Microprocessor Interfacing & Embedded System

Project Title: Fire Protection System using Arduino.

Group Members:

1. Mohamed Ahmed Galal Mohamed
2. Mazen Sayed
3. Aboubaker Adel
4. Mazen Magdy
5. Ahmed Gamal Aldin
6. Abdallah Mohamed
7. Khaled Mohamed

The objectives of the microcontroller-based fire safety system project in an industrial setting can be divided into key areas to ensure comprehensive safety and functionality. The primary aim is to create a robust system that detects fire hazards early and initiates appropriate safety responses.

One of the main objectives is to provide fire detection and prevention. This involves using sensors to monitor for smoke, gas leaks, and other indicators of fire risk. By implementing this early warning system, the goal is to identify potential hazards before they escalate into larger incidents.

The second objective is to establish automated safety responses. Once a fire hazard is detected, the system should automatically shut off the main power source, close gas valves, and activate fire suppression systems. This automated response is designed to minimize damage and contain the hazard, thereby ensuring the safety of workers and reducing property damage.

Sending alerts and notifications to key personnel and local emergency services is another crucial objective. By providing immediate alerts to managers, safety officers, and nearby fire stations, the system ensures a rapid response from both internal and external resources. This objective enhances communication and coordination during emergencies.

Continuous monitoring and real-time feedback are also central to the project's objectives. The system must continuously monitor the industrial environment, providing real-time information to operators. This capability allows for quick action and evacuation if needed, contributing to a safer workplace.

Applications:

The applications of a microcontroller-based fire safety system extend across various industrial and commercial settings. This technology can be used to enhance safety, prevent disasters, and protect both human lives and valuable assets.

Working Procedures:

1. The main component in the circuit is the Arduino Mega. It controls everything.
2. When the system is powered on it automatically starts to sense the environment for toxic gas or flames.
3. The sensors will send this data to Arduino Mega. Arduino Mega will examine them according to the installed program.
4. If the flame is detected the system will activate the emergency alarm, disconnect the building's primary power source, shut down the main gas valve to stop gas flow, activate an emergency high-pressure exhaust fan to remove leakage gas, and GSM & GPS module to send notification and position to the firefighter and authority.
5. During gas leakage the gas valve will be shut off, the alarm & exhaust fan will be activated and notification will be sent.

Block Diagram:

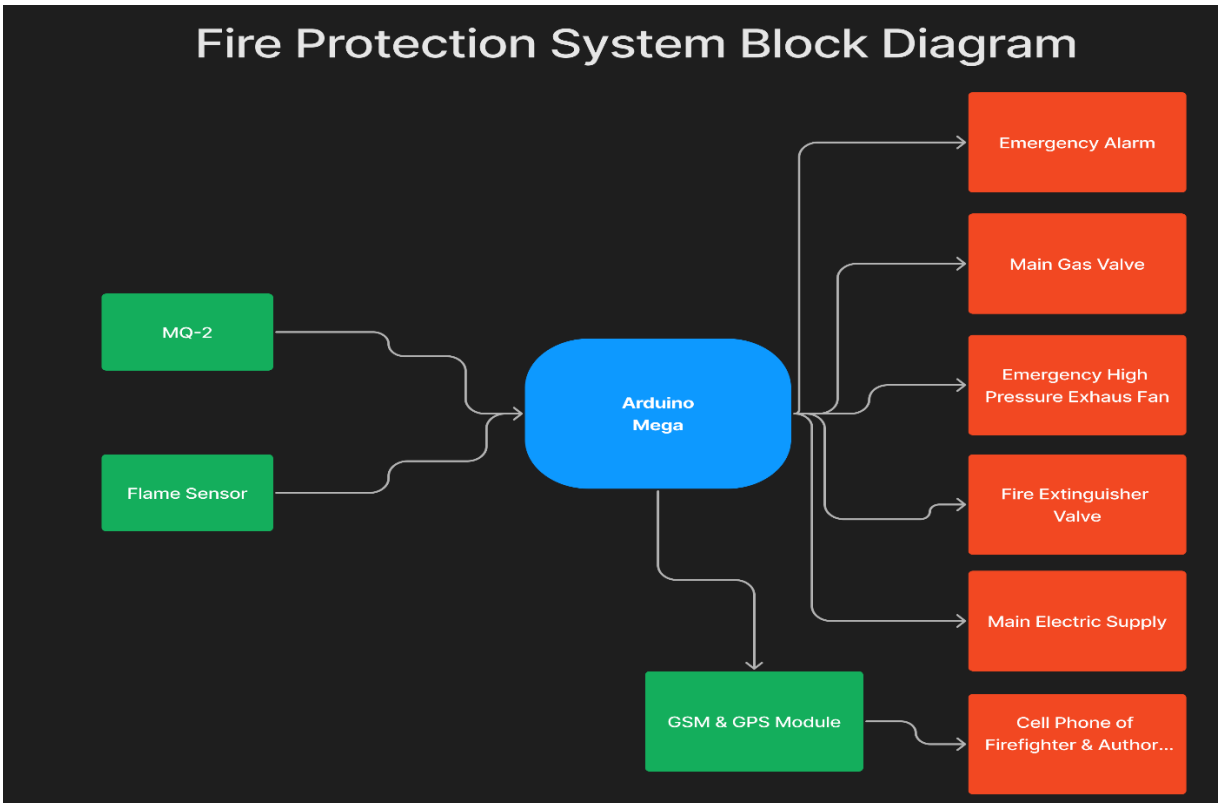

Figure 1: System Block Diagram

Simulation Schematic:

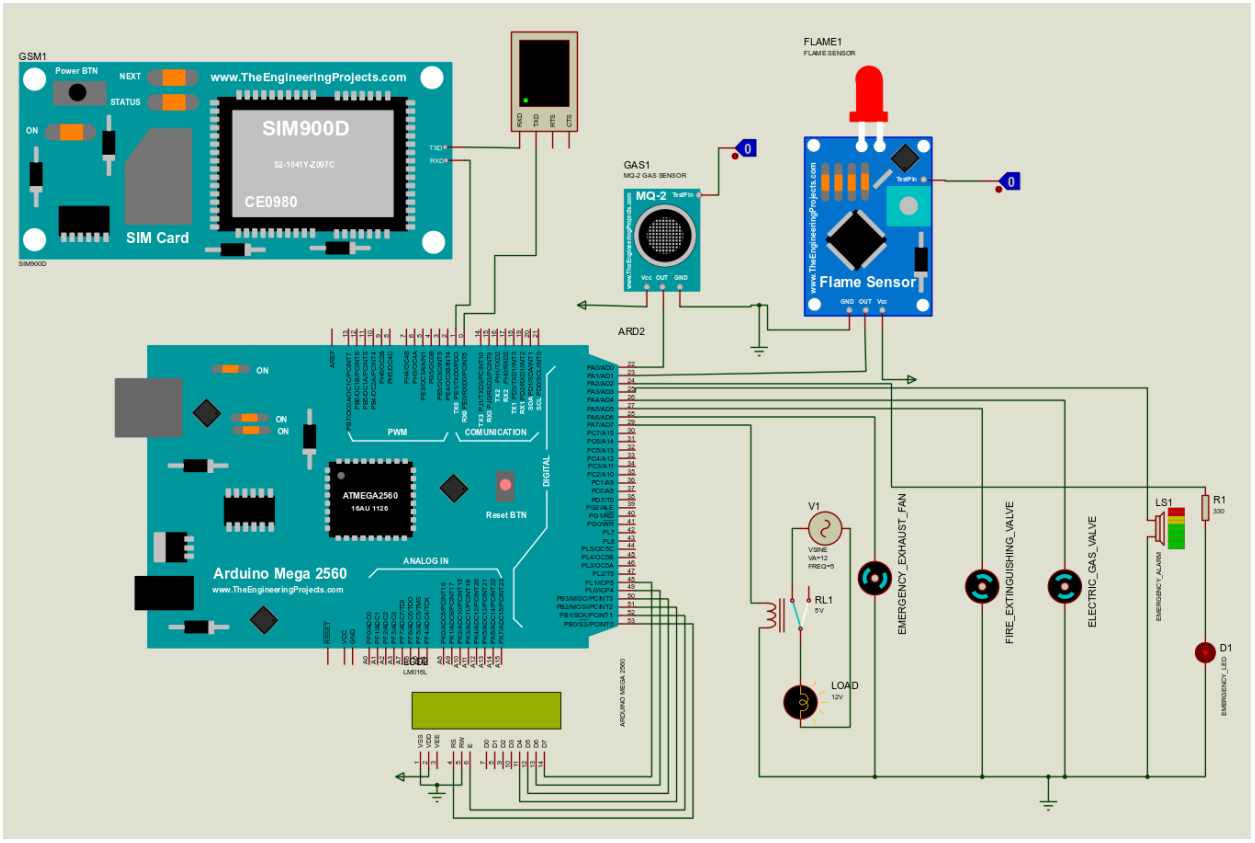

Figure 2: Simulation Schematic Circuit Diagram

Simulation using Arduino Uno:

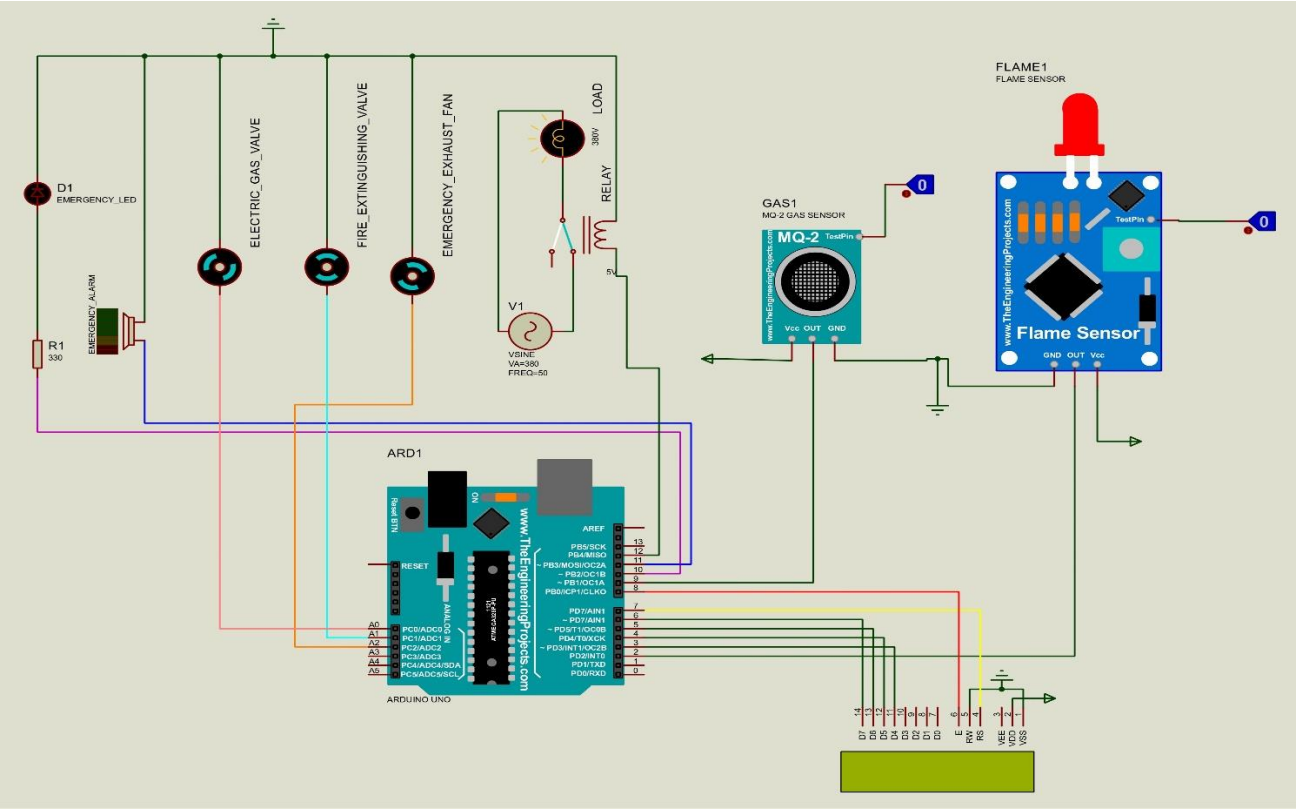

Figure 3: Simulation using Arduino Uno

Practical Approach using Arduino Uno:

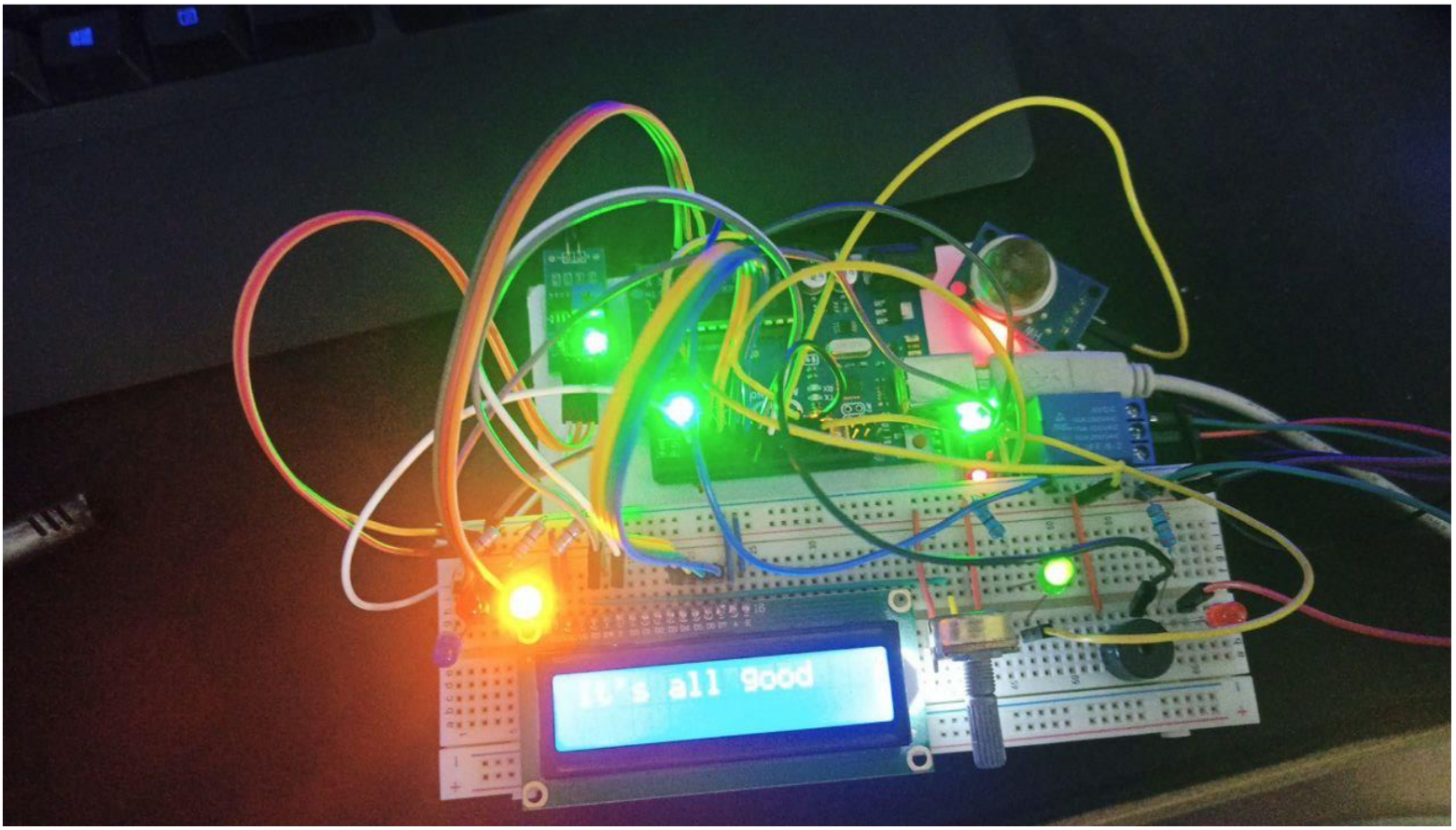

Figure 4: Practical Approach using Arduino Uno

Code:

Fire Protection System Using Arduino Uno

```
/* Includes Section*/
#include <Wire.h>
#include <SoftwareSerial.h>
#include "LiquidCrystal.h"

// Initialize the library by associating any needed LCD interface pin
// with the arduino pin number it is connected to
const int rs = 7, en = 8, d4 = 3, d5 = 4, d6 = 5, d7 = 6;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

/* Variables Section*/
#define Gas_Sensor_Output 9    // The output signal of the gas detector
#define Flame_Sensor_Output 2  // The output signal of the flame sensor
#define Emergency_LED 10
#define Emergency_Alarm 11
#define ELECTRIC_GAS_VALVE A0
#define FIRE_EXTINGUISHING_VALVE A1
#define Emergency_Exhaust_Fan A2
#define LOAD 12

bool Flame_Sensor_state;
bool Gas_Sensor_state;

void setup()
{
  /* Setting Pin Modes */
  pinMode(Flame_Sensor_Output, INPUT);
  pinMode(Gas_Sensor_Output, INPUT);
  pinMode(Emergency_Alarm, OUTPUT);
  pinMode(Emergency_LED, OUTPUT);
  pinMode(ELECTRIC_GAS_VALVE, OUTPUT);
  pinMode(FIRE_EXTINGUISHING_VALVE, OUTPUT);
  pinMode(Emergency_Exhaust_Fan, OUTPUT);
  pinMode(LOAD, OUTPUT);

  digitalWrite(LOAD, HIGH);  // Connect the main building's electricity.
  digitalWrite(ELECTRIC_GAS_VALVE, HIGH);   // Activate the main gas valve

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("It's All good");

  // Begin the serial connection with baud rate 9600
  Serial.begin(9600);
}

void loop()
{
  Flame_Sensor_state = digitalRead(Flame_Sensor_Output);  // Check the output signal of the flame sensor.
  Gas_Sensor_state = digitalRead(Gas_Sensor_Output);      // Check the output signal of the gas sensor.

  // If the flame is detected
  if (Flame_Sensor_state == LOW)
  {
    digitalWrite(Emergency_Alarm, HIGH);  // Activate the emergency alarm.
    digitalWrite(Emergency_LED, HIGH);    // Activate the emergency alarm.
    digitalWrite(LOAD, LOW);              // Disconnect the building's primary power source.
    digitalWrite(ELECTRIC_GAS_VALVE, LOW);          // Shut down the main gas valve to stop gas flow.
    digitalWrite(FIRE_EXTINGUISHING_VALVE, HIGH);   // Activate the extinguishing valve to put down the fire.
    digitalWrite(Emergency_Exhaust_Fan, HIGH);      // Activate an emergency high-pressure exhaust fan to remove leakage gas.

    /* Display on the LCD "Flame Detected!!"*/
    lcd.setCursor(0, 0);
    lcd.print("Fire Alert!!!!");

```

Figure 5: The code of the practical approach using Arduino Uno – First Part

```
    lcd.setCursor(0, 1);
    lcd.print("Flame Detected!!");
  }
  else if (Gas_Sensor_state == LOW)
  {
    digitalWrite(Emergency_Alarm, HIGH);  // Activate the emergency alarm.
    digitalWrite(Emergency_LED, HIGH);    // Activate the emergency alarm.
    digitalWrite(ELECTRIC_GAS_VALVE, LOW);          // Shut down the main gas valve to stop gas flow.
    digitalWrite(Emergency_Exhaust_Fan, HIGH);      // Activate an emergency high-pressure exhaust fan to remove leakage gas.

    /* Display on the LCD "Gas Detected!!"*/
    lcd.setCursor(0, 0);
    lcd.print("Fire Alert!!!!");
    lcd.setCursor(0, 1);
    lcd.print("Gas Detected!!");
  }
  else if (Flame_Sensor_state != LOW && Gas_Sensor_state != LOW)
  {
    delay(500);
    digitalWrite(Emergency_Alarm, LOW);  // Deactivate the emergency alarm.
    digitalWrite(Emergency_LED, LOW);    // Deactivate the emergency alarm.
    digitalWrite(LOAD, HIGH);            // Connect the building's primary power source.
    digitalWrite(ELECTRIC_GAS_VALVE, HIGH);         // Open the main gas valve to stop gas flow.
    digitalWrite(FIRE_EXTINGUISHING_VALVE, LOW);    // Deactivate the extinguishing valve to put down the fire.
    digitalWrite(Emergency_Exhaust_Fan, LOW);       // Deactivate an emergency high-pressure exhaust fan to remove leakage gas.

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("It's all good");
  }
}
```

Figure 6: The code of the practical approach using Arduino Uno – Second Part

Discussion:

Pin Assignments and Initialization

- The code begins by assigning the various components (LCD, gas sensor, flame sensor, emergency LEDs, etc.) to specific Arduino pins. The LiquidCrystal object is initialized to interface with the LCD screen, and the SoftwareSerial object is initialized to communicate with the GSM module for sending SMS alerts.
- The setup() function sets the pin modes for all inputs and outputs. This function also initializes the LCD and serial communication at a 9600 baud rate. It also sets default states for certain actuators, like connecting the main building's electricity (LOAD) and activating the gas valve.

Main Functionality in the Loop

1. The loop() function continuously checks the status of the gas and flame sensors to determine if there is a hazardous condition. If either the flame or gas sensor is activated, the program initiates a series of safety measures, including:
   - Displaying alerts on the LCD screen.
   - Activating an emergency alarm and LED to indicate danger.
   - Deactivating the main power source to reduce risk.
   - Closing the gas valve to stop gas flow.
   - Activating the fire extinguishing valve and an exhaust fan to mitigate fire and gas hazards.
   - Sending an SMS notification using the SendSMS() function.
2. If no hazards are detected, the system resets to a safe state, reconnecting power, reopening the gas valve, and deactivating the emergency measures.
3. The loop also includes a timed refresh for the LCD to clear and display a default message ("It's all good") when no hazards are detected.
4. The SendSMS() function sends an SMS notification to a predefined number. It sends the message once and sets a timer to prevent repeated SMS notifications during the same incident.