Flame Guard Emergency Fire Alert System and Safety Protocols

The report is based on a project within the course COEN 6711 in Summer 2024

Abhinaw Siddharth Bogadhi
Electrical and Computer Engineering
Concordia University
Montreal, Canada
abhinawsiddharth2000@gmail.com

Anish Gundavarapu
Electrical and Computer Engineering
Concordia University
Montreal, Canada
Anish.g7172@gmail.com

Anil Sai Kumar Bandi Venkata
Electrical and Computer Engineering
Concordia University
Montreal, Canada
Venkataanilsaikumar3@gmail.com

***Abstract*— Our project aims to develop Flame Guard, an advanced automated fire alert system that leverages multiple sensor technologies to detect and respond to fire emergencies efficiently. Fire accidents can occur due to various reasons, including short circuits, overuse of heaters, and cooking mishaps. Flame Guard addresses these risks by integrating a fire detector, smoke detector, temperature sensor, and motion detector, creating a comprehensive detection framework that improves accuracy and reliability.**

The system incorporates several key components. The fire detector identifies the presence of fire through flame detection, while the smoke detector senses smoke, indicating a potential fire. The temperature sensor monitors temperature fluctuations that may signal the onset of a fire, and the motion detector identifies unusual movements that might be associated with a fire event or emergency situation. Additionally, Flame Guard includes a DC motorized fan that acts as the ventilation system, activating to reduce smoke and heat, and a water pump that serves as the fire prevention system, deploying water to extinguish the fire.

Upon detecting a fire, Flame Guard initiates safety protocols, including activating the ventilation fan and water pump. If the fire cannot be extinguished by these measures, the system sends an emergency signal to the nearest safety department for further assistance. By integrating these components and automated safety measures, Flame Guard enhances fire safety, ensuring a higher success rate in mitigating fire-related emergencies. We declare that the report is our own work, restating that each member of the group has contributed to the project's development.

I. INTRODUCTION

1.1. Background and Motivation

Fire detection systems are critical in safeguarding properties and human lives by providing early warnings to prevent the escalation of potentially catastrophic events. Traditional fire detection systems often rely on standalone smoke detectors or thermal sensors, which can be limited in their ability to provide comprehensive and timely alerts. With the advent of Internet of Things (IoT) technologies, it is now possible to develop more advanced, interconnected fire detection systems that enhance reliability and responsiveness.

1.2. Objectives

This project aims to develop a comprehensive fire detection system using a Raspberry Pi, integrating multiple sensors to detect smoke, temperature changes, and flames. The system includes the following components:

- **DS18B20 Temperature Sensor**: Monitors environmental temperature to identify abnormal conditions indicative of a fire. The DS18B20 provides precise temperature measurements and is suitable for high-temperature environments.
- **PCF8591 ADC Module with Smoke Sensor**: Detects the presence of smoke particles in the air, providing an early indication of combustion.
- **Flame Sensor**: Senses the presence of flame using infrared technology, enabling immediate detection of visible fire.
- **Relay-Controlled Water Pump**: Acts as an active fire suppression mechanism by automatically activating upon detection of fire indicators.
- **Raspberry Pi Camera Module**: Provides visual confirmation of fire incidents, allowing for real-time monitoring and documentation.
- **Imgur API**: Which stores the image taken my raspberry pi using pi camera.
- **Twilio Messaging Service**: Sends SMS alerts to predefined recipients to ensure rapid response and intervention.

1.3. Significance

By leveraging the capabilities of the Raspberry Pi and integrating various sensors, this project addresses the limitations of traditional fire detection systems. The inclusion of real-time SMS alerts via the Twilio service ensures that responsible parties are immediately notified of any fire hazards, thereby facilitating swift action. Additionally, the implementation of a relay-controlled water pump offers an immediate on-site fire suppression response, which can mitigate damage before external assistance arrives.

II. LITERATURE REVIEW [1] [2] [3]

Extensive research has been conducted on fire detection and monitoring systems with the goal of improving safety and offering prompt responses to fire dangers. Numerous strategies and technological advancements have been investigated to raise the precision and effectiveness of these systems. **Undug and colleagues (2015)[1]** demonstrated a robot that functions as a fire location, detector, and extinguisher and has the ability to send SMS messages. This system sends alerts to the appropriate staff in addition to using sensors to detect and put out fires. Even in the event that the fire breaks out in an unattended area, the incorporation of SMS warnings guarantees that prompt action can be done. A system for home-based fire monitoring and warning was created by **Suresh, Yuthika, and Vardhini (2016)[2]**. This system uses a number of sensors to keep an eye on several environmental factors that could point to a fire. Their research's main goal was to develop a dependable home system that could notify homeowners of a fire via SMS notifications and alarms. **Sulthana et al. (2023)[3]** offered a thorough analysis of current advancements in fire detection techniques. Their research emphasised advances in sensor technology, such as the application of several sensors to increase the precision of detection. The report also stressed how critical it is to use IoT technologies in order to facilitate real-time alarms and remote monitoring, both of which are essential for efficient fire management. Our approach, in contrast to these studies, uses a Raspberry Pi and a number of sensors to build a reliable fire detection system. Comprehensive environmental condition monitoring is made possible by the employment of the Raspberry Pi Camera Module, the MQ2 smoke and flame sensors, the DS18B20 temperature sensor, and the MQ2 smoke sensor. The PCF8591 ADC module's capabilities enable our system to precisely convert and process analog inputs from the smoke sensor. By integrating Twilio's messaging service, real-time SMS notifications are made possible, guaranteeing that in the event of a fire, the appropriate authorities are informed as soon as possible.

The body of research on the subject shows how crucial multi-sensor systems and IoT integration are to fire detection. By adding cutting-edge sensors and real-time communication capabilities, our proposal expands on existing ideas and improves fire detection and response efficiency. Relay-controlled water pump offers automated fire suppression, while Raspberry Pi central processing unit facilitates effective data processing and decision-making. All things considered, our project's utilisation of sensor technology, IoT integration, and real-time alert mechanisms is in line with the developments covered in the literature. Our system strives to increase safety and reduce the risks related to fire threats by offering a complete fire detection and response solution.

III. COMPARISON OF THE FLOW DIAGRAM OF REFERRED PAPER AND OUR WORK.

Devices Used/ Protocol	Conference paper 1	Conference paper 2	Journal paper	Our Implementation
Processor	Arduino Uno R3	PIC16F874 /877 micro-controller	ARM Cortex A7	Raspberry Pi 2B Model ARM Cortex A7
Sensor	**LM35** Flame sensor	Dry Contact Smoke Detector , IR Proximity Sensor, Photo reflective Sensor	CO2 detection, PIR, ultrasonic	**KY-026** Flame Sensor, **MQ-2** Smoke Sensor, **DS18B20** Temperature Sensor, **PIR** Motion Sensor
Wifi Module / GSM	GSM SIM 900	GSM Module	ZigBee, Wi-Fi, Bluetooth	WiFi, Twillio Messaging Service
Programming Language	embedded C	embedded C	Python	C Language
ADC	NA	NA	Yes (PCF8591)	Yes (PCF8591)
IOT Enabled	NO	NO	Yes	Yes
Remarks	It's a simple straightforward implementation with the LM35 flame sensor sensing the fire sends a alert message to owner but this systems lacks the measures to handle in case of errors.	In this paper a separate module for fire detection and fire extinguishing robot has been proposed where the limitation of having the sensors fixed to the walls making it a little uneasy for the detection in some case and the robot won't be able to receive the current updates thus can't avoid the situation.	Comprehensive smart building system using various sensors and protocols for enhanced efficiency and safety. However, it lacks the necessity of fire safety protocols and an safety line in-case the main sensor doesn't work.	Our implementation leverages multiple sensor technologies and IoT for comprehensive fire detection and response. The system integrates flame, smoke, and temperature sensors with real-time alerts and automatic fire suppression mechanisms.

Table 1 Comparison Between Existing and proposed work

IV. HARDWARE SETUP

In this section, we will outline the hardware setup for the fire detection system using the Raspberry Pi 2B Model, integrating the DS18B20 temperature sensor, PCF8591 ADC module with a smoke sensor, flame sensor, relay module, water pump, and Raspberry Pi Camera Module.

1) MQ-2 Sensor

Pins: Since our MQ2 sensor is analogue, its three primary pins are Vcc, GND, and Analogue OUT.

Usability: The MQ2 sensor is made to identify a variety of gases, including butane, propane, smoke, and methane. Because of its adaptability, it is widely used in many IoT smart applications. The MQ2 sensor in this research is primarily used to detect methane gas, which is emitted from biodegradable

Operational Concept: Based on the Metal Oxide Semiconductor (MOS) concept, the MQ2 sensor functions. The metal oxide sensitive layer is heated by a heating device. This layer's resistance to methane gas alters when it comes into touch with it.

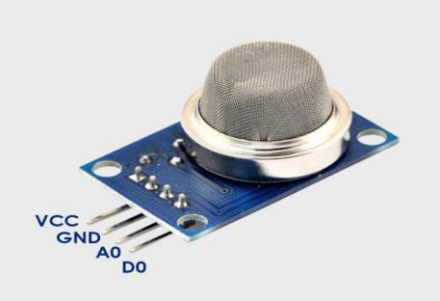

Figure 1 MQ-2 Sensor

2) DS18B20 Temperature Sensor

- VCC: Power supply pin.
- GND: Ground pin.
- DATA: Digital data pin.

Functionality: The DS18B20 is a digital temperature sensor capable of measuring temperatures with high precision. It communicates over a 1-Wire bus, which allows for multiple devices to be connected on the same data line.

Working Principle: The DS18B20 sensor uses a digital protocol to send temperature readings to the Raspberry Pi. It has an internal 12-bit resolution that converts temperature into digital form, making it highly accurate. The sensor is useful for detecting temperature changes in the environment, which can indicate a fire.

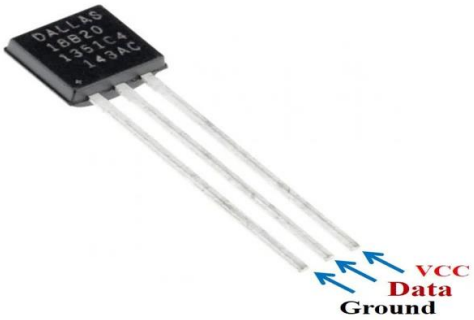

Figure 2 2DS18B20 sensor

3) PCF8591 ADC Module with MQ2 Smoke Sensor

Pins of PCF8591 ADC Module:

- **VCC**: Power supply pin.
- **GND**: Ground pin.
- **SDA**: Serial Data line for I2C communication.
- **SCL**: Serial Clock line for I2C communication.
- **AIN0-AIN3**: Analog input channels.

Functionality: The PCF8591 is an Analog-to-Digital Converter (ADC) that can read analog signals from sensors and convert them into digital values for the Raspberry Pi. It is used in conjunction with the MQ2 smoke sensor.

Figure 33 ADC Converter

4) Flame Sensor

Pins:

- **VCC**: Power supply pin.
- **GND**: Ground pin.
- **D0**: Digital output pin.

Functionality: The flame sensor detects the presence of a flame using infrared technology. It is sensitive to wavelengths typically emitted by fire and can respond quickly to the presence of a flame.

Working Principle: The sensor has an IR receiver that detects infrared light from a flame. When a flame is present, the sensor's output signal changes, which can be read by the Raspberry Pi to trigger alerts or activate fire suppression mechanisms.

Figure 44 Flame Sensor

5) Relay Module

Pins:

- **VCC**: Power supply pin.
- **GND**: Ground pin.
- **IN**: Control signal pin.
- **COM (Common)**: Common contact for relay.
- **NO (Normally Open)**: Normally open contact for relay.

Functionality: The relay module is an electrically operated switch that allows the Raspberry Pi to control high-power devices like the water pump. The relay is triggered by a control signal from the GPIO pin.

Working Principle: When the control signal from the Raspberry Pi is applied to the IN pin, the relay closes the circuit between COM and NO, allowing current to flow to the connected device, such as the water pump.

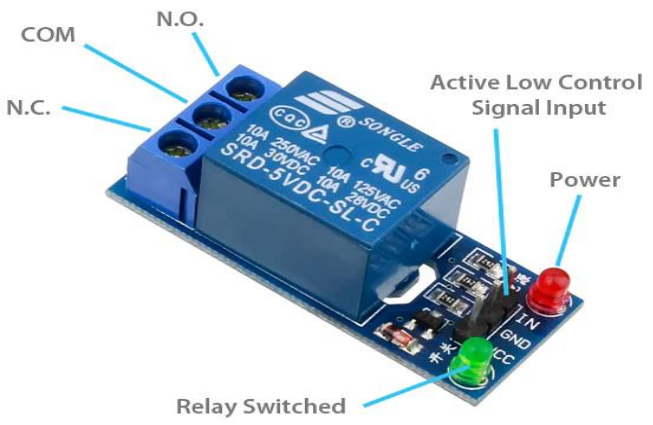

Figure 55 Relay Module

6) Water Pump

Pins:

- **Positive Terminal**: Connects to the relay module's NO contact.
- **Negative Terminal**: Connects to the ground.

Functionality: The water pump is used to dispense water to suppress a fire when triggered by the relay module. It is an essential part of the active fire suppression system.

Working Principle: When the relay module is activated, it closes the circuit, allowing power to flow to the water pump, which then operates to pump water.

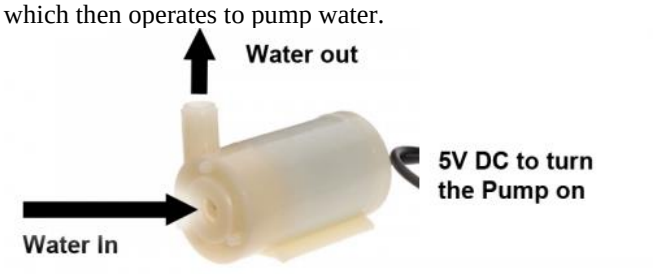

Figure 66 DC Water Pump

7) Raspberry Pi Camera Module

Pins:

- **Camera Interface**: Connects to the dedicated camera port on the Raspberry Pi.

Functionality: The Raspberry Pi Camera Module captures images and videos. It is used for visual confirmation of fire incidents and real-time monitoring.

Working Principle: The camera module is connected to the Raspberry Pi via the CSI (Camera Serial Interface) port. It captures visual data, which can be processed or streamed by the Raspberry Pi for monitoring and documentation purposes.

Figure 77 PI Camera

8) Raspberry Pi 2 Model B

The Raspberry Pi 2 Model B represents a major step forward in the Raspberry Pi series' history, building on the capabilities of its predecessor with notable increases in processing power and memory. The **900MHz** quad-core **ARM Cortex-A7 CPU** in this edition is a significant improvement over the single-core **ARM11 processor** found in the original Raspberry Pi. With this increase in processing power, the **Pi 2 Model B** can now **handle more complicated computations and multitasking**, which improves its usefulness for a variety of **applications**, including **home automation systems and educational tools**. With its **1GB RAM**, the Pi 2 Model B has more than enough capacity to run a **wide range of operating systems** and applications, including Windows **10 IoT Core** and **several Linux versions**. Its increased capacity also makes multimedia applications—like **gaming** and **movie playback**—more manageable, making it a useful tool for both enthusiasts and developers. Compatibility with existing Raspberry Pi accessories is ensured by the **board's retention** of the same form factor as its **predecessors**. It has an Ethernet port for network connectivity, four USB 2.0 ports, an HDMI port for high-definition video output, a microSD card for storage, and a **40-pin GPIO header** for integrating with external sensors and hardware. The Pi 2 Model B is a strong and adaptable platform for do-it-yourself projects and educational uses thanks to all of these characteristics. The Raspberry Pi 2 Model B's low power consumption—roughly 3.5 watts when used normally—is one of its main features. Because of this, it is an **energy-efficient** option for embedded systems and other applications where power efficiency is essential for continuous operation. All things considered, the Raspberry Pi 2 Model B is a strong and versatile single-board computer that provides improved performance and more capabilities, upholding the goals of the Raspberry Pi Foundation to encourage computer science education and stimulate creativity.

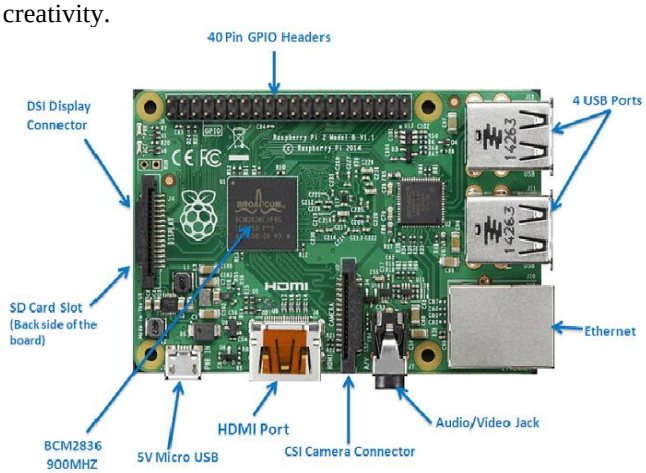

Figure 88 Raspberry Pi 2B Model

9) DC Motor

This project uses a DC motor to operate a fan, which is a crucial part of the reaction mechanism of the fire detection system. Standard 5V DC motors are regarded for their dependability and simplicity in delivering steady mechanical power, and this one is no exception. During a fire, this motor dissipates smoke and cools the surrounding area by

converting electrical energy into mechanical energy that rotates the fan blades. A direct current (DC) voltage is applied to the DC motor, causing its internal armature to rotate. Airflow is produced when this rotation is transferred to the fan blades. With this configuration, the motor operates at a constant speed when energised, making speed control unnecessary. Relay module, which function as switches to regulate the motor's activity depending on signals from the fire detection system, are used to link the motor to the Raspberry Pi. When a fire is detected, the relay module, which is connected to GPIO 19, closes the circuit, enabling the Raspberry Pi to power the motor. The relay is triggered by this action, and the DC motor and fan are powered as a result. By utilising an integrated method, the fan is guaranteed to function efficiently in aiding with **smoke dissipation and environmental cooling**, so augmenting the system's **total fire suppression capabilities**.

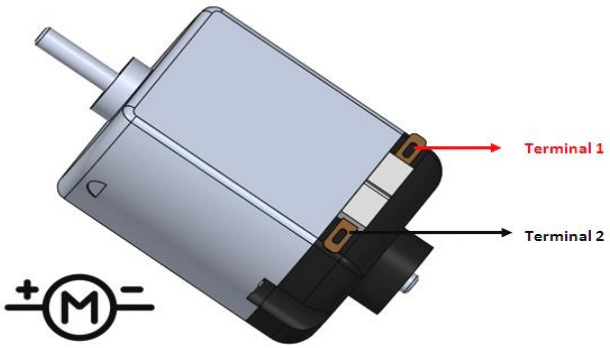

Figure 99 DC Motor

10) PIR Sensor

An essential part of motion detection in this project is the Passive Infrared (PIR) sensor, which gives the fire detection system an extra degree of security and functionality. PIR sensors pick up infrared radiation (IR), which is released by objects in their area of view that are warmer than their surroundings, including human bodies. To measure the difference in infrared radiation between two places that are within its detection range, the PIR sensor is used. The sensor senses a change in infrared levels when a warm item, such a person, passes over these areas and sends out a digital output signal. The Raspberry Pi can then interpret this signal to indicate motion detection. In our configuration, the Raspberry Pi's GPIO pin—typically GPIO 13—is linked to the PIR sensor, allowing it to read the sensor's digital output. The Raspberry Pi receives a strong signal from the PIR sensor when it detects motion, signalling the existence of movement within its detection range. This feature is essential in situations where motion detection is needed to activate other system components or to provide extra safety. The PIR sensor performs more than just motion detection in the fire detection system. The sensor can assist in detecting human presence during a fire, ensuring that notifications and reactions are sent to populated areas only when necessary. A thorough approach to safety and security in the overall system architecture is provided by the sensor's ability to monitor unauthorised entry to critical locations.

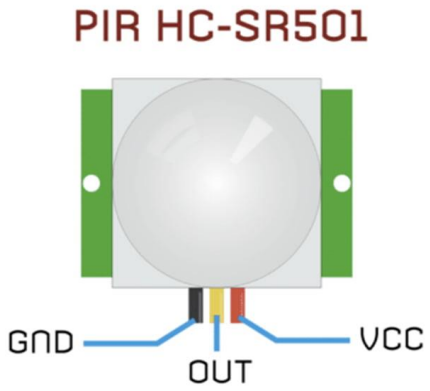

Figure 1010 PIR Motion Sensor

11) Breadboard

A breadboard serves as a crucial prototype platform that allows the various components of the fire detection system to be assembled and tested solder-free. Electronic components can be readily linked and rearranged on the reusable and adaptable breadboard, which makes experimentation and troubleshooting more effective. Component leads and jumper wires can be inserted into the breadboard's grid-like network of connected holes. Connections between components, including resistors, sensors, and the Raspberry Pi, are simple to make because to its electrically connected rows or columns of vertical and horizontal rails. The DS18B20 temperature sensor, MQ2 smoke sensor, flame sensor, relay module, DC motor, and any required resistors and jumper wires are all arranged and connected on the breadboard in our arrangement. The Raspberry Pi's GPIO pins connect to these parts through the breadboard, making the wire layout neat and controllable. The project benefits from a clean and organised layout when utilising a breadboard, which lowers the possibility of short circuits and makes it simpler to adjust the circuit as needed. The development process is streamlined by this modular approach, which also ensures dependable connections throughout the fire detection system and permits prompt alterations.

V. ARCHITECTURE

The fire detection system developed in this project is cantered around the Raspberry Pi 2 Model B, which serves as the central processing unit. The system integrates multiple sensors, a camera module, and a relay-controlled water pump to effectively detect and respond to fire hazards. The **DS18B20** temperature sensor is used to measure ambient temperature and is connected to **GPIO pin 4** on the Raspberry Pi via a 1-Wire protocol. The **MQ2 smoke sensor**, which detects gases such as methane, propane, smoke, and butane, outputs an analog signal that is converted to a digital signal using the **PCF8591 ADC module**. The ADC module **communicates** with the Raspberry Pi over the **I2C interface**, with connections to **GPIO 2 (SDA) and GPIO 3 (SCL)**. Additionally, a flame sensor is used to detect the presence of a flame and provides a digital output connected to GPIO 5 on the Raspberry Pi.

The relay module is controlled by the Raspberry Pi through GPIO 19 and is powered by the Raspberry Pi's 5V and GND pins. It is used to switch the **water pump on or off** based on signals received from the Raspberry Pi, with the pump motor connected to the Normally Open (NO) and Common (COM) terminals of the relay. The **Raspberry Pi Camera Module**, connected to the dedicated CSI port, **captures images** and videos for **real-time monitoring** and verification of fire incidents. To provide **alert notifications**, the system uses the **Twilio messaging service**, which allows the Raspberry Pi to send SMS alerts to predefined contacts via its internet connection and the Twilio API.

The system operates through a well-defined workflow. Upon initialization, the Raspberry Pi boots up and initializes all connected sensors and modules. It then enters a **continuous data acquisition phase** where the DS18B20 sensor monitors ambient temperature, the MQ2 sensor detects smoke levels and other gases with the help of the PCF8591 ADC, and the flame sensor checks for the **presence of a flame**. The Raspberry Pi processes the **incoming data from these sensors**, comparing the readings against predefined **thresholds to determine** if a fire condition exists.

If a fire condition is detected, the Raspberry Pi triggers the relay module, which **activates** the water pump to begin extinguishing the fire. Concurrently, the Raspberry Pi Camera Module **captures images** or videos of the incident for documentation and further analysis. The system also sends an SMS alert via Twilio to notify emergency contacts or relevant **authorities about the fire incident**. Throughout this process, the system continues to monitor the environment, updating its status in real-time to ensure prompt detection and response to any fire hazards.

This architecture leverages the capabilities of the Raspberry Pi 2 Model B and various sensors to create a comprehensive and efficient fire detection system. By integrating temperature, smoke, and flame detection with real-time monitoring and automated response mechanisms, the system enhances safety and provides timely alerts and actions to mitigate fire hazards.

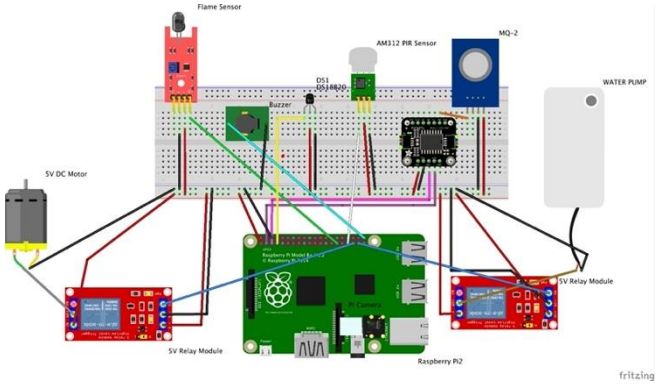

Figure 1111 Connection Diagram

VI. WORKING

A. Hardware Process and Simulation

Using the Raspberry Pi 2 Model B as its primary processing unit, the fire detection system is intended to offer thorough monitoring, detection, and response to fire dangers. Several sensors, a camera module, a relay-controlled water pump, and a motor are all integrated into the system to guarantee effective and efficient fire management.

1) ***Initialization and Data Acquisition****:* The Raspberry Pi initialises all attached components when it powers on. The environmental parameters are continuously monitored by the temperature sensor DS18B20, the smoke sensor MQ2, and the flame sensor. The DS18B20 sensor detects the surrounding air temperature. It is wired via GPIO 4. Smoke and gas levels are detected using the MQ2 sensor, which is connected through the PCF8591 ADC module, which communicates via the I2C interface. The flame sensor detects the presence of flames and is attached to GPIO 5.

2) ***Motion Detection:*** The PIR motion sensor, which is attached to GPIO 13, uses variations in infrared radiation to identify motion. This provides an extra degree of protection and facilitates the detection of human presence within the monitored region.

3) ***Data Processing and Analysis****:* The Raspberry Pi processes the data from these sensors in real-time. It compares the readings against predefined thresholds to determine if there is a fire hazard. The DS18B20 provides temperature data, the MQ2 provides smoke and gas levels, and the flame sensor confirms the presence of fire. The PIR sensor ensures that motion is also monitored for enhanced situational awareness.

4) ***Response Activation :*** Relay module coupled to GPIO 19 (motor) and GPIO 26 (pump) is activated by the Raspberry Pi when sensor data indicates possible fire (high temperature, smoke, or flames detected). The water pump and DC motor are subsequently powered by the relay, and they cooperate to put out the fire by spreading water.

5) ***Real-time Monitoring and Alerts****:* In order to provide visual alerts, the Raspberry Pi Camera Module simultaneously records pictures or videos of the fire incident. In order to guarantee timely notifications and fast action from pertinent authorities or persons, the system additionally makes use of the Twilio messaging service to send SMS alerts to designated contacts.

6) ***Continuous Operation****:* The system keeps an eye on the surroundings at all times it is operating. In order to provide a real-time status report and enable prompt reaction to any new fire dangers, the Raspberry Pi continues to process sensor data.

This integrated approach ensures a robust and reliable fire detection and suppression system, combining sensor technology, data processing, real-time communication, and automatic response mechanisms to enhance safety and fire management.

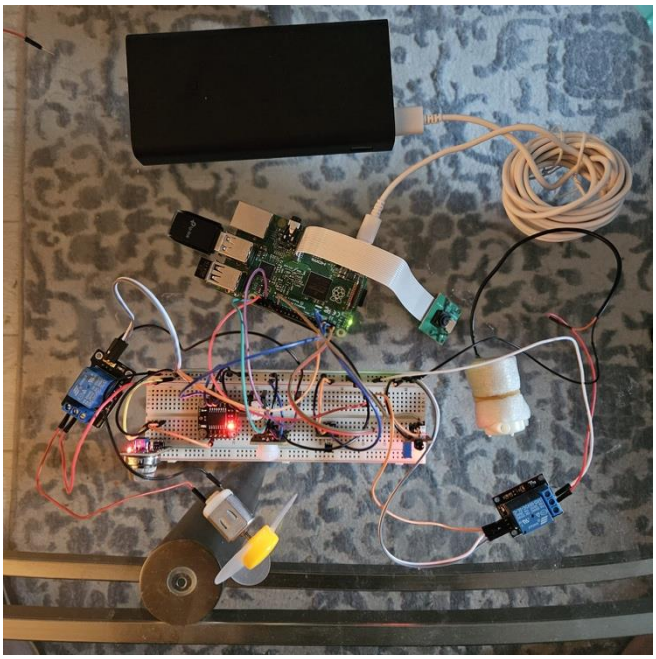

Figure 12 12 Hardware Setup

B. Software Simulation:

1) ***Proteus Overview:***

Proteus is a highly regarded software simulation tool used extensively in the realms of electronic circuit design, simulation, and PCB layout. It enables users to create and test virtual prototypes of electronic circuits, ensuring their functionality is verified before moving on to physical implementation. Here are some essential aspects of Proteus to be aware of before you begin using it.

2) ***Components and Libraries:*** Proteus boasts a vast array of components and libraries, encompassing a wide spectrum of electronic components from numerous manufacturers. These components are systematically organized within libraries, making it convenient for users to navigate and select the necessary components for their circuit designs. The libraries include a variety of elements such as microcontrollers, sensors, actuators, and passive components. This extensive collection of components not only saves time but also reduces the effort required to locate and create models for each specific component.

3) ***Simulation Features:*** Simulation is a core feature of Proteus, providing users with the ability to verify the functionality of their circuit designs prior to physical implementation. Proteus supports multiple simulation modes, including DC, AC, Transient, and Digital simulations. DC simulation facilitates the analysis of a circuit's steady-state behavior by applying constant voltages and currents. AC simulation is utilized to examine the circuit's behavior across various frequencies. Transient simulation allows for the observation of the circuit's response to time-varying signals. Digital simulation is tailored for digital circuits, enabling users to verify the logic and timing of their digital designs.

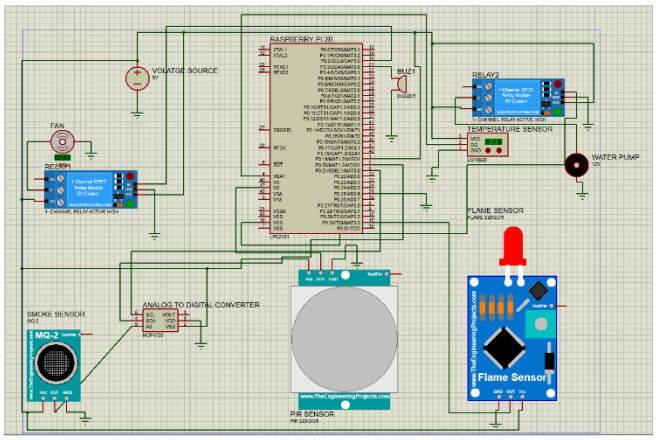

Figure 13 Proteus Software Simulation Circuit.

The figure shows the components and actuators used and how they are arranged in proteus. The following are the steps to setup the components and the pin connection respectively:

4) ***Schematic Capture:***

- → Open Proteus 8 Professional and click on new project.
- → Go to Components and search for Raspberry Pi 2b, MQ2 Smoke Sensor, Fan, Flame Sensor, Temperature Sensor (DS18B20), DC Motor, Relay, PIR Sensor, Analogue to Digital Converter, Voltage Source and Ground.
- → Place the above following components on PCM board.
- → Connect the Flame Sensor as follows:
  - ➢ Vcc to the Voltage Source.
  - ➢ GND to Ground, and
  - ➢ Out to GPIO 15 (on Raspberry Pi 2B).
- → Connect ADC to MQ2 Smoke Sensor as follows:
  - ➢ SCL to GPIO 3.
  - ➢ SDA to GPIO 2.
  - ➢ Vcc and GND to Voltage Source and Ground respectively.
  - ➢ A0 (ADC) to OUT (MQ2).
- → Connect Water Pump (DC Motor) with Relay2:
  - ➢ C (Relay) pin to DC Motor.
  - ➢ IN (relay) to GPIO 19.
  - ➢ Vcc and GND to Voltage Source and Ground respectively.
- → Connect Fan with Relay1 as following:
  - ➢ C (relay) to FAN.
  - ➢ IN (relay) to GPIO 18.
  - ➢ Vcc and GND to Voltage Source and Ground respectively.
- → Connect Temperature Sensor (DS18B20), DQ pin to GPIO 4.
- → Connect the OUT pin on PIR sensor to GPIO 17.
- → Copy the location of .HEX file after compiling the source code and save it for further steps.
- → Now double-click on the device or right-click on the MQ2 Smoke Sensor and go to Program file -> copy and paste the .HEX file path -> click Ok.

→ The above step is followed for Flame Sensor, PIR sensor and Raspberry Pi 2B board.

The following steps give us the smooth execution of software simulation in Proteus 8 Professional software.

C. Work Flow:

The following flow chart shows the workflow of our hardware design, starting with Sensor reading till the alert SMS sent.

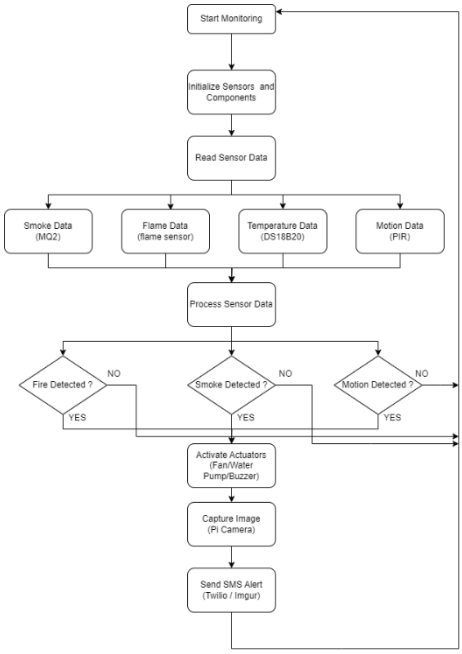

Figure 14 Flow Diagram of the program

The following steps below will give a brief description of the flow chart:

Step 1- Start Monitoring:

→ The system is powered on and begins the monitoring process.

Step 2- Initialize Sensors and Components:

→ All sensors (smoke, flame, temperature, motion) and components (fan, water pump, Pi camera) are initialized and set up for data collection.

Step 3- Read Sensor Data:

→ The system continuously reads data from the connected sensors:
- ➢ Smoke Data from MQ2 sensor.
- ➢ Flame Data from the flame sensor.
- ➢ Temperature Data from the DS18B20 sensor.
- ➢ Motion Data from the PIR sensor.

Step 4- Process Sensor Data:

→ The collected sensor data is processed to determine if any emergency conditions (fire, smoke, abnormal temperature, or unusual motion) are detected.

Step 5- Fire (or) Smoke (or) Motion Detected? (Decision Point):

→ The system checks if the flame sensor data indicates the presence of fire.

→ Yes: If fire (or) smoke (or) motion is detected, proceed to the next step.

→ No: If no fire (or) smoke (or) motion is detected, the system continues to monitor sensor data.

Step 6- Activate Actuators (Fan/Water Pump/Buzzer):

→ Upon detecting fire, smoke, or unusual motion, the system activates the necessary actuators:
- ➢ Turns on the DC motor fan to ventilate smoke.
- ➢ Activates the water pump to extinguish the fire.
- ➢ Sounds a buzzer for alert.

Step 7- Capture Image (Pi Camera):

→ The Pi camera captures an image of the area every 30 seconds for documentation and further analysis.

Step 8- Send SMS Alert (Twilio / Imgur):

→ The system sends an SMS alert to predefined phone numbers using the Twilio API.

→ The captured image is uploaded to an online image hosting service (i.e., Imgur), and the link is included in the SMS alert for remote viewing.

→ The Process returns to start monitoring and runs continuously.

D. Hardware Simulation Results:

Test Case 1- Flame Detection:

→ Test Case: Simulate fire near flame sensor without motion detection.

→ Expected Result: Detect flame, activate buzzer & water pump, send SMS.

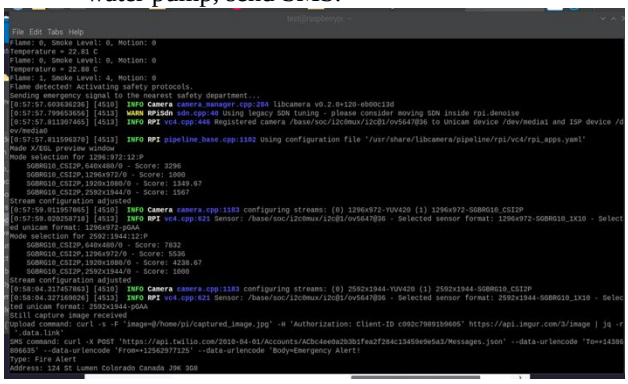

Figure 15 Execution *result for test case 1.*

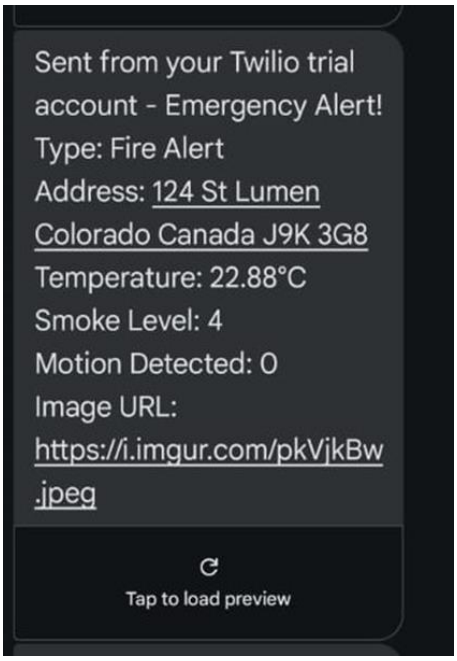

Figure 16 SMS Alert received for Test Case 1.

→ Actual Result:
- ➢ Type- Fire Alert
- ➢ Smoke Level- 4
- ➢ Motion Detection- 0
- ➢ Alert: Activates Water Pump, Buzzer and SMS sent with image.

Test Case 2- Flame Detection with Motion:

→ Test Case: Simulate fire near flame sensor with motion detection.

→ Expected Result: Detect flame, activate buzzer, water pump & motor fan, send SMS.

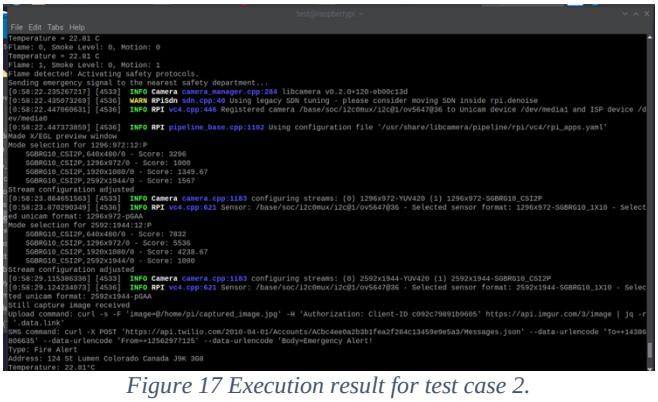

Figure 17 Execution result for test case 2.

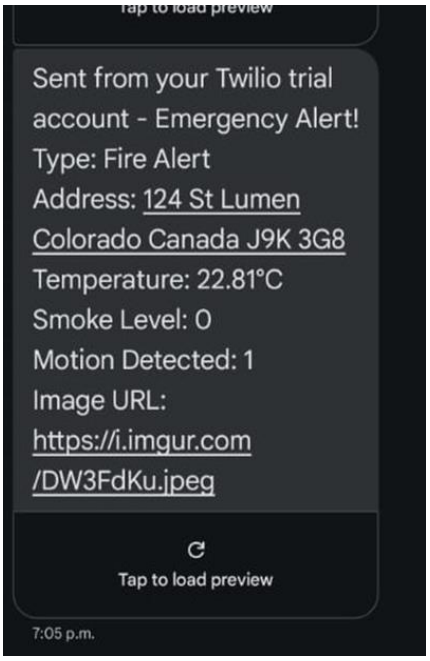

Figure 18 SMS Alert received for Test Case 2.

→ Actual Result:
- ➢ Type- Fire Alert
- ➢ Smoke Level- 0
- ➢ Motion Detection- 1
- ➢ Alert: Activates Water Pump, Fan, Buzzer and SMS sent with image.

Test Case 3- Motion Detection:

→ Test Case: Simulate motion near PIR sensor.

→ Expected Result: Detect motion, activate buzzer, send SMS.

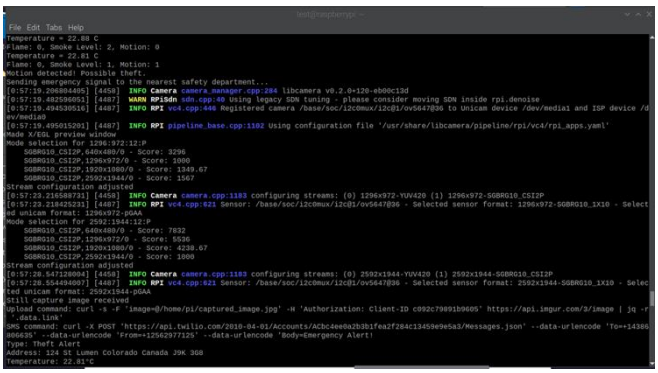

Figure 19 Execution result for test case 3.

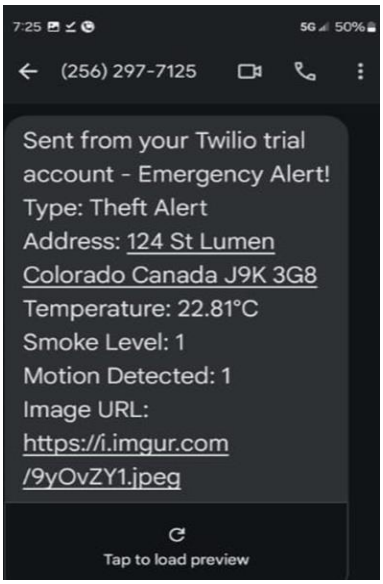

Figure 20 SMS Alert received for Test Case 3.

→ Actual Result:
- Type- Theft Alert
- Smoke Level- 1
- Motion Detection- 1
- Alert: Activates Buzzer and SMS sent with image.

Sample image on Imgur website:

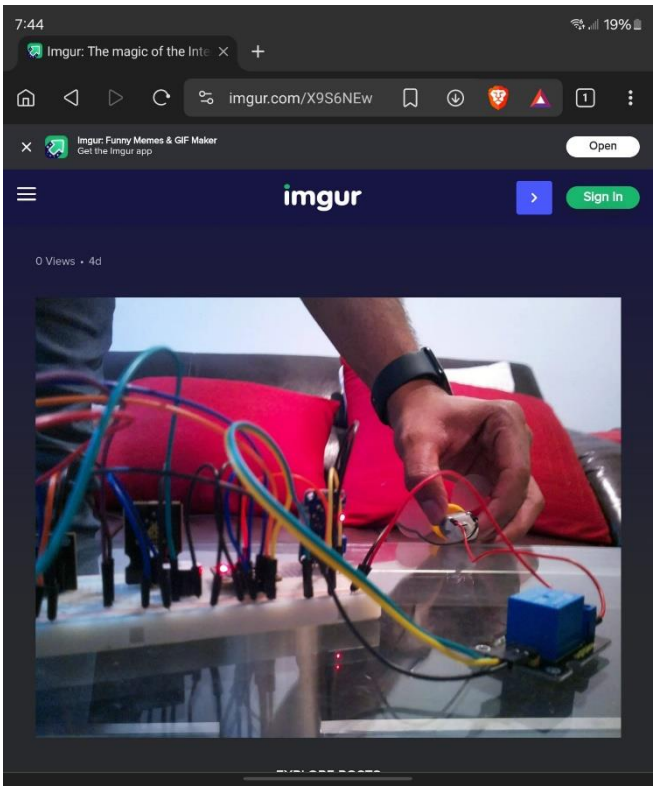

Figure 21 Image uploaded on Imgur website after triggering alerts.

VII. C Language- Code Explaination

A. *Libraries Used*

- **<wiringPi.h>**: Provides functions to use GPIO pins on the Raspberry Pi.
- **<stdio.h>, <stdlib.h>, <stdint.h>**: Standard C libraries for input/output, memory allocation, and fixed-width integer types.
- **<unistd.h>, <fcntl.h>**: For POSIX operating system API functions, such as file control options.
- **<linux/i2c-dev.h>, <sys/ioctl.h>**: For I2C device operations on Linux.
- **<dirent.h>, <string.h>**: For directory entry operations and string manipulation.

B. *GPIO Pins Configuration*

- **#define FLAME_SENSOR_PIN 5, #define PIR_SENSOR_PIN 13**, etc.: Defines for GPIO pins used.
- **pinMode(FLAME_SENSOR_PIN, INPUT)**: Sets the pin mode for the flame sensor as input.
- **pinMode(PIR_SENSOR_PIN, INPUT)**: Sets the pin mode for the PIR sensor as input.
- **pinMode(RELAY_MOTOR_PIN, OUTPUT), pinMode(RELAY_PUMP_PIN, OUTPUT), pinMode(BUZZER_PIN, OUTPUT)**: Sets the pin modes for the relays and buzzer as outputs.

C. *SPI/I2C Configuration*

- **int fd**: File descriptor for I2C device.
- **const char *fileName = "/dev/i2c-1":** Specifies the I2C bus file.
- **ioctl(fd, I2C_SLAVE, PCF8591_ADDRESS)**: Configures the I2C device address.

D. *Temperature Sensor (DS18B20)*:

```
float read_ds18b20_temp() {
    // Code to read temperature data from the sensor
}
```

Reads temperature from the DS18B20 sensor, processes, and returns it.

E. *Smoke Sensor (PCF8591 ADC):*

```
int read_smoke_sensor(int fd) {
    // Code to read smoke level from the ADC
}
```

Reads smoke level from the ADC connected via I2C.

F. *Relay Control Functions*

- **void activate_relay(int pin):** Activates a relay by setting the GPIO pin high.
- **void deactivate_relay(int pin):** Deactivates a relay by setting the GPIO pin low.

G. *Emergency Signal:*

```
void send_emergency_signal() {
    printf("Sending emergency signal to the nearest safety department...\n");
}
```

Prints a message indicating an emergency signal is sent.

H. SMS Sending Function:

```
void send_sms_message(const char *alert_type, float temperature, int smoke_level, int motion_detected) {
   // Code to send an SMS alert with details and an image URL
}
```

Constructs and sends an SMS message with emergency details using the Twilio API. Uploads the image to Imgur and includes the URL in the SMS.

I. Main Function and Logic

```
while (1) {
   // Read sensor data
   int flame_detected = digitalRead(FLAME_SENSOR_PIN);
   int motion_detected = digitalRead(PIR_SENSOR_PIN);
   int smoke_level = read_smoke_sensor(fd);
   float temperature = read_ds18b20_temp();

   // Check conditions and activate safety protocols
   if ((flame_detected && !previous_flame_detected) || (temperature > 50)) {
      // Fire alert logic
   } else if ((smoke_level > 128 && smoke_level != previous_smoke_level) || (temperature > 35)) {
      // Smoke alert logic
   } else if (motion_detected && !previous_motion_detected) {
      // Theft alert logic
   } else {
      // Deactivate alarms and relays
   }

   // Update previous sensor states
   previous_flame_detected = flame_detected;
   previous_motion_detected = motion_detected;
   previous_smoke_level = smoke_level;
   delay(1000);
}
```

- Initializes I2C and GPIO settings.
- Enters an infinite loop (while (1)) to continuously monitor sensors.
- Reads data from flame, motion, smoke sensors, and temperature.
- Checks conditions to determine if there is an emergency (fire, smoke, or theft).
- Activates relays, buzzer, captures an image, and sends SMS alerts if an emergency is detected.
- Deactivates relays and buzzer if no emergency is detected.
- Updates previous sensor states and adds a delay for the next iteration.

VIII. LIMITATIONS

1) ***Accuracy of MQ-2 Smoke Sensor:*** The MQ-2 sensor is capable of reacting to gases such as alcohol, methane, and carbon monoxide in addition to detecting different gases like smoke. However, the consertration of the particles need to accurate for the reliable operation for the equiment. **Consideration:** Making sure the sensor is correctly calibrated for smoke detection in particular. Cross-sensitivity to other gases may result in readings that are erroneous or false positives.

2) ***Alert System Realiability:*** Reliance on internet access to send SMS warnings via Twilio and upload photos to Imgur from the system.

Consideration: Making sure that network activities have reliable error handling and retry capabilities. Test in various network scenarios to confirm alert delivery on time and service dependability.

3) ***Environmental Aspects :*** The system needs to be able to function dependably under a range of environmental circumstances, such as changes in humidity and temperature, as well as possible exposure to dust or other pollutants. The difficult part is choosing sensors and other parts that are rated for the environment in which the system will be used. putting protective measures in place to improve longevity and dependability (such as enclosure for sensors and waterproofing for outdoor installations).

IX. FUTURE SCOPE

Integration with Smart Home Systems is a major achievement when considering the future scope of the fire detection project. This solution allows for easy central control and monitoring through integrations with services like Google Home, Amazon Alexa, and Apple HomeKit. This connection increases the system's responsiveness in an emergency while also improving convenience. Its capabilities are further enhanced by implementing advanced analytics and machine learning, which allows for more accurate detection and predictive maintenance by precisely analysing sensor data. Over time, this sophisticated processing maximises operational efficiency while also improving safety. The enhanced communication capabilities, which support several protocols including Bluetooth and Wi-Fi, provide reliable and adaptable connection—essential for warnings and system updates that happen in real time. By addressing dependability issues during blackouts, integrating solar power improves the sustainability and resilience of the system. In order to provide full safety and environmental monitoring capabilities, Broader Environmental Monitoring extends the system's reach to include monitoring air quality, detecting gas leaks, and checking water levels. All of these developments point to the creation of an advanced, networked system that will guarantee comprehensive environmental and safety monitoring for increased peace of mind and proactive management in addition to early fire detection.

X. CONCLUSION

To sum up, the FlameGuard uses a variety of sensors and Raspberry Pi technologies to provide a strong safety solution, combining creativity and pragmatism. The project's initial emphasis on using the Raspberry Pi 2B model equipped with a camera, PIR motion sensor, temperature sensor, MQ-2 smoke sensor, and DS18B20 temperature sensor highlights its dedication to comprehensive fire detection capabilities. A proactive layer is added by the efficient mitigation of potential fire threats that come with integrating a DC motor with a fan and water pump. The possible future

improvements, like integration with Google Home, Amazon Alexa, and Apple HomeKit, avenues for centralised management and improved user accessibility are made clear. In the future, the use of machine learning and advanced analytics should further improve the system's performance by improving detection accuracy and facilitating predictive maintenance. This strategic integration promotes efficiency and ongoing improvement in addition to improving safety results. The project highlights its suitability for real-world implementation by resolving pragmatic concerns including power resilience through solar integration and regulatory compliance. Users are guaranteed dependability and peace of mind by its adherence to safety rules and capacity to function independently during power outages. In the end, the Flame Guard serves as evidence of technological advancements made with the intention of protecting people and property.

XI. References

[1]. J. Undug, M. P. Arabiran, J. R. Frades, J. Mazo and M. Teogangco, "Fire Locator, Detector and Extinguisher Robot with SMS Capability,"2015 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM), Cebu, Philippines, 2015, pp. 1-5, doi:10.1109/HNICEM.2015.7393197

[2]. Suresh S., Yuthika S. and G. A. Vardhini, "Home based Fire Monitoring and warning system," 2016 International Conference on ICT in Business Industry & Government (ICTBIG), Indore, India, 2016, pp. 1-6, doi: 10.1109/ICTBIG.2016.7892664.

[3]. S. F. Sulthana, C. T. A. Wise, C. V. Ravikumar, R. Anbazhagan, G. Idayachandran and G. Pau, "Review Study on Recent Developments in Fire Sensing Methods," in IEEE Access, vol. 11, pp.90269-90282,2023, doi:10.1109/ACCESS.2023.3306812.

[4]. A. Verma, S. Prakash, V. Srivastava, A. Kumar and S. C. Mukhopadhyay, "Sensing, Controlling, and IoT Infrastructure in Smart Building: A Review," in IEEE Sensors Journal, vol. 19, no. 20, pp. 9036-9046, 15 Oct.15, 2019, doi:10.1109/JSEN.2019.2922409.

[5]. https://www.raspberrypi.com/documentation/

[6]. https://apidocs.imgur.com/

[7]. https://www.twilio.com/en-us

[8]. H. Ghayvat, S. C. Mukhopadhyay, X. Gui, and N. Suryadevara, "WSN- and IOT-based smart homes and their extension to smart buildings," Sensors, vol. 15, no. 5, pp. 10350–10379, 2015.

[9]. dA. Singh, Y. Pandey, A. Kumar, M. K. Singh, A. Kumar, and S. C. Mukhopadhyay, "Ventilation monitoring and control system for high rise historical buildings," IEEE Sensors J., vol. 17, no. 22, pp. 7533–7541, Nov. 2017.

XII. Appendix

C code:

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <dirent.h>
#include <string.h>

#define PCF8591_ADDRESS 0x48

// Define GPIO pins
#define FLAME_SENSOR_PIN 5
#define PIR_SENSOR_PIN   13
#define RELAY_MOTOR_PIN  19
#define RELAY_PUMP_PIN   26
#define BUZZER_PIN       21
#define DS18B20_SENSOR_PATH "/sys/bus/w1/devices/"
#define IMAGE_PATH "/home/pi/captured_image.jpg"
const char *account_sid =
"ACbc4ee0a2bsd4fe52f284c13459e9e5a3";
const char *auth_token =
"473c9114aa9esdf5df910e3ad57d90c64";
const char *from_phone_number = "+12562977125";
const char *to_phone_number = "+14353478566";
const char *address = "124 St Lumen Colorado Canada J9K
3G8";
// Function to read temperature sensor (DS18B20)
float read_ds18b20_temp() {
    DIR *dir;
    struct dirent *dirent;
    char dev[16] = {0};
    char devPath[128] = {0};
    char buf[256] = {0};
    char tmpData[6] = {0};
    ssize_t numRead;
    float temperature = -1;
    dir = opendir(DS18B20_SENSOR_PATH);
    if (dir != NULL) {
        while ((dirent = readdir(dir))) {
            if (dirent->d_type == DT_LNK && strstr(dirent-
>d_name, "28-")) {
                strcpy(dev, dirent->d_name);
                break;
            }
        }
        closedir(dir);
    } else {
        perror("Couldn't open the w1 devices directory");
        return -1;
    }
    if (strlen(dev) == 0) {
        printf("No DS18B20 sensor found.\n");
        return -1;
    }
```

```
    snprintf(devPath, sizeof(devPath), "%s%s/w1_slave",
DS18B20_SENSOR_PATH, dev);
    int fd = open(devPath, O_RDONLY);
    if (fd == -1) {
        perror("Couldn't open the w1 device.");
        return -1;
    }
    numRead = read(fd, buf, sizeof(buf));
    close(fd);
    if (numRead > 0 && strstr(buf, "YES")) {
        char *tempStr = strstr(buf, "t=");
        if (tempStr != NULL) {
            strncpy(tmpData, tempStr + 2, 5);
            tmpData[5] = '\0';
            temperature = strtof(tmpData, NULL) / 1000;
        }
    } else {
        printf("Invalid data read from sensor.\n");
        return -1;
    }
    return temperature;
}
// Function to read smoke sensor connected to PCF8591
ADC
int read_smoke_sensor(int fd) {
    int command = 0x41;
    if (write(fd, &command, 1) != 1) {
        printf("Failed to write to the i2c bus.\n");
        return -1;
    }
    if (read(fd, &command, 1) != 1) {
        printf("Failed to read from the i2c bus.\n");
        return -1;
    }
    if (read(fd, &command, 1) != 1) {
        printf("Failed to read from the i2c bus.\n");
        return -1;
    }
    return command;
}
void activate_relay(int pin) {
    digitalWrite(pin, HIGH);
}
void deactivate_relay(int pin) {
    digitalWrite(pin, LOW);
}
void send_emergency_signal() {
    printf("Sending emergency signal to the nearest safety
department...\n");
}
void capture_image() {
    system("libcamera-still -o " IMAGE_PATH);
}
void send_sms_message(const char *alert_type, float
temperature, int smoke_level, int motion_detected) {
    char body[512];
    snprintf(body, sizeof(body),
             "Emergency Alert!\n"
             "Type: %s\n"
             "Address: %s\n"
             "Temperature: %.2f°C\n"
             "Smoke Level: %d\n"
             "Motion Detected: %d",
             alert_type, address, temperature, smoke_level,
motion_detected);
    // Upload the image to a file hosting service (here using
imgur as an example)
    char upload_command[2048];
    snprintf(upload_command, sizeof(upload_command),
             "curl -s -F 'image=@%s' -H 'Authorization: Client-
ID c095dsf4f91b9605' https://api.imgur.com/3/image | jq -r
'.data.link'", IMAGE_PATH);
    printf("Upload command: %s\n", upload_command);
    FILE *upload_pipe = popen(upload_command, "r");
    if (!upload_pipe) {
        printf("Failed to execute upload command.\n");
        return;
    }
    char asset_url[512];
    if (fgets(asset_url, sizeof(asset_url), upload_pipe) ==
NULL) {
        printf("Failed to read asset URL.\n");
        pclose(upload_pipe);
        return;
    }
    pclose(upload_pipe);
    asset_url[strcspn(asset_url, "\n")] = '\0';
    // Construct the SMS message with the uploaded media
URL
    char command[4096];  // Increased buffer size
    snprintf(command, sizeof(command),
             "curl -X POST 'https://api.twilio.com/2010-04-
01/Accounts/%s/Messages.json' "
             "--data-urlencode 'To=%s' "
             "--data-urlencode 'From=%s' "
             "--data-urlencode 'Body=%s\nImage URL: %s' "
             "-u %s:%s",
             account_sid, to_phone_number,
from_phone_number, body, asset_url, account_sid,
auth_token);
    printf("SMS command: %s\n", command);
    system(command);
}
int main(void) {
    int fd;
    char *fileName = "/dev/i2c-1";
    if ((fd = open(fileName, O_RDWR)) < 0) {
        printf("Failed to open the i2c bus.\n");
        return 1;
    }
    if (ioctl(fd, I2C_SLAVE, PCF8591_ADDRESS) < 0) {
        printf("Failed to acquire bus access and/or talk to
slave.\n");
        return 1;
    }
    if (wiringPiSetupGpio() == -1) {
        printf("wiringPi setup failed\n");
        return 1;
    }
    // Set sensor pins as input
    pinMode(FLAME_SENSOR_PIN, INPUT);
    pinMode(PIR_SENSOR_PIN, INPUT);
    // Set relay pins as output
    pinMode(RELAY_MOTOR_PIN, OUTPUT);
```

```
    pinMode(RELAY_PUMP_PIN, OUTPUT);
    pinMode(BUZZER_PIN, OUTPUT);
    float temperature = 0;
    float previous_temperature = 0;
    int previous_flame_detected = 0;
    int previous_motion_detected = 0;
    int previous_smoke_level = 0;
    while (1) {
        int flame_detected =
digitalRead(FLAME_SENSOR_PIN);
        int motion_detected =
digitalRead(PIR_SENSOR_PIN);
        int smoke_level = read_smoke_sensor(fd);
        temperature = read_ds18b20_temp();
        if (temperature == -1) {
            printf("Failed to read temperature\n");
        } else if (temperature < -50 || temperature > 150) {
            printf("Read an invalid temperature: %.2f C\n",
temperature);
            temperature = previous_temperature; // Use the last
valid temperature
        } else {
            printf("Temperature = %.2f C\n", temperature);
            previous_temperature = temperature; // Update the
last valid temperature
        }
        printf("Flame: %d, Smoke Level: %d, Motion: %d\n",
flame_detected, smoke_level, motion_detected);

        if ((flame_detected && !previous_flame_detected) ||
(temperature > 50)) {
            printf("Flame detected! Activating safety
protocols.\n");
            digitalWrite(BUZZER_PIN, HIGH);
            activate_relay(RELAY_PUMP_PIN);
            if (motion_detected) {
                activate_relay(RELAY_MOTOR_PIN);
            }
            send_emergency_signal();
            capture_image();
            send_sms_message("Fire Alert", temperature,
smoke_level, motion_detected);
        } else if ((smoke_level > 128 && smoke_level !=
previous_smoke_level) || (temperature > 35)) {
            printf("Smoke detected! Activating safety
protocols.\n");
            digitalWrite(BUZZER_PIN, HIGH);
            activate_relay(RELAY_MOTOR_PIN);
            send_emergency_signal();
            capture_image();
            send_sms_message("Smoke Alert", temperature,
smoke_level, motion_detected);
        } else if (motion_detected &&
!previous_motion_detected) {
            printf("Motion detected! Possible theft.\n");
            digitalWrite(BUZZER_PIN, HIGH);
            send_emergency_signal();
            capture_image();
            send_sms_message("Theft Alert", temperature,
smoke_level, motion_detected);
        } else {
            digitalWrite(BUZZER_PIN, LOW);
            deactivate_relay(RELAY_MOTOR_PIN);
            deactivate_relay(RELAY_PUMP_PIN);
        }
        previous_flame_detected = flame_detected;
        previous_motion_detected = motion_detected;
        previous_smoke_level = smoke_level;
        delay(1000);
    }
    close(fd);
    return 0;
}
```