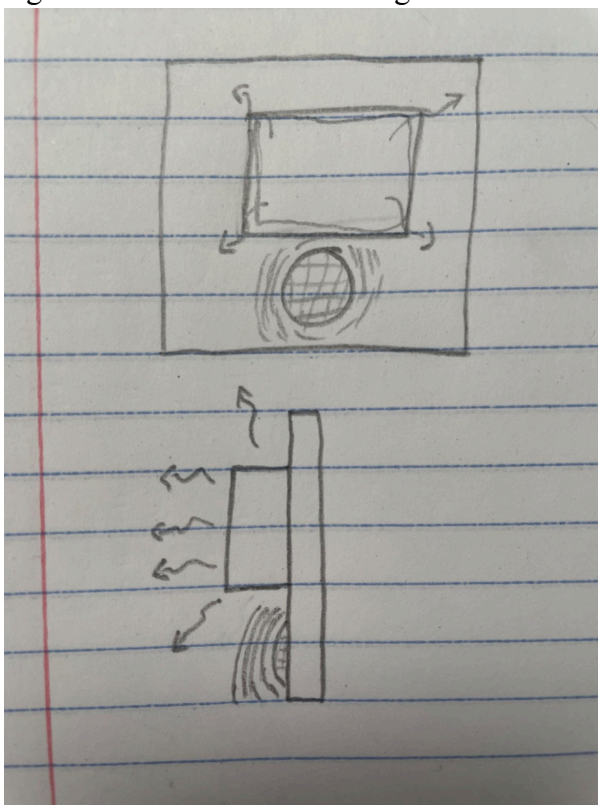
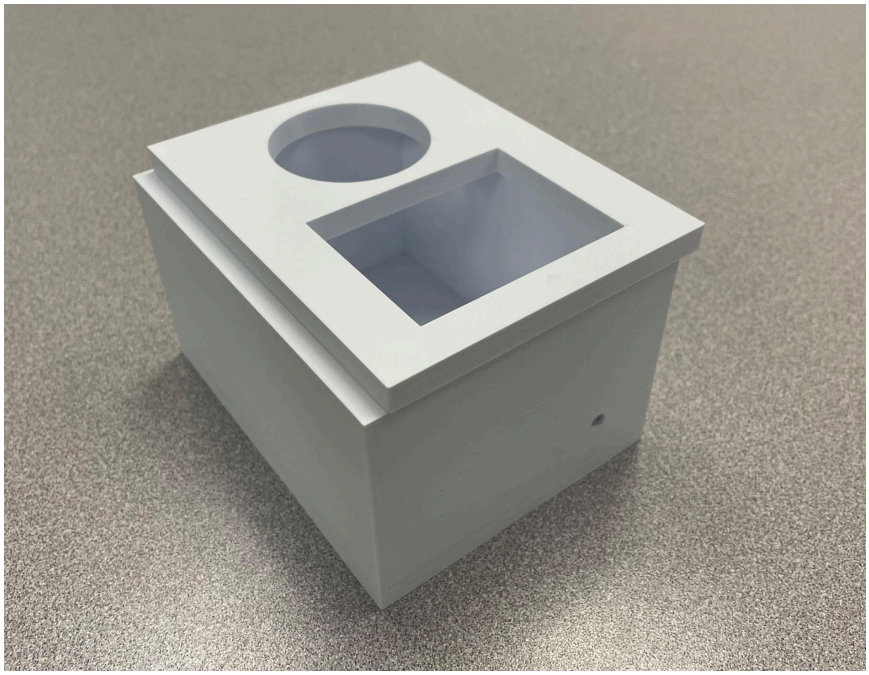


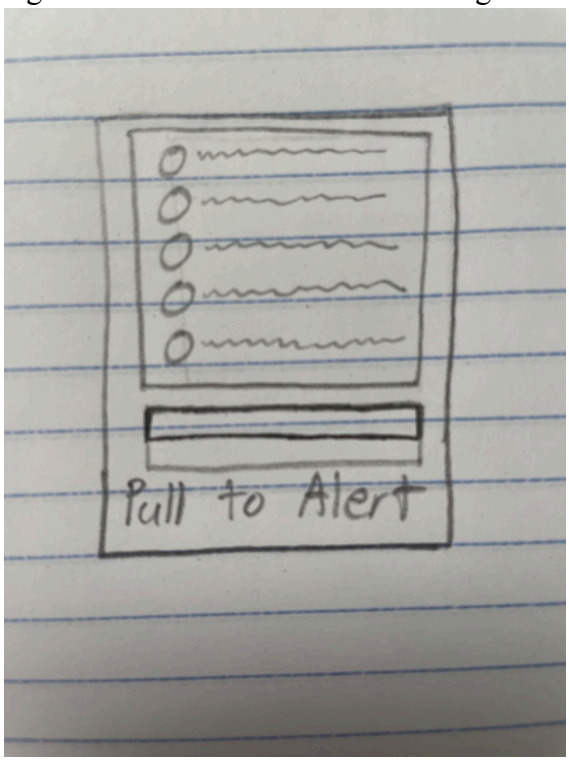
Figure 6.6.4: Alarm Proof of Concept Model



Central Control Unit:

- Small panel placed on wall
- Material: PVC
- Interface screen upon panel
    - Sensors battery
    - Alarms battery
    - Level of smoke in house
    - Level of CO in house
    - State of temp. sensor
- Manual alarm switch

Figure 6.6.5: Control Unit Final Design Sketch



7 Results and Discussion

7.1 Results

Outputs:

The three outputs were successful and met the customer requirements. Each output effectively alerted the user through the targeted senses and was activated by interacting with the other subsystems. These outputs were directly tested using a battery pack to simulate input data from the other subsystems, confirming that each device functioned as expected.

Detection Sensor:

The three sensors successfully met all design criteria and performed as instructed. The output for the final design measured the units as follows:

- **Photoelectric Sensor:** On/Off
- **Temperature Sensor:** Fahrenheit
- **Carbon Monoxide Sensor:** Parts per million (ppm)
- **Ionization Sensor:** Microcuries

Refer to Figure 6.2.2 for an example output of the test. The testing involved turning on the gas stove and allowing it to heat up. The gas smell and heat were used to test the detection sensor system by bringing it close to the stove. The results were printed, and the LED was activated to indicate detection.

App:

The SMART Fire Alarm app successfully met all design criteria and demonstrated reliable performance during testing. The user interface was praised for its simplicity and ease of use. The real-time data display provided accurate and timely information, and the push notification system alerted users to potential hazards even when the app was not actively in use.

Outer Shell:

The outer shell prototype generally met the design criteria. The sensor and alarm components fit within their designated spaces, and there were adequate gaps for most sensors and wiring. However, there were some drawbacks:

- The temperature sensor lacked a proper gap for external attachment, requiring it to be attached to the outside.
- The gap for the photoelectric sensor was not wide enough, causing the alarm to be triggered by the upper cover of the shell.
- The hole for the speaker did not accommodate the speaker's shape, as it was not a perfect circle.
- A prototype central control unit could not be built due to time constraints and was deemed unnecessary for the prototype design.

Overall, while the prototype design was a functional proof of concept, it is not yet ideal for final use. The design was bulky and not suitable for wall mounting. Additionally, PLA plastic was used for the prototype, whereas the final product will use PVC plastic for improved durability and functionality.

7.2 Significant Technical Accomplishments

Outputs:

- **Interacting with Other Subsystems:** The sensor subsystem needed to interface with the sensors directly and via Wi-Fi, especially since the vibrational pad was remote. This requirement was met by integrating Wi-Fi connectivity to enable remote interaction.
- **Simultaneous Activation:** There were challenges in simultaneously activating the two main output devices—inside the main alarm and the vibrational pad. This issue was resolved by using an Ethernet port, which prevented the Raspberry Pi from losing connection and provided clearer signals to the receiver during demonstrations.
- **Soldering:** The output devices were fragile and prone to breaking during testing. Repairing these devices through soldering was difficult and time-consuming, as the small wires were easily damaged. Soldering provided a temporary fix, but the devices often broke again.

App:

- **Cross-Platform Functionality:** Successful development was achieved for both iOS and Android platforms, ensuring broad accessibility for users.
- **Real-Time Data Updates:** The app efficiently handled and displayed real-time environmental data, marking a significant technical achievement with timely updates.
- **User-Friendly Interface:** The app was designed with an intuitive interface, facilitating easy navigation and operation for users of all technical abilities.

8 Conclusions

In this product, a successful fire detection system was developed, featuring multiple alarm sources that target different human senses, thus providing greater accessibility compared to standard smoke detectors. The system was effectively integrated with SMART features, allowing users to interact with their smoke alarm through commonly used mobile devices.

However, several shortcomings remain. Future improvements should focus on reducing compactness and cost. The prototyping and design process was significantly more expensive than for a typical smoke detector, resulting in a final product that is not cost-effective. Additionally, the design is bulkier than standard smoke detectors, leading to issues with crowding and installation space. Custom-made parts could alleviate some of these issues, and once mass production is achieved, the overall cost of the design could be reduced. For future innovations based on this design, custom fabrication will be crucial.

9 References

Combination Smoke & Carbon Monoxide Detectors. (n.d.). Kidde. Retrieved July 11, 2024, from https://www.kidde.com/home-safety/en/us/products/fire-safety/combination-smoke-co-alarms/

Wholesale Home. (n.d.). *BRK First Alert 9120B Hardwired Smoke Alarm, Dual Ionization*. BRK First Alert 9120B Hardwired Smoke Alarm, Dual Ionization. https://www.wholesalehome.com/products/brk-9120b-smoke-alarm-dual-ionization-120v-hardwired-w-battery-backup?currency=USD&variant=6778757123&stkn=42a6a3d82017

MQTT: The Standard for IoT Messaging. (n.d.). MQTT. Retrieved August 11, 2024, from https://mqtt.org/

10 Appendix A: Selection of Team Project

Table 2: Concept Selection Matrix

Criterion	**LED**	**Vibrational Device (via smartphone app)**	**Scent Emitter**
Ease of Installation/Use	+	+	-
Compactness	0	+	0
Durability	+	0	-
User-friendliness	+	+	+
SUM	3	3	-1
Use?	Yes	Yes	No

Table 2 above shows the concept selection matrix created in the design process of the product. The columns of the selection matrix show the three prospective methods for alerting the user of a fire or gas leak. The rows of the selection matrix show criteria called for by the customer needs of the product. The use of LED strip lights show promise in the areas of ease of installation, durability, and user-friendliness. LED strip lights can be easily integrated into a circuit. They are functional as long as the power source to which they are connected is not depleted. LED strip lights are generally bright enough to be seen by anyone who is capable of vision. The vibrational device shows promise in ease of installation, compactness, and user-friendliness. A microcontroller can be used to support communication between the product and a wireless device, allowing for vibrational notifications to be sent to the user. The scent emitter proves to be suboptimal in the areas of ease of installation and durability. A fan to disperse the scent cannot be easily integrated into the rest of the product, and any scented fluid released by the product would need to be replaced with some frequency. Therefore, it was rejected for the final design.

Table 3: Concept Combination Table

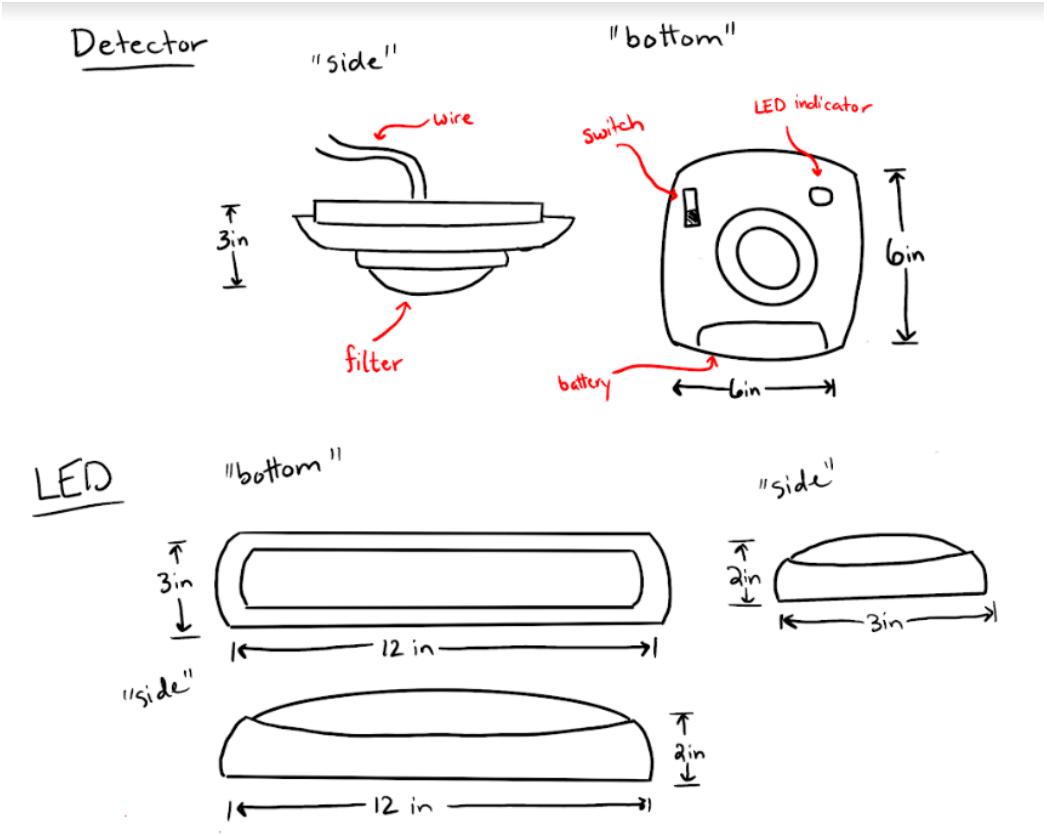
Methods for Signaling	**Methods for Detection**	**Power source**

LEDs	Photoelectric Sensor	Battery
Vibrational Device	Ionization sensor	Electrical socket
Scent Emitter	CO Sensor	

Table 3 above shows a concept combination table created during the design process. They show three subsystems of the product and different ideas for the implementation of each subsystem. For the signaling subsystem, LEDS, a vibrational device (smartphone), and a scent emitter are listed. The implementation for this subsystem would be a combination of the LEDs and the vibrational device. For the detection subsystem, a photoelectric sensor, an ionization sensor, and a carbon monoxide sensor are listed. The implementation of this subsystem would be a combination of the ionization sensor and the carbon monoxide sensor. For the power subsystem, a battery and an electrical socket are listed. For the implementation of this subsystem, the battery was selected.

Hardware Sketch

Our design features a compact and user-friendly smoke and carbon monoxide detector. The device is designed to be mounted on the ceiling with easy integration into existing electrical systems. It includes a filter and battery compartment for straightforward maintenance and replacement. The switch and LED indicator at the bottom provide clear feedback on the device's status.



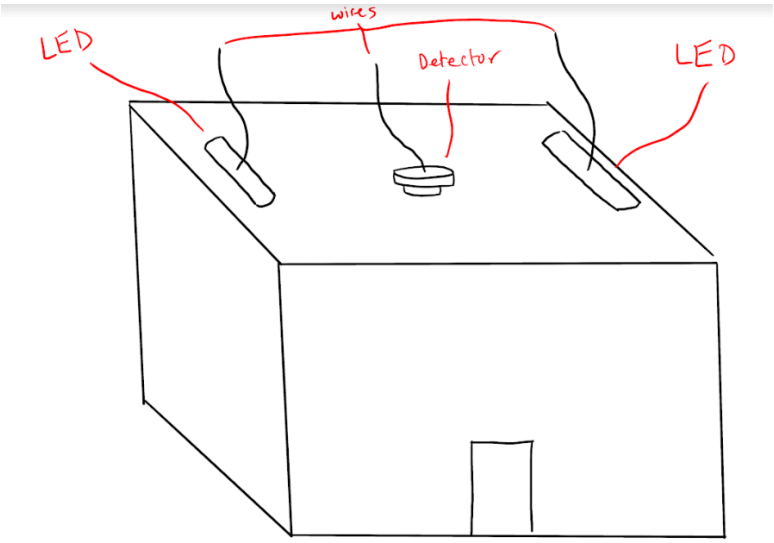


Figure 3: Hardware Sketch of the SMART Fire Alarm System

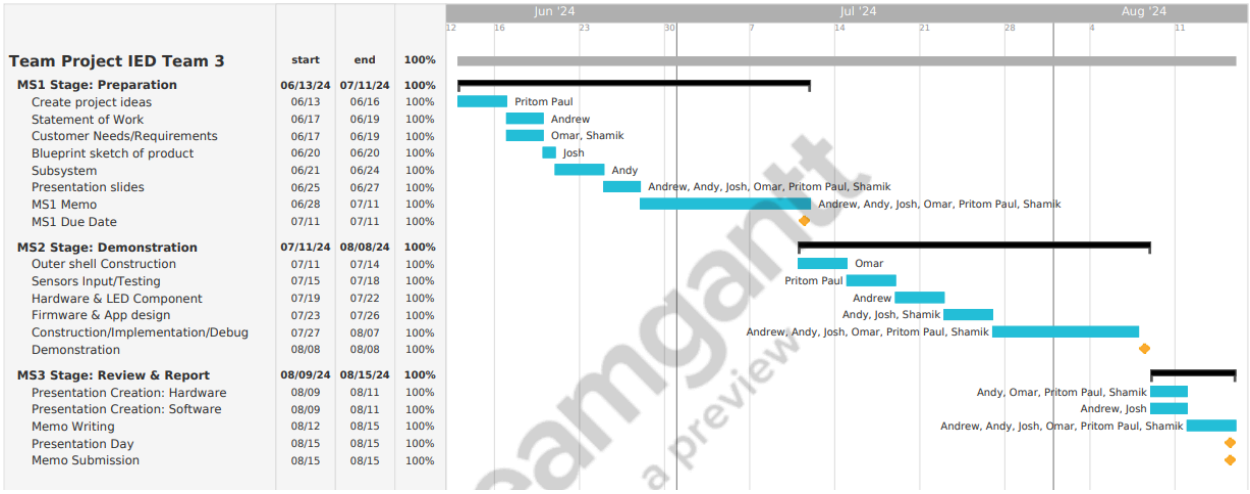
11 Appendix B: Customer Requirements and Technical Specifications

Customer Requirement	**Technical Specification**	**Target Value / Range of Values**
Compact	Volume	3 inches height, 12 inch diameter
Doesn't hurt users	Safety	0 casualties
Non-toxic to the environment	Not Hazardous	Nearly 0 Radioactive Material
Range of smoke detection	Range Radius	25m
Easy to operate	Age	~10+ years
Convenient/Simple	Number of moving parts.	1
Lightweight	Weight	< 1lbs
Backup Battery power	Volts	9V
Inexpensive for a homeowner perspective	Dollars per person	$350 for a pack of 6
Consistent	Time before failure arrives	5 years
Low risk of shock	Percent chance of getting shocked	< 1%

Customer Requirement	Technical Specification	Target Value / Range of Values
Capable of Alerting User	Sound Lights Vibrations	Alarm produces between 65 to 120 decibels 150 candelas 130-180Hz
Integrated App design to control the alarm remotely	IOS/Android Compatible	Compatible with devices running iOS 10.0+ and Android 7.0+

12 Appendix C: Gantt Chart

Chart 12.1:



The Gantt chart was provided by TeamGantt, which was preferred over creating one from scratch in Excel due to its ease of use. The watermark on the image cannot be removed, so it will remain.

The Gantt chart was utilized to list tasks and milestones, assigning them to specific group members along with progress levels and completion dates. This approach facilitated clear visibility of tasks that needed to be completed, helping the team stay organized and on track.

13 Appendix D: Expense Report

Table 5 - Project Expenses

Item	Quantity	Unit Price	Subtotal	Shipping
Lights and Speaker	2	10	20	0
Bonatech MQ-7 Sensors Modules	1	19	19	0
IR Infrared Sensors	1	7	7	0
DHT sensor	1	3	3	0
3D printed items	1	16.13	16.13	0
Total	6	65.13	65.13	

14 Appendix E: Team Members and Their Contributions

14.1 Pritom Paul

The responsibilities of the lead developer for the SMART Fire Alarm System's detection system included ensuring that the input system was correctly implemented on the Arduino, where the software and hardware supported each other. Additionally, it was necessary to design a mini-output result in the terminal output and the physical design of the product as an indicator that the sensor detected something. Close collaboration with both Omar and Andy was required, as the input stage necessitated interaction with Omar, who was responsible for the physical design of the SMART Fire Alarm, and Andy, whose subsystem involved the output communication line used for his input stage. Lastly, contribute to make the document as organized as well as prepare the outline for the presentation.

Throughout the project, assistance was provided to Omar, particularly with the physical design, measurements, and offering tips about 3D printing. Omar was responsible for not only building the physical shell of the product but also correctly measuring the openings for the Arduino and the three sensors, so telling Omar what's the measurement for certain circuits and inputs. The final design implementation, including the physical connection of the Arduino circuits along with the sensors, as well as resolving any issues that arose from gluing and combining two separate subsystems, was carried out. Additionally, assistance was given to Andy with the communication aspects between the Arduino output and his Raspberry Pi input. In this regard, code was added to facilitate easier data reading for his subsystem.

14.2 Josh Suber

As the lead developer for the SMART Fire Alarm System's app frontend, primary responsibilities were undertaken in designing and implementing the user interface to allow effective interaction with the fire alarm system. The app was developed using React Native, ensuring that compatibility was maintained across both iOS and Android platforms.

Key Contributions:

User Interface Design: The app's interface was designed with a focus on accessibility and ease of use. A visually appealing layout featuring large buttons, clear text, and intuitive navigation was created. This design ensured that the app could be quickly understood and operated by users, even in emergency situations.

Cross-Platform Development: React Native was utilized for the app's development, allowing a single codebase to be maintained while ensuring compatibility across both iOS and Android devices. Efforts were made to ensure consistent functionality across these platforms, considering different device specifications and operating system requirements.

Real-Time Data Display: Real-time data from the fire alarm system, including air quality, CO levels, and gas levels, was displayed as a significant aspect of the app. Integration with the backend was performed to ensure continuous updates, providing users with the most current information at all times.

Testing and Validation: Testing efforts for the app were led, focusing on usability, functionality, and performance. The app was tested across various devices and operating systems to identify and resolve any issues. Responsiveness and accuracy in displaying real-time data were also validated during testing.

Through these contributions, a seamless, user-friendly experience was ensured for the SMART Fire Alarm System, enhancing the overall functionality and accessibility of the fire alarm system.

14.3 Andrew Zheng

Andrew's main role for this project was the designer for the output devices and ensuring that the devices when triggered properly alerted the client. This was done with 3 output mechanics including sound, sight, and touch. This was done by using a high pitch speaker for sound, an LED for light, and a vibrational motor for a tactile touch alert.

While creating the output devices, Andrew worked with Pritom for the sensors subsystem, Omar for the outer shell subsystem, and Andy for the communications subsystem. The sensors subsystem had to be compatible with the output devices in the main shell to create an input output signal that could easily trigger the output at the same time as the sensors triggering. Omar created the outer shell which had to be compatible in fitting the output subsystems as well as designing it to be efficient to generate best possible results. Andy was in charge of communications which was directly linked to the output subsystem since the output devices all needed to trigger together, as well as link to the other subsystems and receive the proper data to trigger. This was done via wifi with a Raspberry Pi and an Arduino device.

These contributions to the device were imperative to create a working fire alarm system as well as a proper and user-friendly final product.

14.4 Omar Alsofi

Omar's main role on this project was the creation of an outer shell for the alarm system, housing unit that all the physical components would be placed into. This includes

- A shell for the detector component
- A shell for the alarm component
- A shell for main control unit

When designing the outer shell, Omar worked closely with other members of the team, specifically Pritom, who worked on the detector system, and Andrew, who worked on the alerting system. This was done so the shells themselves could properly house the required components, and the required gaps were left in to allow proper functionality of the device.

Beyond the outer shell, Omar contributed to the research involved with other components of the system, specifically on the types of sensors that would be used and how they functioned. This improved the team's understanding of these sensor types, helping with their implementation into the system itself. Omar also assisted Pritom in looking up sensor parts to order online, helping in researching different parts to order, their functionality and their price.

14.5 Shamik Chatterjee

Shamik's role on the project was the design of the backend for the application portion of the project. He wrote the Lambda functions which would serve as the interface between the Raspberry Pi and the React application. This included a POST function, which would take sensor data sent to the API endpoint via a Python script on the Raspberry Pi and uploaded to the DynamoDB table on the AWS IoT console. It also included a GET function which would be called by the React application via a JavaScript fetch command to retrieve data from the DynamoDB table. He also set up the DynamoDB table which would be used to store the data sent to the IoT console via the POST Lambda function. Finally, he set up the REST API endpoint which would be accessed by both the Raspberry Pi script and the React application.

In addition, Shamik assisted Andrew in getting the Raspberry Pi connected to the AWS IoT console as well as setting up the server running on the Raspberry Pi. Finally, he assisted Pritom in testing the infrared sensor and the carbon monoxide sensor.

14.6 Andy Huang

The main role that Andy held was developing the Communications System, as well as assisting with the development of the vibrational pad, and integrating the communications system with the rest of the main unit. Development of the communications system began with finding a microcontroller or microprocessor that could interact with the Arduino from the sensor system, as well as communicate by WiFi with the Cloud. While another Arduino would have sufficed with its WiFi capabilities, a Raspberry Pi was ultimately chosen for its ease of use and prior experience that Andy had with the device. The Raspberry Pi would also be better suited to handle communicating with the Arduino and sending information to the cloud at the same time. This communication process was done using MQTT, as described in the subsystem section. Communicating with the Arduino using the Raspberry Pi was done with serial communication, which involved a simple USB connection. After this was done, development on the vibrational pad began.

Since the vibrational pad had to connect to the Raspberry Pi wirelessly, the microcontroller attached to it again had to be WiFi capable. The board also needed to consume little power, as the pad had to be battery powered for convenience. For this, the ESP32 Developer Board was selected. This was a common low cost low power board on the market

with both Bluetooth and WiFi capability. Once configured to connect to the Raspberry Pi with MQTT, it was ready to be connected to the vibrational motor in the pad.

15 Appendix F: Statement of Work

Team
Josh Suber, Pritom Paul, Andy Huang, Andrew Zheng, Shamik Chatterjee, Omar Alsofi

---

Semester Objectives

1. Design a SMART Fire Alarm system that integrates advanced accessibility features, including multi-sensory alerts for individuals with disabilities.
2. Ensure the SMART Fire Alarm system fully meets customer requirements in terms of real-time monitoring, remote control, and emergency response.
3. Design the system to be scalable and modular, allowing for easy integration of additional sensors or functionality.
4. Develop a system that provides an innovative and user-friendly solution to current fire alarm systems, focusing on accessibility for all users.
5. Address the challenges related to the compatibility of the system with different mobile platforms and the varying levels of technical expertise of users.
6. Enhance existing fire alarm designs by introducing new features like automated emergency calling and multi-lingual support.
7. Obtain an ‘A’ grade on the final project.
8. Ensure that the final selected design reflects the entirety of the engineering design process, from initial concept through testing and final implementation.
9. Leverage the individual strengths of the team members, such as programming, hardware design, and user interface development, to create a cohesive and functional product.
10. Improve the technical and teamwork skills of the group members through collaboration and problem-solving.

Approach

Utilize effective benchmarking to analyze competing fire alarm systems and identify key areas for improvement, particularly in accessibility and user interaction. Use this research to develop an optimized design that balances simplicity and functionality. Focus on creating a mobile app that integrates seamlessly with the fire alarm system, offering real-time data, multi-sensory alerts, and remote control features. Employ the engineering design process to iterate on the design, addressing user feedback and ensuring compatibility across multiple platforms. Leverage the diverse skill sets of the team to prototype, test, and finalize the system.

Deliverables and Dates

1. Milestone One: System Concept Review (7/16)
2. App Prototype Design (7/23)

3. System Prototype Fabrication and Review (7/23)
4. Modifications and Testing (7/30)
5. Milestone Two: Project Demonstration (8/8)
6. Milestone Three: Team Presentation (8/15)
7. Milestone Three: Final Report (8/16)

16 Appendix G: Professional Development - Lessons Learned - Everyone

16.1 - Josh Suber

Keep:

Team Communication: Regular communication was maintained through team meetings and updates, which ensured that everyone remained aligned and informed, contributing significantly to the smooth progression of the project.

Task Delegation: Roles were clearly defined, allowing each team member to focus on specific tasks, such as the app development that I was assigned. This prevented overlap and confusion, improving overall efficiency.

Problem:

Time Management: Although communication was well maintained, some tasks took longer than expected, particularly during the testing phase of the app. Better time estimation and earlier deadlines for testing could have allowed more time for resolving unexpected issues.

Iterative Feedback: In the early stages, user feedback was not incorporated quickly enough. Necessary adjustments could have been made earlier in the design process had feedback been integrated sooner.

Try:

Prototyping Earlier: Low-fidelity prototypes could have been started earlier in the design phase, which would have allowed user feedback to be gathered sooner, leading to improvements in the final app's functionality.

More Frequent Testing: An increase in the frequency of testing throughout development could have caught issues earlier, ensuring that the final product was more polished.

Overall, the importance of communication, proper time management, and early feedback integration was learned during this project in developing a successful product.

16.2 - Pritom Paul

Keep:

- Team Communication: Continue to have regular weekly meetings with everyone as well as having individuals meetup with people so things can get done. In addition, continue to talk with everyone to ease the tension of the group.
- Detail Oriented: Documentation as well as meeting the criteria of the plan should always be the number 1 priority as we want to make sure all the materials is mentioned

Problem:

- Time management: Due to the amount of work for other classes, some things couldn't be worked on a timely schedule. As such, there's should be a improvement into have a work schedule with doing the task as well as multitasking
- Documentation split: In the writing assignment such as the final report as well as presentation, I tend to do more writing work than others so I need to learn how to split it fairly among members or focus them to take a role/action into filling out the documents.

Try:

- PCB design: Make the circuits look more in a professional item where the circuits and the inputs are embedded into a single circuit board (the pcb). In addition,

Overall, this team project led me to understand that things can get done if you collaborate with your groupmates and help each other with proper scheduling.

16.3 - Omar Alsofi

Keep:

- Meeting: Having regular meetings with the others on our current progress and what we need to do next definitely helped with this project, as it gave us a sense of where we currently are, where we need to put more work in, and how we can assist each other to get everything done.

Problem:

- Writing assignments: I fear that during this project I may have left a large chunk of the writing assignment to be done by my groupmates, and in the future I should work to get more involved to lighten the workload of my teammates.

Try:

- Getting more involved: During this semester I tried getting more involved with the others, inputting my ideas more and collabing more, and while I have made a bit of progress I still feel I have to work to get more involved with the work I perform.

16.4 - Shamik Chatterjee

Keep:

- Bring good energy into meetings. Having a positive energy to meetings can help maintain good relations among group members.
- Be amenable to the ideas of others. Listening to others can help to create an environment of collaboration.

Problem:

- The subsystem on which Shamik worked did not successfully integrate with the other subsystems of the project by the day of the demonstration. To prevent incidents like this, lines of communication should be open with the other subsystems to ensure that the subsystems work together.

Try:

- Be punctual to meetings. Shamik's lack of punctuality could have affected his effectiveness as a team member. Shamik should make an effort to honor agreed upon meeting times by following them more closely.

16.5 - Andrew Zheng

Keep:

- Meeting: Meetings were one of the most important, consistent things that we did in this group. I felt that by meeting every Wednesday and being consistent with our schedule, we had a good sense of where everyone was in the project and allowed everyone to be productive for a period of time as well as designate a proper time to work on the project rather than potentially having group members leaving it till the end.

Problem:

- Documentation: I felt that our group lacked the best way of documenting our work as reflected by our memo grades in the past and presentation. This could be because none of us are particularly punctual and good at documentation but it's something to work on.
- Time Management: We left a lot of tasks to be completed at the last minute. This was mainly due to the troubling timing of the summer semester with finals, homework assignments, and the project all being due within the same week. This caused many academic troubles and struggles between all of us which was not ideal for our collaboration but overall it was handled as best as we could in the moment.

Try:

- Leadership: I felt I didn't take much of a leadership role when it came to the project. I didn't declare much about the details of when we'd meet and work together. This was often left to the other group members. While I was a good team member and did all my work, and always showed up to meetings, I didn't do much of the initial spearheading for our group.

16.5 - Andy Huang

Keep:

- Frequent meetings and check ins: With the Arch semester and loaded schedules of some members, it can be very hard to contribute and work on a group project. I felt that by consistently scheduling meetings and doing check-ins with other members, it greatly helped move the project along. If anyone was behind on their work or was confused on

what to do, it was a lot easier to address it and offer help than to wait till it becomes too late to fix their issues.

Problem:

- Lack of technical planning: In the beginning of the project, there were many details that were brushed off to be figured out later. While that is often true in the beginning of most projects, at the latter end of the project, it came back to hurt the group. It was clear that some of the members were not on the same page, leading to confusion amongst the group.
- Being too forward: At the beginning of the project, I noticed that many of the group members were a little quieter and not as eager to speak in some discussions. As someone who naturally doesn't have trouble speaking to both new and old faces, I took charge and spoke in many discussions. As time progressed, the group dynamic was kind of built around it, leading to more of the creative decisions being on my shoulders. For me that was more than I could bear, and became difficult to juggle alongside my own commitments.

Try:

- Being less outspoken: While I am very good at voicing my opinion, it is clear that some people do not have the same luxury. By resisting the urge to be more outspoken, it should allow other group members to contribute and voice their opinions freely. My voice wouldn't be suppressed, more like being toned down to the level of those around me.

17 Appendix H: Software / Technology Used - Everyone

17.1 Collaboration Among Team Members

- Cisco WebEx Teams and Meetings
- Discord
- Google Drive

17.2 Subsystem Design

- OnShape CAD
- ThinkerCAD
- TeamGantt

17.3 Programming

- Visual Studio Code – Integrated Development Environment
- React Native
- Expo Go
- Android Studio
- Arduino IDE
- Thonny Python Editor
- PlatformIO
- AWS Lambda

17.4 Subsystem Testing/Simulation/Emulation

- OnShape CAD
- 3D printer
- Expo Go
- Android Studio
- Visual Studio Code
- Arduino IDE
- AWS Lambda
- AWS DynamoDB
- AWS Gateway

18 Appendix I: User Manual

To Do: Present the procedure to operate your prototype. Assume that the reader is not one of your instructors or part of your team and has never seen the device operated.

Make sure to include all safety related instructions.

Operating the SMART Fire Alarm App:

1. Setting Up the App:
    a. Upon installation, open the SMART Fire Alarm app on your iOS or Android device.
    b. Follow the on-screen instructions to create a user profile. Enter your name, preferred language, and any other required information.
2. Linking Devices:
    a. Navigate to the "Link Devices" section under the Profile Management menu.
    b. Select “Add Device” and follow the prompts to connect the app to your SMART Fire Alarm system. Ensure that your mobile device is connected to the same Wi-Fi network as the fire alarm system for the initial setup.
3. Navigating the Interface:
    a. The Home Screen displays the current status of your home environment. Tap the central button to deactivate the alarm if necessary.
    b. The Statistics Screen provides real-time updates on Air Clarity, CO levels, and Gas levels. Swipe left or right to switch between different metrics.
    c. Access additional settings and profile management options from the Profile screen by tapping your user icon.
4. Receiving Notifications:
    a. Ensure that push notifications are enabled in your device settings to receive alerts about any changes in your home’s safety status. Notifications will inform you of potential hazards or system alerts even when the app is not open.

19 Appendix J: Software Code

Zip File for Mobile App Code [Josh]:
https://drive.google.com/file/d/1zhiLAP7hjk-UDS7NP-vysU93iQ1sxkEF/view

Arduino IDE code use for Detection Sensors:

```
#include <DHT.h>
#include <ArduinoJson.h>

const int infraredSensorPin = 3;
const int mq7SensorPin = A0;
#define DHTPIN 4

//Define threshold for MQ-7 & DHT sensor
const int coThreshold = 300;
const float tempTreshold = 115.0; //135

//Inilitize the DHT sensor
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);

  //Initialize pins
  pinMode(infraredSensorPin, INPUT);
  pinMode(13,OUTPUT);
  pinMode(12,OUTPUT);

  dht.begin();
}

void loop() {
  //Read the infrared sensor value
  int infraredValue = digitalRead(infraredSensorPin);

  //Read the MQ-7 sensor value
  int mq7Value = analogRead(mq7SensorPin);
```

```
  //Read the temperature from DHT11 sensor
  float temperature = dht.readTemperature(true);

  //check if either sensors detects something
   if (infraredValue == LOW || mq7Value > coThreshold || temperature >
tempTreshold) {
    //Turn on the LED
    digitalWrite(13,HIGH);
    digitalWrite(12,HIGH);
    delay(10);
  } else {
    //Turn off the LED
    digitalWrite(13,LOW);
    digitalWrite(12,LOW);
    delay(10);
  }
  //Print sensor values for output and raspberry pi
  StaticJsonDocument<100> doc;
  doc["Photoelectric"] = infraredValue;
  doc["Carbon value"] = mq7Value;
  doc["Temperature"] = temperature;
  Serial.println();

  serializeJson(doc, Serial);

  //Small delay to avoid rapid switching
  delay(500);
}
```

Python Code for Raspberry Pi:

```
import subprocess #check if the rasp pi is connected to wifi
import time #Timing

import serial #Arduino to Rasp Pi Communication
ser = serial.Serial('/dev/ttyACM0', 9600, timeout=1)

#Function below checks if raspi is ethernet connected
def is_ethernet():
    try:
                result = subprocess.run( ["ip", "link", "show", "eth0"]
,capture_output = True, text = True)
        print("ETHERNET------------")
        return "state UP" in result.stdout
```

```
    except:
        return False

#Function below checks if raspi is wifi connected
def is_wifi():
    try:
                                    result    =    subprocess.check_output(
["iwgetid","-r"]).strip().decode('utf-8')
        print("WIFI -----------")
        return (result == "eduroam" or result == "rpi_wpa2")
    except:
        return False

#Function below will activate alarm if values reach dangerous levels and
read arduinos input
def check_sensor(values,alarm):
    try:
        if ser.in_waiting > 0:
            line = ser.readline().decode("utf-8").strip() #Read each line
            numbers = line.strip().split(",")
               if(int(numbers[0].strip()) == 1 or int(numbers[1].strip())
>=200 or float(numbers[2].strip()) >=115.0):
                alarm = True
            values.append(numbers[0].strip())
            values.append(numbers[1].strip())
            values.append(numbers[2].strip())
    except:
        pass
    return alarm

#Function below will connect to aws iot and send all the data over
def send_to_aws(values,alarm):
    print("Sending the following to AWS IOT: ")
    print(values)
    return

#Function below will connect to esp32 and send signal if connected and
check_sensor goes off
def mqtt(alarm):
    if(alarm != True):
        return
    print("Vibration on")
    command = "mosquitto_pub -h localhost -t vibe/esp -m \"ON\" "
    subprocess.run( ["lxterminal", "--command", command])
    return

def main():
    while True:
        values = []
        alarm = False
          alarm = check_sensor(values,alarm) #When offline, can only check
sensor
        if (alarm == True):
```

```
                print("FIRE DETECTED, ALARMS ON \n ----------------- \n")
#Fire detected
        if is_wifi() or is_ethernet(): #If online, all data is sent to aws
and connect to esp32
            print("Connected to eduroam")
            mqtt(alarm)
            send_to_aws(values,alarm)
        else:
            print("Not connected to eduroam, run local check")
        time.sleep(0.5)
if __name__ == "__main__":
    main()
```

C Code for ESP32:

```
// This is the main file to a program that will connect an esp32 to a rasp
pi over wifi with an mqtt broker
#include <WiFi.h> //Wifi library
#include  "esp_wpa2.h"  //wpa2  library  for  connections  to  Enterprise
networks
#include "wifi_creds.h"
#include "PubSubClient.h"
//SSID NAME
const char* ssid = "eduroam"; // eduroam SSID
const char* mqtt_server = "128.113.94.106";
//MQTT Broker client
WiFiClient espClient;
PubSubClient client(espClient);
//Output voltage pin
const int outputPin = 23;
// MQTT callback function
void callback(char* topic , byte* payload, unsigned int length){
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  //Convert payload to a string
  String message;
  for (int i = 0; i < length; i++){
    message += (char)payload[i];
  }

Serial.println(message);
  if (message == "ON"){
    digitalWrite(23,LOW);
```

```
    Serial.println("Vibration on");
     digitalWrite(23,HIGH); //insert code to turn it off IF you get signal
from aws iot
  }
  else{
    Serial.println("Unsuccessful");
    return;
  }
}
void reconnect(){
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("ESP32Client")) {
      Serial.println("Connected to MQTT Broker");
      // Subscribe to a topic (if needed)
      client.subscribe("vibe/esp"); // Replace with your topic
      client.publish("vibe/esp", "Esp32 connected");
    } else {
      Serial.print("failed ");
      Serial.println(" try again in 5 seconds");
      // Wait before retrying
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  //Setup pwm pin
  pinMode(23,OUTPUT);
  digitalWrite(23,LOW);
  delay(10);
  Serial.print(F("Connecting to network: "));
  Serial.println(ssid);
    WiFi.disconnect(true);    //disconnect  from  WiFi  to  set  new  WiFi
connection
WiFi.begin(ssid,      WPA2_AUTH_PEAP,      EAP_IDENTITY,      EAP_USERNAME,
EAP_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print((("."));
  }
  Serial.println("");
  Serial.println((("WiFi is connected!"));
  Serial.println((("IP address set: "));
  Serial.println(WiFi.localIP()); //print LAN IP
  client.setServer(mqtt_server,1883);
  client.setCallback(callback);
  reconnect();
}
void loop() {
  if (!client.connected()){
    reconnect();
  }
  client.loop();
}

import json
import boto3
from time import gmtime, strftime

dynamodb = boto3.resource('dynamodb')

table = dynamodb.Table('spachz_db');

now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

def lambda_handler(event, context):
 # Log the event data
    print("Received event: " + json.dumps(event, indent=2))

    # Parse the incoming JSON data
    try:

        ir = event['ir_reading']
        temperature = event['temp_reading']
        co = event['co_reading']
```

```
        # Process the sensor data
        # (e.g., store it in a database, perform analysis, etc.)

        response = table.put_item(Item = {
            'ID': 1,
            "ir_reading": ir,
            "temperature": temperature,
            "co": co,
            "timestamp": now
        })

    except Exception as e:
        response_message = {
            "statusCode": 400,
            "body": json.dumps({
                "message": "Error processing data",
                "error": str(e)
            })
        }

    return response

import json
import urllib.request

# Define the API Gateway endpoint
API_URL                                                          =
'https://eb36ui04ff.execute-api.us-east-2.amazonaws.com/dev'

def lambda_handler(event, context):
    try:
        # Make the GET request
        request = urllib.request.Request(API_URL, method='GET')
        with urllib.request.urlopen(request) as response:
            data = response.read().decode('utf-8')

        # Parse the response data if it's JSON
        response_data = json.loads(data)
```

```
        return {
            'statusCode': 200,
            'body': json.dumps({
                'message': 'Data received successfully',
                'data': response_data
            })
        }

    except Exception as e:
        # Handle errors
        return {
            'statusCode': 500,
            'body': json.dumps({
                'message': 'Error occurred',
                'error': str(e)
            })
        }
```

```
import requests
import json


API_ENDPOINT                                                                =
"https://eb36ui04ff.execute-api.us-east-2.amazonaws.com/dev"


def send_data_to_lambda(data):
    headers = {'Content-Type': 'application/json'}
            response   =   requests.post(API_ENDPOINT,   headers=headers,
data=json.dumps(data))
    return response

if __name__ == "__main__":
    data = read_sensor_data()
    response = send_data_to_lambda(data)
    print(f"Response status code: {response.status_code}")
    print(f"Response body: {response.text}")
```