NALAIYATHIRAN

PROJECT BASED EXPERIENTIAL LEARNING

GAS LEAKAGE MONITORING AND ALERTING SYSTEM

BATCH ID : B11-5A1E

GAS LEAKAGE MONITORING AND ALERTING SYSTEM

PROJECT REPORT

IBM NALAIYA THIRAN

Submitted By

TEAM ID: PNT2022TMID17154

TEAM MEMBERS:

SURENDHAR S	**-**	**6113191043807**
SIVAMURUGAN R	**-**	**6113191041097**
VIJAYARAGAVAN S	**-**	**6113191041116**
RAMANATHAN S	**-**	**6113191043802**

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

In

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

MAHENDRA ENGINEERING COLLEGE (AUTONOMOUS), NOVEMBER-2022

MAHENDRA ENGINEERING COLLEGE

(AUTONOMOUS)

Mahendrapuri, Mallasamudram, Namakkal DT-637503.

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

Certify that this project phase-I report titled "**GAS LEAKAGE MONITORING AND ALERTING SYSTEM**" is the bonafide work of **"SURENDHAR S (6113191014106), SIVAMURUGAN R (6113191041097), VIJAYARAGAVAN S (6113191041116), RAMANATHAN S (6113191043802),"**who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Mr.V.SENTHIL KUMARAN	**Dr. R.Kathirvel**
MENTOR	**EVALUATOR**
Associate Professor,	Associate Professor,
Department of ECE,	Department of ECE,
Mahendra Engineering College,	Mahendra Engineering College,
Mahendrapuri,	Mahendrapuri,
Mallasamudram,	Mallasamudram,
Namakkal Dt – 637 503.	Namakkal Dt – 637 503.

GAS LEAKAGE MONITORING AND ALERTING SYSTEM

PROJECT REPORT

1. INTRODUCTION

1.1 Project Overview

In today's world, safety is of the utmost importance, and certain measures must be taken at both work and home to e ensure it. Working or living in a dangerous environment necessitates specific safety measures, whether the subject is electricity or oil and gas. A type of natural gas known as "Liquified Petroleum Gas" (LPG) is compressed under high pressure and stored in a metal cylinder. LPG is extremely vulnerable to fire and can result in catastrophic damage if left unprotected near any fire source. LPG is primarily utilized for cooking and is more readily available than any other natural gas. Sadly, its widespread use makes gas leakage or even a blast a common occurrence. As a result, a system for detecting and monitoring gas leaks is required. Through a flame sensor, the system will keep an eye on fire and flame. The buzzer begins to ring when a fire is detected. Tests have shown that the system can keep track of the wastage of gas and leaks and notify the user. The performance that was produced showed that it was successful in reducing the amount of domestic gas that was wasted.

1.2 Purpose

Nowadays the home safety detection system plays an important role in the security of people. Since all the people from the home goes to work on a daily bases, it makes it impossible to check on the appliances available at home especially LPG gas cylinder, wired circuits, Etc. In the last three years, there is a tremendous hike in the demand for liquefied petroleum gas (LPG) and natural gas. To meet this access amount of demand for energy and replace oil or coal due to their environmental disadvantage, LPG and natural gas are preferred. These gases are mostly used on a large scale in industry, as heating, home appliances, and motor fuel. 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.

LITERATURE SURVEY

1. Rohan Chandra Pandey, Manish Verma, Lumesh Kumar Sahu 2017.
   Internet of Things (IOT) Based Gas Leakage Monitoring and Alerting System with MQ-2 Sensor. This paper choice of using a real time gas leakage monitoring and Sensing the output levels of gas has been clearly observed by the help of this system.

2. Asmita Varma, Prabhakar S, Kayalvizhi Jayavel 2017.
   Gas Leakage Detection and Smart Alerting and Prediction Using IoT. The proposed gas leakage detector is promising in the Field of safety.

3. Chaitali Bagwe, Vidya Ghadi, Vinayshri Naik, NehaKunte2018.
   IOT Based Gas Leakage Detection System with Database Logging, Prediction and Smart Alerting. The system provides constant monitoring and detection of gas leakage along with storage of data in database for predictions and analysis. The IOT components used helps in making the system much more cost effective in comparison with traditional Gas detector systems.

4. Rohan Chandra Pandey, Manish Verma, Lumesh Kumar Sahu, Saurabh Deshmukh 2018.
   Internet of Things (IoT) Based Gas Leakage Monitoring and Alerting System with Mq-6 Sensor. A discussion on how the aims and objectives are met is presented. An overall conclusion IOT based toxic gas detector is it has become more efficient, more applicable to today’s applications and smarter.

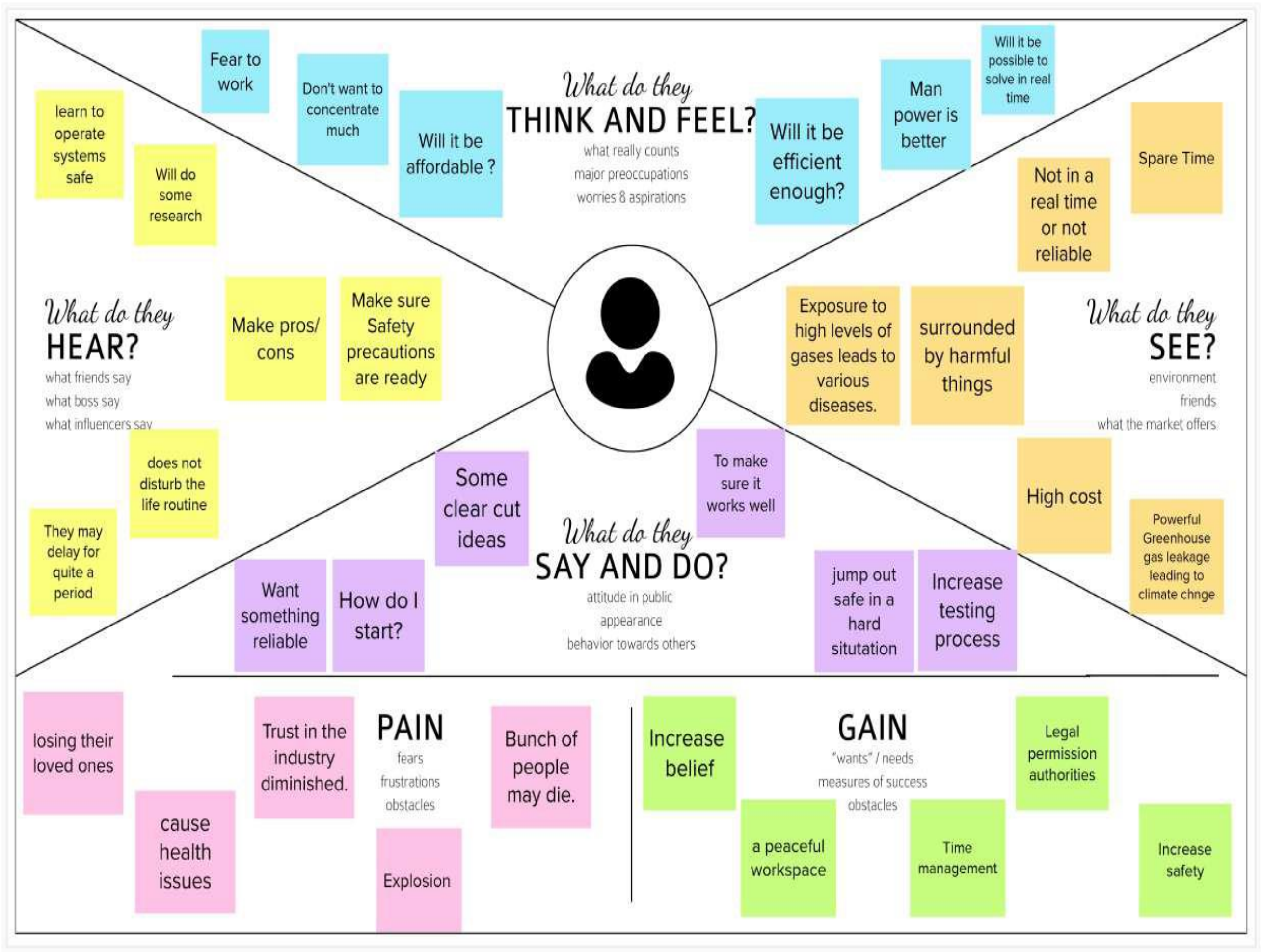
5. Shital Imade, Priyanka Rajmanes, Aishwarya Gavali 2018.
   Gas Leakage Detection and Smart Alerting System Using IoT. In this paper we use IOT technology for enhancing the existing safety standards. While making this prototype has been to bring are volution in the field of safety against the leakage of harmful and toxic gases.

2.1 Problem Statement Definition

Gas leakage leads to various accidents resulting into both financial loss as well as human injuries. In human’s daily life, environment gives the most significant impact to their health issues. The risk of fires, explosion, suffocation, all are based on their physical properties such flammability, toxicity etc. The number of deaths due to the explosion of gas cylinders has been increasing in recent years. The reason for such explosion is due to substandard cylinders, old valves, worn out regulators and lack of awareness using gas cylinders add to risks. Inspections by oil companies found that many LPG consumers are unaware of safety checks of gas cylinders. In other to minimize or eliminate the hazard of gas leakage there is a need for a system to detect and alert on such incidence leading to the development of this project.

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas



3.2 Proposed Solution

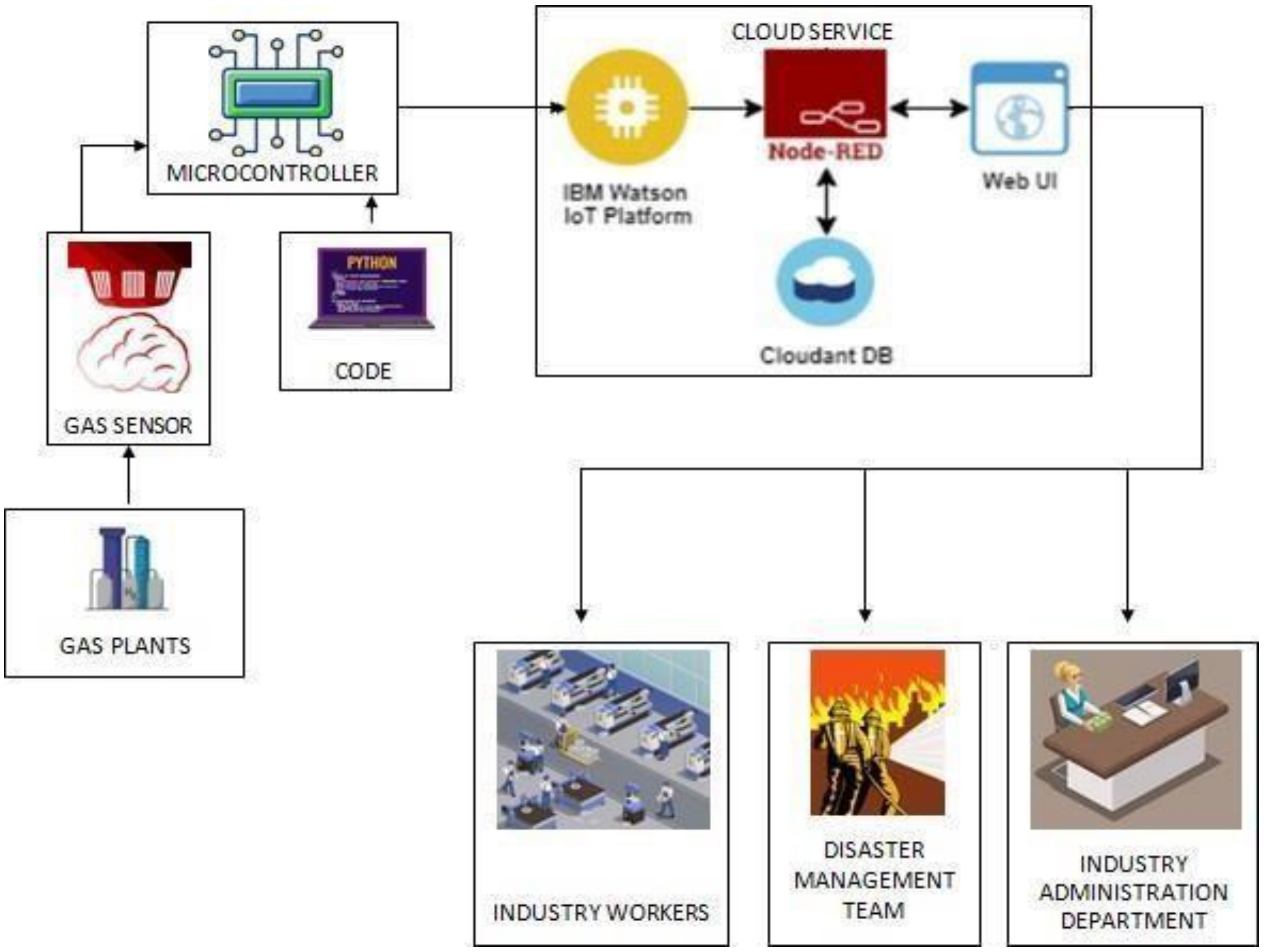
S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Workers who are engaged with a busy industries packed with gas either harmful or harmless needs a way to monitor their gas pipelines continuously and detect early if there is any leakage of gas in their surroundings so that they can work efficiently on major crises rather than worrying about monitoring or leakage of gas, this will indeed reduce the manpower of that industry and create a peaceful environment.
2.	Idea / Solution description	Workers who are engaged with a busy industries packed with gas either harmful or harmless needs a way to monitor their gas pipelines continuously and detect early if there is any leakage of gas in their surroundings so that they can work efficiently on major crises rather than worrying about monitoring or leakage of gas, this will indeed reduce the manpower of that industry and create a peaceful environment.

3.	Novelty / Uniqueness	Even though there are many existing solutions for this problem they failed to satisfy the needs of customer. Some of the solutions are only detecting some particular gases where some others failed to alert the main department and other solutions are with some delays. Our solution not only notify the industry person but also notify the fire fighters so that can take control over the situation and our solution will alert the workers even there is a small leak of gases.
4.	Social Impact / Customer Satisfaction	Our solution will be very helpful for the workers and the society which is associated or located Near by the industries. Our solution will prevent great disasters like Bhopal Gas Tragedy so that so many lives can be saved. Through this project the workers mental pressure will be reduced so that they can concentrate on other works or by relaxing them.
5.	Business Model (Revenue Model)	The main target of our solution is Industries so we have planned to visit industries and explain them about the benefits of our products. So that they can aware of the importance of this solution and use it.
6.	Scalability of the Solution	Our solution can be integrated for further future use because the solution we have provided will be lay on the basic or initial stage of any upgraded version.

3.3 Solution Architecture:

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- There are numerous available projects and solution for this problem. Among them using gas sensor near the map and alerting the industry workers is the working and best solutions in the market.
- In our solution we will place a sensor near the gas plants which will always monitor the gas plants which will constantly send data to the microcontroller.
- Incase if gas leak occurs the value measured by the gas sensor will exceed the threshold value and the controller will be coded to send a message to admin department, workers of that industry and Disaster Management Department of that particular area.
- This message alert will be done through the cloud services.



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 Online Payment for the service.
FR-2	User Access	Access the details using web browser. Access the details using mobile application.
FR-3	User alert	Gets alert as an SMS message. Gets alert alarm in the working area.

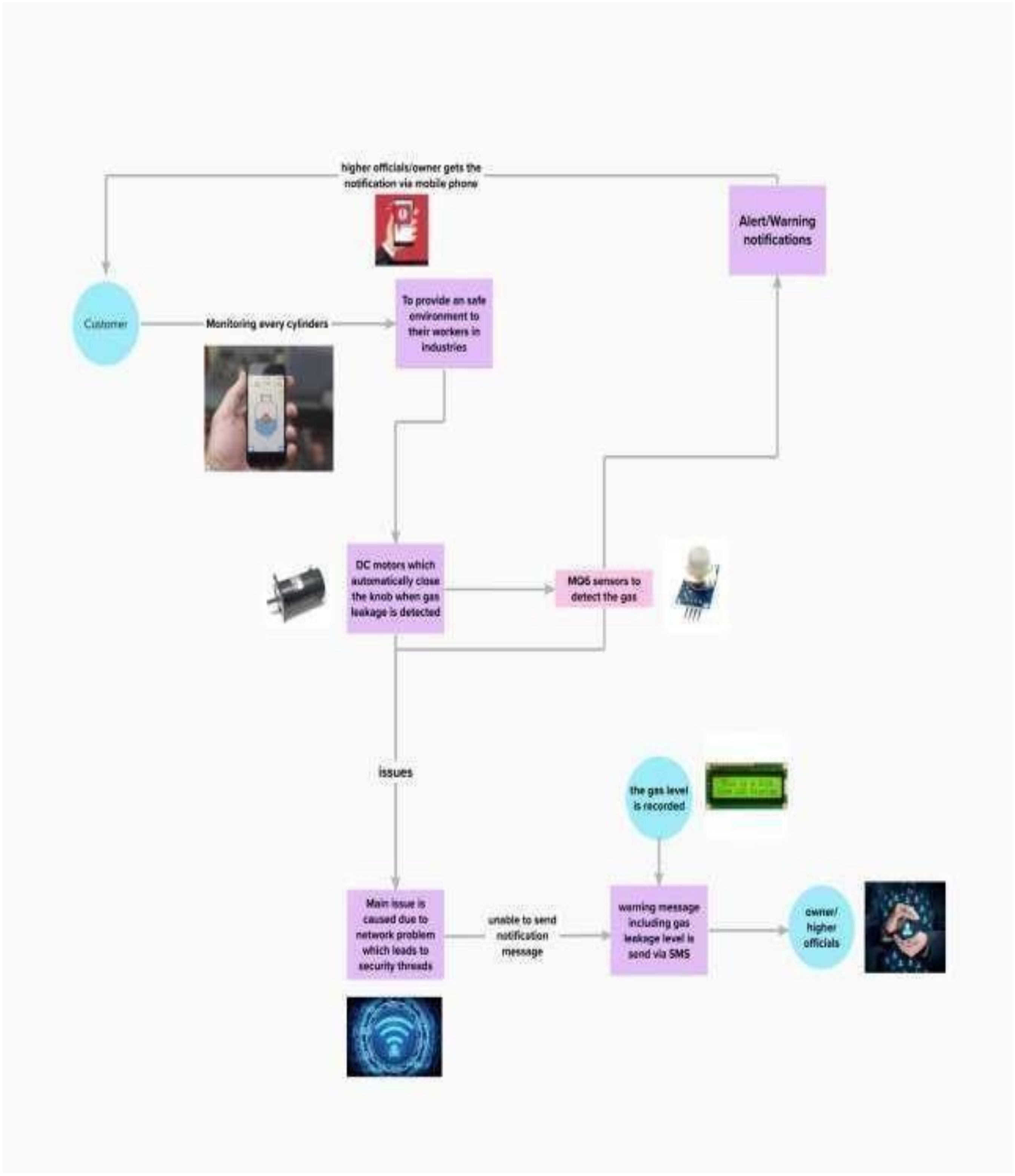
4.2 Non-Functional Requirements.

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

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The device must be usable by the customer anywhere
NFR-2	Security	Data from the sensors are stored securely and away from other data.
NFR-3	Reliability	Data can be retrieved anytime and no data is discarded without customer knowledge.
NFR-4	Performance	No performance delay in case of large number ofdata or more parameters
NFR-5	Availability	The device doesn’t fail even under harsh conditions.Device continues to send parameters, even after analert situation.
NFR-6	Scalability	Device must be capable of measuring conditionseven in a larger industry

PROJECT DESIGN

5.1 Data Flow Diagram.


higher officials/owner gets the notification via mobile phone
Alert/Warning notifications
Customer
Monitoring every cylinders
To provide an safe environment to their workers in industries
DC motors which automatically close the knob when gas leakage is detected
MQ6 sensors to detect the gas
issues
the gas level is recorded
Main issue is caused due to network problem which leads to security threads
unable to send notification message
warning message including gas leakage level is send via SMS
owner/ higher officials

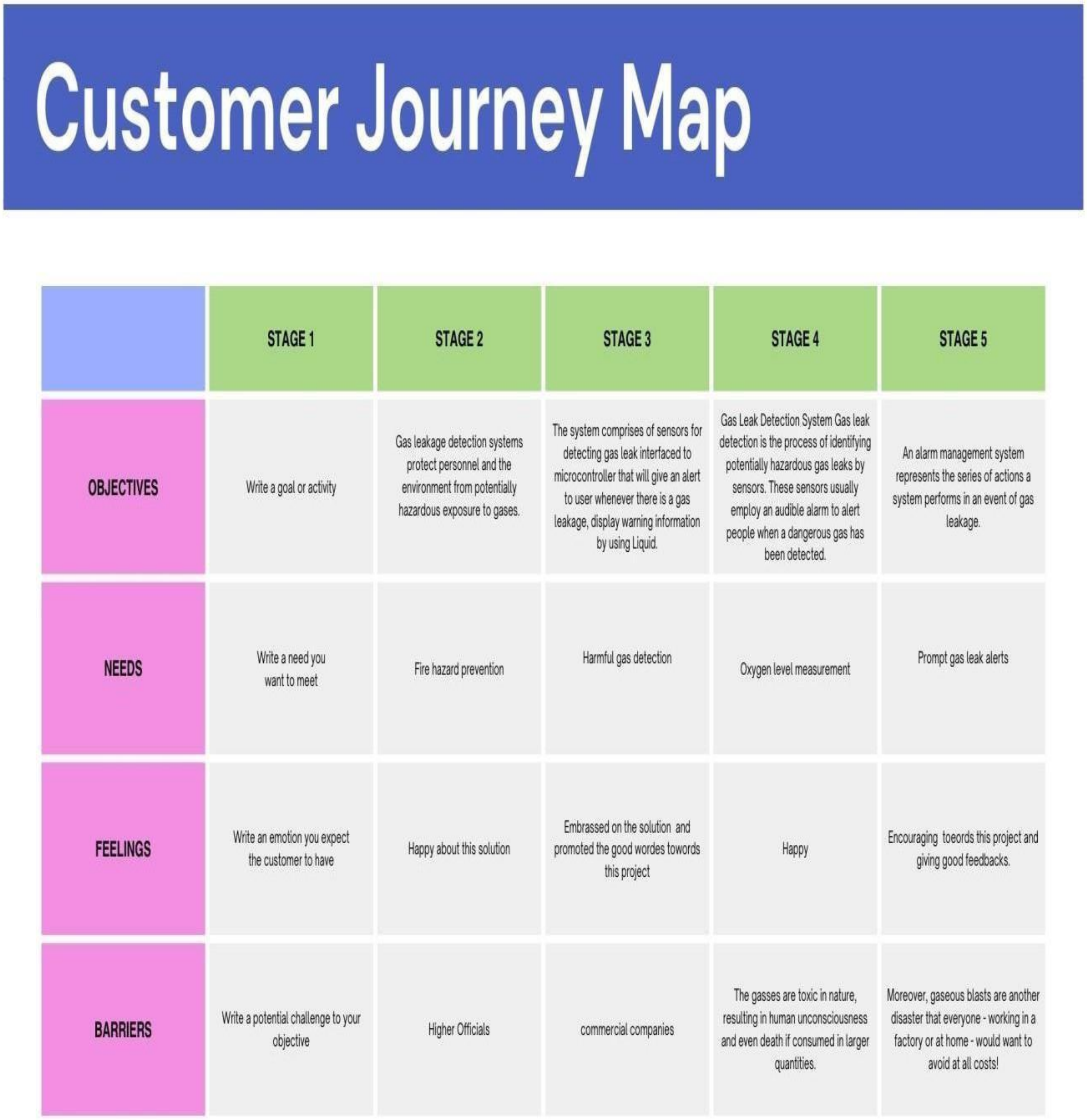
User Stories: Use the below template to list all the userstories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer(family member/industry owner)	Registration	USN-1	As a user ,I can register for the device in the owners mobile application by entering my email and password	I can access my account / dashboard	High	Sprint-1
Customer(higher authority)	confirmation	USN-2	As a user I will receive confirmation message via email and once I receivedI can install the device in the ownersplace	I can receive confirmation email & click confirm	High	Sprint-1
Customer (fire service 101)	Safety measure register	USN-3	As a register I can register the application in owner/family members mobile phone	I can register & access the dashboard with Facebook Login	Low	Sprint-2
Customer (mobile user)	Mobile application	USN-4	As a user I can register bymobile application	I can register for gasdetection device with owners mobile number and the alertmessagewill be send by SMS	Medium	Sprint-1
Customer (credential)	Login	USN-5	As a user I can log into the device by entering email & password in the owner's mobile application	Mail address and passwords are default	High	Sprint-1
Customer (Web user)	Notification	USN-7	As a user when there is a critical situation regarding gas explosion the alert notification willbe received through GSM module	Alert message is sent to owners mobile as an SMS	High	Sprint-1
Customer care Executive	Network Connectivity	USN-8	When there is a gas leakage is detected in the surrounding	Sensor detect the leakage and notifies the owner via message	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Administration	Accessing	USN-9	When there is an issue in accessing the device	Admin/Device operator's advice should be undertaken	High	Sprint-1

5.2 Technology Architecture.

5.3 Customer Journey Map



Customer Journey Map

	STAGE 1	STAGE 2	STAGE 3	STAGE 4	STAGE 5
OBJECTIVES	Write a goal or activity	Gas leakage detection systems protect personnel and the environment from potentially hazardous exposure to gases.	The system comprises of sensors for detecting gas leak interfaced to microcontroller that will give an alert to user whenever there is a gas leakage, display warning information by using Liquid.	Gas Leak Detection System Gas leak detection is the process of identifying potentially hazardous gas leaks by sensors. These sensors usually employ an audible alarm to alert people when a dangerous gas has been detected.	An alarm management system represents the series of actions a system performs in an event of gas leakage.
NEEDS	Write a need you want to meet	Fire hazard prevention	Harmful gas detection	Oxygen level measurement	Prompt gas leak alerts
FEELINGS	Write an emotion you expect the customer to have	Happy about this solution	Embrassed on the solution and promoted the good wordes towords this project	Happy	Encouraging toeords this project and giving good feedbacks.
BARRIERS	Write a potential challenge to your objective	Higher Officials	commercial companies	The gasses are toxic in nature, resulting in human unconsciousness and even death if consumed in larger quantities.	Moreover, gaseous blasts are another disaster that everyone - working in a factory or at home - would want to avoid at all costs!

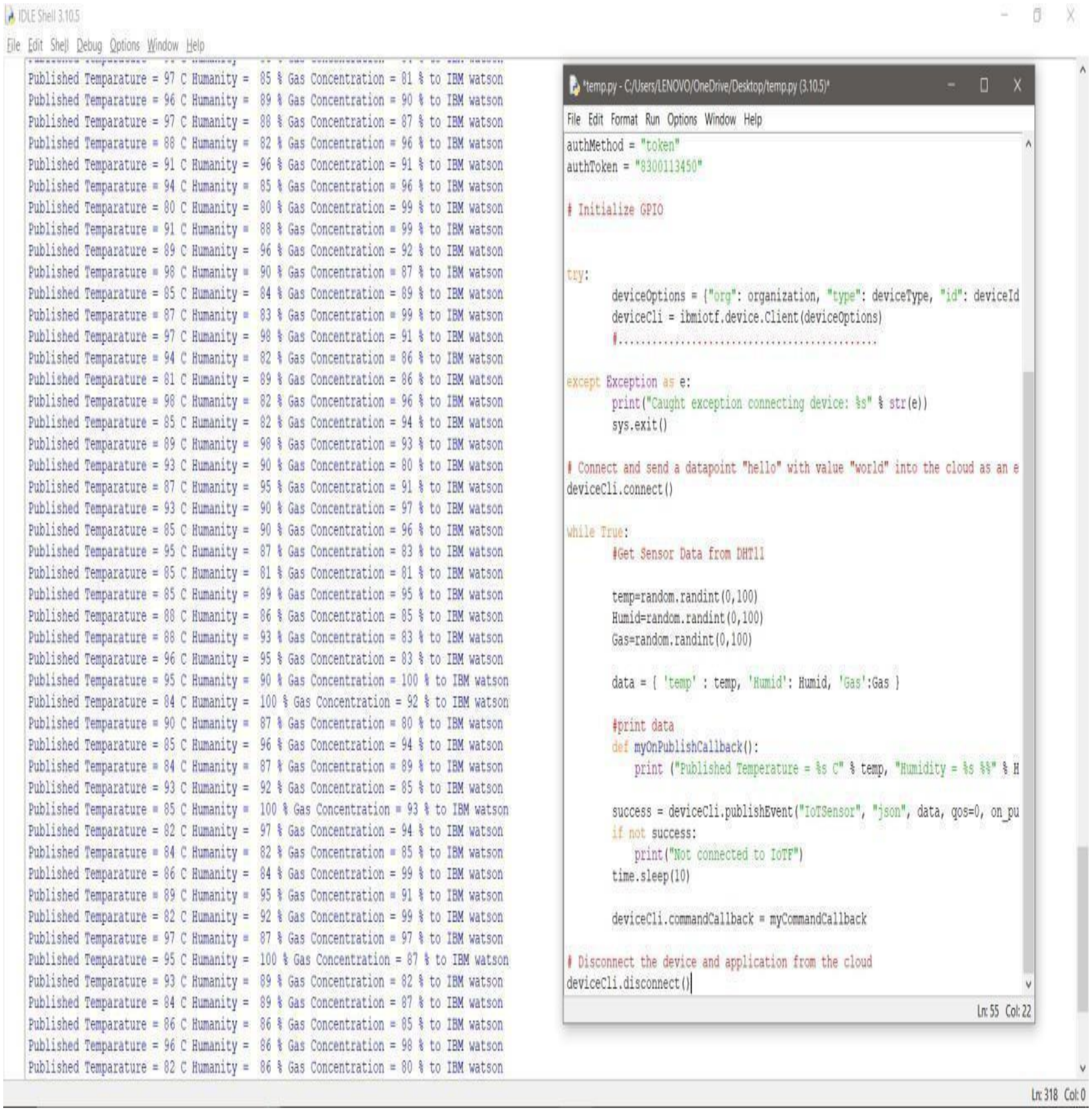
Sprint – 1

Python Code:

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

#Provide your IBM Watson Device Credentials
organization = "lcft5g"
deviceType = "Final"
deviceId = "Hello"
authMethod = "token"
authToken = "8300113450"

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
    anevent of type "greeting" 10 times
  deviceCli.connect()
```

```
while True:
        #Get Sensor Data from DHT11
        temp=random.randint(0,100)
        Humid=random.randint(0,100)
        Gas=random.randint(0,100)

        data = { 'temp' : temp, 'Humid': Humid,’Gas’:gas }
        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" %Humid, “Gas
Concentration = %s”%Gas"to IBM Watson")

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
            print("Not connected to IoTF")
            time.sleep(10)

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

Output:


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
temp.py - C:/Users/LENOVO/OneDrive/Desktop/temp.py (3.10.5)
File Edit Format Run Options Window Help
Ln: 55 Col: 22
Ln: 318 Col: 0

```
Published Temparature = 97 C Humanity =  85 % Gas Concentration = 81 % to IBM watson
Published Temparature = 96 C Humanity =  89 % Gas Concentration = 90 % to IBM watson
Published Temparature = 97 C Humanity =  88 % Gas Concentration = 87 % to IBM watson
Published Temparature = 88 C Humanity =  82 % Gas Concentration = 96 % to IBM watson
Published Temparature = 91 C Humanity =  96 % Gas Concentration = 91 % to IBM watson
Published Temparature = 94 C Humanity =  85 % Gas Concentration = 96 % to IBM watson
Published Temparature = 80 C Humanity =  80 % Gas Concentration = 99 % to IBM watson
Published Temparature = 91 C Humanity =  88 % Gas Concentration = 99 % to IBM watson
Published Temparature = 89 C Humanity =  96 % Gas Concentration = 92 % to IBM watson
Published Temparature = 98 C Humanity =  90 % Gas Concentration = 87 % to IBM watson
Published Temparature = 85 C Humanity =  84 % Gas Concentration = 89 % to IBM watson
Published Temparature = 87 C Humanity =  83 % Gas Concentration = 99 % to IBM watson
Published Temparature = 97 C Humanity =  98 % Gas Concentration = 91 % to IBM watson
Published Temparature = 94 C Humanity =  82 % Gas Concentration = 86 % to IBM watson
Published Temparature = 81 C Humanity =  89 % Gas Concentration = 86 % to IBM watson
Published Temparature = 98 C Humanity =  82 % Gas Concentration = 96 % to IBM watson
Published Temparature = 85 C Humanity =  82 % Gas Concentration = 94 % to IBM watson
Published Temparature = 89 C Humanity =  98 % Gas Concentration = 93 % to IBM watson
Published Temparature = 93 C Humanity =  90 % Gas Concentration = 80 % to IBM watson
Published Temparature = 87 C Humanity =  95 % Gas Concentration = 91 % to IBM watson
Published Temparature = 93 C Humanity =  90 % Gas Concentration = 97 % to IBM watson
Published Temparature = 85 C Humanity =  90 % Gas Concentration = 96 % to IBM watson
Published Temparature = 95 C Humanity =  87 % Gas Concentration = 83 % to IBM watson
Published Temparature = 85 C Humanity =  81 % Gas Concentration = 81 % to IBM watson
Published Temparature = 85 C Humanity =  89 % Gas Concentration = 95 % to IBM watson
Published Temparature = 88 C Humanity =  86 % Gas Concentration = 85 % to IBM watson
Published Temparature = 88 C Humanity =  93 % Gas Concentration = 83 % to IBM watson
Published Temparature = 96 C Humanity =  95 % Gas Concentration = 83 % to IBM watson
Published Temparature = 95 C Humanity =  90 % Gas Concentration = 100 % to IBM watson
Published Temparature = 84 C Humanity =  100 % Gas Concentration = 92 % to IBM watson
Published Temparature = 90 C Humanity =  87 % Gas Concentration = 80 % to IBM watson
Published Temparature = 85 C Humanity =  96 % Gas Concentration = 94 % to IBM watson
Published Temparature = 84 C Humanity =  87 % Gas Concentration = 89 % to IBM watson
Published Temparature = 93 C Humanity =  92 % Gas Concentration = 85 % to IBM watson
Published Temparature = 85 C Humanity =  100 % Gas Concentration = 93 % to IBM watson
Published Temparature = 82 C Humanity =  97 % Gas Concentration = 94 % to IBM watson
Published Temparature = 84 C Humanity =  82 % Gas Concentration = 85 % to IBM watson
Published Temparature = 86 C Humanity =  84 % Gas Concentration = 99 % to IBM watson
Published Temparature = 89 C Humanity =  95 % Gas Concentration = 91 % to IBM watson
Published Temparature = 82 C Humanity =  92 % Gas Concentration = 99 % to IBM watson
Published Temparature = 97 C Humanity =  87 % Gas Concentration = 97 % to IBM watson
Published Temparature = 95 C Humanity =  100 % Gas Concentration = 87 % to IBM watson
Published Temparature = 93 C Humanity =  89 % Gas Concentration = 82 % to IBM watson
Published Temparature = 84 C Humanity =  89 % Gas Concentration = 87 % to IBM watson
Published Temparature = 86 C Humanity =  86 % Gas Concentration = 85 % to IBM watson
Published Temparature = 96 C Humanity =  86 % Gas Concentration = 98 % to IBM watson
Published Temparature = 82 C Humanity =  86 % Gas Concentration = 80 % to IBM watson
```

```python
authMethod = "token"
authToken = "8300113450"

# Initialize GPIO


try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId
        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 e
deviceCli.connect()

while True:
        #Get Sensor Data from DHT11

        temp=random.randint(0,100)
        Humid=random.randint(0,100)
        Gas=random.randint(0,100)

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

        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % H

        success = deviceCli.publishEvent("IoTSensor", "json", data, qos=0, on_pu
        if not success:
            print("Not connected to IoTF")
        time.sleep(10)

        deviceCli.commandCallback = myCommandCallback

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

Sprint – 2

Task:

Sensed data is brought to Node-RED and displayed in dashboard.

Steps:

1. IBM IoT node is used to gather sensor data.
   a. Necessary API key is provided to establish connection.
2. Using functions namely Temperature, Humidity and Gas the data is obtained independently and displayed in dashboard.
3. Dashboard Nodes are used to display the sensed data to the userin a portal.

Sour code:

Temperature:

```
msg.payload = msg.payload.Temp;

return msg;
```

Humidity:

```
msg.payload = msg.payload.Hum;
return msg;
```

Concentration of Gas:

```
msg.payload = msg.payload.gas;
return msg;
```

SPRINT-3

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(6, 7, 8, 9, 10, 11);
float gasPin = A0;
float
gasLevel;
int ledPin = 2;
int buttonPin = 3;
int buzzPin = 4;
int buttonState;
int fan = 5;


void  setup(){
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin,INPUT);
  pinMode(gasPin,INPUT);
  pinMode(fan,OUTPUT);
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0,0);
  lcd.print(" Welcome");
```

```
 lcd.setCursor(0,2);
 lcd.print("Youtube");
 delay(500);
 lcd.clear();
}

void loop(){
 // Read the value from gas sensor and button
 gasLevel = analogRead(gasPin);
 buttonState = digitalRead(buttonPin);

 // call the function for gas detection and button work
 gasDetected(gasLevel);
 buzzer(gasLevel);
 exhaustFanOn(buttonState);
}

// Gas Leakage Detection & Automatic Alarm and Fan ON
void gasDetected(float gasLevel){
 if(gasLevel >= 300){
  digitalWrite(buzzPin,HIGH);
  digitalWrite(ledPin,HIGH);
  digitalWrite(fan,HIGH);
```

```
   lcd.setCursor(0,0);
  lcd.print("GAS:");
  lcd.print(gasLevel);
  lcd.setCursor(0,2);
  lcd.print("FANON");
  delay(1000);
  lcd.clear();
 }else{
  digitalWrite(ledPin,LOW);
  digitalWrite(buzzPin,LOW);
  digitalWrite(fan,LOW);
  lcd.setCursor(0,0);
  lcd.print("GAS:");
  lcd.print(gasLevel);
  lcd.setCursor(0,2);
  lcd.print("FAN OFF");
  delay(1000);
  lcd.clear();
 }
}
//BUZZER
void buzzer(float  gasLevel){
if(gasLevel>=300)
```

```
 {
 for(int i=0; i<=30; i=i+10)
 {
 tone(4,i);
 delay(400);
 noTone(4);
 delay(400);
 }
 }
}
// Manually Exhaust FAN ON
void exhaustFanOn(int buttonState){
 if(buttonState == HIGH){
 digitalWrite(fan,HIGH);
 lcd.setCursor(0,0);
 lcd.print("Button State:");
 lcd.print(buttonState);
 lcd.setCursor(0,2);
  lcd.print("FAN ON");delay(10000); lcd.clear();
 }
}
```

SPRINT 4

```
#include <LiquidCrystal.h>
 LiquidCrystal lcd(5,6,8,9,10,11);
int redled = 2;
 int greenled = 3;
 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.print(analogValue);
 if(analogValue>sensorThresh)
{
digitalWrite(redled,HIGH);
 digitalWrite(greenled,LOW);
tone(buzzer,1000,10000);
 lcd.clear();
 lcd.setCursor(0,1);
 lcd.print("ALERT");
 delay(1000);
lcd.clear();
lcd.setCursor(0,1);
lcd.print("EVACUATE");
delay(1000);
}
else
{
```

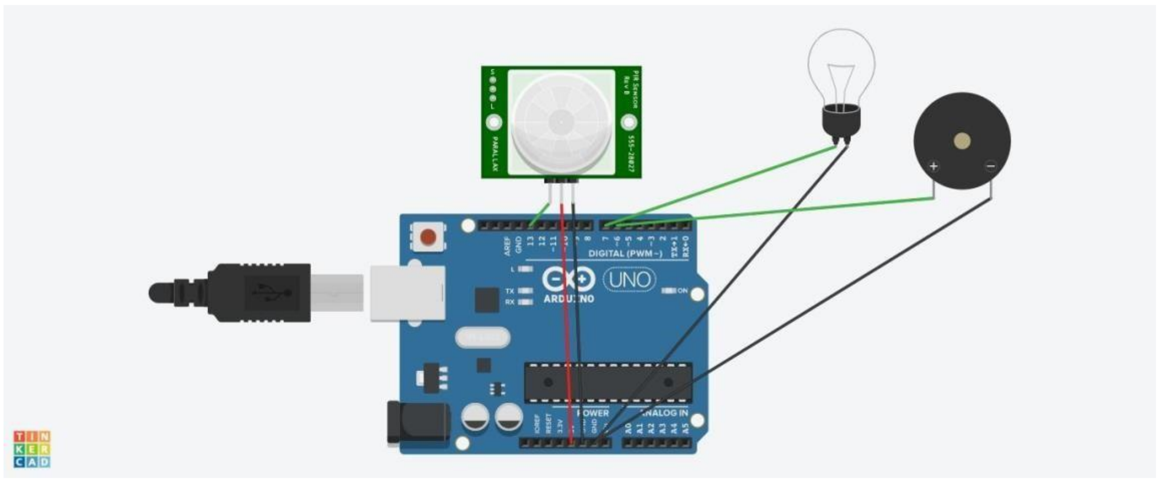
```
digitalWrite(greenled,HIGH);

digitalWrite(redled,LOW);
noTone(buzzer);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("SAFE");
delay(1000);
lcd .clear();
lcd.setCursor(0,1);
lcd.print("ALL CLEAR");
delay(1000);
}
  }
```

}

ASSIGNMENTS

1. **Build a SMART HOME in Thinkercad with 2 Sensors, an Led, Buzzer and submit it.**

SMART HOME AUTOMATION



```
Void setup()
 {
 pinMode(13, INPUT);
 pinMode(7, OUTPUT);
 pinMode(6, OUTPUT);
}

 void loop()  {

 digitalWrite(7,digitalRead(13));

digitalWrite(6,digitalRead(13));

 }
```

2. **Build a python code, Assume u get temperature and humidity values (generated with a random function to a variable) and write a condition to detect an alarm in case of high temperature continuously.**

Program:

```
import random
while(True):
    a=random.randint(10,120)
    b=random.randint(10,120)
          if(a>35 and b>60):
       print(" high temperature and humidity of:",a,b,"% alarm is on")
elif(a<35 and b<60):
      print("Normal temperature and humidity of:",a,b,"% alarm is off")
      break
```

3. **Write python code for blinking LED and Traffic lights for Raspberry pi.**

Solution:

```
import RPi.GPIO as GPIO
import time
import signal
import sys

# Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(9, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)

# Turn off all lights when its end
def allLightsOff(signal, frame):
        GPIO.output(9, False)
        GPIO.output(10, False)
        GPIO.output(11, False)
        GPIO.cleanup()
        sys.exit(0)

signal.signal(signal.SIGINT, allLightsOff)

# Loop for led light
while True:
        # RED LIGHT
        GPIO.output(9, True)
        time.sleep(3)

        # RED AND YELLOW
        GPIO.output(10, True)
        time.sleep(1)
```

```
# GREEN LIGHT
GPIO.output(9, False)
GPIO.output(10, False)
GPIO.output(11, True)
time.sleep(5)

# YELLOW LIGHT
GPIO.output(11, False)
GPIO.output(10, True)
time.sleep(2)

#YELLOW OFF (red start from first loop)
GPIO.output(10, False)
```

4. **Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cms send "alert" to ibm cloud and display in device recent events.**

CODE:

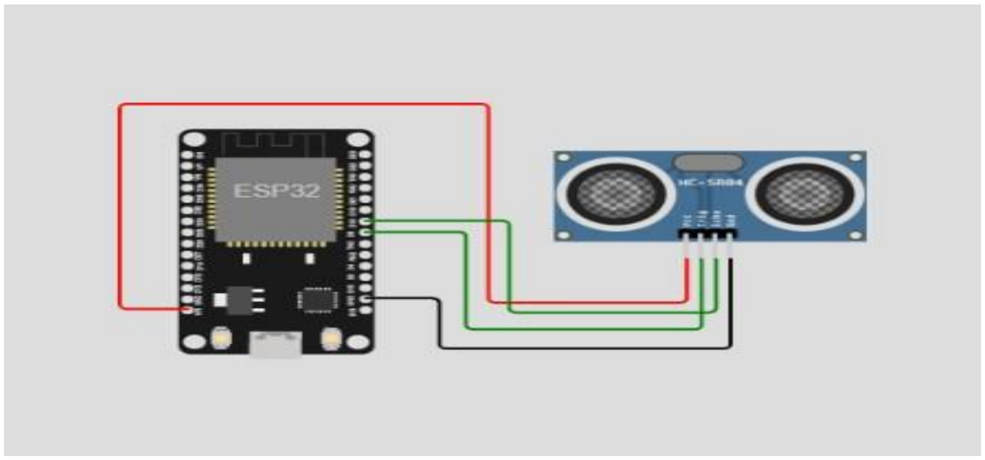
```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-------credentials of IBM Accounts------
#define ORG "Ashfaq1824"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "12345"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup()
{
Serial.begin(115200);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
wificonnect();
mqttconnect();
}
void loop()
{
```

```
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = duration * SOUND_SPEED/2;
Serial.print("Distance (cm): ");
Serial.println(distance);
if(distance<100)
{
Serial.println("ALERT!!");
delay(1000);
PublishData(distance);
delay(1000);
if (!client.loop())
{
mqttconnect();
}
}
delay(1000);
}
void PublishData(float dist)
{
mqttconnect();
String payload = "{\"Distance\":";
payload += dist;
payload += ",\"ALERT!!\":""\"Distance less than 100cms\"";
payload += "}";
Serial.print("Sending payload: ");
Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");
}
else {
Serial.println("Publish failed");
}
}
void mqttconnect()
{
```

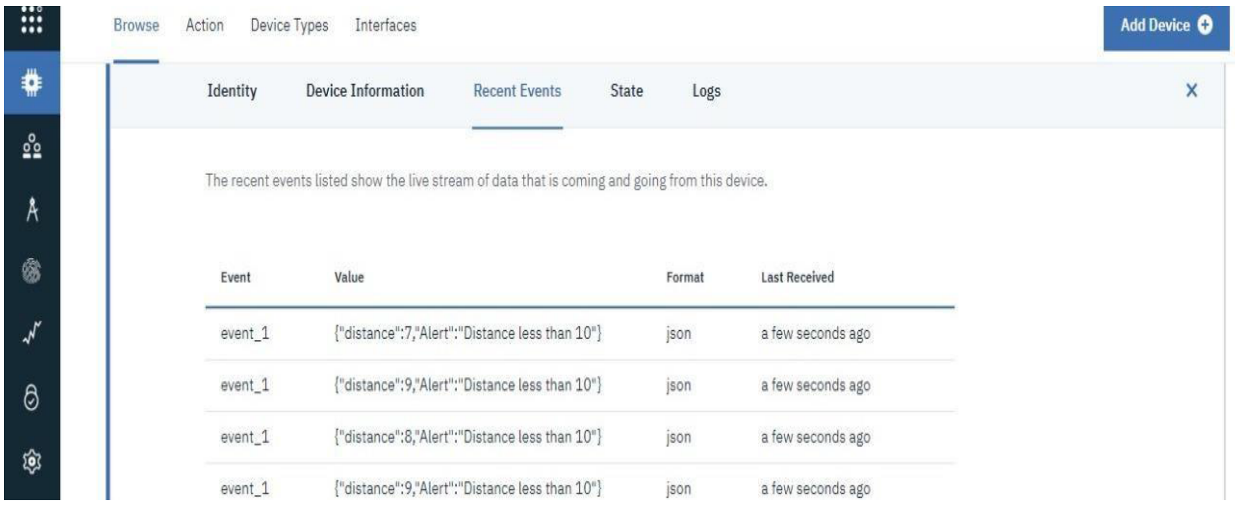
```
if (!client.connected()) {
Serial.print("Reconnecting client to ");
Serial.println(server);

while (!!!client.connect(clientId, authMethod, token)) {
Serial.print(".");
delay(500);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect()
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void initManagedDevice()
{
if (client.subscribe(subscribetopic)) {
Serial.println((subscribetopic));
Serial.println("subscribe to cmd OK");
}
else {
Serial.println("subscribe to cmd FAILED");
}
}
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
{
Serial.print("callback invoked for topic: ");
Serial.println(subscribetopic);
```

```
for (int i = 0; i < payloadLength; i++) {
//Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
Serial.println("data: "+ data3);
data3="";
}
```

SCHEMATIC/CIRCUIT DIAGRAM:



IBM CLOUD OUTPUT:



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

Event	Value	Format	Last Received
event_1	{"distance":7,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":8,"Alert":"Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"Distance less than 10"}	json	a few seconds ago