Software Design Specification

for

Fire detection system

Name	ID
FAHAD ZAID HAMZI	202208800
KHALID YAHYA ALFAIFI	200887040
AHMED AFGHANI	202208980

Table of Contents

Table of Contents **Error! Bookmark not defined.**
Revision History 2
1. Introduction 3
1.1 Purpose 1
1.2 Scope 1
1.3 Overview of the System 1
2. Architectural Design 1
2.1 System Architecture 1
2.2 Hardware Architecture 2
2.3 USE Case Diagram: 3
2.4 Software Architecture 4
2.5 Data Flow Diagrams 4
2.6 Activity Diagrams 4
3. Database Design 5
3.1 Data Models 5
3.2 Database Schema 5
3.3 Database Entity Relationship Diagram 6
4. User Interface Design 6
4.1 User Interface Description 6
4.2 Wireframes 6
5. Security Design 6
5.1 Security Requirements 6
5.2 Authentication and Authorization Design 6
5.3 Encryption Design 6
6. Performance Design: 6
6.1 Performance Requirements: 6
6.2 Load Testing Plan: 7
6.3 Performance Optimization Techniques: 7
7. Error Handling and Logging Design: 7
7.1 Error Handling Plan: 7
7.2 Logging Plan: 7
8. Testing Design: 7
8.1 Testing Requirements: 7
8.2 Testing Plan: 7
8.3 Test Cases: 8
9. Deployment Design: 8
9.1 Deployment Plan: 8
9.2 Rollback Plan: 8
9.3 Disaster Recovery Plan: 8
10. Maintenance and Support Design: 8
10.1 Maintenance Plan: 8
10.2 Support Plan: 8

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this Software Design Specification (SDS) document is to describe the design of a system that connects an ESP32 microcontroller to two sensors (Temperature and Gas detector) to detect fire. The system sends an alarm to a cloud platform when there is a fire, sends a notification every hour to indicate that it is awake, and sends notifications every 5 seconds during a fire. There is a maintenance mode during which the system sends a notification every 20 minutes and no fire alarms are sent.

1.2 Scope

The system will consist of ESP32 microcontroller, temperature sensors, gas sensors, and a cloud platform. The software will include the firmware for the microcontroller and cloud platform integration. The system will be designed to detect fires accurately, send different tpye of alarms, and provide maintenance notifications.

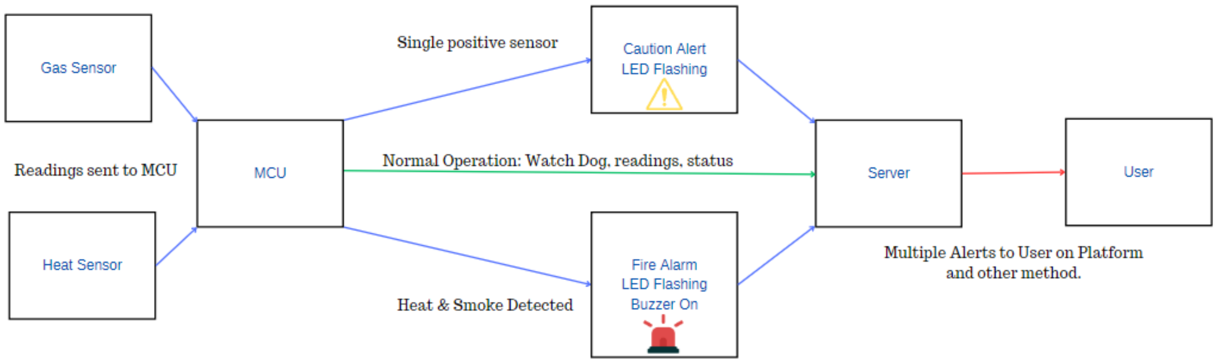
1.3 Overview of the System

The system will consist of atleast two sensors (Temperature and Gas detector) connected to an ESP32 microcontroller. The microcontroller will be connected to a cloud platform. The microcontroller will collect data from the sensors and determine if there is a fire depending on certain circumnstances. The system shall automatically detect false fire alarms and sent a caution alert, If a fire is determined and detected, the microcontroller will send an alarm to the cloud platform. The microcontroller will also send notifications every hour to indicate that it is awake, and every 5 seconds during a fire. In maintenance mode, the microcontroller will send notifications every 20 minutes.

2. Architectural Design

2.1 System Architecture

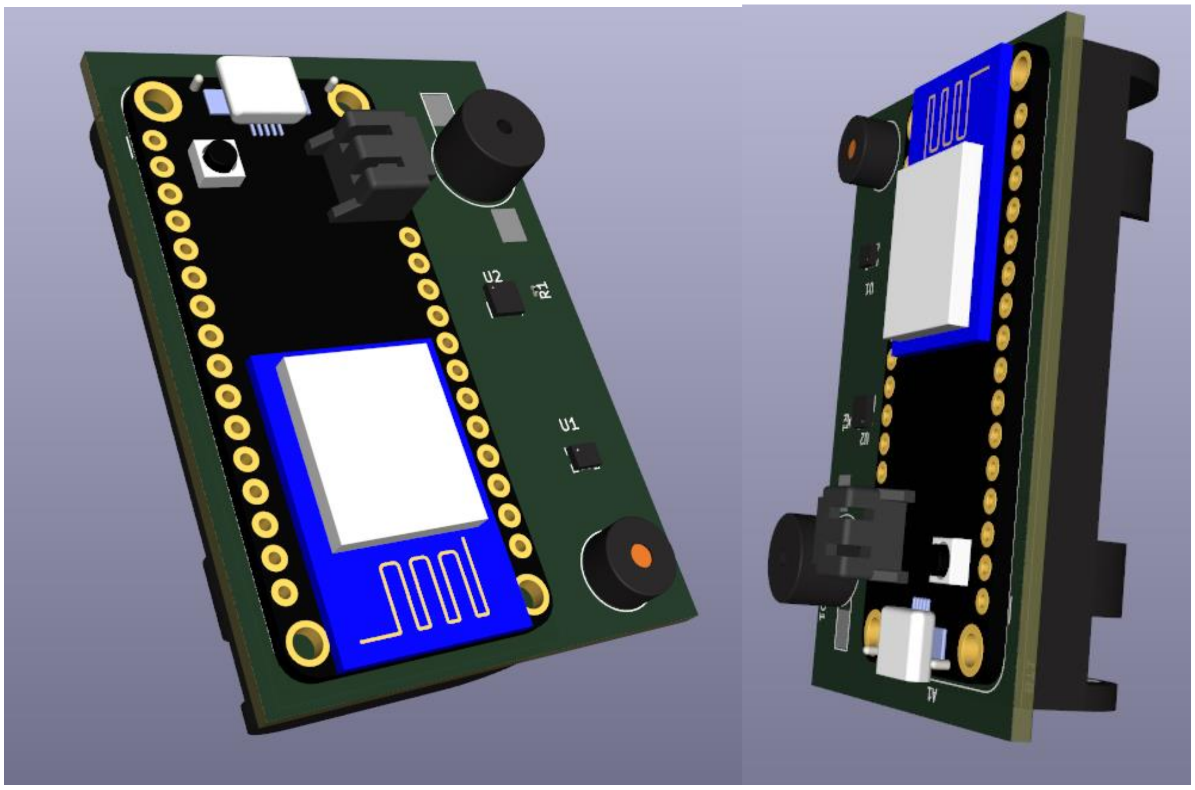
The system will consist of three main components: the ESP 32 microcontroller, the temperature and gas sensors, and the cloud platform. The microcontroller will be the central component, receiving data from the sensors and communicating with the cloud platform. The sensors will be connected to the microcontroller using GPIO pins, and the cloud platform will be accessed through Wi-Fi.

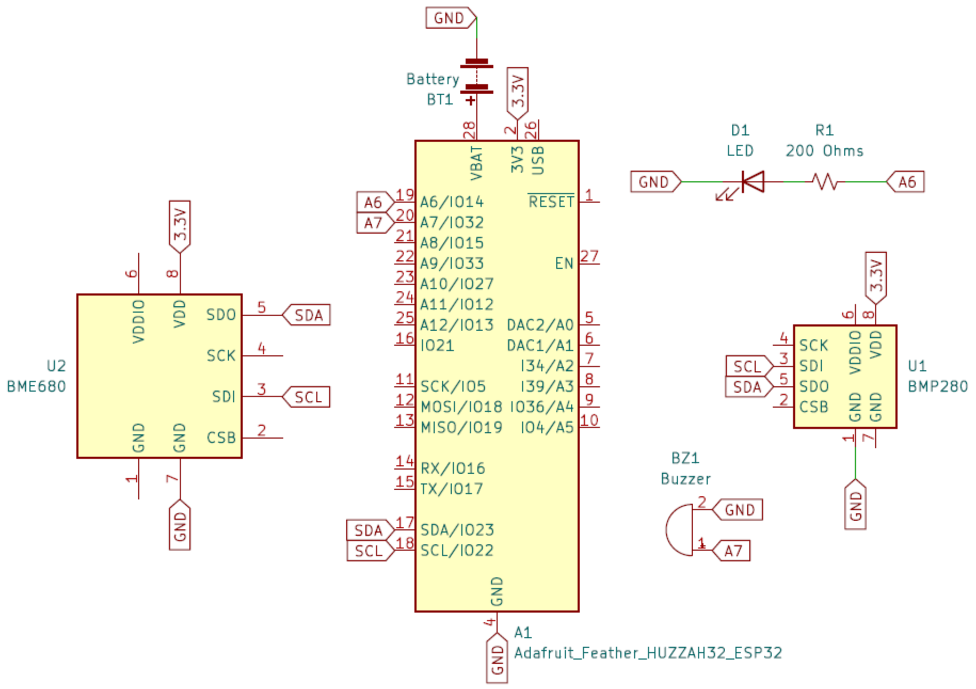


2.2 Hardware Architecture

The hardware architecture will consist of an ESP32 microcontroller, a digital temperature sensor such as BMP280, and a gas detection sensor. The temperature sensor will be connected to the microcontroller using the I2C interface, and the gas sensor will be connected using analog or digital input pins.

Working Devices with Possible System Layout:





2.3 USE Case Diagram:

By creating a use case diagram for this project, we can easily identify the actors who interact with the system, the different use cases or functionalities that the system offers, and the relationships between them.

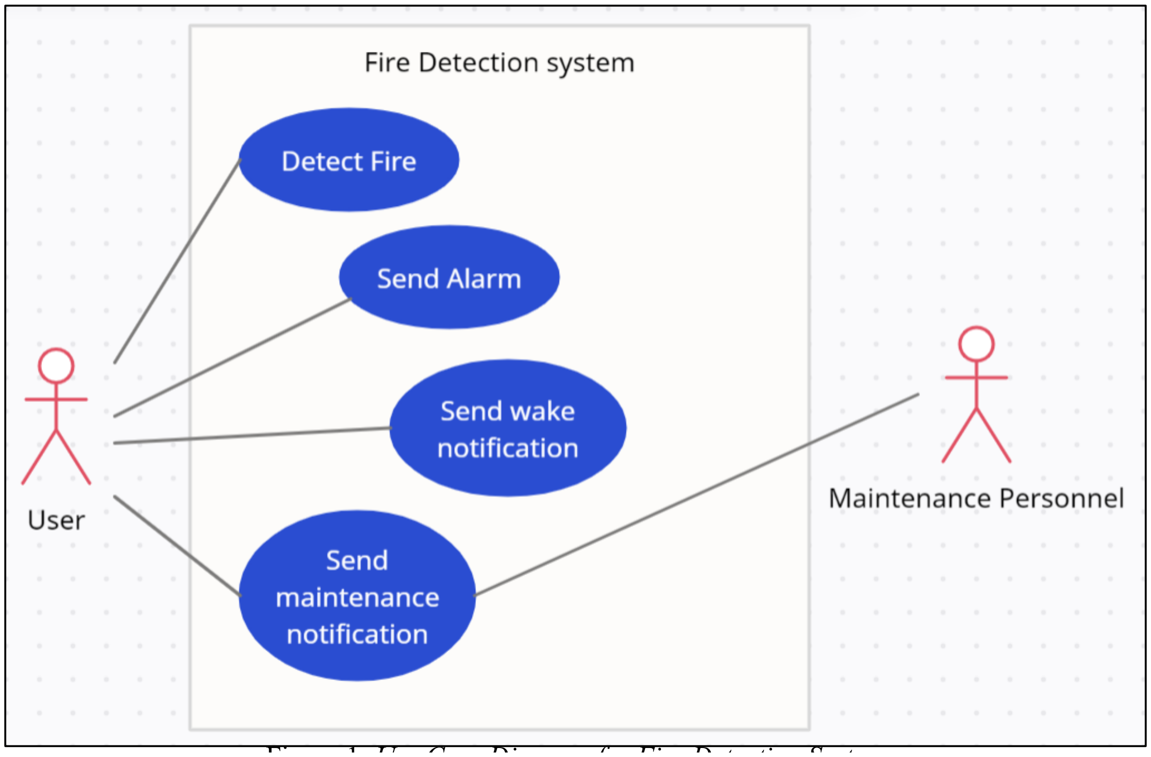


Figure 1: *Use Case Diagram for Fire Detection System*

2.4 Software Architecture

The software architecture will consist of firmware for the microcontroller and cloud platform integration. The firmware will be responsible for collecting data from the sensors, determining if there is a fire, and sending alarms and notifications to the cloud platform.
There are two levels of protections required to determine that an actual fire has accured, heat and gas, not only gas or heat as this may indicate false alarm.
If the MCU detects heat or gas only, it will immediately send caution alert to the cloud server and will be actively updating with more data.
Afterwards, it will either clear the caution alert or determine an actual fire occurance and send a high level alarm using the cloud server.
It can also be connected to fire suppression panel or system to give the signal to activate water sprinklers or gas suppression system.

2.5 Data Flow Diagrams

The data flow diagrams for the system will illustrate the flow of data between the sensors, microcontroller, and cloud platform. The diagrams will be used to help visualize the system and ensure that all data is being properly transmitted.

2.6 Activity Diagrams

The flow of actions that occur when a fire is detected by the system, from the point of detection to the point of notification in the cloud platform.

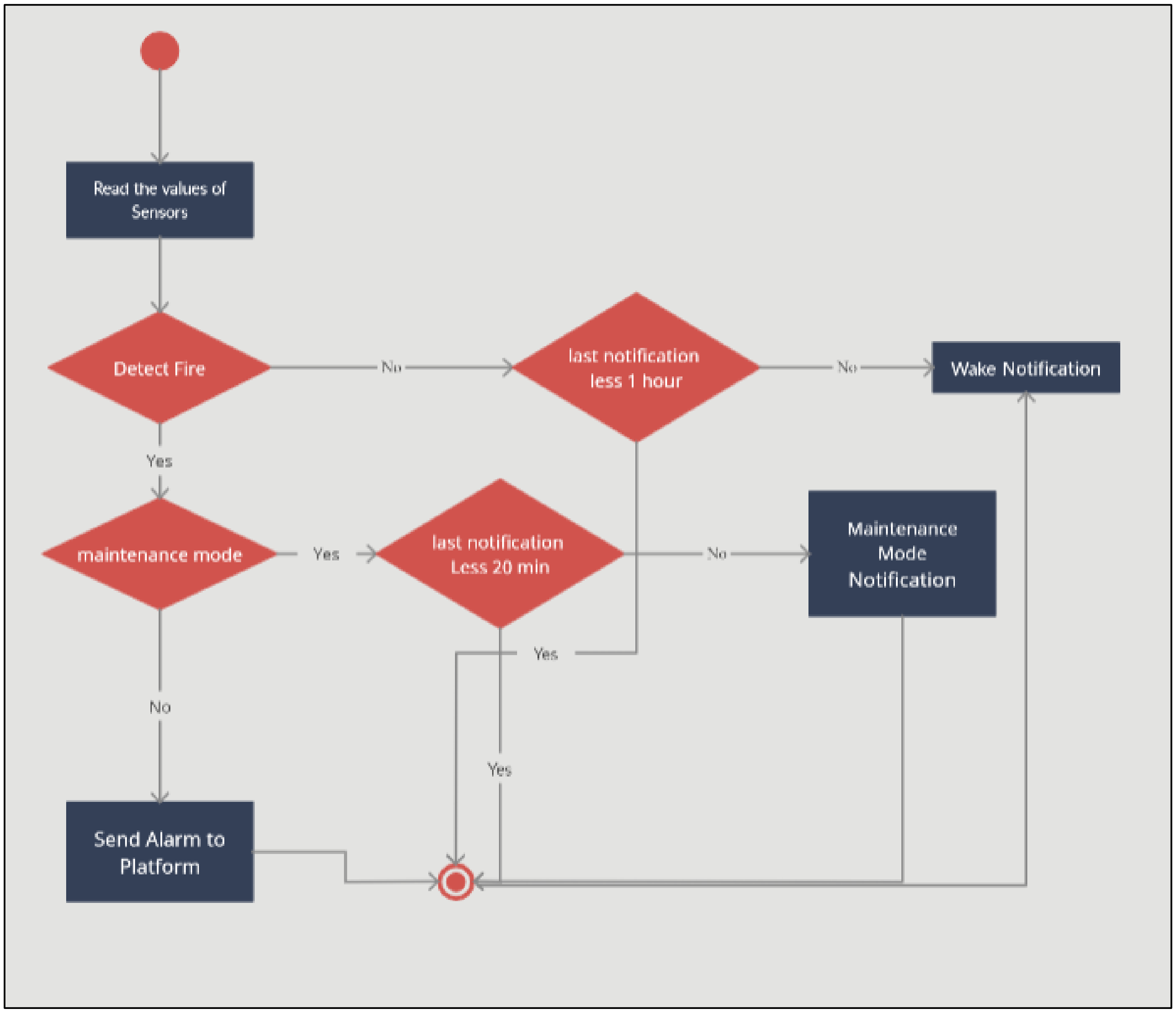

Read the values of Sensors
Detect Fire
No
last notification less 1 hour
No
Wake Notification
Yes
maintenance mode
Yes
last notification Less 20 min
No
Maintenance Mode Notification
Yes
No
Yes
Send Alarm to Platform

Figure 2: Activity Diagram for Fire Detection System

3. Database Design

3.1 Data Models

There will be no database for this system. (N/A)

3.2 Database Schema

Locally the data shall be stored in a SD Card. Remotely it shall be stored on SQL Database.

3.3 Database Entity Relationship Diagram

To be determined.

4. User Interface Design

4.1 User Interface Description

The system will not have a user interface.
Only notification through cloud platform

4.2 Wireframes

There will be no wireframes for this system.

5. Security Design

5.1 Security Requirements

The system will be designed to be secure and protect data during transmission. Wi-Fi encryption will be used to protect communication between the microcontroller and cloud platform.

5.2 Authentication and Authorization Design

There will be no authentication or authorization for this system.

5.3 Encryption Design

Wi-Fi encryption will be used to encrypt communication between the microcontroller and cloud platform.

6. Performance Design:

6.1 Performance Requirements:

The system shall have a response time of less than 7 seconds to detect and alert for fire incidents.

The system shall have a minimum uptime of 99.9%.
The system shall not exceed 50% of the available bandwidth.
The system shall be able to handle up to 1000 requests per hour.

6.2 Load Testing Plan:

Conduct load testing on the system to simulate peak traffic usage.
Analyze the results of the load testing and make any necessary optimizations to improve the system's performance.

6.3 Performance Optimization Techniques:

Try to reduce latency and increase the system's availability.
Implement caching techniques to reduce server response time.
Use compression to reduce the amount of data transferred over the network.

7. Error Handling and Logging Design:

7.1 Error Handling Plan:

Use error handling techniques to handle exceptions and errors that may occur in the system.
Provide descriptive error messages to the user to help them understand the error and how to fix it.
Log all errors and exceptions for future analysis and troubleshooting.

7.2 Logging Plan:

Log all system events, including sensor readings, alarms, notifications, and maintenance mode events.
Use a log aggregation tool to consolidate and analyze logs from multiple sources.
Set up alerts to notify administrators of critical errors or system failures.

8. Testing Design:

8.1 Testing Requirements:

Conduct unit testing to test individual components of the system.
Conduct integration testing to test the interaction between the components of the system.
Conduct system testing to test the system as a whole.
Conduct acceptance testing to ensure that the system meets the requirements of the stakeholders.

8.2 Testing Plan:

Develop test cases for each type of testing.
Execute the test cases and record the results.
Fix any issues identified during testing and repeat the testing process until all issues are resolved.

8.3 Test Cases:

Unit testing: Test sensor readings, alarm triggers, and notification sending functions.
Integration testing: Test the interaction between the sensor readings and alarm triggers, and between the alarm triggers and notification sending functions.
System testing: Test the system as a whole, including sensor readings, alarm triggers, notification sending functions, and maintenance mode events.
Acceptance testing: Test the system against the requirements of the stakeholders.

9. Deployment Design:

9.1 Deployment Plan:

Deploy the system to the cloud platform.
Use containerization techniques to manage the system's components.
Configure the system to be scalable and fault-tolerant.

9.2 Rollback Plan:

Implement a rollback plan in case of deployment failure.
Use a version control system to manage code changes and rollbacks.

9.3 Disaster Recovery Plan:

Implement a disaster recovery plan in case of system failure.
Use backup and restoration techniques to recover from system failures.

10. Maintenance and Support Design:

10.1 Maintenance Plan:

Conduct regular maintenance on the system to ensure its proper functioning.
Update the system components as necessary to keep up with security and performance standards.

10.2 Support Plan:

Provide user support through a ticketing system or other communication channels.

Maintain documentation to help users troubleshoot and resolve issues on their own.
Provide training for system administrators and users to ensure they are able to use the system effectively.