INDUSTRY-SPECIFIC INTELLIGENT FIRE MANAGEMENT SYSTEM

DOMAIN – Internet Of Things

IBM – NALAIYATHIRAN

PROJECT REPORT

TEAM MEMBERS:

TEAM ID: PNT2022TMID47920

SURYA PRAKASH R

RAJESH KUMAR R

MUKESH R

MUTHU SIVA SANKAR M

Title Page No.

1. INTRODUCTION **3**
- Project Overview 3
- Purpose 3

2. LITERATURE SURVEY **4**
- References 4
- Proposed Method 5
- Problem Statement Definition 6

3. IDEATION AND PROPOSED SOLUTION **6**
- Empathy Map Canvas 6
- Ideation and Brainstorming 7
- Proposed Solution 8
- Problem Solution Fit 9

4. REQUIREMENT ANALYSIS **10**
- Functional analysis 10
- Non-Functional analysis 10

5. PROJECT DESIGN **11**
- Data Flow diagrams 11
- Solution & Technical Architecture 12
- User Stories 13
- Milestones 13

6. PROJECT PLANNING & SCHEDULING **15**
- Technology stack 15
- Sprint Delivery Schedule 16

7. SPRINT DELIVERY **16**
- Sprint 1 17
- Sprint 2 23
- Sprint 3 27
- Sprint 4 36

8. RESULTS **39**

9. ADVANTAGES **40**

10. CONCLUSION **41**

11. FUTURE SCOPE **41**

12. APPENDIX **41**

Source Code 42

GitHub & Project Demo Link 44

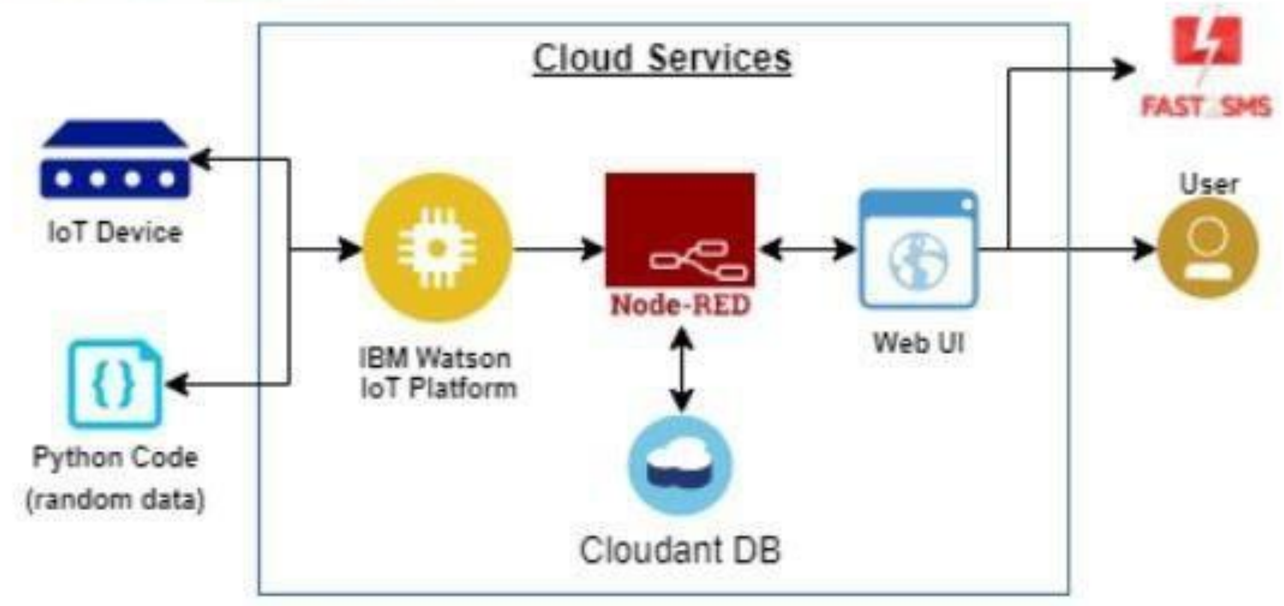
INTRODUCTION:

Fire and smoke kill more people every year than many other accidents. However, the rapid detection of fire and its control can save lives and property damage worth millions. The primary purpose of the fire management system is to provide an early warning of fire so that people can be evacuated and immediate actions can be taken to stop and eliminate its effect as soon as possible.

PROJECT OVERVIEW:

The smart fire management system includes a Gas sensor, Flame sensor and temperature sensors to detect any changes in the environment. 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 Fire station.

Technical Architecture:



PURPOSE:

1. An Intelligent solution to detect fire breakdown in industries at early stages so as to alert people working inside and taking necessary counter actuations to put off it easily without loss of human lives or material damage.

2. A better alternative from the existing technology, where use usually have alarms notifying about the breakdown.

3. Measures various parameters continuously which can be pushed to cloud and those can be accessed anytime to analyse.

4. Immediate support from nearby fire stations in case of uncontrollable fire which can be simultaneously notified to management authorities of the industry.

LITERATURE SURVEY

REFERENCES:

PAPER TITLE	AUTHOR	OBJECTIVE/OUTCOME
A Survey of Fire Safety Measures for Industry Safety Using IOT	N. Savitha; S. Malathi 2019	In the system the fire safety practices is going to implement for the fire crackers industry. In that the root cause for the fire is to be analyzed and prevent from the fire before it is triggered. Through this hazardous fire accidents can be avoided and many lives can be saved.
Design of Distributed Factory Fire Alarm Systems	Li Liu ;Yanke C I ; Haosong chen 2020	The Distributed plant fire alarm system can quickly detect the fire and issues an alarm to reduce the damage caused by the fire. The fire alarm system is a control system that integrates signal detection,transmission, processing and control .It mainly complete the basic function of Fire ,smoke and temperature module monitering fire.

A Microcontroller- based Fire Protection System for the Safety of Industries in Bangladesh	Md. Saiam Dept. of Electrical and Electronic Engineering, Khulna University of Engineering & Technology, Khulna, Bangladesh 2021	The affected area is also triggered by the fire extinguishing equipment. At the same time, it also notifies the manager and the nearby fire station via SMS. This paper presents a simulation and practical arrangement of the system to demonstrate how it can be implemented as a fire prevention equipment.
Safety Robot for Flammable Gas and Fire Detection using Multisensor Technology	Sandeep Prabhakaran; Mathan N	In case of fire accidents, the robot alerts the workstation and sends a mail to the firefighting department with the location read from the GPS module. As the robot works as an autonomous system, it does not need to be controlled remotely. Hence this robot is based on the line following mechanism, it is quite easy to install and can cover a large area efficiently.
Computer Vision Based Industrial and Forest Fire Detection Using Support Vector Machine (SVM)	Md. Abdur Rahman; Sayed Ta++++nimun Hasan; Mohammed Abdul Kader 2022	The proposed strategy works on a very large dataset of fire videos that have been collected both in real-life situations and from the internet. This SVM pipeline model shows the maximum accuracy is 93.33%. The system can fulfill the precision and detect faster real-time fire detection. It's forest and industrial application will aid in the early detection of fires, as well as emergency management, and so immensely contribute to loss prevention.

PROPOSED METHOD:

The fire management system can be used to assessing and controlling the fire risks.

1) If the temperature readings cross the threshold

exhaust fans are powered ON.

2) If flame is detected sprinkles will be switched ON.

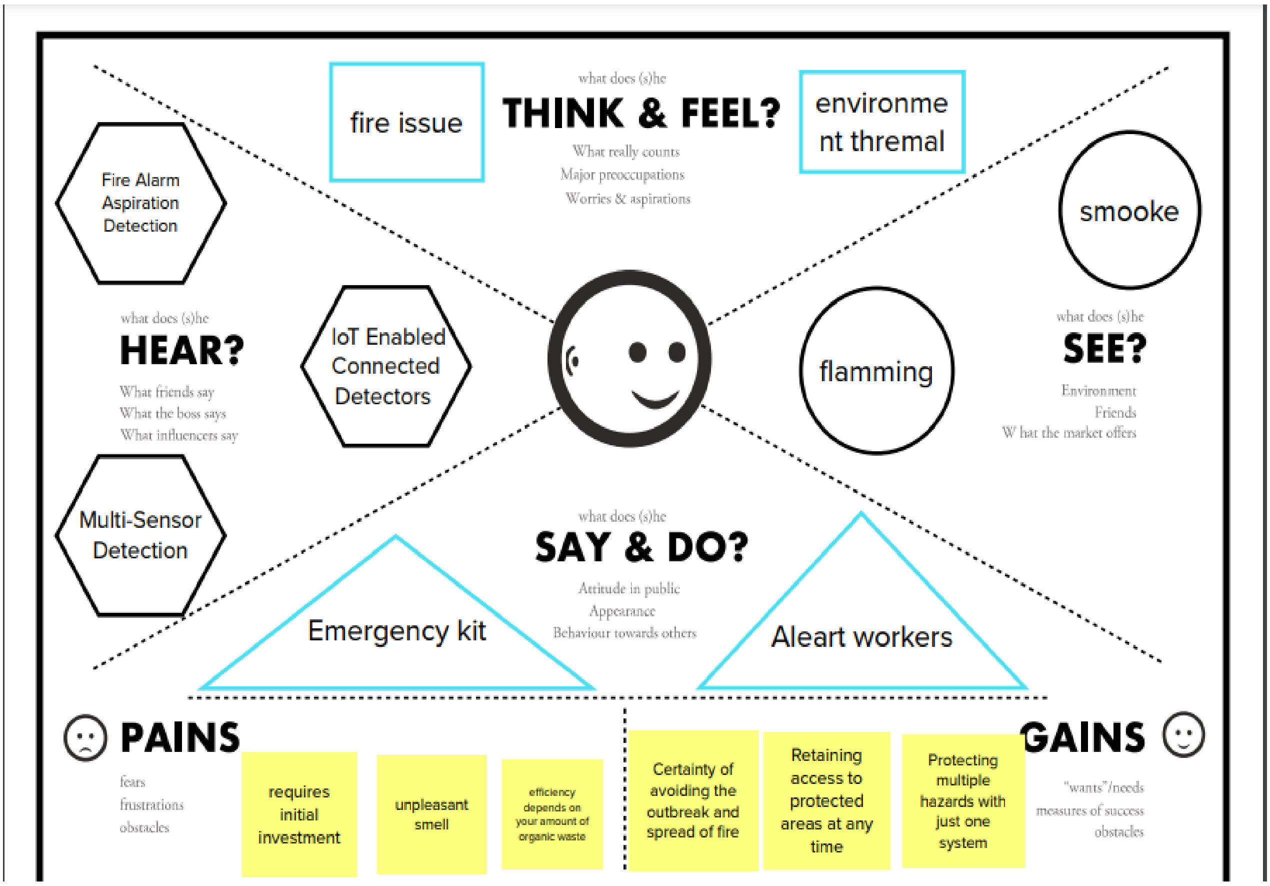
3)Alarms will be turned ON to alert the employees.

4)Message will be sent along with the location to fire station incase if the flame is uncontrollable.

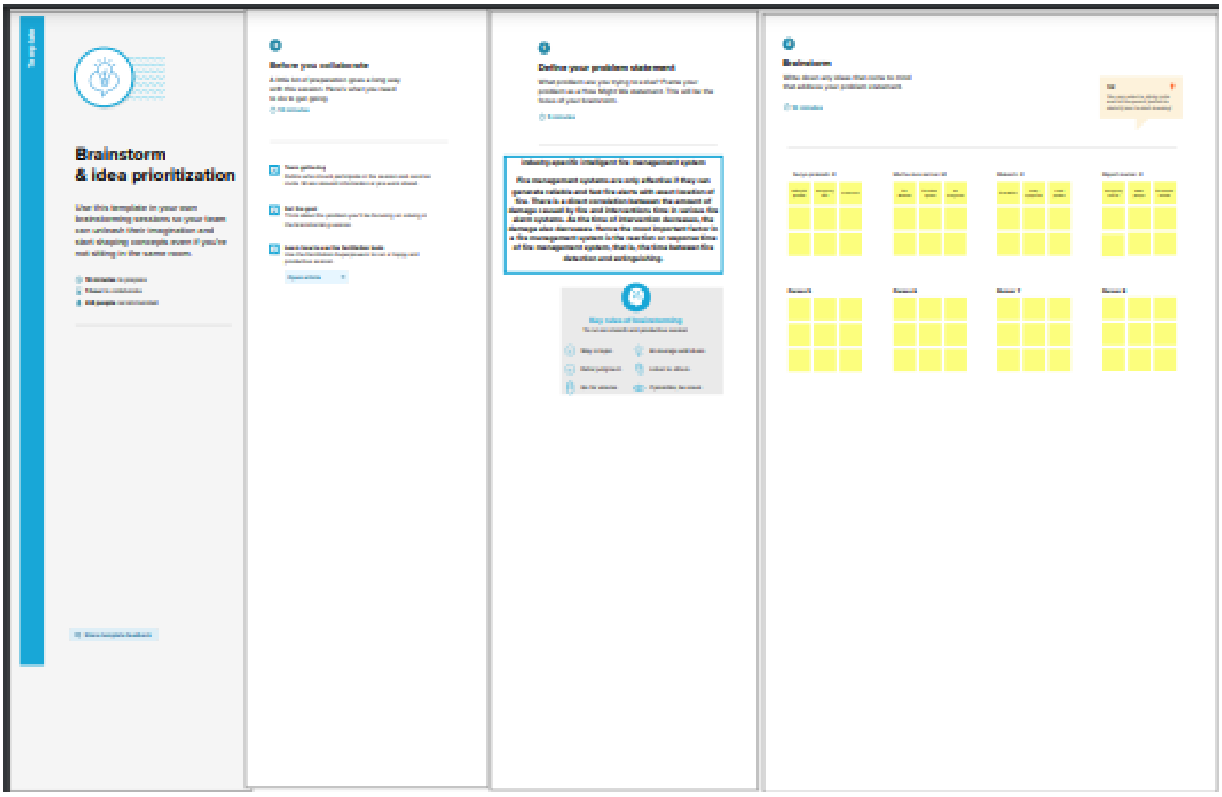
PROBLEM STATEMENT DEFINITION:

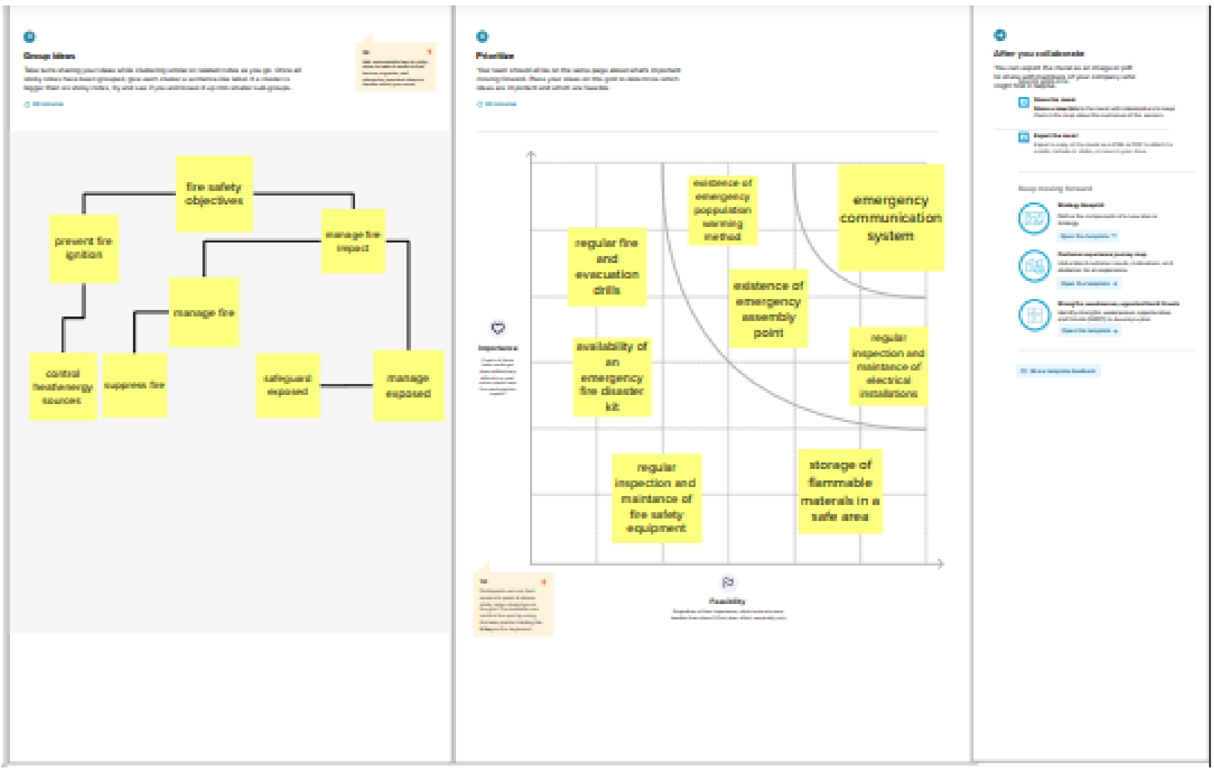
An industry needs a way to avoid the possibilities of unexpected accidents due to fire/gss leakage in the working area by sensing, alerting and taking necessary measures, in order to avoid those accidents because it causes damage to valuable human lives, properties and machinery.

EMPATHY MAP CANVAS:



BRAINSTORMING IDEAS:





PROPOSED SOLUTION:

S.No	Parameter	Description
1	Problem statement	The Smart fire management system helps to detect the changesintheenvironment usinggassensor, flamesensor and temperature sensor and deals with the aftermath caused by any breakdown in the workspace.
2	Idea / Solution description	1) If the temperature readings cross the threshold exhaust fans are powered ON. 2) Ifflameisdetected sprinkleswillbeswitched ON. 3) Alarms will be turned ONtoalert theemployees. 4)Message will be sent along with the location to fire station incase if the flame is uncontrollable.
3	Novelty / Uniqueness	Usageofliquid Nitrogen. Liquidnitrogenwill immediately vaporize causing a cooling effect and makes the site deprived ofoxygen. Using water sprinklers after liquid nitrogen can put off even intense fire effectively.
4	Social impact / Customer satisfaction	1) Harmful gases can be purified and released into atmosphere 2) Cause of impact can be traced from the sensor data that can be analyzed to prevent future accidents.
5	Business model (Revenue model)	1) This is used tocalculate the probability ofignition and spread of fires across a landscape. 2) This outcome allows for a better understanding of how changesinone aspects ofmanagement can affect other aspects of management.
6	Scalability of the solution	1) This can be implemented in all type of industries. 2) Thisdesigncanbeadministered inrestaurants which deals with high usage of fire and gas

PROBLEM SOLUTION FIT:

1. CUSTOMER SEGMENT(S) Workers is our customer	**4. EMOTION BEFORE /AFTER** If the fire is nearby, Remove occupants, enclose the area, activate alarm, call 5555, Try to fight the fire if safe to do so.	**7.BEHAVIOUR** Industrial fire safety measures include those that are intended to prevent ignition of an uncontrolled fire and those that are used to limit the development and effects of the fire after its starts.
2. JOBS TO BE DONE/PROBLEMS Protect the workers from the fire accident.	**5. AVAILABLE SOLUTION** Install and maintain fire alarm. Place fire alarm on our factory or industry. Tens fire alarm once a month.	**8.CHANNELS OF BEHAVIOUR** **OFFLINE:** Establish a fire prevention plan and emergency procedure, Inspect and maintain your equipment and facility. **ONLINE:** In online mode, incase of fire incident in industry it is quickly inform and alert to everyone.
3. TRIGGER In automated system, the presence of fire in the building will be picked up on by designated fire detectors. Then these fire detectors will in turn, trigger the fire alarm.	**6. CUSTOMER CONSTRAINTS** As far as risk to people go, the most appears danger is that from the flames this can cause severe burns to people caught in the fire, particularly if they are trapped and cannot escape the building.	**9. PROBLEM ROOT CAUSE** These accidents can occur from faulty wiring defective products, discarded cigarettes left on flammable materials, smoke and fire detectors that failed to activate.

10. YOUR SOLUTION Detect the presence of fire and alert its presence to the fire officials through mobile hotspot and internet from any place.

SOLUTION REQUIREMENTS:

Functional Requirements:

Following are the functional requirements of the proposed solution.

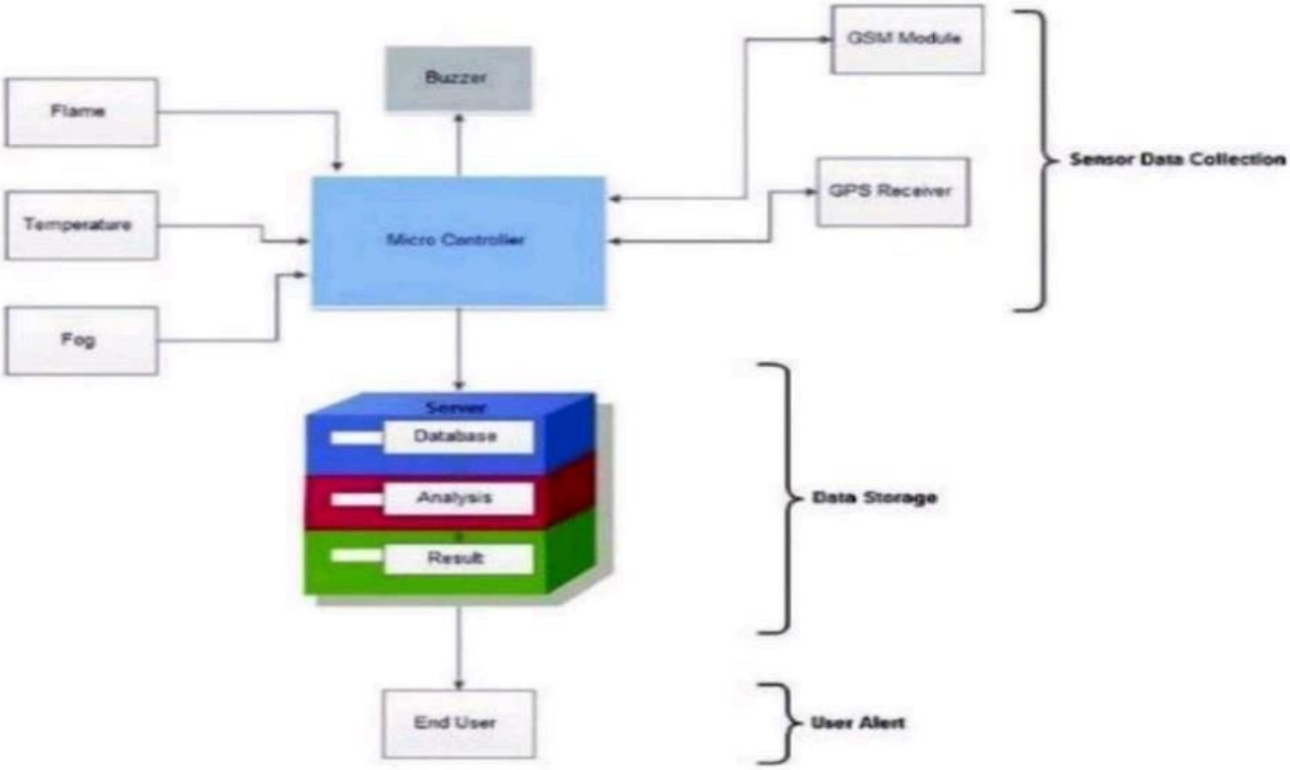
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Sensing function	Fire breakout has to be sensed by smoke detectors. Gas leakage has to be sensed by gas sensors.
FR-2	Alerting function	Blaring of alarms.
FR-3	Actuation function	Activation of sprinklers. Turning ON the exhaust Fan.
FR-4	Notification	Sending SMS with location to the fire station. Sending SMS to the authorities.

Non-functional Requirements:

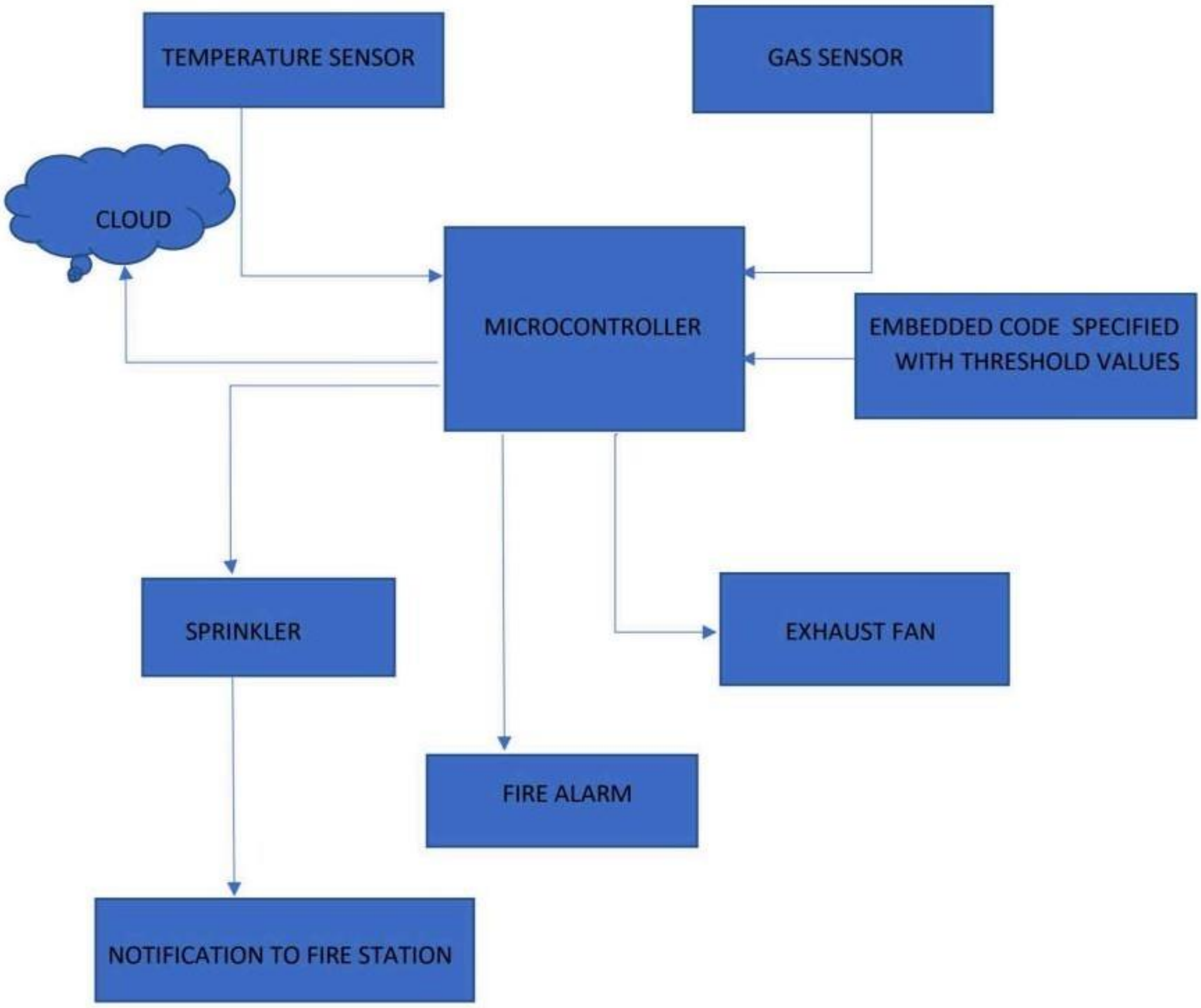
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Ease of use and longevity of the system.
NFR-2	Security	Software remains secured in the face of attacks.
NFR-3	Reliability	High accuracy.
NFR-4	Performance	Faster response.
NFR-5	Availability	Availability of the systems for institutions, restaurants and other public places
NFR-6	Scalability	It accommodates easy modification for various Requirements

DATAFLOW DIAGRAMS:



SOLUTION &TECHNICAL ARCHITECTURE:



USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (industrialist)	Sensor system	USN-1	Smoke sensor detecting the fire breakdown.	I can get information about fire breakdown.	High	Sprint-1
		USN-2	Gas sensors detecting gas leakage.	I can receive information about hazardous gas leakage.	High	Sprint-1
	Alerting system	USN-3	Blaring of alarm.	Alert the employees to leave the building.	Medium	Sprint-1
	Actuation system	USN-4	Turning ON the sprinklers.	To put of fire at the initial stage.	Medium	Sprint-2
		USN-5	Switching ON exhaust fan	To filter the poisonous gas.	High	Sprint-2
Customer (company managemen t)	Notification system	USN-6	Sending SMS to fire station along with current location.	To alert the fire station immediately after analysing its severity.	High	Sprint 3
		USN-7	Notlfying the management about the incident happened.	To be aware of breakdown.	Low	Sprint 4
	Cloud deployment	USN-8	Pushing data to the cloud.	To store and retrieve data for future requirements.	Low	Sprint 4

MILESTONES

1. Prerequisties

-》IBM Cloud Services

-》Software

2.Project Objectives

-》Abstract

-》Brainstorming

3.Create and Configure IBM Cloud Services

-》Create IBM Watson Iot Platform and Device

-》Create Node- Red Service

-》Create A Database in Cloudant DB

4.Develop the Python Script

-》Develop A Python Script

5. Develop A Web Application Using Node-RED Service.

-》Develop The Web Application Using Node-RED

6.Ideation Phase

-》Literature Survey on The Selected Project & Information Gathering

-》Prepare Empathy Map

-》Ideation

7.Project Design Phase -1

-》Proposed Solution

-》Prepare Solution Fit

-》 Solution Architecture

8.Project Design Phase -2

-》Customer journey

-》 Functional Requirement

-》Data Flow Diagram

-》Technology Architecture

9.Project planning Phase

-》 Prepare Milestones & Activity List

-》 Sprint Delivery Plan

10.Project Development Phase

-》Project Development-Delivery Of Sprint-1

-》 Project Development-Delivery Of Sprint-2

-》 Project Development-Delivery Of Sprint-3

-》 Project Development-Delivery Of Sprint-4

TECHNOLOGY STACK:

COMPONENTS AND TECHNOLOGIES:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the Application	Java / Python
3.	Application Logic-2	Logic for a process in the Application	IBM Watson S I I service
4.	Application Logic-3	Logic for a process in the Application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the Application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the Application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration:	Local, Cloud Foundry, Kubernetes, etc.

APPLICATION CHARACTERISTICS:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source Frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g., SHA-256, Encryptions, IAM Controls, OWASP etc.
3.	Scalable Architecture	Justify the scalability of architecture (3 — tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g., use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Technology used

SPRINT DELIVERY PLAN:

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	Surya prakash
	Operating	USN-2	Turning on the exhaust fan as well as the fire sprinkler system in cause of fire and gas leakage.	3	Medium	Muthu siva sankar
Sprint-2	Sending collected data to the IBM Watson platform	USN-3	Sending the data of the Sensors to the IBM Watson.	3	High	mukesh
	Node red	USN-4	Sending the data from the IBM Watson to the Node red.	3	High	Rajesh kumar

Sprint-3	Storing of sensor data	USN-5	Storing in Cloudant database.	2	Medium	Surya prakas h
	Registratio n	USN-6	Entering my email and password to verify authentication process.	1	Medium	mukesh
	Web UI	USN-7	Monitors the situation of the environment which displays sensor information.	3	High	Muthu siva sankar
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	Surya prakash Mukesh Muthu siva sankar Rajesh kumar
	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	Surya prakash Muthu siva sankar Mukesh Rajesh kumar
	Testing	USN-10	Testing of project and Final Deliverables.	1	Low	Surya prakash mukesh Rajesh Muthu siva sankar

SPRINT 1:

```
#include <WiFi.h>//library for wifi

#include <PubSubClient.h>//library for MQtt

#include "DHT.h"// Library for dht11

#define DHTPIN 15 // what pin we're connected to

#define DHTTYPE DHT22 // define type of sensor DHT 11

#define LED 2
```

```
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 "gh6uoi"//IBM ORGANITION ID
#define DEVICE_TYPE "trial1"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234abcd"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
float h, t;
//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format
in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND
COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id
//-----------------------------------------------------------------
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing
parameter like server id,portand wificredential
void setup()// configureing the ESP32
{
 Serial.begin(115200);
 dht.begin();
 pinMode(LED,OUTPUT);
 delay(10);
 Serial.println();
 pinMode(mq2, INPUT);
```

```
 pinMode (flame_sensor_pin , INPUT ); // declaring sensor pin as input pin for Arduino
 pinMode(BUZZER_PIN, OUTPUT);
 wificonnect();
 mqttconnect();
}
void loop()// Recursive Function
{
 t = dht.readTemperature();
 Serial.print("temp:");
 Serial.println(t);
if(t > 60)
{
Serial.println("Alert");
digitalWrite(BUZZER_PIN, HIGH); // turn on
}
else
{
digitalWrite(BUZZER_PIN, LOW); // turn on
}
 int gassensorAnalogmq2 = analogRead(mq2);
Serial.print("mq2 Gas Sensor: ");
Serial.print(gassensorAnalogmq2);
if (gassensorAnalogmq2 > 1500)
{
Serial.println("mq2Gas");
Serial.println("Alert");
}
else
{
Serial.println("No mq2Gas");
```

```
}
flame_pin = digitalRead ( flame_sensor_pin ) ; // reading from the
sensor if (flame_pin == LOW ) // applying condition
{
Serial.println ( " ALERT: FLAME DETECTED" ) ;
digitalWrite ( BUZZER_PIN , HIGH ) ;// if state is high, then turn high the
BUZZER }
else
{
Serial.println ( " NO FLAME DETECTED " ) ;
digitalWrite ( BUZZER_PIN , LOW ) ; // otherwise turn it low
}
 PublishData(t, gassensorAnalogmq2, flame_pin);
 delay(1000);
 if (!client.loop()) {
 mqttconnect();
  }
}
/*....................................retrieving to Cloud…......................... */
void PublishData(float t, float gassensorAnalogmq2 , int flame_pin ) {
 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 += t;
 payload += "," "\"gasvalue\":";
 payload += gassensorAnalogmq2;
 payload += "," "\"flame\":";
 payload += flame_pin;
```

```
 payload += "}";


 Serial.print("Sending payload: ");
 Serial.println(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 Serial monitor 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);
 }
 initManagedDevice();
 Serial.println();
 }
}
void wificonnect() //function defination for wificonnect
{
 Serial.println();
 Serial.print("Connecting to ");
 WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
 while (WiFi.status() != WL_CONNECTED) {
 delay(500);
```

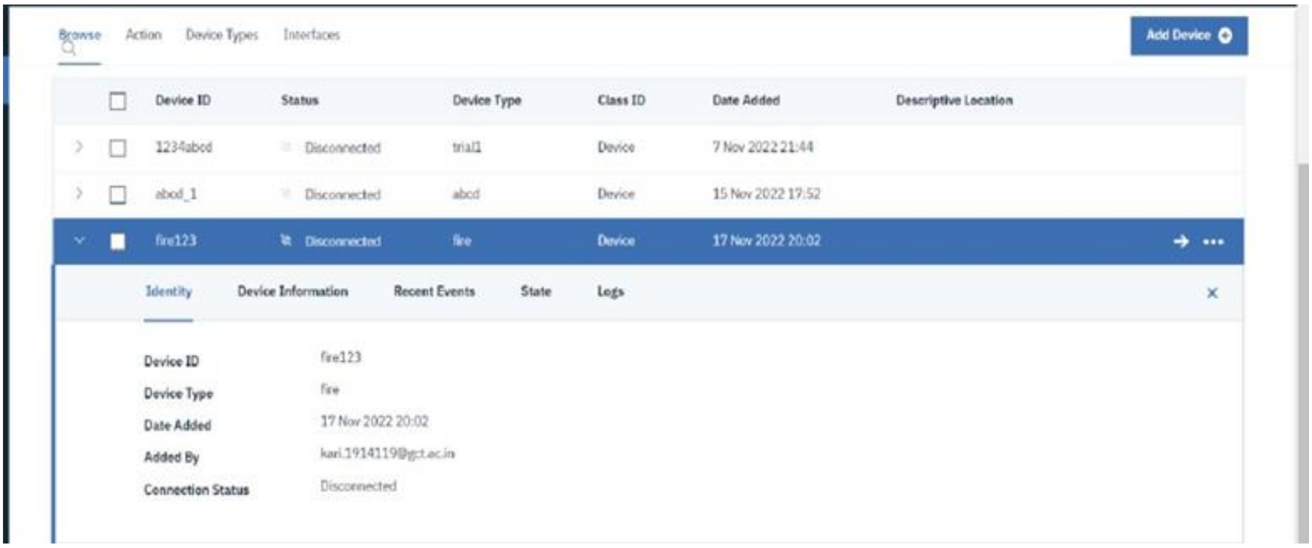
```
 Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void initManagedDevice() {
 if (client.subscribe(subscribetopic)) {
 Serial.println((subscribetopic));
 Serial.println("subscribe to cmd OK");
 } else {
 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++) {
 //Serial.print((char)payload[i]);
 data3 += (char)payload[i];
 }
 Serial.println("data: "+ data3);
 if(data3=="lighton")
 {
Serial.println(data3);
digitalWrite(LED,HIGH);
 }
```

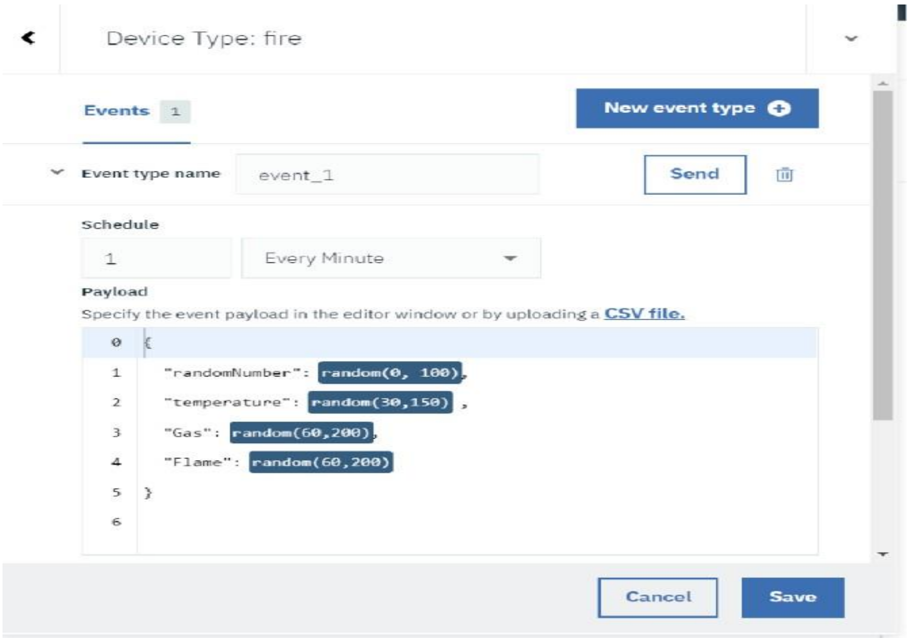
```
else
 {
Serial.println(data3);
digitalWrite(LED,LOW);
 }
data3="";
}
```

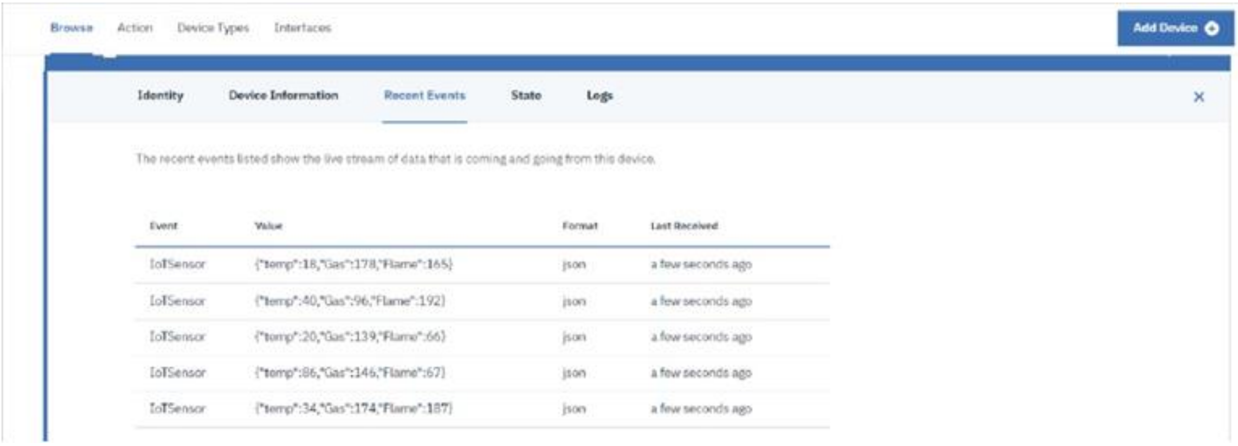
SPRINT 2:

INTERFACING CLOUD WITH INTERNET OF THINGS PLATFORM:

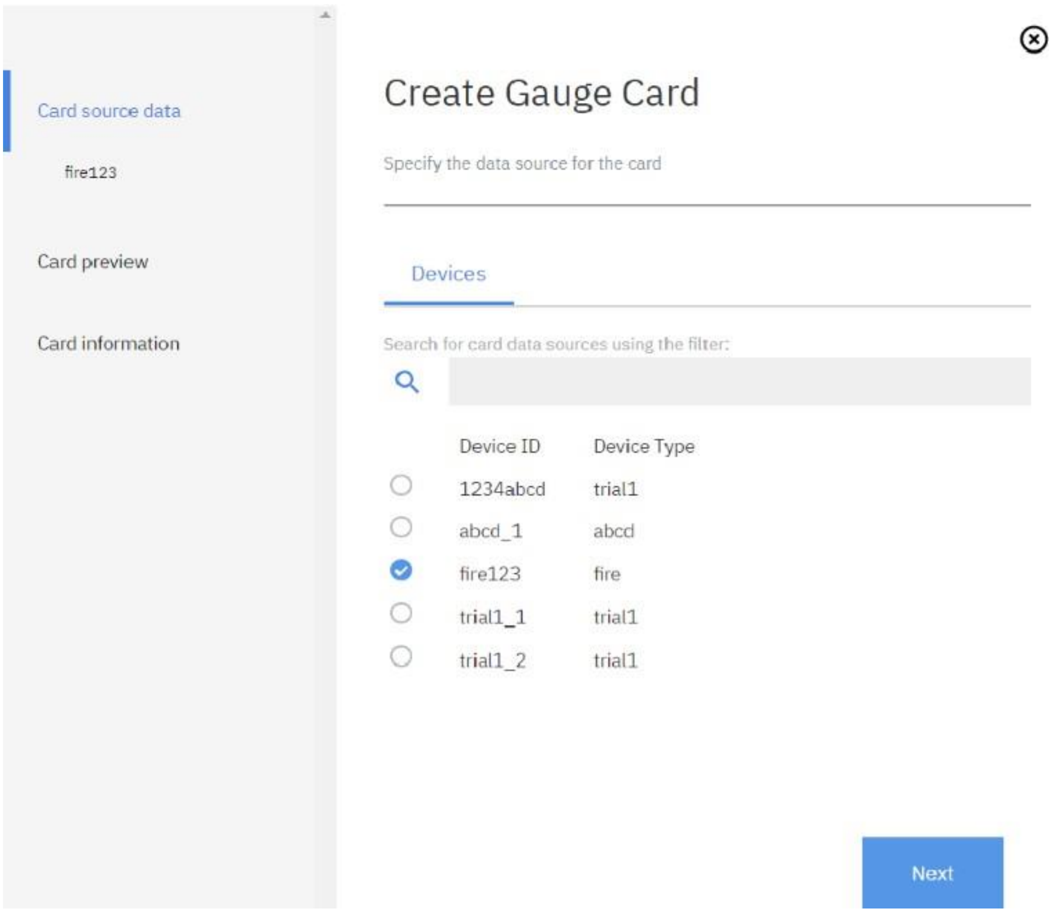
1. **CREATING SIMULATION**

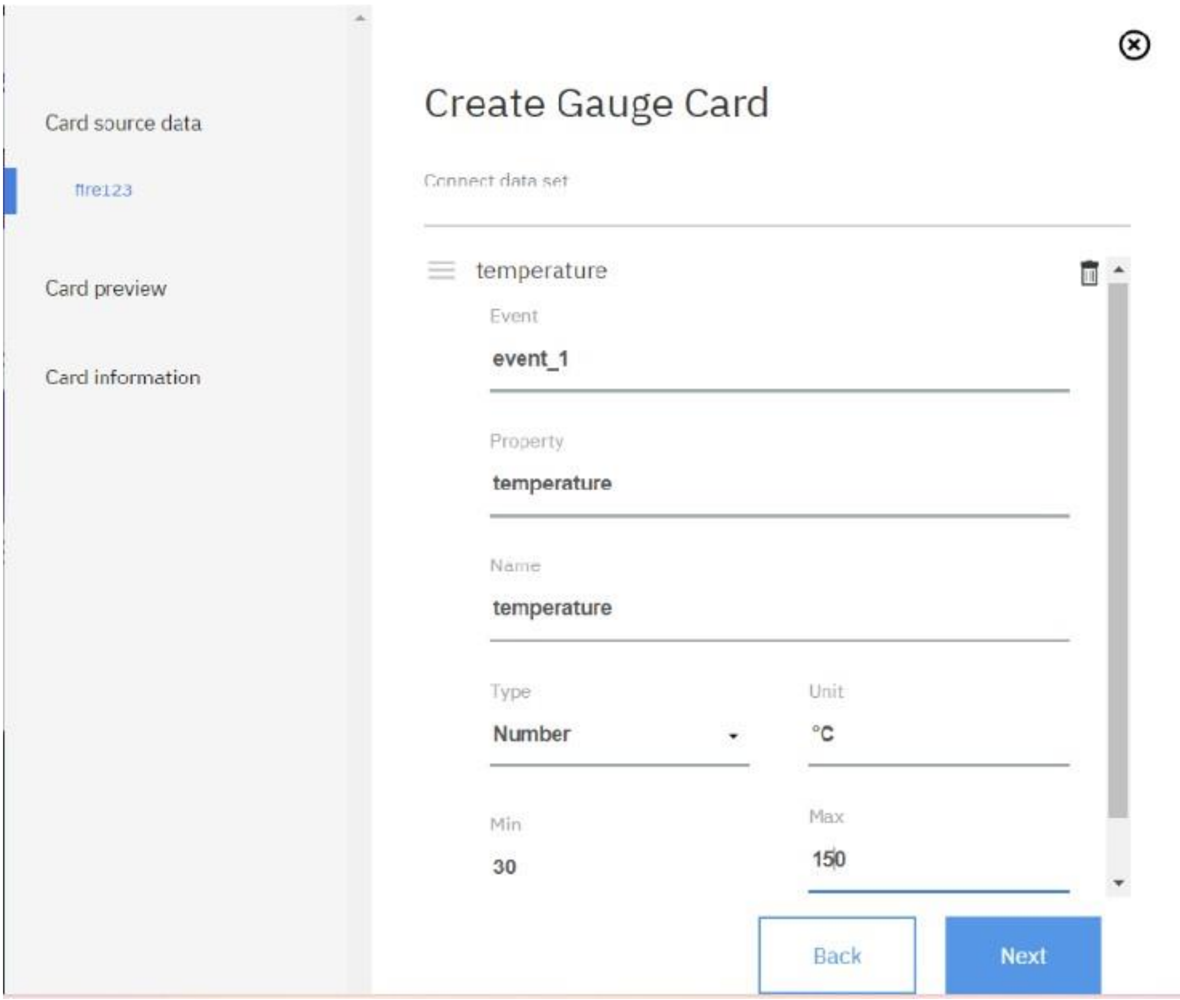


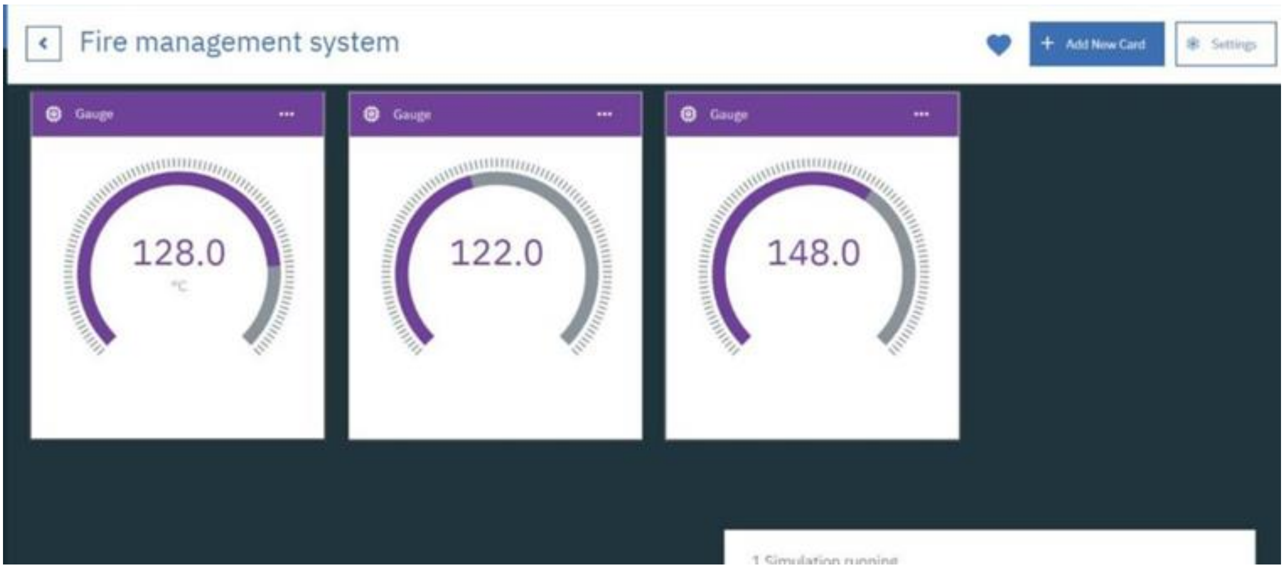




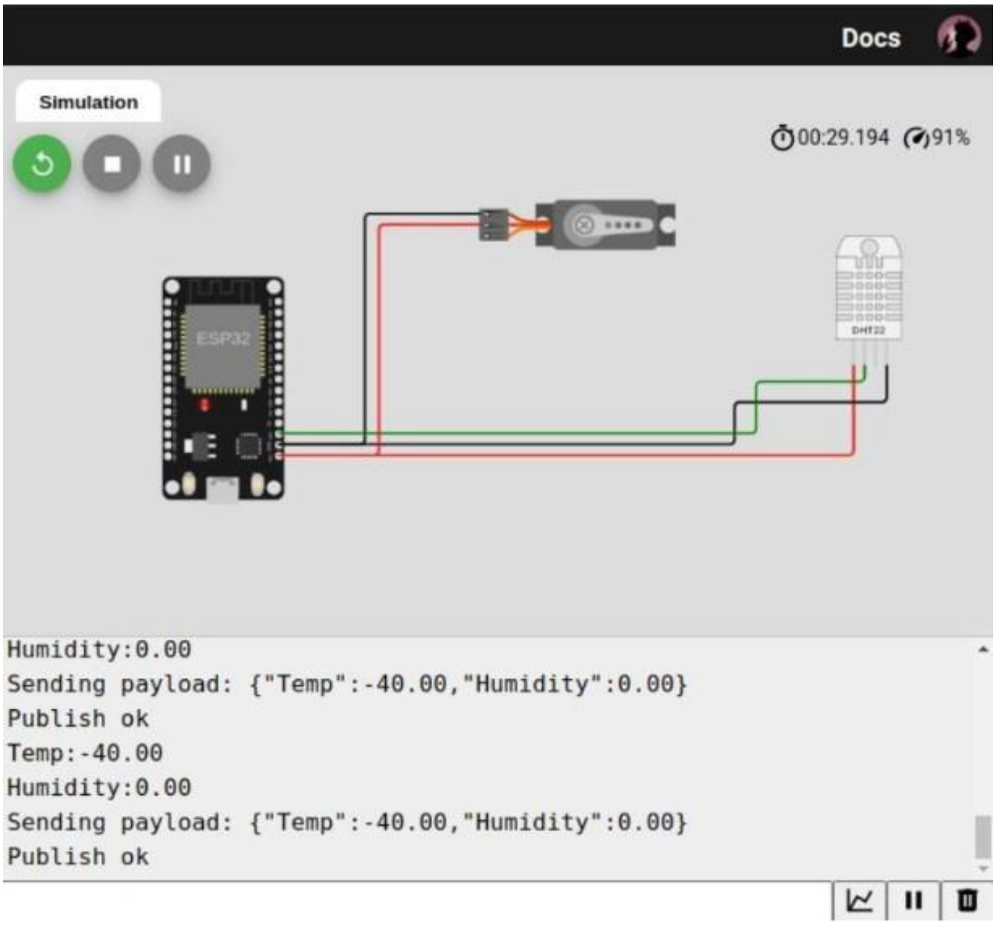
2. ADDING BOARDS:

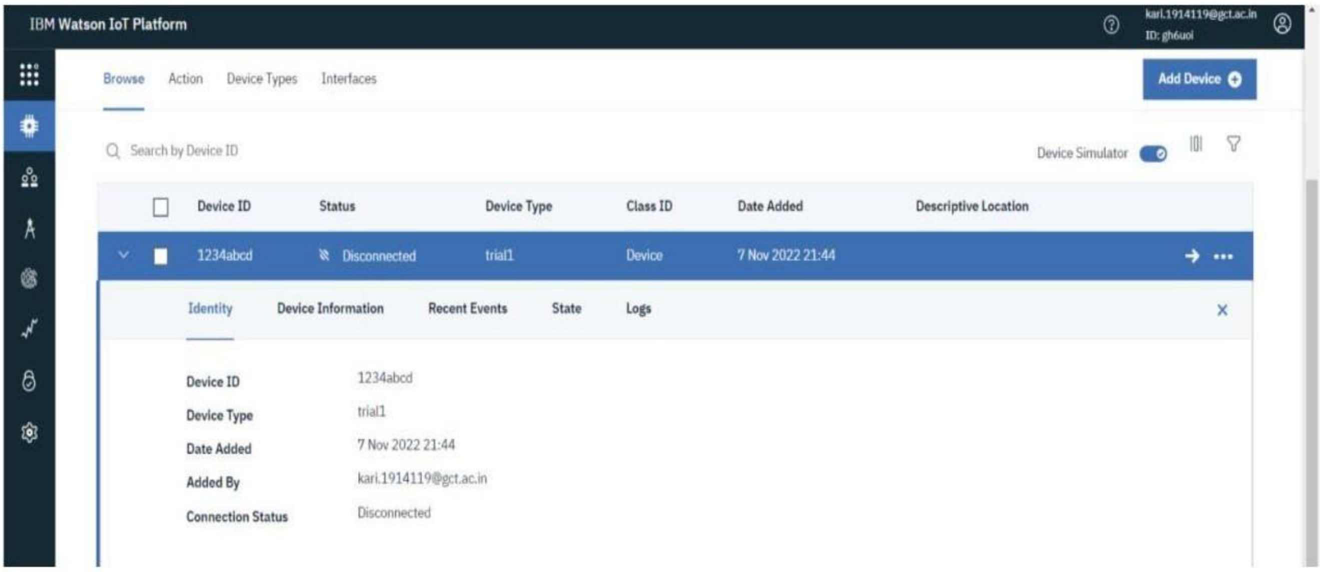


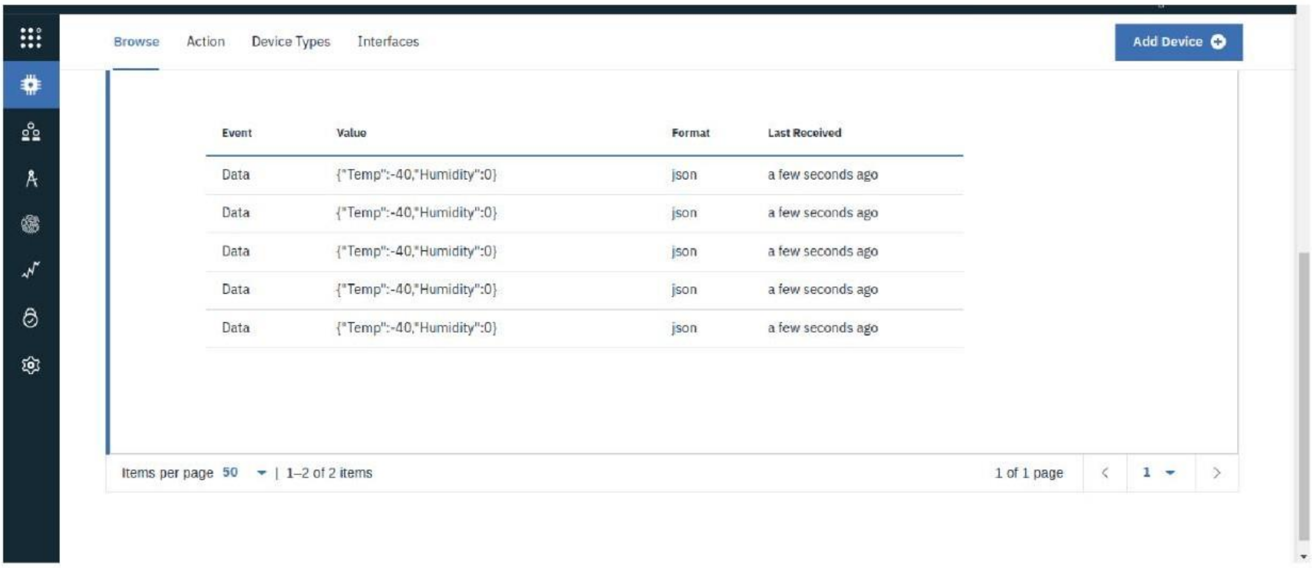

Card source data
fire123
Card preview
Card information
Create Gauge Card
Connect data set
temperature
Event
event_1
Property
temperature
Name
temperature
Type
Number
Unit
°C
Min
30
Max
150
Back
Next


Fire management system
Add New Card
Settings
Gauge
128.0
°C
Gauge
122.0
Gauge
148.0

WOKWI SIMULATION:







SPRINT- 3:

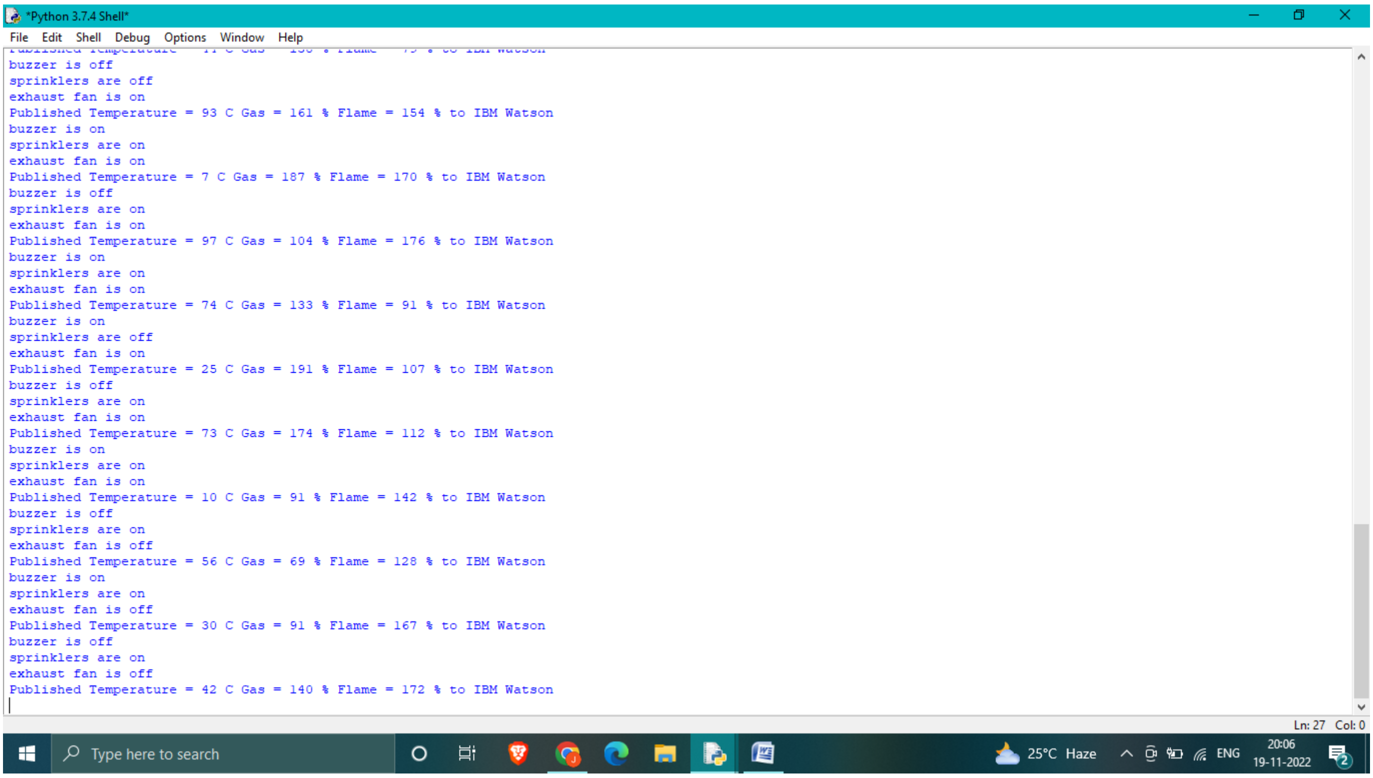
PYTHON SCRIPT:

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random

# Provide your IBM Watson Device Credentials

organization = "gh6uoi"

deviceType = "fire"

deviceId = "fire123"

authMethod = "token"

authToken = "0123456789"

# print(cmd)

try :

    deviceOptions = {"org" : organization, "type" : deviceType, "id" : deviceId, "auth-method" :
authMethod,  "auth-token" : authToken}

    deviceCli = ibmiotf.device.Client ( deviceOptions )

# ..............................................
```

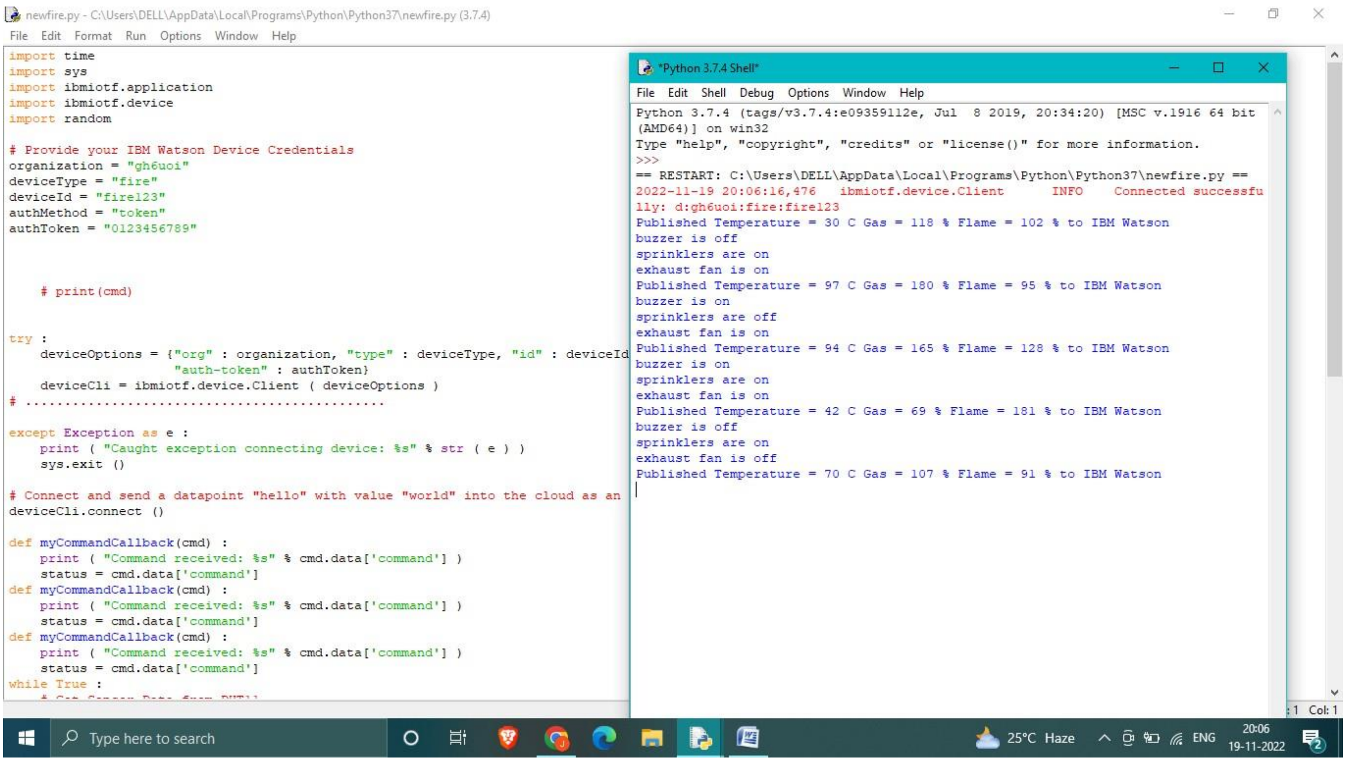
```
except Exception as e :

    print ( "Caught exception connecting device: %s" % str ( e ) )

    sys.exit ()

# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting"
10 times

deviceCli.connect ()


def myCommandCallback(cmd) :

    print ( "Command received: %s" % cmd.data['command'] )

    status = cmd.data['command']

def myCommandCallback(cmd) :

    print ( "Command received: %s" % cmd.data['command'] )

    status = cmd.data['command']

def myCommandCallback(cmd) :

    print ( "Command received: %s" % cmd.data['command'] )

    status = cmd.data['command']

while True :

    # Get Sensor Data from DHT11

    temp = random.randint ( 0, 100 )

    gas = random.randint ( 60, 200 )

    flame = random.randint ( 60, 200 )

    data = {'temp' : temp, 'Gas' : gas,'Flame': flame}

    # print data

    def myOnPublishCallback() :

        print ( "Published Temperature = %s C" % temp, "Gas = %s %%" % gas, "Flame = %s %%" %
flame, "to IBM Watson")

    success = deviceCli.publishEvent ( "IoTSensor", "json", data, qos = 0, on_publish =
myOnPublishCallback )

    if not success :
```

```
        print ( "Not connected to IoTF" )
    time.sleep ( 1 )
    deviceCli.commandCallback = myCommandCallback
# Initialize GPIO
    if temp > 50 :
        print ( "buzzer is on" )
    else :
        print ( "buzzer is off" )
    if flame > 100 :
        print ( "sprinklers are on" )
    else :
        print ( "sprinklers are off" )
    if gas>100:
        print ( "exhaust fan is on" )
    else :
        print ( "exhaust fan is off" )
# Disconnect the device and application from the cloud
deviceCli.disconnect ()
```

OUTPUT OF PYTHON CODE:



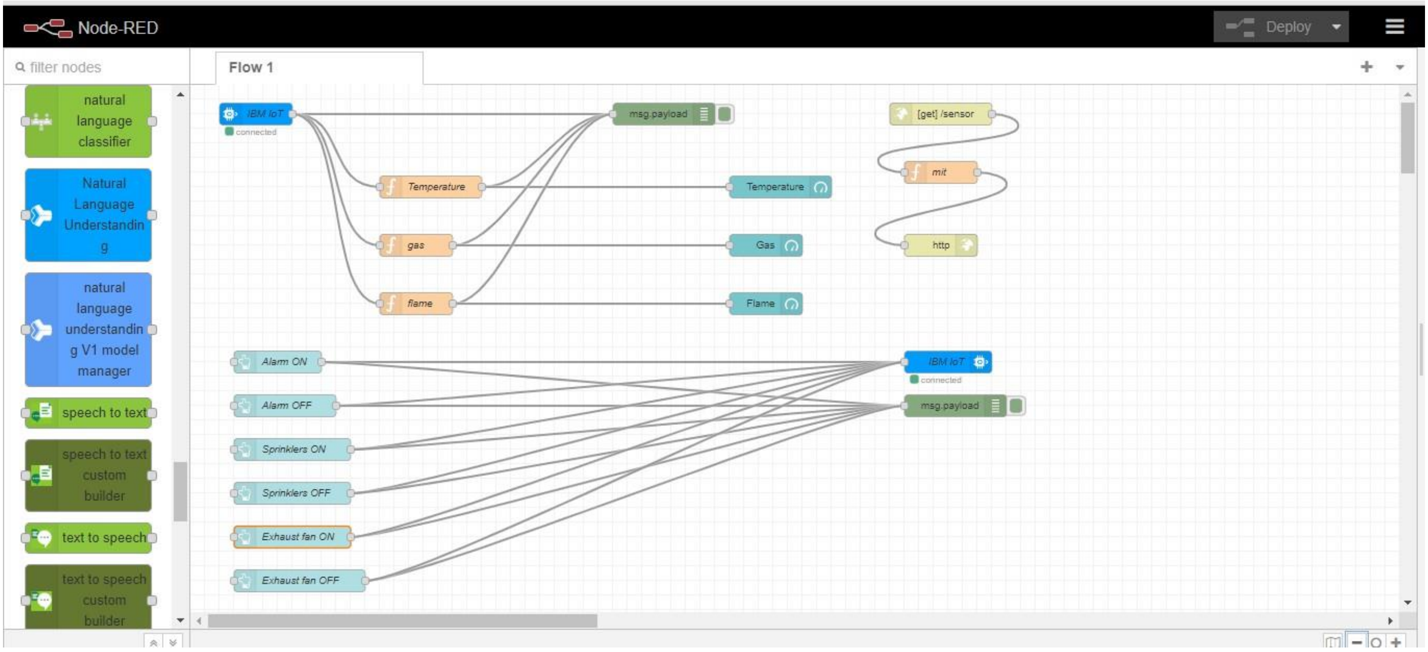
```
*Python 3.7.4 Shell*
buzzer is off
sprinklers are off
exhaust fan is on
Published Temperature = 93 C Gas = 161 % Flame = 154 % to IBM Watson
buzzer is on
sprinklers are on
exhaust fan is on
Published Temperature = 7 C Gas = 187 % Flame = 170 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is on
Published Temperature = 97 C Gas = 104 % Flame = 176 % to IBM Watson
buzzer is on
sprinklers are on
exhaust fan is on
Published Temperature = 74 C Gas = 133 % Flame = 91 % to IBM Watson
buzzer is on
sprinklers are off
exhaust fan is on
Published Temperature = 25 C Gas = 191 % Flame = 107 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is on
Published Temperature = 73 C Gas = 174 % Flame = 112 % to IBM Watson
buzzer is on
sprinklers are on
exhaust fan is on
Published Temperature = 10 C Gas = 91 % Flame = 142 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is off
Published Temperature = 56 C Gas = 69 % Flame = 128 % to IBM Watson
buzzer is on
sprinklers are on
exhaust fan is off
Published Temperature = 30 C Gas = 91 % Flame = 167 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is off
Published Temperature = 42 C Gas = 140 % Flame = 172 % to IBM Watson
```



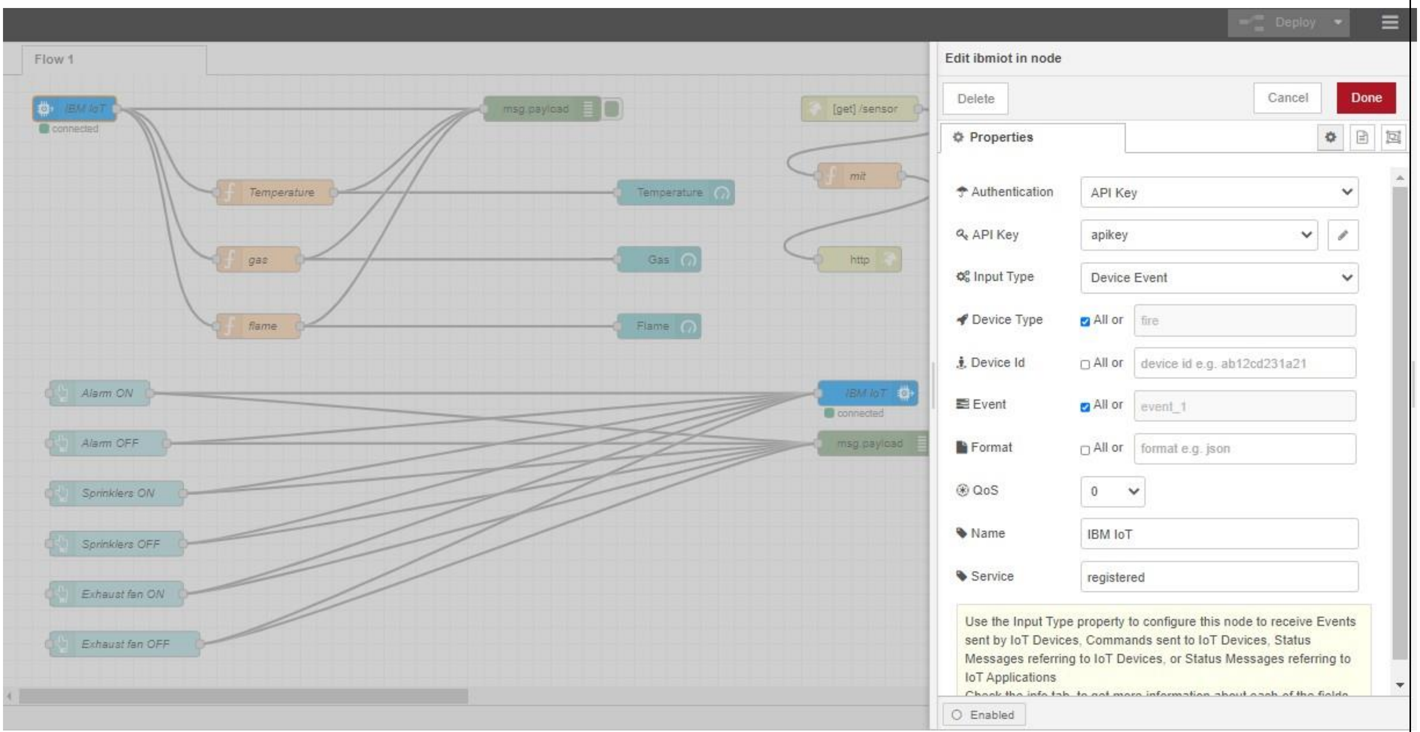
```
newfire.py - C:\Users\DELL\AppData\Local\Programs\Python\Python37\newfire.py (3.7.4)
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

# Provide your IBM Watson Device Credentials
organization = "gh6uoi"
deviceType = "fire"
deviceId = "fire123"
authMethod = "token"
authToken = "0123456789"



    # print(cmd)


try :
    deviceOptions = {"org" : organization, "type" : deviceType, "id" : deviceId
                      "auth-token" : authToken}
    deviceCli = ibmiotf.device.Client ( deviceOptions )
# .............................................

except Exception as e :
    print ( "Caught exception connecting device: %s" % str ( e ) )
    sys.exit ()

# Connect and send a datapoint "hello" with value "world" into the cloud as an
deviceCli.connect ()

def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
while True :
```

```
*Python 3.7.4 Shell*
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Users\DELL\AppData\Local\Programs\Python\Python37\newfire.py ==
2022-11-19 20:06:16,476   ibmiotf.device.Client      INFO    Connected successfu
lly: d:gh6uoi:fire:fire123
Published Temperature = 30 C Gas = 118 % Flame = 102 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is on
Published Temperature = 97 C Gas = 180 % Flame = 95 % to IBM Watson
buzzer is on
sprinklers are off
exhaust fan is on
Published Temperature = 94 C Gas = 165 % Flame = 128 % to IBM Watson
buzzer is on
sprinklers are on
exhaust fan is on
Published Temperature = 42 C Gas = 69 % Flame = 181 % to IBM Watson
buzzer is off
sprinklers are on
exhaust fan is off
Published Temperature = 70 C Gas = 107 % Flame = 91 % to IBM Watson
```

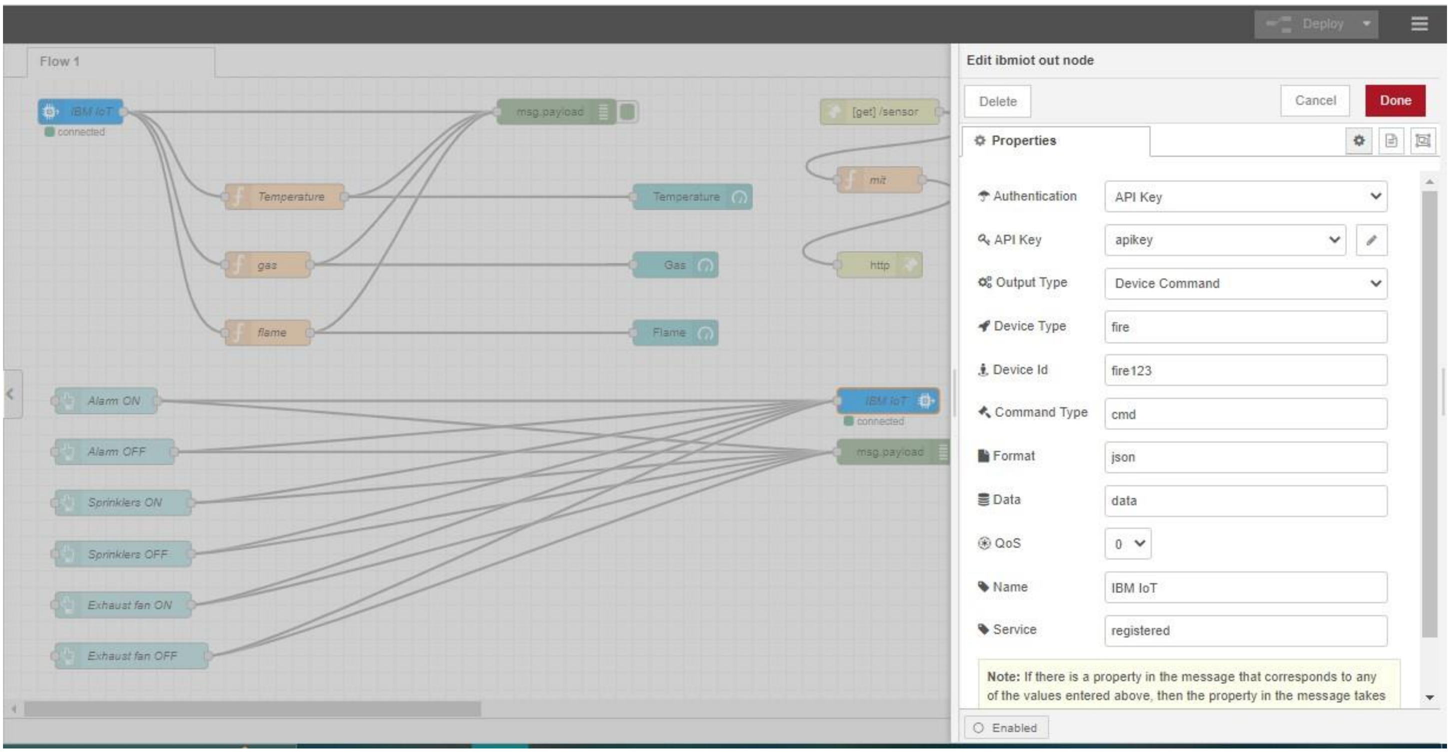
NODE RED:



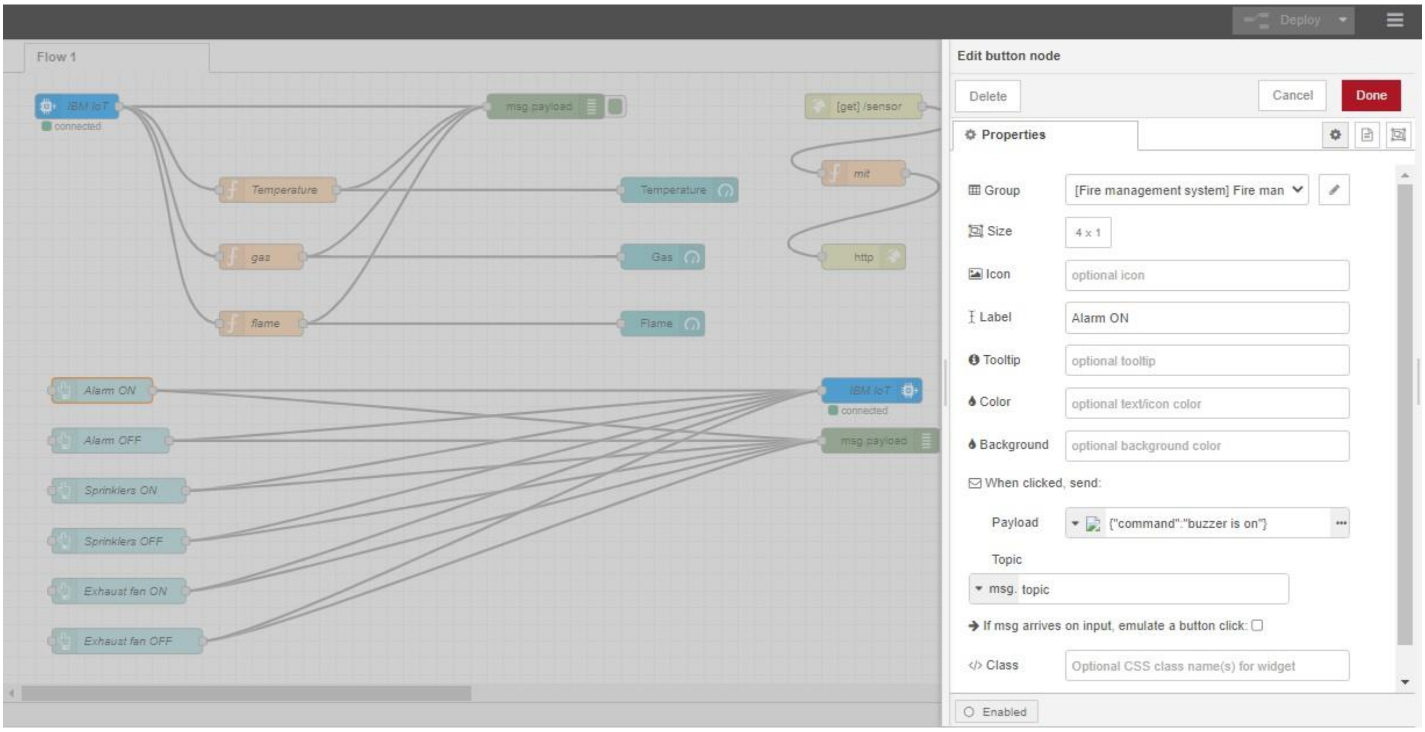
IBMIOT IN NODE:



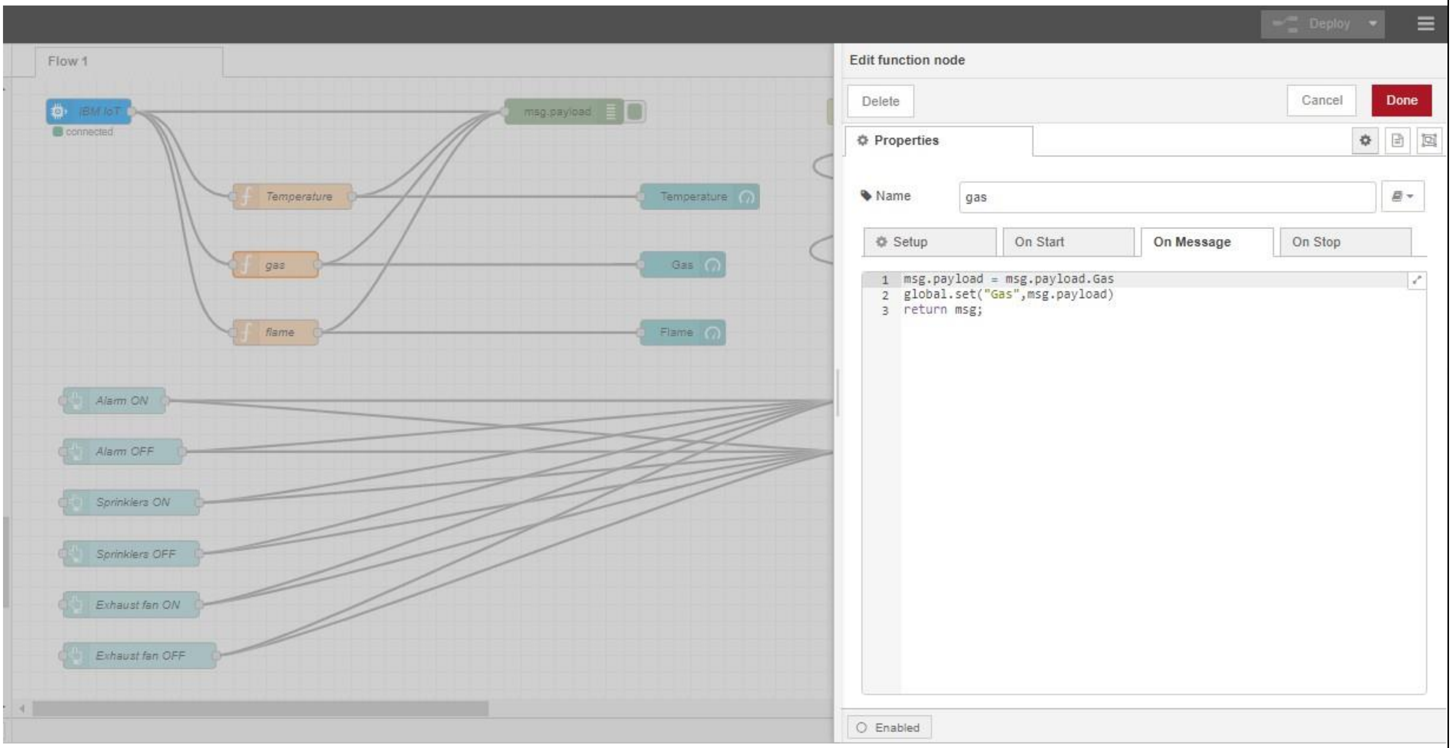
IBMIOT OUT NODE:

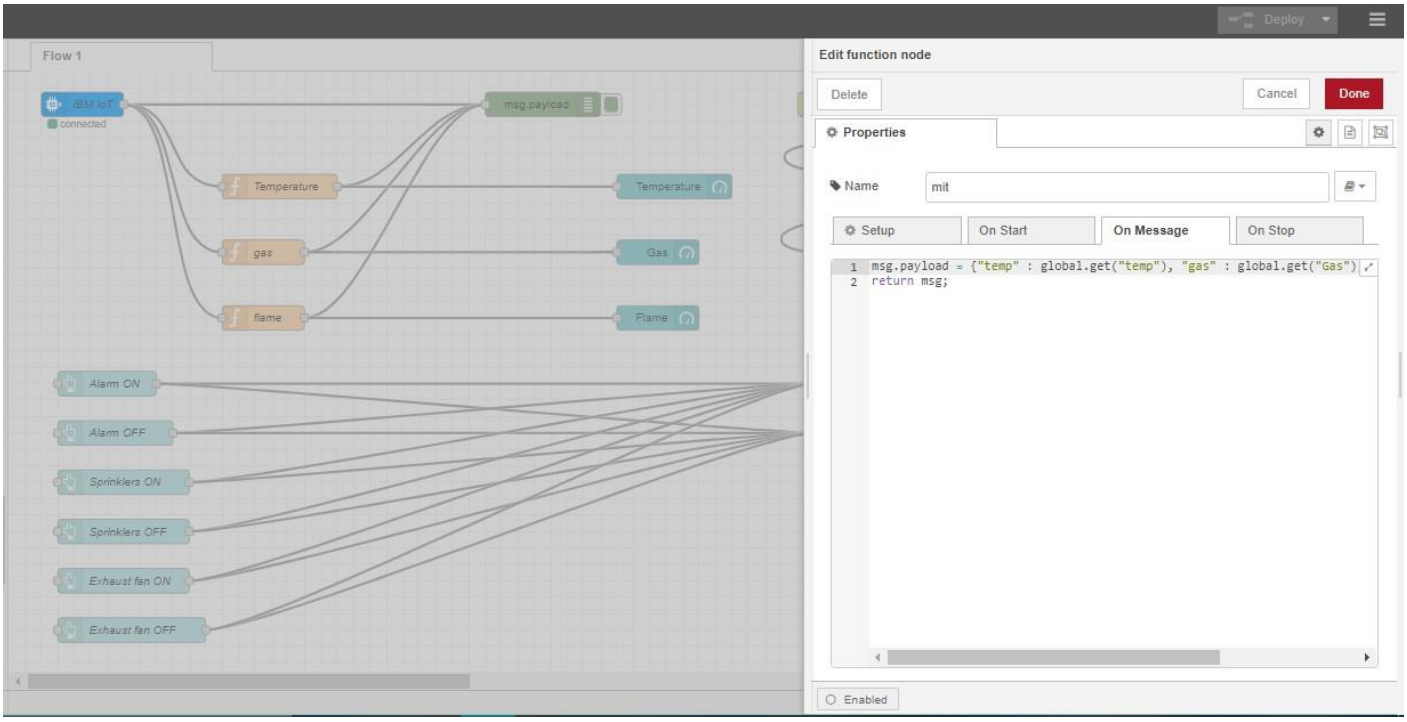


BUTTON NODE:

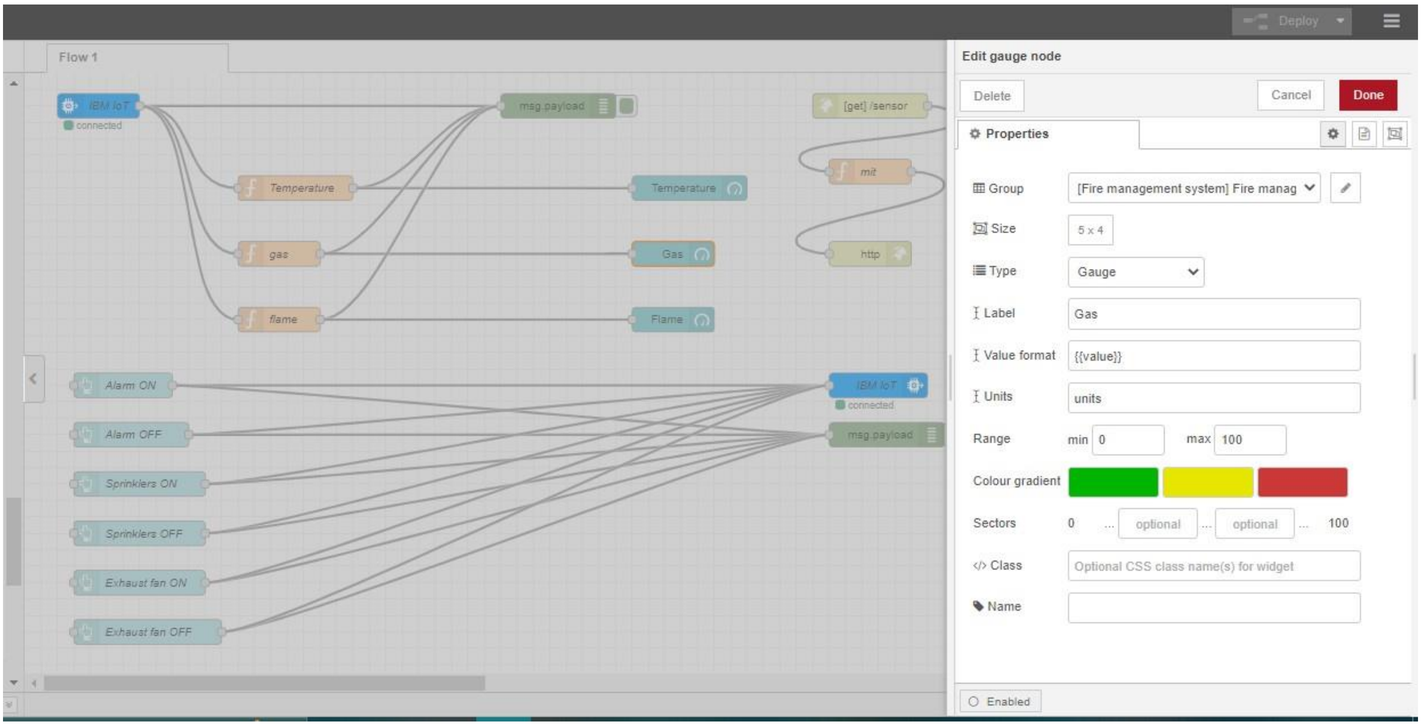


FUNCTION NODE:

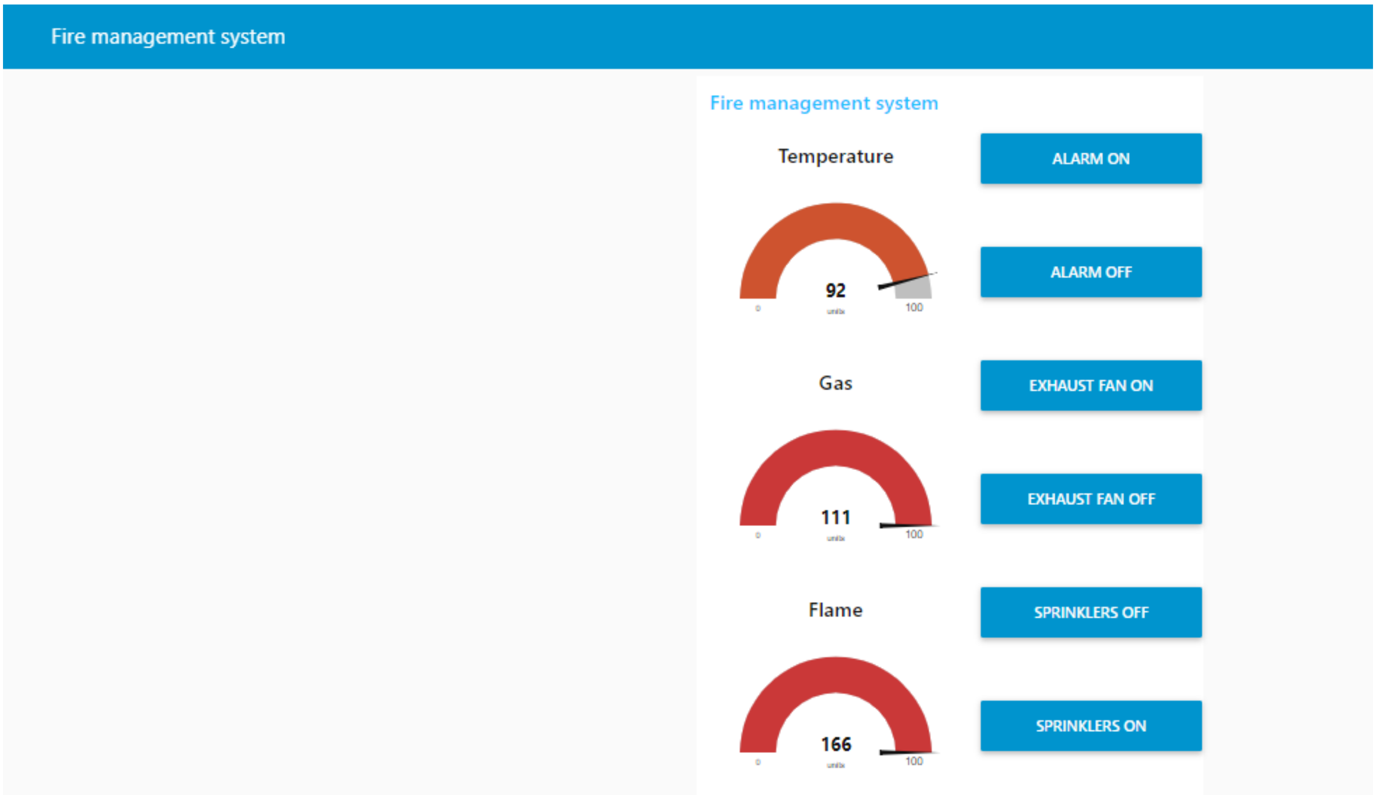




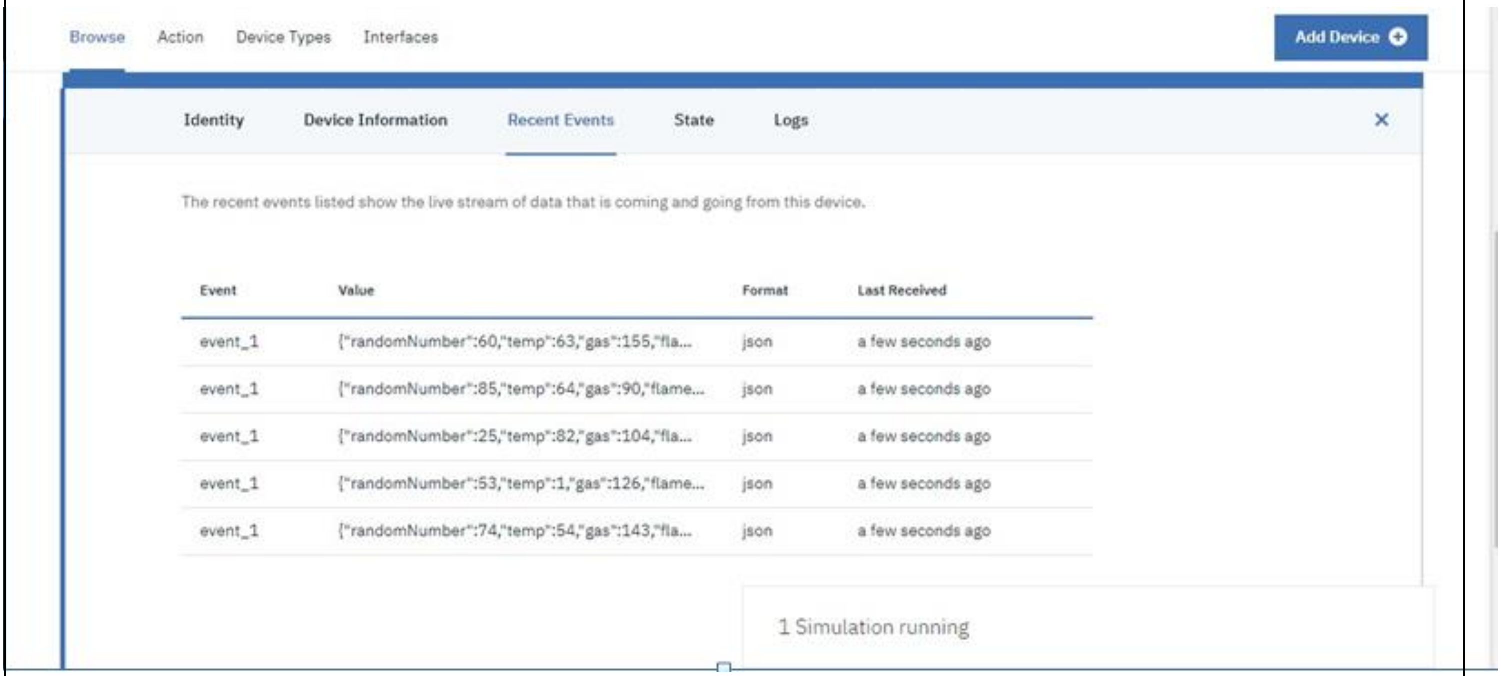
GAUGE NODE:

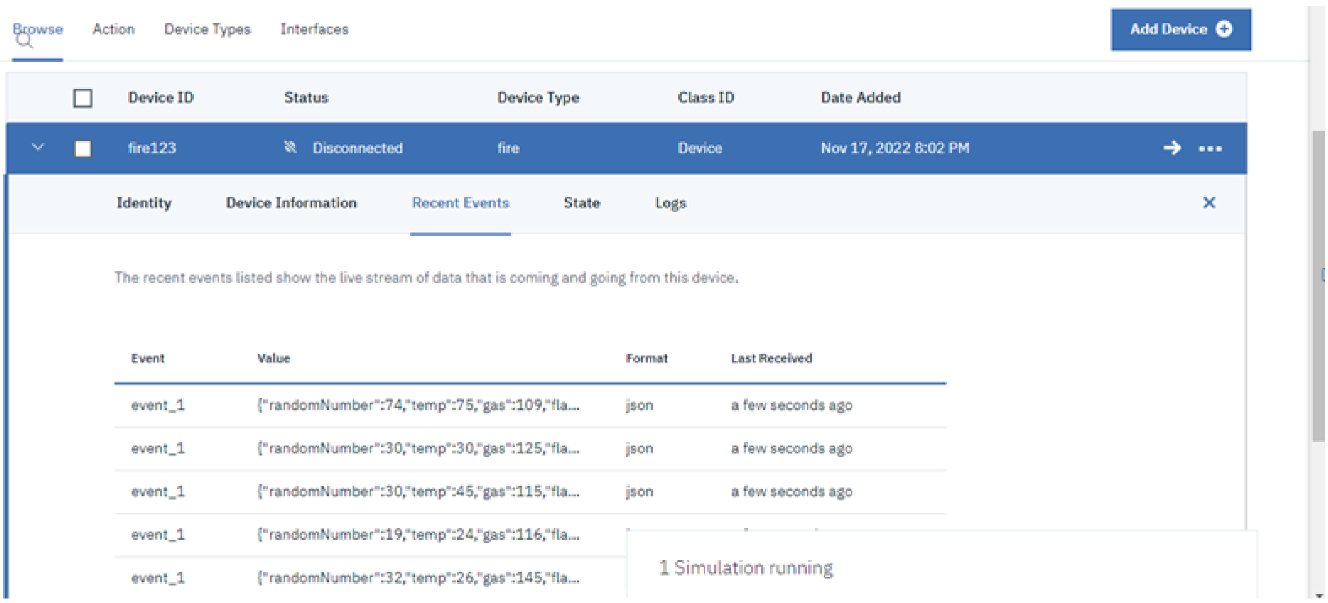


NODE RED WEB UI:

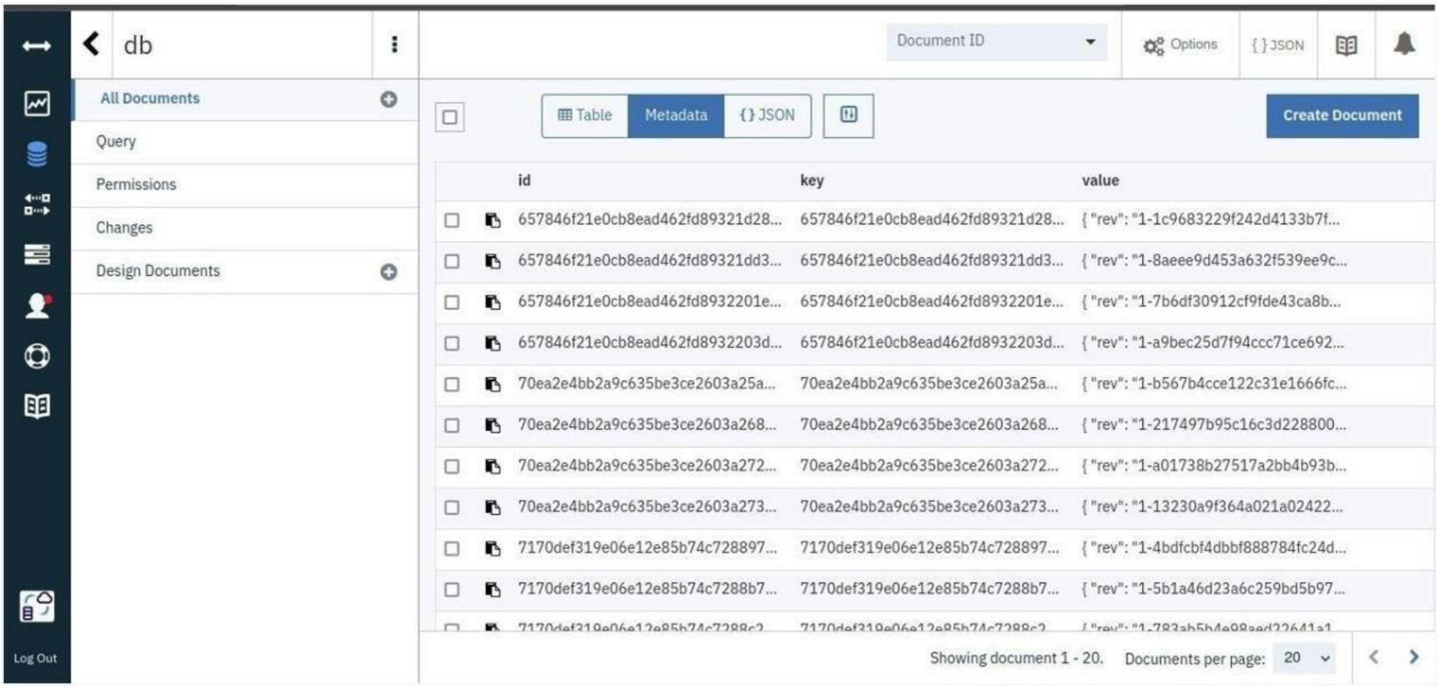


CLOUD INTERFACING WITH NODE RED:





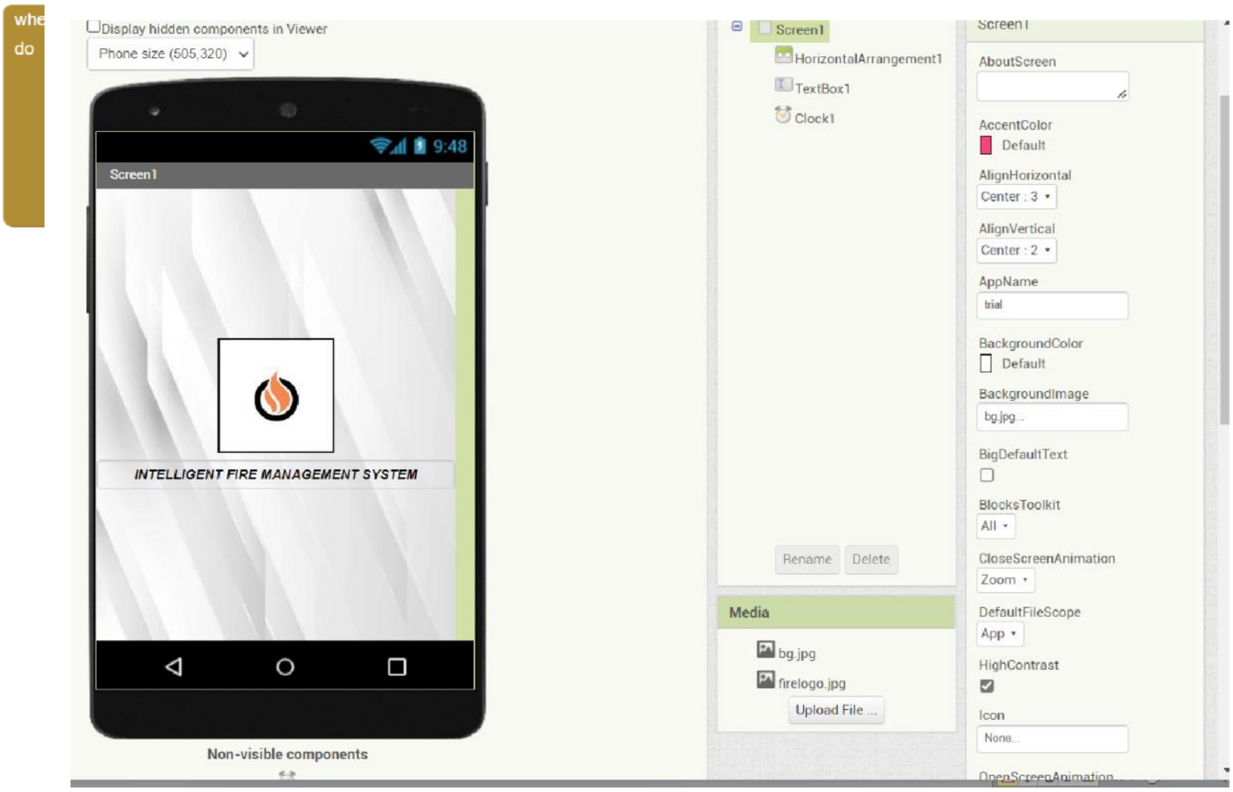
CLOUDANT:

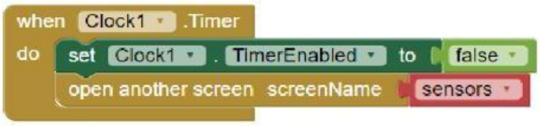

db
All Documents
Query
Permissions
Changes
Design Documents
Log Out
Document ID
Options
{ } JSON
Table
Metadata
{ } JSON
Create Document
id
key
value
657846f21e0cb8ead462fd89321d28... 657846f21e0cb8ead462fd89321d28... { "rev": "1-1c9683229f242d4133b7f...
657846f21e0cb8ead462fd89321dd3... 657846f21e0cb8ead462fd89321dd3... { "rev": "1-8aeee9d453a632f539ee9c...
657846f21e0cb8ead462fd8932201e... 657846f21e0cb8ead462fd8932201e... { "rev": "1-7b6df30912cf9fde43ca8b...
657846f21e0cb8ead462fd8932203d... 657846f21e0cb8ead462fd8932203d... { "rev": "1-a9bec25d7f94ccc71ce692...
70ea2e4bb2a9c635be3ce2603a25a... 70ea2e4bb2a9c635be3ce2603a25a... { "rev": "1-b567b4cce122c31e1666fc...
70ea2e4bb2a9c635be3ce2603a268... 70ea2e4bb2a9c635be3ce2603a268... { "rev": "1-217497b95c16c3d228800...
70ea2e4bb2a9c635be3ce2603a272... 70ea2e4bb2a9c635be3ce2603a272... { "rev": "1-a01738b27517a2bb4b93b...
70ea2e4bb2a9c635be3ce2603a273... 70ea2e4bb2a9c635be3ce2603a273... { "rev": "1-13230a9f364a021a02422...
7170def319e06e12e85b74c728897... 7170def319e06e12e85b74c728897... { "rev": "1-4bdfcbf4dbbf888784fc24d...
7170def319e06e12e85b74c7288b7... 7170def319e06e12e85b74c7288b7... { "rev": "1-5b1a46d23a6c259bd5b97...
Showing document 1 - 20.
Documents per page:
20

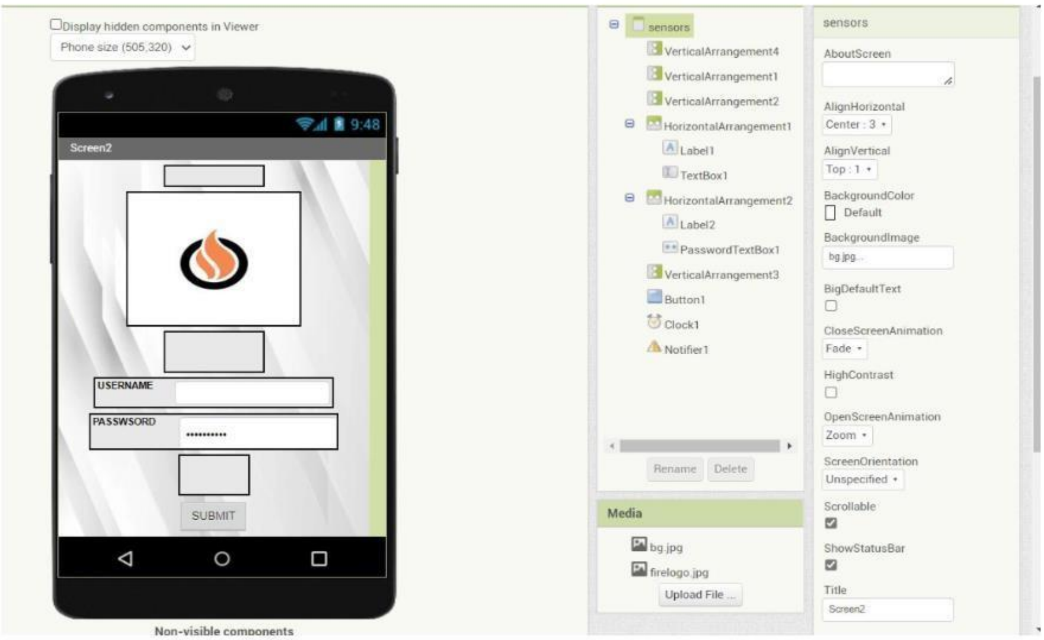

Save Changes
Cancel
"_id": '657846f2*eBcb8ead4&2fd8e321d28fd",
'_rev': "1-Ic9d83229f242d4133b7fae#6B187c43',
"gas": 267.
"temperature' : GB ,
"€lame™ : 9B1,
"f*ne_status'... 'Fire is Detached"
" Gas_status"... "Gas Leakage :ks 0.elected",
"exhaost_fan_status" ... "6 ning"

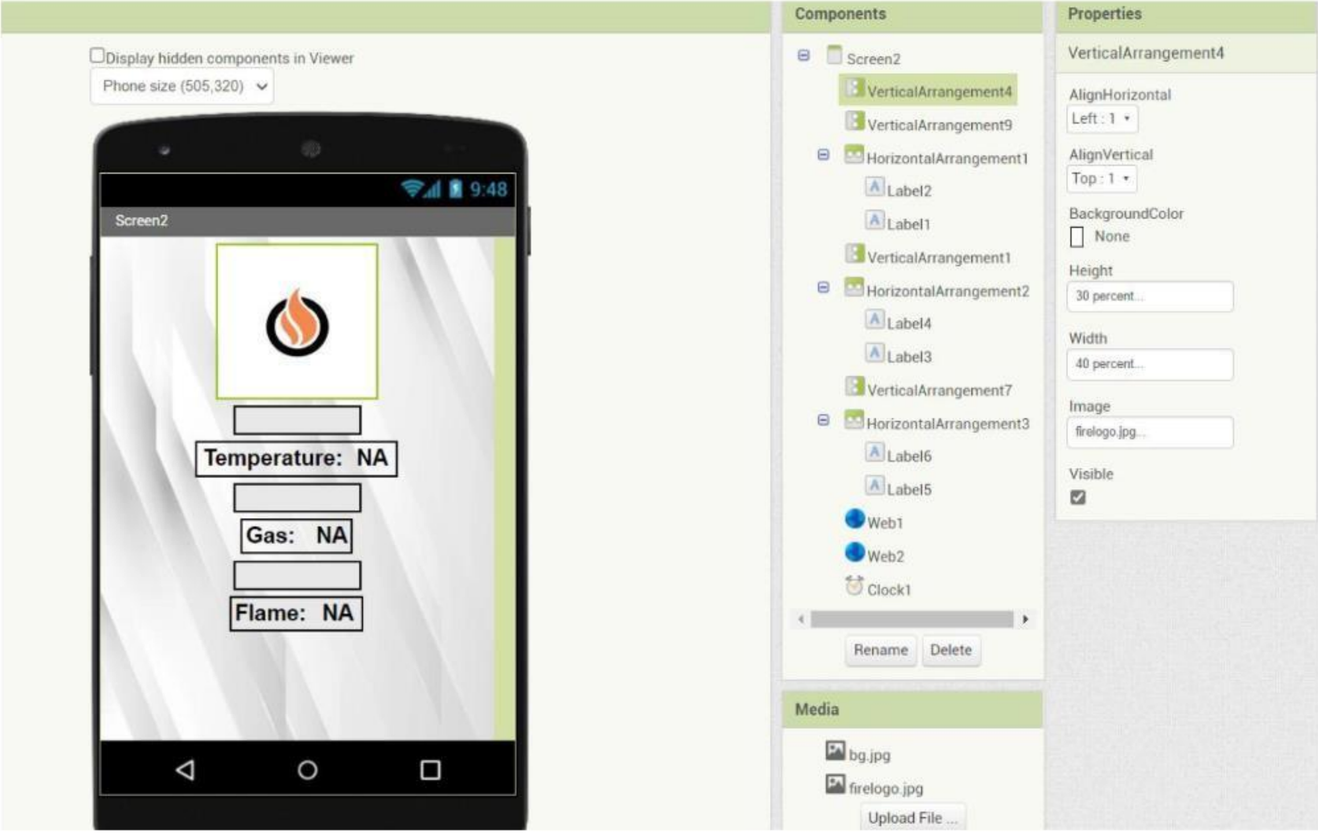
SPRINT – 4:

MIT APP INVENTER:





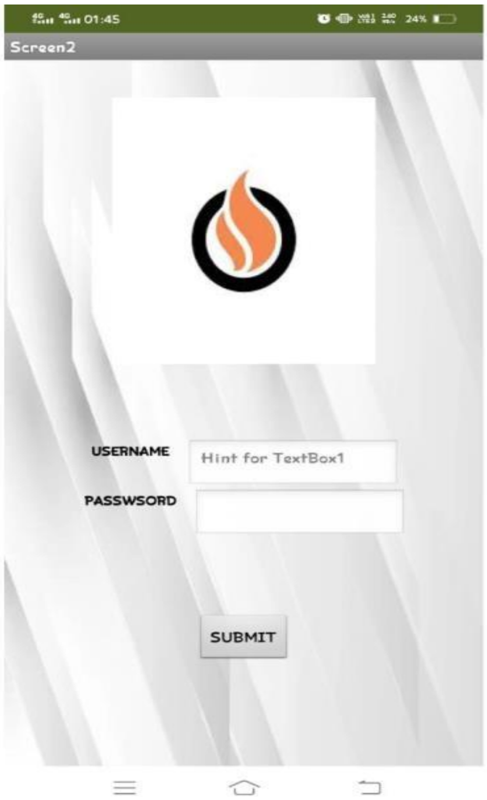




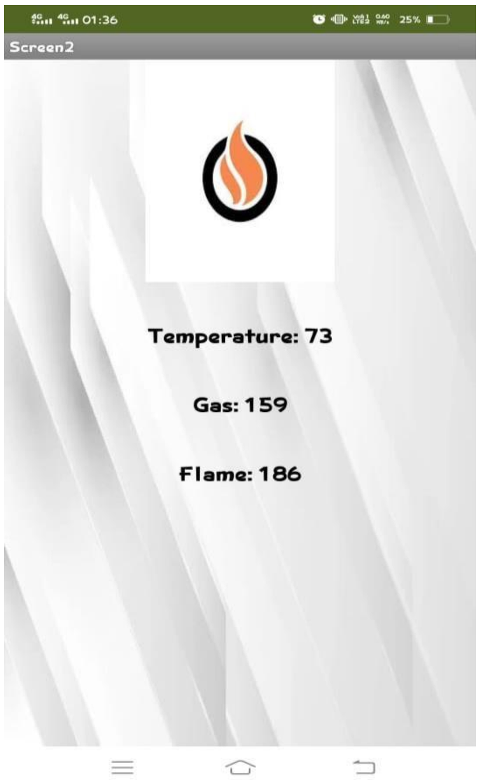
Components
Properties
Display hidden components in Viewer
Phone size (505,320)
Screen2
Temperature: NA
Gas: NA
Flame: NA
VerticalArrangement4
VerticalArrangement9
HorizontalArrangement1
Label2
Label1
VerticalArrangement1
HorizontalArrangement2
Label4
Label3
VerticalArrangement7
HorizontalArrangement3
Label6
Label5
Web1
Web2
Clock1
Rename
Delete
Media
bg.jpg
firelogo.jpg
Upload File ...
AlignHorizontal
Left : 1
AlignVertical
Top : 1
BackgroundColor
None
Height
30 percent...
Width
40 percent...
Image
firelogo.jpg...
Visible

MIT APP INVENTOR
trial
Blocks
Viewer
when Clock1 .Timer
do set Web1 . Url to "http://169.51.194.21:31266/sensor"
call Web2 .Get
when Web1 .GotText
url responseCode responseType responseContent
set Label3 . FontSize to look up in pairs key "gas"
pairs call Web1 .JsonTextDecode
jsonText get responseContent
notFound "not found"
set Label5 . FontSize to look up in pairs key "flame"
Show Warnings


01:40
24%
Screen1
INTELLIGENT FIRE MANAGEMENT SYSTEM


01:45
24%
Screen2
USERNAME
Hint for TextBox1
PASSWSORD
SUBMIT


01:46
24%
Screen2
USERNAME
kari.1914119@gct.ac.in
PASSWSORD
••••••
SUBMIT


01:36
25%
Screen2
Temperature: 73
Gas: 159
Flame: 186

RESULT

CPU USAGE:

The micro version of c++ is make the best use of the CPU. For every loop the program runs in one 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.

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

It automatically detect false fire breakout reducing unnecessary panic by using flow sensors we can confirm that the sprinkler system is working as it intended

All device status can be shown in a dashboard

Users can see the dashboard using a web application 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

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.

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.

APPENDIX

Esp32 - Microcontroller:

ESP32 is a series of low-cost, low-power system on a chip microcontroller 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).

24 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:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
# Provide your IBM Watson Device Credentials
organization = "gh6uoi"
deviceType = "fire"
deviceId = "fire123"
authMethod = "token"
authToken = "0123456789"
# print(cmd)
try :
    deviceOptions = {"org" : organization, "type" : deviceType, "id" : deviceId, "auth-method" : authMethod,
```

```
                    "auth-token" : authToken}
    deviceCli = ibmiotf.device.Client ( deviceOptions )
# ..............................................
except Exception as e :
    print ( "Caught exception connecting device: %s" % str ( e ) )
    sys.exit ()
# Connect and send a datapoint "hello" with value "world" into the cloud as an event of type "greeting"
10 times
deviceCli.connect ()


def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
def myCommandCallback(cmd) :
    print ( "Command received: %s" % cmd.data['command'] )
    status = cmd.data['command']
while True :
    # Get Sensor Data from DHT11
    temp = random.randint ( 0, 100 )
    gas = random.randint ( 60, 200 )
    flame = random.randint ( 60, 200 )
    data = {'temp' : temp, 'Gas' : gas,'Flame': flame}
    # print data
    def myOnPublishCallback() :
        print ( "Published Temperature = %s C" % temp, "Gas = %s %%" % gas, "Flame = %s %%" %
flame, "to IBM Watson")
    success = deviceCli.publishEvent ( "IoTSensor", "json", data, qos = 0, on_publish =
myOnPublishCallback )
    if not success :
```

```
        print ( "Not connected to IoTF" )
    time.sleep ( 1 )
    deviceCli.commandCallback = myCommandCallback
# Initialize GPIO
    if temp > 50 :
        print ( "buzzer is on" )
    else :
        print ( "buzzer is off" )
    if flame > 100 :
        print ( "sprinklers are on" )
    else :
        print ( "sprinklers are off" )
    if gas>100:
        print ( "exhaust fan is on" )
    else :
        print ( "exhaust fan is off" )
# Disconnect the device and application from the cloud
deviceCli.disconnect ()
```