DATE	19.11.2022
TEAM ID	PNT2022TMID42437
PROJECT NAME	HAZARDOUS AREA MONITORING FOR INDUSTRIAL PLANT POWERED BY IOT

1. INTRODUCTION

1.1 PROJECT OVERVIEW:

In our project, we used temperature sensor, humidity sensor and raspberry pi. IBM cloud services for mobile notification and also analysed the data by using python code to generate graph. The internet of things represents a general are concept for the ability of network devices to sensor and collection data from world around us, and then share that data across the internet where it can be processed and utilized for various practical purposes in different aspects of life. We create an IOT based hazards monitoring systems specifically suited to the requirements of mining, refining and manufacturing industries. The systems actively record, processes and analyses the temperature of the surroundings, which is a prime safety parameter in areas where molten metal is processed, manufacturing is done are welds are made.

1.2 PURPOSE:

The main of the project is to avoid accident and death in the gas leakage explosion in industries. Domestically we use natural gas and it is very useful for burning purpose. If this gas is leaked in Industry or factories and not sensed in time, it may lead to fatal disaster, and many cause human and material loss. To monitor the condition, we can integrate the smart devices in the area which are needed to be monitored every device will be acting as a beacon and it is connected to temperature sensors. We can broadcast the temperature data along with the location of that particular area through beacon. The persons who generally monitor these places will be given a wearable device acting as a beacon scanner. Whenever the person enters the desired area then he can view the required parameters and be alerted, these are sent to the cloud.

2 LITERATURE SURVEY

2.1 EXISTING PROBLEM:

In our day-to-day life there are many industries working with various hazardous chemical gases and the workers are often exposed to these gases. The unexpected accident causes a great impact to human lives and the properties. To avoid these situations, we need to develop automatic toxic gases detection and alerting system. The existing detection systems are available to sense only the particular gas and they use GSM technology to indicate the critical situations. The drawback is that the detection system can send a massage to only one person.

2.2 REFERENCES:

1) Arpitha.T, Divya kiran, V.S.N. Sitaram Gupta ,and Punithavathi Duraiswamy , "FPGA-GSM based leakage Detection system".

2) Dr.p.c.jain , Rajesh kushwaha , " Wireless gas sensor network for detection and monitoring of harmful gases in utility areas and industries".

3) Hemlata yadav ,Naomi oyiza, sarfaraz hassan, Dr.sumam lata , K.Jaya chitra ,``IOT based industrial monitoring system.''

4) Mr.Arijit Banik , Mr.Bodhayan Aich , Mr.suman Ghosh , "Microcontroller based Low cost gas leakage dectector with SMS alert ".

5) R.Bera and D.Bhattacharjee, "Development of Smart Wireless Sensing System for Environmental Monitoring," International Journal on Smart Sensing and Intelligent Systems vol.7, no.3, 2014.

6) Sandip S. Patil, Jaykaran Singh, "Monitoring and Controlling Of Hazardous Gases Inside Vehicle and Alerting Using GSM Technology for the Safety of People Inside the Vehicle," International Journal of Science, Engineering and Technology, vol 3, 2015

7) Subhash kumar , sameer choukey, "Gas leakage source localization and Boundary Estimation using mobile wireless sensor network ".

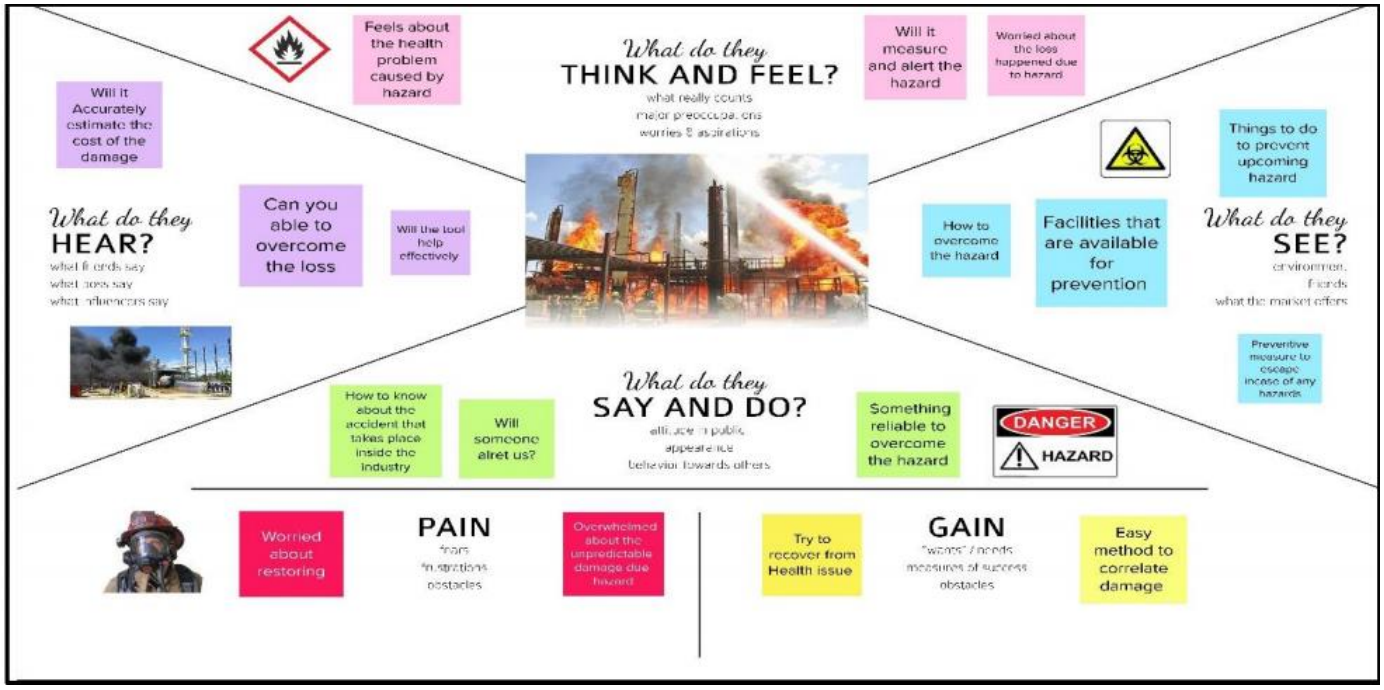
8) Vishesh S et al, 'Ammonia Gas Leakage Monitoring System using MQ-137 Sensors, IoT and Framing suitable Reflexive Actions', International Journal of Advanced Research in Computer and Communication Engineering, vol. 5, no. 11, 2016.

2.3 PROBLEM STATEMENT DEFINITION:

It is difficult to identify the hazardous gases like methane, propane, LPG, CO all the time by the people. In the same way it is difficult to monitor the temperatures in coal mining areas, pharmaceutical industries etc. It is difficult to monitor the people who are entering into prohibited areas. For this we have arranged a system for detection using IOT.

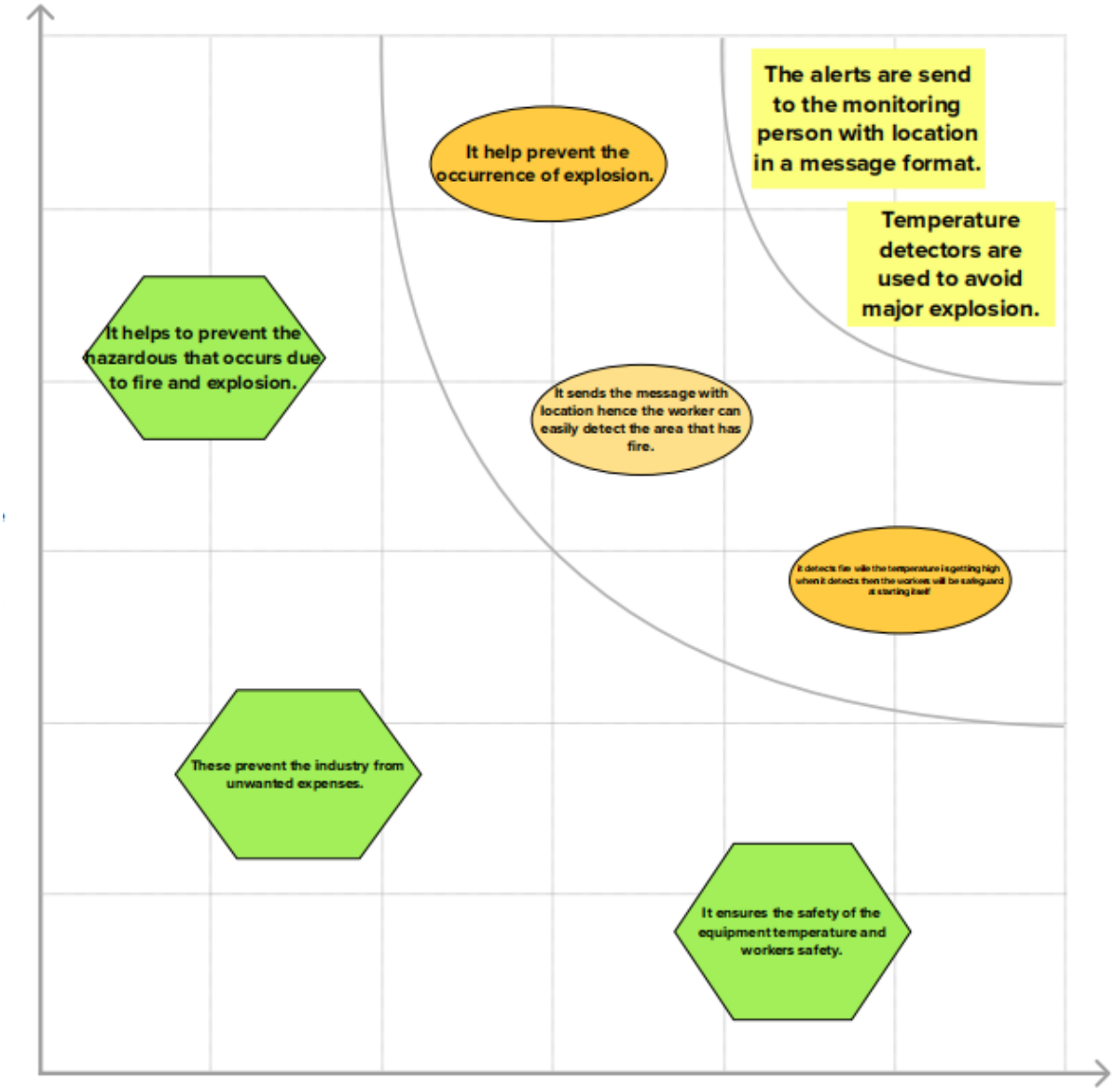
3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

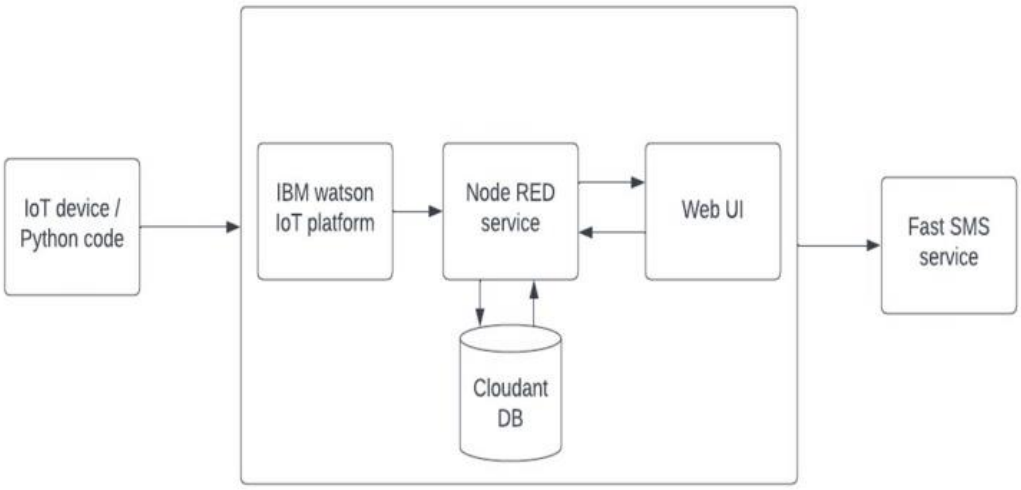


3.2 IDEATION & BRAINSTROMING:

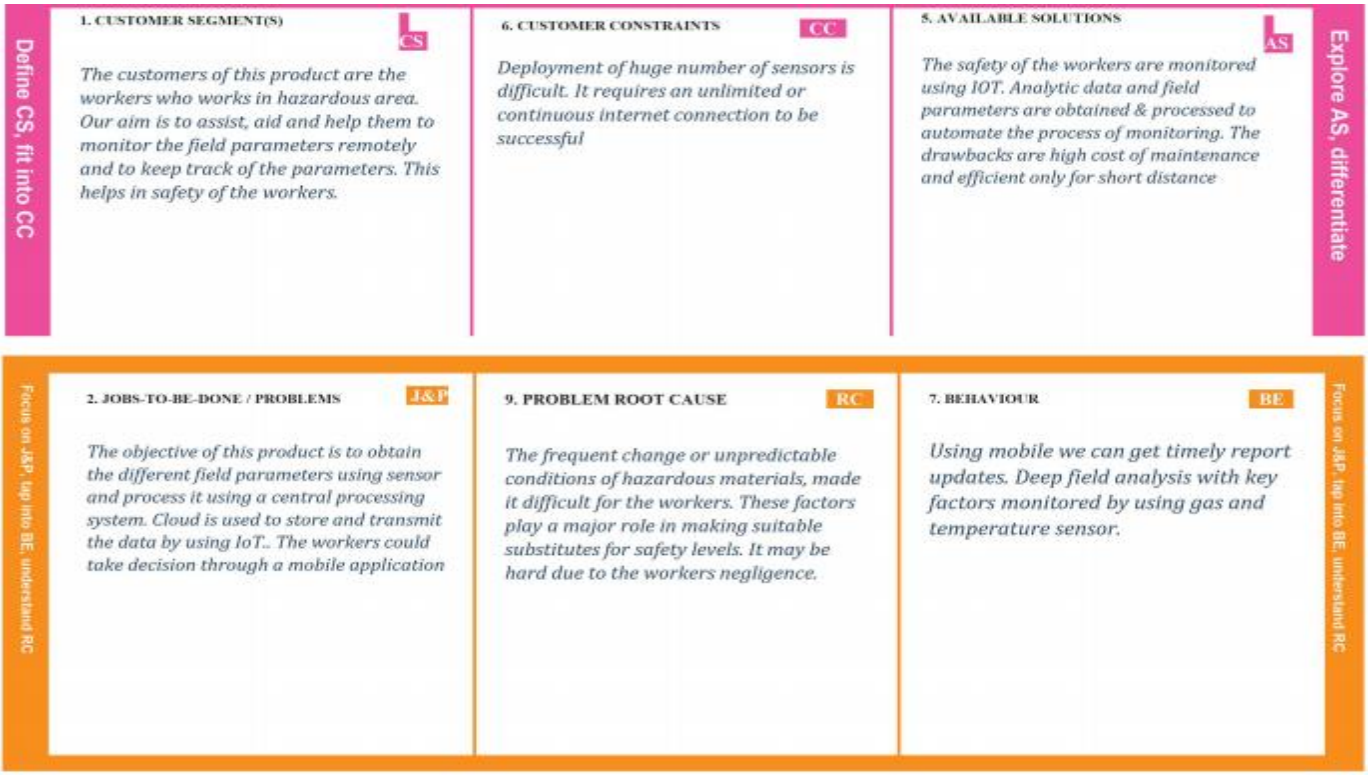
Internet of things aims to achieve higher operational and management efficiencies by machinery equipment and all other actors involved in industrial environment these are the benefits of using IOT in project. In industrial plants there are some areas which are to be monitored hence we can avoid the hazardous in industrial plant. To avoid hazardous occurrence in industrial plant hazardous areas monitoring for industrial plants help us. This monitoring device monitors the areas which should be monitored time to time. Monitoring the industrial plant is done by integrating the smart devices in areas to be monitored. Here in our project the sensors used to detect fire occurrence in industrial plant. Hence every device will be connected to temperature sensors and these devices act as a beacon.

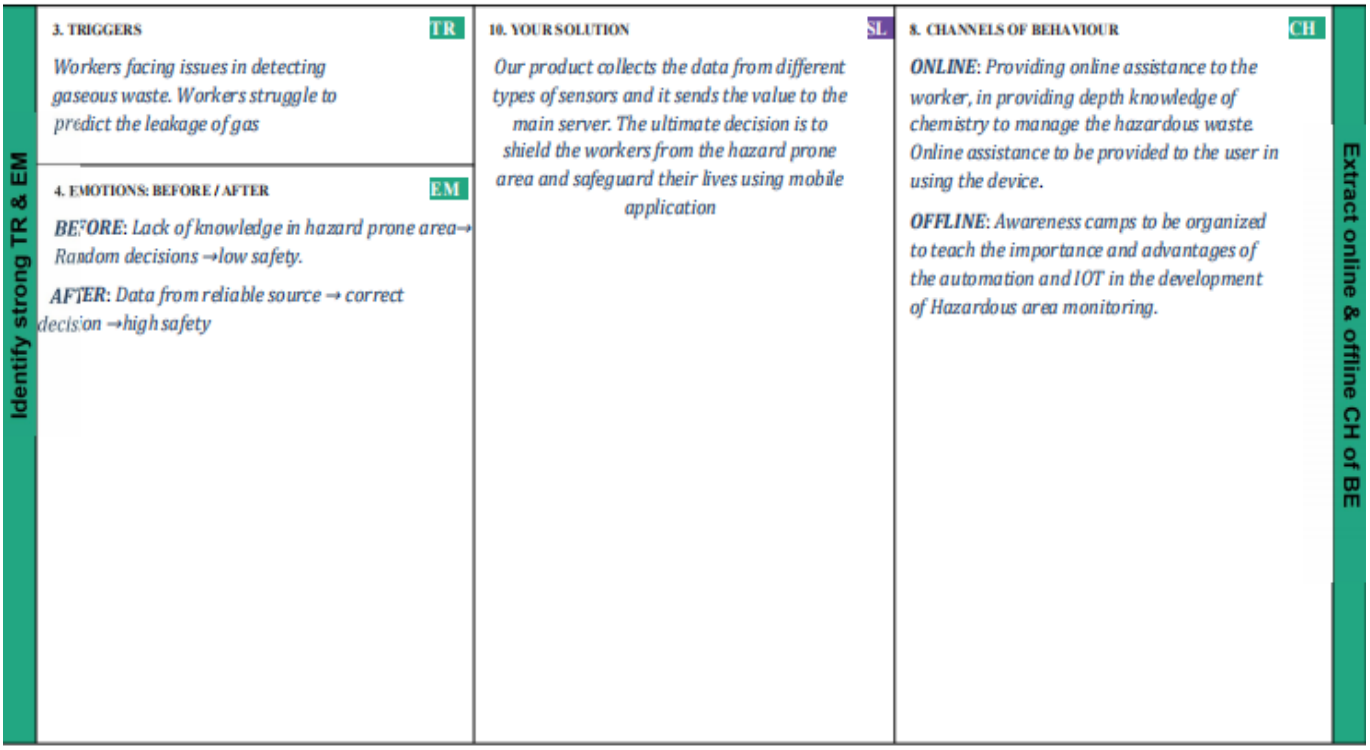


3.3 PROPOSED SOLUTION:



3.4 PROBLEM SOLUTION FIT:





4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENT:

FR No.	Functional Requirement(Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Data Gathering	The smart beacon must be able to detect and the temperature of a particular area in real.
FR-2	Location Detection	The smart beacon must be able to detect when a wearable device has entered an areanear it.
FR-3	Beacon Data Syncing	The smart beacon must be able to share its stored data with both the wearable device and admin dashboard through the cloud.
FR-4	Wearable Device Display	The wearable device must be able to displaythe temperature of the area where the worker is currently present.
FR-5	SMS Notification	If the temperature of the area is found to reach dangerous levels, the worker should be informed via SMS to their phone instructing them to leave the area.

4.2 NON-FUNCTIONAL REQUIREMENT:

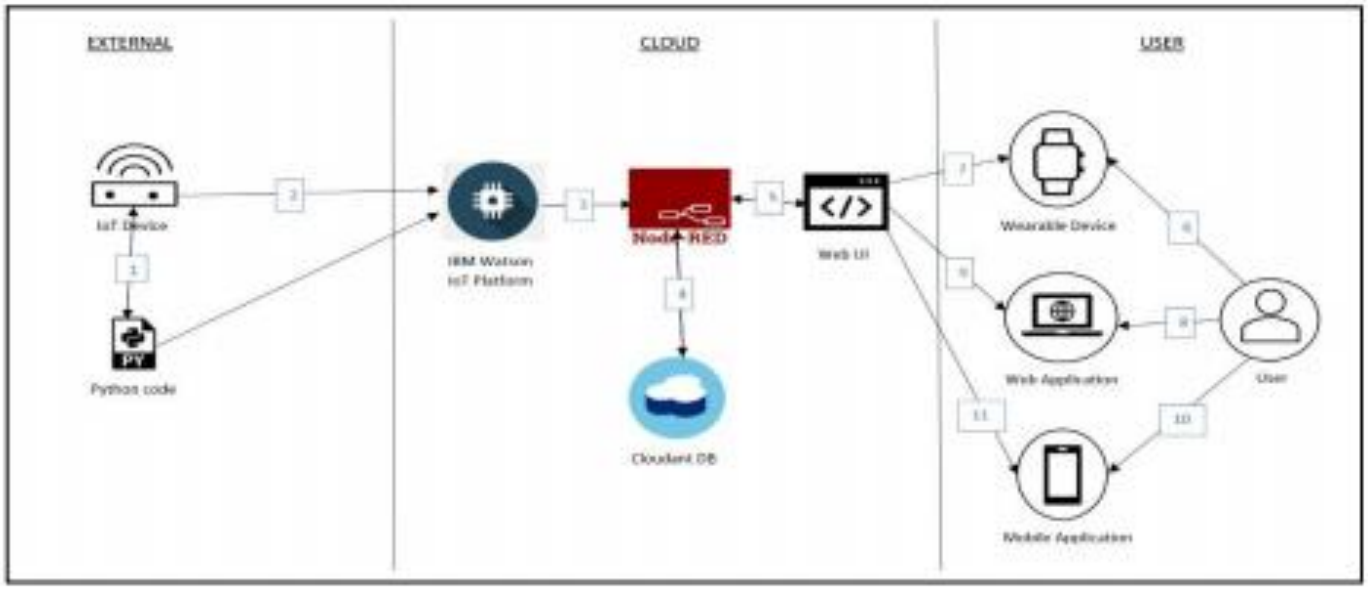
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The wearable device should be slim and not annoy or disturb the workers who are wearing them. They should also reliably display the temperature without large delays and notifications should be clear in cases of detected danger.

NFR-2	**Security**	The connection of the beacons to the cloud and wearable devices should be secure. The security of the database housing all the temperature data should also be bolstered.
NFR-3	**Reliability**	The wearable device should be able to function without any faults even at dangerous temperatures. If a fault is detected it should notify the user and the admin to be immediately repaired and replaced. The beacons should also be regularly maintained to ensure reliability.
NFR-4	**Performance**	The device should update temperature readings in real time and requires high end sensors and processors to do so. The time to send data to the cloud and other devices should also be made as smallas possible.
NFR-5	**Availability**	The user should be able to check the temperature of the area no matter where or at what time they are in the plant. The dashboard should be constantly active so as to ensure safety precautions can be executed whenever danger is detected.
NFR-5	**Availability**	The user should be able to check the temperature of the area no matter where or at what time they are in the plant. The dashboard should be constantly active so as to ensure safety precautions can be executed whenever danger is detected.

NFR-6	**Scalability**	If the area that needs to be monitored needs to be increased all one has to do is install new smart beacon devices and connect them to the same system as the previous beacons. It can also be replicated in different plants With different factors to be monitored giving it highly scalability.

5. PROJECT DESIGN:

5.1 DATA FLOW DIAGRAMS:

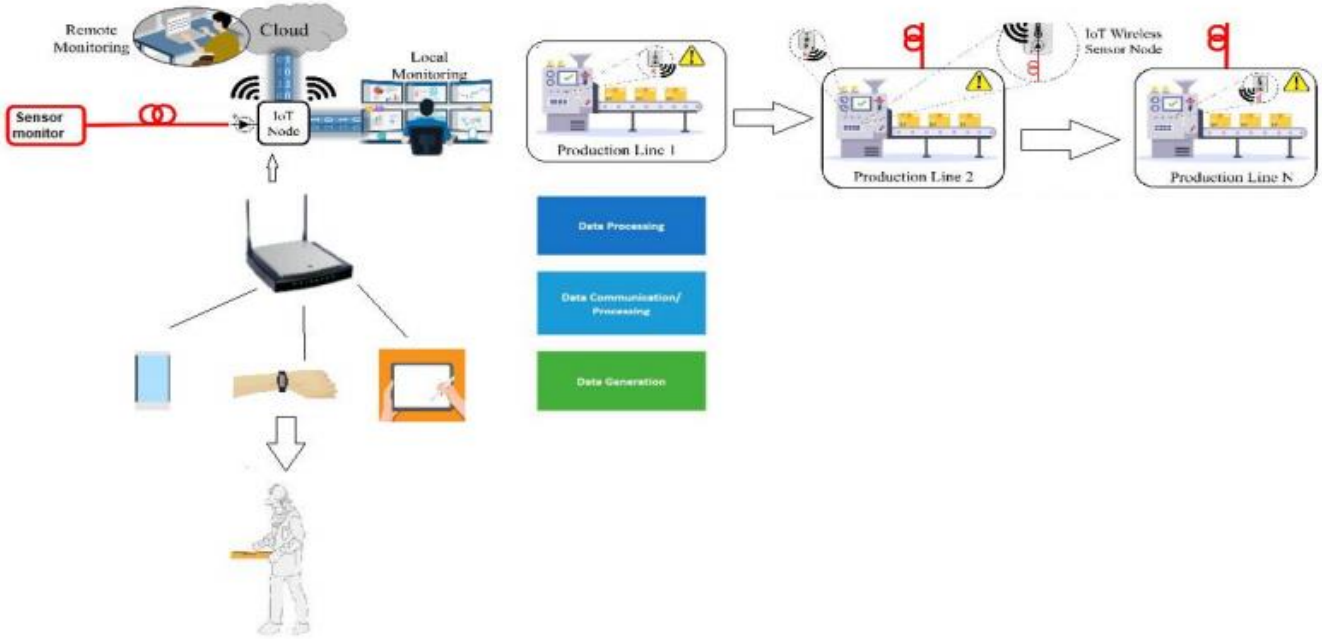


1. Necessary python code for collecting temperature. Details from IOT device are written.
2. IOT device is connected with the IBM Watson IOT platform for gathering data.
3. Next step uses Nodes -Red services after IOT platform is all set.
4. CLOUDANT DB is used for storing and retrieving data.
5. Node -Red services are used to create Wed application and UI designs.
6. {6, 7,8,9,10,11} The user uses Smart watch, Web and mobile app to receive various information and alerts.

5.2 SOLUTION & TECHNICAL ARCHITECTURE:

- To monitored the condition we can integrate the smart device in the area which are needed to be monitored every device will be acting as a beacon and it is connected to temperature sensors .
- In this project, we create an IOT -based hazards monitoring system specifically suited to the requirements of mining refining, and manufacturing industries

- The system actively records, processes and analyses the temperature of the surroundings which is a prime safety parameter in areas where molten metal is processed, manufacturing is done or welts are made, if a parameter is violated, the system sends an immediate notification to a set of a pre-set list of users on their smartphone and continues logging and monitoring data for further analysis to suggest improvements in the safety regulation of the industry.
- Broadcast the temperature data along with the location of that particular area through beacons. The persons who generally monitor these places will be given a wrist band and cell phones by alerting the call and SMS.
- Whenever the person enters the desired area then he can view the required parameters and can be alerted. These are sent to the cloud storage.



6. PROJECT PLANING & SCHEDULING:

6.1 SPRINT PLANING & ESTIMATION:

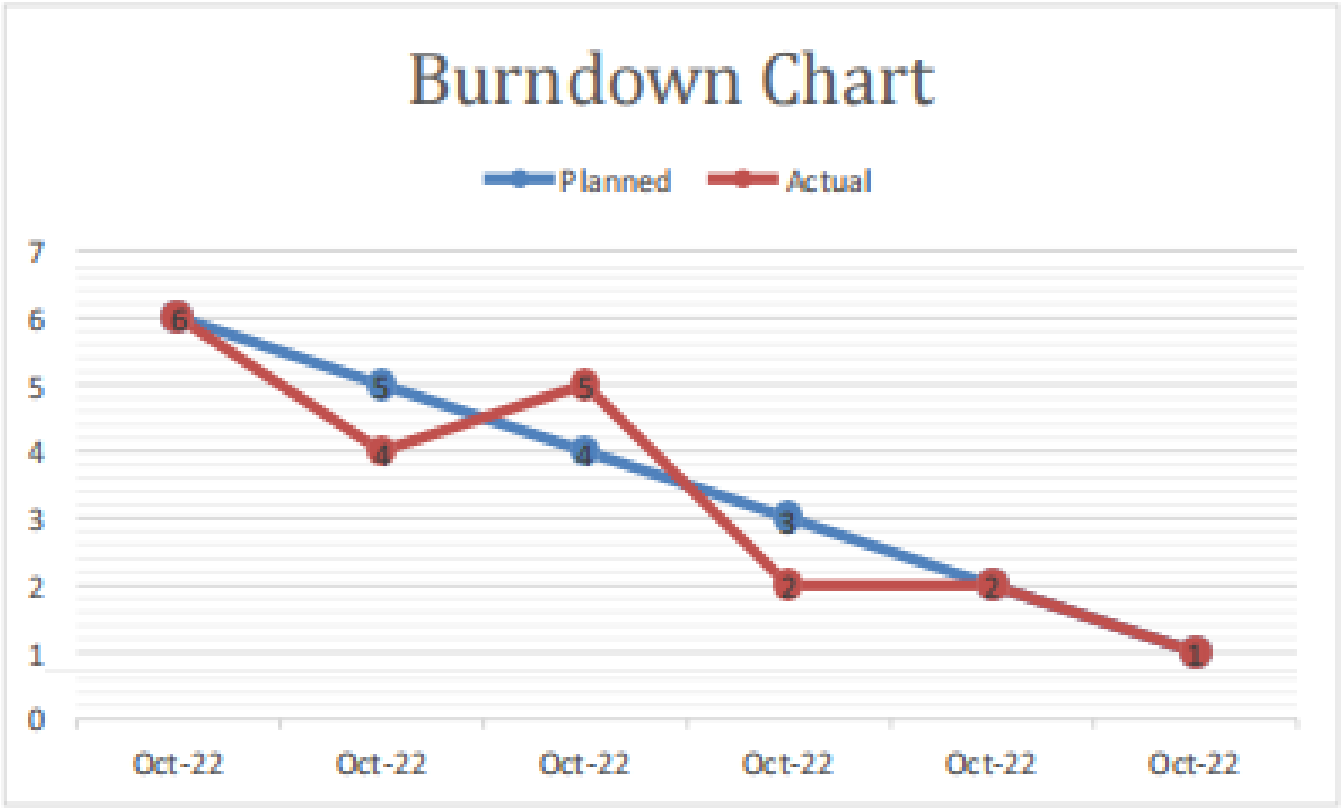
SPRINT	FUNCTIONAL REQUIREMENT(EPIC)	USER STORY NUMER	USER STORY/TASK	STORY POINTS	PRIORITY
Sprint-1	Installation of Beacons	USN-1	First the admin will be installing smart beacons at necessary places.	20	High
Sprint-1	Providing wearable's	USN-1	The admin will be providing everyone at the	20	Medium

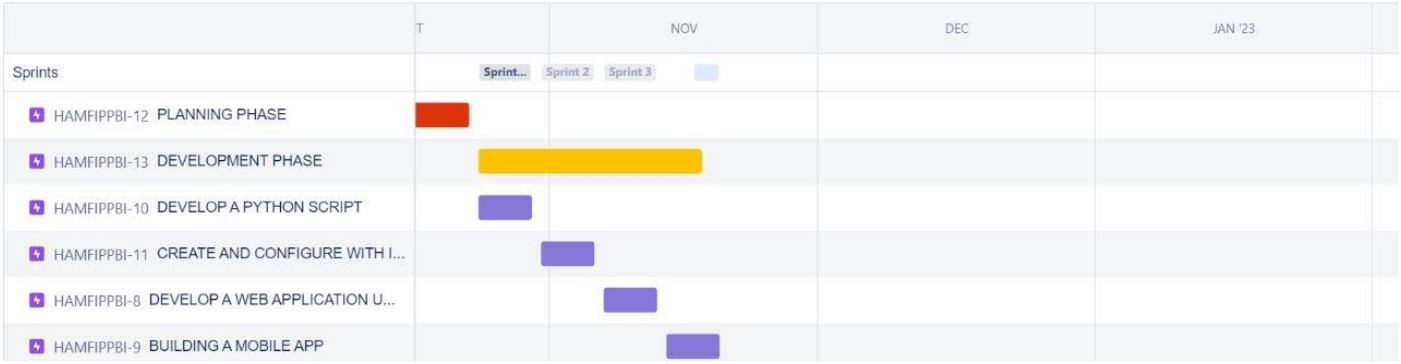
			industry a wearable device.		
Sprint-2	Cloud setup	USN-2	The smart Beacons will connect with the clouds services. Where we can get the real time data from the wearable.	20	High
Sprint-3	Online monitoring via web	USN-3	Websites will be created and connected with the cloud services.	20	High
Sprint-4	Monitoring via mobile	USN-4	Mobile application will be created and fast SMS will be used to alert abnormality to the user.	20	High

6.2 SPRINT DELIVERY SCHEDULE :

SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE (PLANNED)	STORY POINTS COMPLETED (AS ON PLANNED END DATE)	SPRINT RELEASE DATE (ACTUAL)
Sprint-1	20	6 days	24OCT2022	29OCT2022		29OCT2022
Sprint-2	20	6 days	31OCT2022	05NOV2022		05NOV2022
Sprint-3	20	6 days	07NOV2022	12NOV2022		12NOV2022
Sprint-4	20	6 days	14NOV2022	19NOV2022		19NOV2022

6.3 REPORTS FROM JIRA:





7. CODING & SOLUTIONING:

7.1 FEATURE 1:

```
 #define dht_dpin 12       //digital pin, that DHT's data line is connected
#define DHTTYPE DHT22     //When using DHT11, put here DHT11 instead of DHT22

//int temp;               //Use for DHT11 instead of float
float temp;               //Use float for showing decimals of the temperature reading. I recommend using for DHT22, no point to use for DHT11

//int hum;                //Use for DHT11 instead of float
```

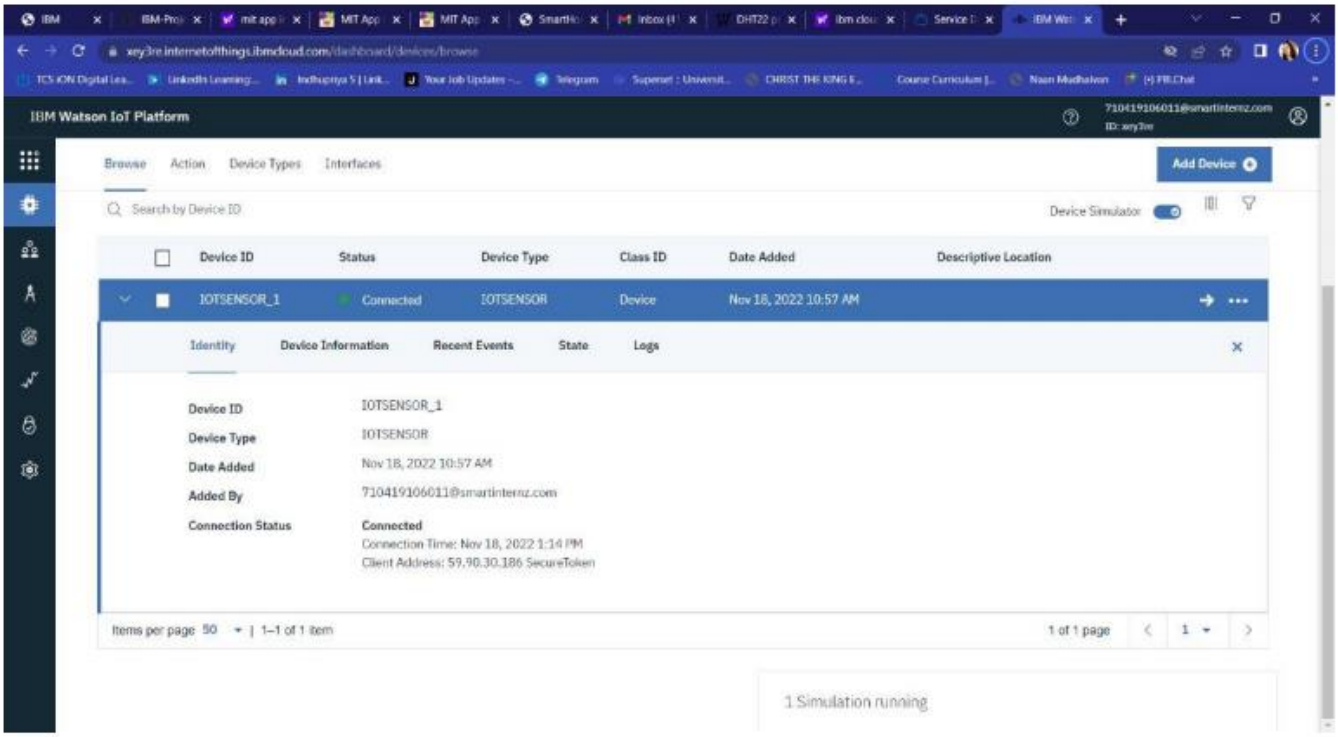
```
float hum;                  //Use float for showing decimals of the humidity reading. I recommend using for DHT22, no point to use for DHT11

void setup() {

Serial. begin(9600);      //Initiate serial monitor
 dht.begin();            //Initiate DHT sensor

}

void loop() {

//delay(1000);                  //wait a sec (recommended for DHT11)
delay(500);                   //wait a 0,5 sec (recommended for DHT22)

temp=dht.readTemperature(false);       //Read temperature of DHT and store it in to variable (temp). FALSE reads in celsius, leave empty for farenheit
hum=dht.readHumidity();             //Read humidity of DHTand store it iin variable (hum).

Serial.print("Temperature: ");       //Print text "Temperature: " in to serial monitor
Serial.println(temp);               //Print variable (temperature value) in to serial port. ln for line break

Serial.print("Humidity: ");          //Print text "Humidity: " in to serial monitor
Serial.println(hum);                //Print variable (temperature value) in to serial port. ln for line break
Serial.println(" ");                //print empty line in to serial monitor

  delay(2500);                      //optional delay, not really any point reading the sensor more than once every 3 seconds
}
```

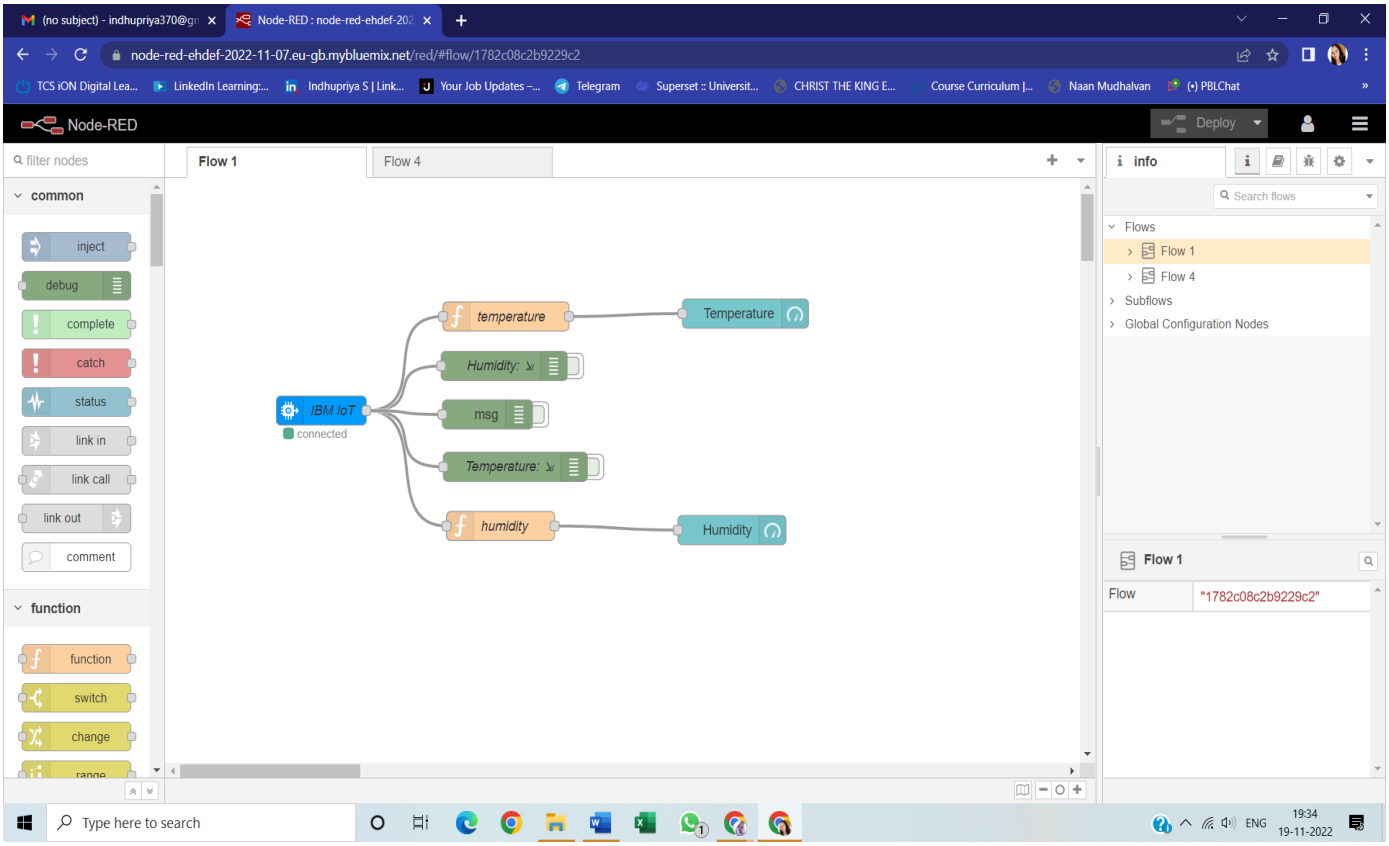
7.2 FEATURE 2:

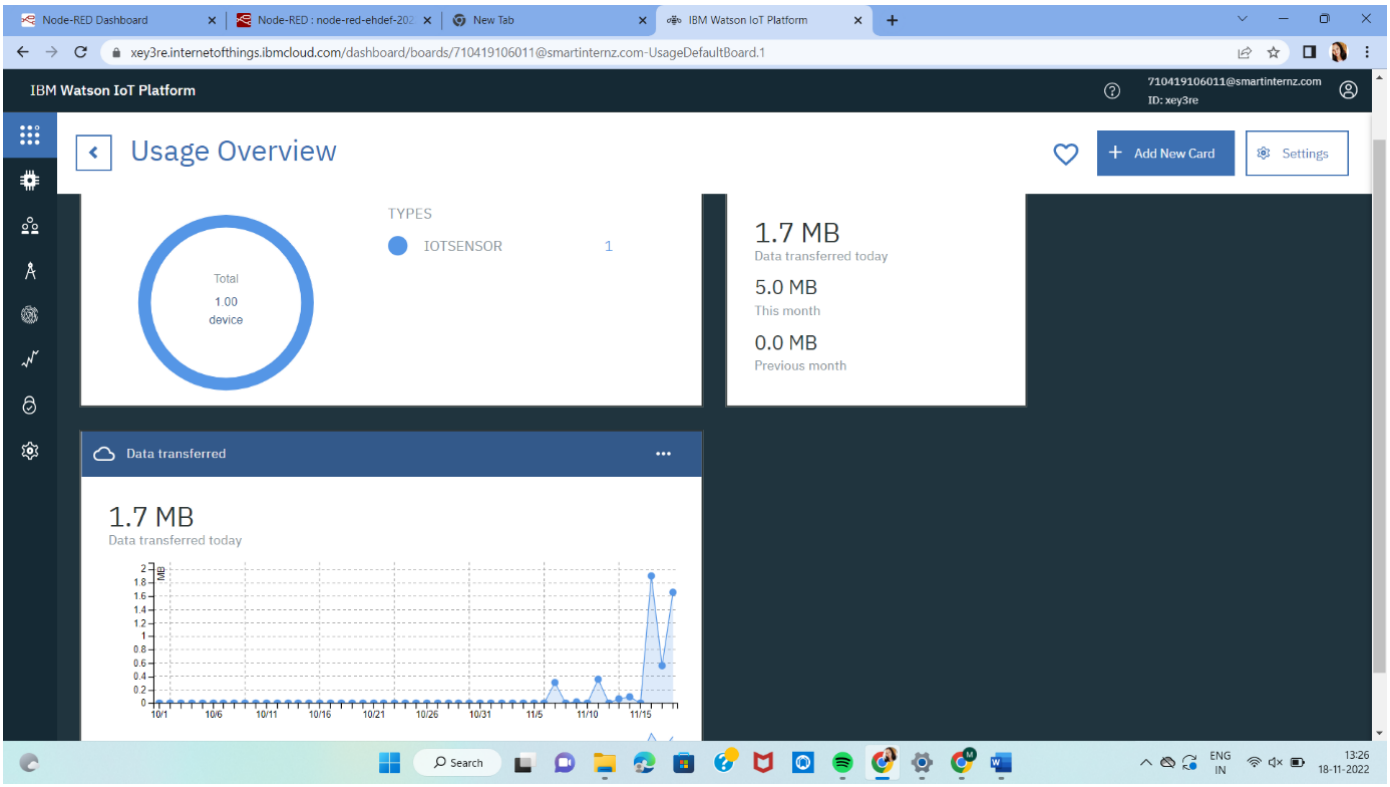
```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 12     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (12, DHT22);// 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 "xey3re"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_controller"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "BME280_sensor"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "BME280_sensor"     //Token
String data3;
float h, t;


//-------- Customise the above values --------
```

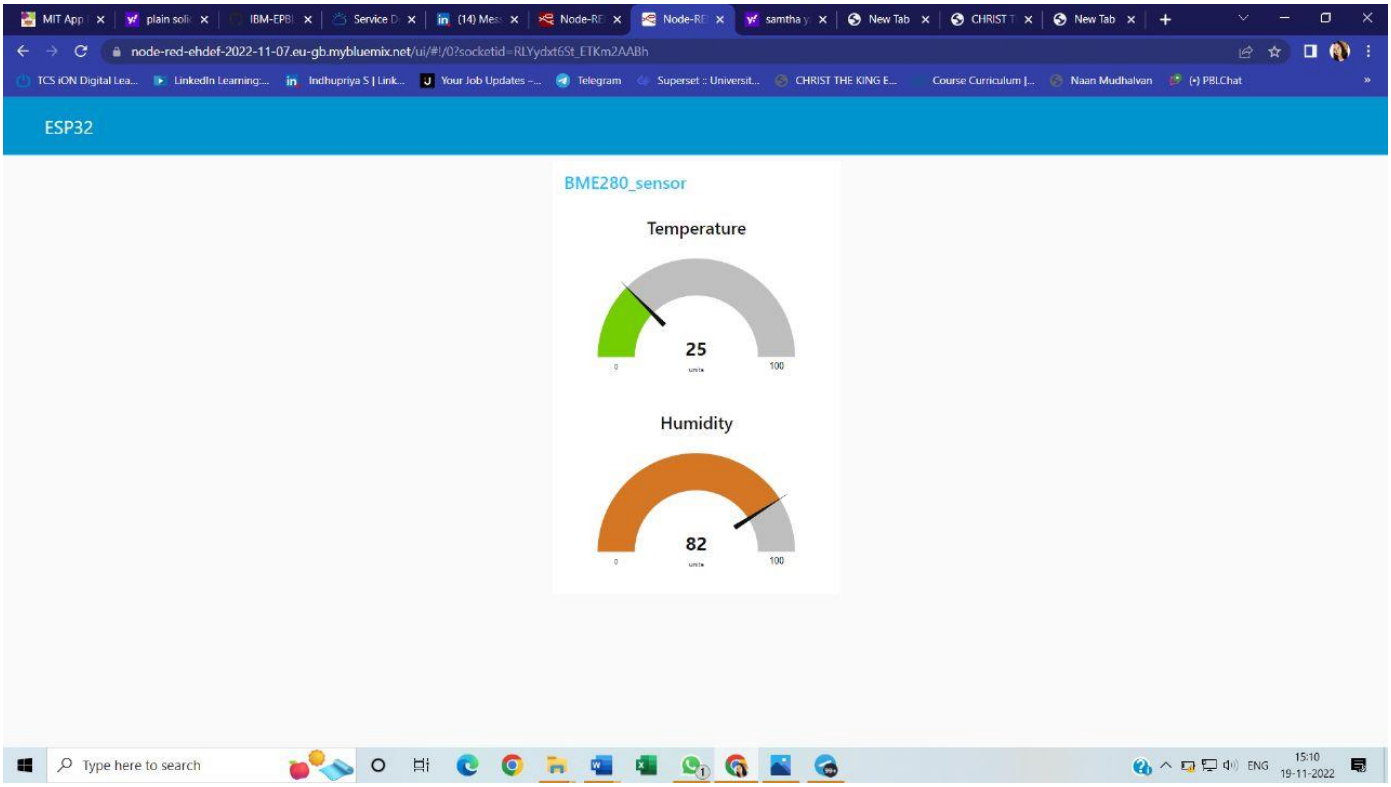
```
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd  REPRESENT command type AND COMMAND 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


// Callback function
void callback(char* topic, byte* payload, unsigned int length) {
  // In order to republish this payload, a copy must be made
  // as the orignal payload buffer will be overwritten whilst
  // constructing the PUBLISH packet.

  // Allocate the correct amount of memory for the payload copy
  byte* p = (byte*)malloc(length);
  // Copy the payload to the new buffer
  memcpy(p,payload,length);
  client.publish("project.indhu.dht", p, length);
  // Free the memory
  free(p);
}


#define dht_dpin 12       //digital pin, that DHT's data line is connected
#define DHTTYPE DHT22     //When using DHT11, put here DHT11 instead of DHT22

//int temp;               //Use for DHT11 instead of float
float temp;                 //Use float for showing decimals of the temperature reading. I recommend using for DHT22, no point to use for DHT11

//int hum;                //Use for DHT11 instead of float
float hum;                  //Use float for showing decimals of the humidity reading. I recommend using for DHT22, no point to use for DHT11

void setup() {

Serial. begin(9600);      //Initiate serial monitor
 dht.begin();             //Initiate DHT sensor

}

void loop() {

//delay(1000);                    //wait a sec (recommended for DHT11)
delay(500);                     //wait a 0,5 sec (recommended for DHT22)
```

```
temp=dht.readTemperature(false);       //Read temperature of DHT and store it in to variable (temp). FALSE reads in celsius, leave empty for farenheit
hum=dht.readHumidity();                //Read humidity of DHTand store it iin variable (hum).

Serial.print("Temperature: ");        //Print text "Temperature: " in to serial monitor
Serial.println(temp);                 //Print variable (temperature value) in to serial port. ln for line break

Serial.print("Humidity: ");           //Print text "Humidity: " in to serial monitor
Serial.println(hum);                  //Print variable (temperature value) in to serial port. ln for line break
Serial.println(" ");                  //print empty line in to serial monitor

  delay(2500);                        //optional delay, not really any point reading the sensor more than once every 3 seconds
}
```




Node-RED
Flow 1
Flow 4
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
switch
change
temperature
Temperature
Humidity:
msg
IBM IoT
connected
Temperature:
humidity
Humidity
info
Flows
Flow 1
Flow 4
Subflows
Global Configuration Nodes
Flow
"1782c08c2b9229c2"


IBM Watson IoT Platform
Usage Overview
Add New Card
Settings
TYPES
IOTSENSOR 1
Total
1.00
device
1.7 MB
Data transferred today
5.0 MB
This month
0.0 MB
Previous month
Data transferred
1.7 MB
Data transferred today



8. TESTING:

8.1 TEST CASES:

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
LoginPage_TC_OO1	Functional	Home Page	Used to take the user into login page		1.Enter username & password 2.click on login button		Application should show below UI elements: a.username text box b.password text box c.Login button	Working as expected	Pass				
LoginPage_TC_OO2	Functional	Home Page	Used to take the user into user page1		1.Click on Log in button		Application should show below UI elements: a.email text box b.password text box c.Login button	Working as expected	Pass	Steps are clear to follow		BUG-1234	

User Page_TC_OO3	Functional	user page1	Used to view the temperature &humidity		1.Enter username & password 2.click on login button	Username: indhu password: indhu	Application should show below UI elements: 1.Temperature 2.Humidity		Pass				
Home Page_TC_OO4	Functional	Home Page	Used to go back to homepage		1.click on go back button		Redirect to home page		Pass				

8.2 USER ACCEPTANCE TESTING:

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Product Name] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	20	4	2	3	29
Duplicate	3	0	3	0	7
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	36	14	13	26	86

3. 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		
Client Application	11	0		11
Security	0	0		

9. RESULTS:

9.1 PERFORMANCES METRICS:

S.NO	PARAMETER	PERFORMANCE
1	Response time	0.2 (Average of 10 trials)
2	Workload	500 users (Calculated based on cloud space)
3	Revenue	Individual users and industries.
4	Efficiency	Simple and straight forward work flow ,which makes the process efficient
5	Down time	Almost no down time due to IBM cloud enabled solution

10. ADVANTAGES & DISADVANTAGES:

ADVANTAGES:

1. This project helps industries in monitoring the rise and fall of temperatures and humidity.
2. The alert system will be triggered when the temperature exceeds normal bounds.
3. In case of emergencies, the admins will be notified in the same instant the workers are alerted.
4. In the web application, admins can view the sensor parameters.

DISADVANTAGES:

1. Some of the drawbacks included air and water pollution and soil contamination that resulted in a significant that resulted in a significant deterioration of quality life expectancy.

2. Industrialization also exacerbated the separation of lab or and capital.

3. These who owned the means of production became dis proportionately rich, resulting in wider income inequality.

4. Industrialization impacted society in other ways.

11. CONCLUSION:

IOT is present and gaining more traction in a lot of fields, and one of the most important fields is industrial application. There are a huge numbers of ways in which industries can make use of IOT to improve working conditions, efficiency, cutting costs and improving the overall growth of the sector .How ever hazardous monitoring and mitigation is often overlooked in industrial area.

11. FUTURE SCOPE:

The use of digital solutions can offer a viable and safe way to address the challenges involved with hazardous area monitoring, minimizing the need for manual inspection and intervention. Periodic data collection gives a more thorough overview of asset condition, removing the need for maintance teams to carry out observations and collect data manually. So the digital solutions can be a better option for monitoring in future.

13. APPENDIX

SOURCE CODE:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt
#include "DHT.h"// Library for dht11
#define DHTPIN 12     // what pin we're connected to
#define DHTTYPE DHT22   // define type of sensor DHT 11
#define LED 2

DHT dht (12, DHT22);// 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 "xey3re"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_controller"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "BME280_sensor"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "BME280_sensor"      //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


// Callback function
void callback(char* topic, byte* payload, unsigned int length) {
  // In order to republish this payload, a copy must be made
  // as the orignal payload buffer will be overwritten whilst
  // constructing the PUBLISH packet.

  // Allocate the correct amount of memory for the payload copy
  byte* p = (byte*)malloc(length);
  // Copy the payload to the new buffer
  memcpy(p,payload,length);
  client.publish("project.indhu.dht", p, length);
  // Free the memory
  free(p);
}


#define dht_dpin 12       //digital pin, that DHT's data line is connected
#define DHTTYPE DHT22     //When using DHT11, put here DHT11 instead of DHT22

//int temp;               //Use for DHT11 instead of float
float temp;                 //Use float for showing decimals of the temperature reading. I recommend using for DHT22, no point to use for DHT11

//int hum;                //Use for DHT11 instead of float
float hum;                  //Use float for showing decimals of the humidity reading. I recommend using for DHT22, no point to use for DHT11

void setup() {

Serial. begin(9600);      //Initiate serial monitor
 dht.begin();             //Initiate DHT sensor

}

void loop() {

//delay(1000);                    //wait a sec (recommended for DHT11)
delay(500);                     //wait a 0,5 sec (recommended for DHT22)
```

```
temp=dht.readTemperature(false);       //Read temperature of DHT and store it in to variable (temp). FALSE reads in celsius, leave empty for farenheit
hum=dht.readHumidity();                //Read humidity of DHTand store it iin variable (hum).

Serial.print("Temperature: ");        //Print text "Temperature: " in to serial monitor
Serial.println(temp);                 //Print variable (temperature value) in to serial port. ln for line break

Serial.print("Humidity: ");           //Print text "Humidity: " in to serial monitor
Serial.println(hum);                  //Print variable (temperature value) in to serial port. ln for line break
Serial.println(" ");                  //print empty line in to serial monitor

  delay(2500);                        //optional delay, not really any point reading the sensor more than once every 3 seconds
}
```

GITHUB &PROJECT DEMO LINK:

GITHUB LINK:

https://github.com/IBM-EPBL/IBM-Project-8377-1658916715

DEMO LINK:

https://drive.google.com/drive/folders/1XCZ8YkoJd_lGlRX9OJgmeDcQSaLZjFO8