

Industry-specific intelligent fire management system

Project Report

Team ID:PNT2022TMID29259

Arun A

Dhamodharan C

Manikandan S

Thamizharasan R

Industry Mentor :Bharadwaj

Faculty Mentor :Arjun P

1. INTRODUCTION.....	3
1.1 Project Overview.....	3
1.2 Purpose.....	3
2. LITERATURE SURVEY.....	3
2.1 Existing problem.....	3
2.2 References.....	4
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.....	6
3.4 Problem Solution fit.....	7
4. REQUIREMENT ANALYSIS.....	8
4.1 Functional requirement.....	8
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.....	13
6. PROJECT PLANNING & SCHEDULING.....	14
6.1 Sprint Planning & Estimation.....	14
6.2 Sprint Delivery Schedule.....	14
6.3 Reports from JIRA.....	16
7. CODING & SOLUTIONING.....	18
7.1 Feature 1.....	18
7.2 Feature 2.....	20
7.3 Feature 3.....	22
8. TESTING.....	23
8.1 Test Cases.....	24
8.2 User Acceptance Testing.....	25
9. RESULTS.....	27
9.1 Performance Metrics.....	27
10. ADVANTAGES & DISADVANTAGES.....	29
11. CONCLUSION.....	29
12. FUTURE SCOPE.....	30
13. APPENDIX.....	30
Source Code.....	31

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.researchgate.net/publication/346121460>

<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:

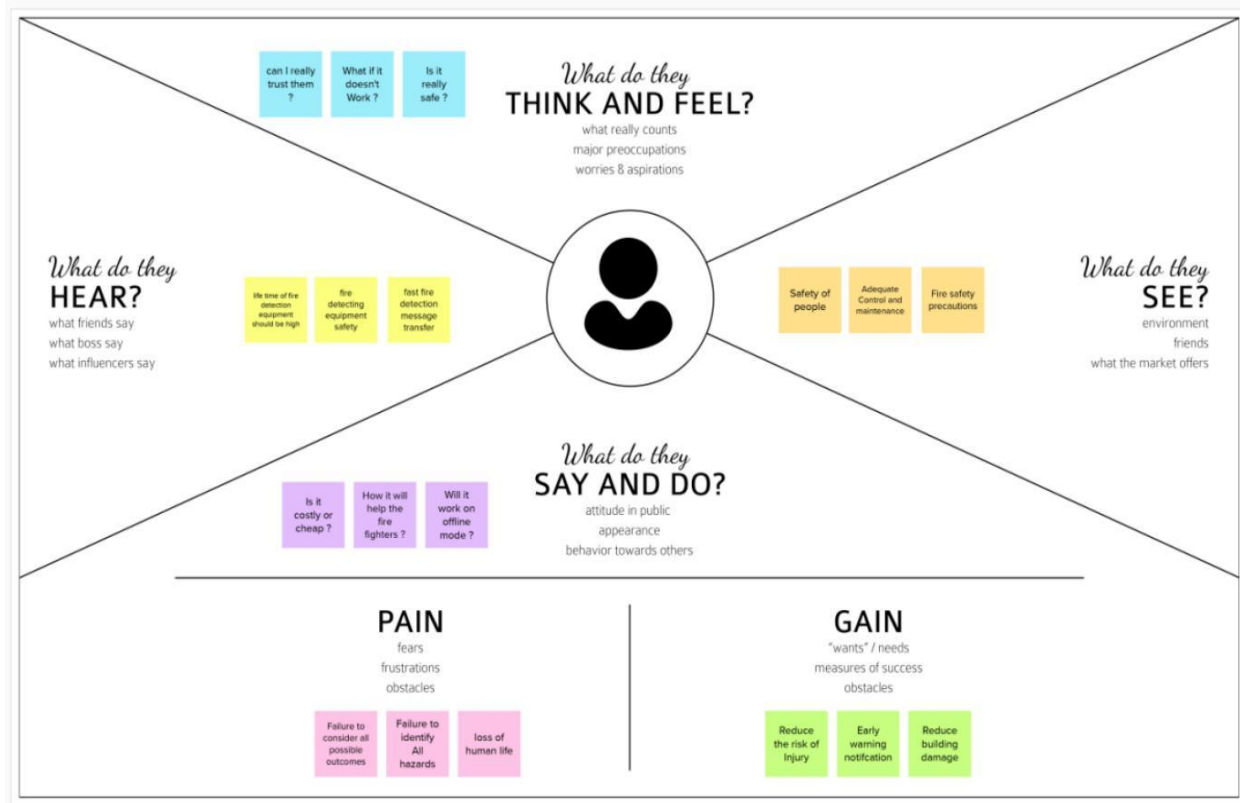
- An empathy map is a straightforward, simple-to-understand picture that summarises information about a user's actions and views.
- It is a helpful tool that enables teams to comprehend their users more fully. It's important to comprehend both the actual issue and the individual who is experiencing it 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.

Empathy Map Canvas

Gain insight and understanding on solving customer problems.

1

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



Share your feedback

3.2 Ideation & Brainstorming

Step 1: Brainstorm, Idea Listing and Grouping:




3.3 Proposed Solution:


S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Improving the safety management system against the fire incidents in industries. Fire management system helps in detecting any changes in the environment.
3.	Novelty / Uniqueness	The best use of integrating certain tasks like temperature monitoring, gas monitoring, fire detection and automatic sprinklers so as to obtain accurate information about exact locations and to get response through SMS notifications and calls.
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.
4.	Social Impact / Customer Satisfaction	It early prevents the accident cost by fire in industries. Forecasting the mishap will notify the industry workers to migrate to better and safer buildings. Provides components with affordable prices and is highly feasible.
5.	Business Model (Revenue Model)	*Reduces the amount of damage *Efficiency of cost *High security
6.	Scalability of the Solution	* This project can be used more efficiently with accurate information requiring. * Easy operability and maintenance. * High Scalability

3.4 Problem Solution fit:

Problem-Solution fit canvas 2.0		Purpose / Vision	
Define CS, fit into	1. CUSTOMER SEGMENT(S) CS Industry Workers as well as others	6. CUSTOMER CONSTRAINTS CC No proper knowledge about the fire management system. Equips the usual fire management procedures such as installing fire extinguisher and sand buckets Thinking about the capital invested on the fire management system.	5. AVAILABLE SOLUTIONS AS Laser technology solutions detect smoke by drawing air into a laser chamber to identify the possible threat. Interfacing the smoke sensor and DHT sensor with the arduino and turning on the fire alarm.
	2. JOBS-TO-BE-DONE / PROBLEMS J&P 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 Every industries have lots of materials that are highly inflammable, in case of any fire outbreak, the fire gets spread drastically. Hence due to the fire outbreak, loss of fire may occur. It causes an huge damage to the industrial properties	7. BEHAVIOUR BE This system can alert the nearby fire stations and the respective authorities in case of a fire occurrence. In an immediate response the fire alarms and water sprinklers will be turned on in order to alert the worker and suppress the fire respectively
Focus on J&P, lap into BE, understand	3. TRIGGERS TR The damage to industry causes an huge loss in production of industry. The loss of lives creates an huge loss of their family and also creates an fear to work in that industry	10. YOUR SOLUTION SL This Fire management system includes temperature sensor, gas sensor, flame sensor. If the temperature increased limit automatically the exhaust fans will be switched on. If suppose the flame is detected automatically the sprinklers will be switched on. If the harmful gases detected automatically the alarm will be switched on. Emergency alerts are noticed to the nearest fire stations and authorities.	8. CHANNELS of BEHAVIOUR CH 1. ONLINE The managers and staff can continuously monitor the temperature and flame. In case of any fire is detected immediately the nearby fire station gets alerted 2. OFFLINE The sensors with the help of intelligence can stop the fire spread at the initial stage itself.
	4. EMOTIONS: BEFORE / AFTER EM Injury or Death: A fire in an industry that results in injury or death will have huge consequences on the business owner or manager responsible for the safety of their employees and/or customers, the family of anyone who is injured or dies and the businesses ability to trade and their reputation		
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 license
 Created by Daria Nepriakhina / Amaltama.com



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	Assemble the model	Place the sensors in the appropriate locations.
FR-2	Hardware	Check that the sensors are operating properly.
FR-3	User Registration	Create user accounts for the software included in the model.
FR-4	Software	Check that alerts and SMS are being delivered correctly.
FR-5	Test the data storage	Verify that sensor data is stored in the database properly.

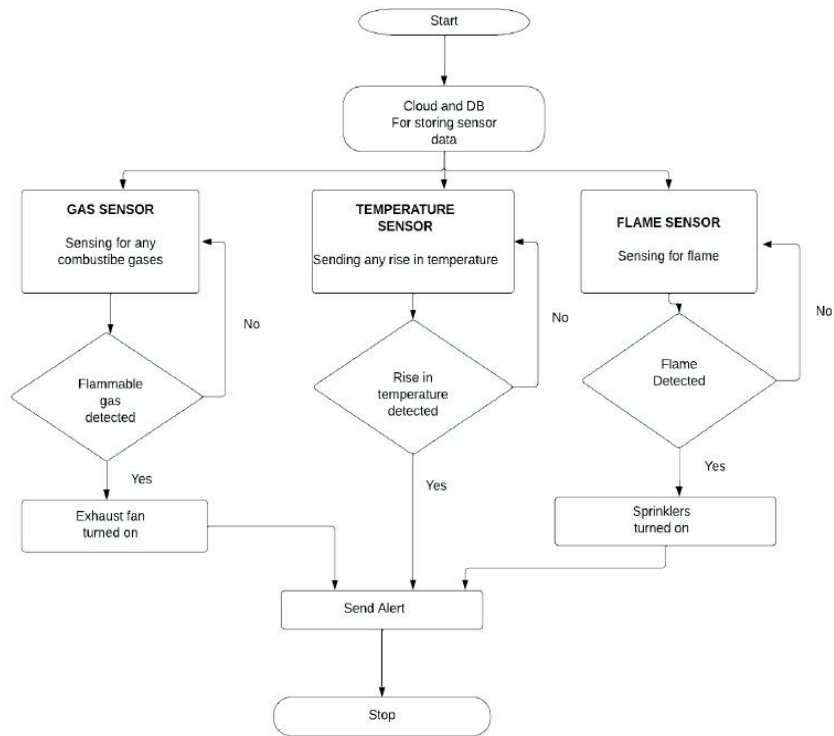
4.2 Non-Functional requirements:

- 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	The model is used in small-scale industries that deal with flammable chemicals.
NFR-2	Security	The model ensures the safety of the industry's workers by taking the necessary precautions in the event of an accident.
NFR-3	Reliability	The model is extremely reliable because it uses less energy.
NFR-4	Performance	The model sends SMS and alert to the appropriate authorities with ease.
NFR-5	Availability	The model responds instantly to the user.

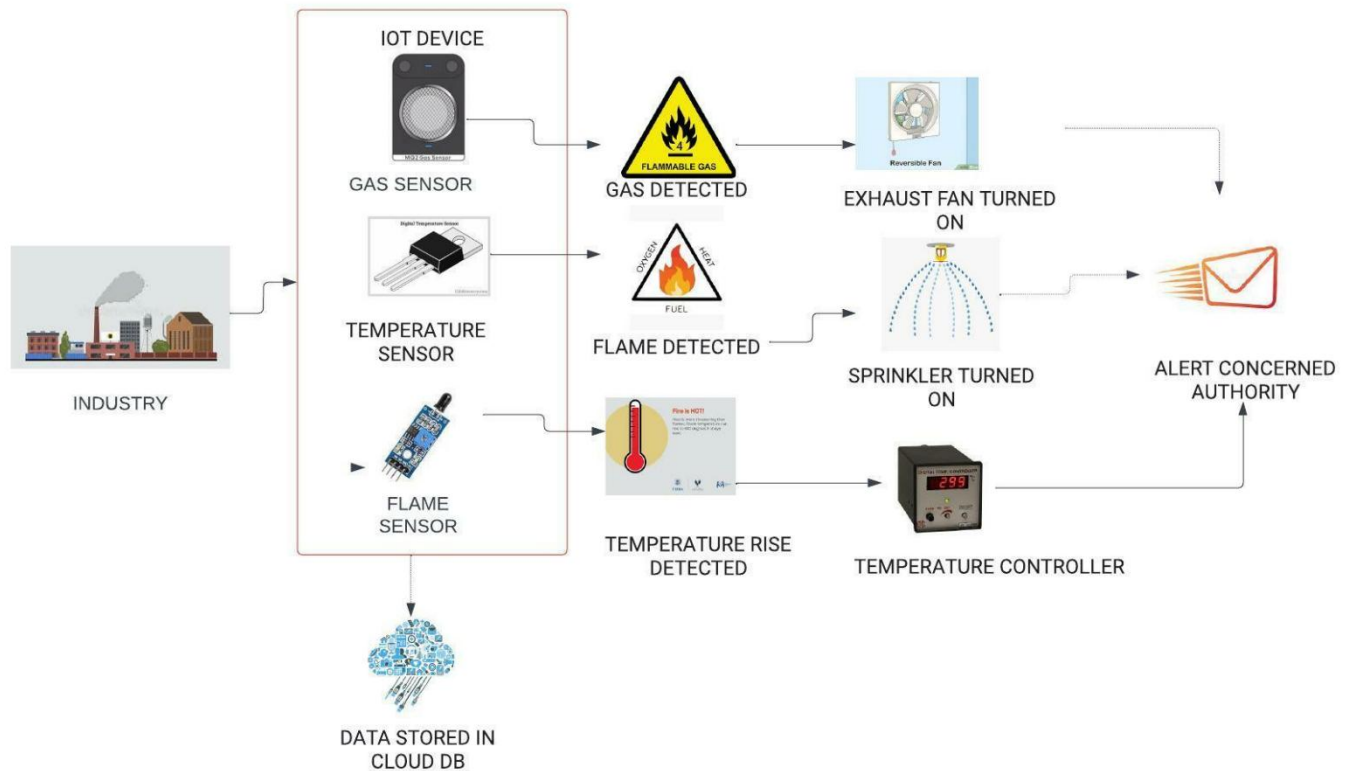
5. PROJECT DESIGN:

5.1 Data Flow Diagrams:

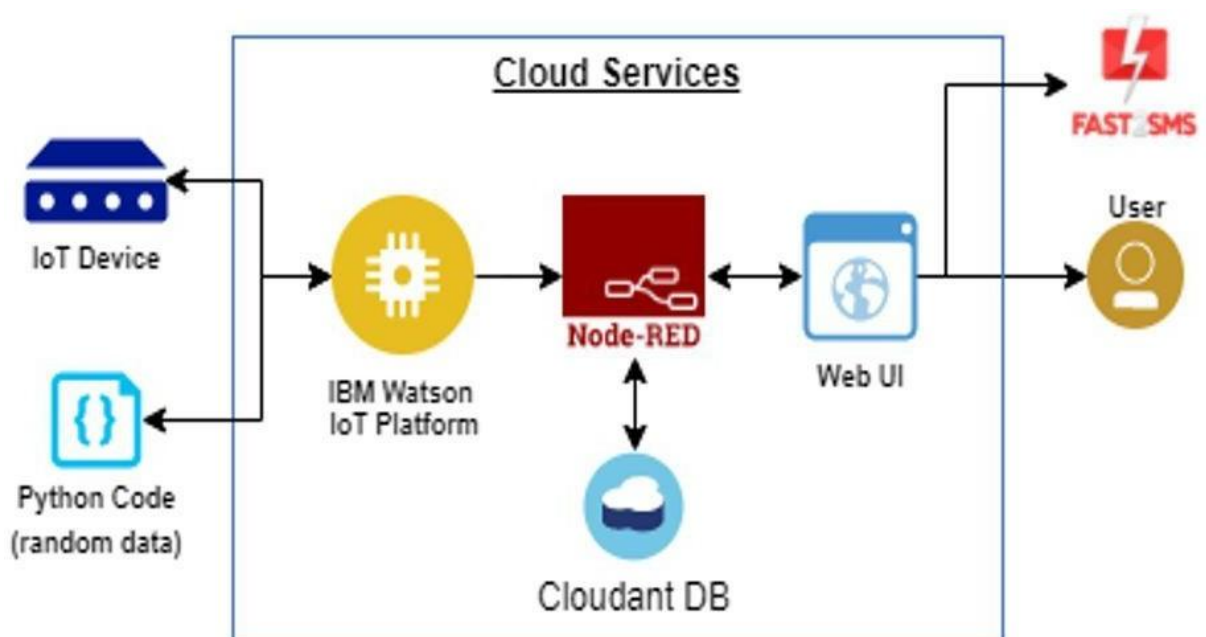


5.2 Solution & Technical Architecture:

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

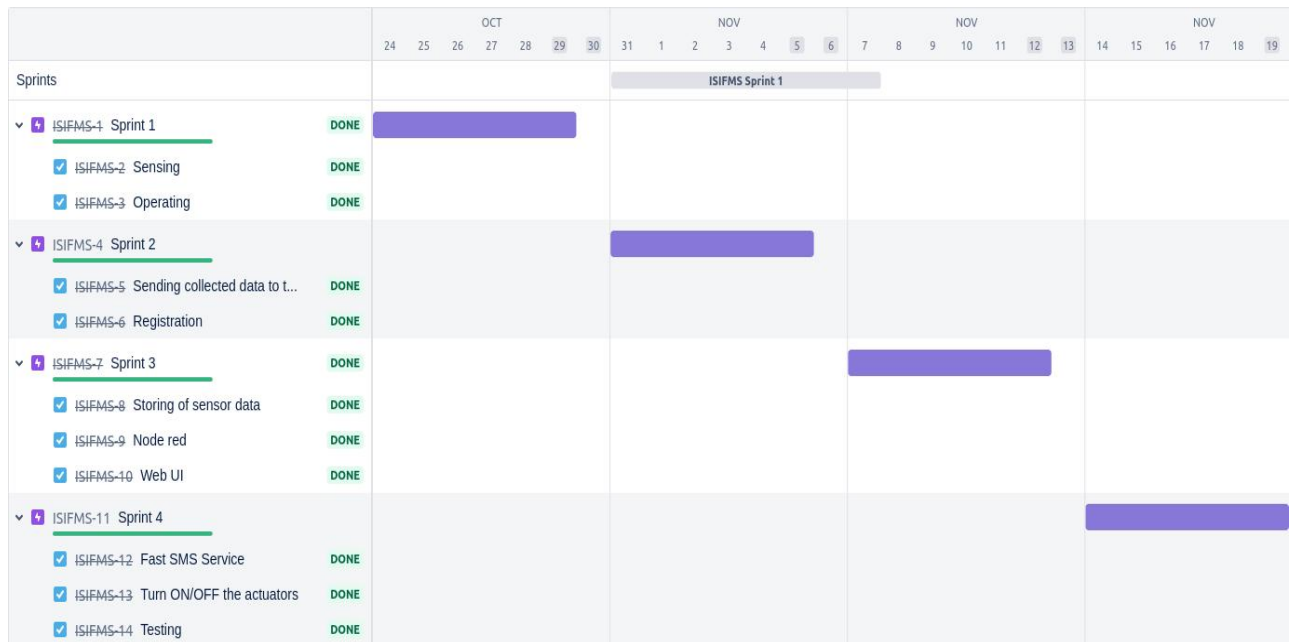
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Sensing	USN-1	Sensing the environment using the sensors.	3	High	A Arun C Dhamodharan S Manikandan R Thamizharasan
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in cause of fire and gas leakage.	3	Medium	A Arun C Dhamodharan S Manikandan R Thamizharasan
Sprint-2	Sending collected data to the IBM Watson platform	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	A Arun C Dhamodharan S Manikandan R Thamizharasan

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Node red	USN-4	Sending the data from the IBM Watson to the Node red.	3	High	A Arun C Dhamodharan S Manikandan R Thamizharasan
Sprint-3	Storing of sensor data	USN-5	Storing in Cloudant database.	2	Medium	A Arun C Dhamodharan S Manikandan R Thamizharasan
	Registration	USN-6	Entering my email and password to verify authentication process.	1	Medium	A Arun C Dhamodharan S Manikandan R Thamizharasan
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	3	High	A Arun C Dhamodharan S Manikandan R Thamizharasan
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	A Arun C Dhamodharan S Manikandan R Thamizharasan
	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	A Arun C Dhamodharan S Manikandan R Thamizharasan

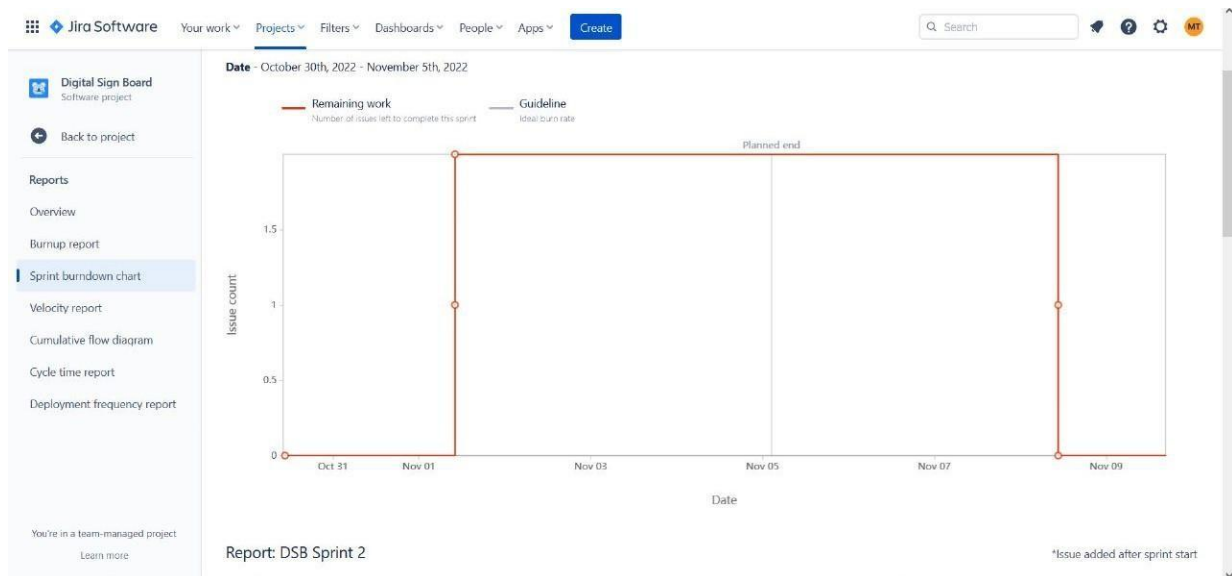
6.2 Sprint delivery schedule:

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	3 Days	24 Oct 2022	29 Oct 2022	6	10 Nov 2022
Sprint-2	6	3 Days	31 Oct 2022	05 Nov 2022	6	13 Nov 2022
Sprint-3	6	3 Days	07 Nov 2022	12 Nov 2022	6	16 Nov 2022
Sprint-4	6	3 Days	14 Nov 2022	19 Nov 2022	6	19 Nov 2022

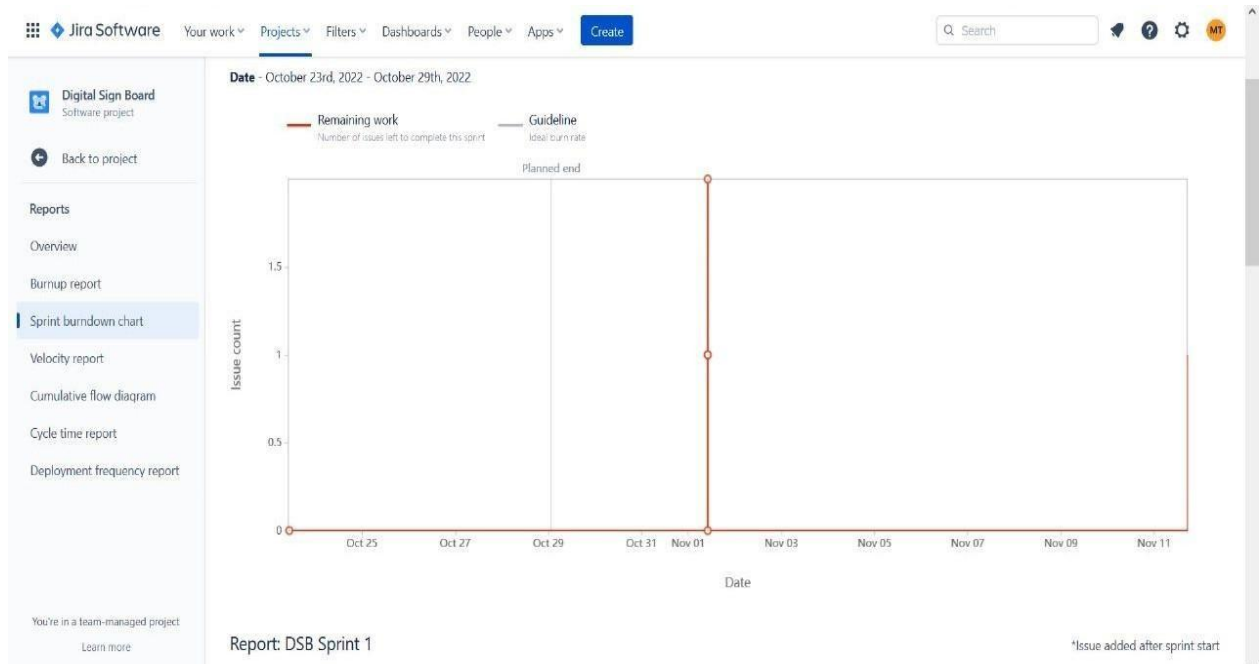
6.3 Reports from JIRA:



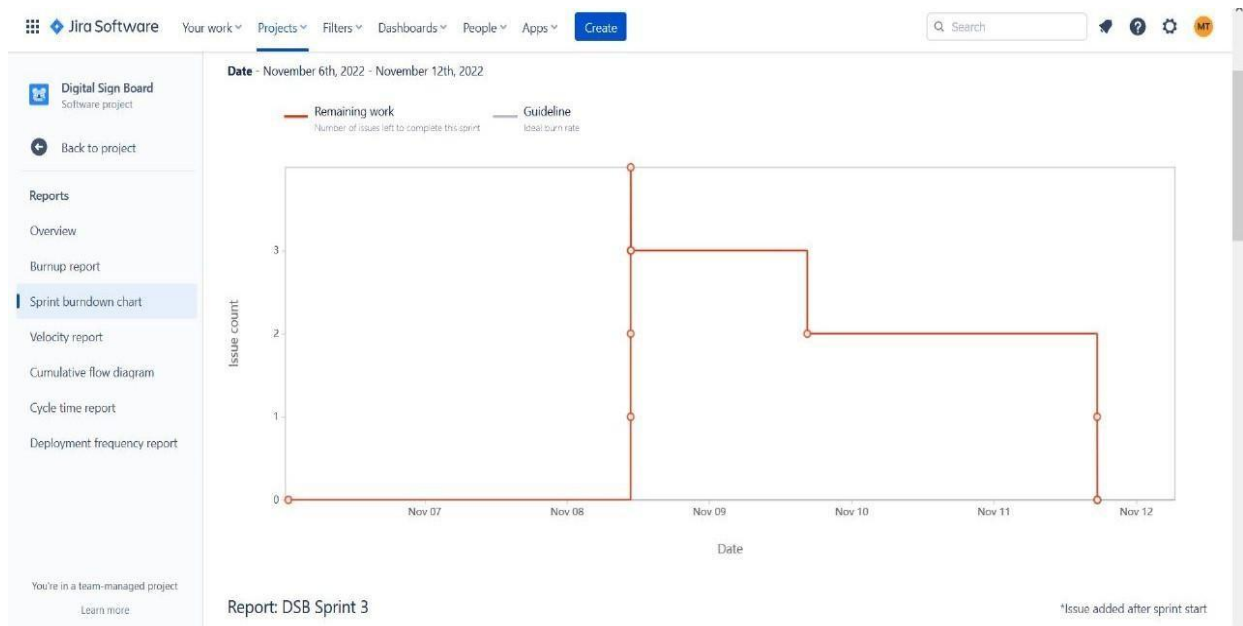
Sprint 1:



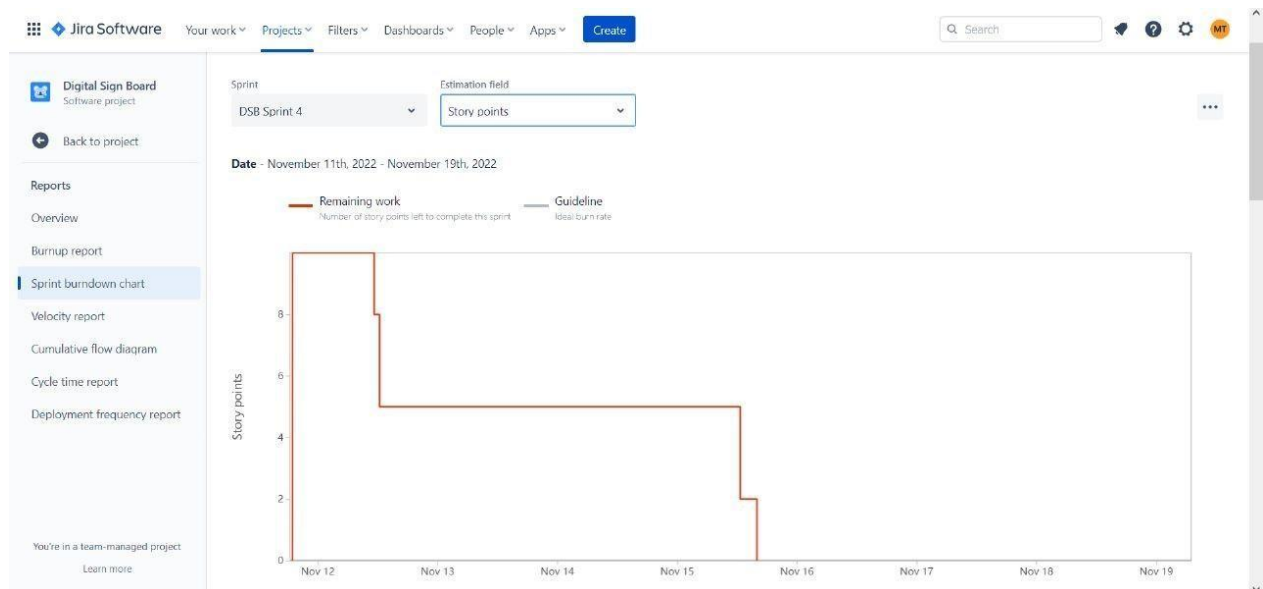
Sprint 2



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 :

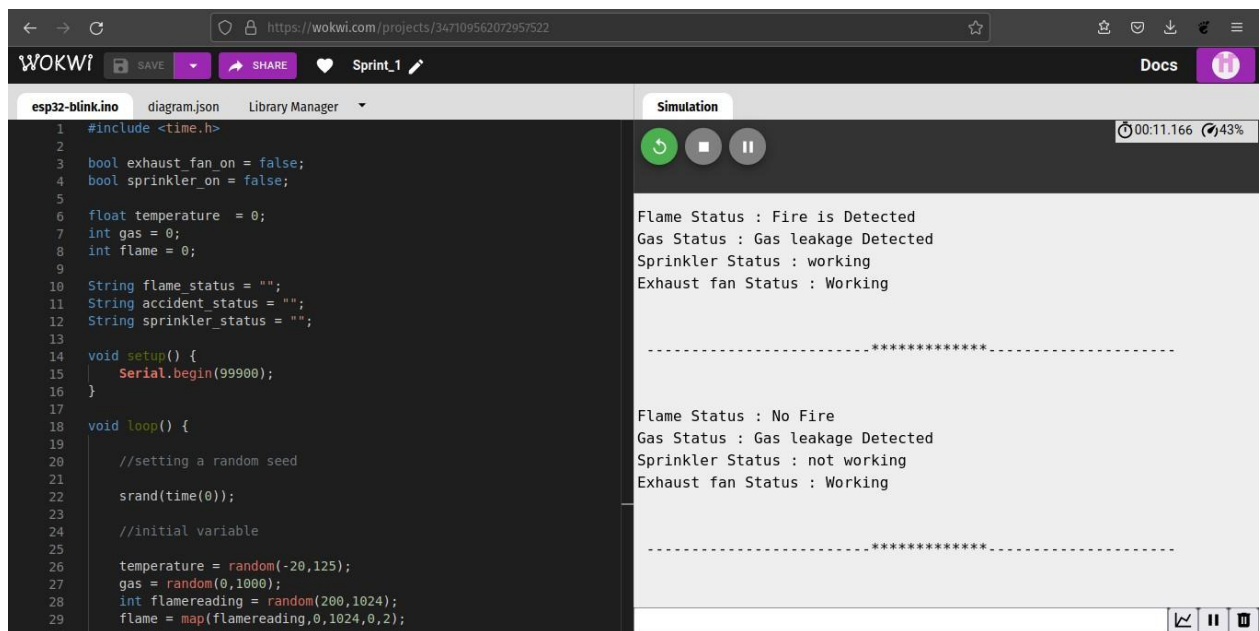
```

```

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+"\",,";

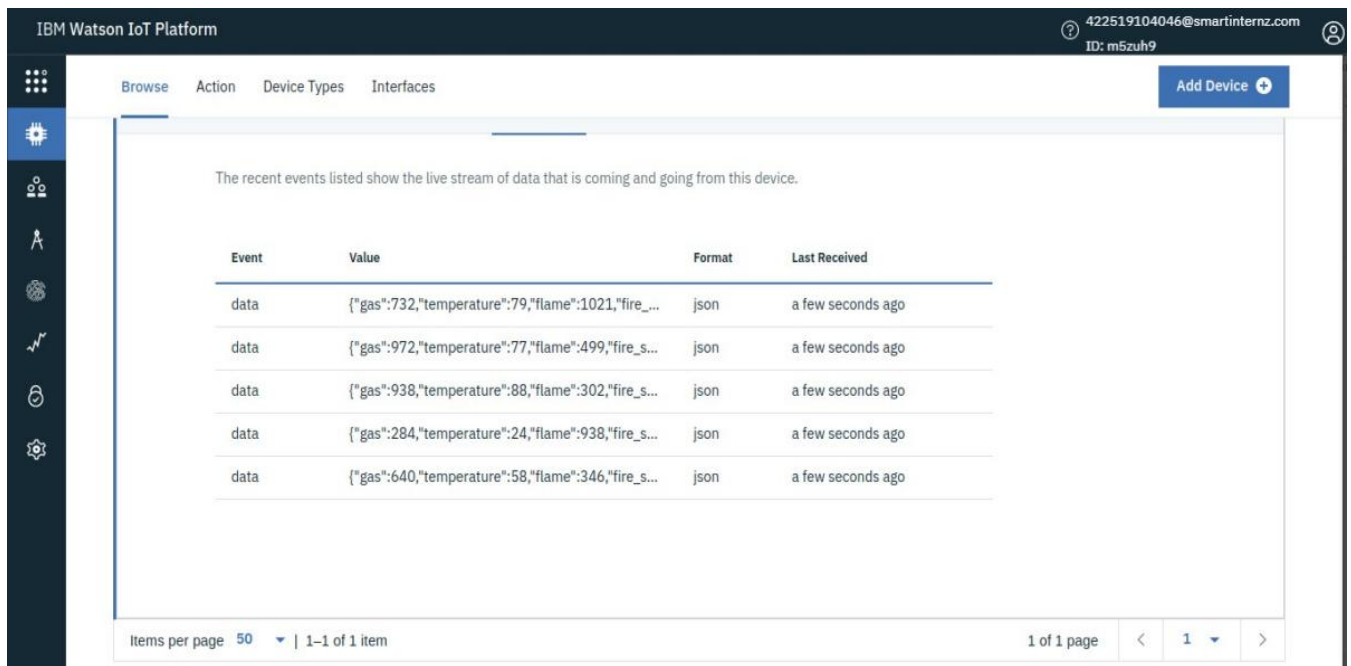
payload+="\"sprinkler_status\":"+"\""+sprinkler_status+"\"
\",,";
    payload+="\"Gas_status\":"+"\""+Gas_status+"\",,";

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

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

```

Result:



The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows the platform name and user information. Below the header, there are navigation tabs: 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area shows a message: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this message is a table with four columns: 'Event', 'Value', 'Format', and 'Last Received'. The table contains five rows of data, each representing a live stream event. At the bottom of the interface, there is a pagination bar showing 'Items per page 50' and '1-1 of 1 item'.

Event	Value	Format	Last Received
data	{"gas":732,"temperature":79,"flame":1021,"fire_...	json	a few seconds ago
data	{"gas":972,"temperature":77,"flame":499,"fire_s...	json	a few seconds ago
data	{"gas":938,"temperature":88,"flame":302,"fire_s...	json	a few seconds ago
data	{"gas":284,"temperature":24,"flame":938,"fire_s...	json	a few seconds ago
data	{"gas":640,"temperature":58,"flame":346,"fire_s...	json	a few seconds ago

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"){ canspr
inkoperate = !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:

The screenshot displays the IBM Watson IoT Platform interface. At the top, the header shows 'IBM Watson IoT Platform' and a user profile with email '422519104046@smartinternz.com' and ID 'm5zuh9'. The main navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains icons for various functions. The main content area shows a table of devices with columns: Device ID, Status, Device Type, Class ID, and Date Added. A device with ID '12345' is highlighted, showing a status of 'Disconnected' and type 'NodeMCU'. Below this, a tabbed interface shows 'Recent Events' selected. A message states: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table of events with columns: Event, Value, Format, and Last Received. The events are as follows:

Event	Value	Format	Last Received
data	{"gas":987,"temperature":10,"flame":430,"fire_s...	json	a few seconds ago
print	{"Fire is Detected ":"SPRINKLER OFF"}	json	a few seconds ago
print	{"Gas Leakage is Detected ":"EXHAUST FAN OFF"}	json	a few seconds ago
data	{"gas":331,"temperature":0,"flame":757,"fire_sta...	json	a few seconds ago
data	{"gas":312,"temperature":72,"flame":619,"fire_s...	js	

At the bottom right, it says '0 Simulations running'.

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

SLNO	INPUT	OUTPUT	RESULT
01.	Gas:218 Temperature:59.30 Flame:369	Exhaust fan on:TRUE Sprinklers:ON	Passed
02.	Gas:437 Temperature:59.30 Flame:693	Exhaust fan on:TRUE Sprinklers:OFF	Passed
03.	Gas:941 Temperature:59.30 Flame:155	Exhaust fan on:TRUE Sprinklers:ON	Passed
04.	Gas:2 Temperature:27.90 Flame:479	Exhaust fan on:FALSE Sprinklers:OFF	Passed
05.	Gas:503 Temperature:59.30 Flame:531	Exhaust fan on:TRUE Sprinklers:OFF	Passed
06.	Gas:933 Temperature:59.30 Flame:207	Exhaust fan on:TRUE Sprinklers:ON	Passed
07.	Gas:722 Temperature:59.30 Flame:855	Exhaust fan on:TRUE Sprinklers:OFF	Passed
08.	Gas:229 Temperature:29.2 Flame:309	Exhaust fan on:TRUE Sprinklers:OFF	Passed
09.	Gas:690 Temperature:75.10 Flame:440	Exhaust fan on:TRUE Sprinklers:ON	Passed
10.	Gas:0 Temperature:59.30 Flame:45	Exhaust fan on:FALSE Sprinklers:OFF	Passed
11.	Gas:1 Temperature:27.80 Flame:53	Exhaust fan on:FALSE Sprinklers:OFF	Passed
12.	Gas:843 Temperature:50 Flame:167	Exhaust fan on:TRUE Sprinklers:OFF	Passed
13.	Gas:347 Temperature:64 Flame:815	Exhaust fan on:TRUE Sprinklers:ON	Passed
14.	Gas:414 Temperature:64.40 Flame:491	Exhaust fan on:TRUE Sprinklers:OFF	Passed

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

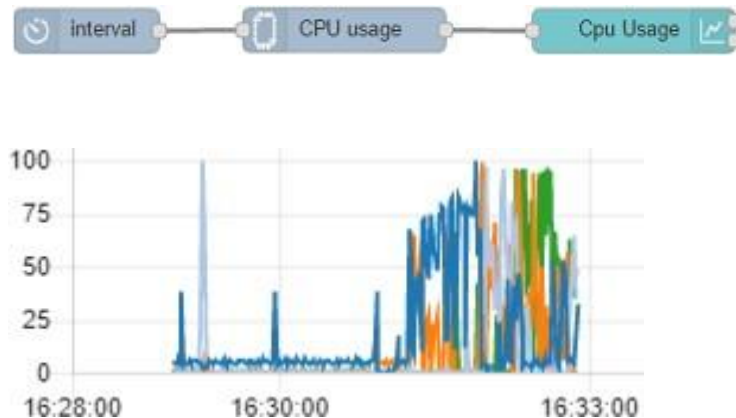
9. RESULTS:

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 = $(5 + 2600) / 2 = 1302.5$

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.

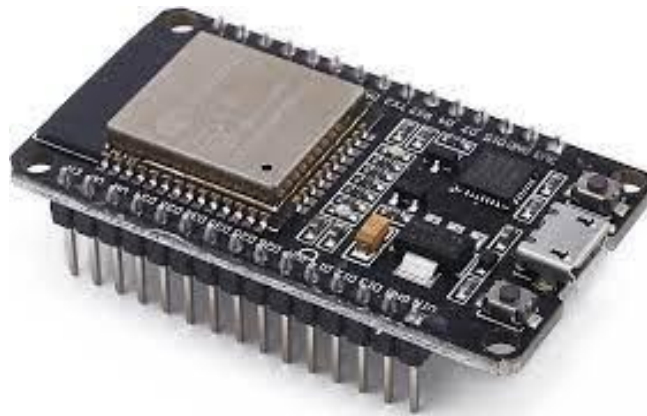
13. APPENDIX:

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_status+"\"}";

    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 : [HTTP://github.com/IBM-EPBL/IBM-Project-24043-1659936123](http://github.com/IBM-EPBL/IBM-Project-24043-1659936123)