Industry-Specific Intelligent Fire Management System

KONGU ENGINEERING COLLEGE , PERUNDURAI

Project Report

Vigneshwaran S

Senthur Kumar B

Vimalan S S

Yogeshwaran S

Team ID: PNT2022TMID04760

Mentor : Magesh kumar G

Title **Page No**

1. Introduction

1.1 Project Overview

The smart fire management system includes a gas, flame, and temperature sensor to detect any environmental changes. Based on the temperature readings and if any gases are present the exhaust fans are powered ON. If any flame is detected the sprinklers will be switched on automatically. Emergency alerts are notified to the authorities and the Fire station.

1.2 Purpose

- To give a detect the status of the room with IoT devices
- To turn on sprinkler and exhaust fan when there is accident
- To send and store the temperature status in a cloud storage
- To give a easy management system on dashboard
- To give a overview of what's happening to the user
- To send a sms to the authorities when there is a fire accident

2. Literature survey

2.1 Existing Problem

The situation is not ideal because many buildings lack automatic alert systems for administrators and authorities and lack advanced processing, making fire management systems in homes and businesses less reliable, efficient, and cost-effective. In order to prevent false alarms, they are using outdated fire safety systems that cannot even activate the sprinkler system. Applications are also being used to monitor the entire system.

2.2 Reference

https://ieeexplore.ieee.org/abstract/document/8261194

https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/6607

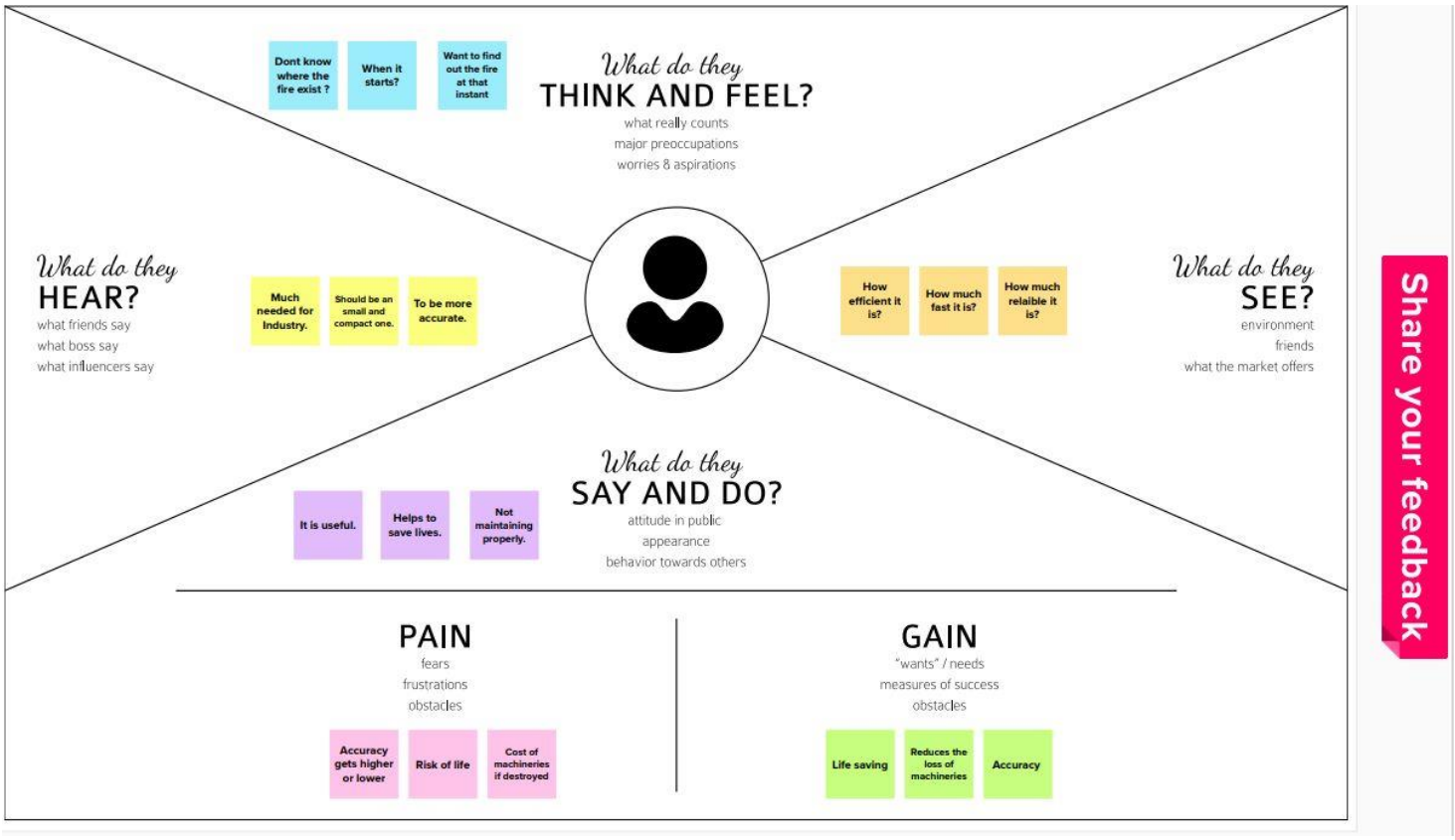
2.3 Problem Statement Definition

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 Many buildings still use outdated fire safety systems that can't even activate the sprinkler system, and they all improperly communicate with one another to prevent false alarms.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 simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes
- It is a useful tool to helps teams better understand their users.Creating an effective solution requires understanding the true problem and the person who is experiencing it
- The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 Ideation and Brainstorming

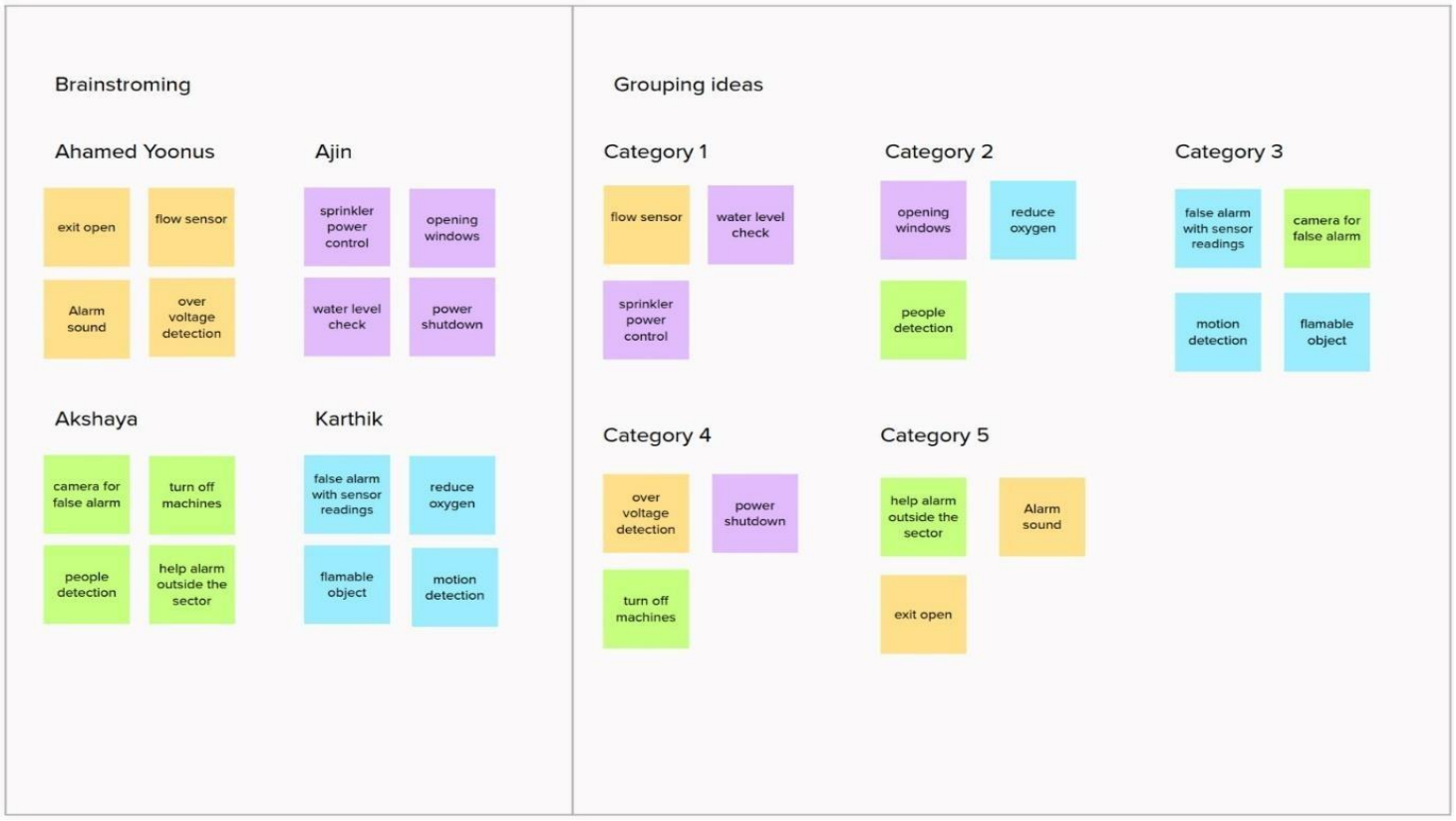
step 1: Team Gathering, Collaboration and Select the Problem Statement

Team was gathered in mural app for collaboration

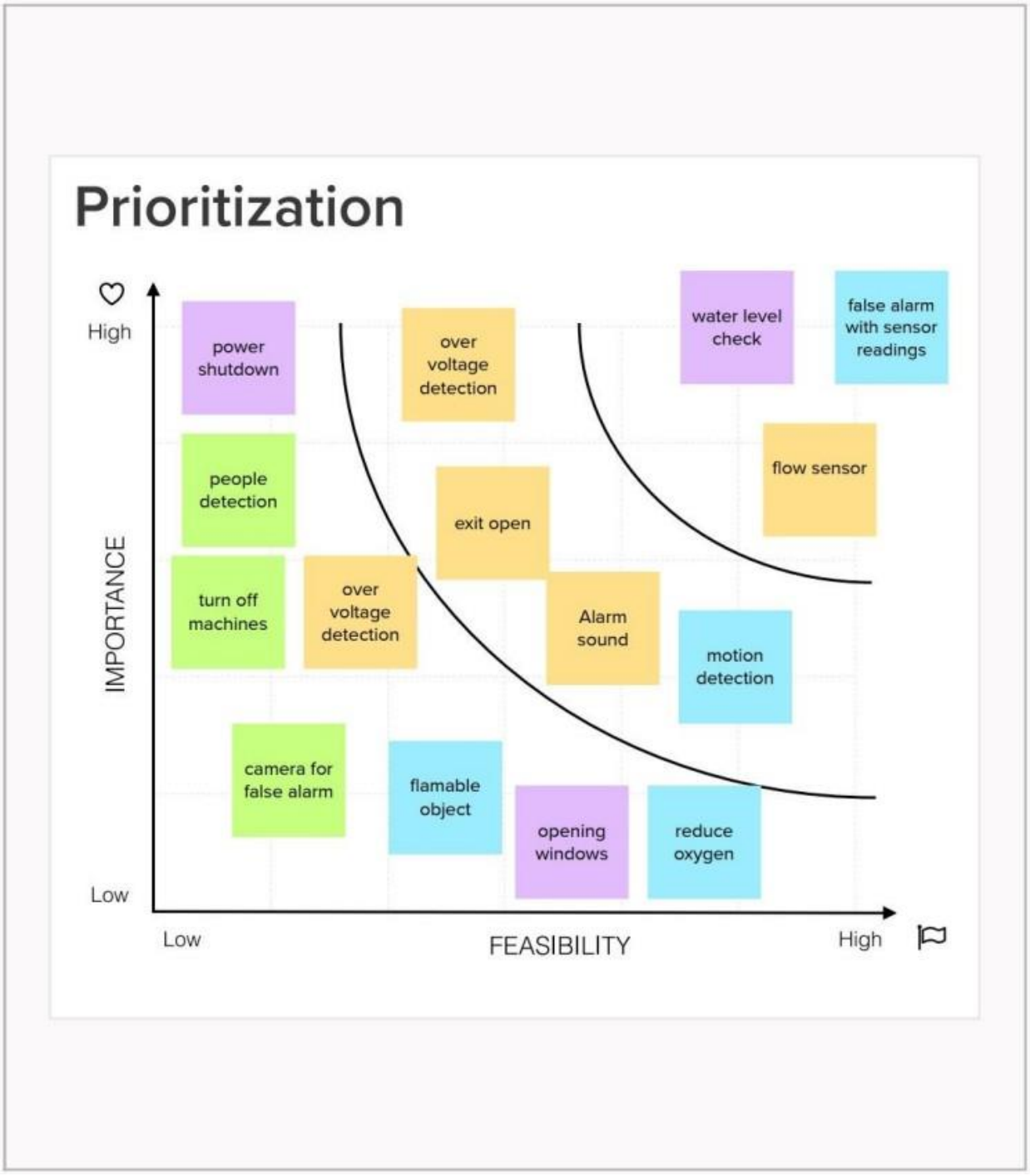
The team members are

- Vigneshwaran S
- Senthur Kumar B
- Vimalan S S
- Yogeshwaran S

step 2:Brainstorm, Idea Listing and Grouping



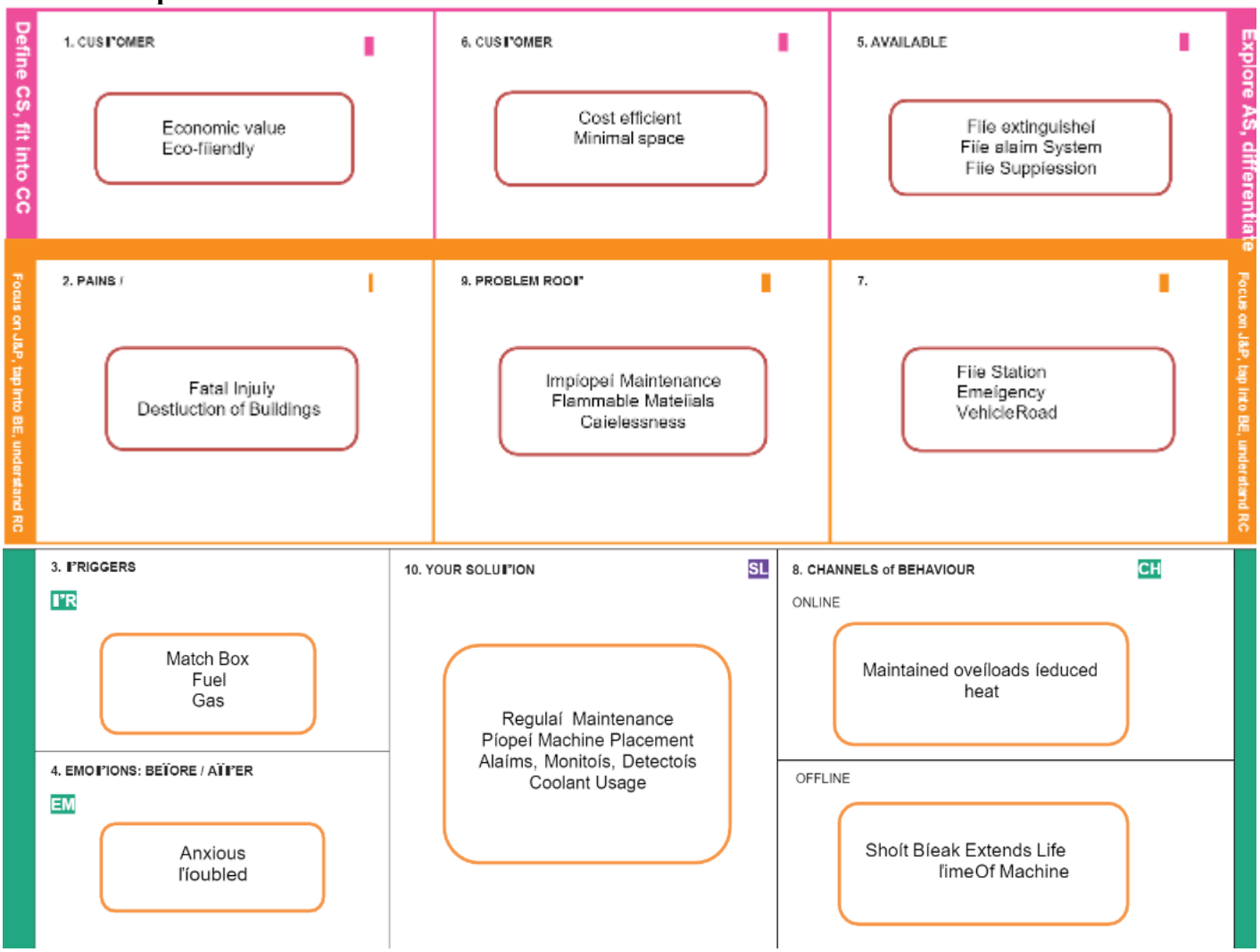
step3:Idea Prioritization



3.3 Proposed Solution

S. No	Parameter	Description
1	Problem Statement (Problem to be solved)	To enhance the security of the industries that use fire management a warning system to minimise the destruction of lives and property
2	Idea / Solution description	The goal is to find smoke and high temperatures. Additionally, the temperature drops by preserving the air's humidity while putting out fires in the event of an accident
3	Novelty / Uniqueness	detects the fire even before it begins. Simple administration and effective workflow
4	Social Impact / Customer Satisfaction	Industry workers should put in fearless hours. Substantially reduce on the destruction. To alert everyone, if there is some prudence.
5	Business Model (Revenue Model)	This technology can be used in any environment to detect fires, notify people, and decrease the loss. This system is used in the safety management system to make the most precise predictions.
6	Scalability of the Solution	The size of this system comes up short. Since it needs little time for management, it is simple to keep up with. The system's price is fair.

3.4 Proposed solution fit



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	Device configuration	New IoT device is created in the cloud The device is configured with the new cloud device
FR-2	Admin dashboard/admin panel	Data from sensors shown in pictorial form Controls are given in the button format
FR-3	Internet connectivity	Make sure fully-fledged internet connectivity is required for smooth communication between device and cloud
FR-4	SMS API	A external SMS API is required

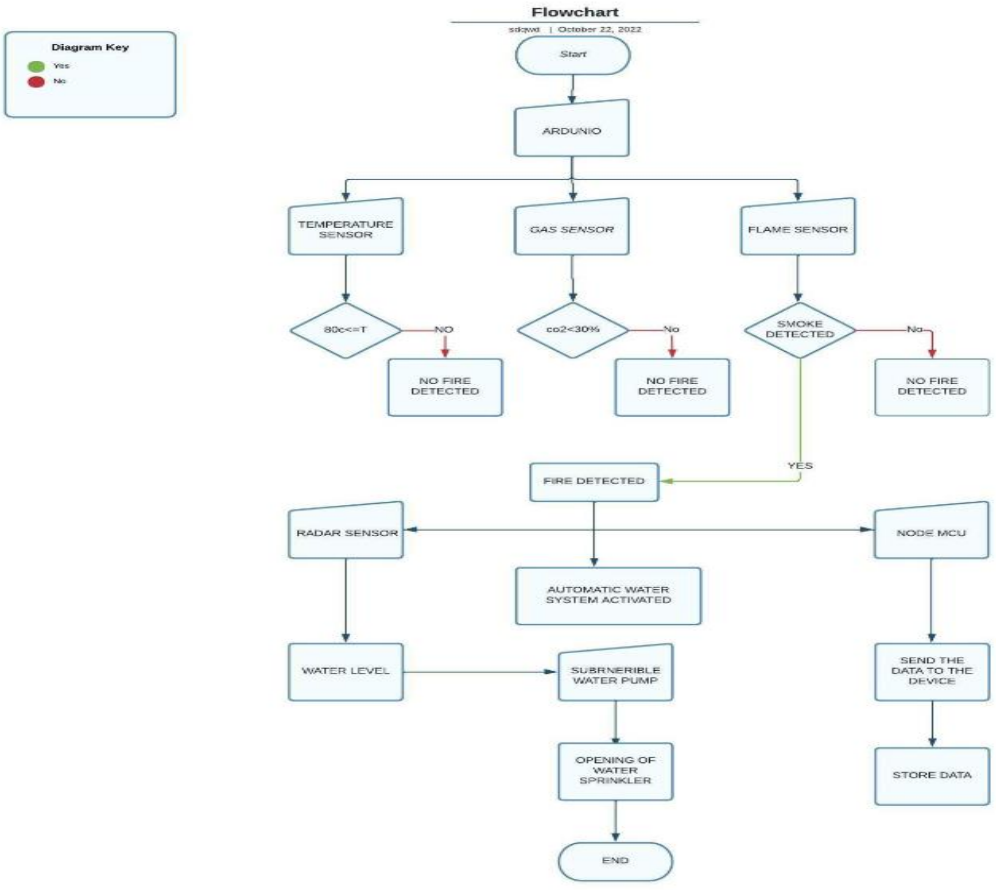
4.2 Non Functional Requirements

- 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**	The dashboard can be used via a web browser It gives an abstract view in an easy-to-use form.
NFR-2	**Security**	As the data is sent through HTTPS the data is encrypted, so it is safe.
NFR-3	**Reliability**	The system is completely reliable as long as the internet and power is reliable
NFR-4	**Performance**	Only the data input and basic checking is done in smart device other heavy tasks are done in cloud.
NFR-5	**Availability**	The entire system is available for your service and for configuration .
NFR-6	**Scalability**	The smart system is scalable,we can add any number of devices as long as the IBM IoT platform supports it.

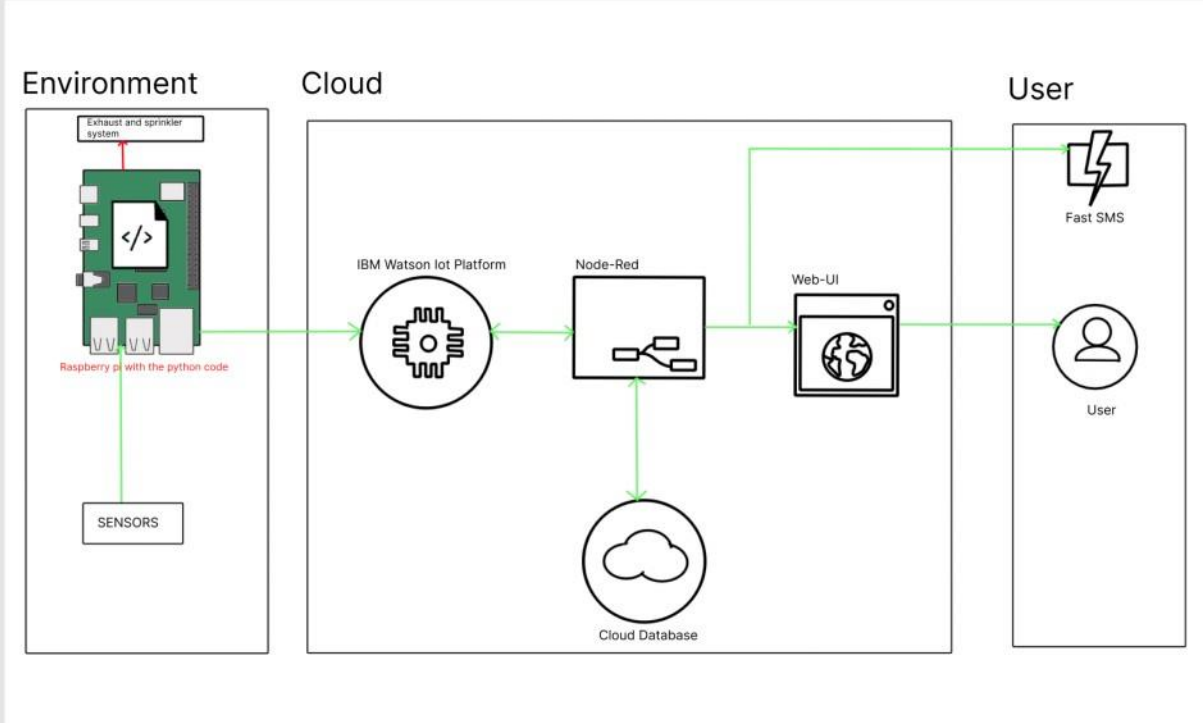
5. Project Design

5.1 Dataflow Diagram

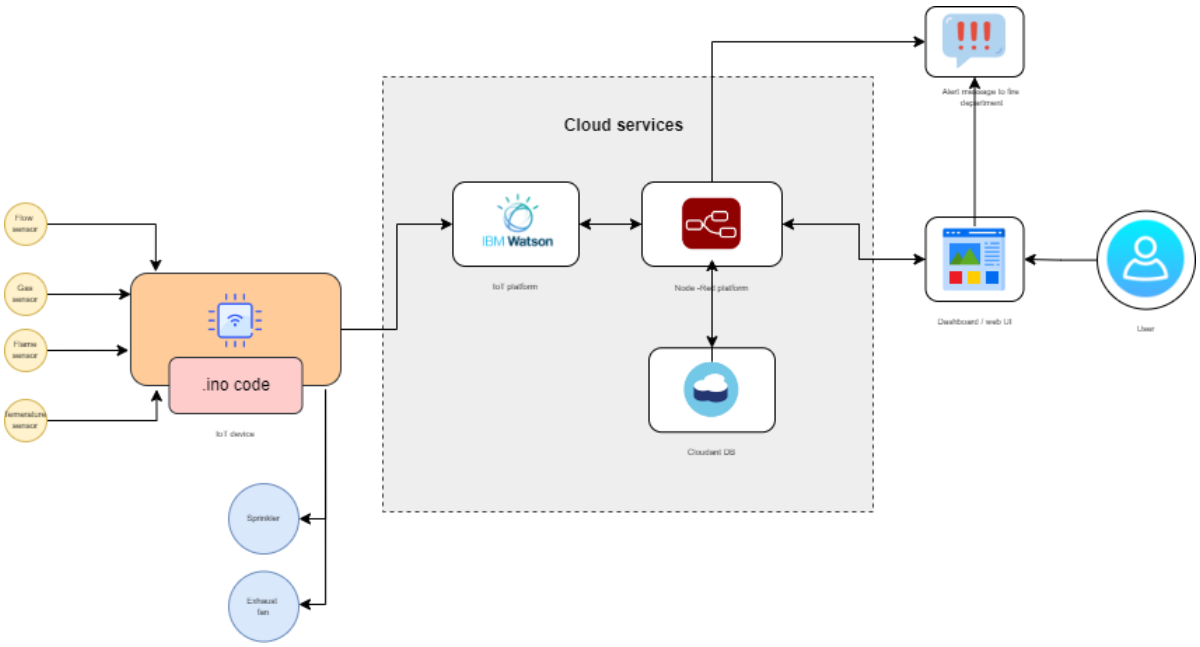


5.2 Solution and Technical architecture

Solution Architecture



Technical Architecture



5.3 User stories

USER TYPE	FUNCTIONAL REQUIREMENT	USER STORY NUMBER	USER STORY/TASK	ACCEPTANCE CRITERIA	PRIORITY	RELEASE
CUSTOMER(MOBILE USER ,WEB USER,CARE EXECUTIVE,ADMINISTRATOR)	REGISTRATION	USN-1	IN ORDER TO REGISTER FOR THE APPLICATION , I MUST ENTER MY EMAIL ADDRESS, PASSWORD, AND CONFIRM MY PASSWORD.	I CAN ACCESS MY ACCOUNT/DASHBOARD	HIGH	SPRINT-1
		USN-2	ONCE I'VE REGISTERED FOR THE APPLICATION, I WILL RECEIVE CONFIRMATION EMAIL AS A USER.	I CAN RECEIVE CONFRIMATION EMAIL & CLICK CONFRIM	HIGH	SPRINT-1
	DASHBOARD	USN-3	I CAN SIGN UP FOR THE APPLICATION AS A USER VIA THE INTERNET.	I CAN REGISTER & ACCESS THE DASHBOARD WITH INTERNET LOGIN	LOW	SPRINT-2
		USN-4	I CAN APPLICATION USER REGISTRATION FOR THE APPLICATION VIA EMAIL	I CAN CONFRIM THE REGISTREATION IN GMAIL	MEDIUM	SPRINT-1
	LOGIN	USN-5	IN ORDER TO ACCESS THE APPLICATION, I MUST ENTER MY EMAIL ADDRESS AND PASSWORD.	I CAN LOGIN WITH MYID AND PASSWORD	HIGH	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	Vigneshwaran S
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in case of fire and gas leakage.	3	Medium	Yogeshwaran S
Sprint-2	Sending collected data to the IBM Watson	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	Vimalan S 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	Senthur kumar B
Sprint-3	Storing of sensor data	USN-5	Storing in Cloud ant database.	2	Medium	Vigneshwaran S
	Node red	USN-6	Sending the data from the IBM Watson to the Node red.	3	High	Yogeshwaran S
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Vimalan S 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	Senthur kumar B
Sprint-3	Storing of sensor data	USN-5	Storing in Cloud ant database.	2	Medium	Vigneshwaran S
	Node red	USN-6	Sending the data from the IBM Watson to the Node red.	3	High	Yogeshwaran S
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Vimalan S S

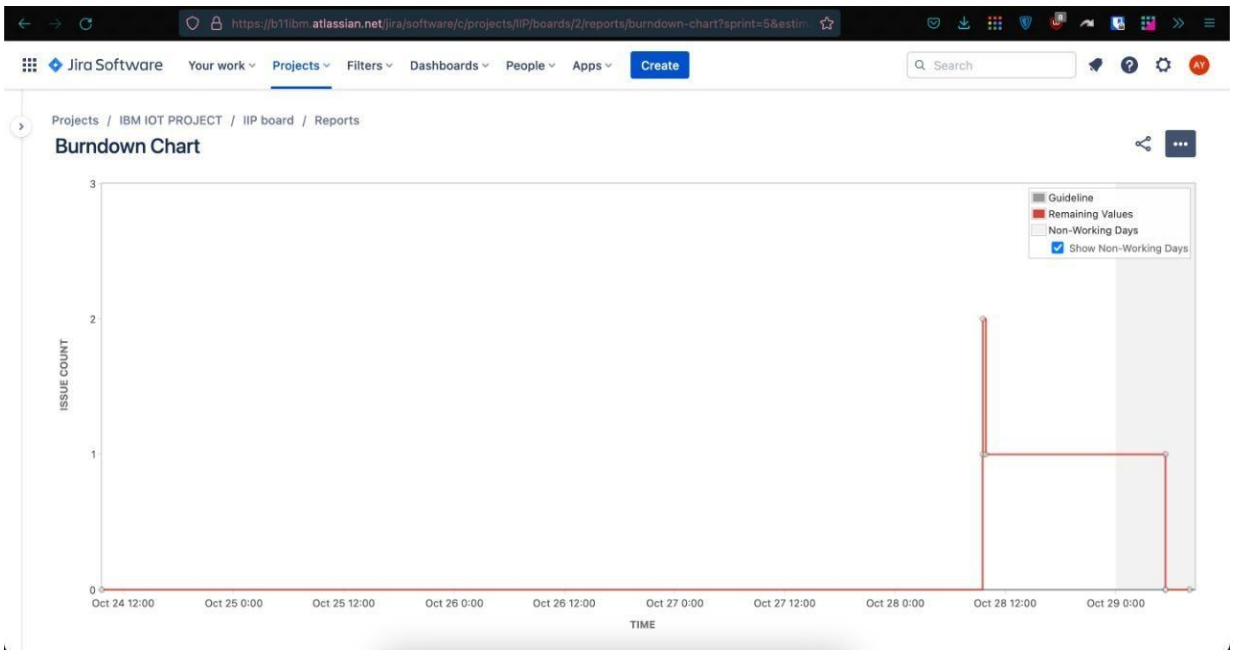
Sprint-4	Fast SMS Service	USN-8	Use Fast SMS to Send alert message once the parameters like temperature, flame and gas sensor readings goes beyond the threshold value	3	High	Senthur kumar B
	Turn ON/OFF the actuators	USN-9	Users can turn off the Exhaust fan as well as the sprinkler system If needed in that Situation.	2	Medium	Vigneshwaran S
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	Yogeshwaran S

6.2 Sprint delivery schedule

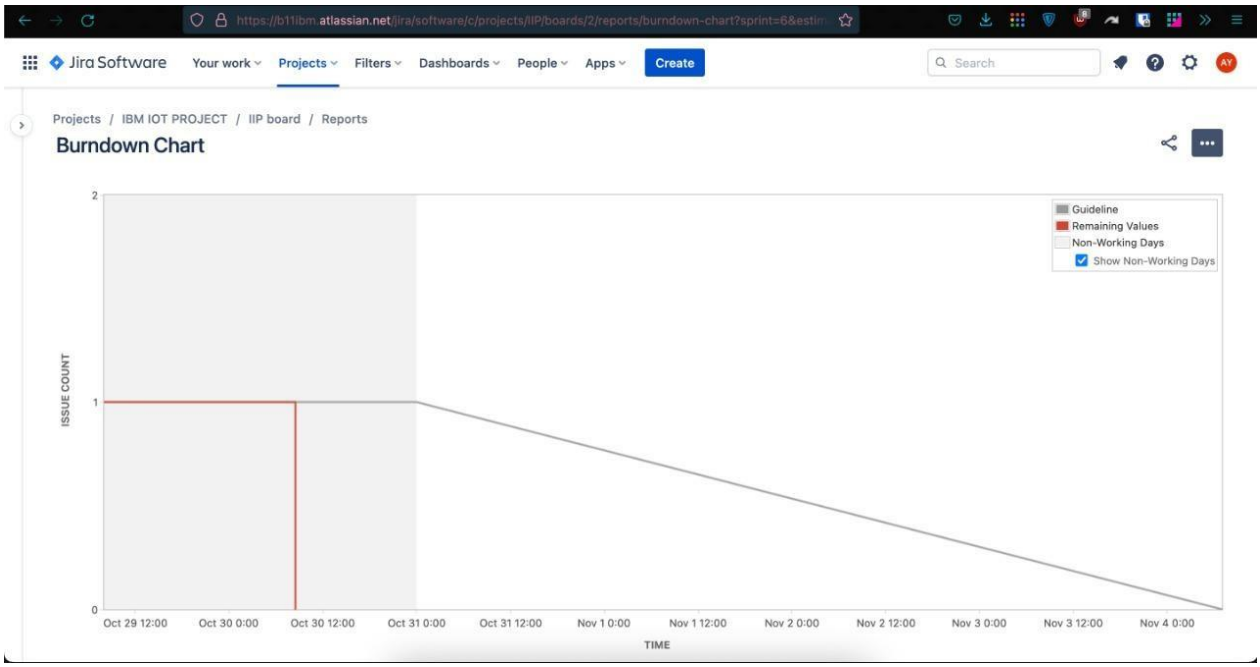
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	6	6 Days	31 Oct 2022	05 Nov 2022	6	05 Nov 2022
Sprint-3	6	6 Days	07 Nov 2022	12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

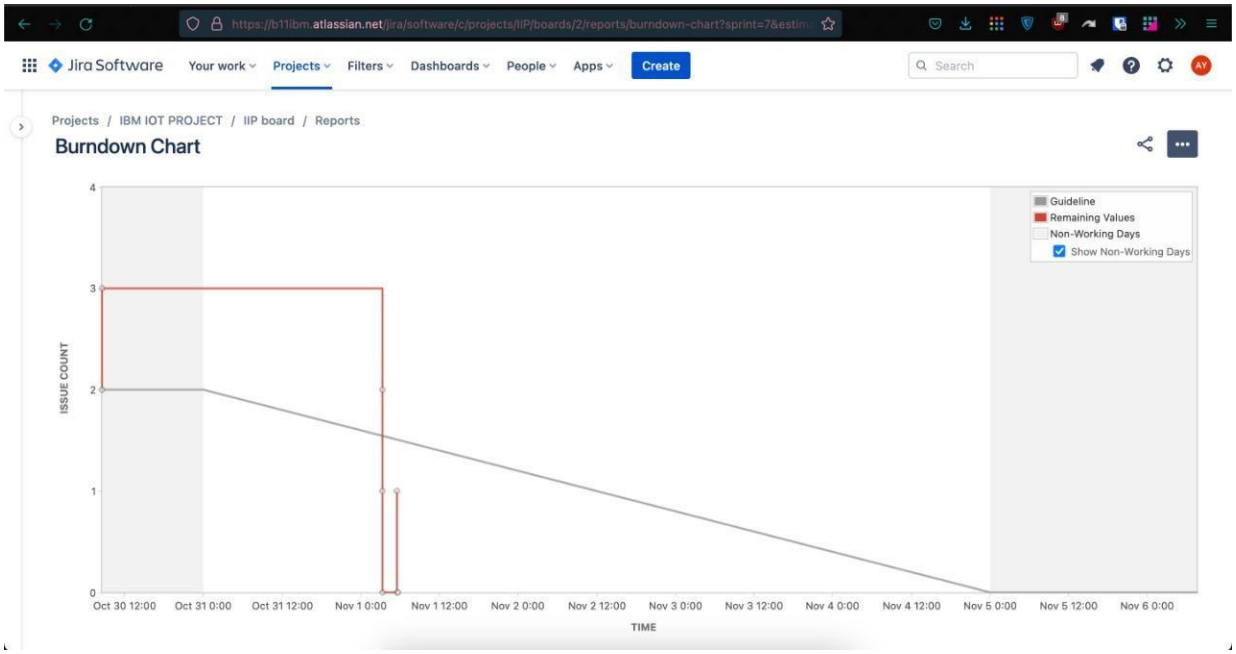
Sprint 1



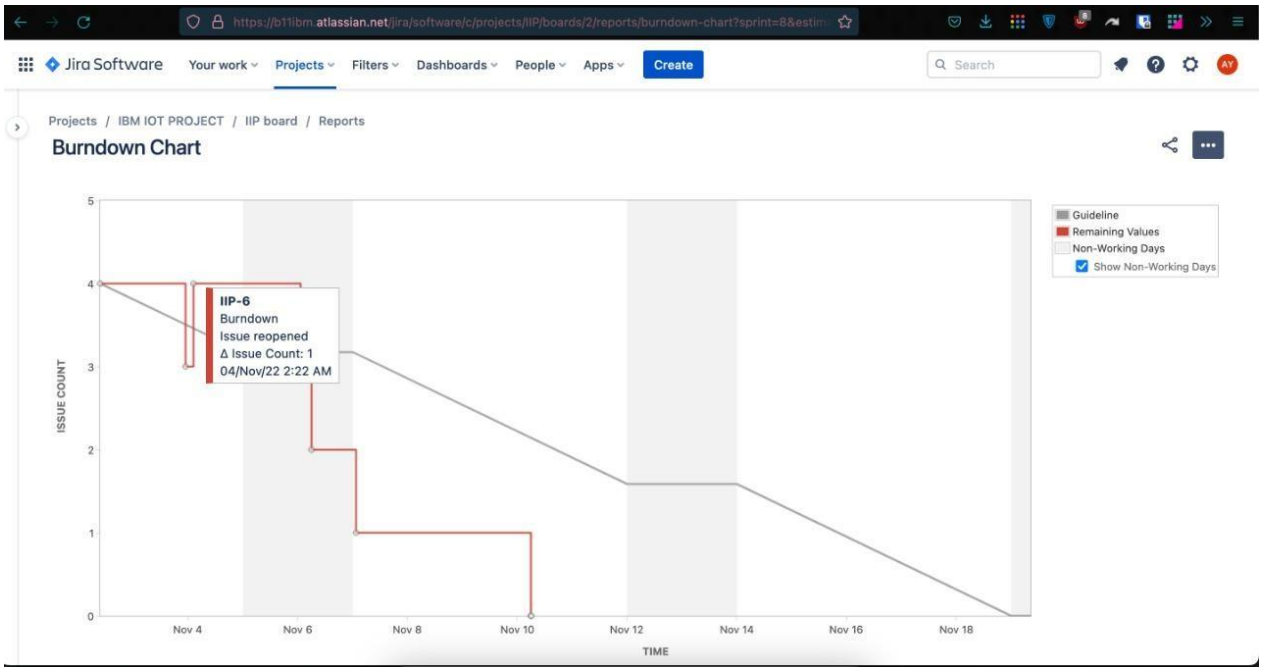
Sprint 2



Sprint 3



Sprint 4



7. Coding and Solutioning

Feature 1 : False alarm checking

```
#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 senor is heated up for acurate
readings
  /*
    - Here random values for readings and stdout were used to show the
      working  of the devices as physical or simulated devices are not
      available.
  */
}

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";
  }

  //Obivously 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  efinition y 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

8. TESTING

8.1 Testcases

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

8.2UAT

Defect analysis

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

Test case analysis

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

9. Results

9.1 performance metrics

CPU usage

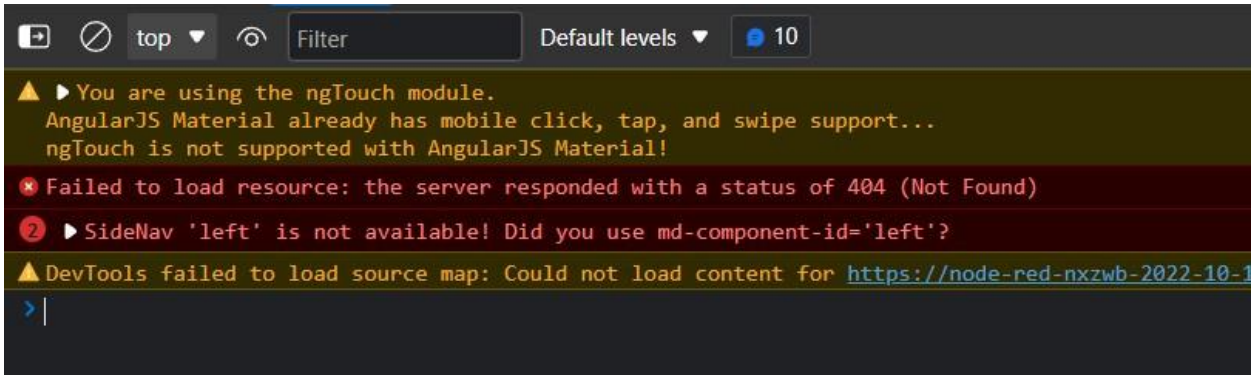
The micro version of c++ is make the best use of the CPU. For every loop the program runs in O(1) time, neglecting the network and communication. The program sleeps for every 1 second for better communication with MQTT. As the program takes O(1) time and the compiler optimizes the program during compilation there is less CPU load for each cycle. The upcoming instructions are on the stack memory, so they can be popped after execution.

Memory usage :

The sensor values , networking data are stored in sram of the ESP32 . It’s a lot of data because ESP32 has only 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

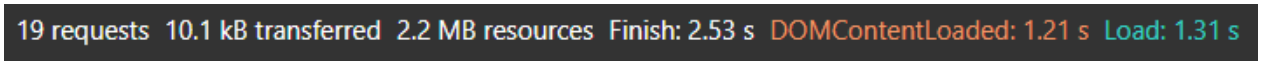
Error rates :

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

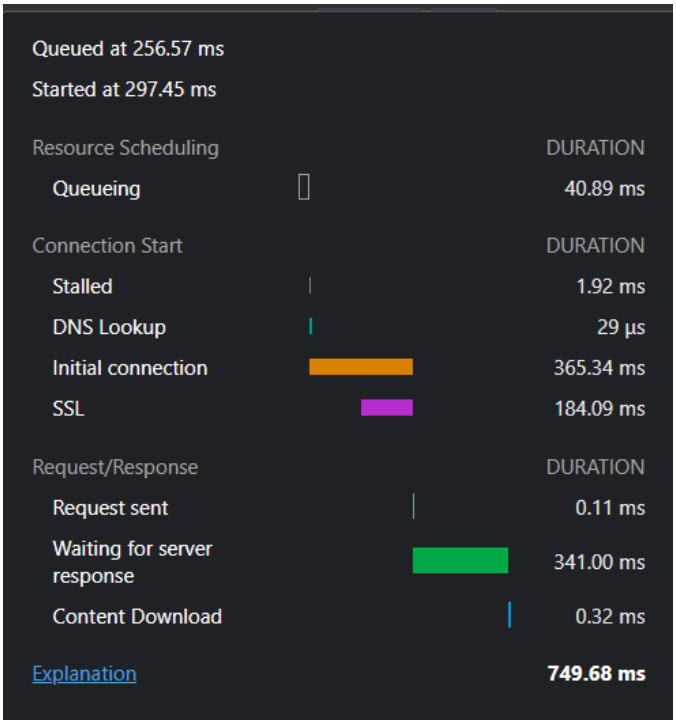


Latency and Response Time :

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



The server also responses quickly . The average time of response is respectable

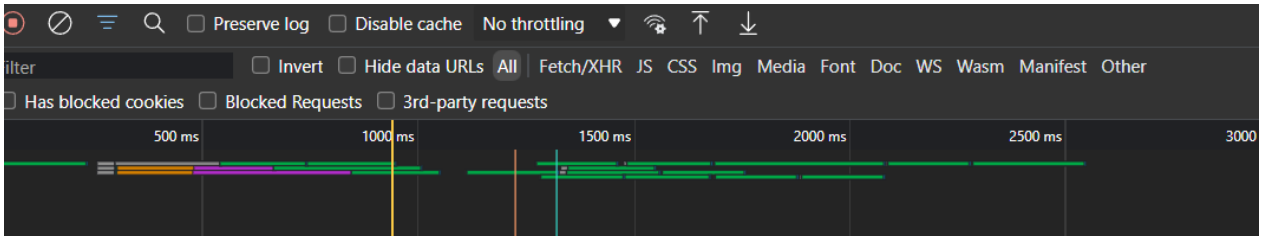


For the data sent from the IoT device (considering the sleep of one second from the IoT), the response is much quicker .We can easily see the delay caused by the sleep function

The average time is well over optimal value

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

= 1302.5



Garbage collection:

In the server-side garbage collection is done by the Node framework.In the IoT device , c++ does not have any garbage collection features . 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

- Active monitoring for gas leakage and fire breakout
- Automatic alerting of admin as well as fire authorities using SMS
- Automatically turning on/off sprinkler as well as exhaust fan

- Authentication is required to turn on/off of sprinkler and exhaust fan as well as sending SMS alert manually

Disadvantages

- Always need to connect with the internet **[Only to Send the SMS alert]**
- If the physical device is damaged the entire operation is collapsed
- Need large database since many data is stored in cloud database every second

11. CONCLUSION

So in conclusion our problem premise is solved using Iot devices by creating a smart management system that solves many inherent problems in the traditional fire management system like actively monitoring for fire breakouts as well as gas leakage and sending SMS alerts to the admin as well as to the fire authorities .

12. FUTURE SCOPE

The existing devices can be modified to work in different specialized environment as well as scale to house use to big labs[Since fire accidents can cause major loss in human lives in homes to big industries] as well as it can be used in public places , vehicles.

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

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>
#include <WiFi.h>
#include <PubSubClient.h>

#define ORG "w9vo0w"
#define DEVICE_TYPE "Vicky"
#define DEVICE_ID "5678"
#define TOKEN "123456789"

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);

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 senor is heated up for acurate readings
  /*
    - Here random values for readings and stdout were used to show the
      working  of the devices as physical or simulated devices are not
      available.
  */

   wifiConnect();
   mqttConnect();
}

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";
}

//Obivously the output.It is like json format 'cause it will help us for future sprints
String payload = "{\"senor_values\":{";
payload+="\"gas_ppm\":";
payload+=gas_ppm;
payload+=",";
```

```
  payload+="\"temperature\":";
  payload+=(int)temperature;
  payload+=",";
  payload+="\"flame\":";
  payload+=flame;
  payload+=",";
  payload+="\"flow\":";
  payload+=flow;
  payload+="},";
  payload+="\"output\":{";
  payload+="\"is_exhaust_fan_on\":"+String((is_exhaust_fan_on)?"true":"false")+",";
  payload+="\"is_sprinkler_on\":"+String((is_sprinkler_on)?"true":"false")+"";
  payload+="},";
  payload+="\"messages\":{";
  payload+="\"fire_status\":\""+flame_status+"\",";
  payload+="\"flow_status\":\""+sprinkler_status+"\",";
  payload+="\"accident_status\":\""+accident_status+"\"";
  payload+="}";
  payload+="}";
  //Serial.println(payload);


  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-22145-1659805853**