GAS LEAKAGE DETECTION AND MONITORING SYSTEM:

TEAM ID: PNT2022TMID29935

TEAM LEAD:

VEERASATHISH.U

TEAM MEMBERS:

NAVANEETHAKRISHNAN.D

SUDESH.T

SATHISH KUMAR.M

1. INTRODUCTION:

1.1 PROJECT OVERVIEW:

This gas detection project main idea is to implement security system for detecting leakage of gas in closed environment. In this project gas leakage is identified by using sensors which works only in closed environment. In present situation there are many cases related to gas leakage which cause innocent people lives and property damage. Implementing this application can be useful for companies, houses, which can save lives of people. This project helps the industries in monitoring the emission of harmful gases. In several areas, the gas sensors will be integrated to monitor the gas leakage. If in any area gas leakage is detected the admins will be notified along with the location. The gas sensor monitored gas level and details send to IBM cloud. The location and gas level send to the web application. In the web application, admins can view the sensor parameters

.

1.2 PURPOSE:

The purpose of this project is to know the details about the gas leakage area in the closed environment in the industries. As the result of this fetched details, we can take the precautionary measures to avoid the over flow of the harmful gases. By these facilities we ensure the safety of the employee's hundred percent. This project idea can be developed in various industries according to the need of t5yhe particular industry.

2. LITERATURE SURVEY:

2.1 EXISTING PROBLEM:

There are many existing problems in this idea implementation. Some of the problems occurs due to poor monitoring of the devices. In the GPS module it may shows the error in finding the exact location of the gas leakage area. The industry area which is poor in internet connection may face a network issue problem in a message sending process, so analysis of the each and every information about that particular industry is compulsory. So, this also one of the existing problems.

2.2 REFERENCES:

[1] Sanjoy Das, Sahana S, Soujanya K Swathi M C, "Gas leakage detection and prevention using IoT": International Journal of Scientific Research % Engineering Trends. Vol 6, Issue 3, May-June 2020, ISSN (online): 2395-566X.

This paper fundamentally manages the advancement of a straightforward gas spill locator at the underlying stage and after that changing this basic gadget into a most progressive gas identifier framework later on. Gas sensors have been specifically utilized which has high affectability for propane (C3H8) and butane (C4H10). Gas leakage system consists of GSM (Worldwide System for versatile communication) module, which sends SMS as soon as gas leakage is detected. Keywords: Arduino, MQ-6 Gas Sensor, LCD, LPG, Stepper.

[2] Dr. Chetana Tukkoji, Mr. Sanjeev Kumar, "Review paper on-LPG Gas leakage detection using IOT": IJEAST –International Journal of Engineering Applied Science & Technology, Vol 4, Issue12, April 2020 IJEAST (online): 603-609.

To alert on Liquefied rock oil Gas (LPG) leakage and preventing any unwanted incident, we need to apply some cautions to discover the discharge. It can bed eveloped associate degree Arduino based LPG gas detector alarm, if gas leakage happens. The LPG detector MQ 6 is associate degree correct LPG sensing device that acquires the signal intensity. The intensity of the LPG leakage is classed into 3 categories, such as LOW, MEDIUM and HIGH based on square measure. This paper conjointly shows the ratio and temperature over the alphanumeric display.

[3] Amatul Munnaza, Rupa Tejaswi, Tarun Kumar Reddy, Saranga Moahan "IoT Based Gas Leakage Monitoring System": Journal of Xi'an University of Architecture & Technology (JXUAT), Vol 12 ISSN No: 1006-7930, Issue 5, 2020.

The foremost object of this work is to monitor gas leakage in any industries using gas sensor and Spartan 6 FPGA process. Structure a cloud-based monitoring system is very important to reduce the cost of preserve servers, to avoid data misplaces and to make the access easy with multiple internet linked devices (computer, tablet, mobile phone) at the similar time anywhere in the world. With Internet of Things (IOT), we can control any electronic equipment.

[4] Gas Leakage Detection and Prevention System, Shreyas Thorat, Neha Tonape, International Journal of Trendy Research, Vol 4, Issue 7, Dec 2020, ISSN NO: 2582-0958.

The objective of this project is to present the design of an automatic alarming system, which can detect and prevent liquefied petroleum gas leakage in various premises. This system alerts the user by sending him a phone call and alerting the neighbours by buzzer alarm after the gas leaks above setpoint1.The servo motor is used to close the gas pipe valves. This device ensures safety and prevents suffocation and explosion due to gas leakage. This project is implemented using Arduino uno and simulated using Arduino ide and proteus software.

[5] Rohan KH1, Navanika Reddy, Pranamya Maddy, Sachit Girish, Dr. Badari Nath K- "IOT based gas leakage detection and Alerting system": JRP Publications, Vol. 1(1), pp no. 002-006, February 2021.

Gas leakages are causing massive explosions in places throughout the World. The conventionally available gas leakage detectors only have the provision to alarm the user who is physically present at the spot. Hence, to overcome this limitation, this project implements a model which sends an email to the user in case there is a leakage. This model detects the leakage of Liquid Petroleum Gas & Benzene. The prototype of this model generates an email to the concerned person using IFTTT web service. An LED is also used as a visual alarm at the site of leakage.

Proposed method:

Gas detection system can be used to detect combustible, flammable and toxic gases, and oxygen depletion. This type of device is used widely in Industry. In the above methodologies they used only GSM and send the message that contains about the gas leaked only.

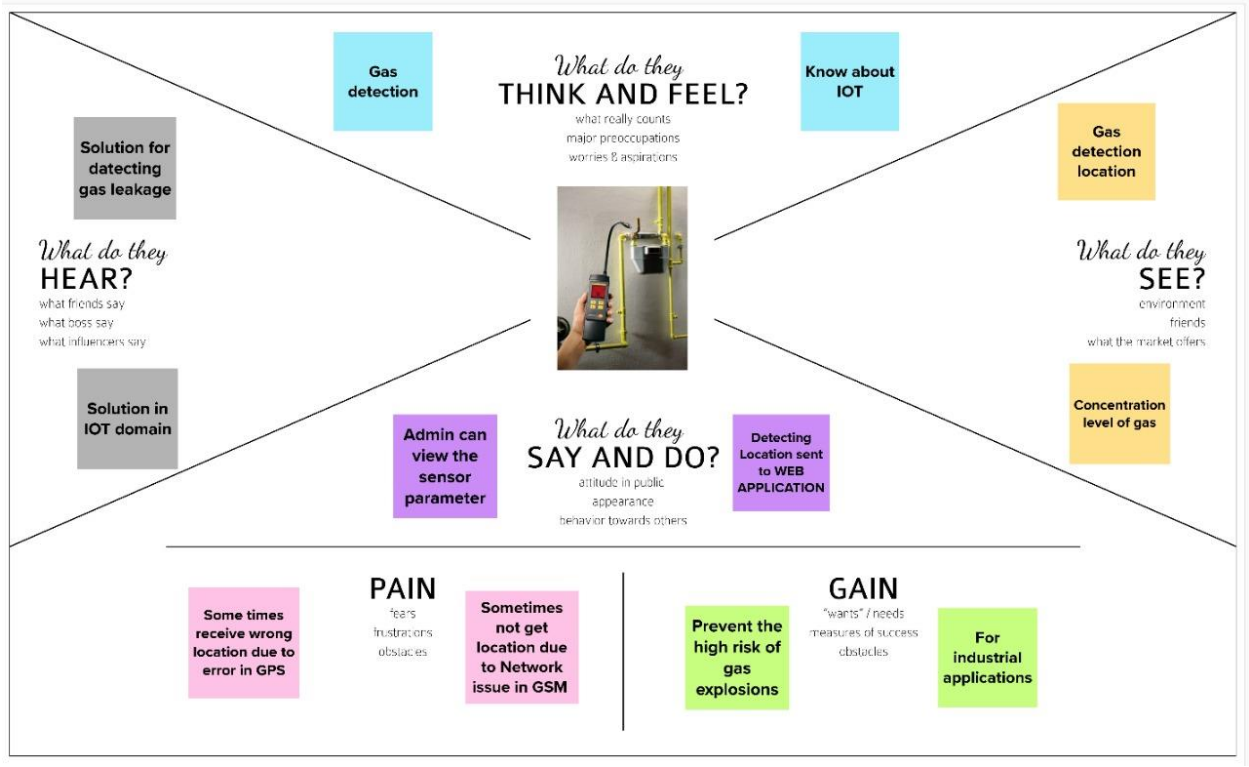
In our method, we are using GPS for identifying the exact location and send the location through message to the admin with the details of concentration level of gas.

2.3 PROBLEM STATEMENT DEFENITION

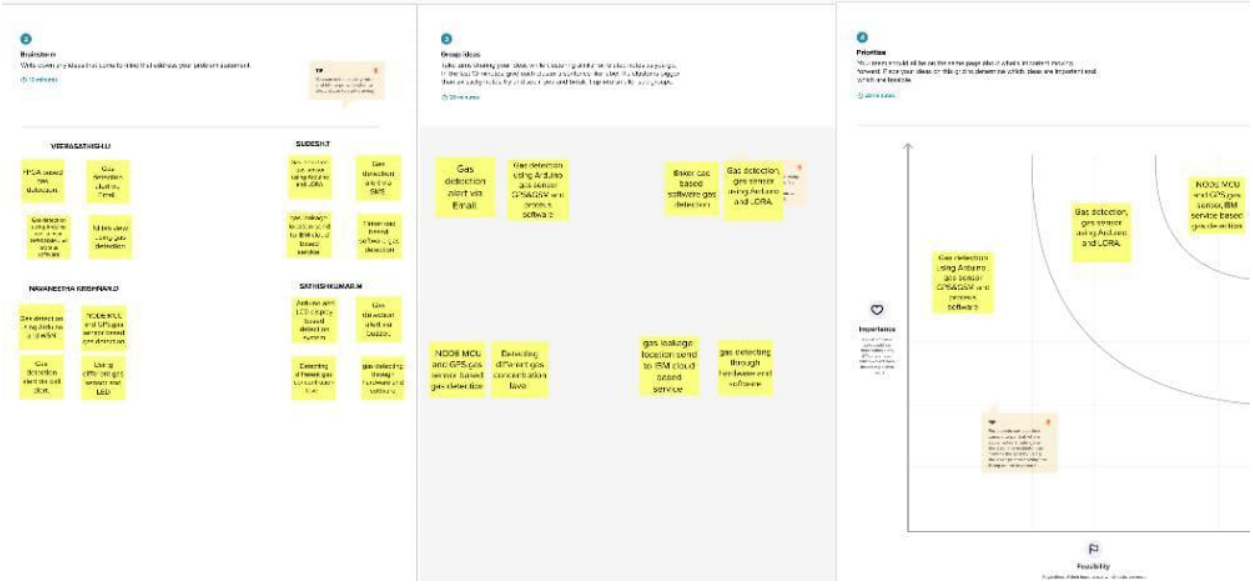
Problem statement	**Iam (Customer)**	**I am trying to**	**But**	**Because**	**Which makes me feel**
GAS Detection and Monitoring	Industry	Work in chemical unit and oil & Gas unit	Harmful gas leakage occurs	Improper main tanance	Using gas sensor can detect
Gas leakage location view through Web application	Industry admin	View the gas leakage location	location will not send.	Only send message alert	Using GPS Can send location

3. IDEATION & PROPOSED SOLUTION:

3.1 EMPATHY MAP CANVAS.



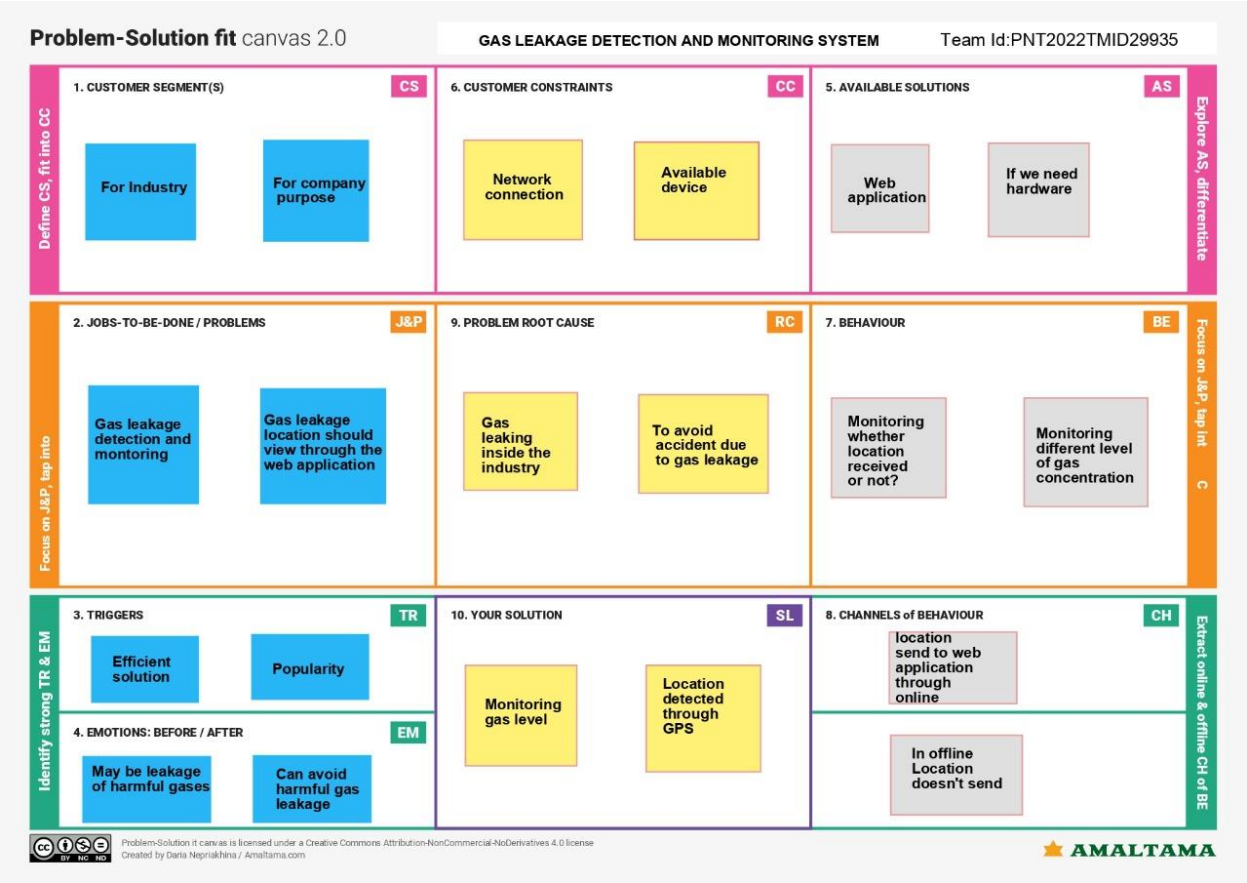
3.2 IDEATION &BRAINSTROMING



3.3 PROPOSED SOLUTION:

S.No	Parameter	Description
1.		
2.	Idea/solution description	parameters.
3.	Novelty/Uniqueness	
4.		
5.		Sell sensor product and software as service model with industry &large communities.
6.	Scalability the solution	High scalability

3.4 PROBLEM SOLUTION FIT:



4. REQUIREMENT ANALYSIS:

4.1 FUNCTIONAL REQUIREMENT:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Gas leakage Location send	Location should send to web application GPS module needed for location detection
FR-4	Gas concentration level	Gas leakage level can view through web application Gas sensor needed for gas leakage level

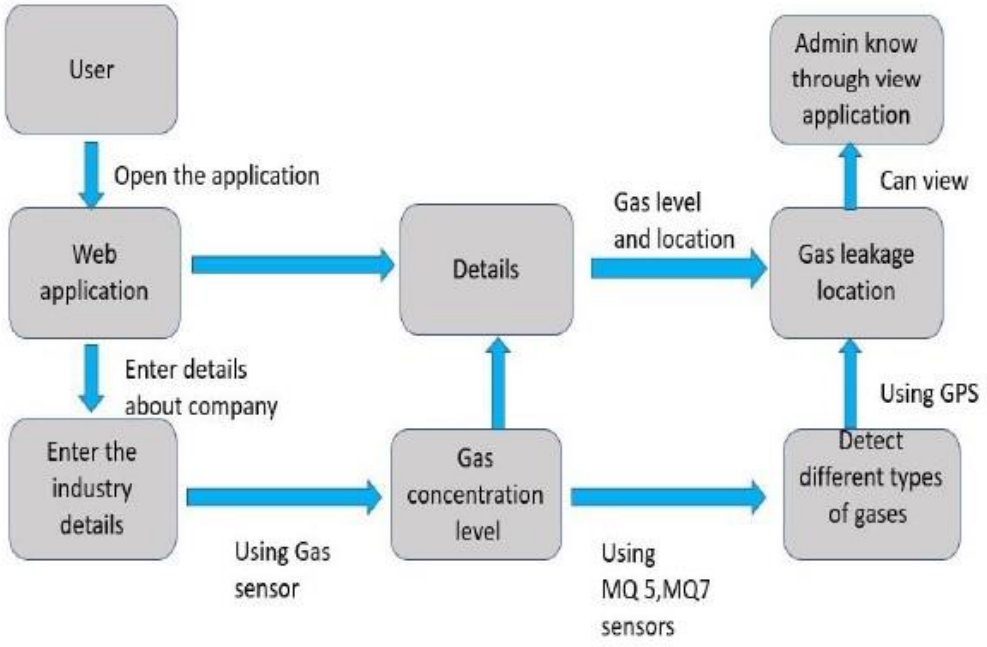
4.2 NON-FUNCTIONAL REQUIREMENTS:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Easy to use and simple web application. Efficiency is good.
NFR-2	**Security**	Web application is highly secure. Software is protected by un authorized access.
NFR-3	**Reliability**	High reliability to work with web application. Application runs accurately.
NFR-4	**Performance**	Gas concentration level updated in web application Immediately.
NFR-5	**Availability**	24*7 can use this application. Available widely.
NFR-6	**Scalability**	Application is used unlimited users. High scalability.

5 PROJECT DESIGN.

5.1 DATA FLOW DIAGRAM.

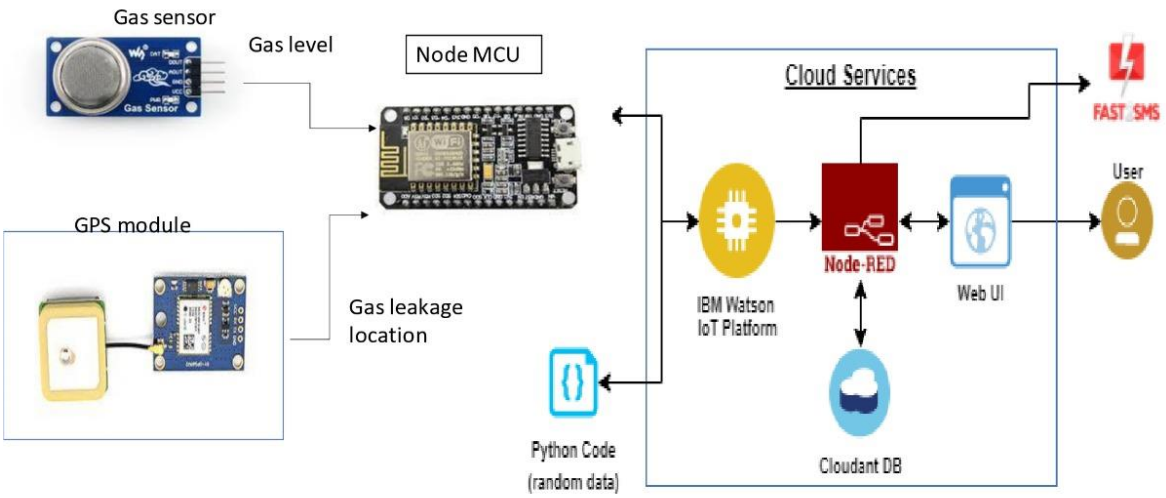


5.2 SOLUTION & TECHNICAL ARCHITECTURE:

Solution architecture

Team ID:PNT2022TMID29935

Project Name: Gas leakage detection and monitoring system



5.3 USER STORIES:

User Type	Functional requirement	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
Customer (Web user)	Registration	USN-1	As a user, I can register for the application by google.	I can access confirmation email.	High	Sprint-1
		USN-2	As a user, I can register for the application by firebox.	I can access confirmation Login.	low	Sprint-2
	Login	USN-3	As a user, I can register for the application through Gmail		Medium	Sprint-1
Administrator	Registration	USN-1	As a user, I can register for the application through Mobile app.	I can access confirmation My account	High	Sprint-1
		USN-2	As a user, I can register for the application through Mobile app.	I can access confirmation email	low	Sprint-2

6. PROJECT PLANNING &SCHEDULING:

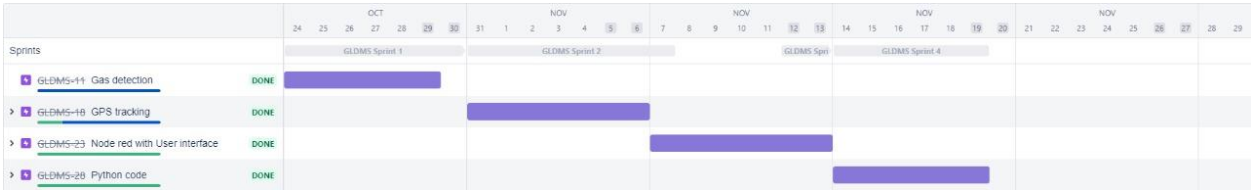
6.1 SPRINT PLANNING &ESTIMATION:

SPRINT	TOTAL STORY POINTS	DURATION	SPRINT START DATE	SPRINT END DATE	STATUS	SPRINT RLEASE DATE (ACTUAL)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	COMPLETED	31 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	COMPLETED	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	COMPLETED	17 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	COMPLETED	19 Nov 2022

6.2 SPRINT DELIVERY SCHEDULE:

Sprint	Functional requirements	User story number	User story/ Task	Total story points	Priority	Team Members
Sprint-1	Gas detection and level monitoring	USN-1	As a user, I can get the gas leakage alert when gas leaking.	20	High	Veerasathish.U Sudesh.T
Sprint-1		USN-2	As a user, I can get the different gas level when	20	Medium	Navaneethakrishnan.D
Sprint-2	GPS tracking	USN-3	As a user, I can get the gas leakage location.	20	Medium	Navaneethakrishnan.D Sathishkumar.M
Sprint-2		USN-4	As a user, I can get the gas leakage location when gas leaking.		High	Veerasath.U Sudesh.T
Sprint-3	Node red creation	USN-5	As a user, I can receive gas leakage level with		High	Navaneethakrishnan.D
Sprint-3		USN-6	As a user, I can receive gas leakage level with location through mobile app.	20	Medium	Veerasathish.U Sudesh.T
Sprint-4	Documentaion	USN-7	As a user, I can get the gas level and leakage	20	Medium	Navaneethakrishnan.D
Sprint-4		USN-8	As a user, I can get the gas leakage location and documentation	20	High	Veerasathish.U Sudesh.T

6.3 REPORTS FROM JIRA:



LINK FOR JIRA PLATFORM:

https://pnt2022tmid29935.atlassian.net/jira/software/projects/GLDMS/boards/1/roadmap?timeline=WEEKS&shared=&atlOrigin=eyJpIjoiZDZkN2VkN2I2ODU1NGVlMDhhNjFkYjgwNTI0ZTk5NWMiLCJwIjoiaiJ9

https://pnt2022tmid29935.atlassian.net/jira/software/projects/GLDMS/boards/1/roadmap?timeline=WEEKS

7.CODING & SOLUTIONING:

7.1 FEATURE 1:

PYTHON CODE FOR GAS LEAKAGE:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random


#Provide your IBM Watson Device Credentials
organization = "x6troc"
deviceType = "PNT2022TMID29935"
deviceId = "6374679606"
authMethod = "token"
authToken = "eSJ(wSv_kaOwuZ?yIl"

# Initialize GPIO

def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status=="lighton":
        print ("led is on")
    else :
        print ("led is off")
    #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()
```

EXPLANATION:

By using this code, we can connect the ibm cloud and the gas sensor. This code helps us to share the gas leakage information from ESP32 to ibm cloud. This ESP32 acts as a NODE mcu which is the WIFI module. In the ibm cloud we can see the gas level.

7.2 FEATURE 2:

CODE FOR GAS SENSOR:

```
 while True:
       #Get Sensor Data from DHT11
      gas=random.randint(0,200)
      data = { 'gas' : gas  }
def myOnPublishCallback():
            if gas>100:
               data = { 'gas' : gas }
```

```
                print ("Published gas_level = %s ppm" % gas, "//Gas alert!!")
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
# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

EXPLANATION:

This above code is for the gas sensor and this code is for the activation of the gas sensor. So that the gas sensor can sense the leakage gas in the close environment industry and send the location of the gas leakage area and the gas level to the ibm cloud.

8. TESTING:

8.1 TEST CASES:

TESTCASE 1:

TO DO: Gas leakage detection alert.

OUTPUT:

When a gas leakage is detected in a particular place. As a industry admin he can see the alert in the web app.

TESTCASE 2:

TO DO: Different level of gases.

OUTPUT:

When a gas leakage is detected in a particular place. As a industry admin he can see the different level of gases in the web app.

TESTCASE 3:

TO DO: Detect gas leakage location.

OUTPUT:

When a gas leakage is detected in a particular place. As a industry admin he can see the gas leakage location in the web app.

8.2 USER ACCEPTANCE TESTIING:

UAT Execution & Report Submission.

Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the[Pro ductName] projectat the time of the release to User Acceptance Testing(UAT).

Defect Analysis

This reportshows the numberof resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	12	2	3	2	20
Duplicate	1	0	4	0	5
External	3	2	0	7	12
Fixed	10	2	4	19	35
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	0	1
Won't Fix	0	5	3	1	9
Totals	26	11	16	29	83

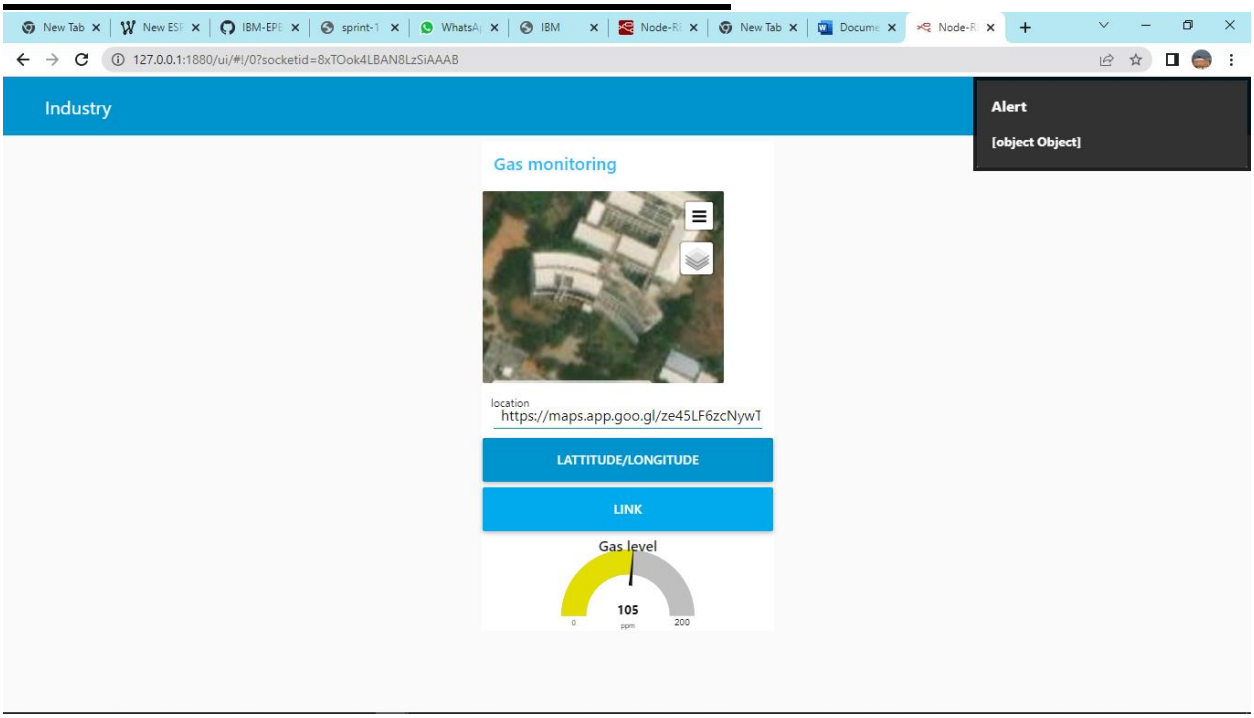
Test Case Analysis

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

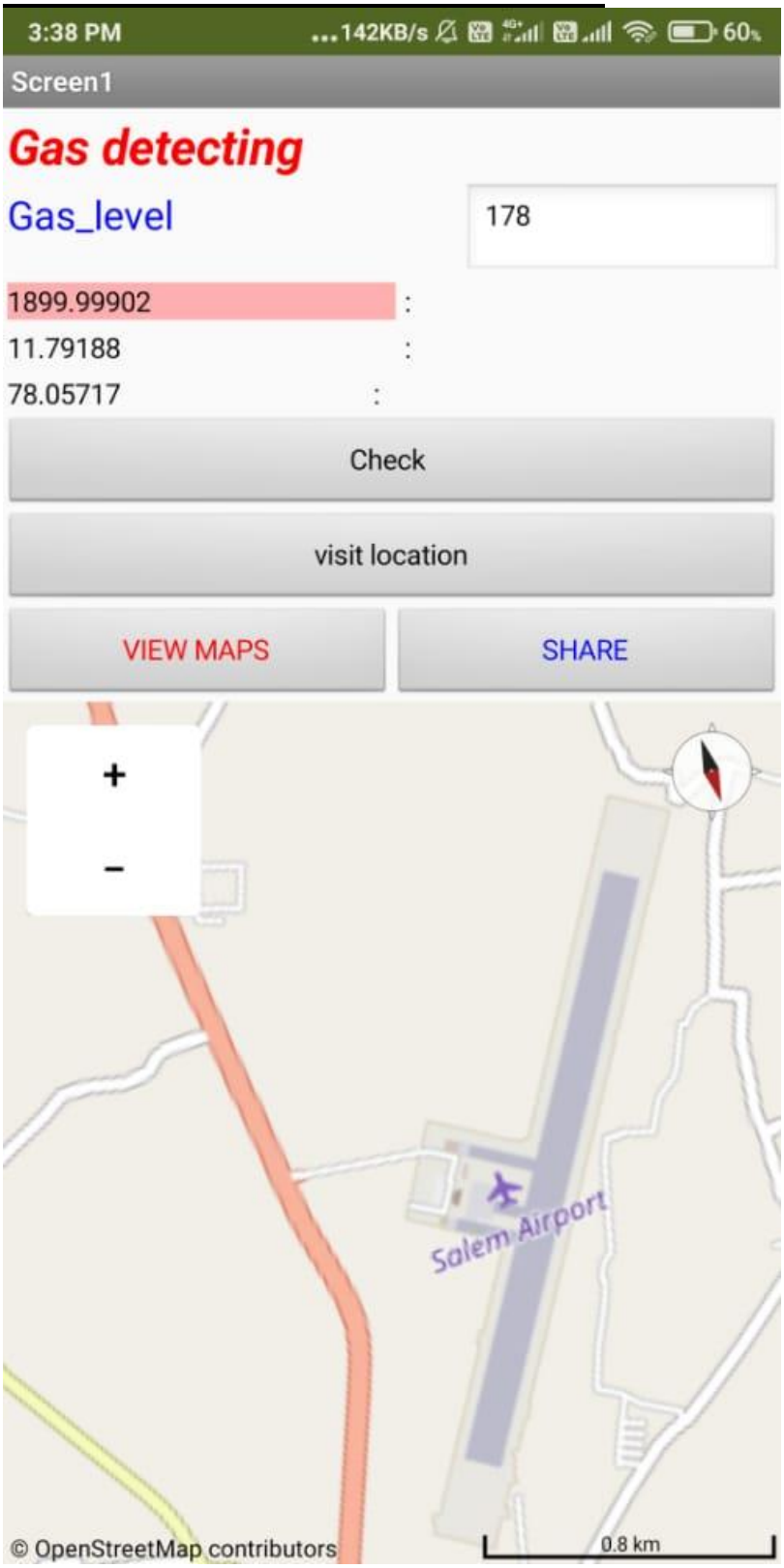
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	6	0	0	6
Gas leakage detection	3	0	0	3
Different level gas detection	2	0	0	2
Gas leaking location	3	0	0	3
Client Application	10	0	0	10
Security	5	0	0	5
Outsource Shipping	4	0	0	4
Exception Reporting	7	0	0	7
Final Report Output	4	0	0	4
Version Control	3	0	0	3

9. RESULTS:

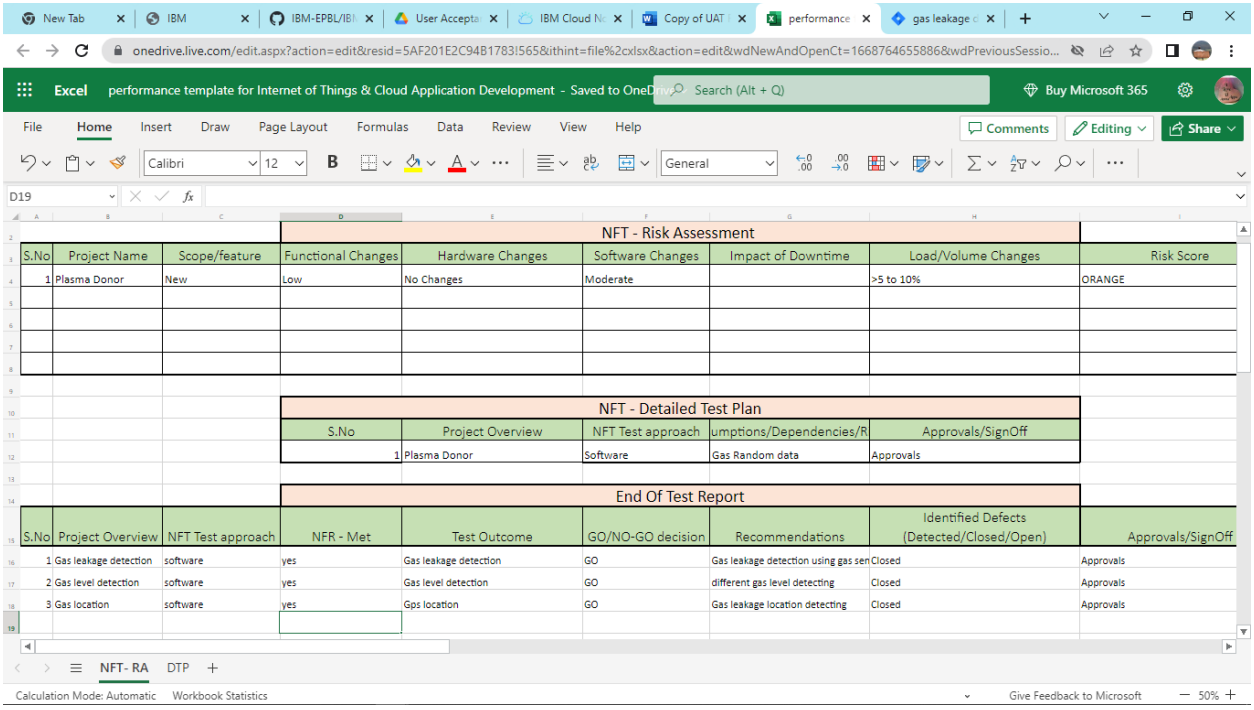
NODE RED UI OUTPUT:



MIT APP INVERTER RESULT :



9.1 PERFORMANCE METRICES:



NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volume Changes	Risk Score
1	Plasma Donor	New	Low	No Changes	Moderate		>5 to 10%	ORANGE

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	umptions/Dependencies/R	Approvals/SignOff
1	Plasma Donor	Software	Gas Random data	Approvals

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Gas leakage detection	software	yes	Gas leakage detection	GO	Gas leakage detection using gas sen	Closed	Approvals
2	Gas level detection	software	yes	Gas level detection	GO	different gas level detecting	Closed	Approvals
3	Gas location	software	yes	Gps location	GO	Gas leakage location detecting	Closed	Approvals

10. ADVANTAGES & DISADVANATAGE:

ADVANTAGES:

- By using this project, we can identify the gas leakage and we can control the gas overflow.
- In this we can see the location of that particular area.
- Rapid precautions will taken by the incharge persons so that the safety of the employees is assured.
- Many gas industries can be benefited by this idea.

DISADVANTAGE:

- Sometimes GPS may show the error and the location generated may wrong.
- If the network connection is lost , the entire process will stop.
- In the open environment the gas leakage detection is very hard.
- Maintenance of the components is mandatory.

11. CONCLUSION:

After this project performance can conclude that detection of the LPG gas leakage is incredible in the project system. Applicable usefully in the industrial and domestic purpose. In danger situations we are able to save the life by using this system. A sensor node sense gas. The simple procedures and ESP32 are used to build the sensor. As a result of this project user can see the gas leakage area along with the latitude and longitude points and gas level also.

12 .FUTURE SCOPE:

Based on the conclusion of the research, the proponents formulated several recommendations for the study. The group would like to recommend to the future researchers that they continue to develop this prototype device to find a way to include the manipulation of GAS leakage and anything that will help to the proposed project.

13 APPENDIX:

SOURCE CODE:

PYTHON CODE:

PYTHON CODE:

```
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random
#Provide your IBM Watson Device Credentials
organization = "x6troc"
deviceType = "PNT2022TMID29935"
deviceId = "6374679606"
authMethod = "token"
authToken = "eSJ(wSv_kaOwuZ?yIl"
# Initialize GPIO
def myCommandCallback(cmd):
print("Command received: %s" % cmd.data['command'])
```

```
status=cmd.data['command']
if status=="lighton":
print ("led is on")
else :
print ("led is off")
#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()
while True:
#Get Sensor Data from DHT11
gas=random.randint(0,200)
data = { 'gas' : gas }
def myOnPublishCallback():
if gas>100:
data = { 'gas' : gas }
print ("Published gas_level = %s ppm" % gas, "//Gas alert!!")
```

```
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

# Disconnect the device and application from the cloud

deviceCli.disconnect()
```

EMBEDDED CODE:

```
#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>

// Watson IoT connection details

#define MQTT_HOST "x6troc.messaging.internetofthings.ibmcloud.com" //Organization ID.messaging.internetofthings.ibmcloud.com

//change 3xr4l4

#define MQTT_PORT 1883

#define MQTT_DEVICEID "6374679606" //d:Organization ID:Device Type:Device ID

//change 3xr4l4

#define MQTT_USER "use-token-auth"

#define MQTT_TOKEN "eSJ(wSv_kaOwuZ?yIl" // change your auth_id :

#define MQTT_TOPIC "iot-2/evt/status/fmt/json"
```

```
#define MQTT_TOPIC_DISPLAY "iot-2/cmd/display/fmt/json"
// Add WiFi connection information
char ssid[] = "raspberr"; // your network SSID (name)
char pass[] = "dayo2022"; // your network password
// MQTT objects
void callback(char* topic, byte* payload, unsigned int length);
WiFiClient wifiClient;
PubSubClient mqtt(MQTT_HOST, MQTT_PORT, callback, wifiClient);
// variables to hold data
StaticJsonDocument<100> jsonDoc;
JsonObject payload = jsonDoc.to<JsonObject>();
JsonObject status = payload.createNestedObject("d");
static char msg[50];
float h;
void callback(char* topic, byte* payload, unsigned int length) {
// handle message arrived
Serial.print("Message arrived [");
Serial.print(topic);
Serial.print("] : ");
payload[length] = 0; // ensure valid content is zero terminated so can treat as c-string
Serial.println((char *)payload);
}
void setup() {
// Start serial console
```

```
Serial.begin(115200);
Serial.setTimeout(2000);
while (!Serial) { }
Serial.println();
Serial.println("ESP8266 IBM Cloud Application");
// Start WiFi connection
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, pass);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi Connected");
// Start connected devices
// Connect to MQTT - IBM Watson IoT Platform
if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN))
{
Serial.println("MQTT Connected");
mqtt.subscribe(MQTT_TOPIC_DISPLAY);
} else {
Serial.println("MQTT Failed to connect!");
ESP.reset();
}
}
```

```
void loop() {
mqtt.loop();
while (!mqtt.connected()) {
Serial.print("Attempting MQTT connection...");
// Attempt to connect
if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN))
{
Serial.println("MQTT Connected");
mqtt.subscribe(MQTT_TOPIC_DISPLAY);
mqtt.loop();
}
else {
Serial.println("MQTT Failed to connect!");
delay(5000);
}
}
h = random(1, 400);
// uncomment this line for centigrade
// t = dht.readTemperature(true); // uncomment this line for Fahrenheit
// Check if any reads failed and exit early (to try again).
if (h > 400) {
// Send data to Watson IoT Platform
status["gas_level"] = h;
serializeJson(jsonDoc, msg, 50);
Serial.println(msg);
```

```
if (!mqtt.publish(MQTT_TOPIC, msg)) {
Serial.println("MQTT Publish failed");
}
}
// Pause - but keep polling MQTT for incoming messages
for (int i = 0; i < 10; i++) {
mqtt.loop();
delay(1000);
}
}
```

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-35267-1660283072/blob/main/Project%20development%20phase/Sprint%201/sprint-1.pdf

Project video demo link:

Gas leakage detection and monitoring system-IBM project Team Id-PNT2022TMID29935