R.M.K.ENGINEERING COLLEGE

(An Autonomous Institution)

R.S.M. Nagar, Kavaraipettai-601 206

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

PROJECT BASED EXPERIENTIAL LEARNING PROGRAM (NALAIYA THIRAN)

HAZARDOUS AREA MONITORING FOR INDUSTRIAL PLANT POWERED BY IOT

A PROJECT REPORT

Submitted by

ANUVARSHINI SS (111719106007)

BHUVANESHWARI S (111719106021)

FIONA M (111719106036)

GEETHIKA KN (111719106040)

TEAM ID: PNT2022TMID15984

INDEX

1. INTRODUCTION:

1.1 PROJECT OVERVIEW:

- Through this, we can monitor the temperature parameters of the hazardous areas in industrial plants.
- The area is integrated with smart beacon devices which will be broadcasting the temperature of that particular area.
- Every person working in those areas will be given smart wearable devices which will be acting as beacon scanners.
- Whenever the person goes near the beacon scanners he can view the temperature on his wearable device and if the temperature is high, he will receive the alerts to the mobile through SMS using API.
- Through this wearable device, the data is sent to the cloud and through the dashboard, the admins of that particular plant can view the data and take necessary precautions if required.

Project Flow:

- Sending random Humidity and Temperature values will be sent to the IBM IoT platform.
- Sensors values can be viewed in the Web Application.
- Notifies the admin the random values cross the threshold value.

To accomplish this, we have to complete all the activities and tasks listed below:

- Create and configure IBM Cloud Services
  - Create IBM Watson IoT Platform.
  - Create a device & configure the IBM IoT Platform.
  - Create Node-RED service.
  - Create a database in Cloudant DB to store location data.
- Develop a web Application using Node-RED Service.
  - Develop the web application using Node-RED.
- Develop a python script to publish the sensor data to the IBM IoT platform

2. LITERATURE SURVEY

S.NO	YEAR	TITLE	AUTHOR	PROJECT DESCRIPTION
1	2016	Implementation of hazardous chemical gas monitoring system using unmanned aerial vehicle (UAV)	Fadhil Mochammad; Aditya Rachman Putra; Bambang Riyanto Trilaksono	Unmanned aerial vehicle (UAV) with hexacopter platform is an effective tool to monitor the level of hazardous chemical gas. The ability to fly at low speed autonomously allows this system to map the hazardous chemical gas level and the hazardous chemical gas distribution in each section of an area. To do this task, an on-board data acquisition system is needed on the UAV to measure the hazardous chemical gas level based on the GPS position of the measurement. The user can interact with UAV to specify the scanning scenario.

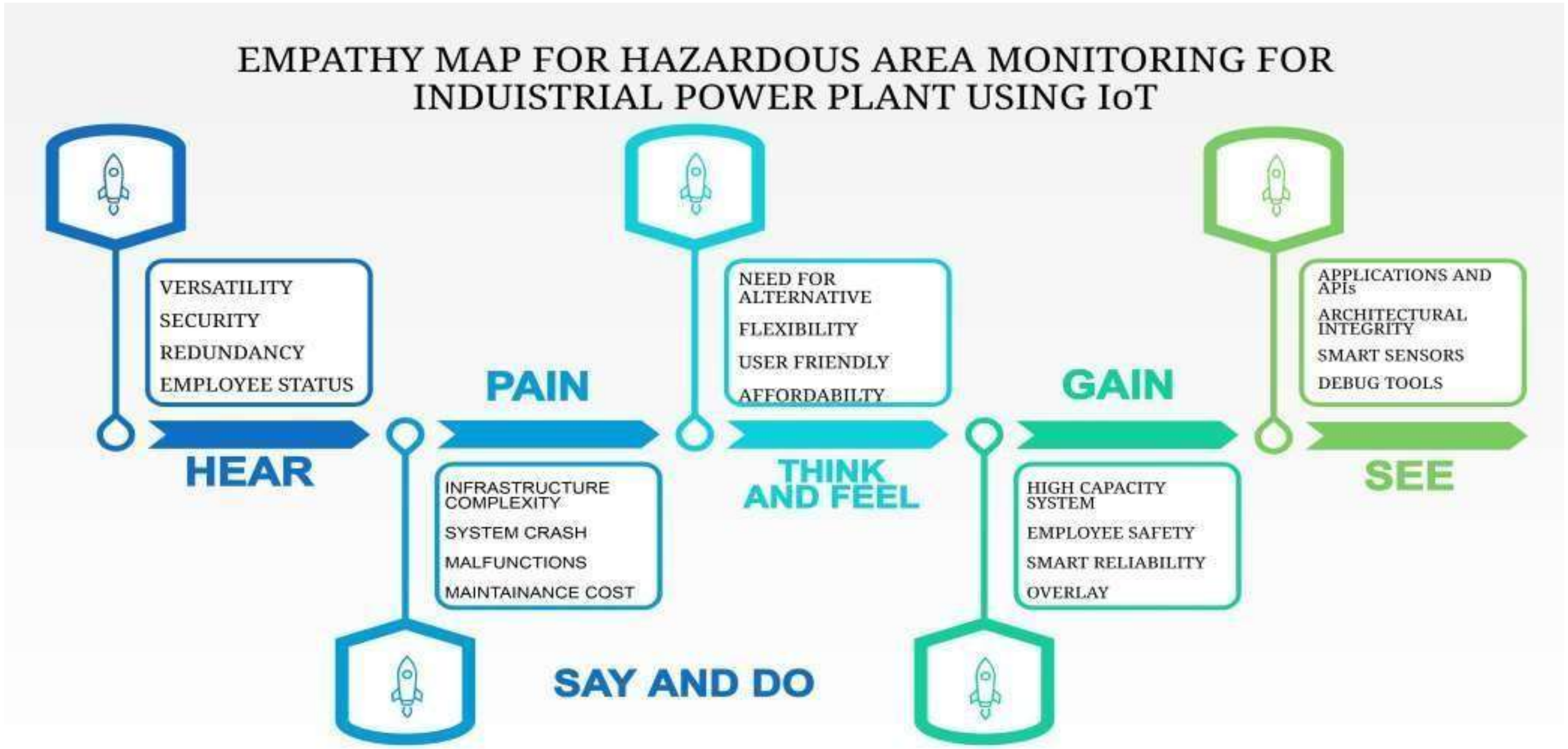
2	2021	Real-Time Design of HMI for Hazardous Gas Control and Monitoring System in Pakistani Mines, Natural Gas Areas and Fertilizer Plants	Misha Urooj Khan ; Aneela Shaheen ; Muhammad Zeeshan ; Asad-ur-Rehman ; Muhammad Adnan ; Malik Tayyab Rehman	Gas leakage is a serious issue in commercial areas and domestic buildings and a major cause of concern in bistros, populated neighborhoods, and automobiles that operate upon compressed natural gas (CNG). Because of the increased gas leaks, residential safety has been a serious concern in recent years. Hazardous gases such as propane and methane are flammable, and if contained in close proximity might trigger explosions Installing gas monitoring systems in sensitive areas is one of the preventative techniques for eliminating fatalities caused by gas leakage.

3	2017	MQTT based environment monitoring in factories for employee safety	Ravi Kishore Kodali; Aditya Valdas	Safety of employees, in any industry, especially at the factory level is a important aspects. In factories where working conditions are harsh and employees need to take great caution while going about their work, it is common for mishaps to occur.It is important that there is a measure of safety for the employees from any possible hazardous situations. As a solution to this problem, we propose a monitoring system to be installed in factories. With this system, we will be able to monitor critical safety parameters of the working environment in these factories so that we are well-aware of the safety situation and the possibility of occurence of any mishap. For the design of this system, we use an ESP8266 Wi-Fi chip enabled microcontroller NodeMCU.

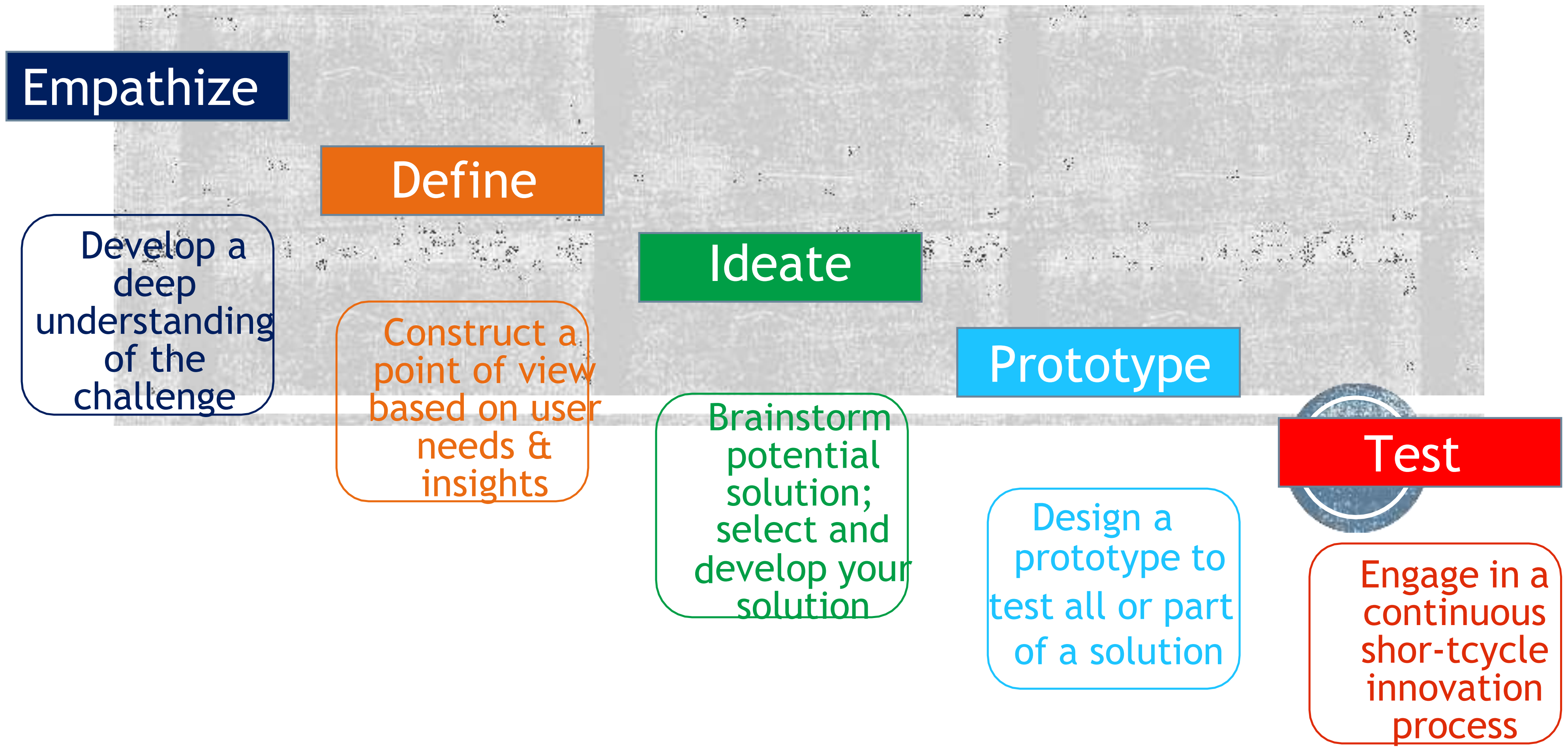
4	2009	Camera systems in hazardous locations to monitor and control the separation of liquid waste products in chemical plants	Ingo Emde ; Wolfgang Berner ; Klaus Mertens	It is essential to have high-quality measuring systems for chemical processes in order to obtain reliable data for the control system. For some process applications it is difficult to find a dependable method to measure important values that are vital for the end product. In the case of a liquid waste product processing facility the difficult measuring task is to monitor the separation of Anilin and water in a vessel. In this application, standard measuring methods cannot monitor the process reliably, especially during a start up phase. To avoid cost-intensive and timeconsuming monitoring by staff directly at the vessel, the task can be solved by using cameras to monitor the process. The additional problem is to design such a solution for a facility that is classified as a Zone 1 hazardous location.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP



3.2 IDEATION

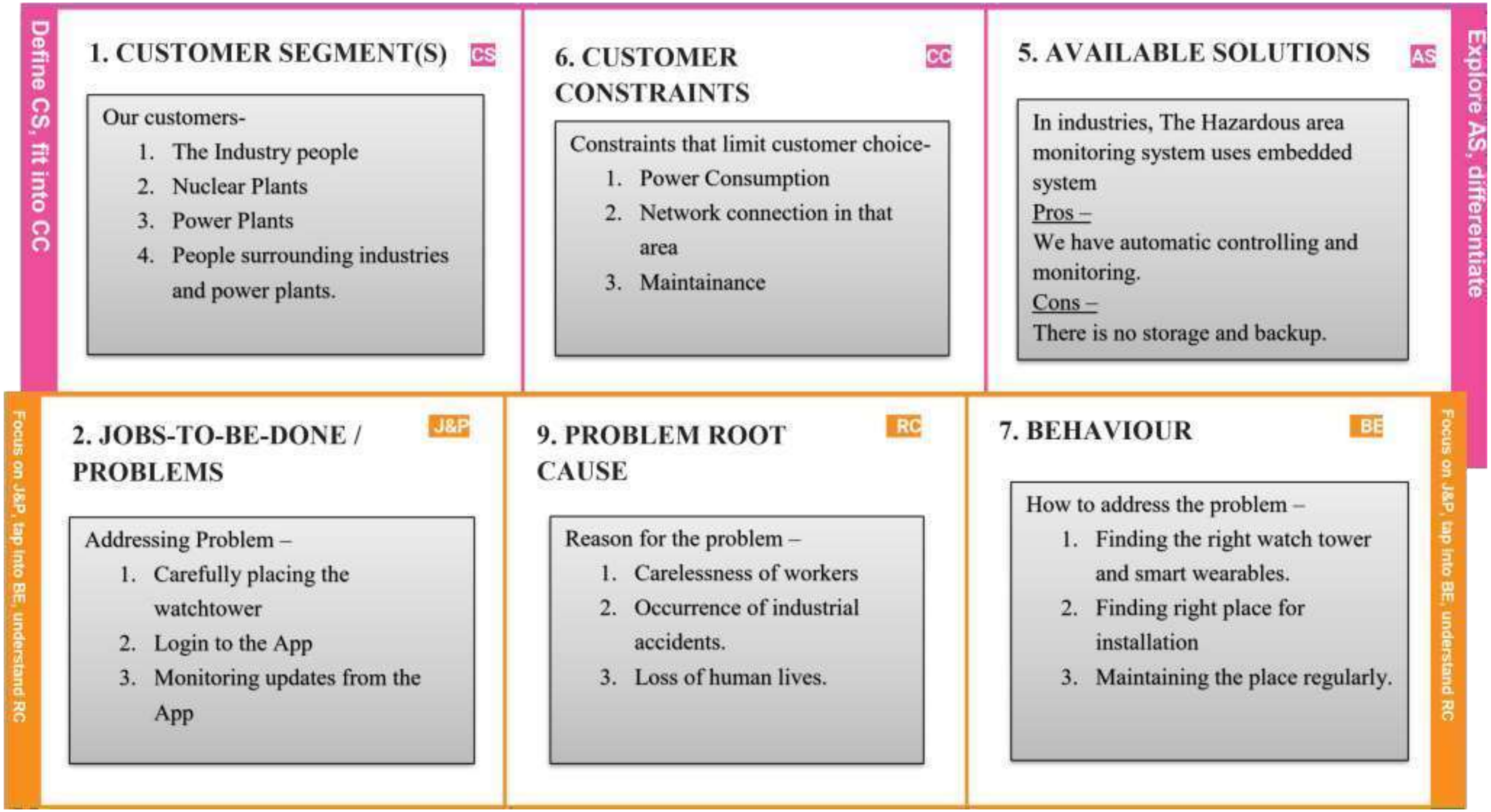

Empathize
Develop a deep understanding of the challenge
Define
Construct a point of view based on user needs & insights
Ideate
Brainstorm potential solution; select and develop your solution
Prototype
Design a prototype to test all or part of a solution
Test
Engage in a continuous shor-tcycle innovation process

3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1	Problem statement	Problems in continuous monitoring of temperature and communication in industrial power plant
2	Idea	The industrial power plants are integrated with smart temperature watch towers which will be sensing and broadcasting the temperature of that particular area. Every person working in those areas will be given smart wearable devices which will be acting as watch tower scanners. Whenever the person goes near it, he can view the temperature on his wearable device and if the temperature is high, he will receive the alerts to the mobile through SMS using API. Through this wearable device, the data is sent to the cloud database and through which the dashboard, the admins of that particular plant can view the data and take necessary actions if required.
3	Novelty	Every persons are given smart wearable devices and we use watch towers for advanced monitoring.
4	Social impact	Connectivity for workers are provided so that they can work safely.
		Equipment's are constantly monitored, so that negative impact on the environment are avoided.

5	Business Model	Can be implemented in different hazardous areas. Can make the wearables more advanced and customizable to ones need.
6	Scalability of the solution	By increasing the number of devices, this can be implemented in a commercial level. In future, other elements like radiation and gases can also be monitored.

3.4 PROPOSED SOLUTION FIT



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Our customers-

1. The Industry people
2. Nuclear Plants
3. Power Plants
4. People surrounding industries and power plants.

6. CUSTOMER CONSTRAINTS CC

Constraints that limit customer choice-

1. Power Consumption
2. Network connection in that area
3. Maintainance

5. AVAILABLE SOLUTIONS AS

In industries, The Hazardous area monitoring system uses embedded system

Pros –

We have automatic controlling and monitoring.

Cons –

There is no storage and backup.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Addressing Problem –

1. Carefully placing the watchtower
2. Login to the App
3. Monitoring updates from the App

9. PROBLEM ROOT CAUSE RC

Reason for the problem –

1. Carelessness of workers
2. Occurrence of industrial accidents.
3. Loss of human lives.

7. BEHAVIOUR BE

How to address the problem –

1. Finding the right watch tower and smart wearables.
2. Finding right place for installation
3. Maintaining the place regularly.

Focus on J&P, tap into BE, understand RC

3. TRIGGERS

TR

Industries are very much prone to fatal accidents due to huge usage of electrical equipment's and appliances. This leads to loss oh Human lives, damage of properties and injuries. This results in creating a monitoring system that can avoid such hazards.

4. EMOTIONS

EM

After using this the workers feel confident and be more concentered in work.

10. YOUR SOLUTION

SL

These accidents can be avoided by continuously monitoring the hazardous area and sending information through smart wearable devices to the workers and notifying them through mobile application.

8. CHANNELS OF BEHAVIOUR

CH

OFFLINE –

1. Watch towers
2. Wearable devices

ONLINE –

1. Cloud storage
2. Mobile application
3. Network connectivity

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements for the proposed solution:

Functional Requirements	**Sub Requirement**
Temperature sensor	To detect temperature of the particular area
Watch tower	To attract attention to a specific location.
Cloud service	To store and access data.
Wearable device	To notify the workers of hazardous area.
Alarm	To alert the workers in nearby sectors.
Mobile application	To communicate with the workers.
Raspberry pi	To process the data and for automation.

4.2 Non-functional Requirements:

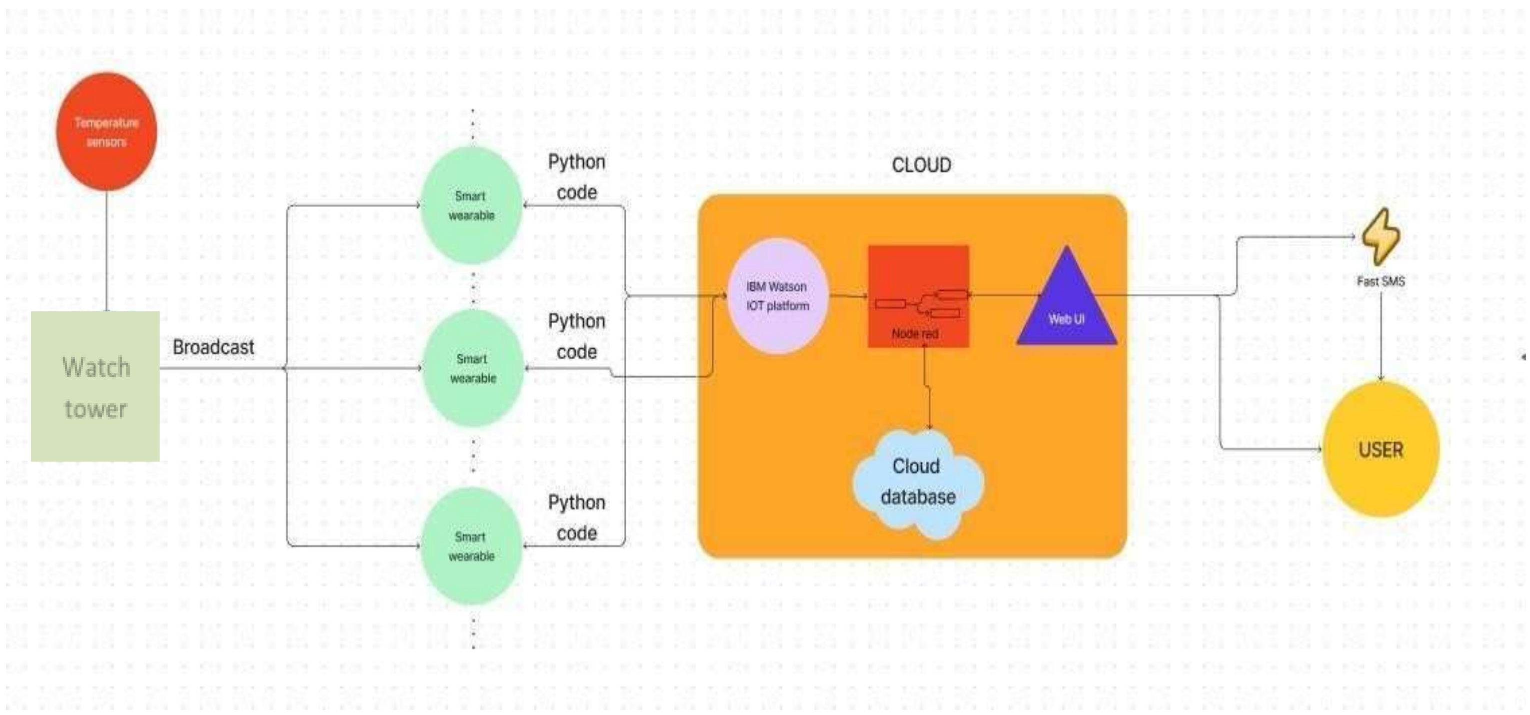
Following are the non-functional requirements of the proposed solution.

NFR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Availability of user-friendly wearable devices.
NFR-2	**Security**	Safety and security of the workers are ensured by installing monitoring beacons in various areas
NFR-3	**Reliability**	Data will be stored in secure cloud storage to prevent unethical hacking.
NFR-4	**Performance**	Server is made to perform well in all conditions.
NFR-5	**Availability**	Information is available through wearable devices and mobile application.
NFR-6	**Scalability**	Easily accessible with high reliability.

5. PROJECT DESIGN:

5.1 Data Flow Diagram:

A data flow diagram (DFD) maps out the flow of information for any process or system. They can be used to analyse an existing system or model a new one. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.



5.2 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Technician	Installation	USN-1	As a user, I must install the smart watch tower at points to ensure the entire area of the plant is covered.	A watch tower can be found inevery area of the plant.	High	Sprint-1
	Data Gathering	USN-2	The watch tower obtain the temperature of their respective area using sensors.	The temperature of areas within the plant is obtained.	High	Sprint-1
	Data Sync	USN-3	The watch tower send their data to the cloud in the real time which is in turn sent to nearby wearable devices and the administrator's dashboard	Data is sent to the cloud successfully and synced with other devices.	High	Sprint-1
Mobile User	Registration	USN-4	As a user, I can register for the applicationby entering my email, password and confirming my password.	I can access my account/ dashboard.	High	Sprint-1
		USN-5	As a User, I will receive confirmation emailonce I have registered for the application.	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-6	As a User, I can login to the application byentering email & password	I can register and access my account	High	Sprint-1
	Dashboard	USN-7	As a User, I can monitor the temperature and humidity.	I can access the account for monitoring the hazardous area	Medium	Sprint-2
End User	Alerting through message	USN-8	I can receive message in the form of visual notification and voice message.	I can detect the hazard and receive notification	High	Sprint-1
	SMS Notification	USN-9	I can get the alert message if the area has any Hazards.	I can be alerted through the SMS notification	Medium	Sprint-2
Web User	Monitoring	USN-10	As a Web User, I can detect the hazard through the website.	I can monitor the hazards like temperature, humidity, toxic gases.	High	Sprint-1

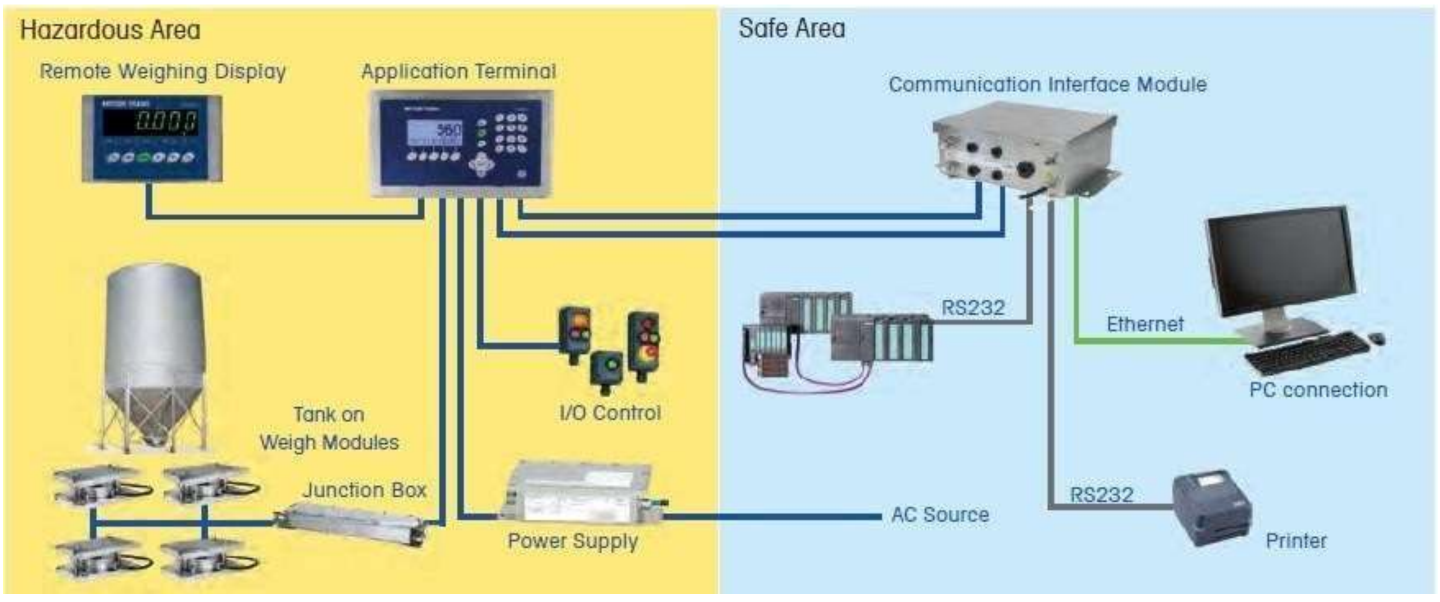
User Type	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Acceptance criteria**	**Priority**	**Release**
Customer care executive	Maintenance	USN-11	As an executive, I manage a team of representatives offering customer support.	I need a team of workers to manage the data.	Low	Sprint-3
Administrator	Admin Dashboard	USN-12	As an Administrator, I can able to access the data through the cloud.	I can access the data sent by the watch tower	High	Sprint-2
	Dashboard Customization	USN-13	As an Administrator, I can customize the dashboard to suit their personal requirements and priorities.	The admin can customize the UI for their dashboard.	Medium	Sprint-2

5.3 Solution Architecture

Solution architecture is a complex process with many sub-processes thatbridges the gap between business problems and technology solutions.

- With the help of this solution we can monitor the temperature parameters ofhazardous area in industrial power plants.
- These areas are integrated with small watch towers which will help inbroadcasting the temperature of that particular area.
- All the workers working in that area will be given smart wearable deviceswhich will be acting as watch tower scanners.
- When a person goes near these scanners he can view the temperature on hiswearable devices and if the temperature is high, he will receive alerts to the mobile through SMS using API.
- The data is sent to cloud through this wearable device and through the dashboard, the admins of that particular plant can view the data and takenecessary precaution if necessary

Solution Architecture Diagram:



5.4 Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Table-1 : Components & 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 /Raspberry Pi
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	Raspberry Pi
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 Service	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service
8.	Sensor nodes	Purpose of External API used in the application	IBM Weather API, Temperature Sensoretc.
9.	Alarm	Purpose of External API used in the application	Alarm and controller, wearable devices, watch tower
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Cloud Storage	Open source cloud storage data is available and accessible 24×7 from a remote location	ownCloud, Nextcloud, Pydio Cells andSeafile
2.	Security Implementations	Protection against aggressive work environment	Real Time Monitoring and ControlSystems
3.	Scalability	Extending coverage and generic solution across all components listed above	IIOT Sensors
4.	Availability	Real Time APIs for immediate data transfer	Python, Raspberry PI GUI
5.	Performance	Maintenance with most minimal intervention in On-Site Process	IIOT

References:

https://www.google.com/url?sa=t&source=web&rct=j&url=https://partheniumprojects.com/hazardous-area-monitoring-for-industrial-plants/&ved=2ahUKEwikh-To3f76AhVgyHMBHYa2At4QFnoECBIQAQ&usg=AOvVaw1ram1VVkt1RmZ7DfaIiiPK

https://www.google.com/url?sa=t&source=web&rct=j&url=https://partheniumprojects.com/hazardous-area-monitoring-for-industrial-plants/%23:~:text%3DEvery%2520device%2520will%2520be%2520acting,acting%2520as%2520a%2520beacon%2520scanner.&ved=2ahUKEwiTpaPt3f76AhWwHbcAHUsXDoYQFnoECEkQBQ&usg=AOvVaw1ram1VVkt1RmZ7DfaIiiPK

https://www.google.com/url?sa=t&source=web&rct=j&url=https://www.ijser.org/researchpaper/Internet-of-Things-for-Industrial-Monitoring- and-Control-Applications.pdf&ved=2ahUKEwiJ0Zi43v76AhUJGrcAHfM1BXsQFnoECDwQAQ&usg=AOvVaw38Hy6dTeMJVE5yX5S-VzYW

https://www.rejigdigital.com/blog/iiot-changing-condition-monitoring-for-industries/#:~:text=IIoT%20can%20intelligently%20monitor%20various,offshore%20drilling%20rigs%20or%20pipelines

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Use the below template to create product backlog and sprint schedule

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-1	Installation of beacons	USN-1	Instalment of beacons at necessary places	10	Medium	Anuvarshini SS Geethika KN
Sprint-1	Providing wearable devices	USN-2	All the workers are provided with the smart wearable device	10	Medium	Anuvarshini SS Bhuvaneshwari S
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Cloud Setup	USN-3	The real time data collected from environment by wearable devices are stored in cloud service network. The smart beacons will connect with the cloud network.	20	High	Fiona M Geethika KN
Sprint-3	Online monitoring Vis web	USN-4	Websites will be crated and connected to Cloud services.	15	Medium	Bhuvaneshwari S Fiona M
Sprint-4	Monitoring via mobile app	USN-5	Mobile Application will be created and fast messages will be generated to notify/alert the abnormality to the user.	20	High	Anuvarshini SS Bhuvaneshwari S Fiona M Geethika KN

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022		05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022		12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022		14 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 theteam’s average velocity (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 suchas Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

https://www.visual-

paradigm.com/scrum/sc

rum-burndown-chart/

https://www.atlassian.co

m/agile/tutorials/burndo

wn-charts

Reference:

https://www.atlassian.com/agile/proj

ect-management

https://www.atlassian.com/agile/tuto

rials/how-to-do-scrum-with-jira-

software

https://www.atlassian.com/agile/tuto

rials/epics

https://www.atlassian.com/agile/tuto

rials/sprints

https://www.atlassian.com/agile/proj

ect-management/estimation

https://www.atlassian.com/agile/tuto

rials/burndown-charts

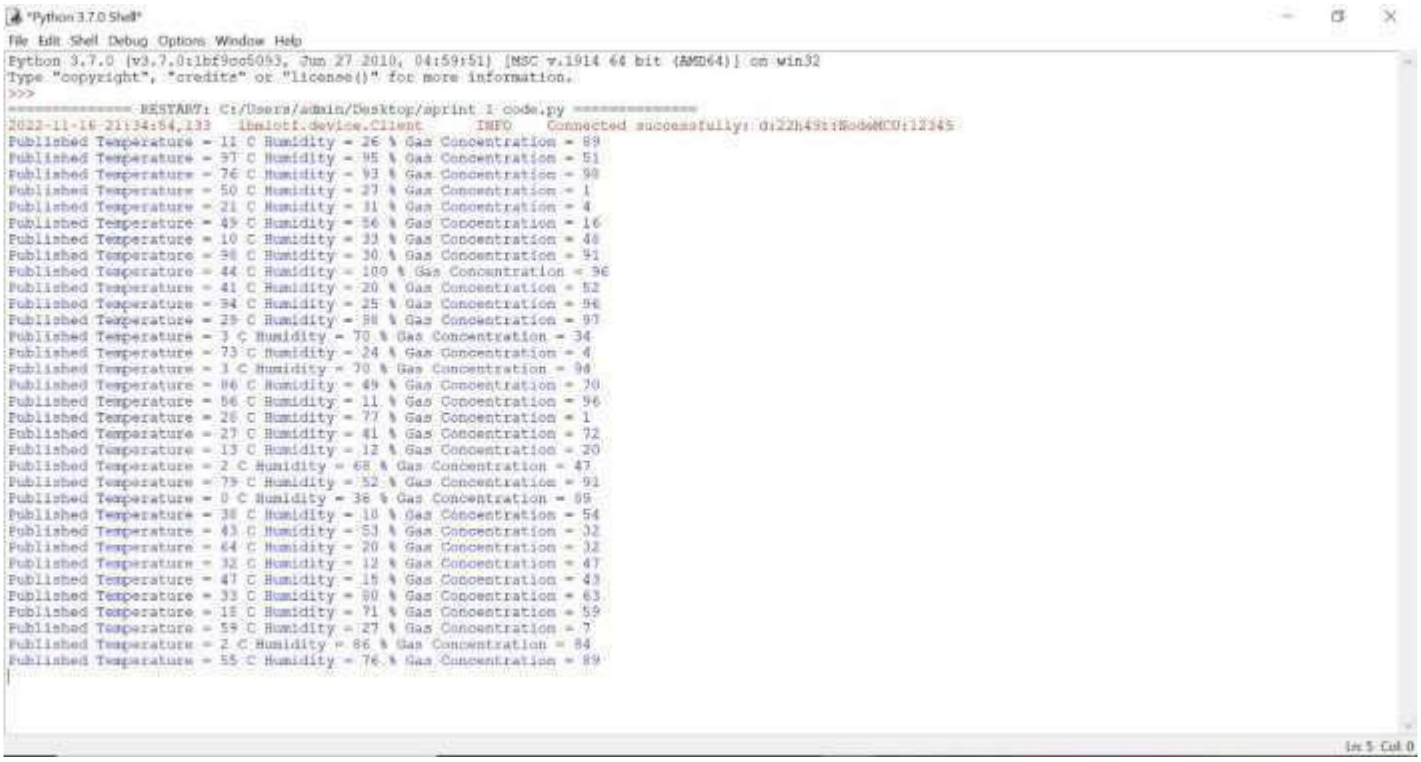
6.2 Sprint Delivery Schedule

6.2.1 Sprint 1

Code:

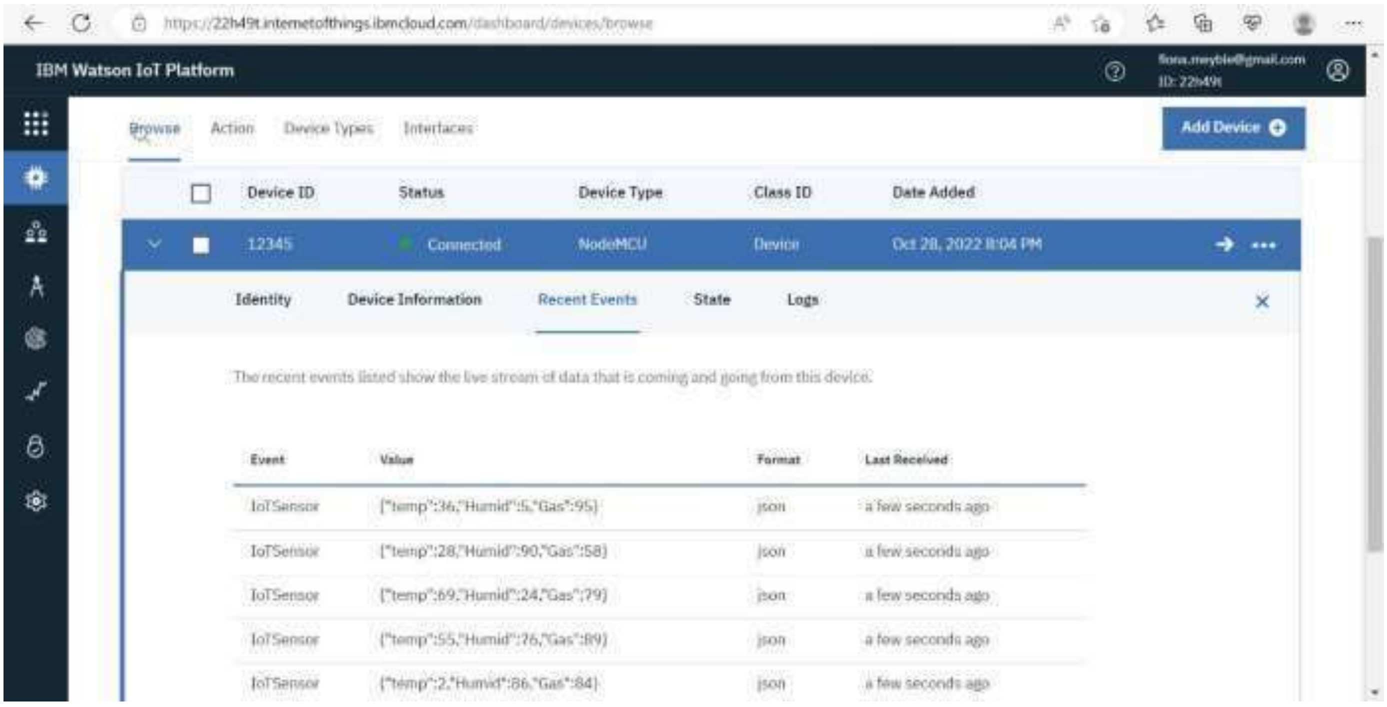
```
 import time
import sys
import random
import ibmiot.application
import ibmiot.device
organization = "22h49t"
deviceType = "NodeMCU"
deviceId = "12345"
authMethod = "use-token-auth"
authToken = "12345678"
try:
    deviceOptions = {"org": organization, "type": deviceType,"id": deviceId, "auth-method": authMethod, "auth-
token":authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
deviceCli.connect()
while True:
    temp=random.randint(0,100)
    Humid=random.randint(0,100)
    Gas=random.randint(0,100)

    data = { 'temp' : temp, 'Humid': Humid, 'Gas':gas }

    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %Humid, "Gas Concentration = %s"
%Gas )
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(10)
    deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
```

Output:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
=============== RESTART: C:/Users/admin/Desktop/sprint 1 code.py ===============
2022-11-16 21:34:54,133   ibmiotf.device.Client      INFO    Connected successfully: d:22h49t:NodeMCU:12345
Published Temperature = 11 C Humidity = 26 % Gas Concentration = 89
Published Temperature = 97 C Humidity = 95 % Gas Concentration = 51
Published Temperature = 76 C Humidity = 93 % Gas Concentration = 98
Published Temperature = 50 C Humidity = 27 % Gas Concentration = 1
Published Temperature = 21 C Humidity = 31 % Gas Concentration = 4
Published Temperature = 49 C Humidity = 56 % Gas Concentration = 16
Published Temperature = 10 C Humidity = 33 % Gas Concentration = 48
Published Temperature = 98 C Humidity = 30 % Gas Concentration = 91
Published Temperature = 44 C Humidity = 100 % Gas Concentration = 96
Published Temperature = 41 C Humidity = 20 % Gas Concentration = 52
Published Temperature = 34 C Humidity = 25 % Gas Concentration = 94
Published Temperature = 29 C Humidity = 98 % Gas Concentration = 97
Published Temperature = 3 C Humidity = 70 % Gas Concentration = 34
Published Temperature = 73 C Humidity = 24 % Gas Concentration = 4
Published Temperature = 3 C Humidity = 70 % Gas Concentration = 98
Published Temperature = 86 C Humidity = 49 % Gas Concentration = 70
Published Temperature = 56 C Humidity = 11 % Gas Concentration = 96
Published Temperature = 26 C Humidity = 77 % Gas Concentration = 1
Published Temperature = 27 C Humidity = 41 % Gas Concentration = 72
Published Temperature = 13 C Humidity = 12 % Gas Concentration = 20
Published Temperature = 2 C Humidity = 68 % Gas Concentration = 47
Published Temperature = 79 C Humidity = 52 % Gas Concentration = 91
Published Temperature = 0 C Humidity = 36 % Gas Concentration = 89
Published Temperature = 38 C Humidity = 10 % Gas Concentration = 54
Published Temperature = 43 C Humidity = 53 % Gas Concentration = 32
Published Temperature = 64 C Humidity = 20 % Gas Concentration = 32
Published Temperature = 32 C Humidity = 12 % Gas Concentration = 47
Published Temperature = 47 C Humidity = 15 % Gas Concentration = 43
Published Temperature = 33 C Humidity = 80 % Gas Concentration = 63
Published Temperature = 18 C Humidity = 71 % Gas Concentration = 59
Published Temperature = 59 C Humidity = 27 % Gas Concentration = 7
Published Temperature = 2 C Humidity = 86 % Gas Concentration = 84
Published Temperature = 55 C Humidity = 76 % Gas Concentration = 89
```

Cloud output:



https://22h49t.internetofthings.ibmcloud.com/dashboard/devices/browse

IBM Watson IoT Platform

fiona.meyble@gmail.com
ID: 22h49t

Browse   Action   Device Types   Interfaces   Add Device

Device ID	Status	Device Type	Class ID	Date Added
12345	Connected	NodeMCU	Device	Oct 28, 2022 8:04 PM

Identity   Device Information   Recent Events   State   Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
IoTSensor	{"temp":36,"Humid":5,"Gas":95}	json	a few seconds ago
IoTSensor	{"temp":28,"Humid":90,"Gas":58}	json	a few seconds ago
IoTSensor	{"temp":69,"Humid":24,"Gas":79}	json	a few seconds ago
IoTSensor	{"temp":55,"Humid":76,"Gas":89}	json	a few seconds ago
IoTSensor	{"temp":2,"Humid":86,"Gas":84}	json	a few seconds ago

6.2.2 Sprint 2

Code:

```
#IBM Watson IOT Platform #pip install wiotp-sdk import
wiotp.sdk.device import time
import random myConfig = {
"identity": {
"orgId": "22h49t",
"typeId": "NodeMCU", "deviceId":"12345"
},
"auth": {
"token": "12345678"
}
}
def myCommandCallback(cmd):
print("Message received from IBM IoT Platform: %s" %
cmd.data['command']) m=cmd.data['command']
client = wiotp.sdk.device.DeviceClient(config=myConfig,
logHandlers=None) client.connect()
while True: temp=random.randint(-20,125)
hum=random.randint(0,100) myData={'temperature':temp,
'humidity':hum}
client.publishEvent(eventId="status", msgFormat="json",
data=myData, qos=0, onPublish=None)
print("Published data Successfully: %s", myData)
client.commandCallback = myCommandCallback time.sleep(2)
client.disconnect()
```

SENSOR CODE:

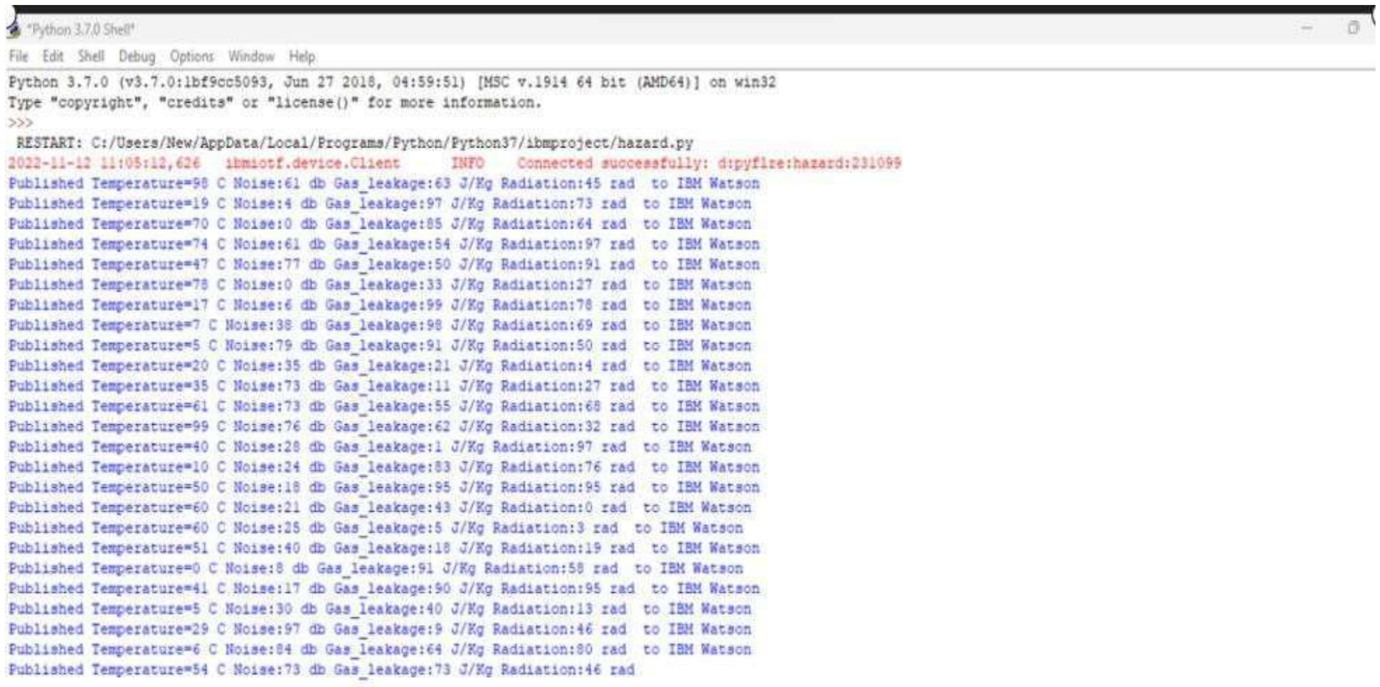
```
#include <dht.h>
#define dht_apin A0 // Analog Pin 0 is connected to DHT sensor
#define mqt_apin A1 // Analog Pin 1 is connected to MQT 135 sensor
dht DHT;
int sensorValue; void setup(){
Serial.begin(9600); //Serial port to communicate with Python code
Serial1.begin(9600); //Serial port to communicate with Wearable
device through Bluetooth (HC-05)
delay(500); //Delay to let system boot
}
void loop(){
DHT.read11(dht_apin); // read analog input pin 0(DHT11) sensorValue
= analogRead(mqt_apin); // read analog input pin 1(MQ135)
//Send Humidity status to Python Code
Serial.print("Current humidity = "); Serial.print(DHT.humidity);
Serial.print("% ");
//Send Temperature status to Python Code
Serial.print("temperature = "); Serial.print(DHT.temperature);
Serial.println("C ");
//Send AirQuality sensor value to Python code
Serial.print("AirQua="); Serial.print(sensorValue, DEC); Serial.println("
PPM")
```

6.2.3 Sprint 3

Code:

```
 import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
        organization="22h49t"
deviceType="NodeMCU"
deviceId="12345"
authMethod="token"
authToken="12345678"
def myCommandCallback(cmd):
print("Command received:%s" % cmd.data['command'])
status=cmd.data['command']
if status=="motoron":
print("Motor is ON")
else:
print("Motor is OFF")
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()
deviceCli.connect()
while True:
temp=random.randint(0,100)
noise=random.randint(0,100)
Gas=random.randint(0,100)
radn=random.randint(0,100)
data={'Temperature' :temp,'Noise':noise,'Gas_leakage':Gas,'Radiation':radn}
def myOnPublishCallback():
print("Published Temperature=%s C" %temp,"Noise:%s db"
%noise,"Gas_leakage:%s J/Kg" %Gas,"Radiation:%s rad "%radn,"to IBM
Watson")
success=deviceCli.publishEvent("IoTSensor","json",data,qos=0,on_publish=myO
nPublishCallback)
if not success:
print("Not connected to IoTF")
time.sleep(1)
deviceCli.commandCallback=myCommandCallback
deviceCli.disconnect()
```

Output:



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/New/AppData/Local/Programs/Python/Python37/ibmproject/hazard.py
2022-11-12 11:05:12,626   ibmiotf.device.Client      INFO    Connected successfully: d:pyfire:hazard:231099
Published Temperature=98 C Noise:61 db Gas_leakage:63 J/Kg Radiation:45 rad  to IBM Watson
Published Temperature=19 C Noise:4 db Gas_leakage:97 J/Kg Radiation:73 rad  to IBM Watson
Published Temperature=70 C Noise:0 db Gas_leakage:85 J/Kg Radiation:64 rad  to IBM Watson
Published Temperature=74 C Noise:61 db Gas_leakage:54 J/Kg Radiation:97 rad  to IBM Watson
Published Temperature=47 C Noise:77 db Gas_leakage:50 J/Kg Radiation:91 rad  to IBM Watson
Published Temperature=78 C Noise:0 db Gas_leakage:33 J/Kg Radiation:27 rad  to IBM Watson
Published Temperature=17 C Noise:6 db Gas_leakage:99 J/Kg Radiation:78 rad  to IBM Watson
Published Temperature=7 C Noise:38 db Gas_leakage:98 J/Kg Radiation:69 rad  to IBM Watson
Published Temperature=5 C Noise:79 db Gas_leakage:91 J/Kg Radiation:50 rad  to IBM Watson
Published Temperature=20 C Noise:35 db Gas_leakage:21 J/Kg Radiation:4 rad  to IBM Watson
Published Temperature=35 C Noise:73 db Gas_leakage:11 J/Kg Radiation:27 rad  to IBM Watson
Published Temperature=61 C Noise:73 db Gas_leakage:55 J/Kg Radiation:68 rad  to IBM Watson
Published Temperature=99 C Noise:76 db Gas_leakage:62 J/Kg Radiation:32 rad  to IBM Watson
Published Temperature=40 C Noise:28 db Gas_leakage:1 J/Kg Radiation:97 rad  to IBM Watson
Published Temperature=10 C Noise:24 db Gas_leakage:83 J/Kg Radiation:76 rad  to IBM Watson
Published Temperature=50 C Noise:18 db Gas_leakage:95 J/Kg Radiation:95 rad  to IBM Watson
Published Temperature=60 C Noise:21 db Gas_leakage:43 J/Kg Radiation:0 rad  to IBM Watson
Published Temperature=60 C Noise:25 db Gas_leakage:5 J/Kg Radiation:3 rad  to IBM Watson
Published Temperature=51 C Noise:40 db Gas_leakage:18 J/Kg Radiation:19 rad  to IBM Watson
Published Temperature=0 C Noise:8 db Gas_leakage:91 J/Kg Radiation:58 rad  to IBM Watson
Published Temperature=41 C Noise:17 db Gas_leakage:90 J/Kg Radiation:95 rad  to IBM Watson
Published Temperature=5 C Noise:30 db Gas_leakage:40 J/Kg Radiation:13 rad  to IBM Watson
Published Temperature=29 C Noise:97 db Gas_leakage:9 J/Kg Radiation:46 rad  to IBM Watson
Published Temperature=6 C Noise:84 db Gas_leakage:64 J/Kg Radiation:80 rad  to IBM Watson
Published Temperature=54 C Noise:73 db Gas_leakage:73 J/Kg Radiation:46 rad
```

6.2.4 Sprint 4

Code:

```
#include <DHT.h>
WiFiClient wifiClient;
String data3;
#define DHTTYPE DHT11
#define DHTPIN 4
#define MQTPIN 34
DHT dht(DHTPIN, DHTTYPE);
#define ORG "22h49t"
#define DEVICE_TYPE "NodeMCU"
#define DEVICE_ID "NodeMCU"
#define TOKEN "12345678"
#define speed 0.034
void callback(char* topic, byte* playload, unsigned int payloadLength);
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, callback , wifiClient);
void publishData();
String command;
String data = "";
long duration;
float dist;
void setup()
{
Serial.begin(115200);
dht.begin();
wifiConnect();
mqttConnect();
}
void loop() {
publishData();
delay(500);
if (!client.loop()) {
mqttConnect();
}
}
void wifiConnect() {
Serial.print("Connecting to "); Serial.print("Wifi");
WiFi.begin("JerroldWi-Fi","75779901");
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.print("WiFi connected, IP address: "); Serial.println(WiFi.localIP());
}
void mqttConnect() {
if (!client.connected()) {
```

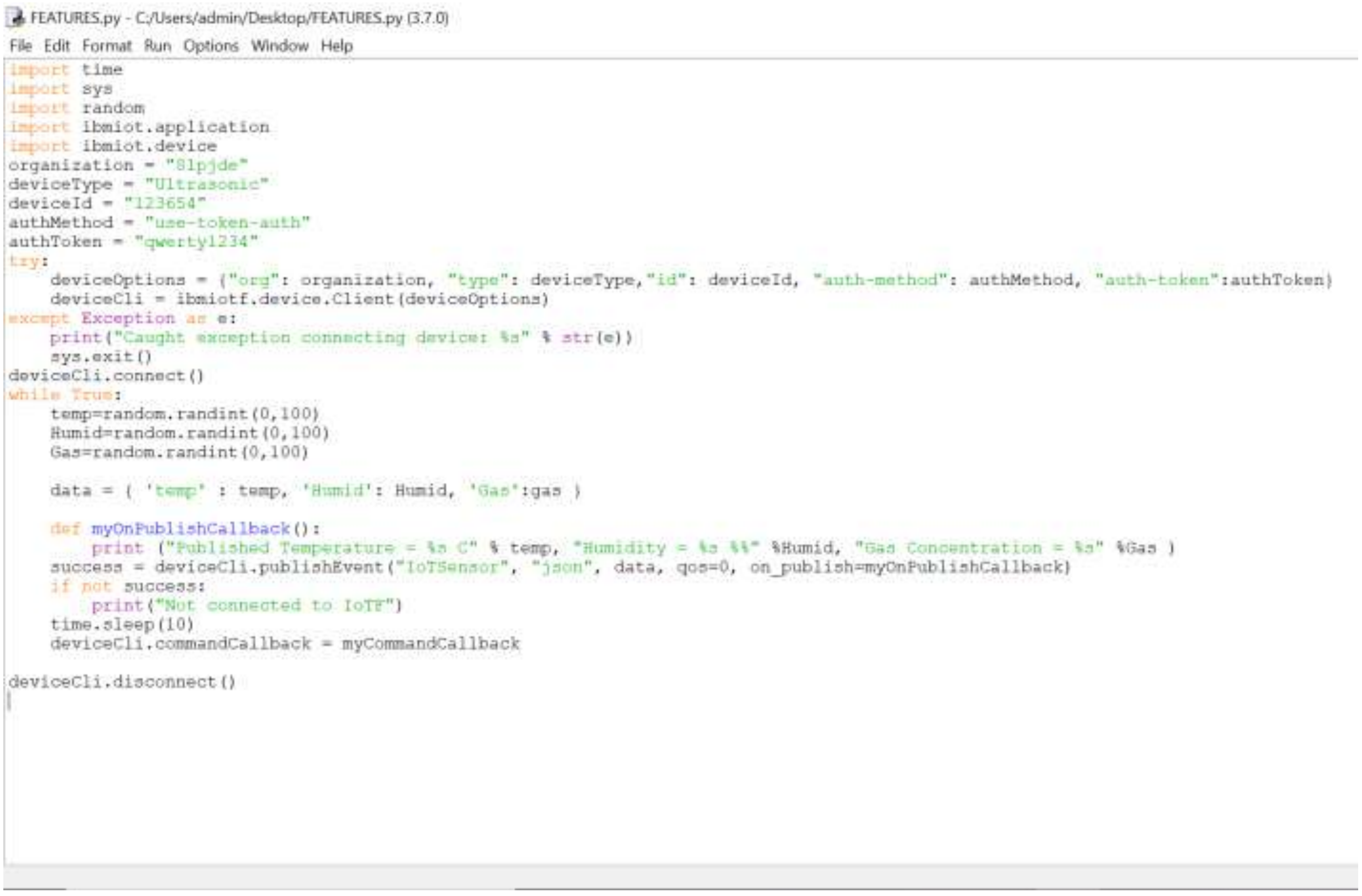
```
Serial.print("Reconnecting MQTT client to "); Serial.println(server);
while (!client.connect(clientId, authMethod, token)) {
Serial.print(".");
delay(500);
}
initManagedDevice();
Serial.println();
}
}
void initManagedDevice() {
if (client.subscribe(topic)) {
Serial.println("IBM subscribe to cmd OK");
} else {
Serial.println("subscribe to cmd FAILED");
}
}
void publishData()
{
int sensorValue = analogRead(MQTPIN); //MQT 135 connected to GPIO 34 (Analog
ADC1_CH6)
Serial.print("AirQua=");
Serial.print(sensorValue, DEC);
Serial.println(" PPM");
float humid = dht.readHumidity();
float temp = dht.readTemperature(true);
String payload = "{\"Humidity\":";
payload += humid;
payload += "}";
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish OK");
}
payload = "{\"Temperature\":";
payload += temp;
payload += "}";
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish OK");
}
payload = "{\"AirQuality\":";
payload += String(sensorValue);
payload += "}";
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish OK");
}
}
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++) {
dist += (char)payload[i];
}
Serial.println("data:" + data3);
if (data3 == "lighton") {
Serial.println(data3);
}
data3 = "";
```

7. CODING & SOLUTIONING

7.1 FEATURE 1

STEP 1:

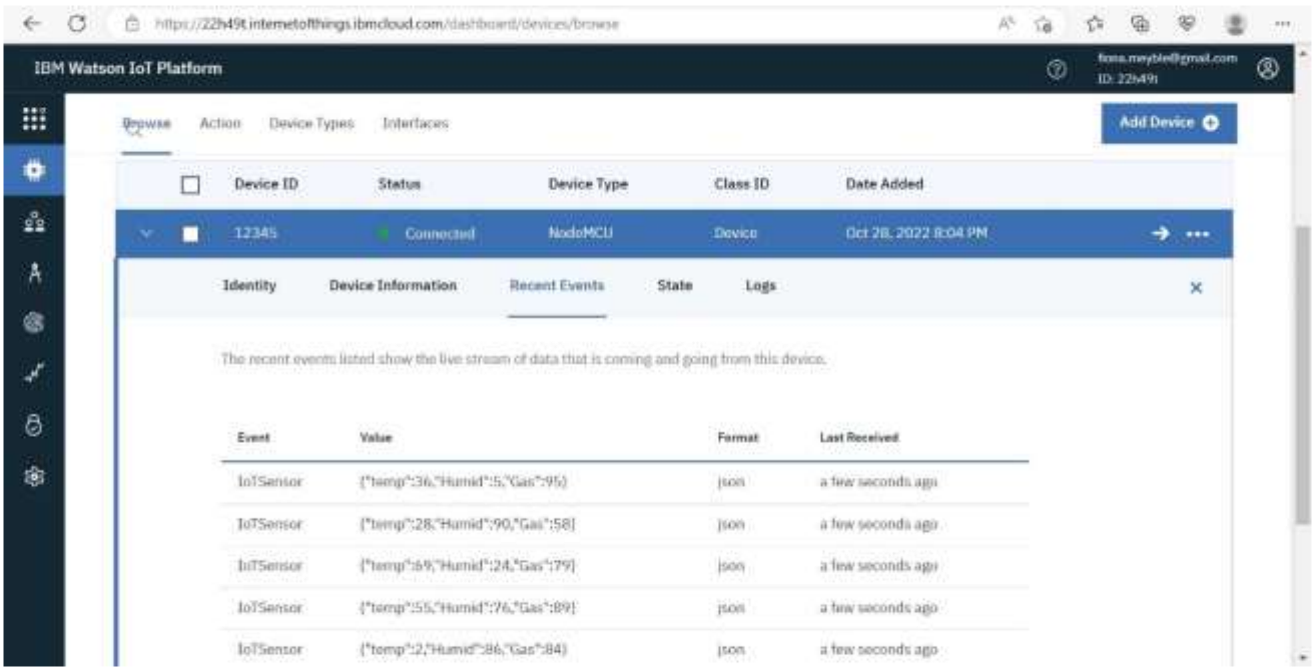
1. Find the temperature and humidity of the particular region where hazardous activities occur in industrial power plant.
2. Connect it to IBM waston IOT platform
3. **3.** Get the output in recent events.


FEATURES.py - C:/Users/admin/Desktop/FEATURES.py (3.7.0)

```python
import time
import sys
import random
import ibmiot.application
import ibmiot.device
organization = "81pjde"
deviceType = "Ultrasonic"
deviceId = "123654"
authMethod = "use-token-auth"
authToken = "qwerty1234"
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()
deviceCli.connect()
while True:
    temp=random.randint(0,100)
    Humid=random.randint(0,100)
    Gas=random.randint(0,100)

    data = { 'temp' : temp, 'Humid': Humid, 'Gas':gas }

    def myOnPublishCallback():
        print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %Humid, "Gas Concentration = %s" %Gas )
    success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(10)
    deviceCli.commandCallback = myCommandCallback

deviceCli.disconnect()
```

Step 2:

1. Output displayed on IBM IOT platform
2. By changing the user details we publish our code to IBM IOT Watson platform and our results are displayed in recent events



Device ID	Status	Device Type	Class ID	Date Added
12345	Connected	NodeMCU	Device	Oct 28, 2022 8:04 PM

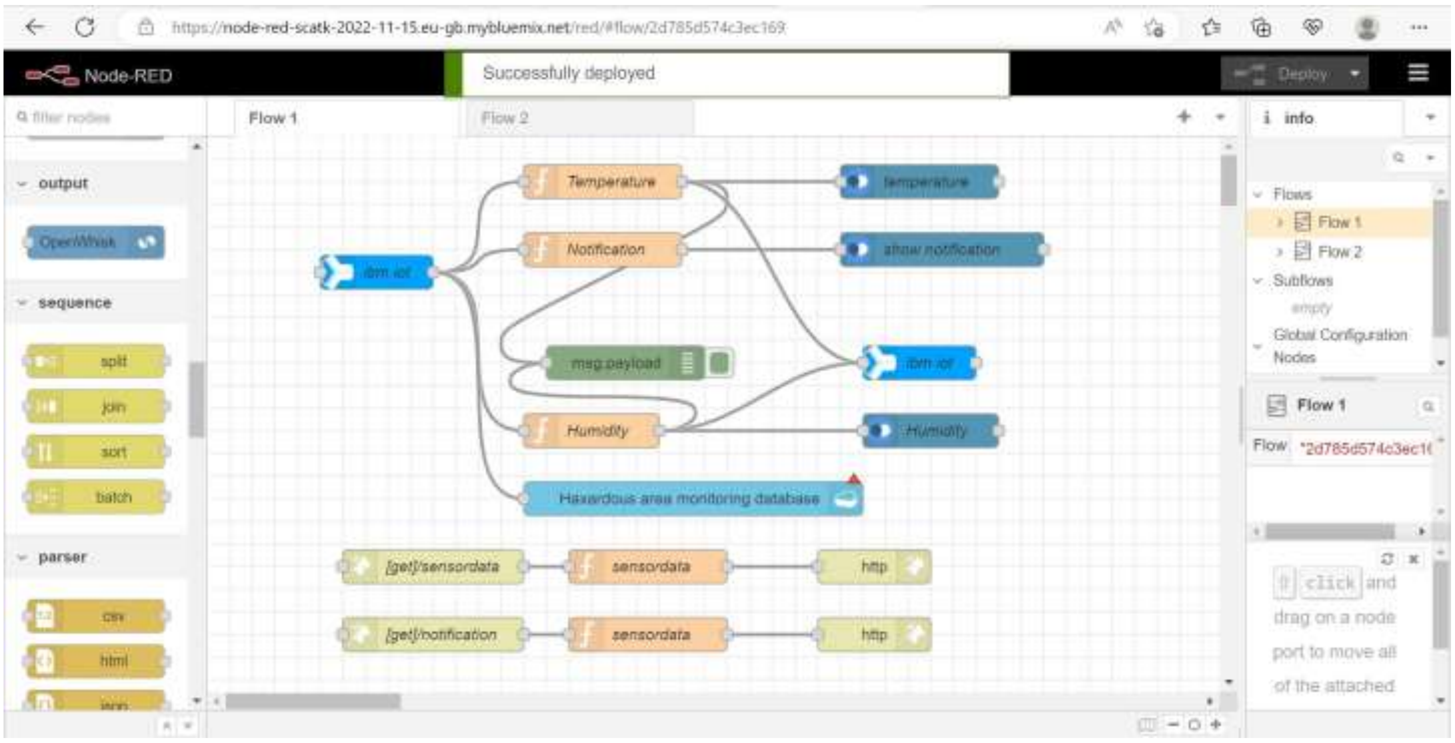
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
IoTSensor	{"temp":36,"Humid":5,"Gas":95}	json	a few seconds ago
IoTSensor	{"temp":28,"Humid":90,"Gas":58}	json	a few seconds ago
IoTSensor	{"temp":69,"Humid":24,"Gas":79}	json	a few seconds ago
IoTSensor	{"temp":55,"Humid":76,"Gas":89}	json	a few seconds ago
IoTSensor	{"temp":2,"Humid":86,"Gas":84}	json	a few seconds ago

7.2 FEATURE 2

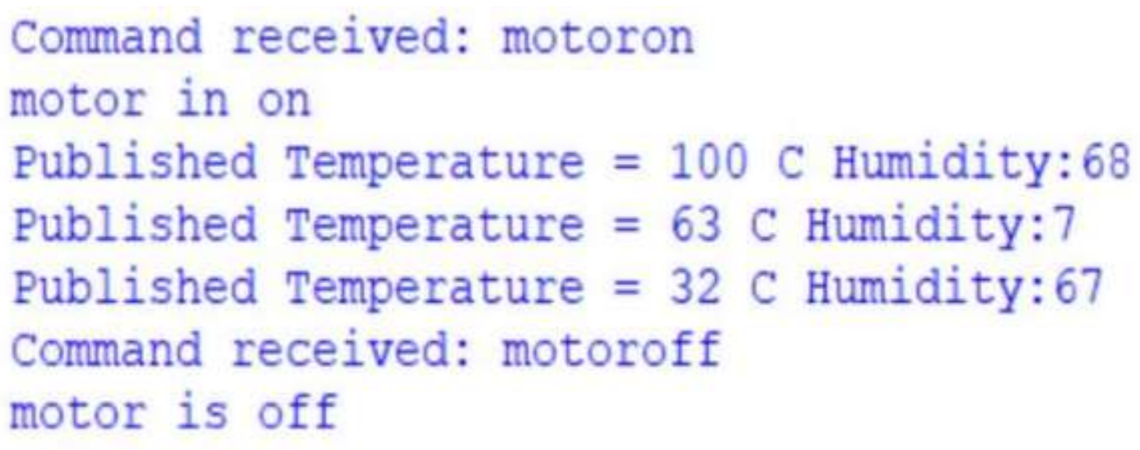
STEP 1:

1. Create node red service
2. Perform necessary connections to receive desired output.
3. Connect the node red to IBM waston IOT Platform.
4. Get the output in recent events



Step 2:

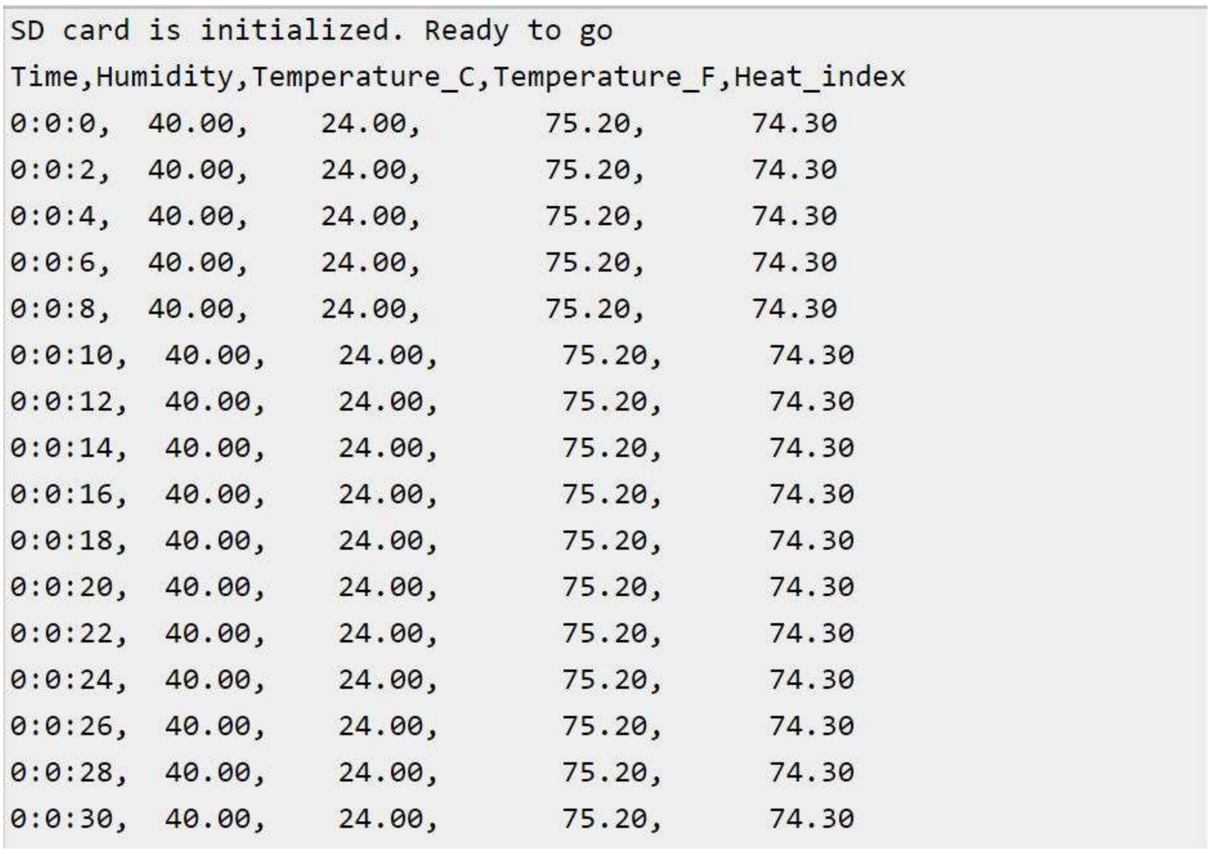
1. Connect our device to mobile app
2. Get Output in the output screen



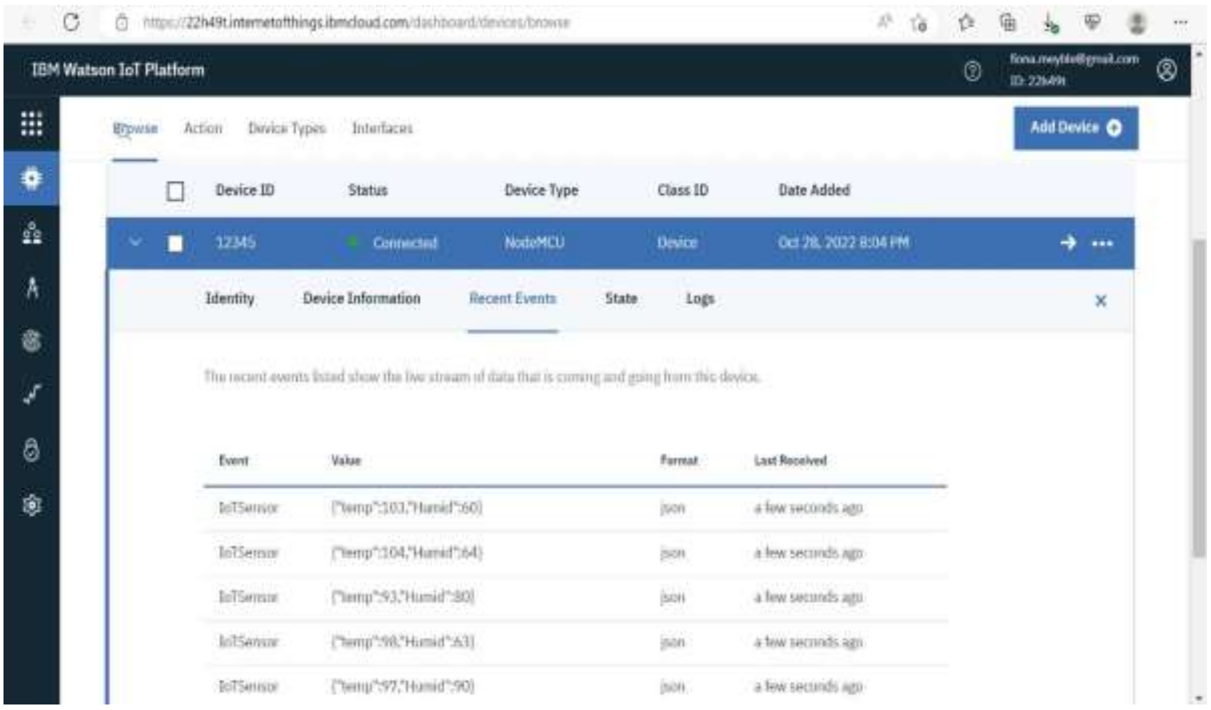
```
Command received: motoron
motor in on
Published Temperature = 100 C Humidity:68
Published Temperature = 63 C Humidity:7
Published Temperature = 32 C Humidity:67
Command received: motoroff
motor is off
```

8. TESTING

8.1 Test Case



```
SD card is initialized. Ready to go
Time,Humidity,Temperature_C,Temperature_F,Heat_index
0:0:0,  40.00,    24.00,       75.20,      74.30
0:0:2,  40.00,    24.00,       75.20,      74.30
0:0:4,  40.00,    24.00,       75.20,      74.30
0:0:6,  40.00,    24.00,       75.20,      74.30
0:0:8,  40.00,    24.00,       75.20,      74.30
0:0:10,  40.00,    24.00,       75.20,      74.30
0:0:12,  40.00,    24.00,       75.20,      74.30
0:0:14,  40.00,    24.00,       75.20,      74.30
0:0:16,  40.00,    24.00,       75.20,      74.30
0:0:18,  40.00,    24.00,       75.20,      74.30
0:0:20,  40.00,    24.00,       75.20,      74.30
0:0:22,  40.00,    24.00,       75.20,      74.30
0:0:24,  40.00,    24.00,       75.20,      74.30
0:0:26,  40.00,    24.00,       75.20,      74.30
0:0:28,  40.00,    24.00,       75.20,      74.30
0:0:30,  40.00,    24.00,       75.20,      74.30
```



Device ID	Status	Device Type	Class ID	Date Added
12345	Connected	NodeMCU	Device	Oct 28, 2022 8:04 PM

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
IoTSensor	{"temp":103,"Humid":60}	json	a few seconds ago
IoTSensor	{"temp":104,"Humid":64}	json	a few seconds ago
IoTSensor	{"temp":93,"Humid":80}	json	a few seconds ago
IoTSensor	{"temp":98,"Humid":63}	json	a few seconds ago
IoTSensor	{"temp":97,"Humid":90}	json	a few seconds ago

8.2 USER ACCEPTANCE TESTING

PURPOSE OF DOCUMENT

The purpose of this document is to briefly explain the test coverage and open issues of the Hazardous area monitoring for industrial plant powered by IOT project at the time of the release to User Acceptance Testing (UAT).

DEFECT ANALYSIS

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	8	4	2	3	20
Duplicate	1	0	3	0	4
External	3	4	0	1	6
Fixed	10	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	4	1	1	8
Totals	22	14	13	26	75

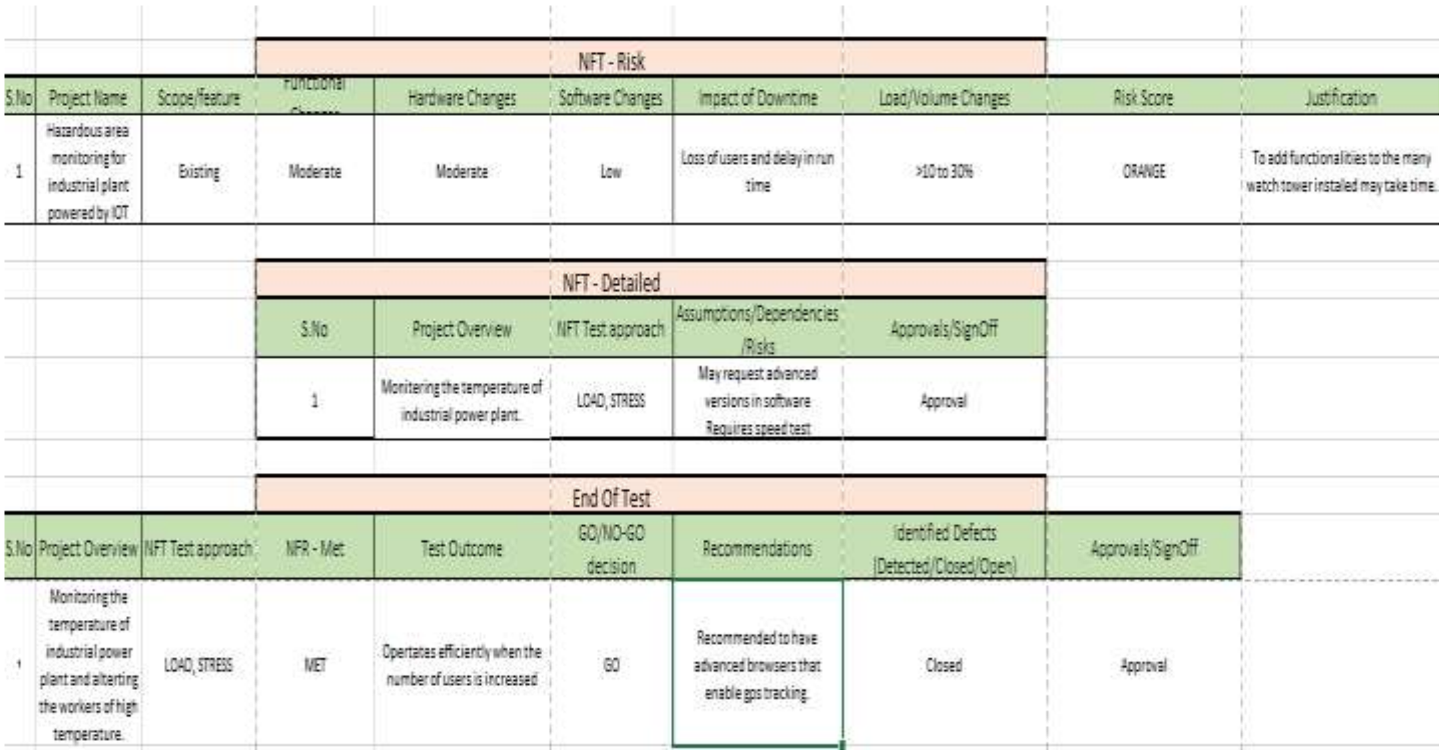
TEST CASE ANALYSIS

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	40	0	0	40
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9. RESULT

PERFORMANCE METRICS



NFT - Risk

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score	Justification
1	Hazardous area monitoring for industrial plant powered by IOT	Existing	Moderate	Moderate	Low	Loss of users and delay in run time	>10 to 30%	ORANGE	To add functionalities to the many watch tower instaled may take time.

NFT - Detailed

S.No	Project Overview	NFT Test approach	Assumptions/Dependencies /Risks	Approvals/SignOff
1	Monitering the temperature of industrial power plant.	LOAD, STRESS	May request advanced versions in software Requires speed test	Approval

End Of Test

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Monitoring the temperature of industrial power plant and alterting the workers of high temperature.	LOAD, STRESS	MET	Opertates efficiently when the number of users is increased	GO	Recommended to have advanced browsers that enable gps tracking.	Closed	Approval

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- With the help of watch tower wide range of area can be monitored easily.
- It provides security to the people working in the area with the help of cloud server and sensor is used to note the temperature change.
- Improved route performance. The wearables are more advanced and customizable to ones need.
- This system is flexible, it is user friendly and affordable.
- Provides safety to all the employees working and provides smart reliability overlay to overall system.

DISADVANTAGES:

- Maintenance cost of the system is high, replacing the defected components can crash the system.
- There can be a delay in sending alerts to the mobile through SMS using API.
- The carelessness of the workers in that particular plant, without viewing the data they cannot take necessary actions which leads to an accident.

11. CONCLUSION

Currently, IoT is present and gaining more traction in a lot of fields, and one of the most important field is industrial applications. There are a huge number of ways in which industries can make use of loT to improve working conditions, efficiency, cutting costs and improving the overall growth of the sector. However, hazard monitoring and mitigation is often overlooked in industrial areas.

Therefore, this project specifically aims to make use of loT to actively monitor and analyze various factors in a typical heavy industrial zone like temperature and levels of gases in the environment. If the above parameters exceed the recommended safe values, the system can track the same and issue alerts. Also, the data generated in real time can provide important information about how smoothly the work is going on in different zones.

This system can be deployed in many industrial areas like mining, under- ground factories, metal refineries, automatic welding factories and even heavy parts production lines. It will help to provide a safe and efficient working environment in such areas, while also opening new paths to improve the safety parameters of these places.

12. FUTURE SCOPE

Hazardous substances, particularly flammable or toxic, are increasingly used or produced in industrial processes. This involves many risks such as gas leaks, which can be harmful and even fatal to humans. Therefore, it is necessary to monitor the occurrence and concentration of various gases.

Area Monitoring helps safety personnel to protect health and life of coworkers in places and situations where fix systems cannot be used. These working environments are mostly very tough. Invisible dangers, hazardous substances, particularly flammable or toxic may cause many risks such as gas leaks, harmful and even fatal to humans' health. Safety specialists struggle daily with complexity and challenges to keep the industrial facilities, but above all – people, well-protected.

Gas detectors require attention, administration, record-keeping, and more-resulting in interruptions and logistical challenges that distract from larger safety objectives.

Therefore, it is so crucial to use a solution that combines the simplicity of a smart home device and delivers monitoring capabilities you can truly rely on.

13. APPENDIX

SOURCE CODE

```
#include <SD.h>
#include "DHT.h"

#define DHTPIN 8
#define DHTTYPE DHT22
 long seconds=00;
long minutes=00;
long hours=00;
 int CS_pin = 10;

DHT dht(DHTPIN, DHTTYPE);
File sd_file;


void     setup()          {
Serial.begin(9600);
pinMode(CS_pin,    OUTPUT);
dht.begin();
  // SD Card Initialization
if (SD.begin())  {
    Serial.println("SD card is initialized. Ready to go");
  }
else  {
    Serial.println("Failed");
return;
  }

  File sd_file = SD.open("data.txt", FILE_WRITE);
   if (sd_file)  {
Serial.print("Time
");
    Serial.print(",");
    Serial.print("Humidity");
    Serial.print(",");
    Serial.print("Temperature_C");
Serial.print(",");
    Serial.print("Temperature_F");
    Serial.print(",");
    Serial.println("Heat_index");
     sd_file.print("Time");
sd_file.print(",");
sd_file.print("Humidity");
sd_file.print(",");
sd_file.print("Temperature_C");
sd_file.print(",");
sd_file.print("Temperature_F");
sd_file.print(",");
    sd_file.println("Heat_index");
  }   sd_file.close(); //closing the
file
}
void loop()  {
 File sd_file = SD.open("data.txt", FILE_WRITE);
if (sd_file)  {     senddata();
  }
  // if the file didn't open, print an error:
else  {
    Serial.println("error opening file");
  }
  delay(1000);
}  void senddata()
{
  for(long seconds = 00; seconds < 60; seconds=seconds+2)  {
    float temp = dht.readTemperature(); //Reading the temperature as Celsius
and storing in temp
    float hum = dht.readHumidity();      //Reading the humidity and storing in
hum     float fah = dht.readTemperature(true);
    float heat_index = dht.computeHeatIndex(fah, hum);

    sd_file.print(hours);
sd_file.print(":");
sd_file.print(minutes);
sd_file.print(":");
sd_file.print(seconds);     sd_file.print(",
");     sd_file.print(hum);
sd_file.print(",    ");
sd_file.print(temp);     sd_file.print(",
");     sd_file.print(fah);
sd_file.print(",      ");
sd_file.println(heat_index);

    Serial.print(hours);
    Serial.print(":");
    Serial.print(minutes);
Serial.print(":");
    Serial.print(seconds);
    Serial.print(",  ");
    Serial.print(hum);
    Serial.print(",    ");
    Serial.print(temp);
    Serial.print(",       ");
    Serial.print(fah);
    Serial.print(",      ");
    Serial.println(heat_index);
     if(seconds>=58)       {
minutes= minutes + 1;
    }     if (minutes>59)
{       hours = hours + 1;
minutes = 0;
    }      sd_file.flush(); //saving
the file
     delay(2000);
}
  sd_file.close();   //closing the file
}
```

GITHUB AND DEMO LINK

https://github.com/IBM-EPBL/IBM-Project-30266-1660143065

Simulation link:

https://wokwi.com/projects/322324536938725971

Demo link:

https://drive.google.com/file/d/1lYZwyt1yWv1Rl0zv9ltRihxVwXQ1Qgff/view?usp=drivesdk