Industry-specific intelligent fire management system

SNS College of Technology , Coimbatore

Project Report

Manikandan B

Kannaka Subbu Lakshmi B R

Bhavithra G

Arunagirinathan S

Team ID:PNT2022TMID17652

Industry Mentor : Santoshi

Faculty Mentor : S.V.Lakshmi

Title	**Page No**

1. INTRODUCTION

1.1 Project Overview

The "Industry specific-Intelligent fire management system's" goal is to prevent unintentional fire accidents in industries and to take the necessary precautions to prevent any mishaps. A Gas sensor, Flame sensor, and Temperature sensor are all part of the smart fire management system to monitor environmental changes. The sprinklers will be turned on automatically if any flame is found. The model includes a MQ2 gas sensor for detecting methane and propane gases, an IR flame sensor module for detecting flames, and an LM35 temperature sensor for measuring the surroundings. Based on the Temperature readings and if any Gases are present, the exhaust fans are turned ON. These readings are continuously tracked by IBM Watson IOT Platform and saved in Cloudant DB. Through the Nexmo SMS API, the police and fire station will be informed if any variations take place. Authorities and the fire station are informed of emergency notifications.

1.2 Purpose:

- To provide an easy management system on the dashboard .
- Providing an overview of the user's experience.
- The ability to use IoT devices to detect the status of a room
- To turn on sprinklers and exhaust fans in the event of an accident.
- To send and store temperature status in cloud storage.
- To send an SMS to the authorities in the event of 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. The sprinkler system cannot even be activated since they are utilising outdated fire safety technologies, and none of them effectively interact with one another to prevent false alarms. Applications are also being used to monitor the entire system.

2.2 References:

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

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

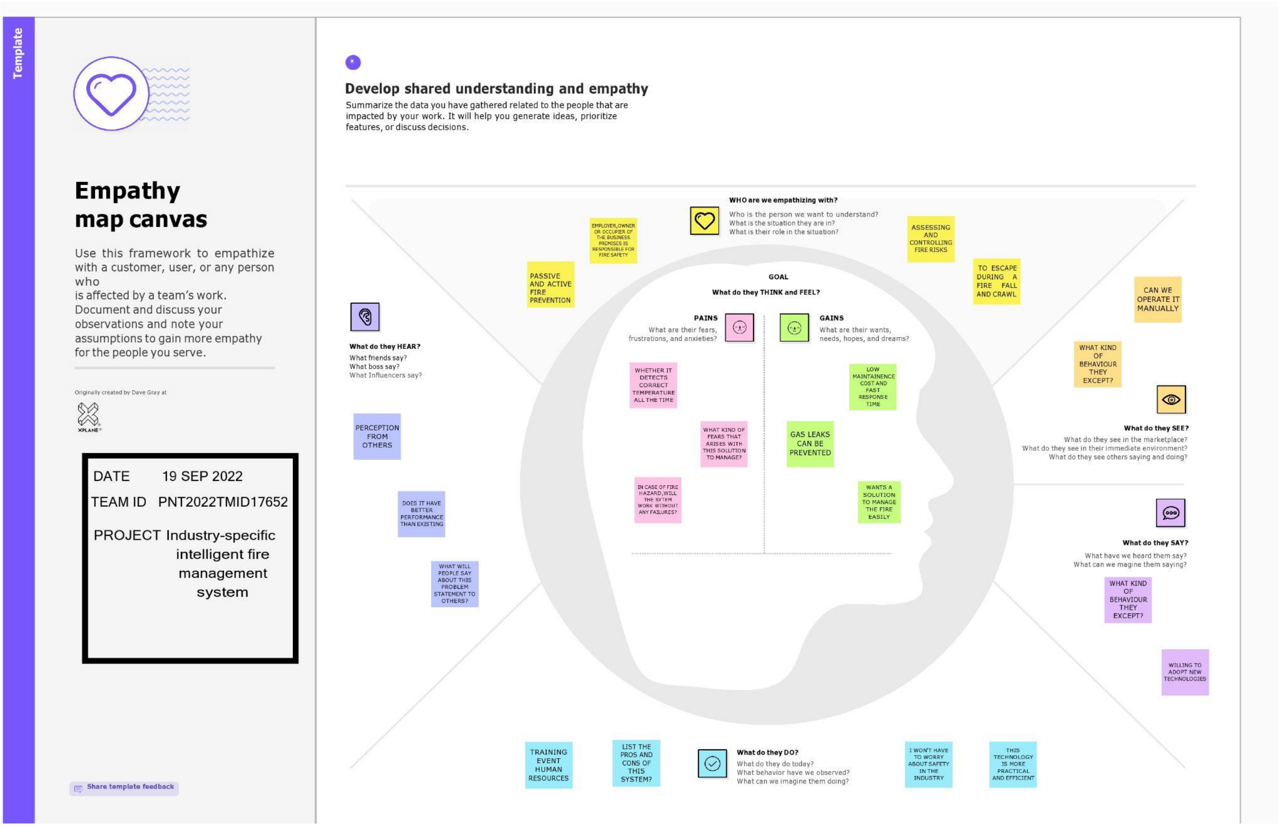
2.3 Problem Statement Definition:

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 & PROPOSED SOLUTION:

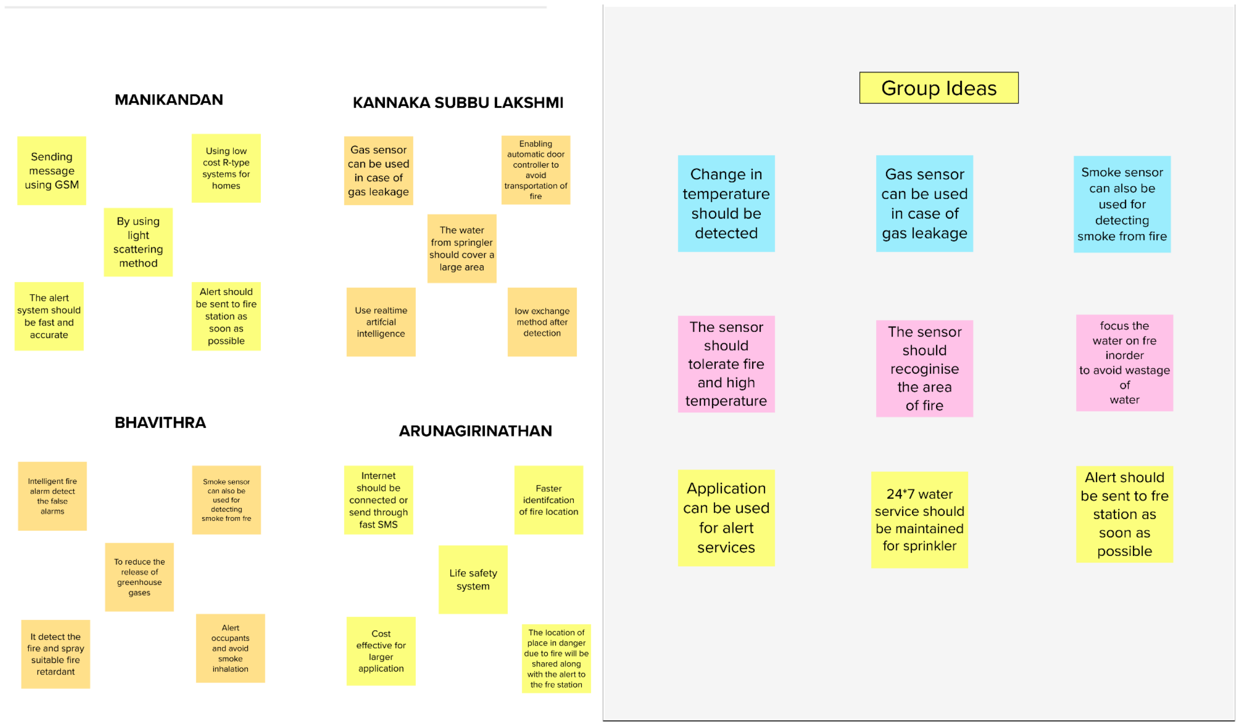
3.1 Empathy Map Canvas:

- An empathy map is a straightforward, simple-to-understand picture that summarises information about a user's actions and views.
- It is a helpful tool that enables teams to comprehend their users more fully. It's important 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 situations from the user's perspective, including goals and problems, through the exercise of constructing the map.



3.2 Ideation & Brainstorming

Step 1:Brainstorm, Idea Listing and Grouping:



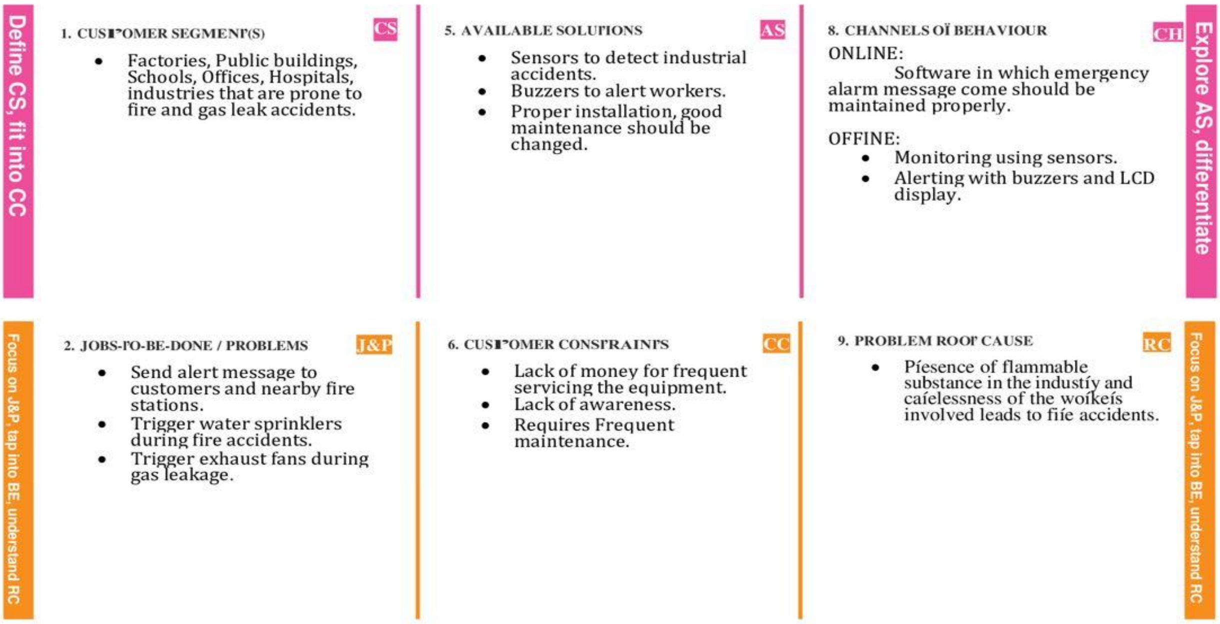
Step 2:Idea Prioritisation:

3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To address this problem, this aims to implement a smart fire detection system that would not only detect the fire using integrated sensors but also alert property owners, emergency services, and local police stations to protect lives and valuable assets simultaneously.

2.	Idea / Solution description	The proposed model in this problem statement employs different integrated detectors, such as heat, smoke, and flame. The signals from those detectors go through the system algorithm to check the fire's potentiality and then broadcast the predicted result to various parties using GSM modem associated with the system. Finally, the main feature of the proposed system is to minimise false alarms, which, in turn, makes this system more reliable.
3.	Novelty / Uniqueness	To get real-life data without putting human lives in danger, an IoT technology has been implemented to provide the fire department with the necessary data.
4.	Social Impact / Customer Satisfaction	● Highly accurate. ● It prevents accidents caused by fire in industries. ● No need for manpower. ● Human risk is low.
5.	Business Model (Revenue Model)	● High Secure. ● Our model will help industries by preventing huge losses that occur due to fire accidents.
6.	Scalability of the Solution	Since our model is cost effective because of usage of multiple sensors any and every kind of industry can use our Industry Specific Intelligent Fire Management System and it produces least false alarms.

3.4 Problem Solution fit:



4. REQUIREMENT ANALYSIS

4.1 Functional requirement:

- A system's or component's function is defined by a functional requirement, where a function is defined as the behaviour between inputs and outputs.
- It specifies what the software system " should do" ?
- It is defined at the component level and aids in software functionality verification.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Emergency alert	Alert through SMS. Alert through audible and visible alarms
FR-2	User Understanding	Based on the data, the user understands that if any of the data is above the threshold value, then there is a fire burst.
FR-3	User action	In case of fire bursts, the user needs to take actions like find the best escape route, evacuate the workers and take necessary actions to control the fire.

FR-4	Control functions	Activation of duct mounted smoke mounted detector will shut down the heating ventilation and air conditioning equipment to prevent the migration of smoke to non-affected areas of the building
FR-5	Location notification	Location of fire must be sent to fire department through an alarm.

4.2 Non-Functional requirements:

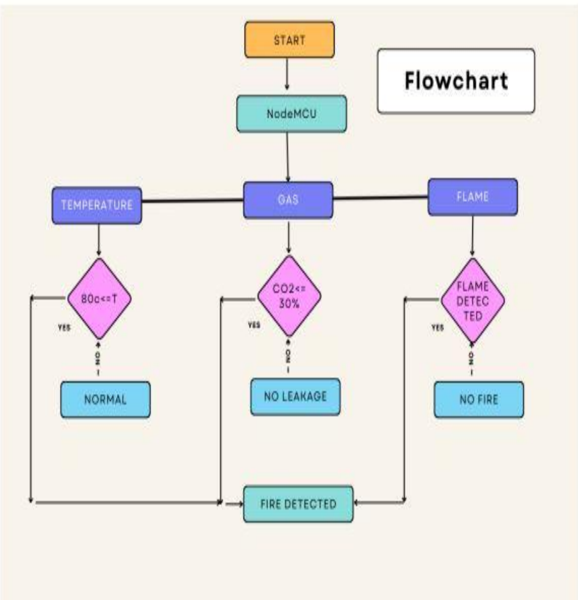
- A software system's quality characteristic is defined by a non-functional need.
- The question of "How should the software system fulfil the functional requirements?" is constrained.
- Assists you in evaluating the software's performance

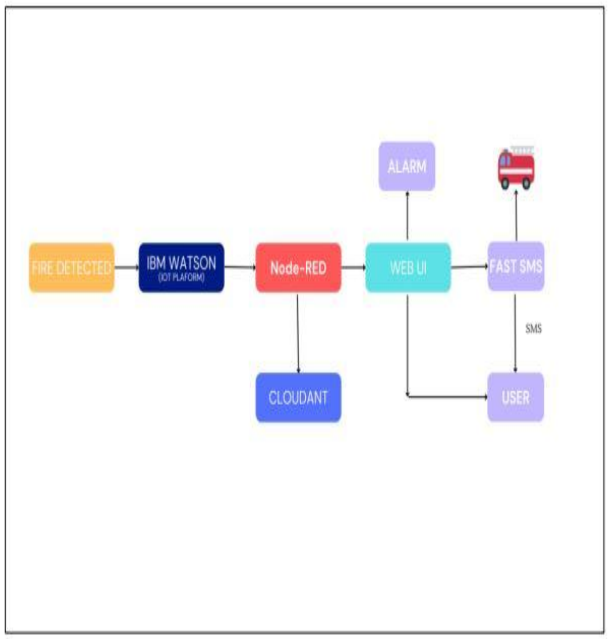
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	· Visual and audio signalization. · It provides zonal coverage. · Protect your property
NFR-2	Security	· Warn people when smoke ,fire,carbon monoxide. · Ensure the protection of both valuable items and human life.
NFR-3	Reliability	· Response timer will be faster · Reliable fire alarm systems are largely influenced. · It may be capable of precisely identifying the smoke, and it doesn't issue an erroneous warning or signal.

NFR-4	Performance	1. Detect a fire. 2. Alert occupants of the fire condition. 3. Activate safety control functions. Alert the local fire department.
NFR-5	Availability	· Ability to use the system for other types of emergency communication. · It is useful to people because it is accessible throughout the day and night.
NFR-6	Scalability	The sensors and boards used in this system should be able to easily alter and overhaul in accordance with required changes.

5. PROJECT DESIGN:

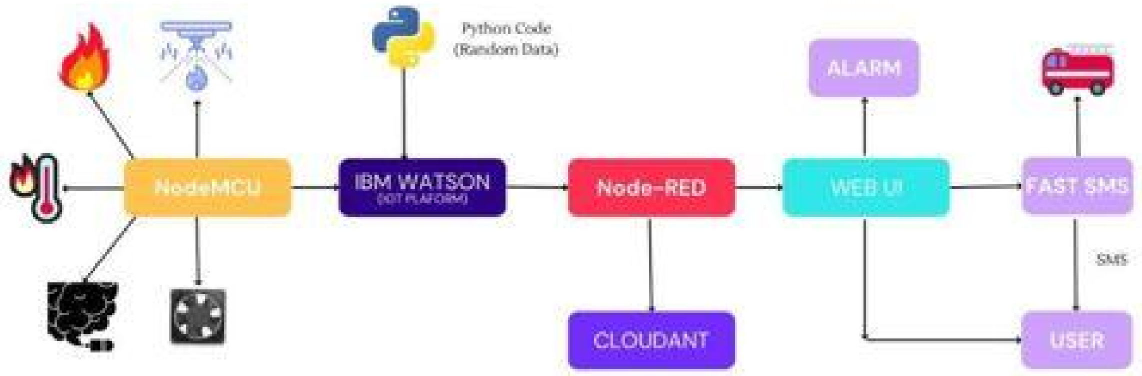
5.1 Data Flow Diagrams:



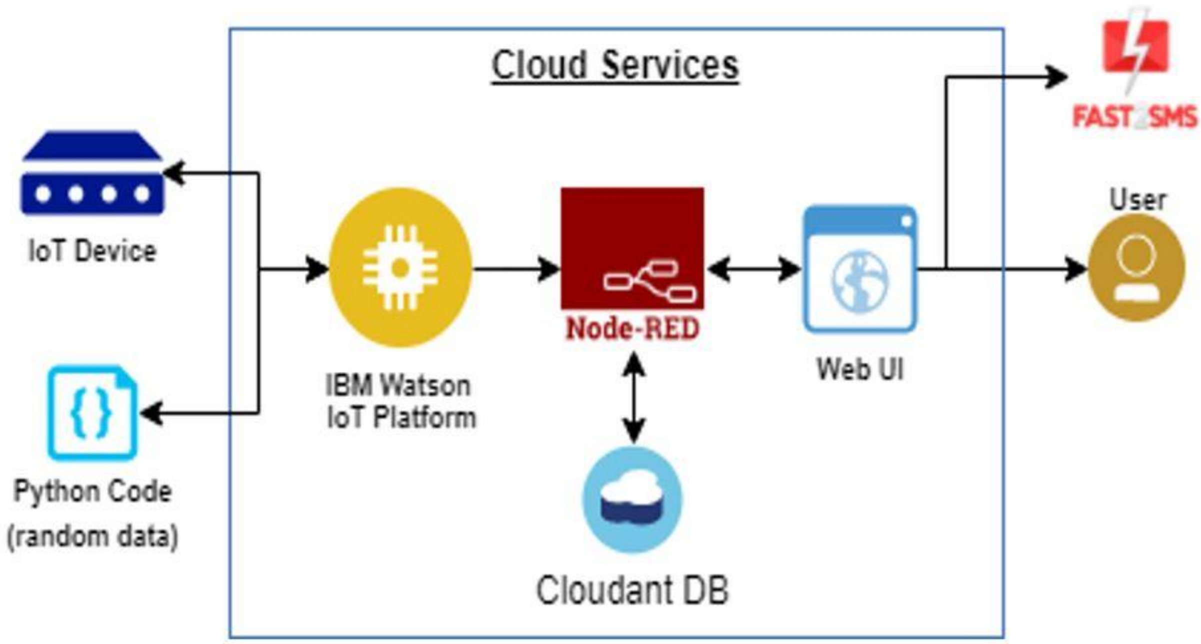


5.2 Solution & Technical Architecture:

Solution Architecture:



Technical Architecture:



S. No	Component	Description	Technology
1.	User Interface	Web UI, Node-RED, MIT app	IBM IoT Platform, IBM Node red, IBM Cloud
2.	Application Logic-1	Create Ibm Watson IoT platform and create node- red service	IBM Watson, IBM cloud ant service, IBM node-red
3.	Application Logic-2	Develop python script to publish and subscribe to IBM IoT Platform	python
4.	Application Logic-3	Build a web application using node-red service	IBM Node-red
5.	Database	Data Type, Configurations etc.	MySQL
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant
7.	File Storage	Developing mobile application to store and receive the sensors information and to react accordingly	Web UI, python
8.	External API-1	Using this IBM fire management API, we can track the temperature of the incident place and where the fire had been attacked.	IBM fire management API
9.	External API-2	Using this IBM Sensors it detects the fire, gas leaks, temperature and provides the activation of sprinklers to web UI	IBM Sensors
10.	Machine Learning Model	Using this we can derive the object recognition model	Object Recognition Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Server Configuration	IBM cloud ant, IBM IoT Platform

5.3 User stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Release
Sensing	Sensing	USN-1	Sensing the environment using the sensors.	3	High	Sprint-1
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in cause of fire and gas leakage.	3	Medium	Sprint-1
Sensor Data	Sending collected data to the IBM Watson platform	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	Sprint-2
	Registration	USN-4	Entering my email and password to verify authentication process.	3	High	Sprint-2
	Storing of sensor data	USN-5	Storing in Cloudant database.	2	Medium	Sprint-3
	Node red	USN-6	Sending the data from the IBM Watson to the Node red.	3	High	Sprint-3
Web User	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Sprint-3
Notification	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	Sprint-4
Extinguish	Turn ON/OFF the actuators	USN-9	User can turn off the Exhaust fan as well as the sprinkler system If need in that Situation.	2	Medium	Sprint-4
	Testing	USN-10	Testing of project and Final Deliverables.	1	Low	Sprint-4

6. PROJECT DESIGNING AND PLANNING:

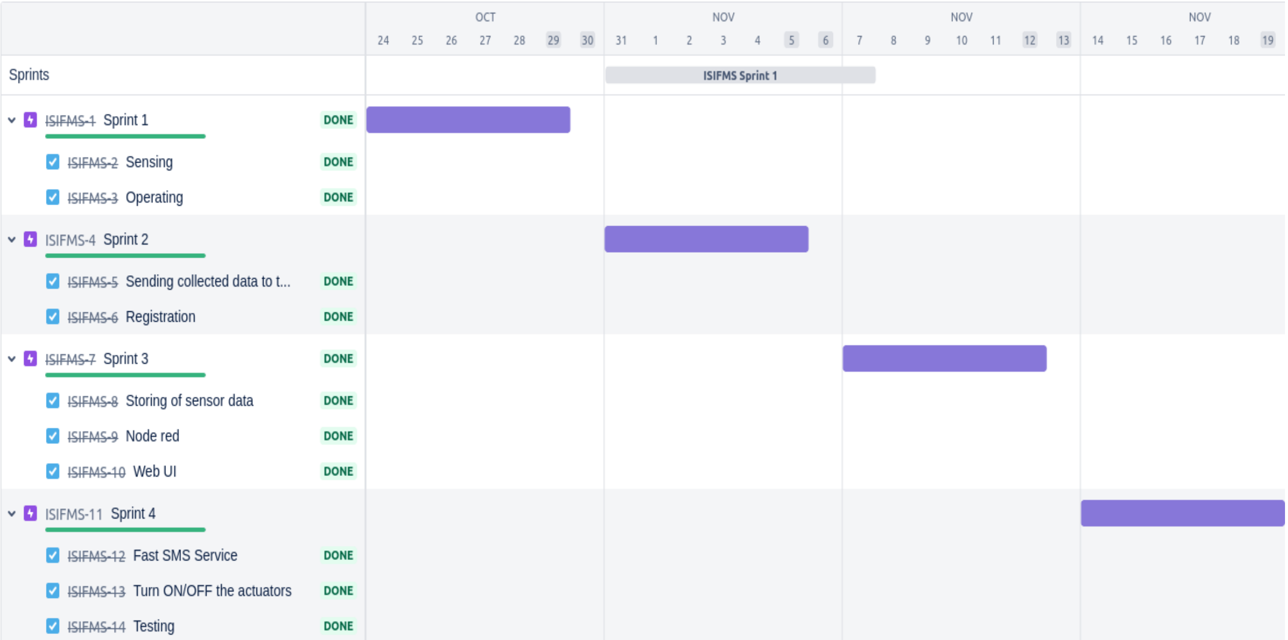
6.1 Sprint planning and estimation

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	Manikandan
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in cause of fire and gas leakage.	3	Medium	Kannaka Subbu Lakshmi
Sprint-2	Sending collected data to the IBM Watson platform	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	Manikandan, Arunagirinathan
	Registration	USN-4	Entering my email and password to verify authentication process.	3	High	Manikandan, Kannaka Subbu Lakshmi
Sprint-3	Storing of sensor data	USN-5	Storing in Cloudant database.	2	Medium	Bhavithra, Arunagirinathan
	Node red	USN-6	Sending the data from the IBM Watson to the Node red.	3	High	Kannaka Subbu Lakshmi
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Manikandan, Bhavithra
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	Manikandan, Kannaka Subbu Lakshmi
	Turn ON/OFF the actuators	USN-9	User can turn off the Exhaust fan as well as the sprinkler system If need in that Situation.	2	Medium	Manikandan, Kannaka Subbu Lakshmi
	Testing	USN-10	Testing of project and Final Deliverables.	1	Low	Arunagirinathan, Bhavithra

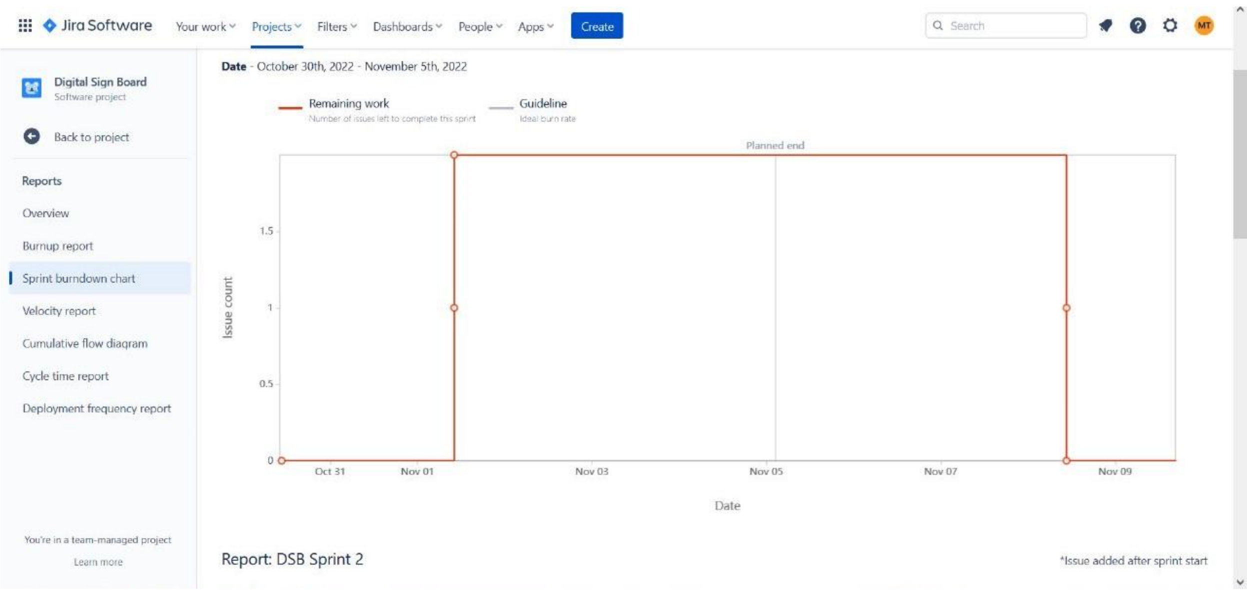
6.2 Sprint delivery schedule:

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-4	6	6 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

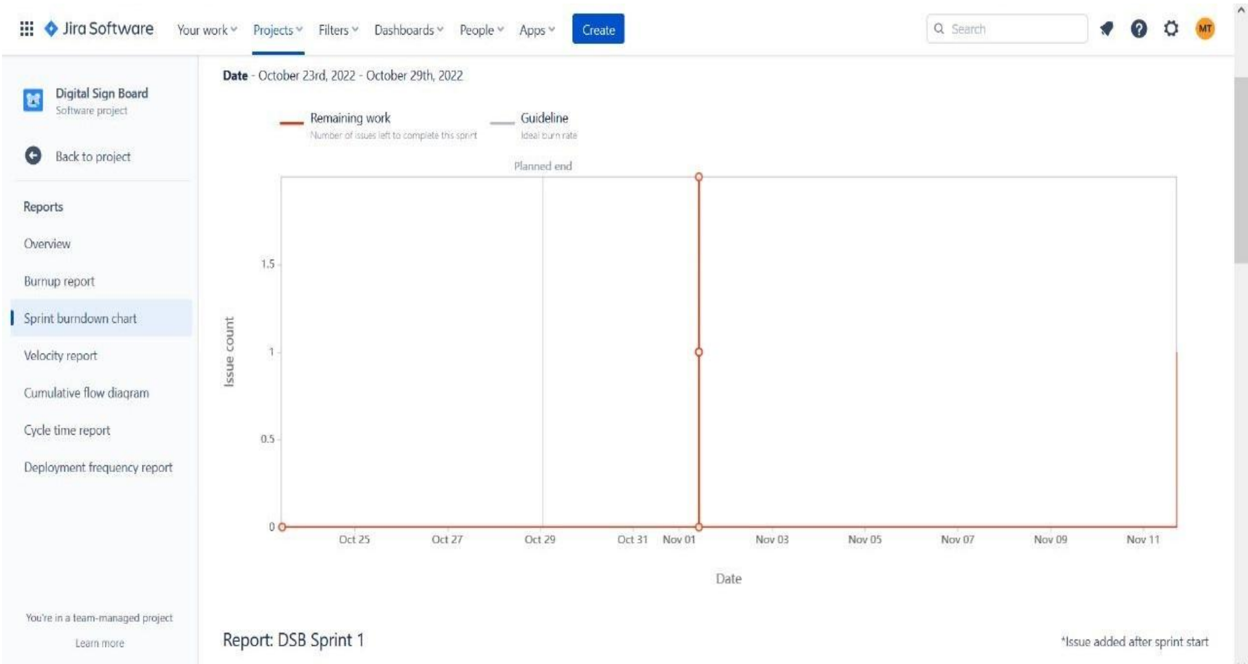
6.3 Reports from JIRA:



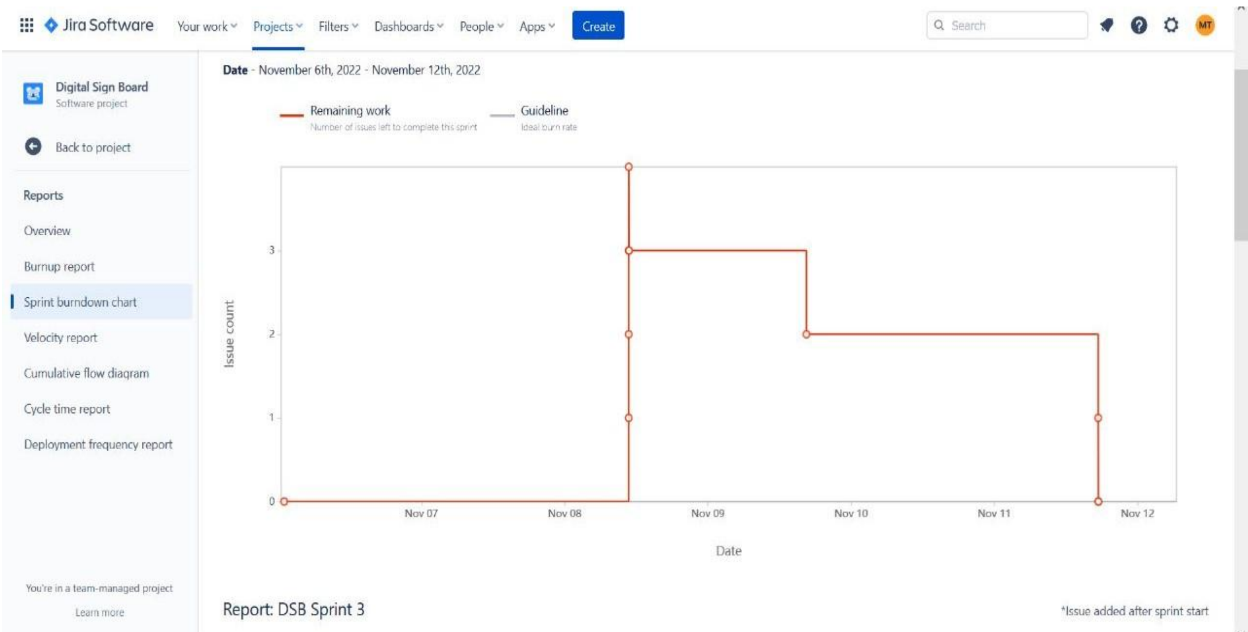
Sprint 1:



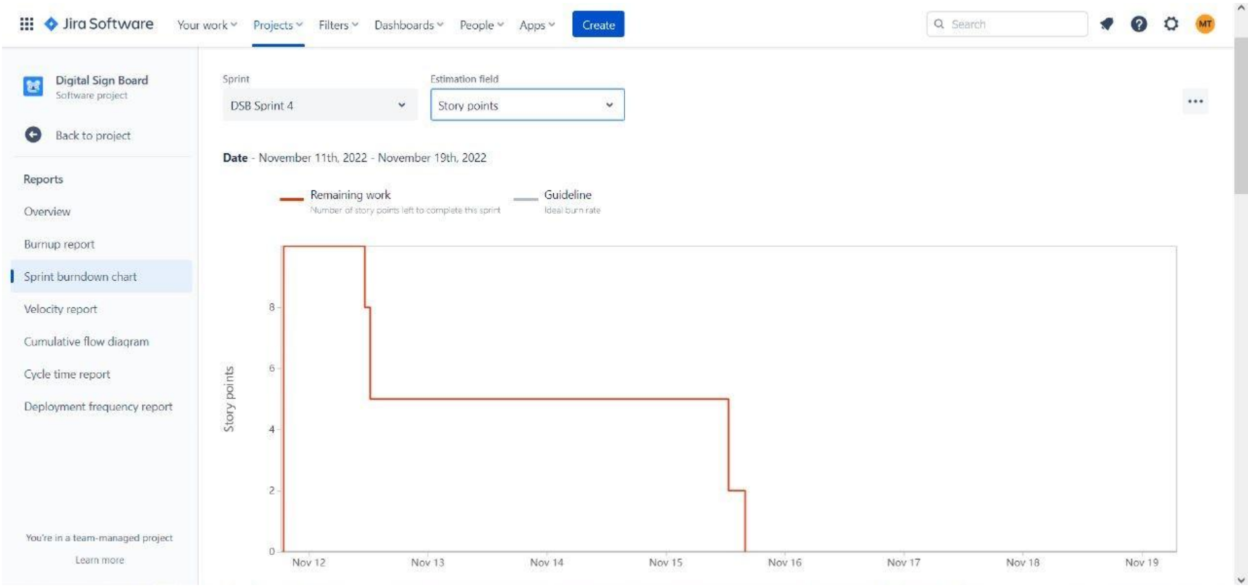
Sprint 2



Sprint 3



Sprint 4



7. CODING AND SOLUTIONING:

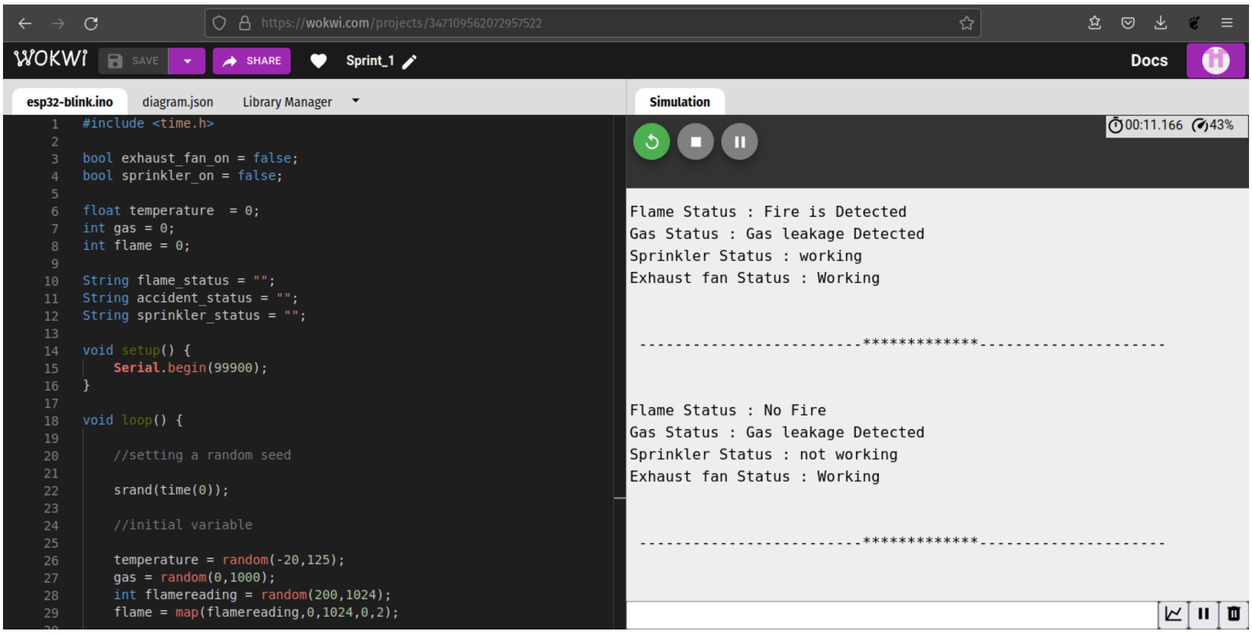
Feature 1 : alarm checking

```
//initial variable

temperature = random(-20,125);
gas = random(0,1000);
int flamereading = random(200,1024);
flame = map(flamereading,0,1024,0,2);

//set a flame status

switch (flame) {
case 0:
     flame_status = "No Fire";
     Serial.println("Flame Status :
"+flame_status);
```

```
        break;
    case 1:
        flame_status = "Fire is Detected";
        Serial.println("Flame Status :
"+flame_status);
        break;
    }

    //Gas Detection

    if(gas > 100){
        Serial.println("Gas Status : Gas leakage
Detected");
    }
    else{
        exhaust_fan_on = false;
        Serial.println("Gas Status : No Gas leakage
Detected");
    }


    //send the sprinkler status
    if(flame){
        sprinkler_status = "working";
        Serial.println("Sprinkler Status :
"+sprinkler_status);
    }
    else{
        sprinkler_status = "not working";
        Serial.println("Sprinkler Status :
"+sprinkler_status);
    }

    //toggle the fan according to gas

    if(gas > 100){
        exhaust_fan_on = true;
        Serial.println("Exhaust fan Status :
```

```
Working");
    }
    else{
        exhaust_fan_on = false;
        Serial.println("Exhaust fan Status : Not
Working");
    }
```

Result:



Explanation

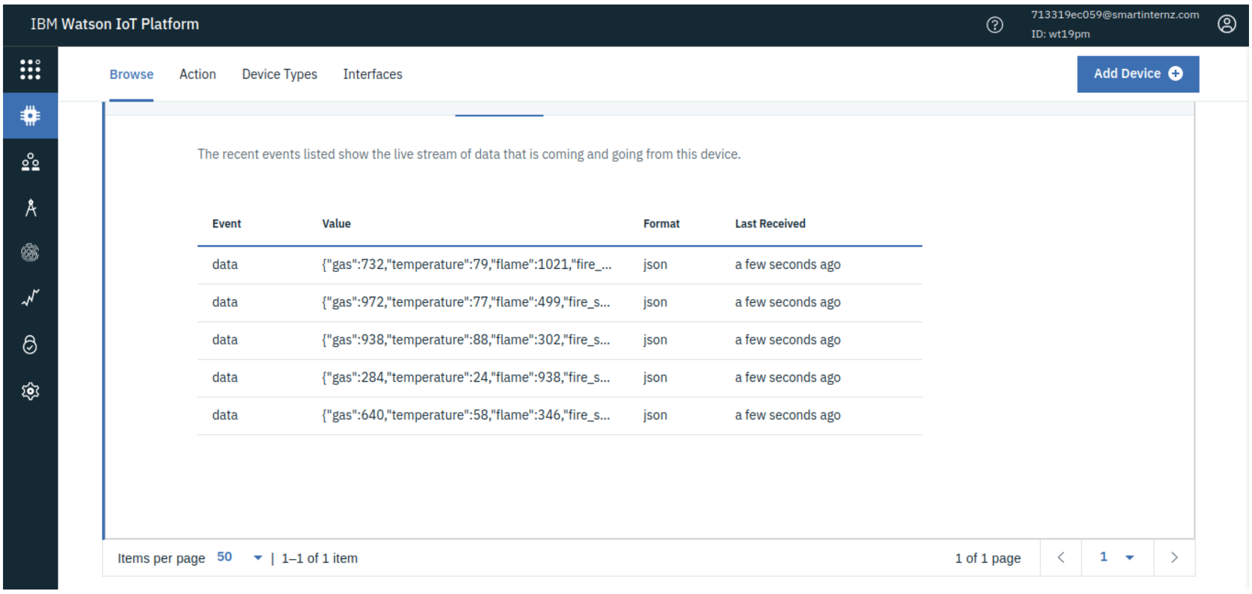
- This set of code checks for false alarms.
- It also gives the current status of actuators.

Feature 2 : Sending data into IBM Watson (JSON)

```
    String payload = "{";
```

```
    payload+="\"gas\":";
    payload+=gas;
    payload+=",";
    payload+="\"temperature\":";
    payload+=(int)temperature;
    payload+=",";
    payload+="\"flame\":";
    payload+=flamereading;
    payload+=",";
    payload+="\"fire_status\":\""+flame_status+"\",";

payload+="\"sprinkler_status\":\""+sprinkler_status+"\
",";
    payload+="\"Gas_status\":\""+Gas_status+"\",";

payload+="\"exhaust_fan_status\":\""+exhaust_fan_statu
s+"\"}";

    if(client.publish(publishTopic, (char*)
payload.c_str()))
    {
        Serial.println("Publish OK");
    }
    else{
        Serial.println("Publish failed");
    }
```

Result:



Explanation:

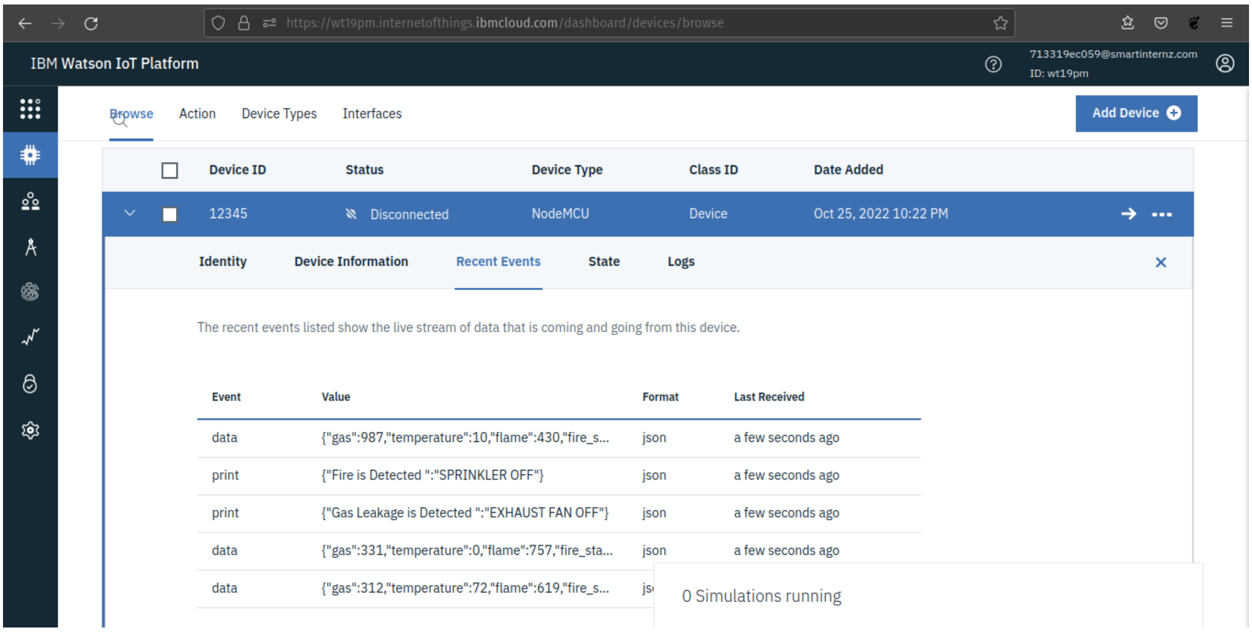
- It sends the data to IBM IoT Watson platform.

Feature 3 :

```
//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)==67993){
        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="";
}
```

Result:



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.

8. TESTING:

8.1 Testcases

Date	15-Nov-22
Team ID	PNT2022TMID17652
Project Name	Project - Industry - Specific Intelligent I
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Date	Expected Result	Actual Result	Status	Executed By	TC for Automation(Y/N)
TC_OO1	Functional	IBM cloud	Create the IBM Cloud services which are being used in this project.	IBM Cloud Login ID & Password	1.Go to IBM Cloud signup page 2.Enter e-mail id and other credentials 3.Enter a password	https://cloud.ibm.com/login	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO2	Functional	IBM Cloud	Configure the IBM Cloud services which are being used in completing this project.	IBM Cloud Login ID & Password	1.Go to Cloud login 2.Enter user ID & Password 3.Verify login by the popup display	https://cloud.ibm.com/login	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO3	Functional	IBM Watson IoT Platform	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	IBM Watson IoT Platform Login ID & Password	1.Login to IBM Cloud 2.Click Catalog 3.Search IoT and click create 4.Go to resource list and search Internet of Things platform 5.Press Launch and click Sign in IBM Watson Platform	https://w1nouz.internetofthings.ibmcloud.com/dashboard/devices/browse	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO4	Functional	IBM Watson	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials.	IBM Watson IoT Platform Login ID & Password	1.Login to IBM Watson Platform 2. Click Add Device 3.Enter the details and click Finish. Create Device ID & Device type 4.Turn on Device Simulator and click simulation running. Enter the values of temperature, pH & turbidity level 5.Click Send & Save. Verify the displayed result of the levels	Temperature, pH and turbidity sensor values are generated randomly in simulation	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO5	Functional	IBM Cloud(Node Red)	Configure the connection security and create API keys that are used in the Node-RED service for accessing the IBM IoT Platform.	Node Red Installation	1.Install node red and open node red in command prompt 2.Select IBM input in IoT	https://cloud.ibm.com/developer/appservice/create-app?starterKit=59c9d5bd-4d31-3611-897a-f94eea80dc9f&defaultLanguage=undefined	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO6	Functional	Node Red	Create a Node-RED service.	Node Red Installation	1.Select IBM IoT input in Node. In IBM IoT Watson Platform, go to apps and click on generate API keys. 2.Copy & paste generated API key and token in the IBM IoT input. After entering all details, click the done button. 3.Add debug to the IBM IoT and rename as Msg.payload and click on done. Click gauge from the dashboard and fill the details & add functions to the gauge. Check the generated values from the debug message. 4.Edit function node, connect them, add another gauge and functions, name them as "Temperature", "Gas" &"Humidity" 5.Finally add light ON/OFF buttons to the IBM IoT and debug. Verify the output from NODE RED using Local host link	Values of sensors and button for light ON/OFF is displayed	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO7	Functional	Python 3.7.0	Develop a python script to publish random sensor data such as temperature, humidity level and Gas level to the IBM IoT platform	Python 3.7.0(64 bit) installation	1.Download and install Python 3.7.0 2.Develop python code	https://www.python.org/downloads/release/python-370/	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO8	Functional	Python 3.7.0	After developing python code, commands are received just print the statements which represent the control of the devices.	Python 3.7.0(64 bit) installation	1.Downlinstall Python 3.7.0 2.After python code	Get the output from the code	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_OO9	Functional	M Cloudant D	Publish Data to The IBM Cloud	IBM Cloud Login ID & Password	1.Run the python code 2.Verify the displayed output	Publishment of python code	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
Sensor	Functional	ESP32 Controller	Sensor data is taken and parsed as JSON	Circuit Connections	1.Open the simulator in wokwi and simulate the device	Random value generated	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_O11	Storage	IBM Cloudant DB	Configure the Node-RED flow to receive data from the IBM IoT platform and also use Cloudant DB nodes to store the received sensor data in the cloudant DB		1.Go to IBM cloud, search Cloudant in Catalog, Add new dashboard, go to Node Red 2.Connect to cloudant and verify the results	Cloudant is connected by NODE RED	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N
TC_O12	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. 2. or click the sent alert button in	"Fire alert at Industries Hurry" and the trigger inputs	Results verified	No	Pass	Manikandan B , Kannaka Subbu Laakshmi B R , Bhavithra G , Arunagirinathan S	N

8.2 UAT:

Defect analysis:

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	11	5	2	3	21
Duplicate	1	0	3	0	4
External	4	5	0	1	10
Fixed	10	2	3	20	35
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	26	17	12	26	81

Test case analysis:

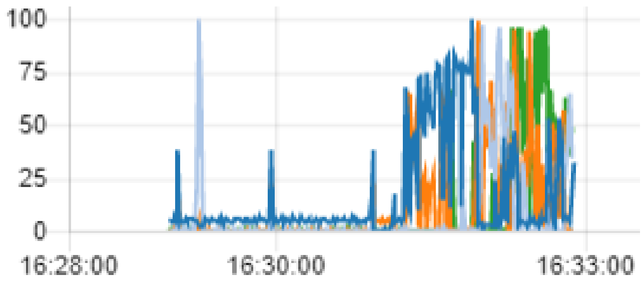
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULTS:

9.1 performance metrics

CPU usage

- Watson employs a cluster of ninety IBM Power 750 servers, each of which uses a 3.5 GHz POWER7 eight-core processor, with four threads per core. In total, the system has 2,880 POWER7 processor threads and 16 terabytes of RAM. According to John Rennie, Watson can process 500 gigabytes per second.



Memory usage

- The sensor values , networking data are stored in sram of the ESP32 . It's a lot of data because ESP32 has only a limited amount of memory (520 KB) .
- For each memory cycle the exact addresses are overwritten with new values to save memory and optimal execution of the program.

Error rates

- The exceptions are handled in a proper way as it does not affect the usability of the system**.**
- The errors rates are very low as the backend and dashboard is handled with node-red.

Latency and Response Time

- 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.

- Average time = $(5ms + 2600ms)/2 = 1302.5$

Garbage collection

- But it is not necessary in this scenario as the memory is used again for storing the data . Any dangling pointer or poorly handled address space is not allocated.

10. ADVANTAGES AND DISADVANTAGES:

Advantages

- Checking constantly for gas leaks and fire starts.
- SMS-based automatic notification of administrative and fire authorities. Turning the exhaust fan and sprinklers on and off automatically.
- Sprinkler and exhaust fan operation, as well as manually sending SMS alerts, require authentication.
- It immediately detects erroneous fire breakout, which lessens needless fright. We may verify that the sprinkler system is operating as intended by employing flow sensors.
- A dashboard is capable of displaying all device status.

Disadvantages

- To send the SMS alert, constant internet connection is required
- The entire operation falls apart if the physical apparatus is broken.
- A huge database is required since the cloud database stores a lot of data every second.

11. 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.
- The live value is shown in the dashboard when this circuit uses a temperature, flame, and gas sensor.

12. 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. In the event of any fire accidents, the police and fire station are notified.

13. 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.ESP32 is created and developed by Espressif Systems, a Shanghai-based Chinese company, and is manufactured by TSMC using their 40 nm process.

Features:

- **Memory**: 320 KiB RAM, 448 KiB ROM
- **Wireless connectivity:**
  - Wi-Fi: 802.11 b/g/n
  - Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi)
- **Peripheral interfaces:**
  - 34 × programmable GPIOs
  - 12-bit SAR ADC up to 18 channels

Sensors:

DHT22 - Temperature 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). It's fairly simple to use but requires careful timing to grab data.

Technical Detail:

- Low cost
- 3 to 5V power and I/O
- 2.5mA max current use during conversion (while requesting data)
- Good for 0-100% humidity readings with 2-5% accuracy

MQ5 - Gas sensor:

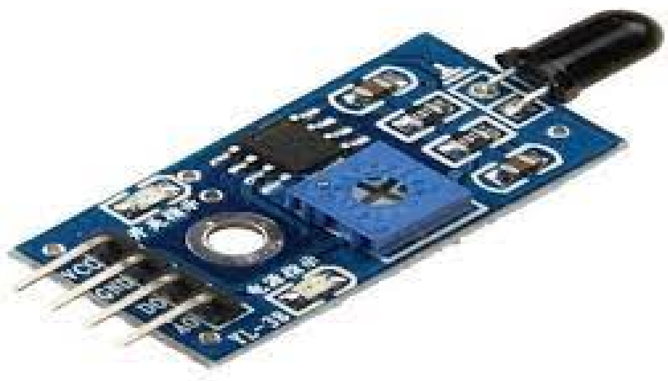
MQ-5 gas sensor has high sensitivity to butane, propane, methane and can detect methane and propane at the same time. It also can detect kinds of flammable gases, especially LPG(propane). It is a kind of low–cost sensor for many applications.

Flow Sensors:

A flow sensor is an electronic device that measures or regulates the flow rate of liquids and gases within pipes and tubes. Flow sensors are generally connected to gauges to render their measurements. Flow sensors are able to detect leaks, blockages, pipe bursts, and changes in liquid concentration.Flow sensors are of two groups:contact and non-contact flow sensors.

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. It includes an alarm system, a natural gas line, propane & a fire suppression system. This sensor is used in industrial boilers. The main function of this is to give authentication whether the boiler is properly working or not. The response of these sensors is faster as well as more accurate compared with a heat/smoke detector because of its mechanism while detecting the flame.



Source code:

```
#include <time.h>

#include <WiFi.h>

#include <PubSubClient.h>


#define ORG "wt19pm"

#define DEVICE_TYPE "NodeMCU"

#define DEVICE_ID "12345"

#define TOKEN "12345678"


char server[] = ORG
```

```
".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":"
DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);

float temperature  = 0;
int gas = 0;
int flame = 0;

String flame_status = "";
String Gas_status = "";
String exhaust_fan_status = "";
String sprinkler_status = "";


void setup() {
  Serial.begin(99900);
   wifiConnect();
   mqttConnect();
}
```

```
void loop() {

  srand(time(0));

    //initial variables and random generated data

    temperature = random(-20,125);
    gas = random(0,1000);
    int flamereading = random(200,1024);
    flame = map(flamereading,200,1024,0,2);

    //set a flame status

    switch (flame) {
    case 0:
        flame_status = "No Fire";
        break;
    case 1:
        flame_status = "Fire is Detected";
        break;
    }

    //send the sprinkler status

    if(flame==1){
        sprinkler_status = "Working";
```

```
}
else{
    sprinkler_status = "Not Working";

}

//toggle the fan according to gas reading

if(gas > 100){
    Gas_status = "Gas Leakage is Detected";
    exhaust_fan_status = "Working";

}
else{
    Gas_status = "No Gas Leakage is Detected";
    exhaust_fan_status = "Not Working";
}

//json format for IBM Watson

String payload = "{";
payload+="\"gas\":";
payload+=gas;
payload+=",";
payload+="\"temperature\":";
payload+=(int)temperature;
```

```
    payload+=",";
    payload+="\"flame\":";
    payload+=flamereading;
    payload+=",";
    payload+="\"fire_status\":\""+flame_status+"\",";

payload+="\"sprinkler_status\":\""+sprinkler_status+"\
",";
    payload+="\"Gas_status\":\""+Gas_status+"\",";

payload+="\"exhaust_fan_status\":\""+exhaust_fan_statu
s+"\"}";

    if(client.publish(publishTopic, (char*)
payload.c_str()))
    {
        Serial.println("Publish OK");
    }
    else{
        Serial.println("Publish failed");
    }
    delay(1000);



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

```
      mqttConnect();
    }
}



void wifiConnect()
{
  Serial.print("Connecting to ");
  Serial.print("Wifi");
  WiFi.begin("Wokwi-GUEST", "", 6);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.print("WiFi connected, IP address: ");
  Serial.println(WiFi.localIP());

}



void mqttConnect()
{
  if (!client.connected())
  {
    Serial.print("Reconnecting MQTT client to ");
```

```
    Serial.println(server);
    while (!client.connect(clientId, authMethod,
token))
    {
      Serial.print(".");
      delay(500);
    }

    Serial.println();
  }
}
```

Github Link : [**https://github.com/IBM-EPBL**/IBM-Project-44113-1660722212](https://github.com/IBM-EPBL/IBM-Project-44113-1660722212)

Demo Video : [**https://youtu.be/nAaq8L7xb8w**](https://youtu.be/nAaq8L7xb8w)