Project Report

Smart Fire Detector System using ESP32

Course: Embedded Systems and IoT LAB

Course code: SWE-466

Submitted To :

Nawshad Ahmed Chowdhury

Assistant Professor and Head

Department of EEE

Metropolitan University Bangladesh

Submitted By :

MD Abdus Salam Shanto (213-134-005)

Fahmida Rahman Anu (213-134-007)

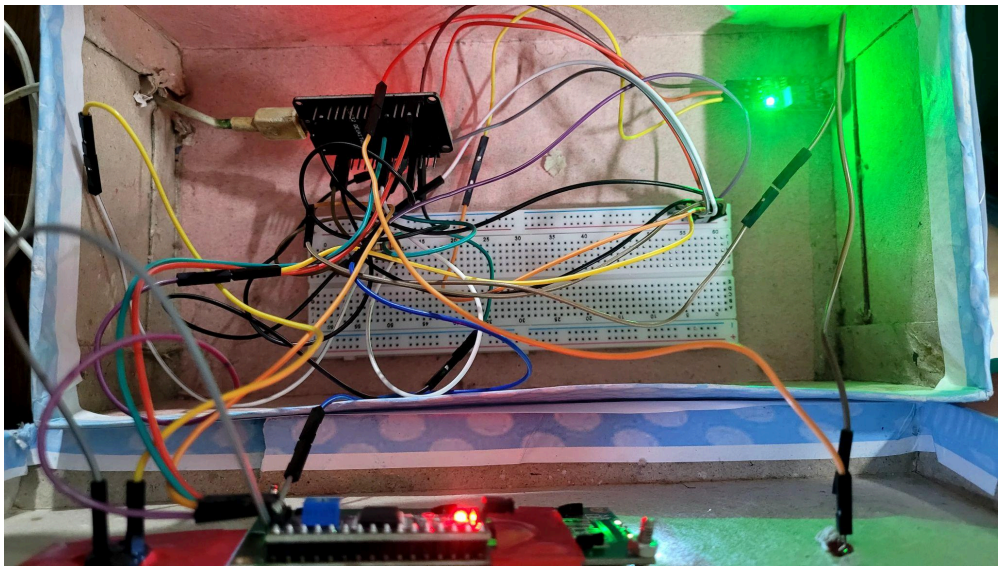
Nabila Rayhana Islam (221-134-006)

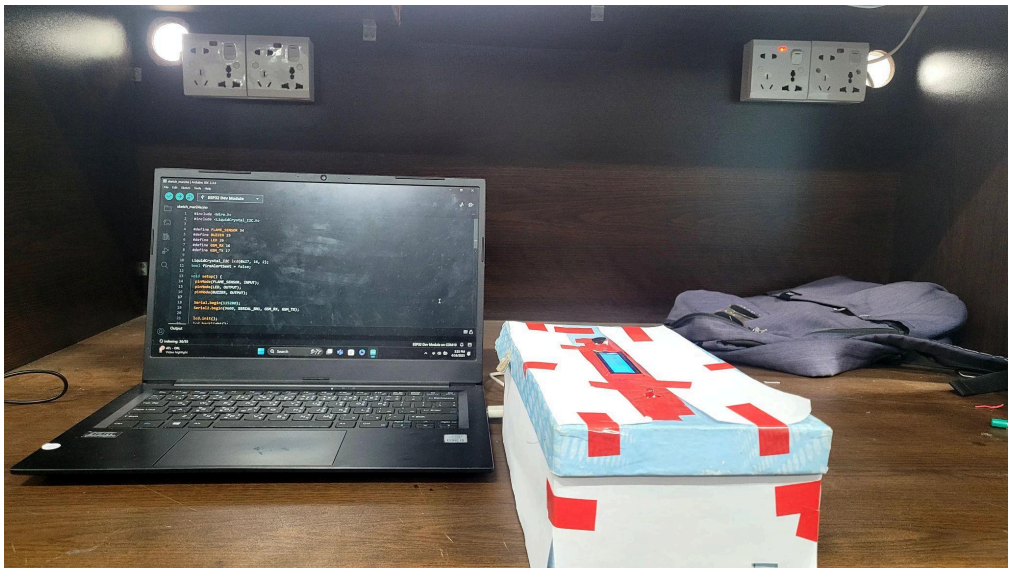
Prottoy Chakroborty (221-134-035)

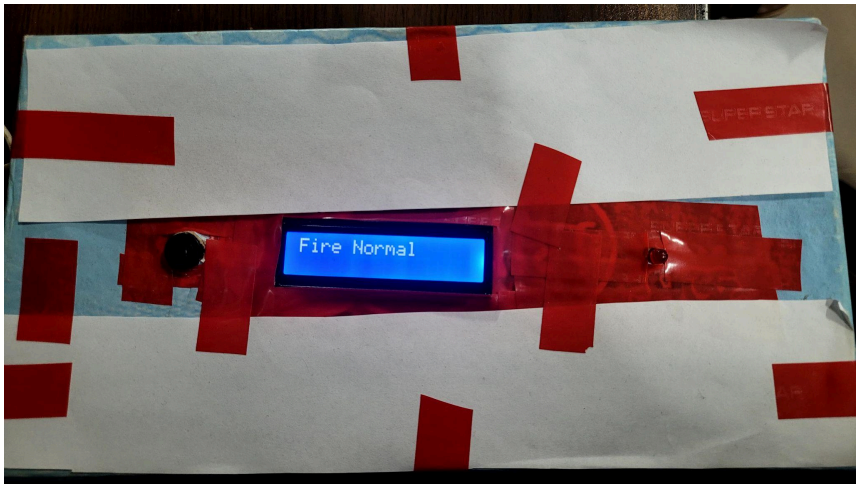
Objective :

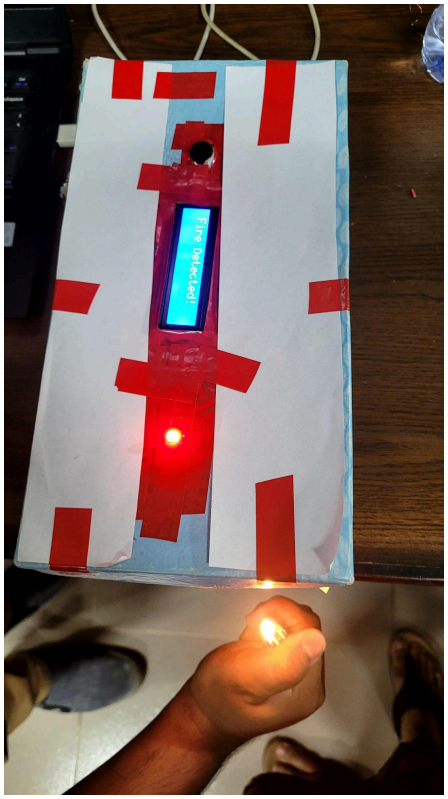
To design and implement a Smart Fire Detection System using ESP32 that can detect fire using a flame sensor, alert users via buzzer and LED, display system status on an LCD, and (originally intended to) send SMS notifications to a predefined phone number using a GSM module.

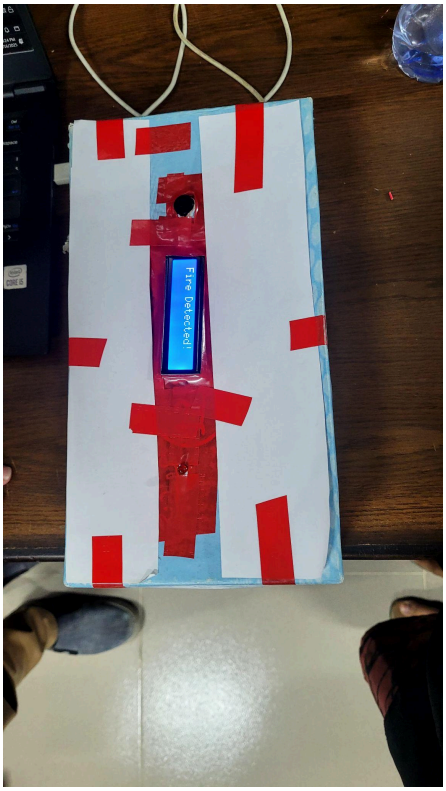
Project Photographs:











- LCD Display Showing Status
- Flame Sensor Wiring
- Full Breadboard Setup
- GSM Module Connection (Before damage)
- ESP32 Pin Map

Procedure:

1. **Component Selection:** Selected ESP32, flame sensor, GSM module (SIM800L), LCD with I2C, buzzer, LED, jumper wires, breadboard, and external 5V power supply.
2. **Circuit Design:**
    - Flame Sensor connected to GPIO 34
    - Buzzer connected to GPIO 25
    - LED connected to GPIO 26
    - LCD via I2C (SDA: GPIO 21, SCL: GPIO 22)
    - GSM TX to GPIO 17, RX to GPIO 16 (before module failure)
3. **Programming:**
    - The flame sensor returns LOW when the fire is detected.
    - LCD shows "Fire Detected" or "Fire Normal" based on input.
    - Buzzer and LED alert on fire.
    - GSM-based SMS was implemented in code, but testing was limited due to hardware failure.
4. **Testing & Finalization:**
    - Multiple rounds of debugging and real flame testing.
    - Buzzer, LED, and LCD feedback working.
    - SMS functionality was coded but not demonstrated due to a damaged GSM module.

Problem Solving:

- **Buzzer Not Working**

Solved by: ***MD Abdus Salam Shanto***
Initially, the buzzer did not respond. A 220kΩ resistor was used, which restricted current and prevented the buzzer from functioning. Shanto replaced it with a 220Ω resistor, and the buzzer started working correctly. To ensure the issue wasn't related to power, he also tested the circuit using a 3.7V battery and a direct 5V adapter. These tests helped confirm that the buzzer problem was due to incorrect resistance. During this power testing, the GSM module was also checked and unfortunately identified as damaged, so it could not be used in the final demo.

- **Flame Sensor Not Detecting Fire**

Solved by: ***Prottoy Chakroborty***
The sensor was first connected to an AO (analog pin), which is used for motion detection. Prottoy corrected the connection to DO (digital output), which is proper for flame detection. The sensor then worked perfectly.

- **LCD Not Displaying Text**

Solved by: ***Nabila Rayhana Islam***
LCD powered on but showed no characters. Nabila realized that the I2C module was missing. Once she added the I2C converter and rechecked the wiring, the LCD worked fine using `LCD.init()`.

- **LED Not Blinking**

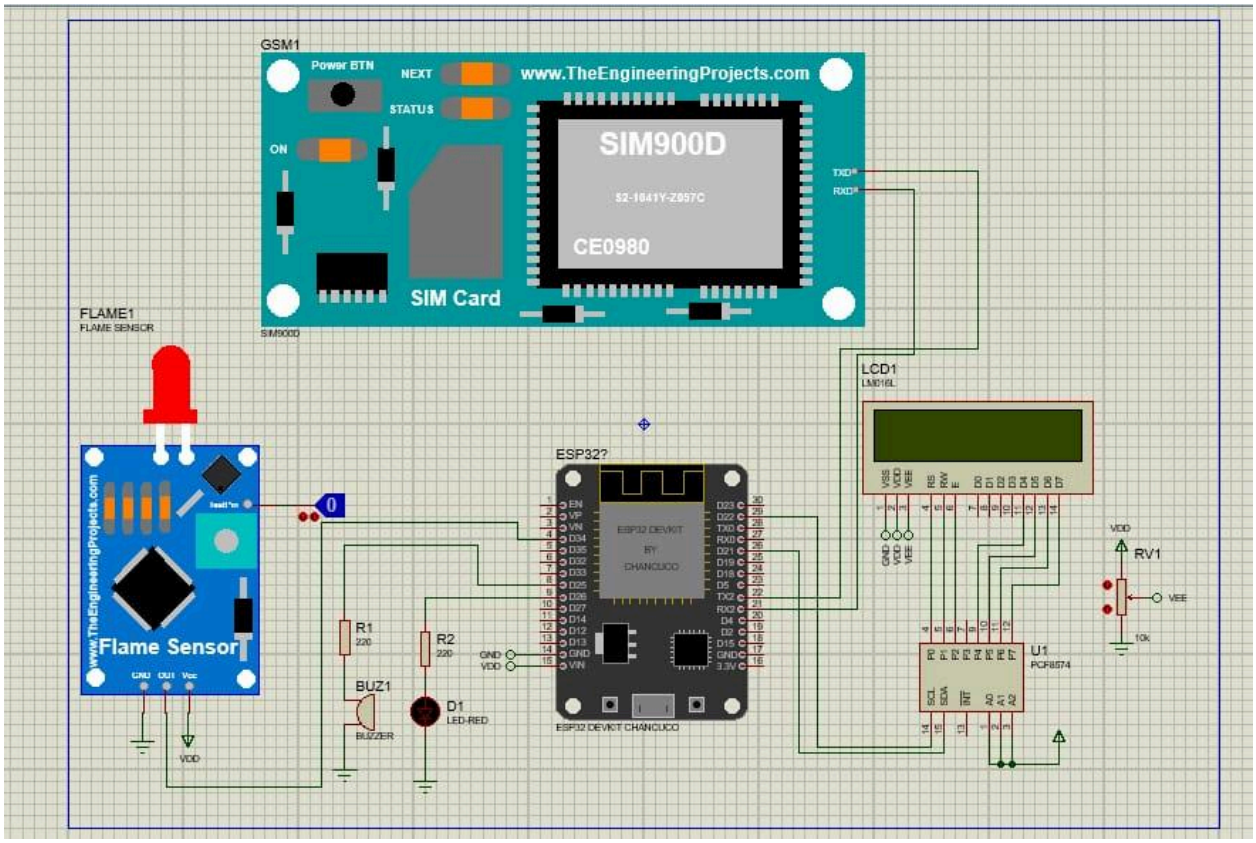
Solved by: ***Fahmida Rahman Anu***
Anu first used a 220kΩ resistor which caused the LED to stay off. She replaced it with a 220Ω resistor and implemented LED blinking in code, which fixed the issue.

- **Code Debugging & Integration**

Solved by: *All Team Members*
The whole team collaborated to fix syntax errors, GSM timing issues, and flame sensor logic. Together, we ensured the system was stable and responsive.

Appendix:

- Arduino Sketch (github)
- Circuit Diagram Image:



- Port Map:
  - Flame Sensor: GPIO 34
  - Buzzer: GPIO 25
  - LED: GPIO 26
  - LCD I2C: SDA = GPIO 21, SCL = GPIO 22
  - GSM Module: TX = GPIO 17, RX = GPIO 16 (planned, not functional in the final demo)