

Industry-specific intelligent fire management system

KONGU ENGINEERING COLLEGE , PERUNDURAI

Project Report

Swathisri E

Team ID: PNT2022TMID04725

Vibin T

Subhash S

Swathi 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.....	3
3. IDEATION & PROPOSED SOLUTION.....	4
3.1 Empathy Map Canvas	4
3.2 Ideation & Brainstorming	5
3.3 Proposed Solution	7
3.4 Problem Solution fit.....	8
4. REQUIREMENT ANALYSIS.....	9
4.1 Functional requirement	9
4.2 Non-Functional requirements	9
5. PROJECT DESIGN	10
5.1 Data Flow Diagrams	10
5.2 Solution & Technical Architecture.....	11
5.3 User Stories.....	12
6. PROJECT PLANNING & SCHEDULING	12
6.1 Sprint Planning & Estimation	12
6.2 Sprint Delivery Schedule	13
6.3 Reports from JIRA	13
7. CODING & SOLUTIONING.....	15
7.1 Feature 1	15
7.2 Feature 2	19
7.3 Feature 3	20
8. TESTING	22
8.1 Test Cases	22
8.2 User Acceptance Testing.....	22
9. RESULTS.....	23
9.1 Performance Metrics.....	23
10. ADVANTAGES & DISADVANTAGES	25
11. CONCLUSION.....	26
12. FUTURE SCOPE	26
13. APPENDIX	27
Source Code	28
GitHub & Project Demo Link	40

1. Introduction

1.1 Project Overview

A gas, flame, and temperature sensor is part of the smart fire management system to track any alterations in the environment. The exhaust fans are turned ON based on the temperature data and whether any gases are present. The sprinklers will be turned on automatically if any flame is found. The Fire station and the authorities are informed of emergency notifications.

1.2 Purpose

- To give an easy management system on dashboard
- To give a user an overview of what's happening
- To give a detection of the status of the room using IOT devices
- To turn on sprinklers and exhaust fans when there is an accident
- To detect the flow of water
- To send and store the temperature status in a cloud storage
- To send a text message to the authorities when there is a fire accident

2. Literature survey

2.1 Existing Problem

The lack of a dependable, effective, cost-effective, modern processing, or feature-rich fire management system in many buildings, as well as the fact that it lacks an automatic alarm system for administrators and authorities, make the situation less than ideal. In order to reduce false alarms, they are deploying outdated fire safety equipment that cannot even trigger the sprinkler system. Applications are also being used to monitor the entire system.

2.2 Reference

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

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

2.3 Problem Statement Definition

The fire management systems in homes and businesses are not very dependable, efficient, or affordable, and they lack features like an automatic alert system for administrators and authorities. Many buildings still use outdated fire safety systems that can't even activate the sprinkler system, and they all improperly communicate with one another to prevent false alarms. They also use applications to monitor the entire system.

3. Ideation and Proposed solution

3.1 Empathy map canvas

- An empathy map is a straightforward, simple-to-understand picture that records information about a user's actions and attitudes.
- It is a helpful tool for teams to use to understand their users better. It is necessary to comprehend both the actual issue and the individual who is experiencing it in order to develop a workable solution.
- Participants learn to think about issues from the user's perspective, including goals and obstacles, through the process of building the map.

Build empathy and keep your focus on the user by putting yourself in their shoes.



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

- Swathisri
- Vibin
- Subhash
- Swathi

step 2:Brainstorm, Idea Listing and Grouping

3

Group ideas

Take turns sharing your ideas while clustering similar ones. In the last 10 minutes, give each cluster a sentence-like title. If you have more than six sticky notes, try and see if you can break it up.

20 minutes



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

SWATHI SRI



VIBIN T



SUBASH S



SWATHI M



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

Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<p>A major fire broke out in the huge cracker manufacturing hub in Sivakasi. Many people lost their lives in this accident.</p> <p>The chemicals used for manufacturing fireworks are highly sensitive to friction, impact, heat and static electricity.</p> <p>These friction and impact causes fire accidents. The fire accidents or blasts occur either due to sparks from electrical fittings or during loading and unloading of boxes containing fireworks.</p> <p>These fire accidents cause great loss to the industry and also to the lives of the people working in the industries.</p> <p>So, to overcome this problem we have proposed a solution that uses sensors to detect the fire before it causes damage, sprinklers are used to control the fire and a fire alarm is used to alert the workers about the fire breakage.</p>

		This can also be used in all the other industries like textile industries, mining industries etc..
2.	Idea / Solution description	<p>In the proposed model, a gas sensor, flame sensor and temperature sensors are used for the detection of fire.</p> <p>Gas Sensor</p> <p>Gas sensors (also known as gas detectors) are electronic devices that detect and identify different types of gasses. They are commonly used to detect toxic or explosive gasses and measure gas concentration. Gas sensors are employed in factories and manufacturing facilities to identify gas leaks, and to detect smoke and carbon monoxide in homes. Gas sensors vary widely in size (portable and fixed), range, and sensing ability. They are often part of a large Embedded systems, such as hazmat and security systems, and they are normally connected to an audible alarm or interface. Because gas sensors are constantly interacting with air and other gasses, they have to be calibrated more often than many other types of sensors.</p> <p>In general gas sensors have the potential to detect all fires because every fire is emitting gas and an according fire detector is not dependent from the release of heat or smoke.</p> <p>Flame sensor</p> <p>The flame sensor detects the presence of fire or flame based on the Infrared (IR) wavelength emitted by the flame. It gives logic 1 as output if a flame is detected, otherwise, it gives logic 0 as output. Arduino Uno checks the logic level on the output pin of the sensor and performs</p>

3.4 Problem solution fit

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) Who is your customer? My customers are those who own organization and industries which can be easily subjected into fire accident. Examples: oil Industries, cotton industries	6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. Fire extinguishers hanging on the walls are weight and the industries or organization has less number of extinguishers so many of them could not use it when occurrence of fire.	5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? what have they tried in the past? In most places, fire extinguishers are hung on the walls for safety purposes, when there is an occurrence of fire, extinguishers are make used it by using man power and in some places ,an automatic fire detection system is also there.	Explore AS, differentiate

Focus on J&P, tap into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides If fire occur in any organization or industries this fire alarm detects the fire and indicates through the alarm and sprinkles the water to extinguish the fire.	9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. In crackers industries there used chemicals for crackers which are easily inflammable so it can cause fire accident even if a tiny sparks occurs as it happened once in a sivakasi.. Likewise many other problems can also be the root cause of fire	7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related find the right solar panel installer, calculate usage and benefit; indirectly associated: customers spend free time on volunteering(i.e. Green peace) Monitoring the fire with the help of the automatic fire detection system in the respective places to avoid fire accidents ,avoid using old machines and safety precautions are to be followed.	Focus on J&P, tap into BE, understand RC

Identify strong TR	3. TRIGGERS What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. If the industries has any easily flammable materials,any past bad fire accident then the automatic fire detection system could be installed. If any gas smell is felt and indication of fire ,peoples begin to act using extinguishers or water	10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. Industry specific-Intelligent fire management System .Different types of sensors like gas sensor, Temperature sensor and Flame sensor are used to detect any changes in the environment, based on the readings or detection of gas exhaust fans are switched on and if there is an occurrence of fire, sprinklers	8. CHANNELS of BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 Creating fire awareness programs Installation of smoke detection system 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. Outline clear pathways to exit doors Using flame retardant materials in interiors Use tools and machines properly	Identify strong TR
	4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your			

4. Requirement analysis

4.1 Functional Requirements

- A functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between inputs and outputs.
- It specifies “what should the software system do?”
- Defined at a component level
- Usually easy to define
- Helps you verify the functionality of the software

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Resource discovery	Searching devices and services of interest to the entity.
FR-4	Resource Management	Planning and assigning the technology to a program.
FR-5	Code Management	Integrating sensors to web UI
FR-6	Event Management	Merging data to users.

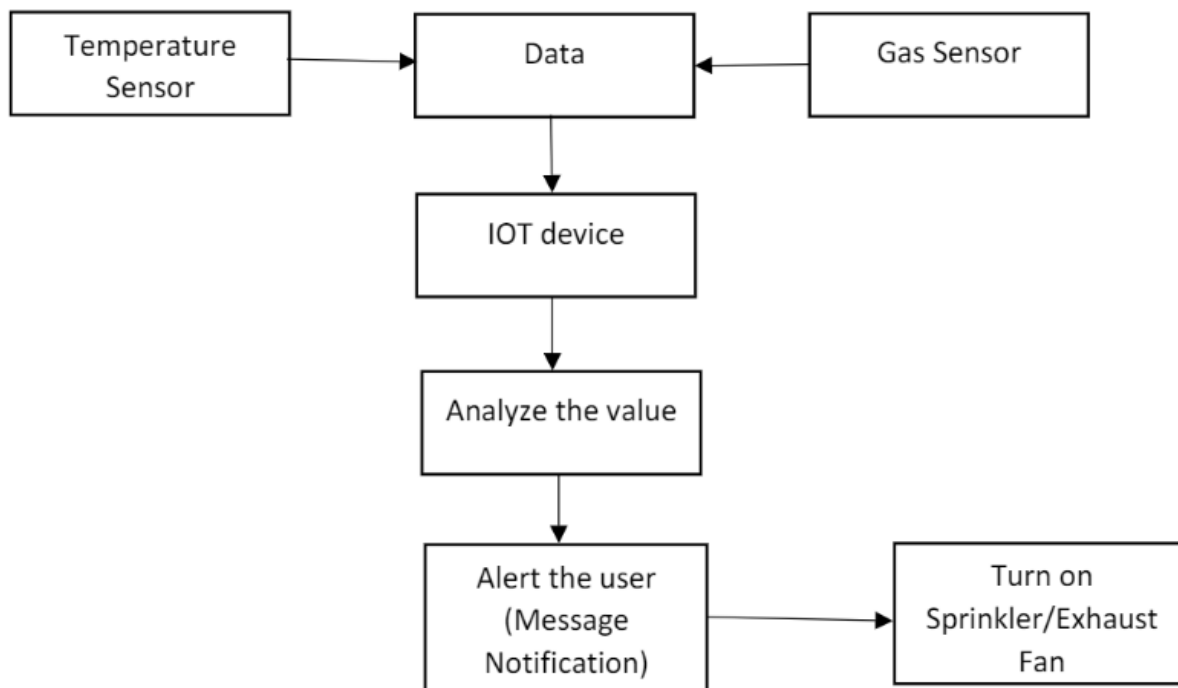
4.2 Non functional Requirements:

- A non-functional requirement defines the quality attribute of a software system
- It places constraint on “How should the software system fulfil the functional requirements?”
- It is not mandatory
- Applied to system as a whole
- Usually more difficult to define
- Helps you verify the performance of the software

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	This application will carry the process effectively and safely.
NFR-2	Security	The data is maintained with high level security against unauthorised access.
NFR-3	Reliability	This system is less reliable in case when the sensors become obsolete.
NFR-4	Performance	The system responds rapidly if it detects gas.
NFR-5	Availability	All-time the detection system is readily to detect the smoke
NFR-6	Scalability	The response time is less when it is compared with other systems.

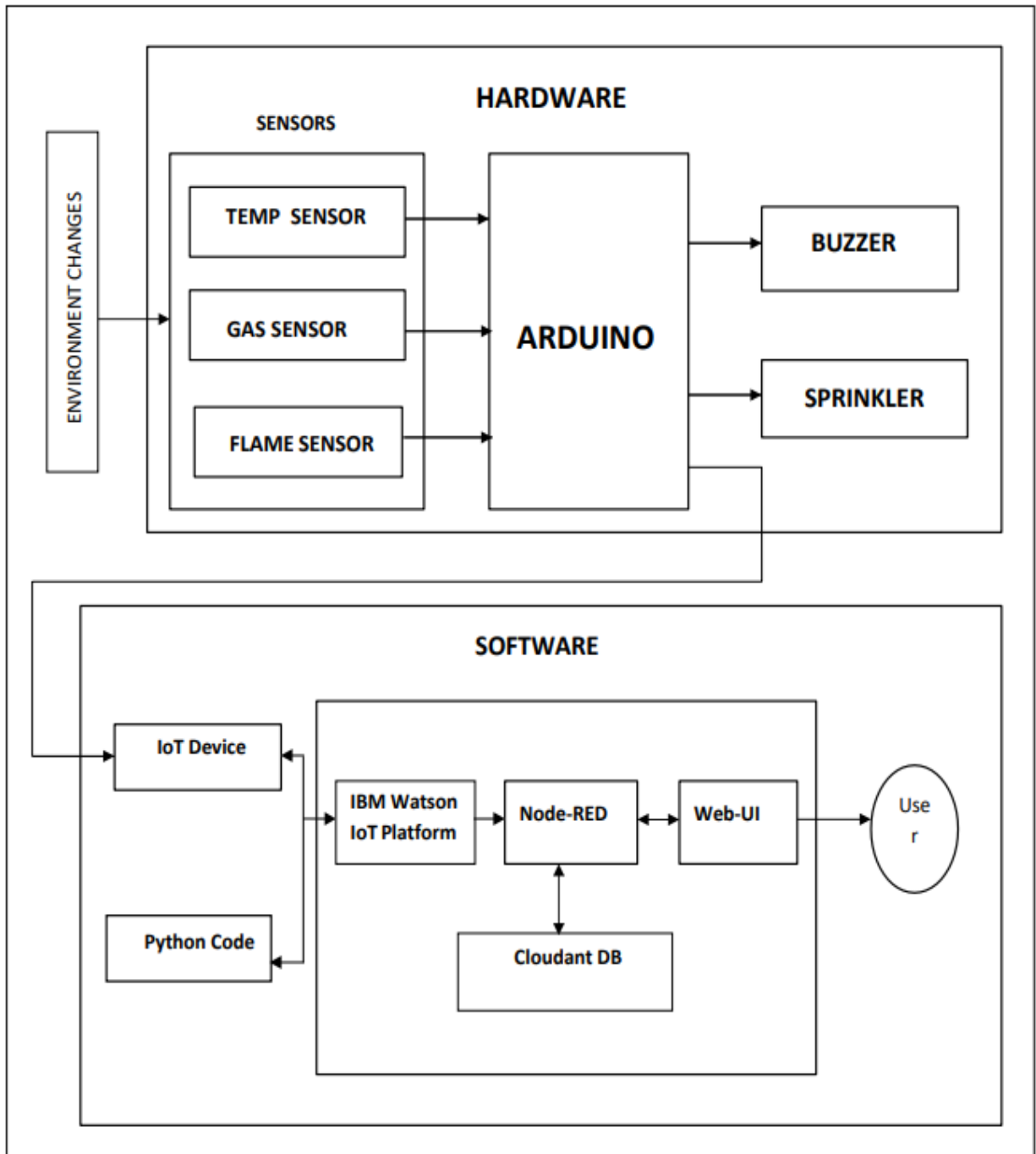
5. Project Design

5.1 Dataflow Diagram

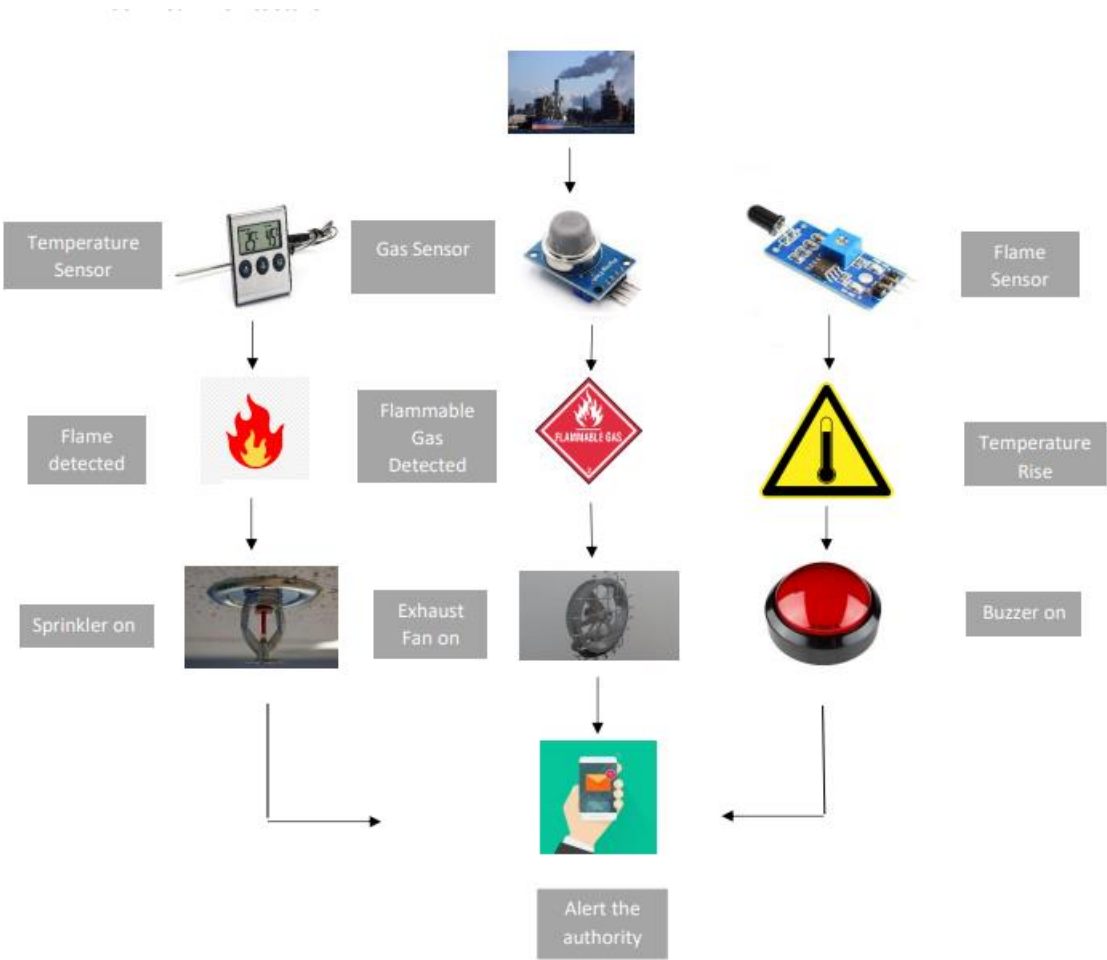


5.2 Solution and Technical architecture

Solution Architecture



Technical Architecture



5.3 User stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	User can register for the application by entering the username and Password	I will be able to access account	High	Sprint-1
	Login	USN-2	User can login to the application by entering the username and password	I can login to my account	High	Sprint-1
Sensing	Sensor	USN-3	Sensor senses the fire and the smoke	Indicated by sensor	High	Sprint-1
Extinguish	Actuators	USN-4	If the sensor detects the fire, sprinkler will be turned on.	Sprinkler will be turned on	High	Sprint-1
ata	Cloud	USN-5	All the values can be stored in the database.	Store the data	High	Sprint-1
Intimation	Siren	USN-6	If fire is detected, user can alert through the buzzer.	Evacuate	High	Sprint-1
Notification	Event Management	USN-7	Notification will be sent to the respective authority.	Notify	Medium	Sprint-1

6. Project design and planning

6.1 Sprint planning and estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Sensing	USN-1	Sensing the environment using the sensors.	3	High	Swathisri E
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in case of fire and gas leakage.	3	Medium	Vibin T
Sprint-2	Sending collected data to the IBM Watson	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	Subash S

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Registration	USN-4	Entering my email and password to verify authentication process.	3	High	Swathi M
Sprint-3	Storing of sensor data	USN-5	Storing in Cloud ant database.	2	Medium	Swathisri E
	Node red	USN-6	Sending the data from the IBM Watson to the Node red.	3	High	Vibin T
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Subash S

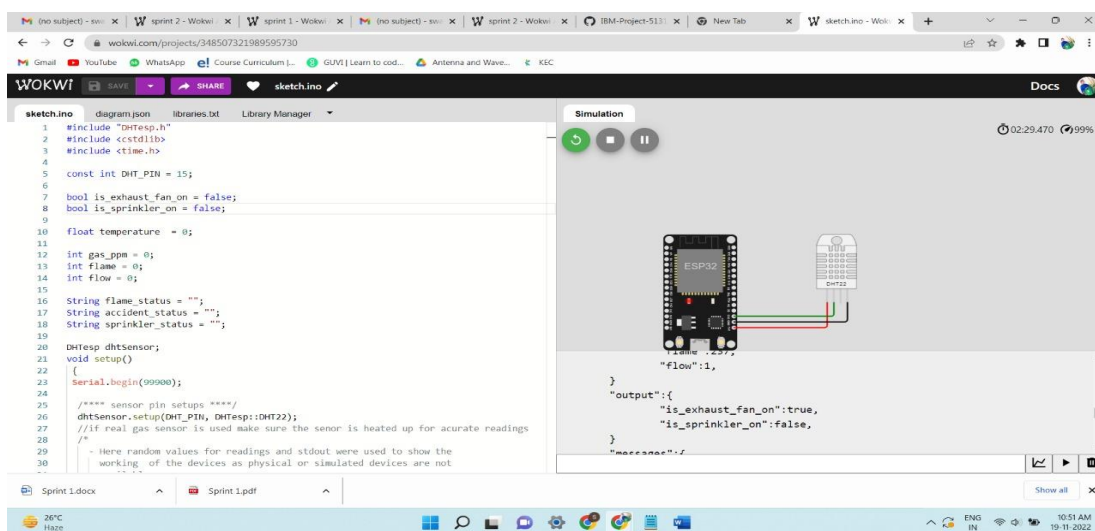
Sprint-4	Fast SMS Service	USN-8	Use Fast SMS to Send alert message once the parameters like temperature, flame and gas sensor readings goes beyond the threshold value	3	High	Swathi M
	Turn ON/OFF the actuators	USN-9	Users can turn off the Exhaust fan as well as the sprinkler system If needed in that Situation.	2	Medium	Swathisri E
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Testing	USN-10	Testing of project and Final Deliverables.	1	Low	Vibin T

6.2 Sprint delivery schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	6	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12 Nov 2022	6	12 Nov 2022
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-4	6	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

6.3 Reports from JIRA

Sprint 1



WOKWI SAVE SHARE Docs

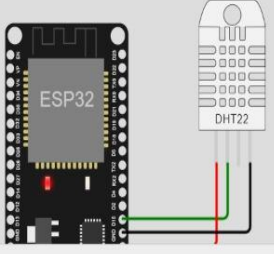
esp32-dht22.ino diagram.json libraries.txt Library Manager

```

1 #include "DHTesp.h"
2 #include <stdlib.h>
3 #include <time.h>
4
5 const int DHT_PIN = 15;
6
7 bool is_exhaust_fan_on = false;
8 bool is_sprinkler_on = false;
9
10 float temperature = 0;
11
12 int gas_ppm = 0;
13 int flame = 0;
14 int flow = 0;
15
16 String flame_status = "";
17 String accident_status = "";
18 String sprinkler_status = "";
19
20 DHTesp dhtSensor;
21
22
23 void setup() {
24   Serial.begin(99900);
25
26   /* sensor pin setups */
27   dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
28   //if real gas sensor is used make sure the sensor is heated up for accurate readings
29   /*
30   - Here random values for readings and stdout were used to show the

```

Simulation 00:27.430 105%



```

}
"messages":{
  "fire_status":No Fire,
  "flow_status":now it shouldn't,
  "accident_status":nil,
}
}

```

Sprint 2

WOKWI SAVE SHARE Docs

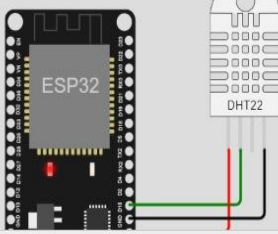
esp32-dht22.ino diagram.json libraries.txt Library Manager

```

1 #include "DHTesp.h"
2 #include <stdlib.h>
3 #include <time.h>
4 #include <WiFi.h>
5 #include <PubSubClient.h>
6
7
8 #define ORG "in8onk"
9 #define DEVICE_TYPE "abcd"
10 #define DEVICE_ID "12"
11 #define TOKEN "12345678"
12
13 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
14 char publishTopic[] = "iot-2/evt/data/fmt/json";
15 char authMethod[] = "use-token-auth";
16 char token[] = TOKEN;
17 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
18
19 WiFiClient wifiClient;
20 PubSubClient client(server, 1883, wifiClient);
21
22 const int DHT_PIN = 15;
23
24 bool is_exhaust_fan_on = false;
25 bool is_sprinkler_on = false;
26
27 float temperature = 0;
28
29 int gas_ppm = 0;
30 int flame = 0;

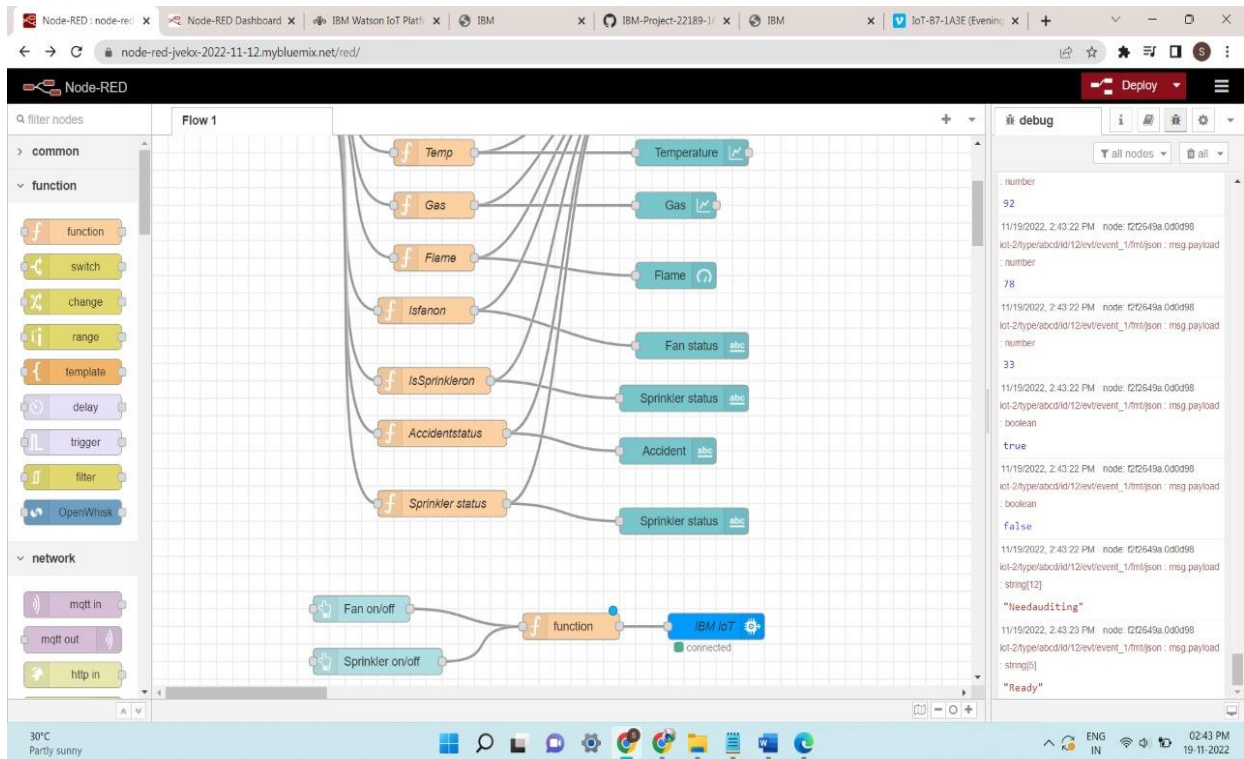
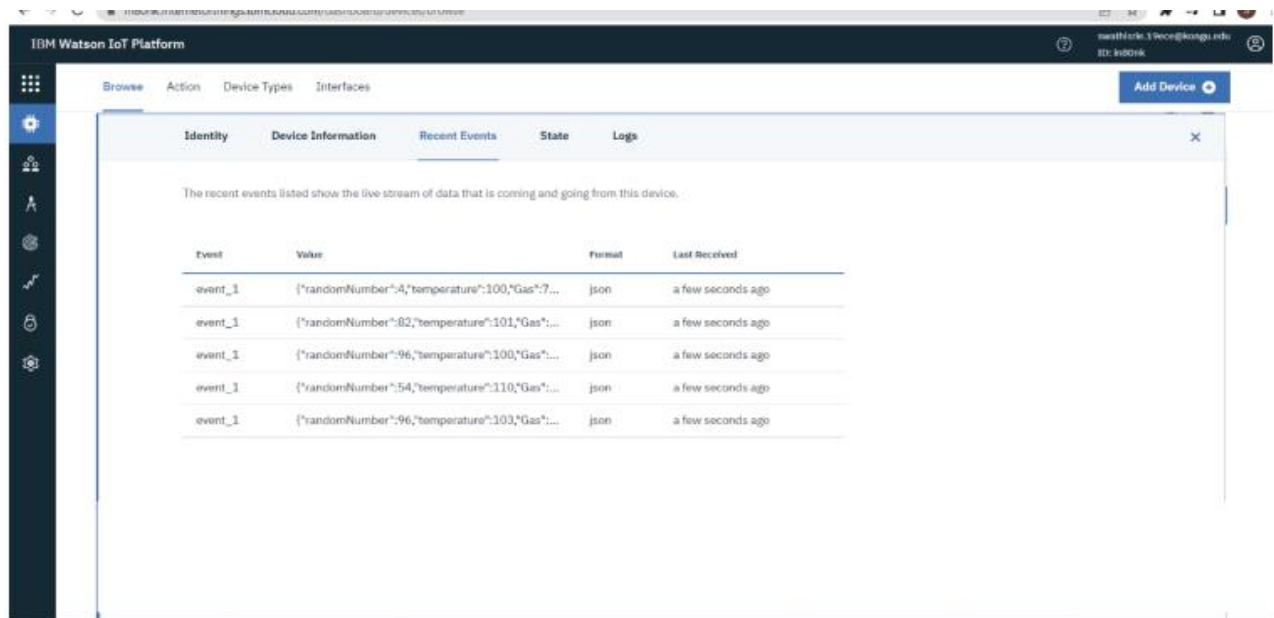
```

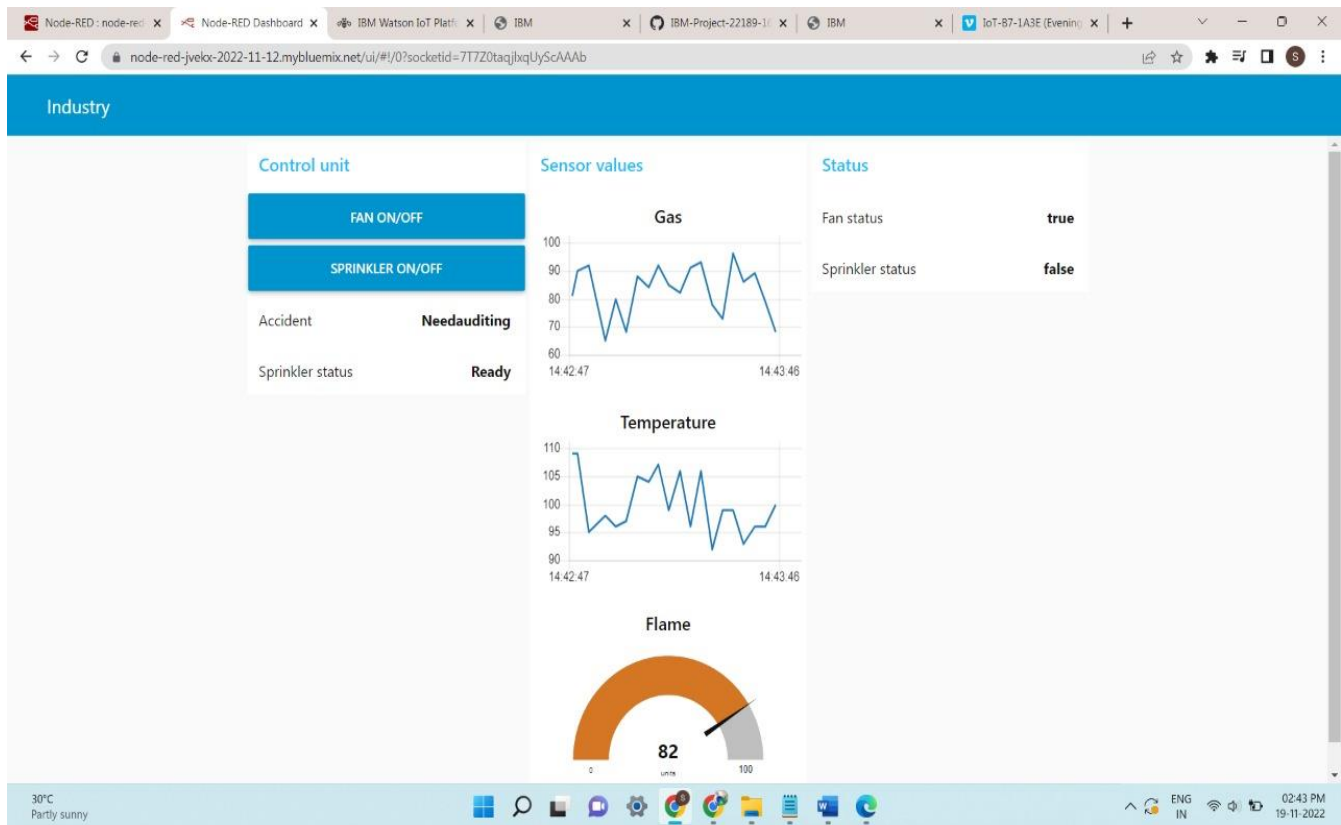
Simulation 01:39.158 102%



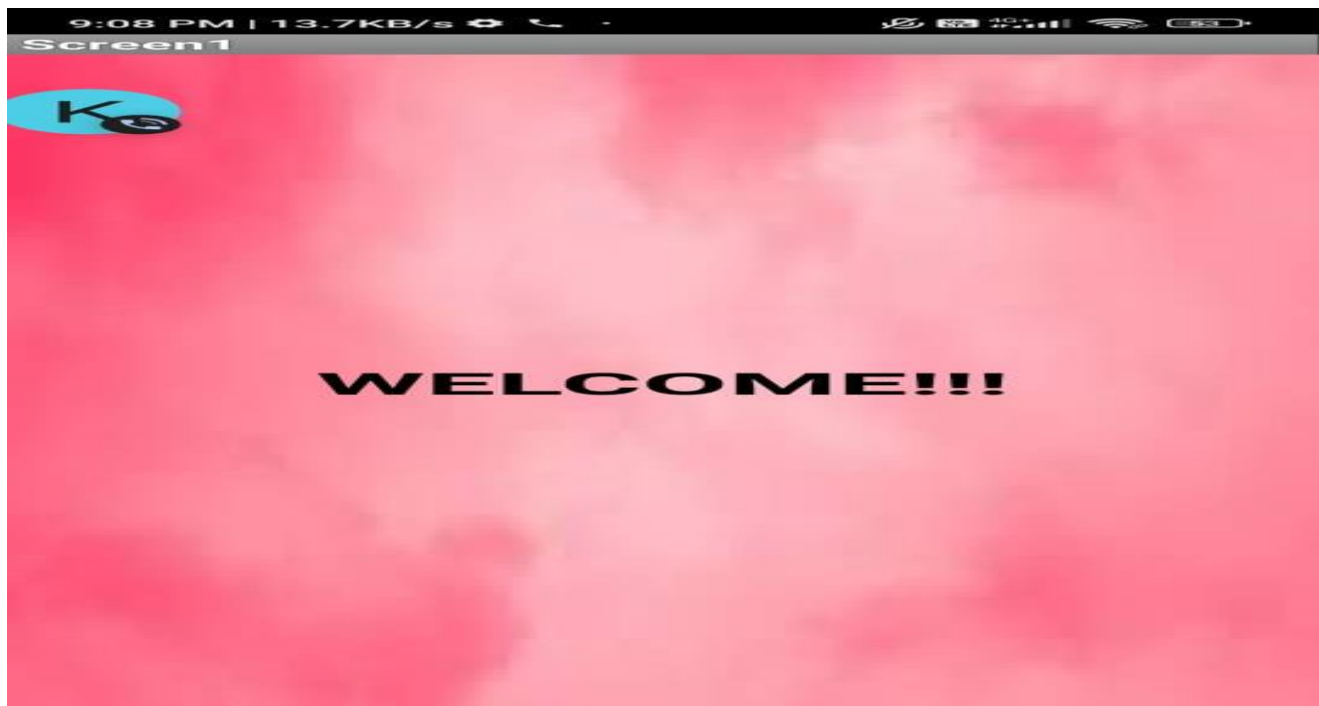
Publish OK
Publish OK
Publish OK
Publish OK
Publish OK
Publish OK

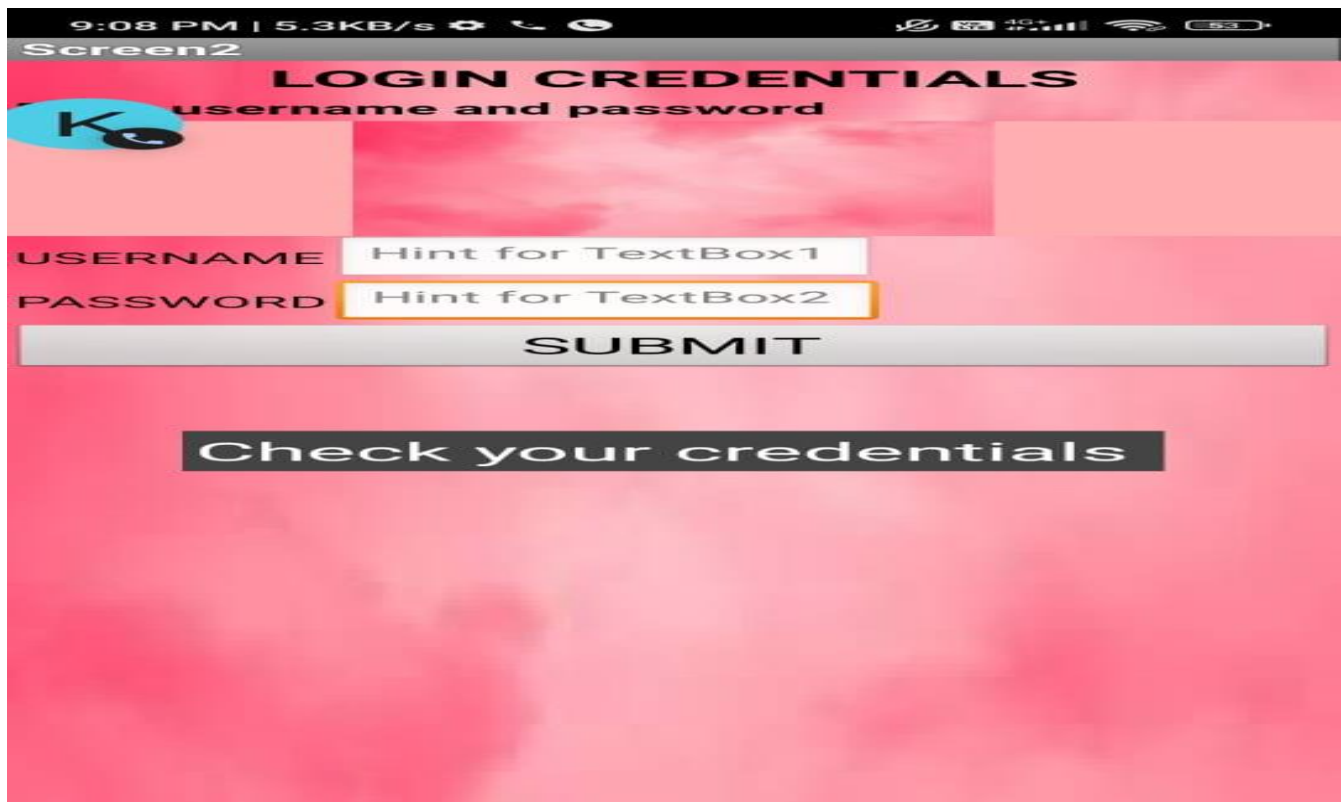
Sprint 3

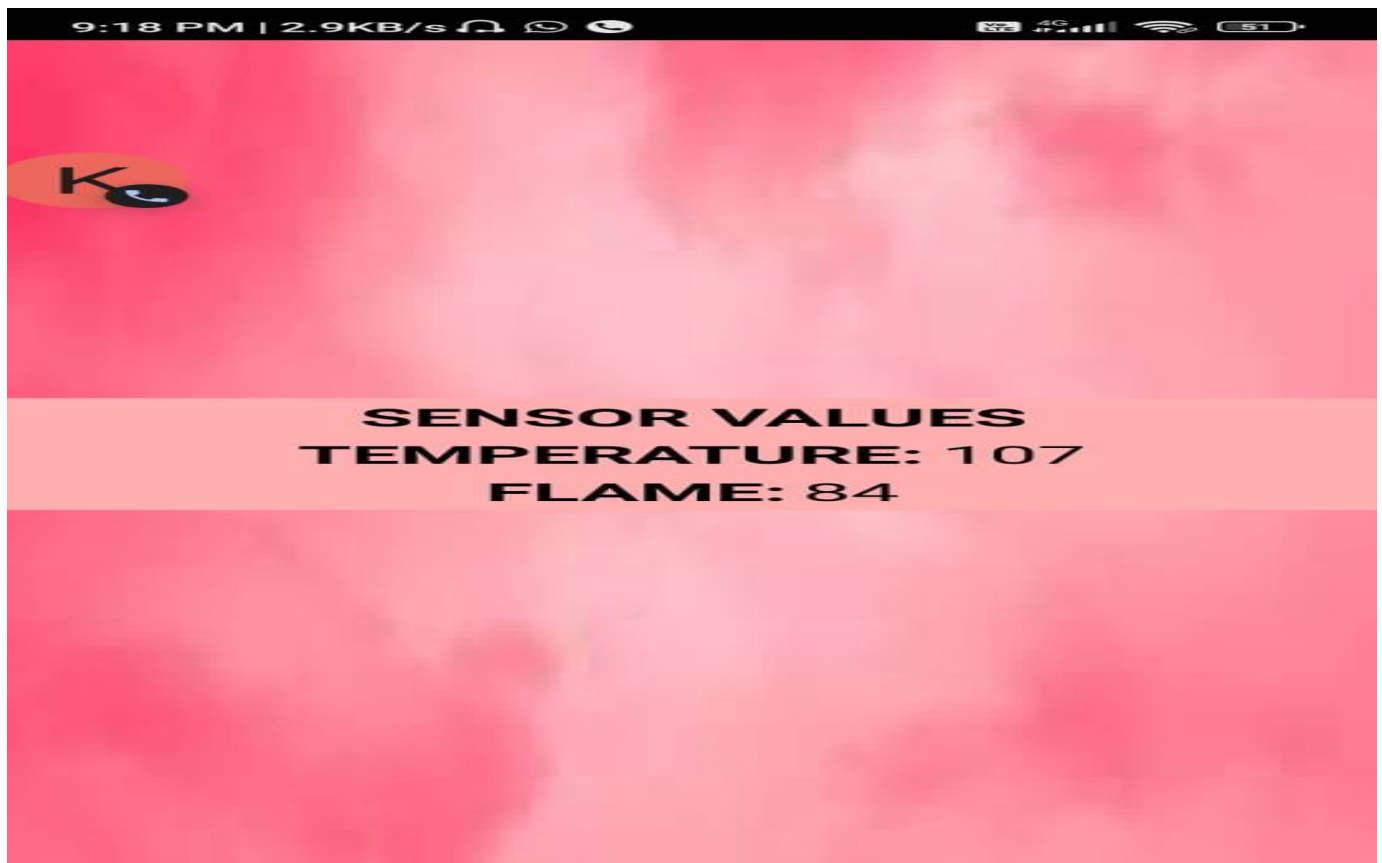




Sprint 4







7.Coding and Solutioning

Feature 1 : False alarm checking

```
Void loop()
{
    //setting a random seed
    srand(time(0));
    //initial variable activities like declaring , assigning
    temperature = data.temperature;
    gas_ppm = rand()%1000;
    int flamereading = rand()%1024;
    flame = map(flamereading,0,1024,0,1024);
    int flamerange = map(flamereading,0,1024,0,3);
    int flow = ((rand()%100)>50?1:0);
    //set a flame status based on how close it is.....
    switch (flamerange) {
    case 2:    // A fire closer than 1.5 feet away.
        flame_status = "Close Fire";
        break;
    case 1:    // A fire between 1-3 feet away.
        flame_status = "Distant Fire";
        break;
    case 0:    // No fire detected.
        flame_status = "No Fire";
        break;
    }
    //toggle the fan according to gas in ppm in the room
    if(gas_ppm > 100){
        is_exhaust_fan_on = true;
    }
    else{
        is_exhaust_fan_on = false;
    }
    //find the accident status 'cause fake alert may be caused by some mischief activities
    if(temperature < 40 && flamerange ==2){
        accident_status = "need auditing";
        is_sprinkler_on = false;
    }
    else if(temperature < 40 && flamerange ==0){
        accident_status = "nothing found";
        is_sprinkler_on = false;
    }
}
```

```

}
else if(temperature > 50 && flamerange == 1){
    is_sprinkler_on = true;
    accident_status = "moderate";
}
else if(temperature > 55 && flamerange == 2){
    is_sprinkler_on = true;
    accident_status = "severe";
}else{
    is_sprinkler_on = false;
    accident_status = "nil";
}
//send the sprinkler status
if(is_sprinkler_on){
    if(flow){
        sprinkler_status = "working";
    }
    else{
        sprinkler_status = "not working";
    }
}
else if(is_sprinkler_on == false){
    sprinkler_status = "now it shouldn't";
}
else{
    sprinkler_status = "something's wrong";
}
//Obviously the output.It is like json format 'cause it will help us for future sprints
String out = "{\n\t\"senor_values\":{";
out+="\n\t\t\"gas_ppm\": "+String(gas_ppm)+", ";
out+="\n\t\t\"temperature\": "+String(temperature,2)+", ";
out+="\n\t\t\"flame\": "+String(flame)+", ";
out+="\n\t\t\"flow\": "+String(flow)+", \n\t}";
out+="\n\t\"output\":{";
out+="\n\t\t\"is_exhaust_fan_on\": "+String((is_exhaust_fan_on)?"true":"false")+", ";
out+="\n\t\t\"is_sprinkler_on\": "+String((is_sprinkler_on)?"true":"false")+", ";
out+="\n\t}";
out+="\n\t\"messages\":{";
out+="\n\t\t\"fire_status\": "+flame_status+", ";
out+="\n\t\t\"flow_status\": "+sprinkler_status+", ";
out+="\n\t\t\"accident_status\": "+accident_status+", ";
out+="\n\t}";
out+="\n}";
Serial.println(out);

```

```

    delay(1000);
}

```

Explanation

- This set of code checks for false alarm
- It also sets the current status
- This also handles the permission management of whether a device would 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 IoT 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 action taken by the user is received as a command and stored in a buffer
- The event in the device is done according to the command
- It checks for a secret encrypted pin for performing that event

7. TESTING

7.1 Testcases

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID
Sensor_OO1	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	
Sensor_OO2	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	
Work_OO1	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	
Work_OO2	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	
Work_OO3	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	
Work_OO4	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	
Database_OO1	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	
SMS_OO1	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 phonenumber	Working as expected	Pass		N	
Work_OO5	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	
Auth_OO1	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 successful	working as expected	Pass		N	
Auth_OO2	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	
SMS_OO2	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	

7.2UAT

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

8. Results

8.1 performance metrics

CPU usage

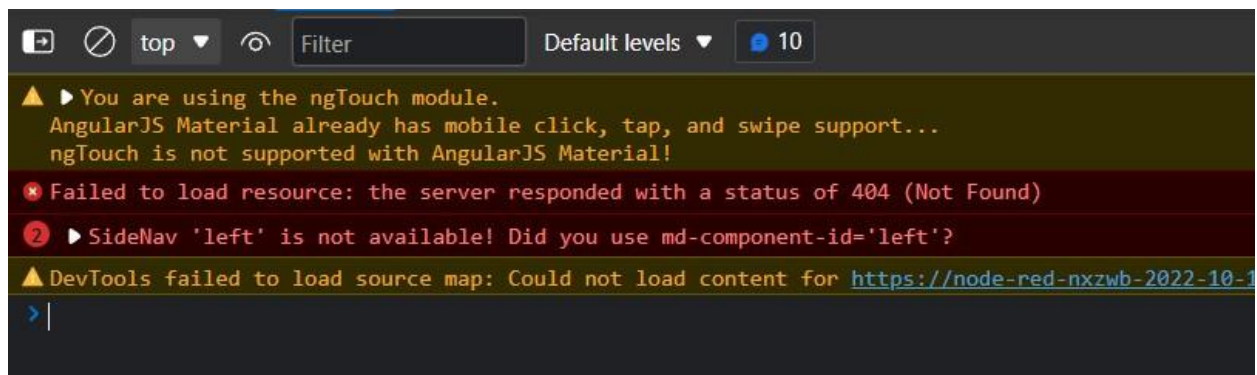
The CPU is utilized to its fullest extent by the tiny version of C++. The programme runs in $O(1)$ time for each loop, ignoring the network and communication. Every second, the software naps to improve communication with MQTT. There is less CPU demand every cycle since the programme runs in $O(1)$ time and is optimized by the compiler during compilation. The following instructions can be popped after execution since they are stored in the stack memory.

Memory usage :

The ESP32's sram is where the sensor values and networking information are kept. There is a lot of data because the ESP32 only has 520 KB of memory. To conserve memory and provide the best possible programme performance, the exact addresses are replaced with fresh values for each memory cycle.

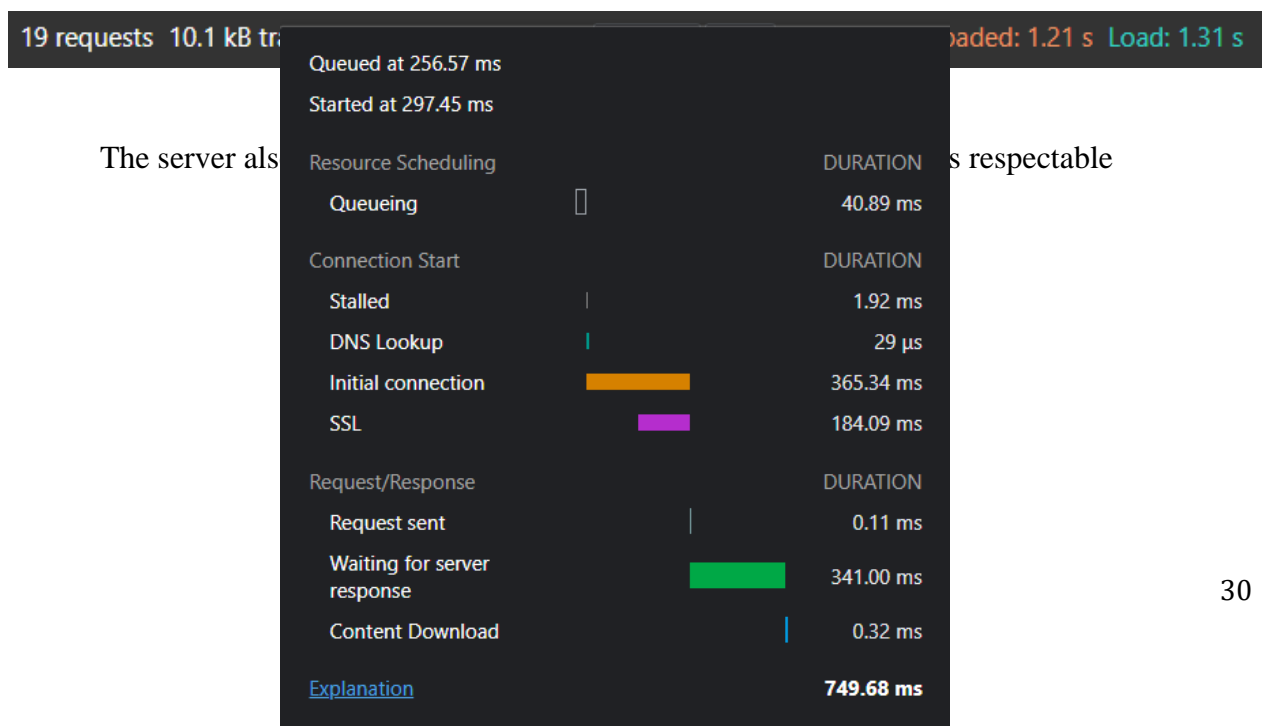
Error rates :

Due to node-handling red's of the backend and dashboard, error rates are extremely low. The handling of the exceptions is appropriate because it has no negative effects on the system's usability.



Latency and Response Time :

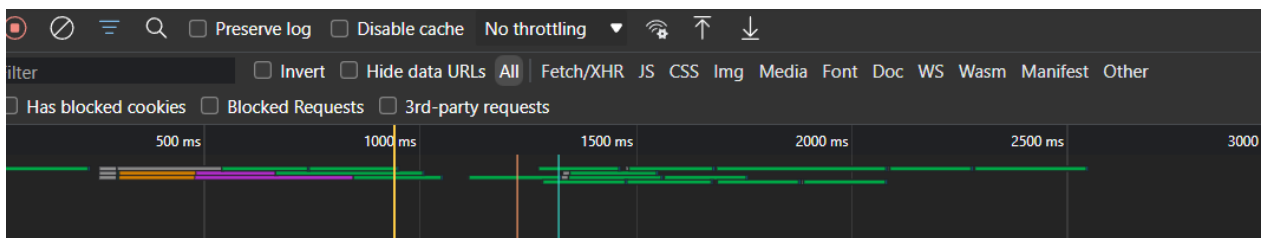
The DOM handling of the received data is optimal and latency is low .After the DOM is loaded the entire site is loaded to the browser



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

The average time is well over optimal value

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



Garbage collection:

The Node framework handles garbage collection on the server side. C++ does not have any garbage collection features for IoT devices. However, since the RAM is being utilised to store the data once more in this instance, it is not essential. No address space is allocated if it has a hanging pointer or was improperly handled.

9. Advantages and Disadvantages

Advantages

- Constant watch for gas leaks and fire outbreaks
- Automatic SMS notification of administrative and fire authorities
- Automatically turning on and off the exhaust fan and sprinklers

- It automatically detects fake fire breakout, reducing needless panic. Authentication is required to manually send SMS alerts and turn on/off sprinklers and exhaust fans.
- We may verify that the sprinkler system is functioning as intended by employing flow sensors.
- In a dashboard, every device's status can be displayed.
- The dashboard is accessible to users via a web application.

Disadvantages

- Constant internet connection required [Just for the SMS alert]
- The operation collapses if the physical device is damaged.
- Since a lot of data is kept in cloud databases every second, a huge database is required.

10. CONCLUSION

So, to sum up, our problem premise is resolved using IoT devices by developing a smart management system that addresses many inherent issues in the conventional fire management system. For example, the system actively monitors for fire breakouts as well as gas leakage and sends SMS alerts to the admin as well as the fire authorities.

11. FUTURE SCOPE

Since fire mishaps can result in significant loss of human life in both homes and large companies, the existing devices can be upgraded to operate in a variety of specialised environments and scaled for use in both public spaces and automobiles.

12. APPENDIX

Esp32 - Microcontroller :

ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth

Memory: 320 KiB SRAM

CPU: Tensilica Xtensa LX6 microprocessor @ 160 or 240 MHz

Power: 3.3 V DC

Manufacturer: Espressif Systems

Predecessor: ESP8266

Sensors :

DHT22 - Temperature and Humidity sensor

The DHT22 is a basic, low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin (no analog input pins needed).

Flow Sensors

A flow sensor (more commonly referred to as a “flow meter”) is an electronic device that measures or regulates the flow rate of liquids and gasses within pipes and tubes.

MQ5 - Gas sensor

Gas sensors (also known as gas detectors) are electronic devices that detect and identify different types of gasses. They are commonly used to detect toxic or explosive gasses and measure gas concentration.

Flame sensors

A flame-sensor is one kind of detector which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. The flame detection response can depend on its fitting

Source code:

```
#include "DHTesp.h"
#include <cstdlib>
#include <time.h>

const int DHT_PIN = 15;

bool is_exhaust_fan_on = false;
bool is_sprinkler_on = false;

float temperature = 0;

int gas_ppm = 0;
int flame = 0;
int flow = 0;

String flame_status = "";
String accident_status = "";
String sprinkler_status = "";

DHTesp dhtSensor;

void setup()
{
  Serial.begin(99900);
```

```

/**** sensor pin setups ****/

dhtSensor.setup(DHT_PIN, DHTesp::DHT22);

//if real gas sensor is used make sure the sensor is heated up for accurate 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.
*/
}

void loop() {
  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  //setting a random seed
  srand(time(0));

  //initial variable activities like declaring , assigning
  temperature = data.temperature;
  gas_ppm = rand()% 1000;
  int flamereading = rand()% 1024;
  flame = map(flamereading,0,1024,0,1024);
  int flamerange = map(flamereading,0,1024,0,3);
  int flow = ((rand()% 100)>50?1:0);

  //set a flame status based on how close it is.....
  switch (flamerange) {
    case 2: // A fire closer than 1.5 feet away.
      flame_status = "Close Fire";
      break;
    case 1: // A fire between 1-3 feet away.

```

```

    flame_status = "Distant Fire";
    break;
case 0: // No fire detected.
    flame_status = "No Fire";
    break;
}

//toggle the fan according to gas in ppm in the room
if(gas_ppm > 100){
    is_exhaust_fan_on = true;
}
else{
    is_exhaust_fan_on = false;
}

//find the accident status 'cause fake alert may be caused by some mischief activities
if(temperature < 40 && flamerange == 2){
    accident_status = "need auditing";
    is_sprinkler_on = false;
}
else if(temperature < 40 && flamerange == 0){
    accident_status = "nothing found";
    is_sprinkler_on = false;
}
else if(temperature > 50 && flamerange == 1){
    is_sprinkler_on = true;
    accident_status = "moderate";
}
else if(temperature > 55 && flamerange == 2){
    is_sprinkler_on = true;

```

```

    accident_status = "severe";
}else{
    is_sprinkler_on = false;
    accident_status = "nil";
}

```

```

//send the sprinkler status

```

```

if(is_sprinkler_on){
    if(flow){
        sprinkler_status = "working";
    }
    else{
        sprinkler_status = "not working";
    }
}
else if(is_sprinkler_on == false){
    sprinkler_status = "now it shouldn't";
}
else{
    sprinkler_status = "something's wrong";
}

```

```

//Obviously the output.It is like json format 'cause it will help us for future sprints

```

```

String out = "{\n\t\"senor_values\":{";
out+="\n\t\t\"gas_ppm\": "+String(gas_ppm)+", ";
out+="\n\t\t\"temperature\": "+String(temperature,2)+", ";
out+="\n\t\t\"flame\": "+String(flame)+", ";
out+="\n\t\t\"flow\": "+String(flow)+",\n\t}";
out+="\n\t\"output\":{";

```

```

out+="\n\t\t"is_exhaust_fan_on\":" +String((is_exhaust_fan_on)?"true":"false")+",";
out+="\n\t\t"is_sprinkler_on\":" +String((is_sprinkler_on)?"true":"false")+",";
out+="\n\t}";
out+="\n\t\t"messages\":"{ ";
out+="\n\t\t\t"fire_status\":" +flame_status+",";
out+="\n\t\t\t"flow_status\":" +sprinkler_status+",";
out+="\n\t\t\t"accident_status\":" +accident_status+",";
out+="\n\t}";
out+="\n}";
Serial.println(out);

delay(1000);
}

```

Github Link : <https://github.com/IBM-EPBL/IBM-Project-22189-1659807641>