

INDUSTRY - SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

PROJECT REPORT

| | |
|----------------|------------------|
| TEAM ID | PNT2022TMID41386 |
| FACULTY MENTOR | NITHYA D |

TEAM MEMBERS:

BHARATHI K

AYISHA A

ANITHASRI M

DIVYA M

| Title | Page No |
|--|-----------|
| 1. INTRODUCTION..... | 3 |
| 1.1 Project Overview | 3 |
| 1.2 Purpose..... | 3 |
| 2. LITERATURE SURVEY | 3 |
| 2.1 Existing Problem..... | 3 |
| 2.2 References..... | 3 |
| 2.3 Problem Statement Definition..... | 4 |
| 3. IDEATION & PROPOSED SOLUTION..... | 4 |
| 3.1 Empathy Map Canvas..... | 4 |
| 3.2 Ideation & Brainstorming | 5 |
| 3.3 Proposed Solution..... | 8 |
| 3.4 Problem Solution Fit..... | 9 |
| 4. REQUIREMENT ANALYSIS..... | 9 |
| 4.1 Functional Requirement..... | 10 |
| 4.2 Non - Functional Requirement..... | 10 |
| 5. PROJECT DESIGN..... | 11 |
| 5.1 Data Flow Diagram..... | 11 |
| 5.2 Solution & Technical Requirements..... | 12 |
| 5.3 User Stories..... | 13 |
| 6. PROJECT PLANNING & SCHEDULING..... | 15 |
| 6.1 Sprint Planning & Estimation..... | 15 |
| 6.2 Sprint Delivery Schedule..... | 15 |
| 6.3 Report from JIRA..... | 16 |
| 7. CODING & SOLUTIONING..... | 18 |
| 7.1 Feature 1..... | 18 |
| 7.2 Feature 2..... | 20 |
| 7.3 Feature 3..... | 21 |
| 8. TESTING..... | 23 |
| 8.1 Test Cases..... | 23 |
| 8.2 User Acceptance Testing..... | 24 |
| 9. RESULTS..... | 24 |
| 9.1 Performance Metrics..... | 24 |
| 10. ADVANTAGES & DISADVANTAGES..... | 25 |
| 11. CONCLUSION..... | 27 |
| 12. FUTURE SCOPE..... | 27 |
| 13. APPENDIX..... | 29 |
| Source Code..... | 28 |
| Github & Project Demo Link..... | 40 |

1.INTRODUCTION

1.1 Project Overview

The smart fire management system includes a gas, flame, and temperature sensor to detect any environmental changes. The exhaust fans are turned on based on the temperature readings and the presence of any gases. If a flame is detected, the sprinklers will automatically activate. Emergency alerts are sent to the authorities and the Fire Station.

1.2 Purpose

- > To provide a detect the status of the room using IoT devices
- > To turn on sprinkler and exhaust fan when there is an accident
- > To detect the flow of water
- > To send and store the temperature status in a cloud storage
- > To provide an easy management system on dashboard
- > To provide an overview of what is happening to the user

2.LITERATURE SURVEY

2.1 Existing Problem

The situation is not ideal because fire management systems in homes and industries are not very reliable, efficient, or cost-effective, and lack advanced processing and features such as an automatic alert system for administrators and authorities. They are using older fire safety systems that cannot even activate the sprinkler system and do not communicate with one another properly to prevent false alarms. They also monitor the entire system using applications.

2.2 Reference

<https://pdfs.semanticscholar.org/f3e7/a7c0cf2d448be592421045033506e845e6c2.pdf>

<https://www.mdpi.com/2224-2708/7/1/11>

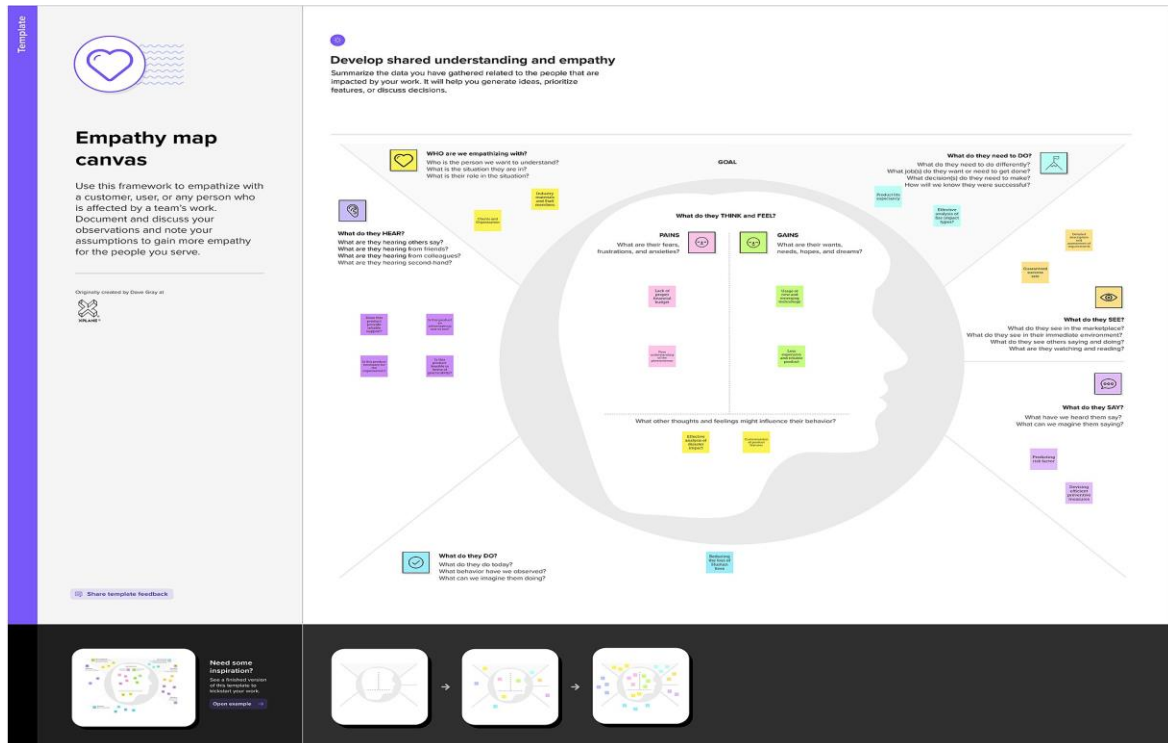
2.3 Problem Statement Definiton

The fire management system in houses and industries is not very reliable, efficient, cost effective, and does not have any advanced processing and does not have any features like automatic alert system for admin and authorities and in many buildings there are using older fire safety system that cannot even activate the sprinkler system and all of them do not communicate with each other properly to prevent false alarms.

3.IDEATHON AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

- > An empathy map is a simple, easy-to-understand visual that captures knowledge about a user's behaviors and attitudes.
- > It is a useful tool for assisting teams in better understanding their users.
- > Creating an effective solution necessitates understanding the true problem and the person experiencing it.
- > The map-making exercise helps participants consider things from the user's perspective, including his or her goals and challenges.



3.2 Ideation and Brainstorming

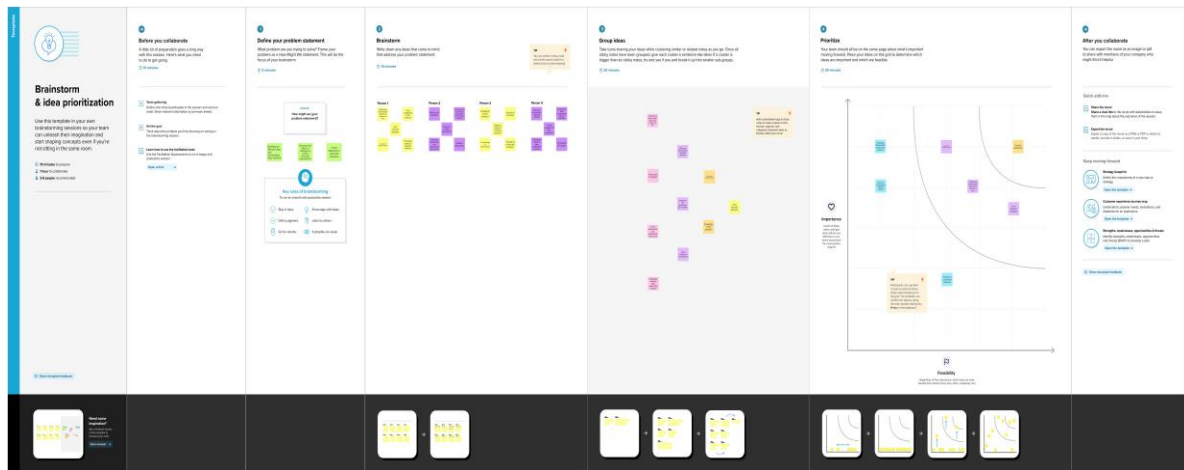
Step 1: Team Gathering, Collaboration and Select the Problem Statement

Team was gathered in mural app for collaboration

The team members are

- Bharathi K
- Ayisha A
- Anithasri M
- Divya M

Step 2: Brainstorm, Idea Listing and Grouping



Step3: Idea Prioritization

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes





3.3 Proposed Solution

| S No | Parameter | Description |
|------|--|---|
| 1. | Problem Statement (Problem to be solved) | To improve the safety management system in industries. Improving the safety management system against the fire incidents in industries. |
| 2. | Idea / Solution description | To implement the fire safety management in industry based on IOT using Arduino uno board with fire detection and fire extinguisher system. And using some sensors (Humidity sensor, Flame sensor, smoke sensor) with GPS tracking system. |
| 3. | Novelty / Uniqueness | An integrated system of temperature monitoring, gas monitoring, fire detection automatically fire extinguisher with accuracy of information about locations and response through SMS notification and call. |
| 4. | Social Impact / Customer Satisfaction | It early prevents the accident cost by fire in industries. Nearby locations so maximum extend more accurate reliability Compatibility design integrated system |
| 5. | Business Model (Revenue Model) | This product can be utilized by an industry. This can be thought of as a productive and helpful item as industries great many current rescuing people and machine from the fire accident. |
| 6. | Scalability of the Solution | It is trying to execute this technique as we need to introduce an Arduino gadget which was modified with an Arduino that takes received signals from sensors. Easily operatable and can be maintained. Required low time for maintain. Cost is reasonable value |

3.4 Proposed Solution Fit

| Problem-Solution Fit canvas | | Purpose / Vision | Version: |
|---|--|--|--|
| Define CS, fit into CL | 1. CUSTOMER SEGMENT(S) CS Industry members as well as others | 6. CUSTOMER LIMITATIONS CL <small>EG. BUDGET, DEVICES</small> The customer should just click the alert message to enhance the further step to stop the fire. Proper network connection and available devices are needed. | 5. AVAILABLE SOLUTIONS AS <small>PLUSES & MINUSES</small> The customer used to call for the emergency number 101 to call the fire service team to stop the fire at that time of reporting many products in the industry gets damaged and many lives were death. Now with the use of our product the industry can sense the fire explosion and stop at the initial stage itself. So, it is quite much more easy. |
| | 2. PROBLEMS / PAINS + ITS FREQUENCY PR We are solving the problem of fire spread by automatically detecting the fire at the ignition stage and stop the fire spread easily using Artificial Intelligence and IOT based ideations. | 9. PROBLEM ROOT / CAUSE RC The fire causes a lot of damages in the industry. Usually when it gets fired in an industry the fire service team is called to stop the fire. But now our solution use can stop the fire without the help of fire service. | 7. BEHAVIOR + ITS INTENSITY BE At once the message is send to the customers mobile from the sensors controlled Intelligence the customer himself can give the access to stop the fire spread on the whole |
| Focus on PR, map into BE, understand RC | 3. TRIGGERS TO ACT TR We can ask our customer to get an experience about our product. We can insist they must need of our product. | 10. YOUR SOLUTION SL We can just access the message from the IOT devices combined with sensors to stop the fire spread at the ignition stage itself. It is much easier, safe to handle. | 8. CHANNELS of BEHAVIOR CH ONLINE Notifications send can be accessed. |
| | 4. EMOTIONS <small>BEFORE / AFTER</small> EM Before: Customer is not finding a proper rid for the fire spread problem. After: Now with the help of our product the customer can easily enhance the problem. | | OFFLINE The sensors with the help of intelligence can stop the fire spread at the initial stage itself |
| Identify strong TR & EM | | | Extract online & offline CH of BE |


 Problem-Solution fit canvas is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. Designed by Daria Nepriakhina / [ideahackers.nl](https://www.ideahackers.nl) - we tailor ideas to customer behaviour and increase solution adoption probability.


 KAPWING
 IDEAHACKERS.NL

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

A functional requirement defines a system or component's function, where a function is

- > Defined as a specification of behavior between inputs and outputs
- > It defines "what the software system should do"
- > Defined at the component level
- > Usually simple to define
- > Aids in testing the software's functionality

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
|--------|-------------------------------|--|
| FR-1 | User Registration | <ul style="list-style-type: none">• Registration through Form• Registration through mobile number |
| FR-2 | User Confirmation | <ul style="list-style-type: none">• Confirmation via message• Confirmation via call |
| FR-3 | User Login | Login through site or App using respective username and password |
| FR-4 | User Upload | Client ought to be able to upload the information |
| FR-5 | Fire Detection Monitoring | The sensors located will monitor the industry 24/7 and keeps updating the end user. |
| FR-6 | Location notification | Location of fire will be sent to the fire department through alarm or message |

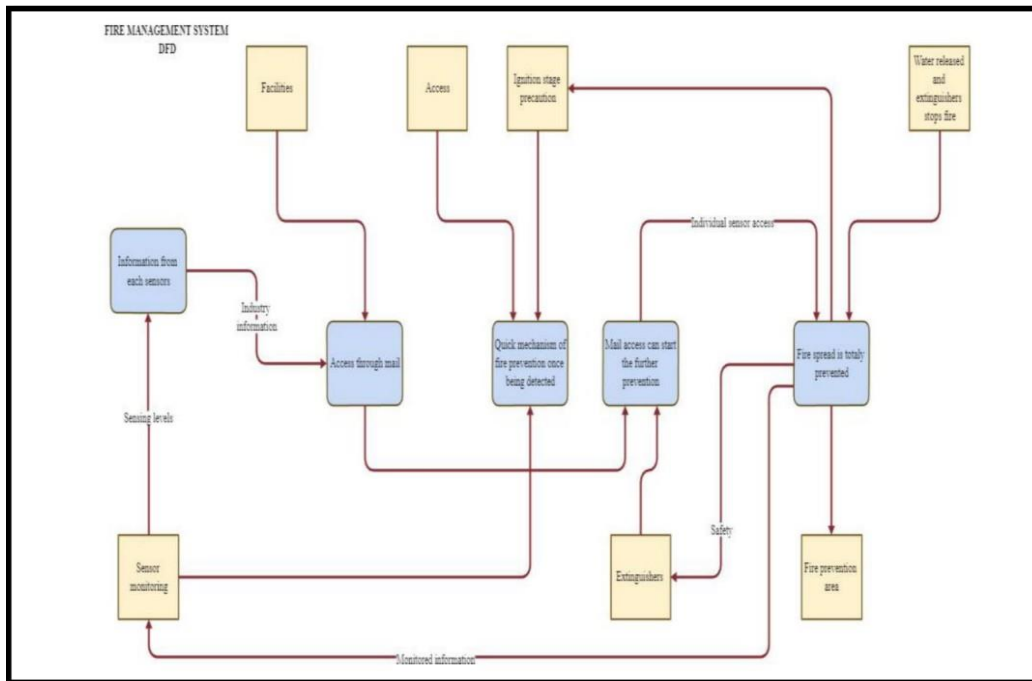
4.2 Non - Functional Requirements

- > A non-functional requirement defines a software system's quality attribute.
- > It limits "How should the software system fulfill the functional requirements?"
- > It is not required Applied to the entire system
- > Usually more difficult to define
- > Aids in the verification of software performance

| FR . No | Non - Functional Requirement | Description |
|---------|------------------------------|---|
| NFR-1 | Usability | <ul style="list-style-type: none"> ● It is the simple and Economic ● Easy to use |
| NFR-2 | Security | <ul style="list-style-type: none"> ● The software remains resilient in the face of attacks ● The Web application is highly secured |
| NFR-3 | Reliability | <ul style="list-style-type: none"> ● Response timer will be faster ● It has high Reliability ● The application runs accurately |
| NFR-4 | Performance | If Fire detected it will be immediately notified through the web application, and it also maintain track periodically. |
| NFR-5 | Availability | We will be Monitoring the Industry by day and Night (24/7). In case of Fire detected we willbe intimating the management rapidly. |

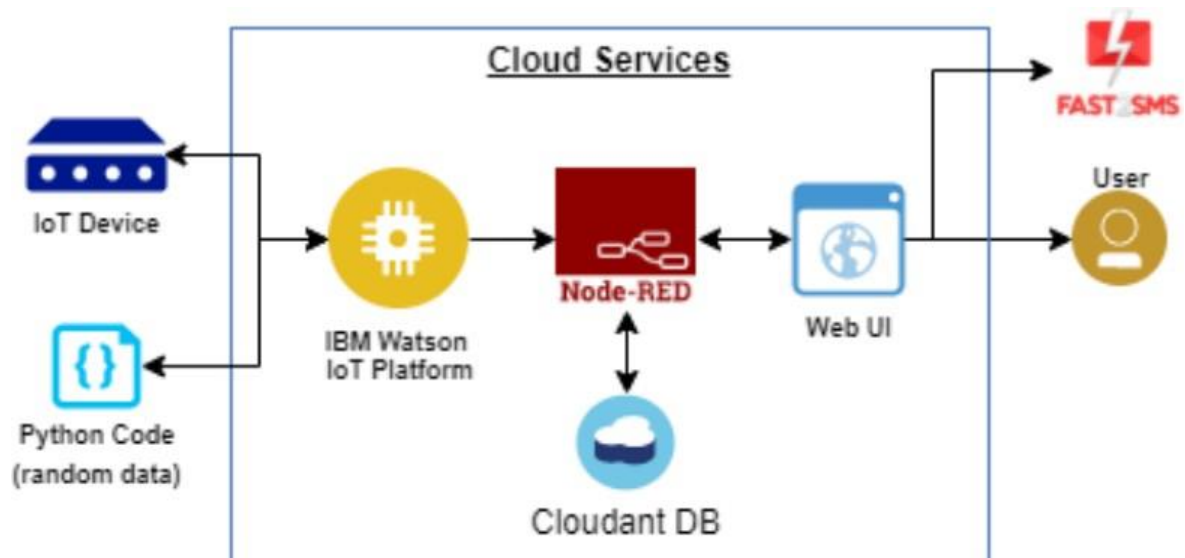
5.PROJECT DESIGN

5.1 Data flow Diagram

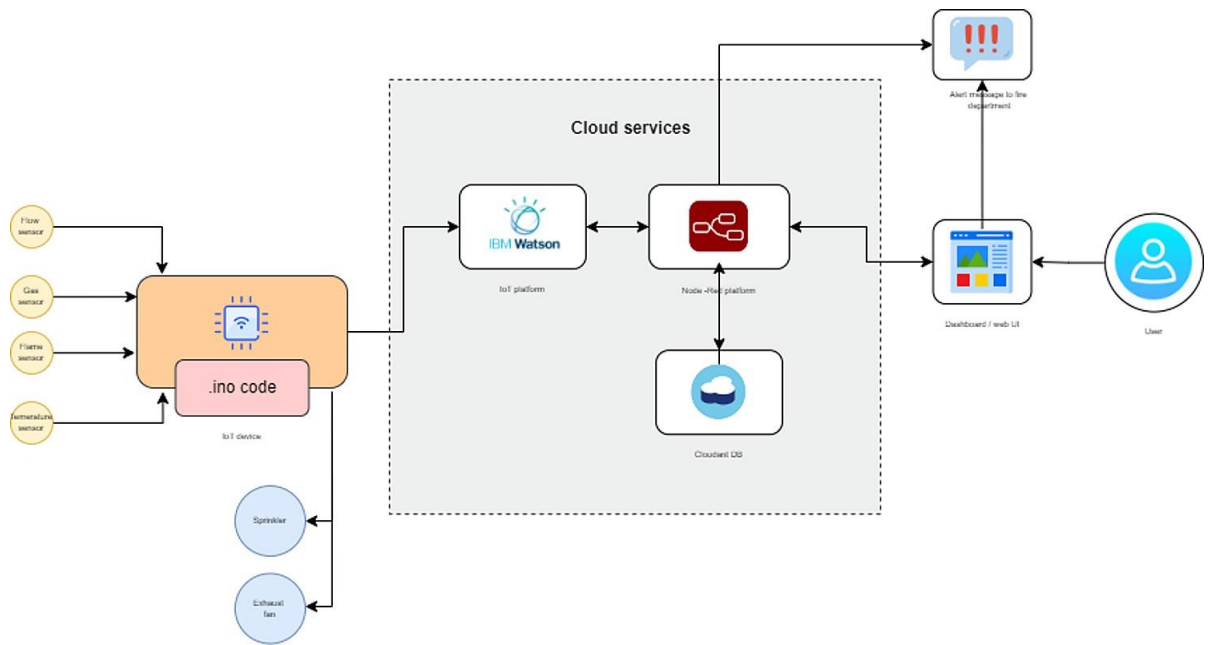


5.2 Solution and Technical Architecture

Solution Architecture



Technical Architecture



5.3 User Stories

| User Type | Functional Requirement | User Story Number | User Story/Task | Acceptance Criteria | Priority | Release |
|---|------------------------|-------------------|---|---|----------|------------|
| Customers (Mobile user, web user, care executive, Administrator) | Registration | USN - 1 | As a user, I can register for the application by entering my mail, password, and confirming my password | I can access my account/ dashboard | High | Sprint - 1 |
| | | USN - 2 | As a user, I will receive confirmation email once I have registered for the application | I can receive confirmation email & click confirm | High | Sprint - 1 |
| | Dashboard | USN - 3 | As a user, I can register for the application through internet | I can register & access the dashboard with Internet login | Low | Sprint - 2 |
| | | USN - 4 | As a user, I can register for the application through Gmail | I can confirm the registration in Gmail | Medium | Sprint - 1 |
| | Login | USN - 5 | As a user, I can log into the application by entering email & password | I can login with my ID & password | High | Sprint - 1 |

6.PROJECT DESIGN AND PLANNING

6.1 Sprint Planning and Estimation

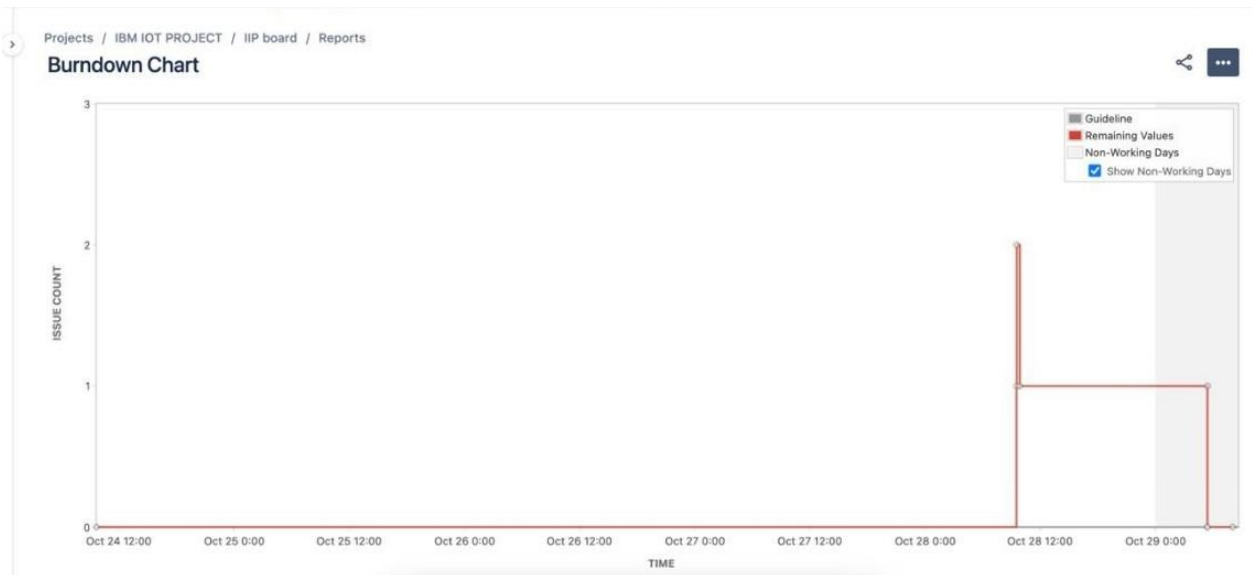
6.2 Sprint Delivery Schedule

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority | Team Members |
|----------|-------------------------------|-------------------|---|--------------|----------|--------------|
| Sprint-1 | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Bharathi K |
| Sprint-1 | | USN-2 | As a user, I will receive confirmation email once I have registered for the application | 1 | High | Anithasri M |
| Sprint-2 | | USN-3 | As a user, I can register for the application through Facebook | 2 | Low | Divya M |
| Sprint-1 | | USN-4 | As a user, I can register for the application through Gmail | 2 | Medium | Ayisha A |
| Sprint-1 | Login | USN-5 | As a user, I can log into the application by entering email & password | 1 | High | Bharathi K |

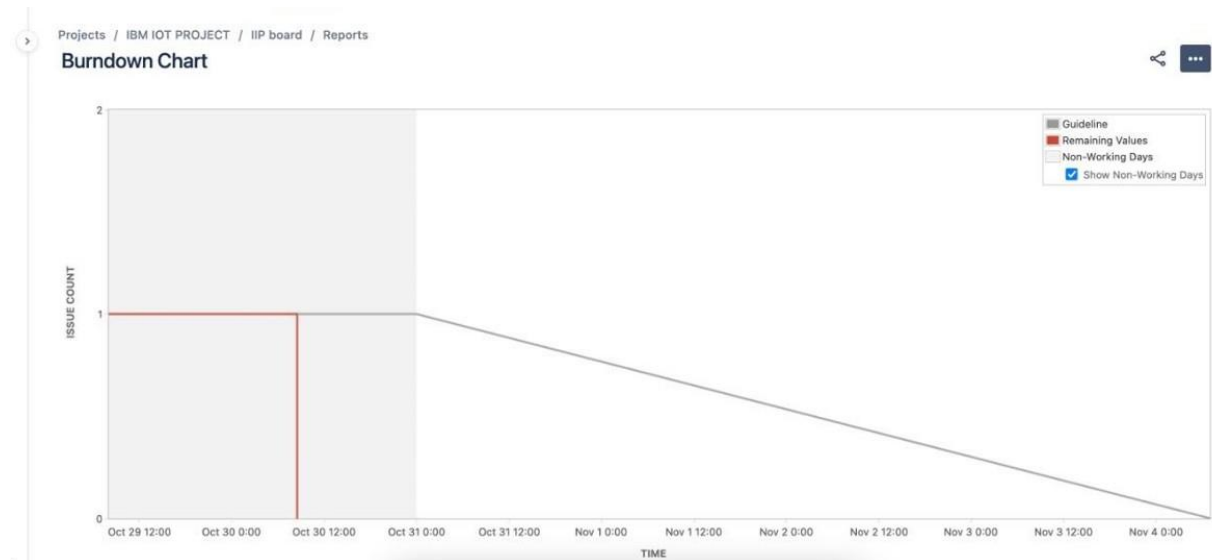
| | | | | | | |
|----------|-------------------|---------|--|---|--------|-------------|
| Sprint-1 | User Interface | USN-6 | As a user, I should not need any pre requisites to handle the UI | 1 | Medium | Bharathi K |
| Sprint-1 | Dashboard | WUSN-1 | As a web user, able to access the inputs from the sensors | 2 | High | Anithasri M |
| Sprint-1 | View Manner | CCE-1 | As a customer care, Data visualization must be in good understandable view. | 2 | High | Divya M |
| Sprint-1 | Taste | CCE-2 | As a customer care, I can able to monitor the place affected due to fire. | 1 | High | Bharathi K |
| Sprint-1 | Colour Visibility | CCE-3 | As a customer care, I should know the time it started. | 1 | High | Ayisha A |
| Sprint-2 | Risk Tolerant | ADMIN-1 | Administrator should handle the system, server and take care of the application. | 1 | High | Bharathi K |

6.3 Reports from JIRA

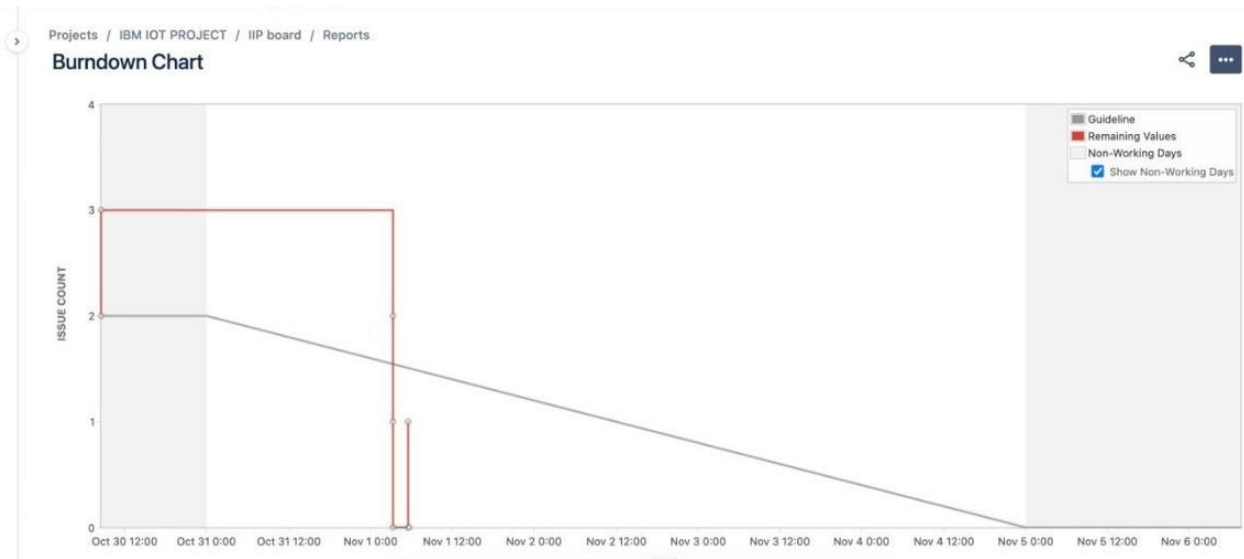
Sprint 1



Sprint 2



Sprint 3



Sprint 4



7.CODING & SOLUTIONING

Feature 1: False Alarm Checking

```
if(temp < 45 )
{
if(flame > 650 )
{
accidentstatus = "Need Auditing";
if(canfanoperate)
isfanon = true;
else
isfanon = false;
issprinkon = false;
}

else if(flame <= 10)
{

accidentstatus = "nothing happened";
isfanon = false;
issprinkon = false;
}
}
else if(temp >= 45 && temp <= 55 )
{
if(flame <=650 && flame >100 )
{
if(cansprinkoperate)
issprinkon = true;
else
issprinkon = false;
accidentstatus = "moderate";
if(gas > 160 && canfanoperate )
{
isfanon = true;
}
else{
isfanon = false;
}
}
else if(flame <= 100 && flame > 10)
{
```

```

    if(cansprinkoperate)
    issprinkon = true;
    else
    issprinkon = false;
    isfanon = false;
    accidentstatus = "moderate";
}
}
else if(temp > 55){ if(flame > 650){
gas = 500 + rand()%500;
accidentstatus = "severe";
if(cansprinkoperate)
issprinkon = true; else
issprinkon = false; if(canfanoperate)
isfanon = true; else
isfanon = false;
}
}
else if(flame < 650 && flame > 400 )
{
gas = 300 + rand()%500;
accidentstatus = "severe";
if(cansprinkoperate)
issprinkon = true; else
issprinkon = false;
if(canfanoperate)
isfanon = true;
else
isfanon = false;
}
}
else {
accidentstatus = "Need moderate Auditing";
isfanon = false;
issprinkon = false;
}
if(issprinkon){ if(flow)
{
sprinkstatus = "working";
}
else
{
sprinkstatus = "not working";
}
}

```

```

}
else if(!issprinkon)
{
sprinkstatus = "ready";
}
else
{
sprinkstatus = "something's wrong";
}

```

Explanation

- > This set of code checks the false alarms and sets the current status
- > It also handles the permission management of whether a device will work or not

Feature 2

```

void PublishData(float temp, int gas ,int flame ,int flow,bool
isfanon,bool issprinkon)
{

```

```

mqttconnect();

```

```

String payload = "{\"temp\":";
payload += temp;
payload += "," "\"gas\":";
payload += gas;
payload += "," "\"flame\":";
payload += flame;
payload += "," "\"flow\":";

```

```

payload += ((flow)?"true":"false");

```

```

payload += "," "\"isfanon\":";

```

```

payload += ((isfanon)?"true":"false");
payload += "," "\"issprinkon\":";
payload += ((issprinkon)?"true":"false");
payload += "," "\"cansentalert\":";
payload += ((cansentalert)?"true":"false");
payload += "," "\"accidentstatus\":";
payload += "\"" + accidentstatus + "\"";

```

```

payload += ", \"\sprinkstatus\":";
payload += "\""+sprinkstatus+"\"";
payload += "}";

if (client.publish(publishTopic, (char*) payload.c_str()))
{
  Serial.println("Publish ok");// if it sucessfully upload data on the
}
else
{
  Serial.println("Publish failed");
}
}

```

Explanation

> It sends the data to IBM Watson Platform

Feature 3

```

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)

{

  Serial.print("callback invoked for topic: ");
  Serial.println(subscribetopic);
  for (int i = 0; i < payloadLength; i++)
  {
    data3 += (char)payload[i];
  }

  Serial.println("data: "+ data3);

  const char *s =(char*) data3.c_str();
  double pincode = 0;
  if(mjson_get_number(s, strlen(s), "$.pin", &pincode)){
    if(((int)pincode)==137153){ const char *buf;
    int len;

```

```

if (mjson_find(s, strlen(s), "$.command", &buf, &len))

{

String command(buf,len);
if(command=="cantfan"){
canfanoperate = !canfanoperate;

}

else if(command=="cantsprink"){ cansprinkoperate = !cansprinkoperate;
}else if(command=="sentalert"){ resetcooldown();
}
}
}
data3="";
}

```

Explanation

- > The user's action is received as a command and stored in a buffer
- > The event in the device is performed in accordance with the command
- > It searches for a secret encrypted pin to perform that event

8.TESTING

| Test case ID | Feature Type | Component | Test Scenario | Pre-Requsite | Steps To Execute | Test Data | Expected Result | Actual Result | Status | Comments | TC for Automation(Y/N) | BUG ID | Executed By |
|--------------|--------------|---------------------|---|---|---|--|---|---------------------|--------|----------|------------------------|--------|-------------|
| Sensor_001 | Functional | Microcontroller | Sensor data is properly taken | The connections to the circuit | 1.Open the simulator in wokwi. | Random values generated | Get the values and print it in the | Working as | Pass | | N | | Akshaya |
| Sensor_002 | Functional | Microcontroller | Sensor data is parsed as json | The microcontroller should | 1.Open the simulator in wokwi. | Random values generated | Get the values and print it in the | Working as | Pass | | N | | Karthick |
| Work_001 | Functional | Microcontroller | To check for fake alarm | The sensor values are taken | 1.Simulate the device(do a practical | Random values generated | Accident status is properly updated | Working as | Pass | | N | | Ajin |
| Work_002 | Functional | Microcontroller and | The data should be sent to IBM | The device setup is completed | 1.Start the simulation in wokwi. | Random values generated | The values are shown in recent | Working as | Pass | | N | | Akshaya |
| Work_003 | Functional | Node-red | The data should be sent to | The necessary packages | 1.Login to node red editor | values got from the iot | The debug area should show the | Working as | Pass | | N | | Yoonus |
| Work_004 | Functional | Node-red | Verify that the json data is parsed | A configured node-red with | 1.Login to node red editor | values got from the iot | the debug menu shows the output | Working as | Pass | | N | | Yoonus |
| Database_001 | Storage | Cloudant | The received data is stored in database in a key value pair | The node red is connected with cloudant node | 1.login to cloudant dashboard. 2.create new database. 3. connect the database with node red and then give the database name in required field | values got from the iot device | After sending the data the data is stored in cloudant | Working as expected | Pass | | N | | Karthick |
| SMS_001 | API | sms API | The sms is sent when there is fire alert | The node red should be configured to send a post request | 1.Simulate the fire in the simulator(if real hardware is used real fire is used). 2.or click the sent alert button in | "Fire alert at xyz industries Hurry" And the trigger inputs | sms receiving to the given phonenum | Working as expected | Pass | | N | | Ajin |
| Work_005 | Functional | UI | Even at times of emergency sometimes manual control is required | the dashboard interaction elements is connected to the node red | 1. in the dashboard enter the correct pin 2.click the action to be done | The action by user | manual command system works only | Working as expected | Pass | | N | | yoonus |
| Auth_001 | Functional | UI | Verify that the correct pin is entered | text filed is given in dashboard to enter pin | 1.The correct pin is entered 2.then necessary action is required | 1234 | command is sent successfull | working as expected | Pass | | N | | Akshaya |
| Auth_002 | Functional | UI | Verify that it handles when wrong pin is entered | text filed is given in dashboard to enter pin | 1.The correct pin is entered 2.then necessary action is required | 141324 63363 1 001 fds | Show a message that the entered pin is wrong | Working as expected | Pass | | N | | Karthick |
| SMS_002 | Functional | Microcontroller | Verify that the message is not sent continuously when there is fire It sends a message then waits for 10 minutes even after that if the fire exists it sends again | the sms functionality should be implemented | 1.Simulate a fire accident scenario 2.or click the send alert button on the dashboard 3.wait for the message to be sent | the event is simulated or triggered | The service should not spam continuous messages to authorities as fire won't be down within fraction of seconds | Working as expected | Pass | | N | | Ajin |

8.2UAT

Defect Analysis

| Resolution | Severity 1 | Severity 2 | Severity 3 | Severity 4 | Subtotal |
|----------------|------------|------------|------------|------------|----------|
| By Design | 9 | 0 | 2 | 1 | 12 |
| External | 0 | 0 | 1 | 0 | 1 |
| Fixed | 19 | 24 | 25 | 14 | 82 |
| Not Reproduced | 0 | 0 | 2 | 0 | 2 |
| Skipped | 0 | 0 | 0 | 0 | 0 |
| Won't Fix | 0 | 0 | 0 | 0 | 0 |
| Totals | 28 | 24 | 30 | 15 | 97 |

Test Case Analysis

| Section | Total Cases | Not Tested | Fail | Pass |
|---------------------|-------------|------------|------|------|
| Client Application | 4 | 0 | 0 | 4 |
| Security | 2 | 0 | 0 | 2 |
| Exception Reporting | 11 | 0 | 0 | 11 |
| Final Report Output | 5 | 0 | 0 | 5 |

9.RESULTS

9.1 Performance Metrics

CPU Usage:

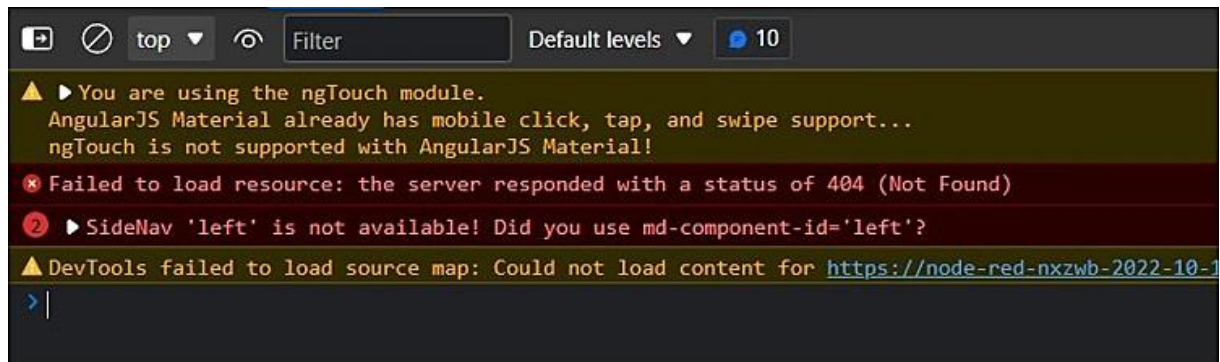
The micro version of C++ makes the most efficient use of the CPU. The program runs in $O(1)$ time for each loop, ignoring the network and communication. To improve communication with MQTT, the program sleeps every 1 second. Because the program runs in $O(1)$ time and the compiler optimizes it during compilation, there is less CPU load per cycle. The following instructions are stored on the stack memory and can be popped after execution.

Memory Usage:

The sensor values and networking data are saved in the ESP32's sram. It's a lot of information because the ESP32 only has 520 KB of memory. To save memory and ensure optimal program execution, the exact addresses for each memory cycle are overwritten with new values.

Error Rates:

The error rates are very low because the backend and dashboard are handled with node-red. Exceptions are handled properly so that the system's usability is not affected.

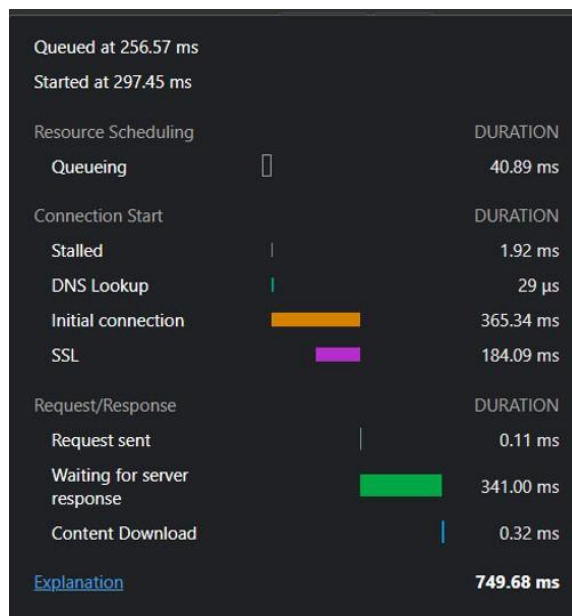


Latency and Respose Time:

The DOM handling of the received data is optimal and latency is low .After the DOM is loaded the entire site is loaded to the browser.

19 requests 10.1 kB transferred 2.2 MB resources Finish: 2.53 s DOMContentLoaded: 1.21 s Load: 1.31 s

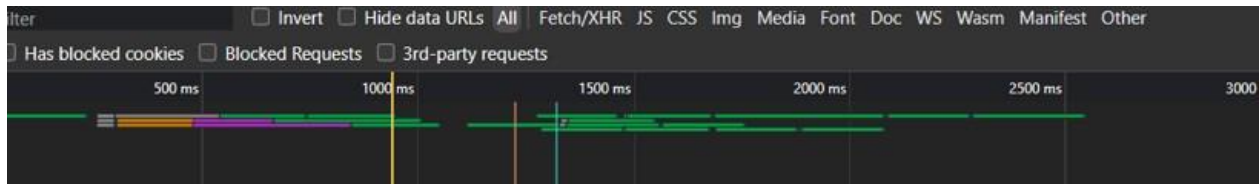
In addition, the server responds quickly. The average response time is acceptable.



For the data sent from the IoT Device (considering the sleep of one second from the IoT), the response is much faster. We can see the delay caused by the sleep function.

The average time is well over optimal value

$$\begin{aligned}\text{Average Time} &= (5\text{ms} + 2600\text{ms})/2 \\ &= 1302.5\end{aligned}$$



Garbage Collection:

The Node framework handles garbage collection on the server side. C++ does not have garbage collection features in IoT devices. However, in this case, it is not necessary because the memory will be used again to store the data. There is no allocation of any dangling pointers or poorly handled address space.

10. ADVANTAGES & DISADVANTAGES

Advantages:

- > Active detection of gas leaks and fire outbreaks
- > SMS alerting of administrators and fire authorities
- > Turning on/off sprinklers and exhaust fans automatically
- > To manually turn on/off sprinklers and exhaust fans, as well as send SMS alerts, authentication is required
- > It detects false fire outbreaks automatically, reducing unnecessary panic
- > We can confirm that the sprinkler system is functioning properly by using flow sensors
- > A dashboard can display the status of any device
- > The dashboard can be viewed by users via a web application

- > The dashboard can be viewed by users via a web application

Disadvantages:

- > Always require an internet connection [only to send the SMS alert]
- > If the physical device fails, the entire operation fails
- > Because a large amount of data is stored in the cloud database every second, a large database is required

11.CONCLUSION

So we conclude that, our problem premise is solved using IoT devices by developing a smart management system that solves many inherent problems in traditional fire management systems, such as actively monitoring for fire breakouts and gas leakage and sending SMS alerts to administrators and fire authorities.

12.FUTURE SCOPE

The existing devices can be modified to work in various specialized environments, as well as scaled to house use to large labs [Because fire accidents can cause significant loss of human lives in homes to large industries], as well as used in public places and vehicles.

13.APPENDIX

ESP32 - Microcontroller:

The ESP32 is a low-cost, low-power system-on-a-chip microcontroller family with integrated Wi-Fi and dual-mode Bluetooth.

- Memory: 320 KiB SRAM

- CPU: Tensilica Xtensa LX6 Microprocessor @ 160 or 240 MHz
- Power: 3.3 VDC
- Manufacturer: Espressif Systems
- Predecessor: ESP8266

Sensors:

DHT22 - Temperature & Humidity Sensor:

The DHT22 is a simple and inexpensive digital temperature and humidity sensor. It measures the surrounding air with a capacitive humidity sensor and a thermistor and outputs a digital signal on the data pin (no analog input pins needed).

Flow Sensors:

A flow sensor (also known as a "flow meter") is an electronic device that measures or controls the flow rate of liquids and gases through pipes and tubes.

MQ5 - Gas Sensor:

Gas sensors (also referred to as gas detectors) are electronic devices that detect and identify various types of gasses. They are frequently used to detect toxic or explosive gases as well as to measure gas concentration.

Flame Sensor:

A flame-sensor is a type of detector that is intended to detect and respond to the occurrence of a fire or flame. The response to flame detection can be affected by its fitting.

Source Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include "DHT.h"// Library for dht11
#include <cstdlib>
#include <time.h>
#include <mjson.h>
#define DHTPIN 15    // what pin we're connected to
#define DHTTYPE DHT22    // define type of sensor DHT 11
DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and typr of dht connected

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "fvs923"

#define DEVICE_TYPE "zenabc"
#define DEVICE_ID "221"
#define TOKEN "12345678"

String data3 = "";

String accidentstatus = "";
String sprinkstatus = "";
float temp =0;
bool isfanon = false;

bool issprinkon = false;
bool cansprinkoperate = true;
bool canfanoperate = true;
```

```

bool cansentalert = false;
int gas = 0;
int flame = 0;
int flow = 0;
long int cooldown= 600;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char subscribetopic[] = "iot-2/cmd/command/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

//-----

WiFiClient wifiClient; // creating the instance for wificlient

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing
parameter like server id,portand wificredential

void setup()// configureing the ESP32

{

Serial.begin(115200); dht.begin();
//if real gas sensor is used make sure the senor is heated up for acurate readings

/*

- Here random values for readings and stdout were used to show the

working of the devices as physical or simulated devices are not available.

```

```

*/ delay(10);
Serial.println();
wificonnect();
mqttconnect();
}

void loop()

{

temp = dht.readTemperature();

//setting a random seed (only for random values not in real life scenarios)

srand(time(0));

//initial variable activities like declaring , assigning gas = rand()%400;

int flamereading = rand()%1024;

flame = map(flamereading,0,1024,0,1024);
int flow = ((rand()%100)>50?1:0); //find the accident status 'cause fake alert may be caused by some mischief
activities

if(temp < 45 ){ if(flame > 650 ){
accidentstatus = "Need Auditing";
if(canfanoperate)
isfanon = true; else
isfanon = false;
issprinkon = false;
}

else if(flame <= 10){

```

```
accidentstatus = "nothing happened";  
isfanon = false;  
issprinkon = false;
```

```
}
```

```
}else if(temp >= 45 && temp <= 55 )  
{  
  if(flame <=650 && flame >100 )  
  {
```

```
    if(cansprinkoperate)
```

```
      issprinkon = true; else  
      issprinkon = false;  
      accidentstatus = "moderate";  
      if(gas > 160 && canfanoperate )  
      {  
        isfanon = true;
```

```
      }
```

```
    else{
```

```
      isfanon = false;
```

```
    }
```

```
  }
```

```
  else if(flame <= 100 && flame > 10)  
  {  
    if(cansprinkoperate)
```



```

issprinkon = true; else
issprinkon = false;
isfanon = false;
accidentstatus = "moderate";
}
}
else if(temp > 55)
{
    if(flame > 650)
    {
        gas = 500 + rand()%500;

        accidentstatus = "severe";
        if(cansprinkoperate)
        issprinkon = true; else
        issprinkon = false;
        if(canfanoperate)
        isfanon = true; else
        isfanon = false;

    }

}

else if(flame < 650 && flame > 400 )
{

    gas = 300 + rand()%500;
    accidentstatus = "severe";
    if(cansprinkoperate)
    issprinkon = true; else
    issprinkon = false;
    if(canfanoperate)
    isfanon = true;
    else
    isfanon = false;

```

```
}

}

else {

    accidentstatus = "Need moderate Auditing";
    isfanon = false;
    issprinkon = false;

}

if(issprinkon){ if(flow)
{
    sprinkstatus = "working";

}

else{

    sprinkstatus = "not working";

}

}

else if(!issprinkon)
{
    sprinkstatus = "ready";
}

else {

    sprinkstatus = "something's wrong";
```

```
}
```

```
PublishData(temp,gas,flame,flow,isfanon,issprinkon);
```

```
//a cooldown period is set as the values and situations are random in real life sceanarios the time can be  
reduced or neclected
```

```
if(accidentstatus=="severe" && cooldown >= 600)
```

```
{
```

```
    cooldown = 0;
```

```
    sendalert();
```

```
PublishData(temp,gas,flame,flow,isfanon,issprinkon);
```

```
    cansentalert = false;
```

```
}
```

```
if(cooldown > 999999)
```

```
{
```

```
    cooldown = 601;
```

```
}
```

```
delay(1000);
```

```
++cooldown;
```

```
if (!client.loop())
```

```
{
```

```
    mqttconnect();
```

```
}
```

```
}
```

```

/*.....retrieving to
Cloud. */

void PublishData(float temp, int gas ,int flame ,int flow,bool isfanon,bool issprinkon) {

mqttconnect(); //function call for connecting to ibm

/*

creating the String in in form JSon to update the data to ibm cloud

*/

String payload = "{\"temp\":";
payload += temp;
payload += ", \"gas\":";
payload += gas;
payload += ", \"flame\":";
payload += flame;
payload += ", \"flow\":";
payload += ((flow)?"true":"false");
payload += ", \"isfanon\":";
payload += ((isfanon)?"true":"false");
payload += ", \"issprinkon\":";
payload += ((issprinkon)?"true":"false");
payload += ", \"cansentalert\":";
payload += ((cansentalert)?"true":"false");
payload += ", \"accidentstatus\":";
payload += "\"" + accidentstatus + "\"";
payload += ", \"sprinkstatus\":";
payload += "\"" + sprinkstatus + "\"";
payload += "}";
if (client.publish(publishTopic, (char*) payload.c_str())) {

```

Serial.println("Publish ok"); // if it successfully upload data on the cloud then it will print publish ok in Serial monitor or else it will print publish failed

} else {

Serial.println("Publish failed");

}

}

void mqttconnect() {

if (!client.connected())

{

Serial.print("Reconnecting client to ");

Serial.println(server);

while (!client.connect(clientId, authMethod, token))

{

Serial.print(".");

delay(500);

}

initManagedDevice();

Serial.println();

}

}

void wificonnect() //function definition for wificonnect

{

```

Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED)
{
  delay(100);

  Serial.print(".");

}

Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

void initManagedDevice() {

  if (client.subscribe(subscribetopic))
  {
    Serial.println((subscribetopic));
    Serial.println("subscribe to cmd OK");
  } else {

    Serial.println("subscribe to cmd FAILED");

  }

}

//handles commands from user side

```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
```

```
{
```

```
Serial.print("callback invoked for topic: ");
```

```
Serial.println(subscribetopic);
```

```
for (int i = 0; i < payloadLength; i++)
```

```
{
```

```
data3 += (char)payload[i];
```

```
}
```

```
Serial.println("data: "+ data3);
```

```
const char *s =(char*) data3.c_str();
```

```
double pincode = 0;
```

```
if(mjson_get_number(s, strlen(s), "$.pin", &pincode))
```

```
{
```

```
if(((int)pincode)==137153)
```

```
{
```

```
const char *buf; int len;
```

```
if (mjson_find(s, strlen(s), "$.command", &buf, &len)) // And print it
```

```
{
```

```
String command(buf,len);
```

```
if(command=="cantfan"){
```

```
//this works when there is gas sensor reads high value and if there should be a
```

```
//manual trigger else it will be automate canfanoperate = !canfanoperate;
```

```
}
```

```
else if(command=="cantsprink")
```

```
{
```

```
cansprinkoperate = !cansprinkoperate;
```

```
}else if(command=="sentalert"){
```

```
//this works when there is accident status is severe and if there should be a
```

```
//manual trigger else it will be automate resetcooldown();
```

```
}
```

```
}
```

```
}
```

```
}
```

```
data3="";
```

```
}
```

```
void resetcooldown()
```

```
{
```

```
cooldown = 0;
```

```
}
```

```
//sent alert request to node-red void sendalert(){
```

```
cansentalert = true; cooldown = 0;
```

```
}
```

Github Link: <https://github.com/IBM-EPBL/IBM-Project-45026-1660727900>

Demo Video: <https://youtu.be/TevNXy1SyFg>