INDUSTRY SPECIFIC - INTELLIGENT FIRE MANAGEMENT SYSTEM

Team ID: PNT2022TMID16087

Team Members

VEERADHARSHINI R(927619BEC4230)

SHUBIKSHA P (927619BEC4194)

SRIRAGAVI R (927619BEC4207)

VANDHANA M (927619BEC4226)

INTRODUCTION

PROJECT OVREVIEW:

Industrial-based intelligent fire management system that can control security and safety of the industry intelligently within the minimum time and the design of a system using wireless sensor networks, fire alarm sensor, and human detecting sensor to address the problems with existing disaster emergency response systems in times of fire hazard. The system has decentralized control that can intelligently guide evacuees based on the detection of humans for removing them from industry to minimize the loss of human life and industrial assist. The existing system was able to secure the industry but not within enough time as the system was designed using various sensors but not as a single unit to address the problems in times of fire or any other. Each sensor was connected to the system separately and function individually which makes the system slow. The modified system can secure the industry intelligently within minimum time as the system is designed using different sensors as a single unit to address the problems in times of fire or any other. A fire alarm system warns people when smoke, fire, carbon monoxide or other fire-related or general notification emergencies are detected. These alarms may be activated automatically from smoke detectors and heat detectors or may also be activated via manual fire alarm activation devices such as manual call points or pull stations. Alarms can be either motorized bells or wall mountable sounders or horns. They can also be speaker strobes which sound an alarm, followed by a voice evacuation message which warns people inside the building not to use the elevators. Fire alarm sounders can be set to certain frequencies and different tones including low, medium, and high, depending on the country and manufacturer of the device. Most fire alarm systems in Europe sound like a siren with alternating frequencies. Fire alarm electronic devices are known as horns in the United States and Canada and can be either continuous or set to different codes. Fire alarm warning devices can also be set to different volume levels.

PURPOSE:

A fire alarm system is the detection system designed to protect the building population in the event of fires and gas leaks. The system continuously monitors the building areas for fire and harmful gases and provides early warning so that proper mitigating actions can be taken in a timely manner. Growing awareness about fire protection systems and benefits of better technology for fire prevention are driving growth in the fire alarm market. Users are looking for the latest technology, and the market is seeing growing investment. For facility owners and managers, one of the most effective ways to enhance overall protection could be through the integration of the fire alarm system with other building systems, such as mass notification systems and security systems. By integrating these systems together, users can improve oversight and management of multiple systems from a single point of control. Integrated systems can be more cost-effective, as multiple systems can be managed from a central station and fewer resources are required. In case of emergency, users may also be able to respond in a more efficient manner. In particular, integration with a mass notification system can be extremely helpful. In an event of fire or other emergency, safety personal can effectively relay critical announcements to building occupants. Similarly, integration with a fire alarm system can also help building occupants in safe evacuations, as systems can be programmed to open locks automatically in the event of fire.

.

LITERATURE SURVEY

EXISTING PROBLEM:

The existing system Recently, it has sometimes been impossible for fire-fighting personnel to access the site of a fire, even as the fire causes tremendous property damage and loss of human life, due to hightemperatures or the presence of explosive materials. In such environments, firefighting robots can be useful for extinguishing a fire. Thus ire-fighting robots are operated where firefighters cannot work. Besides that, firefighting robots can be used for protecting firefighters from extreme danger in the petrochemical chemical dangerous product, and toxicity or exploder fire accidents. Therefore, it also canreduce the human injury from a fire burning

REFERENCES:

S.no	Year	Researcher	Title	Methodology	Remarks
01	2021	Ravindra Koggalage , Manjula Welihinda, Hasitha Nuwan3	GSM-BASED SMART FIRE AND HIGH-TEMPERATURE DETECTION SYSTEM	MQ5 Gas sensor,GSM, Arduino ATmega328	Indicating firethrough SMS Alert using Gas Sensor
02	2019	J Olivares-Mercado	Early Fire Detection on Video Using LBPand Spread Ascending of Smoke	LBP (Local Binary Pixel)	Thus, the smoke detection method using an image from video sequences has been identified

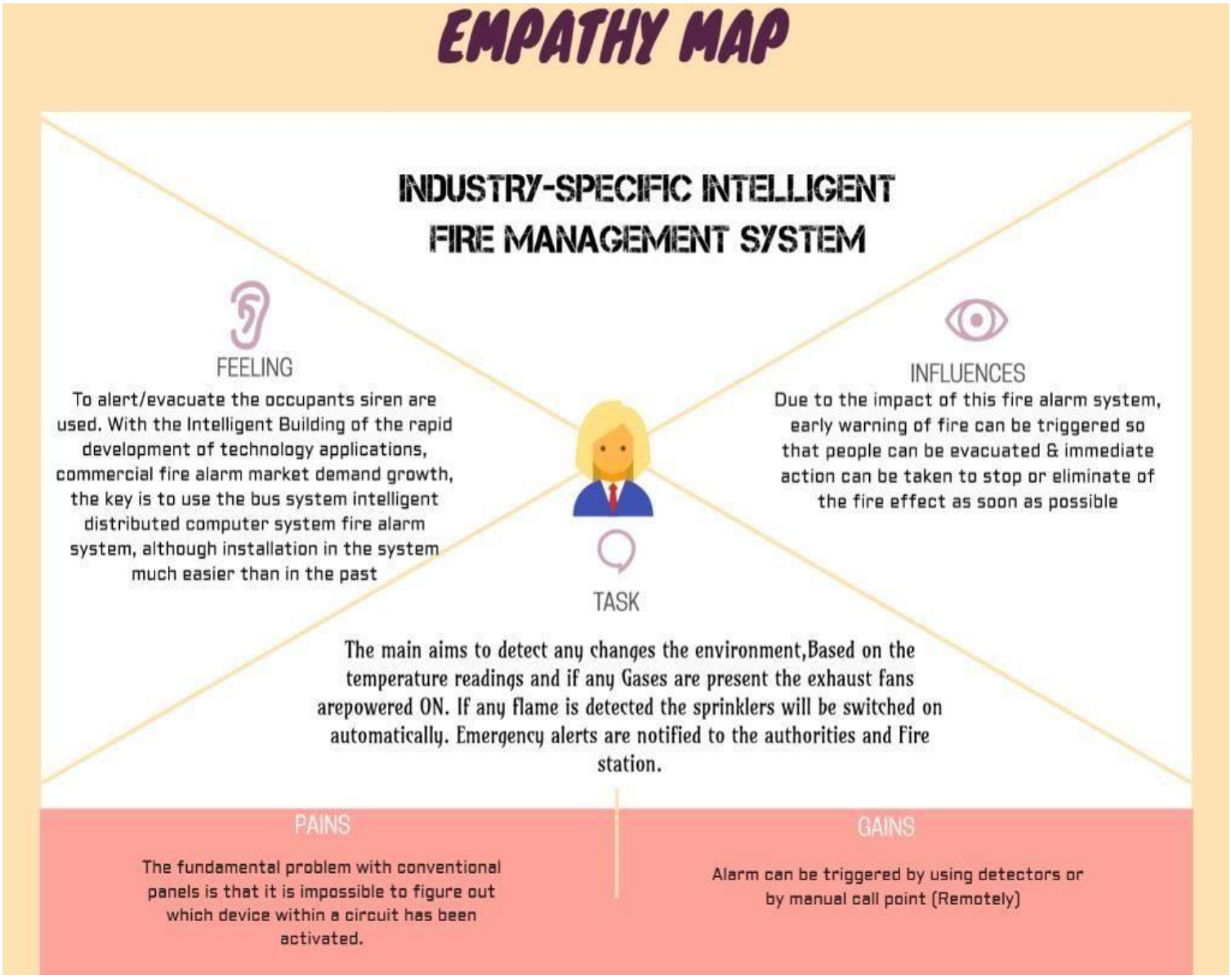
03	2019	Park J.H., Lee S., YunS., Kim H., Kim W.T.	Dependable Fire Detection Systemwith Multifunctiona l Artificial Intelligence Framework	Adaptive fuzzy algorithms and Multifunctio nalAI framework - Fire Detection System (MAI- FDS)	The highest accuracy of Fire Detection has been achieved usingMAI

PROBLEM STATEMENT DEFINITION STATEMENT:

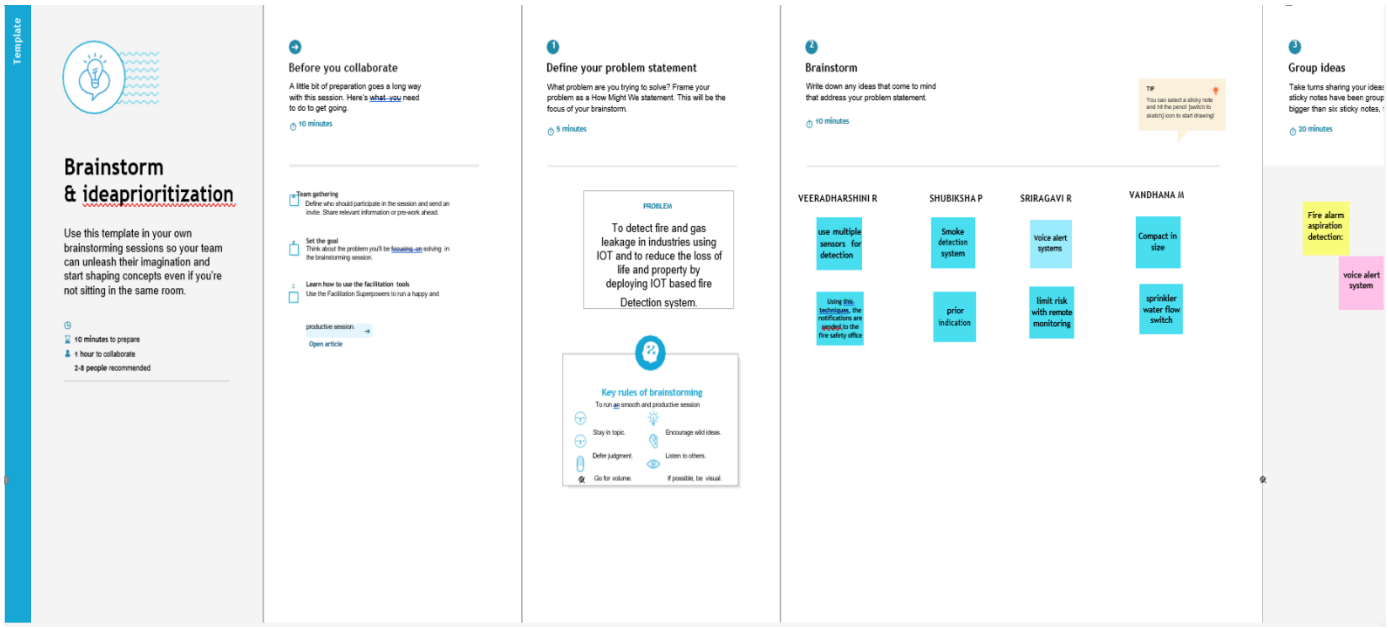
The fire management system in houses and industries is not very reliable, efficient, cost effective, and does not have any advanced processing and does not have any features like automatic alert system for admin and authorities and in many buildings, there are using older fire safety system that cannot even activate the sprinkler system and all of them do not communicate with each other properly to prevent false alarms. Safety is a crucial consideration in the design of residential and commercial buildings to safeguard against the loss of life and damage to property. The existing fire alarm system on market nowadays is too complex in terms of its design and structure. Since the system is too complex, it needs regular maintenance to be carried out to make sure the system operates well. Meanwhile, when the maintenance is being done to the existing system, it could raise the cost of the system

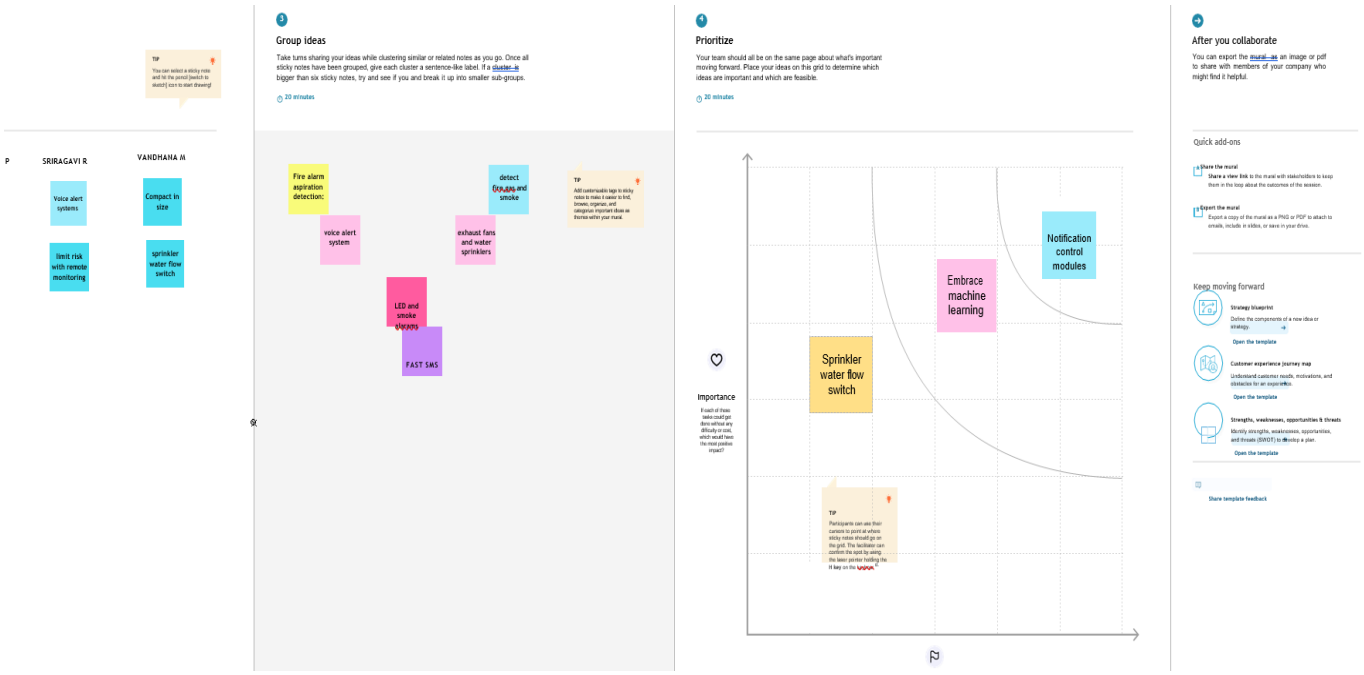
IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:



IDEATION AND BRAINSTORMING:



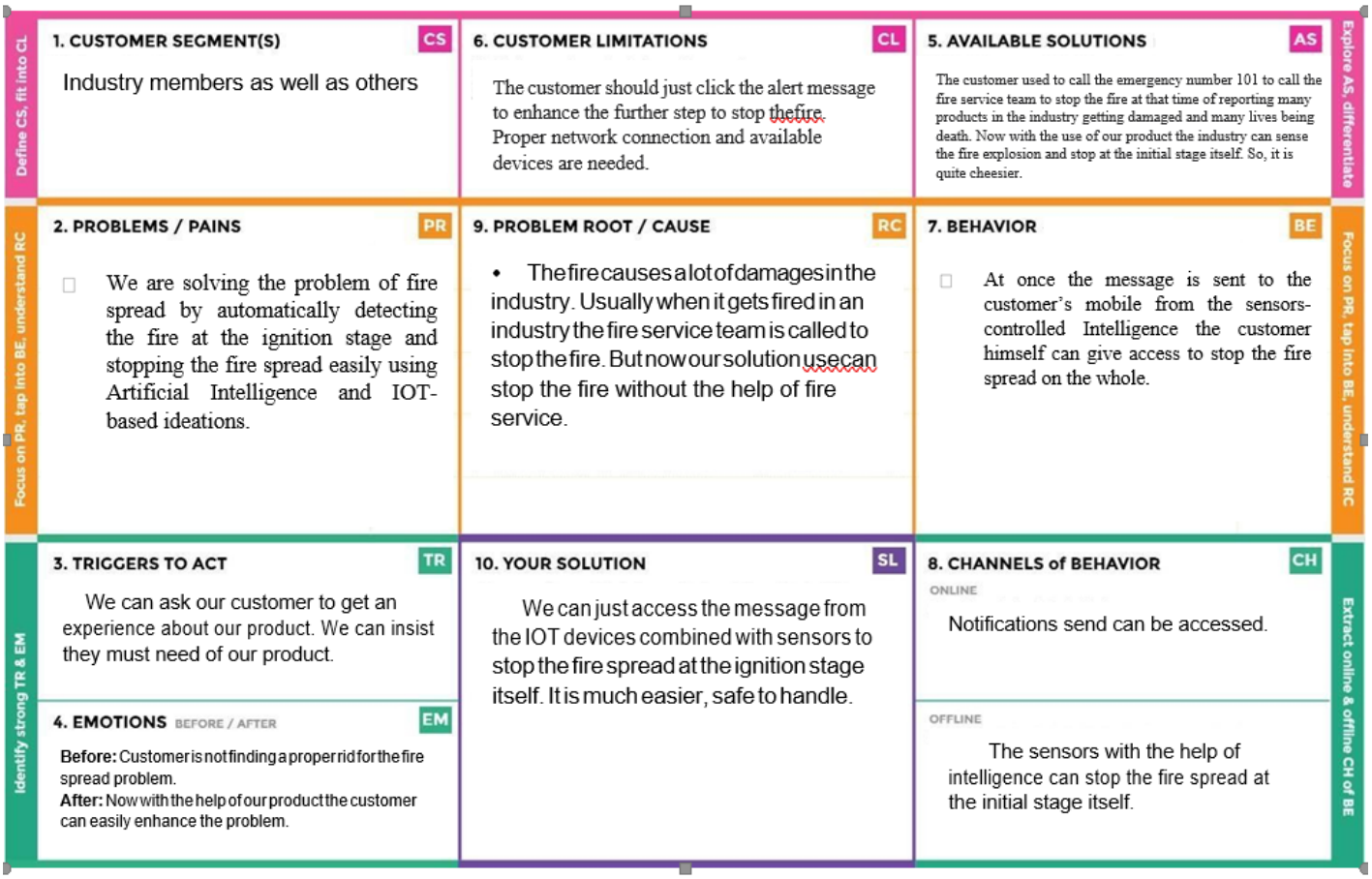


PROPOSED SOLUTION:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To improve the safety management system in industries. Improving the safety management system against fire incidents in industries.
2.	Idea / Solution description	To implement fire safety management in the industry based on IOT using Arduino Uno board with fire detection and fire extinguisher system. And using some sensors (Humidity sensor, Flame sensor, smoke sensor) with GPS tracking system.
3.	Novelty / Uniqueness	An integrated system of temperature monitoring, gas monitoring, and fire detection automatically fire extinguisher with the accusation of information about locations and response through SMS notification and call.
4.	Social Impact / Customer Satisfaction	▪ It early prevents accident costs by fire in industries. ▪ Nearby locations so maximum extend more accurate reliability. ▪ Compatibility design integrated system.

5.	Business Model (Revenue Model)	
6.	Scalability of the Solution	▪ This project can be used more efficiently with the accurate information required. ▪ Easy operability and maintenance. ▪ Required low time for maintaining ▪ Cost is a reasonable value.

PROBLEM SOLUTION FIT:



REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENT:

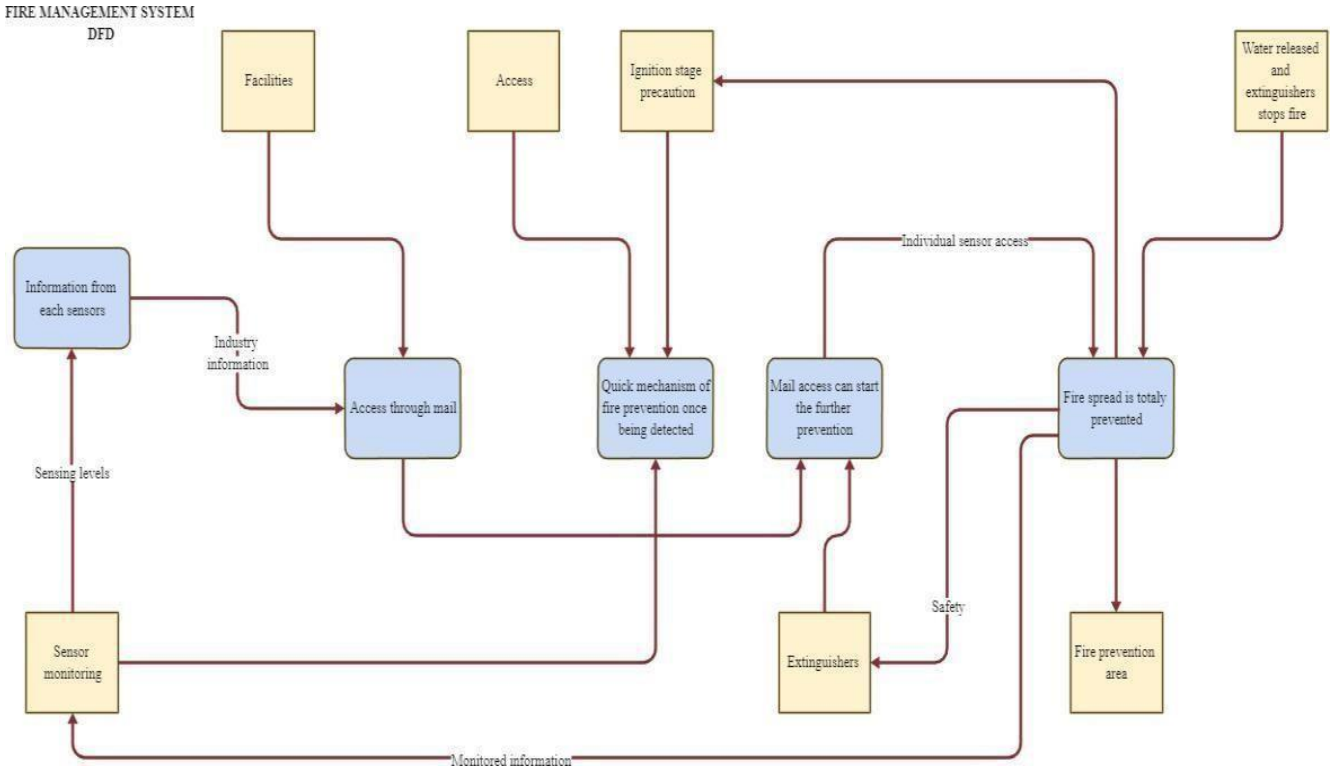
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through website or application Registration through Social medias (like Instagram, Facebook) Registration through LinkedIn
FR-2	User Confirmation	Verification via Email Verification via OTP
FR-3	User Login	Login through website or App using the respective username and password
FR-4	User Access	Allows the app requirement
FR-5	User Guide	Guides the basic steps of using the application
FR-6	User Upload	User should be able to send the data
FR-7	User Solution	Data report should be generated and delivered to user for per every 24 hours
FR-8	User Data Sync	API interface to increase to invoice system

NON FUNCTINAL REQUIREMENT:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Usability requirements can consider language barriers and localization tasks. Usability can be assessed from the below functions. Efficiency ofuse. Low perceived workload. Easy and simple UI.
NFR-2	**Security**	Access permissions for the particular system information may only be changed by the system's data administrator.
NFR-3	**Reliability**	The database update process must roll back allrelated updates when any update fails.
NFR-4	**Performance**	The front-page load time must be no more than 2 seconds for users that access the website using an VoLTE mobile connection.
NFR-5	**Availability**	New module deployment mustn't impact front page, product pages, and check out pages availability and mustn't take longer than one hour.The rest of the pages that may experience problems must display a notification with a timershowing when the system is going to be up again.
NFR-6	**Scalability**	We can increase scalability by adding memory, servers, or disk space. On the other hand, we can compress data, use optimizing algorithms. The website attendance limit must be scalable enoughto support 500,000 users at a time.

PROJECT DESIGN

DATA FLOW DIAGRAM:



SOLUTION AND TECHNICAL ARCHITECTURE:

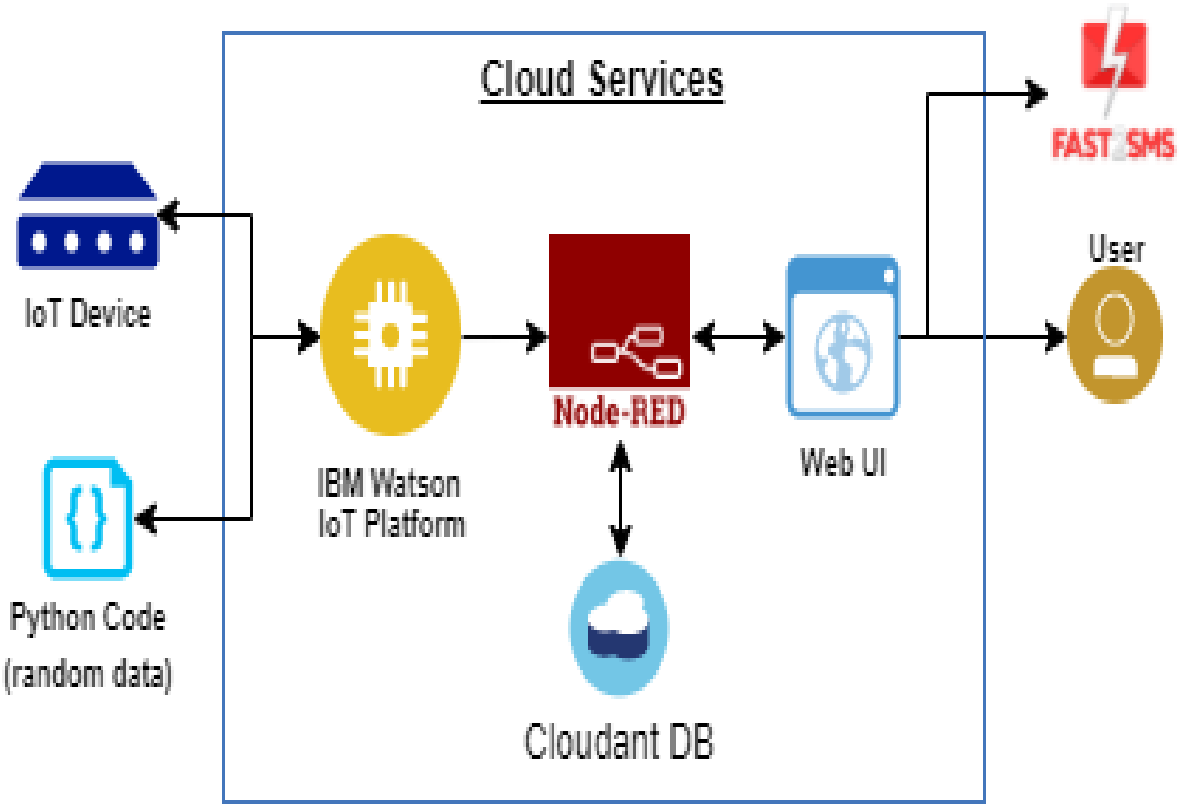


Table-1 : Components & Technologies:

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 cloudant service,ibm node-red
3.	Application Logic-2	Develop python script to publish and subscribe toIBM 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 receivethe sensors information and to react accordingly	Web UI,python
8.	External API-1	Using this IBM fire management API we can trackthe 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 ofsprinklers to web UI	IBM Sensors
10.	Machine Learning Model	Using this we can derive the object recognitionmodel	Object Recognition Model
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / CloudCloud Server Configuration	IBM cloudant, IBM IoT Platform

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	MIT app Inventor	MIT License
2.	Security Implementations	IBM Services	Encryptions, IBM Controls
3.	Scalable Architecture	sensor-IoT Cloud based architecture	cloud computing and AI
4.	Availability	Mobile, laptop, desktop	MIT app
5.	Performance	Detects the Fire, gas leak,temperature	sensors

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	As a user, I can register for the application by entering my mail, password, and confirming my password	I can access my account/ dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Dashboard	USN-3	As a user, I can register for the application through internet	I can register & access the dashboard with Internet login	Low	Sprint-2
			As a user,	I can		

		USN-4	I can register for the application through Gmail	confirm the registration in Gmail	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login with my id and password	High	Sprint-1

PROJECT PLANNINGAND SCHEDULING

SPRINT PLANNINGAND 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	Veeradharshini
	Operating	USN-2	Turning on the exhaust fan as well asthe fire sprinkler system in cause of fire and gas leakage.	3	Medium	Shubiksha
Sprint-2	Sending collected data tothe IBM Watson platform	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	Veeradharshini, Sriragavi

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Registration	USN-4	Entering my email and password toverify authenticatio n process.	3	High	Veeradharshini, Shubiksha
Sprint-3	Storing of sensordata	USN-5	Storing in Cloud ant database.	2	Medium	Vandhana, Sriragavi
	Node red	USN-6	Sending the data from the IBMWatson to the Node red.	3	High	Shubiksha
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	1	Low	Veeradharsh ini, Vandhana
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	Veeradharshini, Shubiksha
	Turn ON/OFF theactuators	USN-9	User can turn off the Exhaust fan as well as the sprinkler system If need in that Situation.	2	Medium	Veeradharshini, Shubiksha

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	Sriragavi, Vandhana

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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

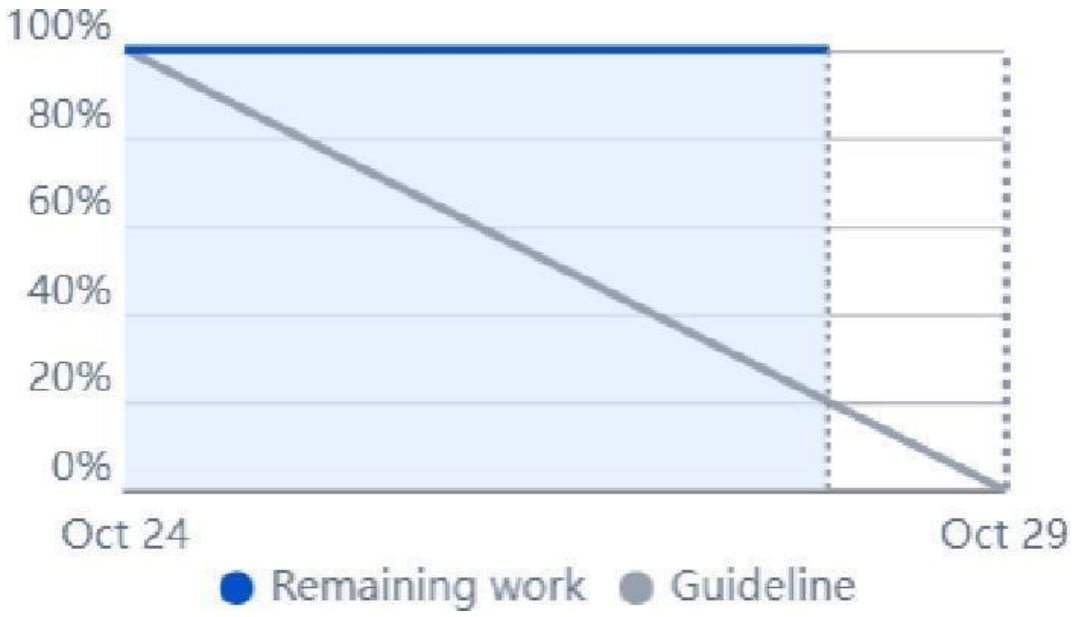
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's averagevelocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progressover time.



CODING AND SOLUTIONING

FEATURE 1:

CODING AND SOLUTIONING

FEATURE-1

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

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

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

Explanation

➤ This set of code checks the false alarms and sets the current status

➤ It also handles the permission management of whether a device will work or not

FEATURES 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+"\"";
20 payload += "," "\"sprinkstatus\":";
payload += "\""+sprinkstatus+"\"";
```

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

EXPLANATION

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

TESTING

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Sensor_001	Functional	Microcontroller	Sensor data is properly taken	The connections to the circuit	1.Open the simulation in wokwi.	Random values generated.	Get the values and print it in the	Working as	Pass		N		Akshaya
Sensor_002	Functional	Microcontroller	Sensor data is parsed as json	The microcontroller should	1.Open the simulation in wokwi.	Random values generated.	Get the values and print it in the	Working as	Pass		N		Karthick
Work_001	Functional	Microcontroller	To check for fake alarm	The sensor values are taken	1.Simulate the device(do a practical	Random values generated.	Accident status is properly updated	Working as	Pass		N		Ajin
Work_002	Functional	Microcontroller and	The data should be sent to IBM	The device setup is completed	1.Start the simulation in wokwi.	Random values generated.	The values are shown in recent	Working as	Pass		N		Akshaya
Work_003	Functional	Node-red	The data should be sent to	The necessary packages	1.Login to node-red editor	values got from the iot	The debug area should show the	Working as	Pass		N		Yeshas
Work_004	Functional	Node-red	Verify that the json data is parsed	A configured node-red with	1.Login to node-red editor	values got from the iot	the debug menu shows the output	Working as	Pass		N		Yeshas
Database_001	Storage	Cloudant	The received data is stored in database in a key value pair	The node red is connected with cloudant node	1.login to cloudant dashboard. 2.create new database. 3. connect the database with node red and then give the database name in required field	values got from the iot device	After sending the data the data is stored in cloudant	Working as expected	Pass		N		Karthick
SMS_001	API	sms API	The sms is sent when there is fire alert	The node red should be configured to send a post request	1.Simulate the fire in the simulator(if real hardware is used real fire is used) 2.or click the sent alert button in	"Fire alert at xyz industries Hurry" And the trigger inputs	sms receiving to the given phonenum	Working as expected	Pass		N		Ajin
Work_005	Functional	UI	Even at times of emergency sometimes manual control is required	the dashboard interaction elements is connected to the node red	1. in the dashboard enter the correct pin 2.click the action to be done	The action by user	manual command system works only	Working as expected	Pass		N		yeshas
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 96	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

UAT

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

RESULTS:

CPU Usage:

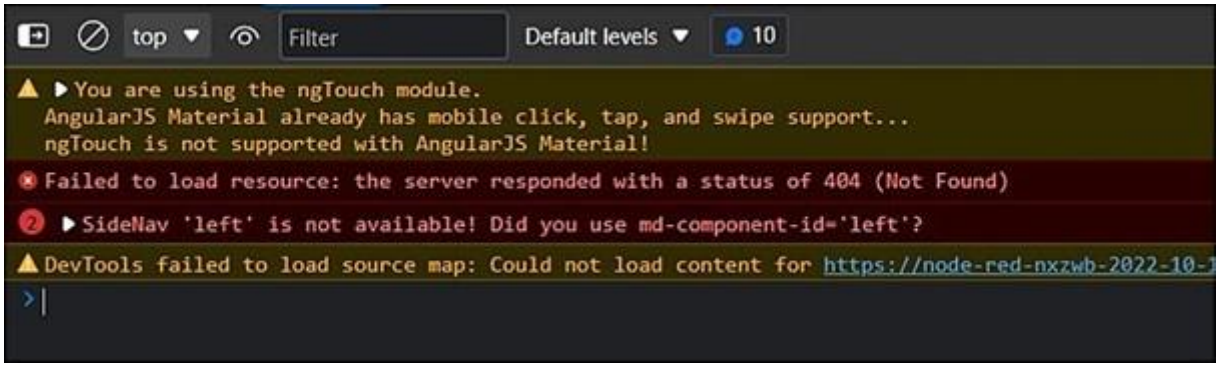
The micro version of C++ makes the most efficient use of the CPU. The program runs in O(1) time for each loop, ignoring the network and communication. To improve communication with MQTT and the program sleeps every 1 second. Because the program runs in O(1) time and the compiler optimizes it during compilation,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

Memory Usage:

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

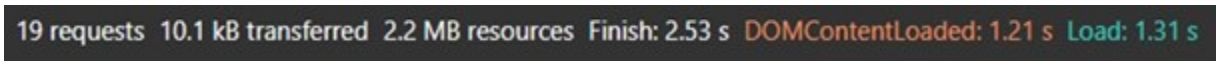
Error Rates:

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

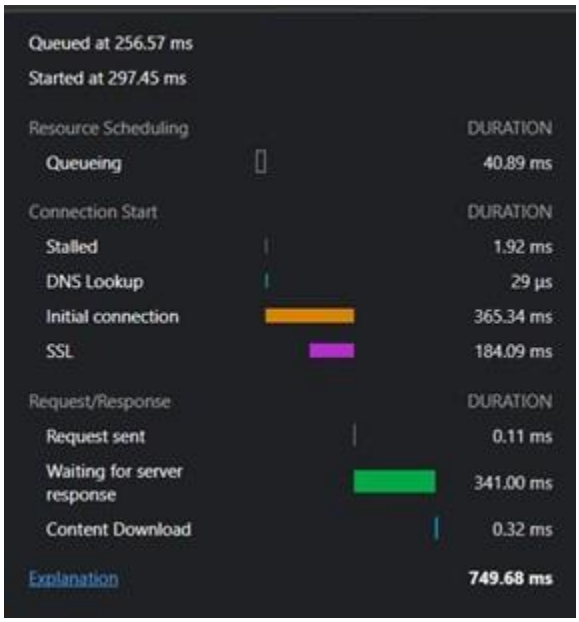


Latency and Response Time:

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



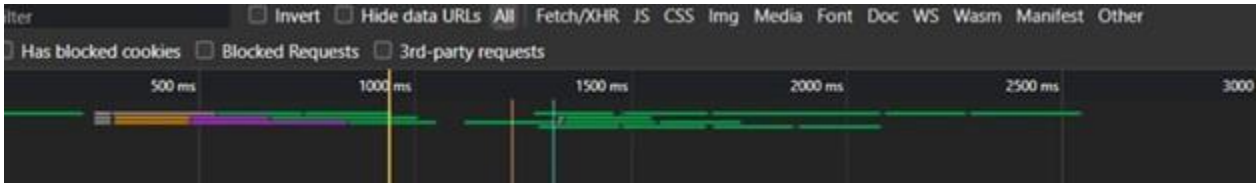
In addition, the server respondsquickly. The average response time is acceptable.



For the data sent from the IoT Device (considering the sleep of one second from the IoT), the response is muchfaster. We can 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:

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

ADVANTAGES AND DISADVANTAGES:

ADVANTAGE:

- ➤ Active detection of gas leaks and fire outbreaks
- ➤ SMS alerting of administrators and fire authorities
- ➤ Turning on/off sprinklers and exhaust fans automatically
- ➤ Tomanually turn on/off sprinklers and exhaust fans, as well as send SMS alerts, authentication isrequired
- ➤ It detectsfalse fire outbreaks automatically, reducing unnecessary panic
- ➤ We can confirm that the sprinkler system is functioning properly by using flow sensors
- ➤ A dashboard can display the status of any device

DISADVANTAGE:

- ➤ Always requirean internet connection [only to send the SMS alert]
- ➤ If the physical device fails, the entire operation fails.

➤ Because a large amount of data is stored in the cloud database every second, a large database isrequired

CONCLUSION:

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

FUTURE SCOPE:

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

APPENDIX

ESP32 - Microcontroller:

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

- Memory: 320 KiB SRAM
- CPU: Tensilica Xtensa LX6 Microprocessor @ 160 or 240 MHz
- Power: 3.3 VDC
- Manufacturer: Espress if Systems
- Predecessor: ESP8266

SENSORS:

DHT22 - Temperature & Humidity Sensor:

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

Flow Sensors:

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

MQ5 - Gas Sensor:

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

Flame Sensor:

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

Source Code:

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

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

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

#define ORG "fvs923"

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

String data3 = "";
String
accidentstatus ="";
```

```
String sprinkstatus
= ""; float temp
=0;
bool isfanon = false;
 bool issprinkon = false;
bool cansprinkoperate
= true;bool
canfanoperate = true;
bool cansentalert
= false;int gas = 0;
int flame
=0;
int flow =
0;
long int cooldown= 600;
char server[] = ORG
".messaging.internetofthings.ibmcloud.com";char
publishTopic[] = "iot-2/evt/data/fmt/json";
char subscribetopic[] = "iot-
2/cmd/command/fmt/String";char
authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
//---------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the
```

```
predefined client id by passingparameter like server id,portand
wificredential
void setup()// configureing the ESP32
{
Serial.begin(115200); dht.begin();
//if real gas sensor is used make sure the senor is heated up for acurate readings
/*
- Here random values for readings and stdout were used to show the
working of the devices as physical or simulated devices are not available.
*/
delay(10);
Serial.print
ln();
wificonnec
t();
mqttconnec
t();
}
void loop()
{
temp = dht.readTemperature();
//setting a random seed (only for random values not in real life scenarios)
srand(time(0));
//initial variable activities like declaring , assigning gas = rand()%400;
```

```
int flamereading = rand()%1024;

flame = map(flamereading,0,1024,0,1024);
int flow = ((rand()%100)>50?1:0); //find the accident status 'cause fake alert may be
caused by some mischiefactivities
if(temp < 45 ){ if(flame
> 650 ){
accidentstatus = "Need
Auditing";
if(canfanoperate)
isfanon =
true; else
isfanon =
false;
issprinkon =
false;
}
else if(flame <= 10){
accidentstatus = "nothing
happened";isfanon = false;
issprinkon = false;

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

if(cansprinkoperate)
issprinkon = true; else
```

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

}
else if(flame <= 100 && flame > 10)
{
if(cansprinkoperate)
issprinkon =
true; else
issprinkon =
false; isfanon =
false;
accidentstatus = "moderate";
}
}
else if(temp > 55)
{
 if(flame > 650)
{
gas = 500 + rand()%500;
accidentstatus =
"severe";
```

```
if(cansprinkoperate
) issprinkon = true;
else issprinkon =
false;
if(canfanoperate)
isfanon =
true; else
isfanon =
false;
}
else if(flame < 650 && flame > 400 )
{
 gas = 300 +
rand()%500;
accidentstatus =
"severe";
if(cansprinkoperate
) issprinkon = true;
else issprinkon =
false;
if(canfanoperate)
isfanon =
true;else
isfanon = false;
}

}

else {
```

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

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

}
else{
sprinkstatus = "not working";
}

}

else if(!issprinkon)
{
sprinkstatus = "ready";
}
else {
sprinkstatus = "something's wrong";
}
PublishData(temp,gas,flame,flow,isfanon,issprinkon);

//a cooldown period is set as the values and situations are random in real life
sceanarios the time can bereduced or neclected
if(accidentstatus=="severe" && cooldown >= 600)
{
 cooldown
= 0;
sendalert();
PublishData(temp,gas,flame,flow,isfanon,iss
```

```
 prinkon);cansentalert = false;
}
if(cooldown > 999999)
{
cooldown = 601;
}
delay(1000);
++cooldown;
if (!client.loop())
{
mqttconnect();
}


}
/*..... retrieving to
Cloud.  */
void PublishData(float temp, int gas ,int flame ,int flow,bool isfanon,bool issprinkon) {
mqttconnect();  //function call for connecting to ibm
/*
creating the String in in form JSon to update the data to ibm cloud
*/
String payload =
"{\"temp\":";payload
+= temp;
payload += ","
"\"gas\":";payload
+= gas;
```

```
payload += ","
"\"flame\":";payload
+= flame;
payload += "," "\"flow\":";
payload += ((flow)?"true":"false");
payload += "," "\"isfanon\":";
payload +=
((isfanon)?"true":"false");
payload += ","
"\"issprinkon\":"; payload +=
((issprinkon)?"true":"false");
payload += ","
"\"cansentalert\":";
payload +=
((cansentalert)?"true":"false");
payload += ","
"\"accidentstatus\":"; payload
+= "\""+accidentstatus+"\"";
payload += ","
"\"sprinkstatus\":"; payload +=
"\""+sprinkstatus+"\""; payload
+= "}";
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");    // if it sucessfully upload data on the cloud then it
will print publish ok in Serialmonitor or else it will print publish failed

} else {
```

```
Serial.println("Publish failed");


}
}
void mqttconnect() {
if (!client.connected())
{
Serial.print("Reconnecting
client to ");
Serial.println(server);
while (!!!client.connect(clientId, authMethod, token))
{
Serial.print(
".");
delay(500);
}

initManagedDe
vice();
Serial.println();
}

}

void wificonnect()        //function defination for wificonnect
Serial.println();
Serial.print("Connecting to
"); WiFi.begin("Wokwi-
GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED)
```

```
{
delay(100);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi
connected");
Serial.println("IP address:
");
Serial.println(WiFi.localIP()
);
```

```
void initManagedDevice() {
 if (client.subscribe(subscribetopic))
 {
 Serial.println((subscribetopi
 c)); Serial.println("subscribe
 to cmd OK");
 }
 else {
 Serial.println("subscribe to cmd FAILED");
 }
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
 {
 Serial.print("callback invoked
 for topic: ");
 Serial.println(subscribetopic);
 for (int i = 0; i < payloadLength; i++)
 {
 data3 += (char)payload[i];
 }
 Serial.println("data: "+
 data3); const char *s
 =(char*) data3.c_str();
 double pincode = 0;
 if(mjson_get_number(s, strlen(s), "$.pin", &pincode))
 {
 if(((int)pincode)==137153)
```

```
{
const char *buf; int len;
if (mjson_find(s, strlen(s), "$.command", &buf, &len))  // And print it

{

String
 command(buf,len);
 if(command=="\"cant
 fan\""){
//this works when there is gas sensor reads high value and if there should be a


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

else if(command=="\"cantsprink\"")
{
cansprinkoperate = !cansprinkoperate;
}else if(command=="\"sentalert\"")
```

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

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


}


}


}
data3="";
}
void resetcooldown()
{
cooldown = 0;
}
//sent alert request to node-red void
sendalert(){cansentalert = true;
cooldown = 0;
}
```

GitHub Link:
https://github.com/IBM-EPBL/IBM-Project-25420-1659963040

Demo Video:
https://drive.google.com/file/d/1JcNyJdD2fwlgT4IANDfyc7IQhzTADihI/view?usp=share_link