KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI

COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

IoT-BASED FIRE ALARM SYSTEM – DETECTS SMOKE AND ALERTS VIA AN IOT DASHBOARD

MEMBERS

AKOSAH TRISHELLE MAAME AMA POKUAA – 1815522

PAPA YAW OWUSU ASIAMAH – 1818422

IBRAHIM IBRAHIM – 7100221

NADUTEY PARTEY SAMUEL – 1824922

ABA FAMOUS MAWULENU KWAKU – 1812022

AMOAKO SOMAILA – 1816422

DJANGMAH PRINCE – 1821622

NICOLE KUSIWAA DANSO – 1821022

AKUETTEH GIFTY NAA TEKI – 1815722

ASANTE OPOKU CHRISTOS - 1817922

TABLE OF CONTENTS

PROBLEM IDENTIFICATION

Fire accidents pose a significant hazard to life and property. Traditional fire alarm systems provide only local alerting, making timely intervention difficult, especially when no one is present. The aim of this project is to develop an **IoT-based fire alarm system** that detects fire and smoke using sensors and provides alerts to users locally. By leveraging a virtual environment, the system enables real-time monitoring and faster response times, enhancing safety and reducing potential damage.

METHODOLOGY

The fire alarm system consists of multiple sensors interfaced with a microcontroller that collects and processes data. Unlike cloud-based IoT solutions, this system utilizes an LCD display and a buzzer for local alerts. The key steps involved include:

- **Sensor interfacing** – Connecting sensors to a microcontroller for data collection.
- **Microcontroller programming** – Writing code to process sensor readings and trigger alerts.
- **Local alert system** – Displaying warnings on an LCD and triggering an audible buzzer.
- **Simulation and testing** – Using a virtual environment to validate system functionality.

HARDWARE COMPONENTS

Although this project is a simulation, the hardware components modelled in the Wokwi diagram and JSON document are tailored for fire detection and alerting. Below is a detailed list of the components used:

- **ESP32 (ESP32-DevKitC-V4):** The ESP32 microcontroller serves as the system's central processing unit. It collects sensor data, processes it, and triggers local alerts via an LCD and a buzzer.
- **MQ-2 Smoke Sensor:** The MQ-2 gas sensor is used to detect smoke and harmful gases such as carbon monoxide, LPG, and methane. It provides both analog (AOUT) and digital (DOUT) outputs, which are connected to the ESP32 (pins 34 and D2, respectively) to measure gas concentration. If smoke is detected, the system triggers an alert.
- **Buzzer:** A buzzer provides an audible alarm to warn individuals nearby of a fire hazard. It is connected to the ESP32 (signal pin 16, GND as ground) and is activated when smoke is detected.
- **LCD Display:** An LCD screen is used to display fire alerts and sensor readings, compensating for the absence of an IoT cloud platform.
- **LED (Red):** A red LED provides a visual indicator for fire detection. It is wired to the ESP32 (anode on pins 26 and 22, cathode on GND). When the system detects smoke, the LED turns on, providing an additional warning signal alongside the buzzer.
- **Power Supply:** The ESP32 microcontroller is powered through a 5V supply via USB or the VIN pin. All components (sensor, LED, buzzer, LCD) derive power from the ESP32, ensuring a stable power distribution.

SYSTEM DESIGN AND WORKING PRINCIPLE

The fire alarm system follows a structured design consisting of three key modules:

Sensing Module

- The system continuously monitors environmental conditions using a smoke sensor.
- When smoke or abnormal gas levels are detected, the sensor generates a signal indicating a possible fire outbreak.

Processing Unit

- The ESP32 microcontroller receives sensor readings and processes the data.
- The sensor values are compared against predefined threshold limits to determine if a fire condition exists.

Alert System

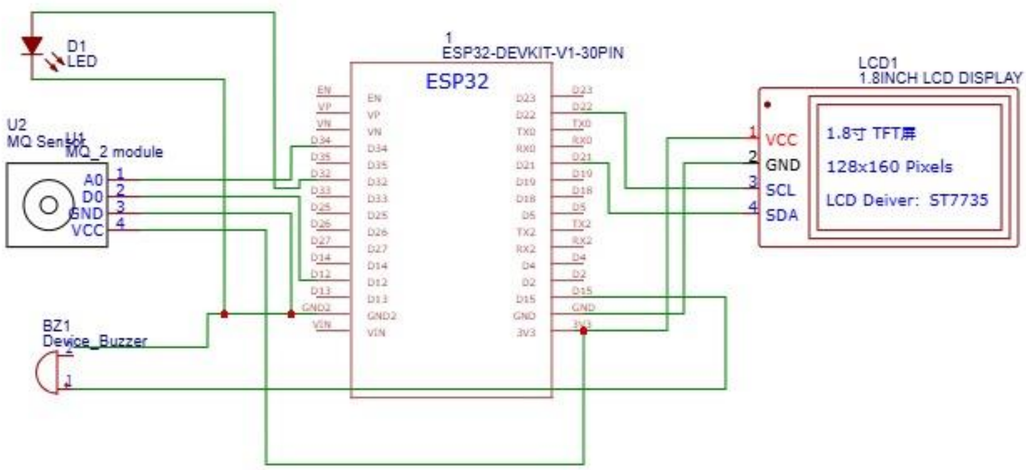
- If a fire is detected, an LCD display provides a visual warning message.
- A buzzer and LED are activated to provide an audible and visual alarm.
- This design ensures **real-time monitoring and rapid alerts** without requiring an IoT cloud platform.

IMPLEMENTATION AND DEMONSTRATION

Since this is a simulation-based project, implementation is done in a virtual environment using Wokwi. Key aspects of the demonstration include:

- **Software Simulation:** The Wokwi virtual simulator is used to model the behavior of the fire alarm system.
- **Testing Scenarios:** Simulated fire conditions validate the system's accuracy and responsiveness.
- **Local Alert System:** Fire alerts are displayed on an LCD and accompanied by an audible buzzer.
- **System Architecture:** A detailed block diagram illustrates the interaction between sensors, the microcontroller, and the local alert components.

SCHEMATIC



CONCLUSIONS AND FUTURE ENHANCEMENTS

Conclusions

This fire alarm system presents an efficient and innovative solution for fire detection. By utilizing a virtual environment, the system demonstrates how local alerts via an LCD display and a buzzer can effectively warn individuals in the event of a fire. The simulation successfully validates the system's ability to detect fire hazards and trigger immediate alerts without relying on an IoT cloud platform.

Future Enhancements

To further improve the system, the following enhancements can be considered:

- **IoT Cloud Integration:** Adding cloud-based remote monitoring for wider accessibility.
- **AI-Based Fire Prediction:** Implementing machine learning algorithms to predict fire outbreaks based on historical data.
- **Camera Integration:** Adding a camera module to visually verify fire incidents.
- **Mobile App Development:** Creating a dedicated mobile application for improved user accessibility.

This project highlights the importance of real-time fire safety applications and serves as a foundation for future developments. With additional advancements, this system can be expanded into a fully functional, hardware-based implementation.

REFERENCES

- Wokwi. (n.d.). **Wokwi IoT Simulator**. Retrieved from https://wokwi.com
- EasyEDA. (n.d.). **EasyEDA Documentation**. Retrieved from https://docs.easyeda.com
- Espressif Systems. (n.d.). **ESP32 Documentation**. Retrieved from https://docs.espressif.com/projects/esp-idf/en/latest/esp32
- Arduino. (n.d.). **Arduino Documentation**. Retrieved from https://docs.arduino.cc