PROJECT REPORT

PROJECT NAME	GAS LEAKAGE MONITORING & ALERTING SYSTEM FOR INDUSTRIES
TEAM MEMBERS	DINESH KUMAR M AJITH KUMAR M AJAY KUMAR S BALA SUBRAMANIAN S

CONTENTS

1.INTRODUCTION

1.1 Project Overview:

The internet of Things is a developing topic of technical, social, and economic significance. The usage of the gas brings great problems in the domestic as well as working places. The inflammable gas, which is excessively used in the work places (Industries). The leakage of the gas causes destructible impact to the lives and as well as to the heritage of the people. Most of the societies have fire safety mechanism. But it can use after the fire exists. As a result, a system for detecting and monitoring gas leaks is required. Through a flame sensor, the system will sense fire and flame. The buzzer begins to ring when a fire is detected. Once the leakage is detected it will alert the user about the leakage. The performance that was produced showed that it was successful in detecting the gas leakage

1.2 Purpose:

The design of a sensor-based automatic gas leakage detector with an alert system has been proposed. This is an affordable, less power using, lightweight, portable, safe, user friendly, efficient, multi featured and simple system device for detecting gas. To monitor this gas leak, the system includes an MQ6 gas detector. This sensor detects the amount of leaking gas present in the surrounding atmosphere. In this way, the consequences of an explosion or gas leak can be avoided.

2.LITERATURE SURVEY :

2.1 Existing Problem:

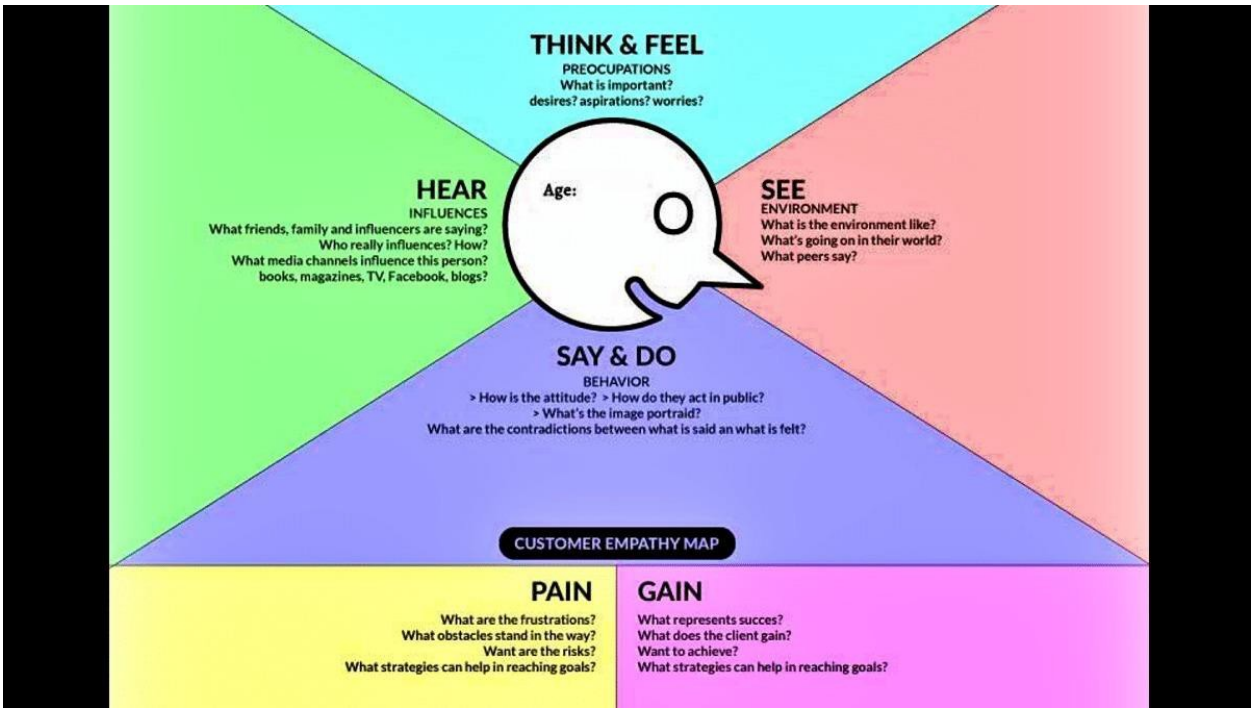
Gas leakage is nothing but the leak of any gaseous molecule from a pipeline, or cylinder etc in the industries. Gas Leakages in open or closed areas can prove to be dangerous .This can occur either purposefully or even unintendedly. As we are aware that these kinds of leaks are dangerous to our health, and when it becomes explosive it could cause great danger to the people, industry and the environment. Therefore, we have used IoT technology to make a Gas Leakage Detector for society which has Smart Alerting techniques involving sending a text message to the concerned authority and the ability to view their gas level to prevent the gas leak. This will detect the harmful gases in the environment and alerting everyone through sending notifications.

2.2 Problem Statement Definition:

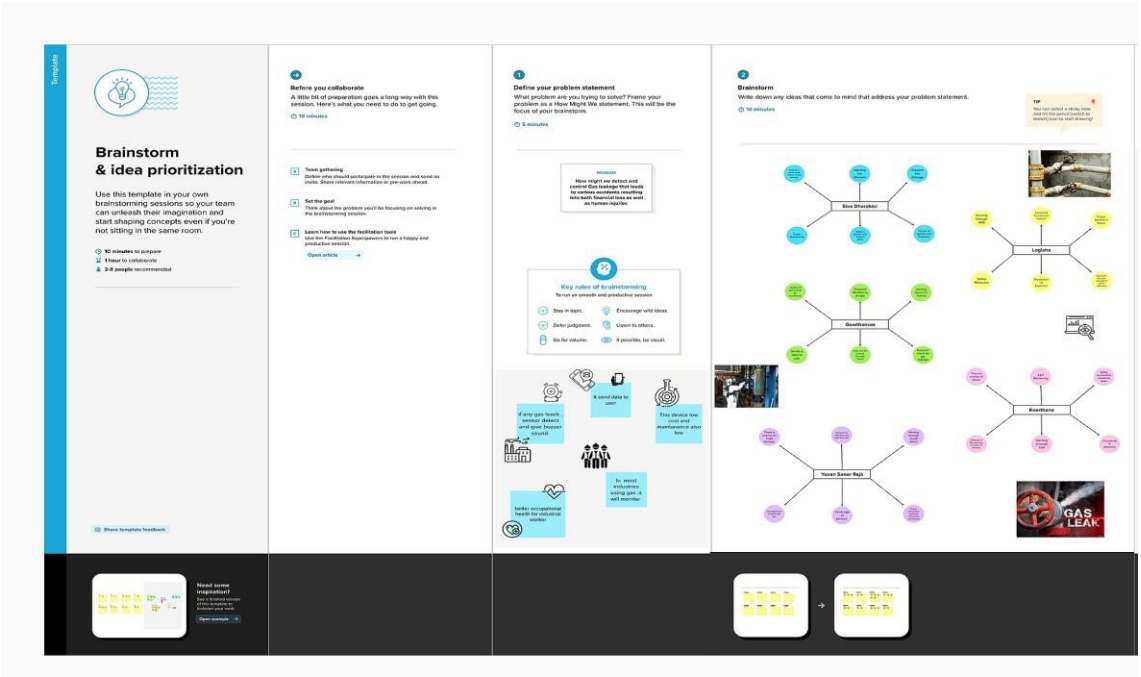
In most industries, one of the key parts of any safety plan for reducing risks to personnel and plant is the use of early-warning devices such as gas detectors. These can help to provide more time in which to take remedial or protective action. They can also be used as part of a total, integrated monitoring and safety system for an industrial plant. Rapid expansion of oil and gas industry leads to gas leakage incidents which are very serious and dangerous. Solutions need to be found out at least to minimize the effects of these incidents since gas leaks also produce a significant financial loss. Solution should be in a way to reduce the gas leak by obtaining gas level and alerting the concerned authority by sending alert messages.

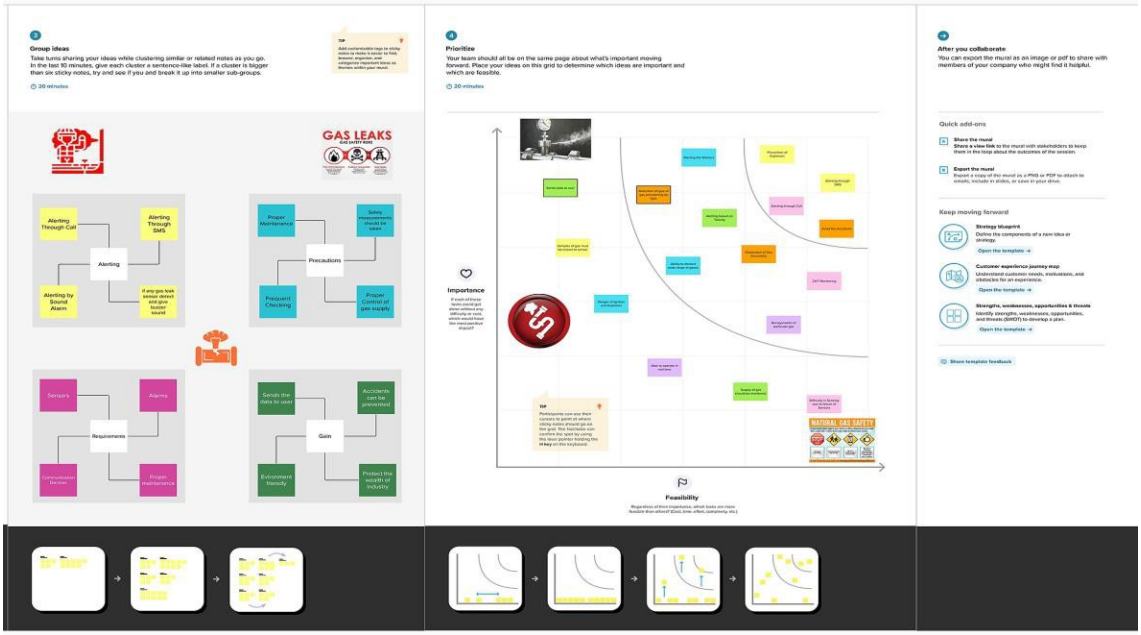
3. IDEATION & PROPOSED SOLUTION:

3.1 Empathy Map Canvas:

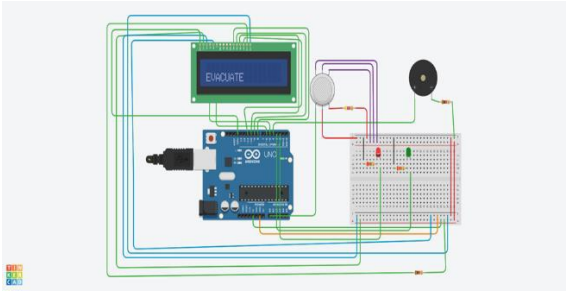


3.2 Ideation & Brainstorming:

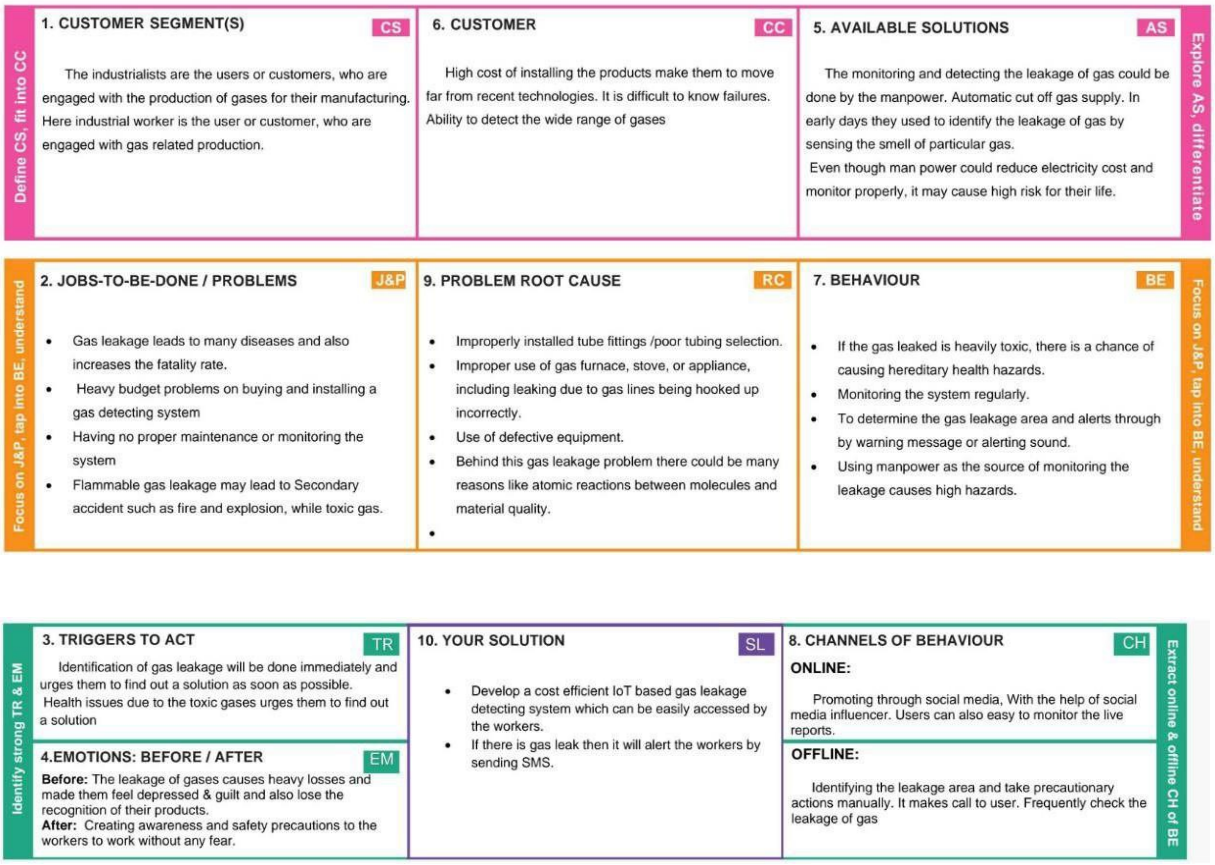




3.3 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To detect the gas leakage to alert the user through notification
2.	Idea / Solution description	In order to have a control over such conditions we proposed system that uses sensors which is capable of detecting the gases such as LPG, CO2, CO and CH4. This system will not only able to detect the leakage of gas but also alerting through audible alarms.
3.	Novelty / Uniqueness	• Ability to predict the hazardous situation • Low cost
4.	Social Impact / Customer Satisfaction	• This model is vital for the society as there are lot of people unable to detect the gas leakage prior the fire accident. • we have used the IoT technology to make a Gas Leakage Detector for society which having Smart Alerting techniques involving sending text message to the concerned authority and an ability performing data analytics on sensor readings.
5.	Business Model (Revenue Model)	
6.	Scalability of the Solution	Develop a proposed system which include some safety factors.

3.4 Problem Solution fit:



4.

REQUIREMENT ANALYSIS:

4.1 Functional requirement:

Business Requirements	User Requirements	Product Requirements
The gas leakage detection system can be deployed in homes, hotels, factory units, LPG cylinder storage areas, and so on. The main advantage of this IoT and Arduino-based application is that it can determine the leakage and send the data over to a site. It can be monitored, and preventive measures can be taken to avoid any disaster.	The gas leakage detection system can be optimized for detecting toxic gasses along with upgrading them with smoke and fire detectors to identify the presence of smoke and fire. Ensuring worker safety is important but making using of the right technology is even more vital.	Detecting gasses is necessary regardless of your business role or individual purpose. Certain technologies at play make such IoT devices what they are, and if you want to indulge in IoT application development, you must know what they are and what purpose they can fulfill.

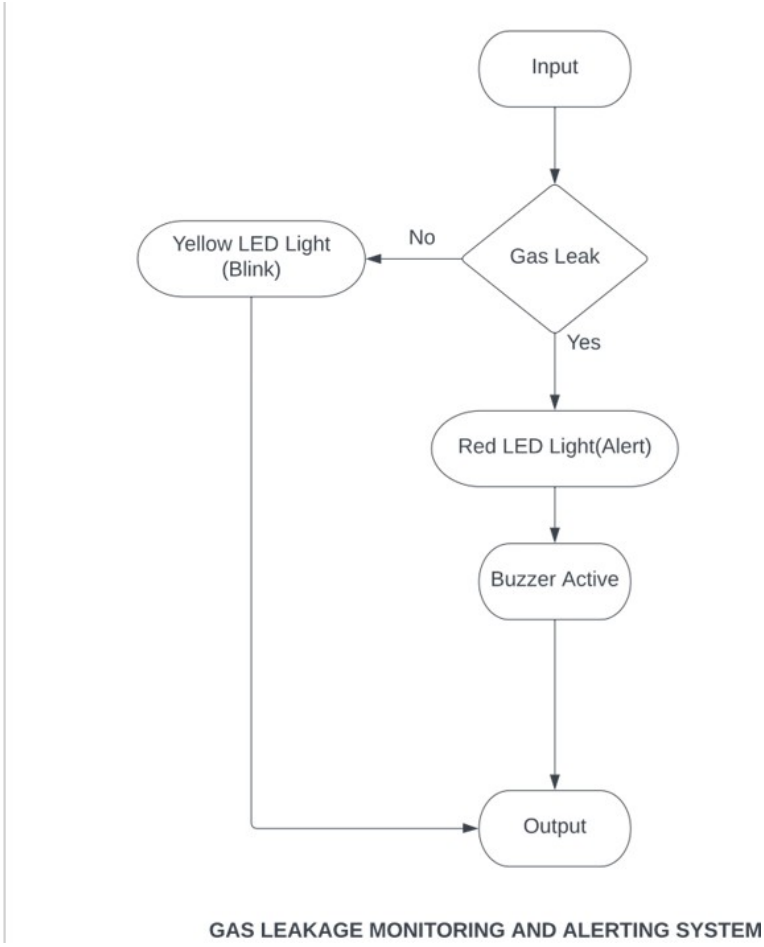
4.2 Non-Functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The sensors used to detect the gas leakage which helps to prevent the high risk of gas explosion and also can prevent the causalities within and outside the covering area of the industries.
NFR-2	Security	The device is intended for the use of industries or factories, where there is a use of explosive gas is a source of risk. This device will help and secures from the causes.
NFR-3	Reliability	Gas leakage detecting system detects the gas leakage at industries or factories which detects the small amount of gas leakage as soon and sends the alerting SMS to users.
NFR-4	Performance	The Gas leakage detecting system is a device with an alarm setting. Whenever there is a gas leak ,which is greater than the threshold level, the inbuild sensor detects and alerts the user within a minute much before it can cause any accidents.
NFR-5	Availability	The gas leakage detecting system is readily available in the market which is extremely expensive, but here we are providing a low-cost circuit for gas leakage detecting system and also it is user friendly

NFR-6	Scalability	The system is very simple and easy to maintain with cost efficient. A backup power supply will be included in the design to prevent from the power failure conditions. It has the capability to works for a period of time without any damage in the system components.

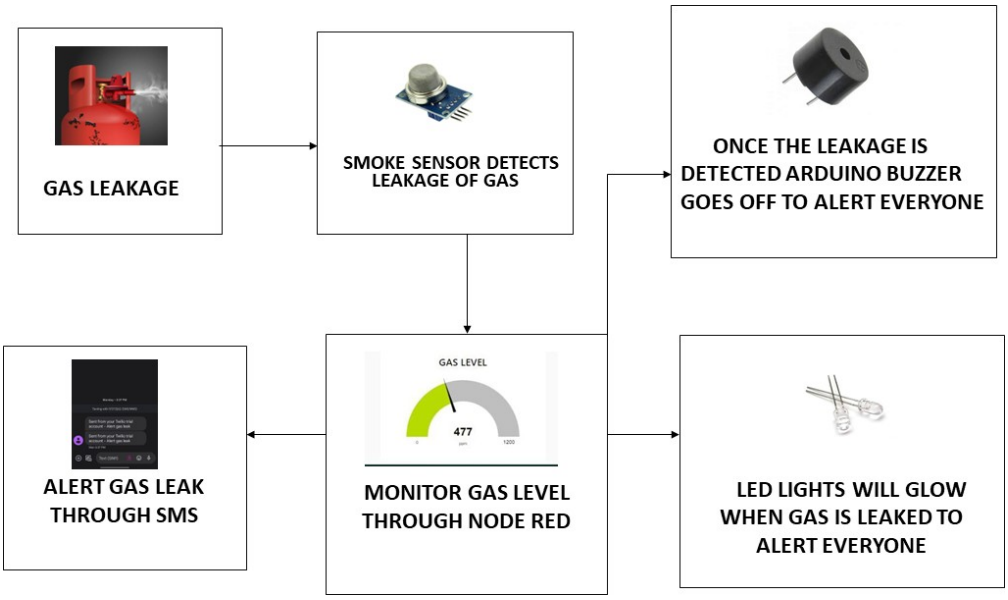
PROJECT DESIGN:

5.1 Data Flow Diagrams:



GAS LEAKAGE MONITORING AND ALERTING SYSTEM

5.2 Solution & Technical Architecture:



5.3 User Stories:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Monitor the gas leakage	USN-1	The Industrialist have own industries so the industry owner must take of workers.The workers have family so the industries give security assurance of workers.	2	High	Shanmugam S Vigneashwaran B Vishnu V Thirumurugan M
Sprint-2	Avoid From Disaster	USN-2	The gas leakage occur at the time fire service will take care to protect the people from the disaster.	1	High	Shanmugam S Vigneashwaran B Vishnu V Thirumurugan M
Sprint-3	Detect the gas	USN-3	We have monitor the gas by 24/7 hrs. To avoid leakage,the industry have quality pipes to transfer the gas and proper maintanence service once in a month. The industry must take care of what are the necessary process to avoid the gas leakage.	2	Low	Shanmugam S Vigneashwaran B Vishnu V Thirumurugan M
Sprint-4	The model is trained and tested by sample dataset.	USN-4	The programmer design the model to detect the gas leakage.	2	Medium	Shanmugam S Vigneashwaran B Vishnu V Thirumurugan M
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-5	Warning message	USN-5	Incase any gas leakage occur, the device give the alarm and alert message to concerned user within a minute.	1	High	Shanmugam S Vigneashwaran B Vishnu V

6 PROJECT PLANNING & SCHEDULING:

6.1 Sprint Planning & Estimation:

6.1.1 SPRINT PLAN

6.1.2 ANALYZE THE PROBLEM

6.1.3 PREPARE an ABSTRACT, PROBLEM STATEMENT

6.1.4 LIST A REQUIRED OBJECT NEEDED

6.1.5 CREATE A PROGRAM CODE AND RUN IT

6.1.6 MAKE A PROTOTYPE TO IMPLEMENT

6.1.7 TEST WITH THE CREATED CODE AND CHECK THE DESIGNED PROTOTYPE

6.2 Sprint Delivery Schedule:

Sprint	Functional Requirement (Epic)	User Story	User Story / Task	Story Point	Priority
Sprint-1	Create	US-1	Create the IBM Cloud services which are being used in this project.	5	High
Sprint-1	Configure	US-2	Configure the IBM Cloud services which are being used in completing this project.	1	Medium
Sprint-1	Create	US-3	IBM Watson IoT platform acts as the mediator to connect the web application to IoT devices, so create the IBM Watson IoT platform.	1	Medium
Sprint-1	Configure	US-4	Configure the IBM Watson IoT which are being used to display the output.	13	High

Sprint-2	Create	US-1	In order to connect the IoT device to the IBM cloud, create a device in the IBM Watson IoT platform and get the device credentials.	13	High
Sprint-2	Configure	US-2	Configure a device in the IBM Watson IoT platform and get the device credentials.	3	Medium
Sprint-2	Create	US-3	Create a Node-RED service.	3	High
Sprint-2	Configure	US-4	Configure the connection security and create API keys that are used in the Node- RED service for accessing the IBM IoT Platform.	1	Medium
Sprint-3	Develop	US-1	Develop a python script to publish random sensor data such as temperature, Flame level and Gas level to the IBM IoTplatform	1 3	High

Sprint-3	Configure	US-2	After developing python code and commands just run the code	1	Medium
Sprint-3	Print	US-3	Print the statements which represent the control of the devices.	1	Low
Sprint-3	Publish	US-4	Publish Data to The IBM Cloud	5	High
Sprint-4	Create	US-1	Create Web UI in Node- Red	5	High
Sprint-4	Configure	US-2	Configure the Node-RED flow to receive data from the IBMIoT platform	5	High
Sprint-4	Configure	US-3	Use cloudant DB nodes to store the received sensor data in the cloudant DB	5	High
Sprint-4	Publish	US-4	Publish the received data in webapplication	5	High

7. CODING & SOLUTIONING:

```
import time
import sys
import random
import  wiotp.sdk.device# IBM IoT Watson Platform Module
import ibmiotf.device
import tkinter as tk # Python GUI Package
from tkinter import ttk # Python GUI
import time
from threading import Thread



organization = "vens1r"
deviceType = "GASLEAKAGE"
deviceId = "1234"
authMethod = "token
authToken = "12345678"
# Tkinter root window
root = tk.Tk()
root.geometry('350x300')  # Set size of root window
root.resizable(True, True) # root window non-resizable
root.title('Gas Leakage Monitoring And Alerting System for Industries ')



# Layout Configurations
root.columnconfigure(0, weight=1)
```

```
    root.columnconfigure(1, weight=3)
     current_gas = tk.DoubleVar()
def get_current_gas(): # function returns current gas level value
        return '{: .2f}'.format(current_gas.get())
def slider_changed(event): # Event Handler for changes in sliders
        print(' -------')
        print('Gas Level: {: .2f}'.format(current_gas.get()))
        print(' -------')
        gas_label.configure(text=str(get_current_gas()) +" ppm")  # Displays current gas level as
    label content


    # Tkinter Labels

    # label for the gas level slider /
    slider_gas_label = ttk.Label(root,text='Set Gas Level:')
    slider_gas_label.grid(column=0,row=0,sticky='w')


    #  Gas Level slider
    slider_gas = ttk.Scale(root,from_=0,to=1300,orient='horizontal',
    command=slider_changed,variable=current_gas)
    slider_gas.grid(column=1,row=0,sticky='we')


    # current gas level label
```

```
current_gas_label = ttk.Label(root,text='Current Gas Level:')
current_gas_label.grid(row=1,columnspan=2,sticky='n',ipadx=10,ipady=10)



# Gas level label (value gets displayed here)
gas_label = ttk.Label(root,text=str(get_current_gas()) +" ppm")
gas_label.grid(row=2,columnspan=2,sticky='n')




def publisher_thread():
    thread = Thread(target=publish_data)
    thread.start()

def publish_data():
    # Exception Handling
    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()  # Connect to IBM Watson IoT Platform


while True:
    gas_level = int(current_gas.get())
    temp_level =random.randint(10,80)
    hum=random.randint(0,10)
    pre=random.randint(0,20)


    data={
'gas_level' : gas_level,
'Temperature':temp_level,
'Humidity':hum,
'Pressure':pre
}


    def myOnPublishCallback():
        if ( gas_level >600):
            print("Gas Level = %s ppm" % gas_level, "Automatic sprinkler turned on")


        print("Published Gas Level = %s ppm" % gas_level, "to IBM Watson")
        print("Published humidity = %s ppm" % hum, "to IBM Watson")
        print("Published temperature = %s ppm" % temp_level, "to IBM Watson")
        print("Published pressure = %s ppm" % pre, "to IBM Watson")
```

```
        success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)

publisher_thread()

root.mainloop() # startup Tkinter GUI

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

CODE:

ibmmain.py - C:\Users\Sunmugam\Desktop\ibmmain.py (3.9.1)

File Edit Format Run Options Window Help

```
import time
import sys
import random
import  wiotp.sdk.device# IBM IoT Watson Platform Module
import ibmiotf.device
import tkinter as tk # Python GUI Package
from tkinter import ttk # Python GUI
import time
from threading import Thread


organization = "vensir"
deviceType = "GASLEAKAGE"
deviceId = "1234"
authMethod = "token"
authToken = "12345678"
# Tkinter root window
root = tk.Tk()
root.geometry('350x300')  # Set size of root window
root.resizable(True, True) # root window non-resizable
root.title('Gas Leakage Monitoring And Alerting System for Industries ')


# Layout Configurations
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=3)


current_gas = tk.DoubleVar()



def get_current_gas(): # function returns current gas level value
    return '{: .2f}'.format(current_gas.get())



def slider_changed(event): # Event Handler for changes in sliders
    print('--------')
    print('Gas Level: {: .2f}'.format(current_gas.get()))
    print('--------')
```

Ln: 17 Col: 0

ENG 08:36 24-11-2022

ibmmain.py - C:\Users\Sunmugam\Desktop\ibmmain.py (3.9.1)

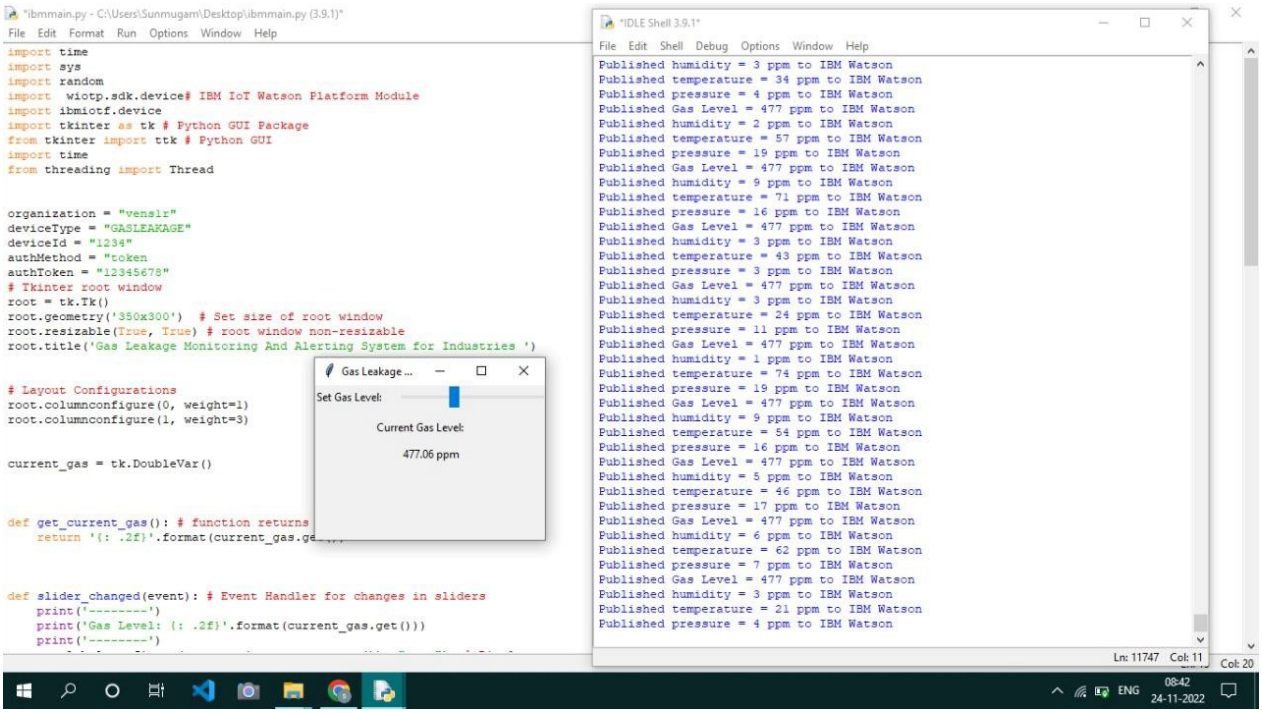
File Edit Format Run Options Window Help

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

    deviceCli.connect()  # Connect to IBM Watson IoT Platform

    while True:
        gas_level = int(current_gas.get())
        temp_level =random.randint(10,80)
        hum=random.randint(0,10)
        pre=random.randint(0,20)

        data={
   'gas_level' : gas_level,
    'Temperature':temp_level,
    'Humidity':hum,
    'Pressure':pre
    }

        def myOnPublishCallback():
            if ( gas_level >600):
                print("Gas Level = %s ppm" % gas_level, "Automatic sprinkler turned on")

            print("Published Gas Level = %s ppm" % gas_level, "to IBM Watson")
            print("Published humidity = %s ppm" % hum, "to IBM Watson")
            print("Published temperature = %s ppm" % temp_level, "to IBM Watson")
            print("Published pressure = %s ppm" % pre, "to IBM Watson")



        success = deviceCli.publishEvent("event", "json", data, qos=0, on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
        time.sleep(1)

publisher_thread()

root.mainloop() # startup Tkinter GUI

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

Ln: 17 Col: 0

ENG 08:40 24-11-2022

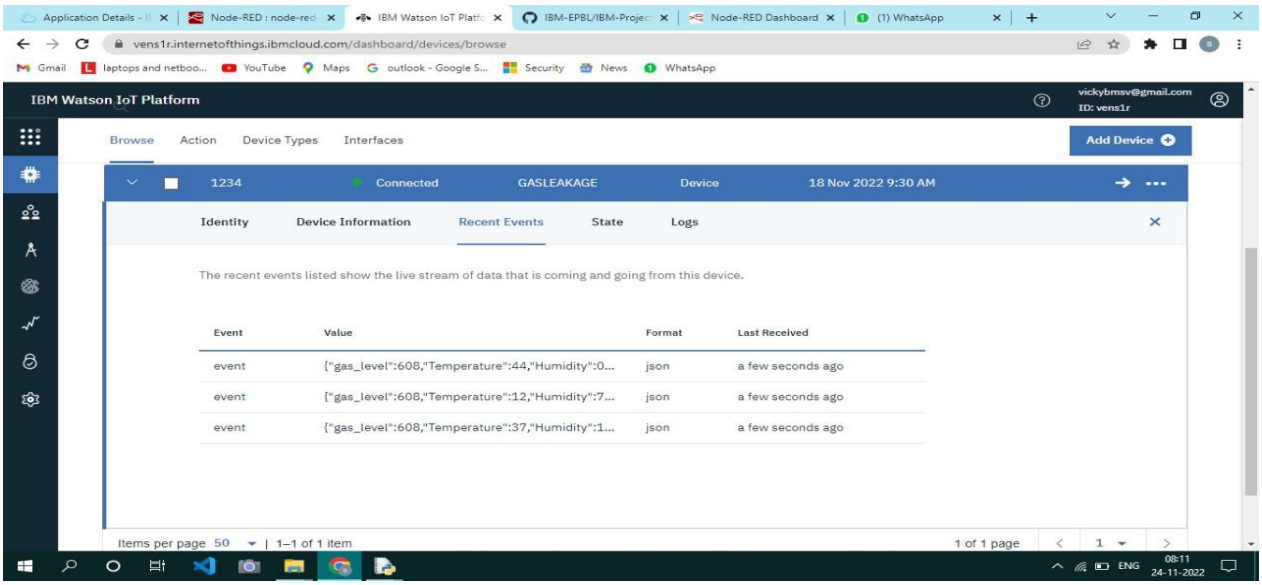
OUTPUT:

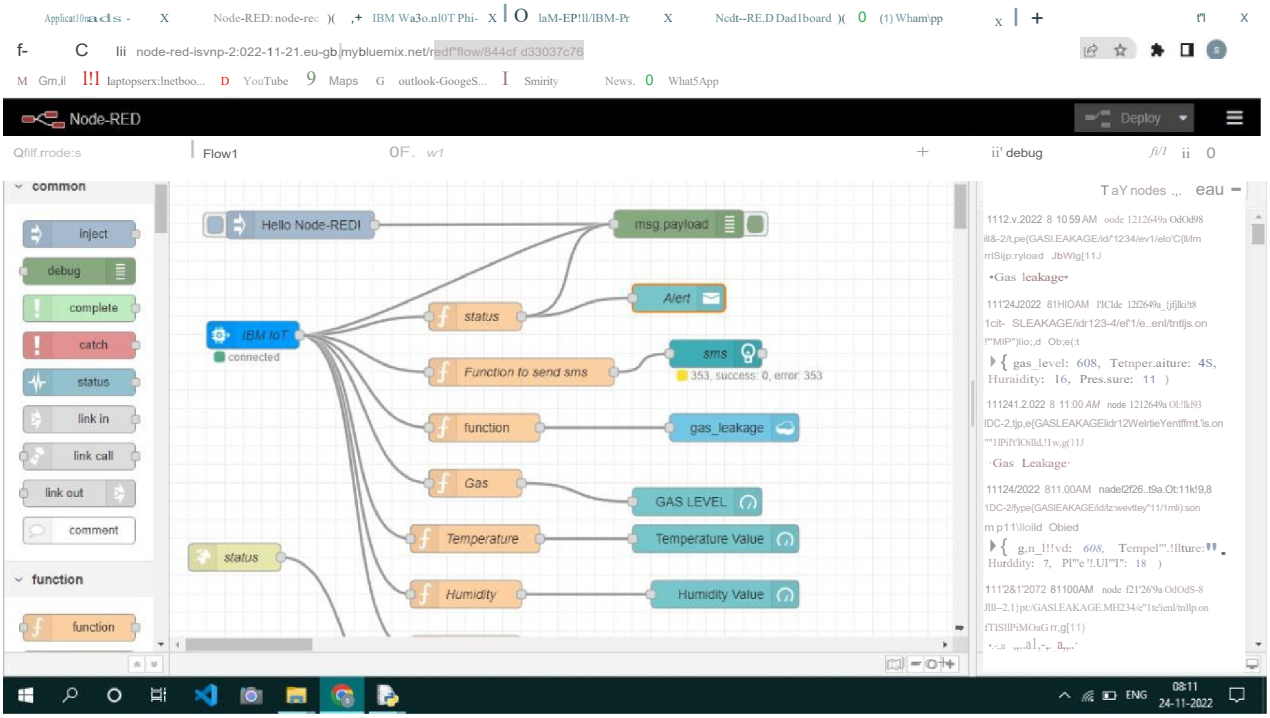

ibmmain.py - C:\Users\Sunmugam\Desktop\ibmmain.py (3.9.1)
File Edit Format Run Options Window Help
import time
import sys
import random
import wiotp.sdk.device# IBM IoT Watson Platform Module
import ibmiotf.device
import tkinter as tk # Python GUI Package
from tkinter import ttk # Python GUI
import time
from threading import Thread
organization = "yenslr"
deviceType = "GASLEAKAGE"
deviceId = "1234"
authMethod = "token
authToken = "12345678"
# Tkinter root window
root = tk.Tk()
root.geometry('350x300') # Set size of root window
root.resizable(True, True) # root window non-resizable
root.title('Gas Leakage Monitoring And Alerting System for Industries ')
# Layout Configurations
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=3)
current_gas = tk.DoubleVar()
def get_current_gas(): # function returns
return '{: .2f}'.format(current_gas.ge
def slider_changed(event): # Event Handler for changes in sliders
print('--------')
print('Gas Level: {: .2f}'.format(current_gas.get()))
print('--------')
Gas Leakage ...
Set Gas Level:
Current Gas Level:
477.06 ppm
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Published humidity = 3 ppm to IBM Watson
Published temperature = 34 ppm to IBM Watson
Published pressure = 4 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 2 ppm to IBM Watson
Published temperature = 57 ppm to IBM Watson
Published pressure = 19 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 9 ppm to IBM Watson
Published temperature = 71 ppm to IBM Watson
Published pressure = 16 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 3 ppm to IBM Watson
Published temperature = 43 ppm to IBM Watson
Published pressure = 3 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 3 ppm to IBM Watson
Published temperature = 24 ppm to IBM Watson
Published pressure = 11 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 1 ppm to IBM Watson
Published temperature = 74 ppm to IBM Watson
Published pressure = 19 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 9 ppm to IBM Watson
Published temperature = 54 ppm to IBM Watson
Published pressure = 16 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 5 ppm to IBM Watson
Published temperature = 46 ppm to IBM Watson
Published pressure = 17 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 6 ppm to IBM Watson
Published temperature = 62 ppm to IBM Watson
Published pressure = 7 ppm to IBM Watson
Published Gas Level = 477 ppm to IBM Watson
Published humidity = 3 ppm to IBM Watson
Published temperature = 21 ppm to IBM Watson
Published pressure = 4 ppm to IBM Watson
Ln: 11747 Col: 11
Col: 20
ENG
08:42
24-11-2022

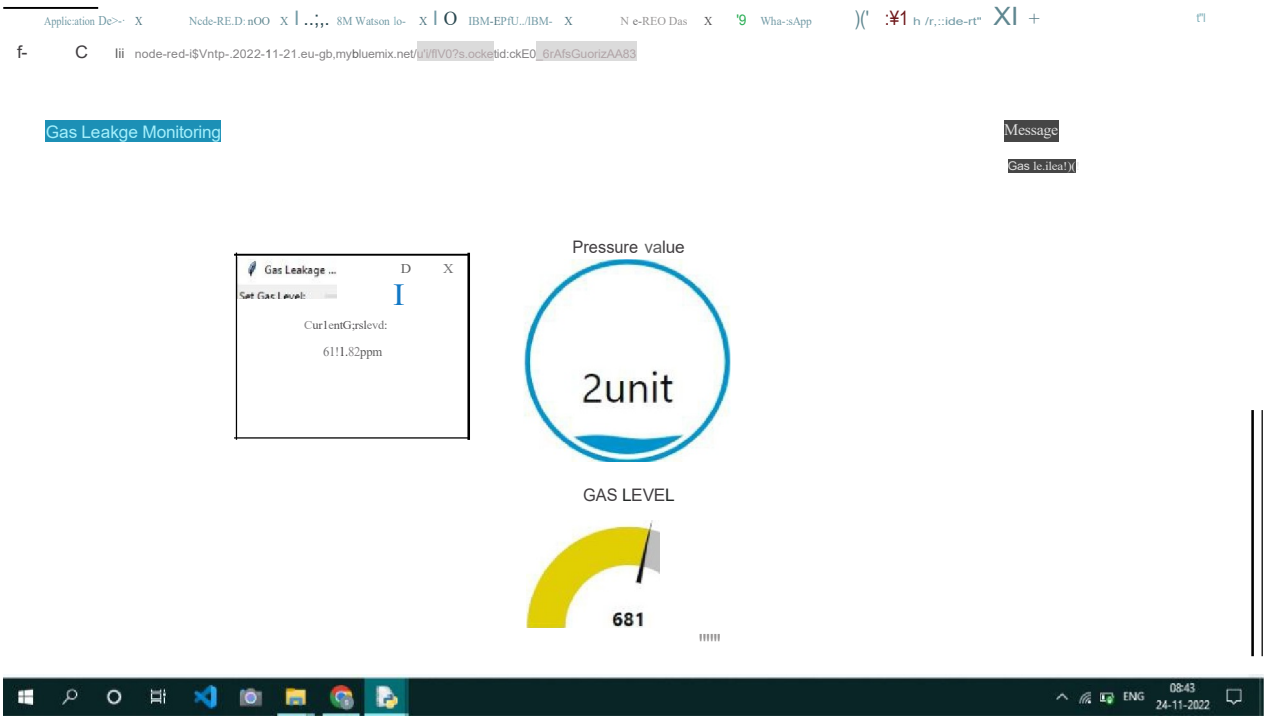
8. Testing:

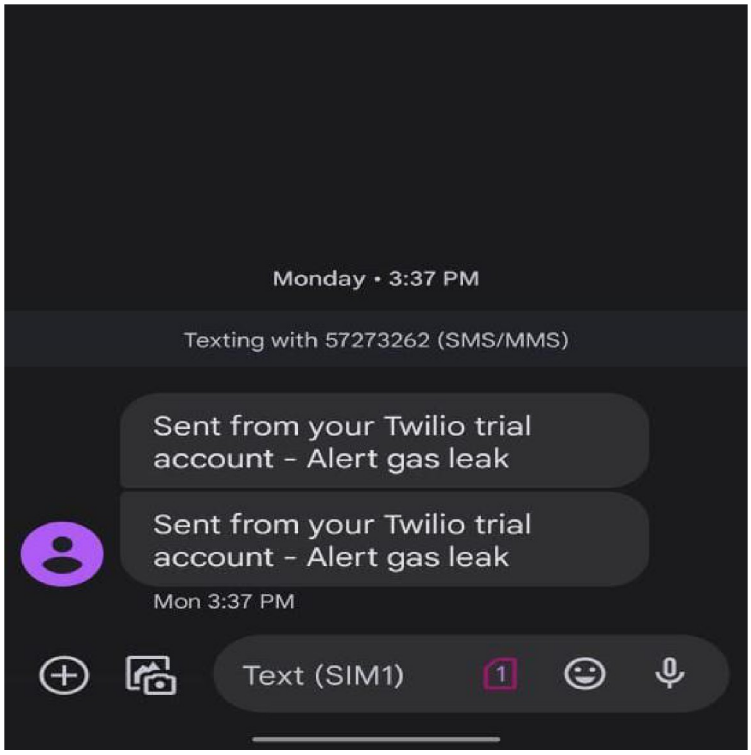
TEST CASE	PRE CONDITION	TEST STEPS	TEST DATA	EXPECTED RESULT
Verify the IBM Watson connection	User should have network connection	1. Run the python code 2. In IBM Watson Connection is enabled	Connected Successfully	Connected Successfully
Run the node red and increase the gas level	Install the node red and python IDLE	1. Enter the url 2. Gas Level will be displayed.	gas_level:660, Temperature: 20, Pressure:10, Humidity:5	User can view the gas level and warning popup message
Send the SMS	User should have Twilio account	1. Enter the twilio account details 2. Enter the message which you want to send	Alert gas leak	User can view the SMS

TESTING SCREENSHOTS:




Node-RED
Deploy
Flow1
debug
common
inject
debug
complete
catch
status
link in
link call
link out
comment
function
function
Hello Node-RED!
msg.payload
Alert
status
IBM IoT
connected
sms
Function to send sms
353, success: 0, error: 353
function
gas_leakage
Gas
GAS LEVEL
Temperature
Temperature Value
status
Humidity
Humidity Value
ENG
08:11
24-11-2022


Gas Leakge Monitoring
Message
Gas le.ilea!)(
Gas Leakage ...
D
X
Set Gas Level:
Cur1entG;rslevd:
61!1.82ppm
Pressure value
2unit
GAS LEVEL
681
ENG
08:43
24-11-2022



Applic:ation - X Ncde-RE.D: nOO x | ..;,. 8M Watson lo- x ı O IBM-EPfU../IBM- x N e-REO Das)('9 Wha-sApp :¥1 h /n::ide--rt" Xl +

f- C lii node-red-i$Vntp-.2022-11-21.eu-gb.mybluemix.net/u"r/flV0?s.ocketid:ckE0_6rAfsGuorizAA83

Gas Leakge Monitoring

Message

V01.1 are safe

I Ga5Leabge D X

Ga5L el: I

CurientG;rslevd:

409.09ppm

Pressure value

3unit

GAS LEVEL

409

ENG 08:45 24-11-2022

9. Result:

The system can be taken as a small attempt in connecting the existing primary gas detection methods to detect gas leakage. The gases are sensed in an area of 1m radius of the sensor. The accuracy of sensors is not up to the mark thus stray gases are also detected which creates an amount of error in the outputs of the sensors, especially in case of methane. Further the availability and storage of toxic gases like hydrogen sulphide also creates problems for testing the assembled hardware. As the system operates outside the pipeline, the complication of system maintenance and material selection of the system in case of corrosive gases is reduced. Thus, the system at this stage can only be use data primary indicator of leakage inside a plant. The system monitors the gas level and alert the concerned authority through SMS.

10. Advantages/Disadvantages:

10.1 Advantages:

1. Get real-time alerts about the gaseous presence in the atmosphere.
2. Prevent fire hazards and explosions.
3. Supervise gas concentration levels.
4. Ensure worker's health.
5. Real-time updates about leakages.
6. Cost-effective installation.
7. Data analytics for improved decisions.
8. Measure oxygen level accuracy.
9. Get immediate gas leak alerts.

10.2 Disadvantages:

1. It requires air or oxygen to work.
2. It gets reacted due to heating of wire.
3. It can be poisoned by lead, chlorine and silicon

11. CONCLUSION:

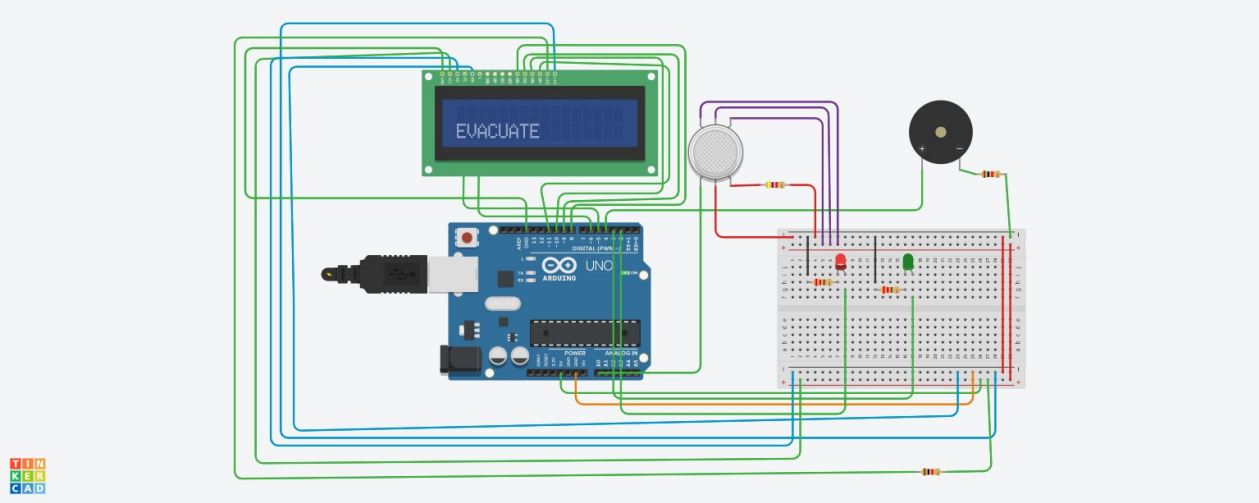
This gas leak detector system contains two features, this includes the SMS Gateway feature for only sending warning information regarding the gas leak to user, and the gas level alert for the warning alert. There is some improvement which can be applied for the future work, such as regarding the SMS Gateway, it need to enhance with feature such as notifying the user whenever the remaining credit balance is insufficient. Therefore, it is recommended to add this kind of features in the future work for better refinement.

12. FUTURE SCOPE:

We propose to build the system using an MQ6 gas detection sensor and interface it with an Arduino Uno microcontroller along with an LCD Display. This system uses the gas sensor to detect any gas leakages. The gas sensor sends out a signal to the microcontroller as soon as it encounters a gas leakage. The microcontroller processes this signal and a message is displayed on the LCD to alert the user.

13. APPENDIX:

13.1 Circuit Diagram:



13.2 Components:

The design of a sensor-based automatic gas leakage detector with an alert and control system. The components are

S.NO	NAME OF THE COMPONENT	QUANTITY
1	Arduino Uno R3	1
2	LCD 16x2	1
3	Piezo	1
4	Gas sensor	1
5	1 k ohm Resistor	1
6	2.3 k ohm Resistor	1
7	4.7 k ohm Resistor	1
8	Red LED	1
9	Green LED	1

13.3 Source Code:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(5,6,8,9,10,11);

int redled = A5; int
greenled = A3; int
buzzer = 4; int sensor =
A0; int sensorThresh =
400; void setup()
{
```

```
pinMode(redled, OUTPUT);
pinMode(greenled,OUTPUT);
pinMode(buzzer,OUTPUT);
pinMode(sensor,INPUT);
Serial.begin(9600);
lcd.begin(16,2);
}
void loop()
{
  int analogValue = analogRead(sensor);
  Serial.println(analogValue);
  if(analogValue>sensorThresh)
  {
    digitalWrite(redled,HIGH);
    digitalWrite(greenled,LOW);
    tone(buzzer,1000,10000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("ALERT");
    Serial.print("ALERT");
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("EVACUATE");
    Serial.println(" -- EVACUATE");
    delay(1000);
}
  else
  {
    digitalWrite(greenled,HIGH);
    digitalWrite(redled,LOW);
    noTone(buzzer);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SAFE");
    Serial.print("SAFE");
```

```
    delay(1000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("ALL CLEAR");
    Serial.println(" -- ALL CLEAR");
    delay(1000);
}
}
```