PROJECT REPORT

Team Members	Kommi sai Venkata pavan kumar - 410719106050 Koturu Srihari Sai - 401719106052 Parvathala Suneel Kumar - 410719106104 Singiri Chanukya - 410719106095
Team ID	PNT2022TMID28798
Project Name	Project –Gas leakage monitoring and alerting system for industries

INTRODUCTION

Project Overview

Gas Detectors have been in the market for a very long time and have been vastly used. They have wide range of applications and can be found in industrial plants, refineries, pharmaceutical manufacturing, paper pulp mills, aircraft and ship-building facilities, wastewater treatment facilities, vehicles, indoor air quality testing and homes. There are a lot of ways in which the Gas Detectors could be characterized. They are categorized on the basis of what type of gas they detect, what is the technology behind the making of the sensor and sometimes even the components which are used that affect their operation mechanism (semiconductors, oxidation, catalytic, photoionization, infrared, etc.). Gas Detectors are also widely characterized as fixed or portable detectors. They are characterized on the basis of which category of risk they fall in, ExOx-Tox, the three categories of risk - Ex – Risk of explosion by flammable gases - Ox – Oxygen Risk of asphyxiation by oxygen displacement Risk of increase of flammability by oxygen enrichment - Tox – Risk of poisoning by toxic gases, the list of categorization goes on. As a result we cannot have a single system or a group of systems which we can call the best but instead there is a plethora of devices available for matching the varying user requirements.

Purpose

The gas detectors can be used for the detection of combustible, flammable and poisonous gases and for loss of oxygen, and also to detected a gas leak or other pollutants. It makes the area where the leak occurs an warning sound and instructs operators to leave the area.

LITERATURE REVIEW

Existing Problem

In industries,the existing Problem in gas monitoring is that there is no efficient system for monitoring the gas leakage, the good system are of high cost and also the installation process is too complicated.Then the affordable of the system is high and the systems are sometimes making disasters and the number of sensors is unpredictable and the positioning of equipment is improper

References

1) Bing Han, Qiang Fu, Hanfang How, 'Methane Leakage Monitoring Technology For Natural Gas Stations And Its Application', IEEE 5th International Conference on Computer and Communications,2001.

2) Shruthi Unnikrishnan,1 Mohammed Razil, Joshua Benny, Shelvin Varghese and C.V. Hari, 'LPG Monitoring And Leakage Detection System', Department of Applied Electronics and Instrumentation Engineering, Rajagiri School of Engineering and Technology, Rajagiri Valley, Kakkanad, Kochi, India.

3) J.Vijayalakshmi, Dr.G.Puthilibhai, S.R.Leoram Siddarth, 'Implementation Of Ammonia Gas Leakage Detection & Monitoring System Using Internet Of Things', West Tambaram, Chennai.

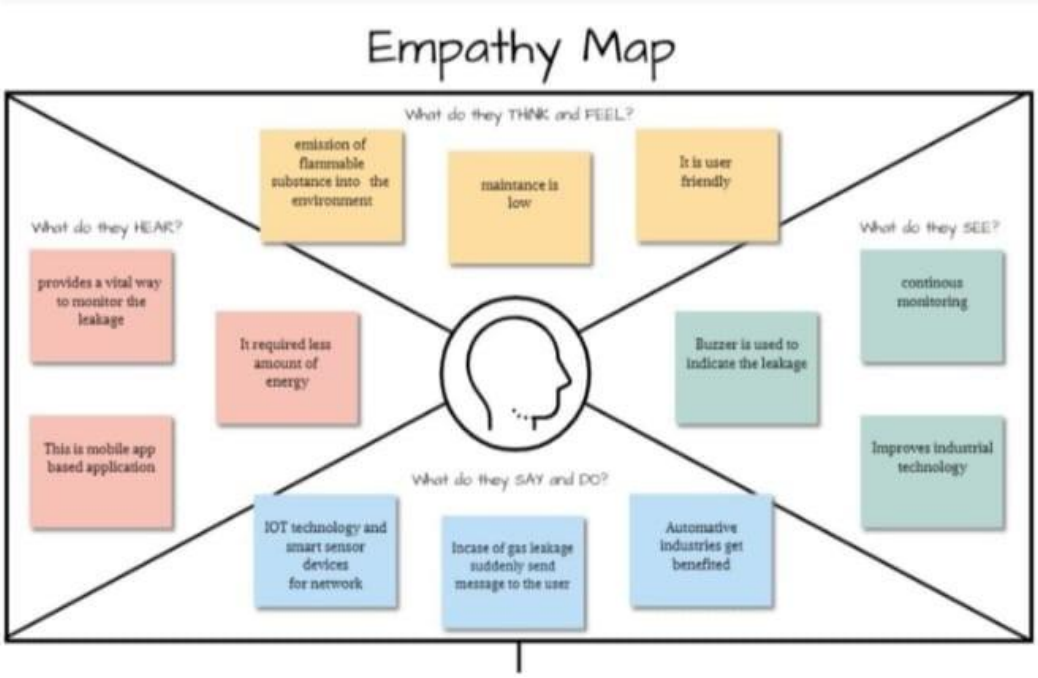
4) Makiko Kawada, Tadao Minagawa, Eiichi Nagao, Mitsuhito Kamei, Chieko Nishida and Koji Ueda, 'Advanced Monitoring System For Gas Density Of GIS', Mitsubishi Electric Corporation

Problem Statement Definition

For monitoring gas leakage in the industry and Control the gas leakage ,we create a system for monitoring gas leakage and makes the installation propose simple.

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas



What we think to create device which helps us to control emission of flammable substance into the environment. It should be user friendly and low cost for maintenance. For that we see continuous monitoring device and buzzer is to indicate the leakage.

Ideation & Brainstorming

The ideas are In case of higher gas leakage and fire accidents, a notification can be given to the fire station and hospital through software application.The level of gas in the industry can be informed through speakers periodically.When gas gets leaked, a notification can be passed to hospital.Sensor can be placed in the entrance for counting the workers who have been moved out in case of emergency.

In addition to alarm, a voice notes which alerts by saying the level of leakage can be designed.The alerting message can also be forwarded to the management of the industry.Sprinklers or extinguishers can be fixed which helps in case of inflammation by the leakage.Windows and gates can be opened automatically through sensors placed on that.

Proposed Solution

To Develop an efficient system & an application that can monitor and alert the users(workers), our product 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. In the web application, admins can view the sensor parameters. It is fastest alerts to the workers and user friendly. For social impact it is Cost efficient and easy installation and provide efficient results and can work with irrespective of fear.Since the product is cost efficient, it can be placed in many places in the industries. Even when the gas leakage is more, the product sense the accurate values and alerts the workers effectively.

Problem Solution fit

REQUIREMENT ANALYSIS

Functional Requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Visibility	Level of gas can be monitored by users if there is any leakage, alerts can be sent through messages.
FR-2	User Reception	The data like the level of gas can be send through messages
FR-3	User Understanding	The user can monitor the level of gas with the help of the data. If there is an increase in gas level then the alert will be given. They also get notified by the alert
FR-4	User Convenience	Through message we can easily get data of gas level and in case of gas leakage, it can directly send notifications to nearby police station and hospital.
FR-5	User Performance	When the user gets notified, he could turn on the exhaust fan/sprinkler

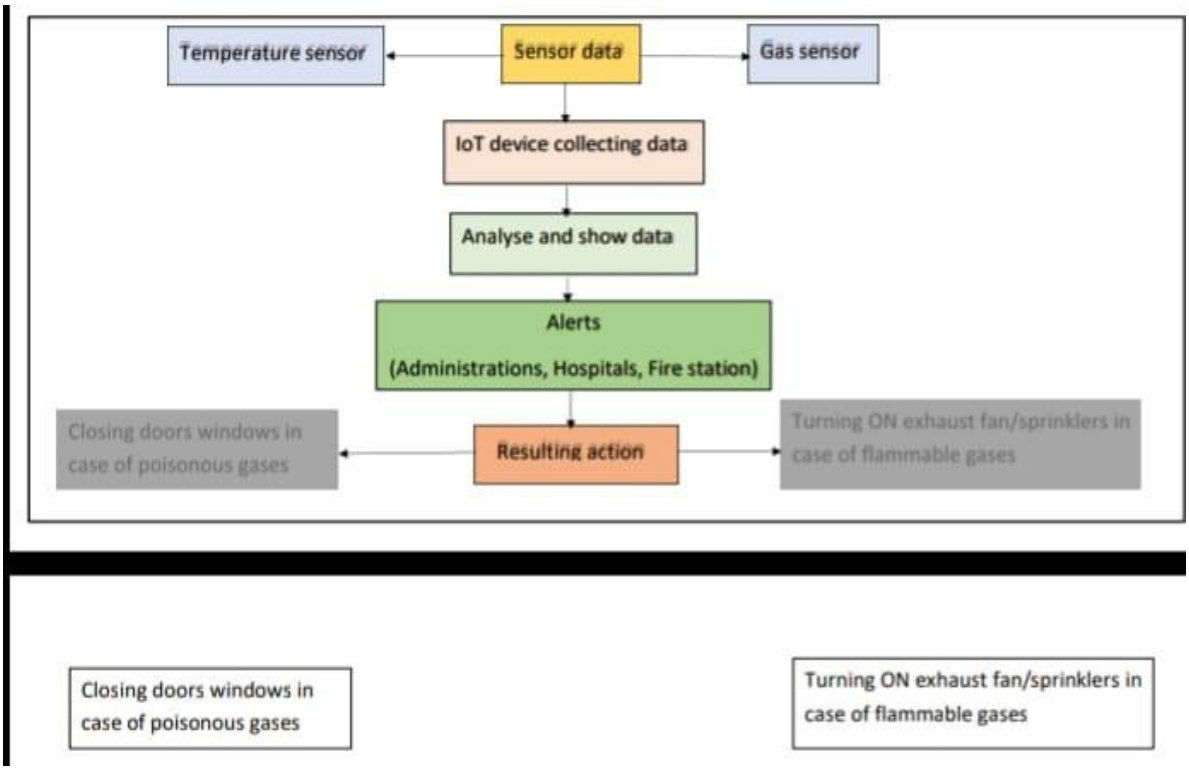
Non-Functional Requirement

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	It updates the data regularly as well as protects the workers.
NFR-2	**Security**	As a result of emergency alert, we can be able to protect both the humans and properties.
NFR-3	**Reliability**	Can be able to provide accurate values. It might have a capacity to recognize the smoke accurately and does not give a false
NFR-4	**Performance**	Sprinklers and exhaust fans are used in case of emergency.
NFR-5	**Availability**	It can be used for everyday; it includes day and nights.
NFR-6	**Scalability**	Sensors can be replaced every time it fails.

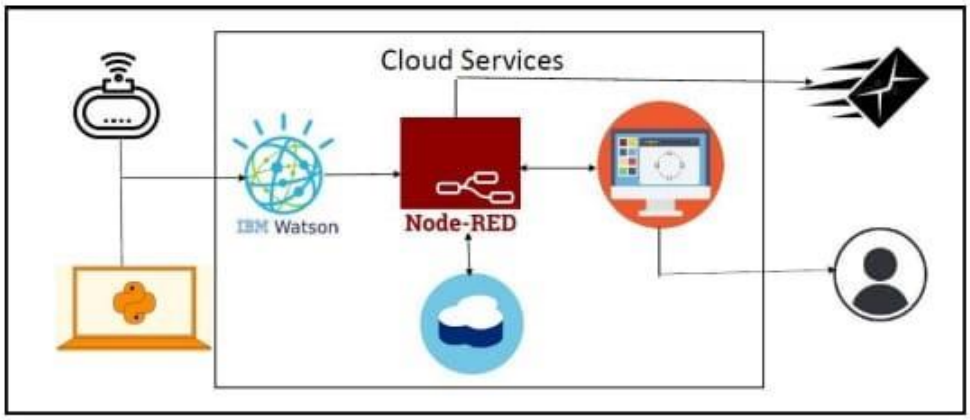
PROJECT DESIGN

Data Flow Diagram



This is the data flow diagram of gas leakage monitoring and detection.Here the data from temperature sensor and gas sensor is collected from IOT device and the data is analyzed. If the alert action requires it alerts and the requied measures are taken.

Solution & Technical Diagram



This is the technical diagram of gas leakage monitoring and detection.Here the data from temperature sensor and gas sensor is collected and is connectd to IBM watson(cloud) . Node red is connectd to cloud and the result of the data from the cloud flows .

User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	User can enter into the web application	I can access my account / dashboard	High	Sprint-1
		USN-2	User can register their credentials like email id and password	I can receive confirmation email & click confirm	High	Sprint-1
	Login	USN-3	User can log into the application by entering email & password	I can login to my account	High	Sprint-1
	Dashboard	USN-4	User can view the temperature	I can view the data given by the device	High	Sprint-2
		USN-5	User can view the level of gas	I can view the data given by the device	High	Sprint-2
Customer (Web user)	Usage	USN-1	User can view the web page and get the information	I can view the data given by the device	High	Sprint-3
Customer	Working	USN-1	User act according to the alert given by the device	I can get the data work according to it	High	Sprint-3
		USN-2	User turns ON the exhaust fan/sprinkler when the leakage occurs	I can get the data work according to it	High	Sprint-4
Customer Care Executive	Action	USN-1	User solve the problems when someone faces any usage issues	I can solve the issues when some one fails to understand the procedure	High	Sprint-4
Administrator	Administration	USN-1	User stores every information	I can store the gained information	High	Sprint-4

PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	IDE	USN-1	Installing all the softwares which is required like python IDE	2	High	Udaya keerthi VS Surendhran P Sudharsan NS Sasi Varunan G
Sprint-1	Checking the simulation with conditions	USN-1	Simulating the circuits and experimenting	2	High	Udaya keerthi VS Sasi Varunan G Surendhran P Sudharsan NS

Sprint-2	Software	USN-2	·IBM Watson Iot ·NodeRed integration	2	High	Udaya Keerthi VS Sudharsan NS Surendhran P Sasi Varunan G
Sprint-2	Software	USN-2	Test the device and workflow.	2	High	Udaya Keerthi VS Sudharsan NS Surendhran P Sasi Varunan G
Sprint-3	Application Development	USN-3	Using MIT App Inventor create an App	2	High	Udaya Keerthi VS Sudharsan NS Surendhran P Sasi Varunan G
Sprint-3	Testing	USN-3	Testing the Application.	2	High	Udaya Keerthi VS Sudharsan NS Surendhran P Sasi

						Varunan G
Sprint-4	WEB UI	USN-4	User interface with the Software	2	High	Udaya Keerthi VS Sudharsan NS Surendhran P Sasi Varunan G

Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Resources Initialization	Create and initialize accounts in various public APIs like OpenWeatherMap API.	1	LOW	Surendhiran Sasi Varunan Sudharsan Udaya Keerthi
Sprint-1	Local Server/Software Run	Write a Python program that outputs results given the inputs like weather and location.	1	MEDIUM	Surendhiran Sasi Varunan Sudharsan Udaya Keerthi
Sprint-2	Push the server/software to cloud	Push the code from Sprint 1 to cloud so it can be accessed from anywhere	2	MEDIUM	Surendhiran Sasi Varunan Sudharsan Udaya Keerthi
Sprint-3	Hardware initialization	Integrate the hardware to be able to access the cloud functions and provide inputs to the same.	2	HIGH	Surendhiran Sasi Varunan Sudharsan Udaya Keerthi
Sprint-4	UI/UX Optimization & Debugging	Optimize all the shortcomings and provide better user experience.	2	LOW	Surendhiran Sasi Varunan Sudharsan Udaya Keerthi

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

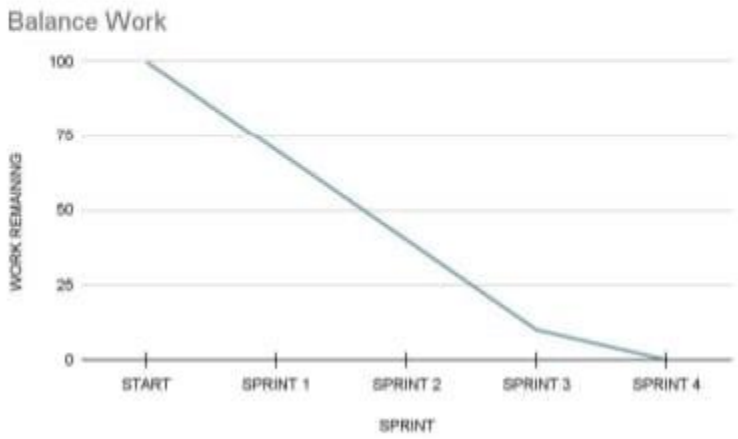
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	31 Oct 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	07 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	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 the team's average velocity (AV) per iteration unit (story points per day)

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

Burndown Chart:



CODING & SOLUTIONING

Feature 1 Code: import time

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

#Provide your IBM Watson Device Credentials organization =
"6a4pz2" deviceType = "Node_1"
 import random

#Provide your IBM Watson Device Credentials organization =
"pi0ywk" deviceType = "Node_1"
deviceId = "12345"
 authMethod = "use-token-auth"
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
```

```
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status == "alarmon":
        print ("Alarm is on please all Evacuate Fans On")
    elif status == "alarmoff":
        print ("Alarm is off and Fans Off")
    elif status == "sprinkleron":
        print ("Sprinkler is On Evacuate Faster")
    elif status == "sprinkleroff":
        print("Sprinkler is Off")
    else:
        print("Please send proper command")
    #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
    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 random function

        temp=random.randint(0,120)
        Humid=random.randint(0,100)
```

```
        gas=random.randint(0,1500)
        data={'temp':temp,'Humid':Humid,'gas':gas}
        #print data
        def myOnPublishCallback():
            print (" Published Temperature = %s C" % temp, "Humidity = %s
%%" % Humid, "Gas_Level = %s ppm" %gas, "to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("\n Not connected to IoTF")
        if temp>60 :
            print("\n Fire Detected due to gas Leak ! Alarm ON! Sprinkler
ON! Call The Fire Police \n")
        elif gas>350:
            print("\n Gas is Leaking \n")

        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud deviceCli.disconnect()
```

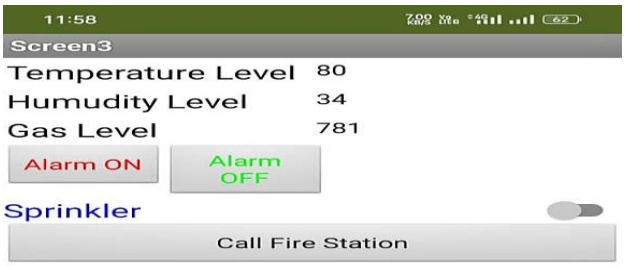
In this code:

```
        if temp>60 :
        print("\n Fire Detected due to gas Leak ! Alarm ON! Sprinkler
ON! Call The Fire Police \n")
        elif gas>350:
        print("\n Gas is Leaking \n")
```

if temperature is greater than 60 the fire could be occured ,then the alaram will be on to alert and the sprinklers on . else if gas is grater than 350 then the gas leakage indication only occur.

Feature 2

We have developed the application in MIT app inventer which can monitor the temperature ,humidity and gas leakage.It also has the features like sprinklers and alarm which will indicate .If the situation is uncontrollable ,we can call the fire station through the Call fire Station button.



TESTING

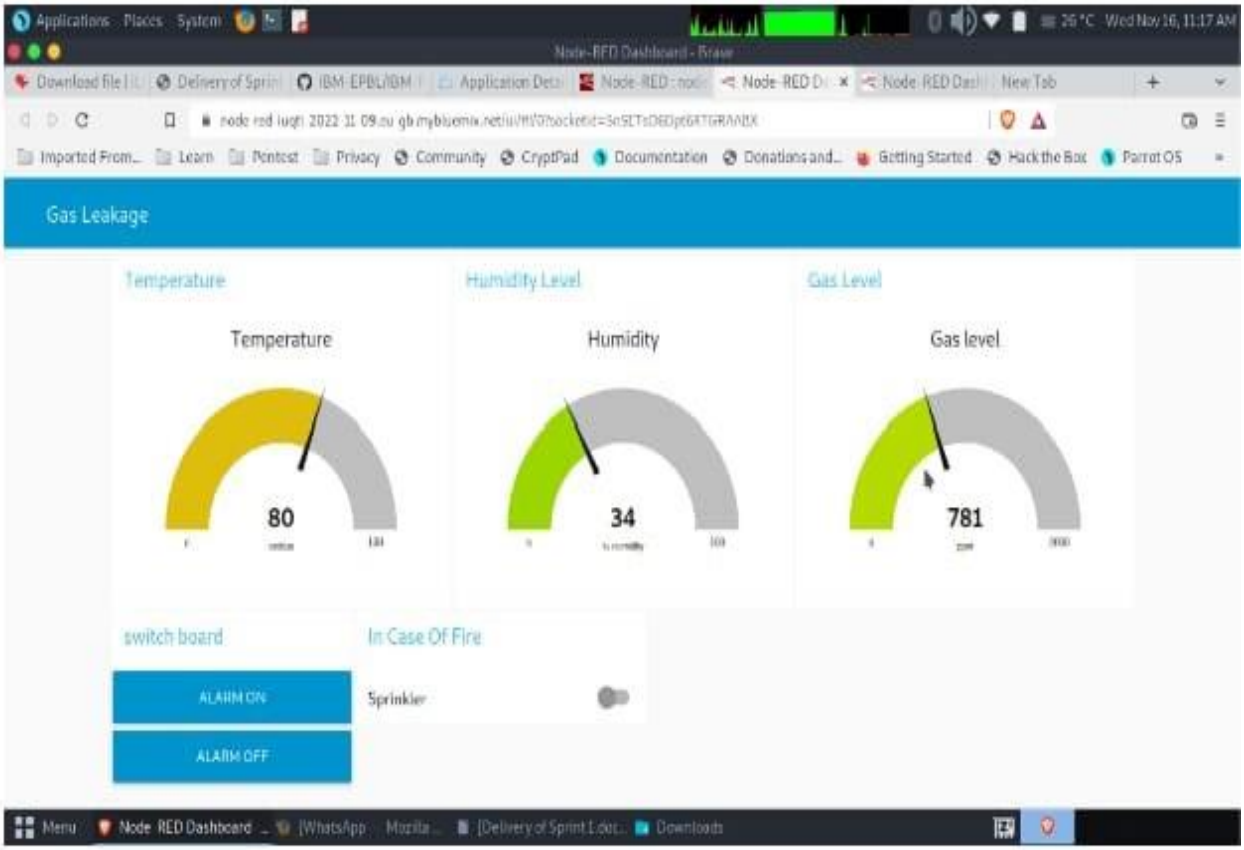
Test Cases

ALL PASSED

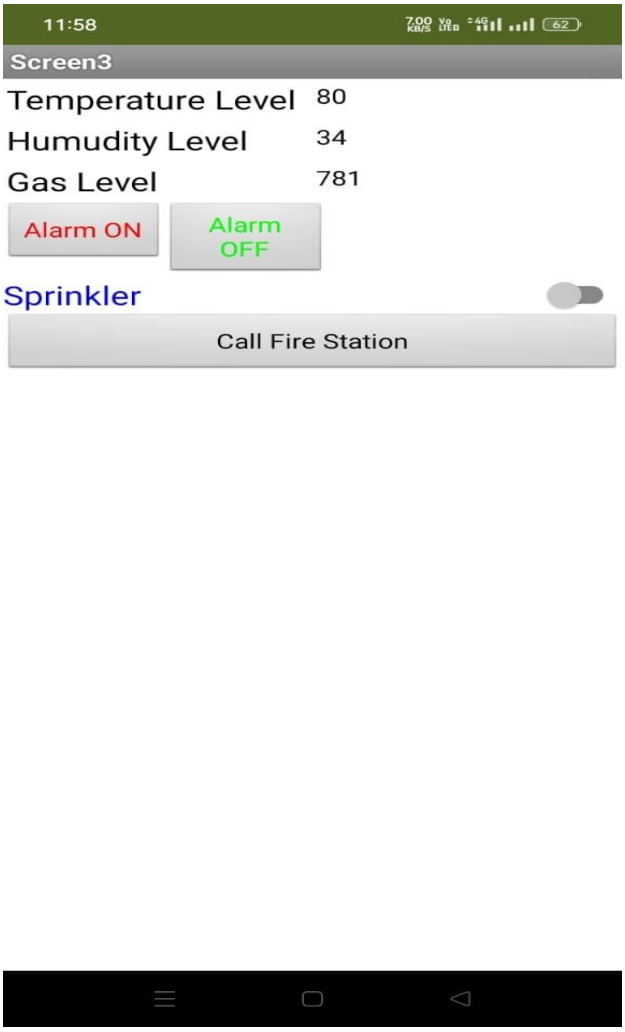
RESULTS

Performance Metrics

Below image represents the result of node red dash board



The next Image below here represents the output of MIT-inventer :



ADVANTAGES & DISADVANTAGES

Advantages:

1) Installation process is simple.

2) Efficient system for monitoring

Disadvantages :

1) As it involves IOT it requires high speed internet connectivity to give accurate results
2) There is a possibility of ignoring real signal as false alarm

CONCLUSION AND FUTURE SCOPE

The proposed gas leakage detector is promising in the field of safety. The attempt while making this prototype has been to bring a revolution in the field of safety against the leakage of harmful and toxic gases to minimize and hence nullify any major or minor hazard being caused due to them. Nevertheless there is always scope of improvement and some of the features that will improve the system and make it even better and reliable have been mentioned below:

A. Extended Features of System

The behaviour of the gases is dependent on the temperature and humidity of the air around. A gas at certain concentration might not be flammable at low temperature but might have explosive nature at high temperature. For this reason addition of a Temperature and Humidity Sensor will be very helpful.

B. Performing Big Data Analytics on the sensor readings

Analytics could be performed on the sensor readings. The readings from sensors could be used for forming predictions of situations where there can be a mishap. Instead of straightaway alarming when the concentrations have gone high, algorithms could be worked upon which could determine such situations prior to their occurrence. Combining the gas sensor readings with the readings from temperature and humidity sensor would increase the precision of the system. The cases of false alarms being raised will reduce down to very small percentages.

C. Dedicated Application for System

A dedicated mobile application could be made for the system. The features of the application would be:

1. Getting the details of the concentration levels of the house within a tap of a button.
2. Since it is a safety device it is important for it to be perfectly calibrated and maintained at all times. The app can make sure to send reminders about getting the system checked every once in a while.
3. The user can add or remove the recipients who will receive the information of leakage whenever they require

Source Code

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


#Provide your IBM Watson Device Credentials organization =
"6a4pz2”
deviceType = "Node_1" deviceId = "12345"
 authMethod = "use-token-auth"
 authToken = "12345678"

# Initialize GPIO


def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status=cmd.data['command']
    if status == "alarmon":
        print ("Alarm is on please all Evacuate Fans On")
    elif status == "alarmoff":
        print ("Alarm is off and Fans Off")
    elif status == "sprinkleron":
        print ("Sprinkler is On Evacuate Faster")
    elif status == "sprinkleroff":
        print("Sprinkler is Off")
    else:
        print("Please send proper command")
    #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 random function

      temp=random.randint(0,120)
      Humid=random.randint(0,100)
      gas=random.randint(0,1500)
      data={'temp':temp,'Humid':Humid,'gas':gas}
      #print data
      def myOnPublishCallback():
         print (" Published Temperature = %s C" % temp, "Humidity = %s
%%" % Humid, "Gas_Level = %s ppm" %gas, "to IBM Watson")

      success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_publish=myOnPublishCallback)
      if not success:
         print("\n Not connected to IoTF")
      if temp>60 :
         print("\n Fire Detected due to gas Leak ! Alarm ON! Sprinkler
ON! Call The Fire Police \n")
      elif gas>350:
         print("\n Gas is Leaking \n")

      time.sleep(10)
```

```
    deviceCli.commandCallback = myCommandCallback

# Disconnect the device and application from the cloud deviceCli.disconnect()
```

Github

GITHUB : **https://github.com/IBM-EPBL/IBM-Project-51519-1660980228**