

# **GAS LEAKAGE MONITORING AND ALERTING SYSTEM**

## **A PROJECT REPORT**

*Submitted by*

**HARIHARAN S B (717819F216)**

**MANOJ M (717819F226)**

**NANDHAKUMAARAN S (717819F229)**

**NAREESH R (717819F230)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**INFORMATION TECHNOLOGY**

**KARPAGAM COLLEGE OF ENGINEERING,  
COIMBATORE**

**ANNA UNIVERSITY:: CHENNAI 600 025**

**NOVEMBER 2022**

- 1. INTRODUCTION**
  - 1.1 Project Overview
  - 1.2 Purpose
- 2. LITERATURE SURVEY**
  - 2.1 Existing problem
  - 2.2 References
  - 2.3 Problem Statement Definition
- 3. IDEATION & PROPOSED SOLUTION**
  - 3.1 Empathy Map Canvas
  - 3.2 Ideation & Brainstorming
  - 3.3 Proposed Solution
  - 3.4 Problem Solution fit
- 4. REQUIREMENT ANALYSIS**
  - 4.1 Functional requirement
  - 4.2 Non-Functional requirements
- 5. PROJECT DESIGN**
  - 5.1 Data Flow Diagrams
  - 5.2 Solution & Technical Architecture
  - 5.3 User Stories
- 6. PROJECT PLANNING & SCHEDULING**
  - 6.1 Sprint Planning & Estimation
  - 6.2 Sprint Delivery Schedule
  - 6.3 Reports from JIRA
- 7. CODING & SOLUTIONING (Explain the features added in the project along with code)**
  - 7.1 Feature 1
  - 7.2 Feature 2
  - 7.3 Database Schema (if Applicable)
- 8. TESTING**
  - 8.1 Test Cases
  - 8.2 User Acceptance Testing
- 9. RESULTS**
  - 9.1 Performance Metrics
- 10. ADVANTAGES & DISADVANTAGES**
- 11. CONCLUSION**
- 12. FUTURE SCOPE**
- 13. APPENDIX**
  - Source Code
  - GitHub & Project Demo Link

# **1 INTRODUCTION**

## **1.1 PROJECT OVERVIEW**

Most of the fire-breakouts in industries are due to gas leaks. These cause dreadful damage to the equipment, human life leading to injuries, deaths, and environment. Currently available leakage detectors warn the people around using on-site alarms. So, this project proposes a leakage detector which sends the warning to the concerned people through SMS. This detector senses the presence of harmful gases particularly, LPG, Methane and Benzene. LPG and Methane gases catch fire easily resulting in blasts. Benzene is carcinogen effecting the health of workers, if inhaled in higher concentrations. Hence, detection of these gases is essential. This low cost project includes MQ6, MQ4 and MQ135 gas sensors which detect LPG, Methane and Benzene gas leaks respectively and uses ESP-32 as a Wi-Fi module. The concentration levels of the above mentioned gases are uploaded in the UBIDOTS cloud and the login details are included in the alert message so that the user can check, if needed. The prototype of the proposed system generates a sound alert using buzzer on detection of a dangerous leakage and sends an SMS to the concerned person using IFTTT web service. Different color LEDS are used to specify the gas leaked for example, RED LED indicates the presence of LPG.

## **1.2 PURPOSE**

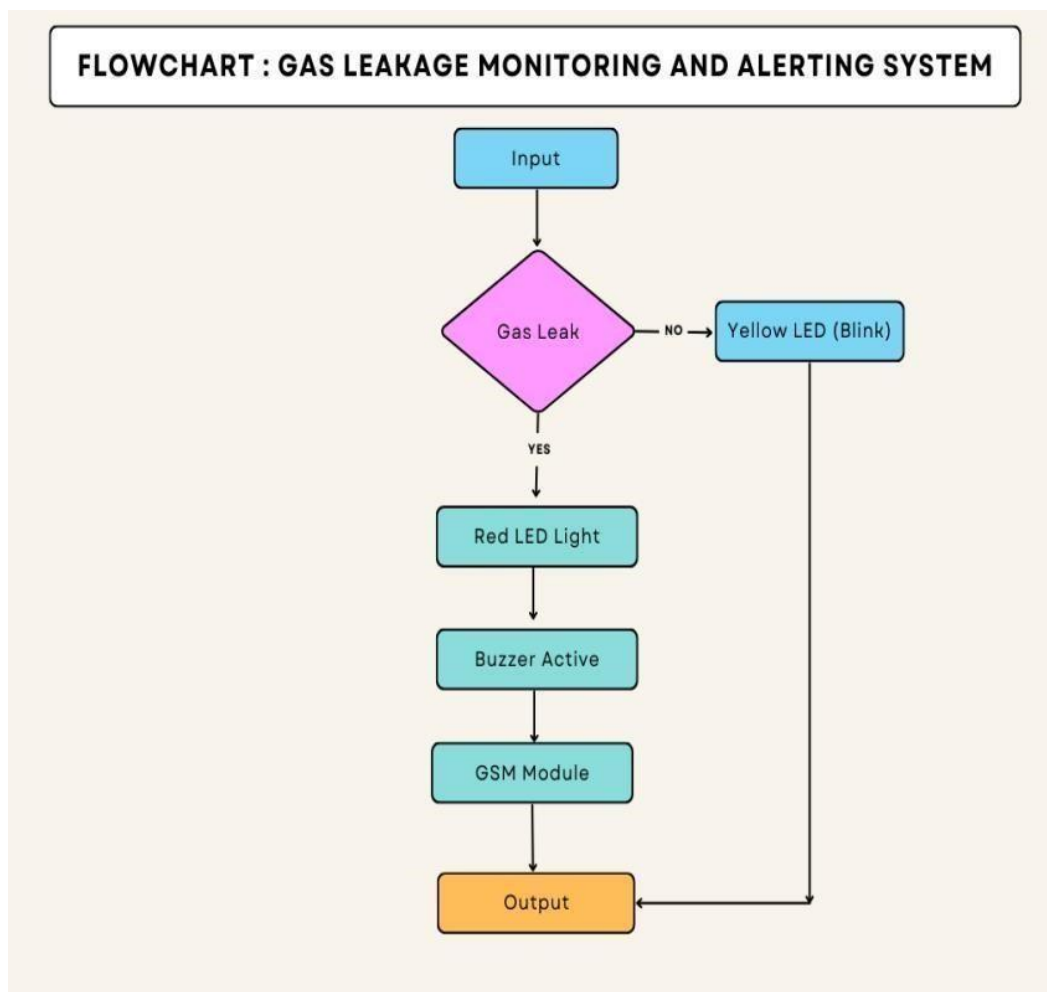
The aim of this project is to detect combustible, flammable and toxic gases and oxygen depletion. This type of device is widely used in industry and can be used in locations such as on oil rigs to monitor manufacturing processes and emerging technologies such as photovoltaic. They may be used in firefighting.

Gas leak detection is the process of identifying potentially hazardous gas leaks by sensors. Additionally, a visual identification can be done using a thermal camera. These sensors usually employ an audible alarm to alert people when a dangerous gas has been detected.

# **2 LITERATURE SURVEY**

## 2.1 EXISTING PROBLEM

Arduino UNO (Atmega-328) is the main unit of the system which performs the following tasks. A signal conditioning of the Arduino UNO is done by output signal of the sensor, provided input to Arduino. The detection results displayed on LCD. Indicates the people of danger in work place, factory, home. Buzzer activity with beep(siren) sound is made. Also send alert SMS to the in charge of the plant number is saved in SIM card by using GSM modem. The SMS received depends upon the leak of gas in the detection area of the sensor.



## 2.2 REFERENCES

- [1] Shrivastava, A., Prabhaker, R., Kumar, R., & Verma, R. GSM based gas leakage detection system. International

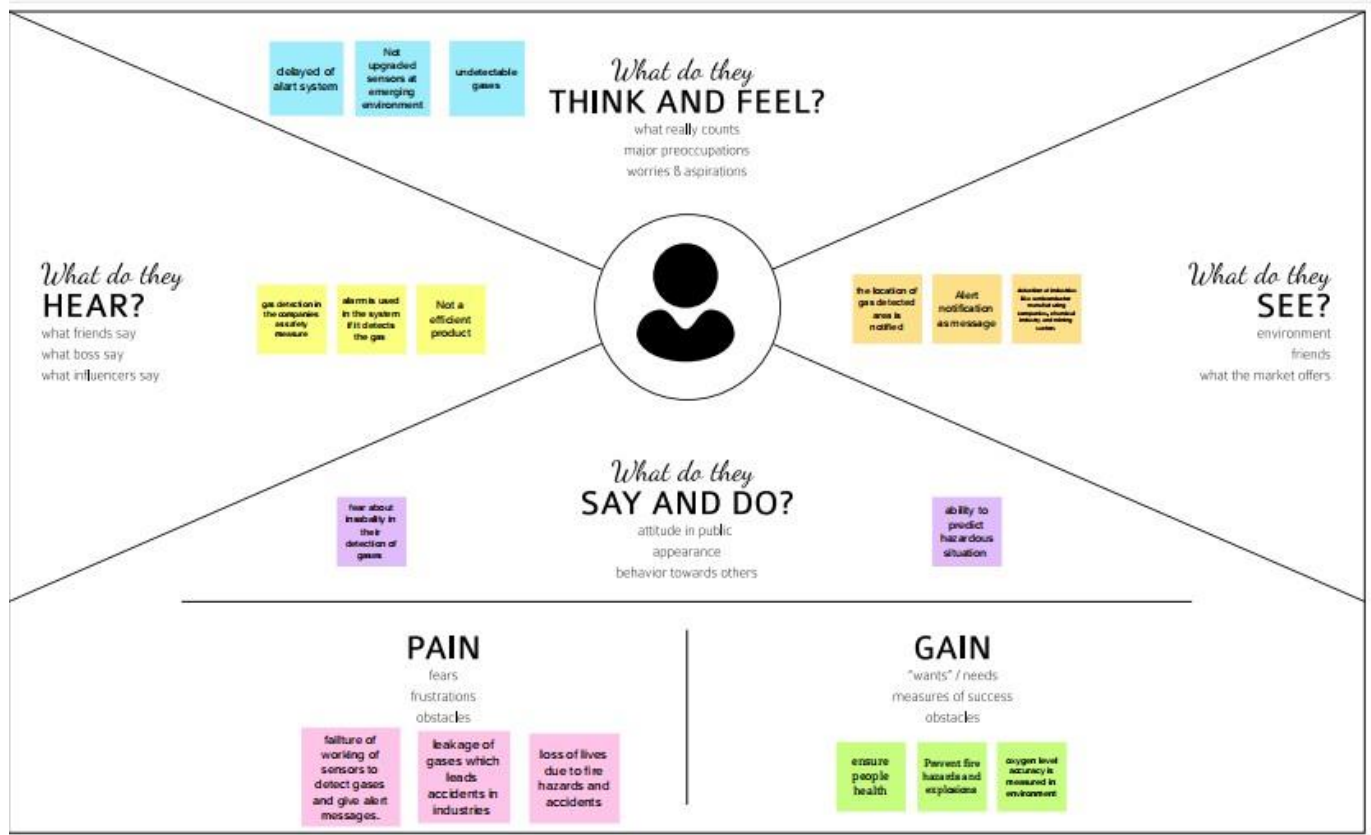
- Journal of Emerging Trends in Electrical and Electronics (IJETEE-ISSN: 2320-9569), 2013; 3(2):42-45.
- [2] Hema, L. K., Murugan, D., & Chitra, M. WSN based Smart system for detection of LPG and Combustible gases. In National Conf. on Architecture, Software systems and Green computing-2013.
- [3] Ramya, V., & Palaniappan, B. Embedded system for Hazardous Gas detection and Alerting. International Journal of Distributed and Parallel Systems (IJDPS), 2012; 3(3):287-300.
- [4] Priya, P. D., & Rao, C. T. Hazardous Gas Pipeline Leakage Detection Based on Wireless Technology. International Journal of Professional Engineering Studies, India, 2014; 2(1).
- [5] Jero, S. E., & Ganesh, A. B. 2011, March. PIC18LF4620 based customizable wireless sensor node to detect hazardous gas pipeline leakage. In 2011 International Conference on Emerging Trends in Electrical and Computer Technology (pp. 563-566). IEEE.
- [6] Anusha, O., & Rajendra prasad, C. H. Experimental investigation on road safety system at crossings. International Journal of Engineering and Advanced Technology, 2019; 8(2):214–218.
- [7] Pravalika, V., & Rajendra Prasad, C. Internet of things based home monitoring and device control using Esp32. International Journal of Recent Technology and Engineering, 2019; 8(1 Special Issue 4):58–62.
- [8] Sanjay Kumar, S., Ramchandrarao, P., & Rajendra Prasad, C. Internet of things based pollution tracking and alerting system. International Journal of Innovative Technology and Exploring Engineering, 2019; 8(8):2242–2245
- [9] Deepak, N., Rajendra Prasad, C., & Sanjay Kumar, S. Patient health monitoring using IOT. International Journal of Innovative Technology and Exploring Engineering, 2018; 8(2):454–457. <https://doi.org/10.4018/978-1-5225-8021-8.ch002>
- [10] Ramu, M., & Prasad, C. R. Cost effective atomization of Indian agricultural system using 8051 microcontrollers. International journal of advanced research in computer and communication engineering, 2013; 2(7):2563-2566.

## **2.3 PROBLEM STATEMENT DEFINITION**

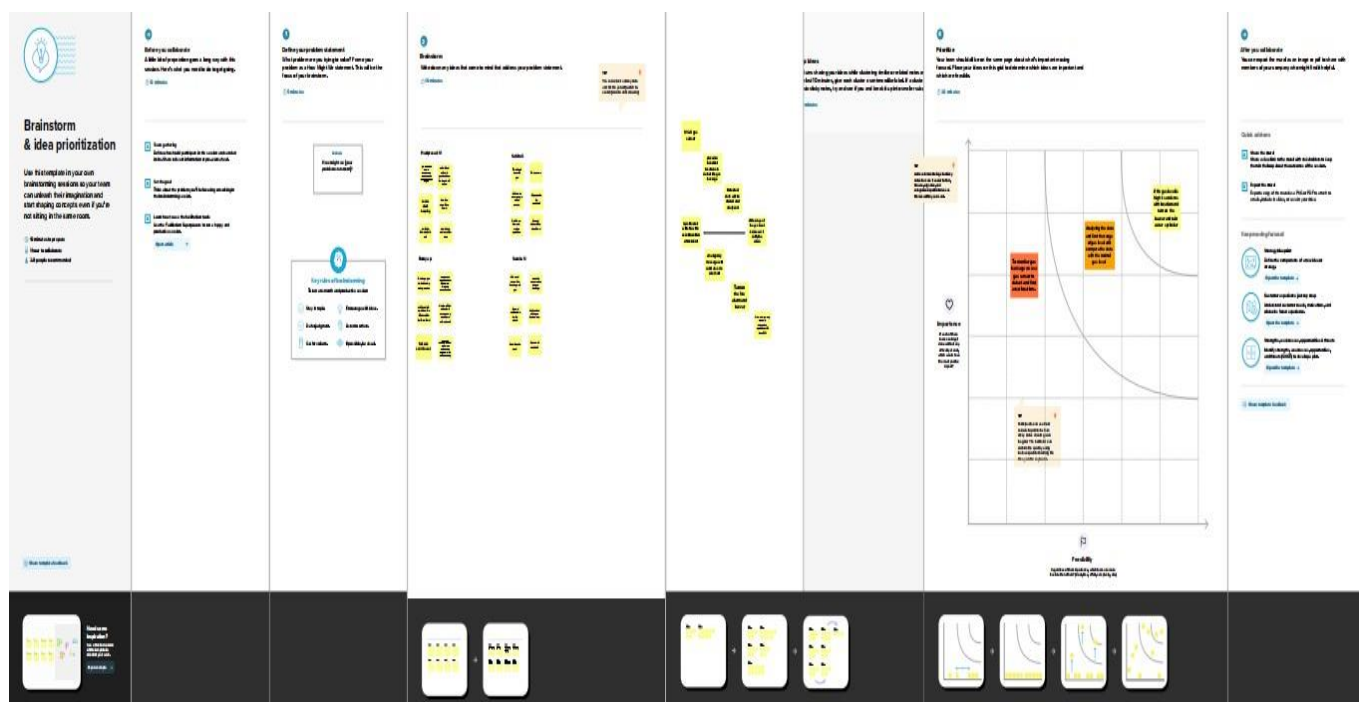
The issue from the environment and the air quality in industrial area is increased the alertness and responsibility regarding the public and workers health. Gas leakage leads to various accidents resulting into both financial loss as well as human injuries. In human's daily life, environment gives the most significant impact to their health issues. The risk of fires, explosion, suffocation, all are on their physical properties such flammability, toxicity etc. The number of based deaths due to the explosion of gas has been increased recently. Thus, a gas detector is invented to ease human on detecting the presence of those dangerous gases within an area to prevent disaster happen. The sensor-enabled solution helps prevent the high risk of gas explosions and affecting any casualties within and outside the premises. The gas sensors help to detect the concentration of the gases present in the atmosphere to avoid hazardous consequences like fire breakouts for humans and workers in industries.

## **3 IDEATION AND PROPOSED SOLUTION**

### **3.1 EMPATHY MAP CANVAS**



## 3.2 IDEATION AND BRAINSTORMING



### 3.3 PROPOSED SOLUTION

#### Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	In industries gas leakage leads to major accidents and property damage which may affect both the employs as well as the industry.
2.	Idea / Solution description	<p>This project helps the industries in watching the emission of harmful gases</p> <p>In many areas, the gas sensors are integrated to observe the gas outflow</p> <p>If in any space gas outflow is detected the admins are notified together with the placement</p> <p>In the net application, admins will read the sensing element parameters.</p>
3.	Novelty / Uniqueness	In this project with the help of senor we will find the location, where the gas is leaked and turn on the buzzer,
4.	Social Impact / Customer Satisfaction	Inhaling leaked gas in an indoor space, such as your home can result in a lack of oxygen in the air and lead to hypoxia. That can, in turn, lead to severe headaches, fatigue, decreased vision, short breaths, and even loss of consciousness. It can be prevented by our project.



5.	Business Model (Revenue Model)	In industries, home, laboratory, hospital there is possibility of gas leakages. So this model will become an undeniable product or source for almost all companies . To prevent the gas leakage.
6.	Scalability of the Solution	As the product is offered with subscription service , further development in both software and hardware can be made.

### 3.4 PROBLEM SOLUTION FIT

CUSTOMER SEGMENT	CUSTOMER CONSTRAINTS	AVAILABLE SOLUTION
Industries and home	<ul style="list-style-type: none"> <li>If the power supply fails, the system won't work.</li> <li>Failure of device/components may have dire consequences ,fatal accidents can occur.</li> </ul>	<ul style="list-style-type: none"> <li>In the method periodic check done by manually and partial sensing methodology is used.</li> <li>It raises alarm whenever gas leaked or fire is detected at any place in a factory</li> </ul>
JOBS-TO-BE-DONE / PROBLEMS	PROBLEM ROOT CAUSE	BEHAVIOUR
<ul style="list-style-type: none"> <li>Needs to sense the gas leakage by sensors</li> <li>Used to measure the precise value of gas leakage</li> <li>Needs to send the SMS to the concerned person</li> </ul>	<ul style="list-style-type: none"> <li>Faulty piping and improper use of gas furnace, or appliance.</li> <li>The leakage of gas may happen due to the human error and false chemical reaction.</li> <li>Lack of service done in the gas valve.</li> </ul>	Focus their major concentrate on precautions/ safety measure to make avoid unwanted losses.
TRIGGERS	YOUR SOLUTION	CHANNELS of BEHAVIOUR
<p>Sudden loss of life of workers and damage to industrial property due to gas leakage</p> <p><b>Emotions</b>  BEFORE : insecure, instability  AFTER : worry, Sorrow ,pain</p>	By using gas , smoke and temperature sensor it would accurately sense the working environment and concentration of above gases are uploaded in the database and monitored. For additionally security auto air ventilation and automatic door locking mechanism is placed .	<p>ONLINE :  Information will be conveyed to concerned person by SMS to avoid risky situations.</p> <p>OFFLINE :  If a concerned person does not aware about the gas leakage it is difficult to handle.</p>

## 4 REQUIREMENT ANALYSIS

## 4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

<b>FR No.</b>	<b>Functional Requirement (Epic)</b>	<b>Sub Requirement (Story / Sub-Task)</b>
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.

<b>NFR No.</b>	<b>Non-Functional Requirement</b>	<b>Description</b>
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 of data or more parameters
NFR-5	Availability	The device doesn't fail even under harsh conditions. Device continues to send parameters, even after an alert situation.
NFR-6	Scalability	Device must be capable of measuring conditions even in a larger industry

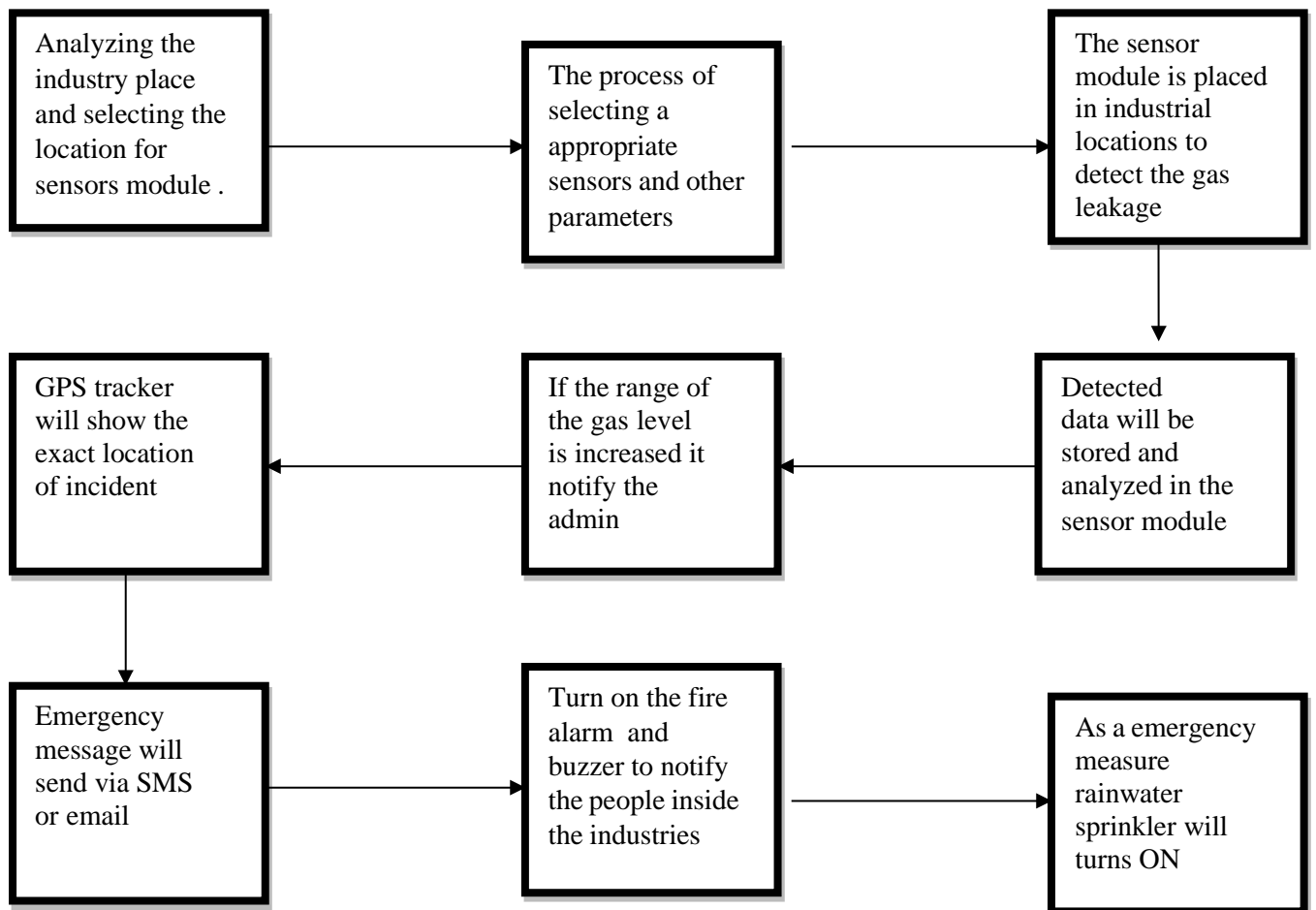
## 5 PROJECT DESIGN

### 5.1 DATA FLOW DIAGRAMS



## 5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Gas Leakage Monitoring & Alerting System for Industries has all the features to prevent the gas leakage and notify the authority. This model is built with help of sensors. In this model we use python programming language



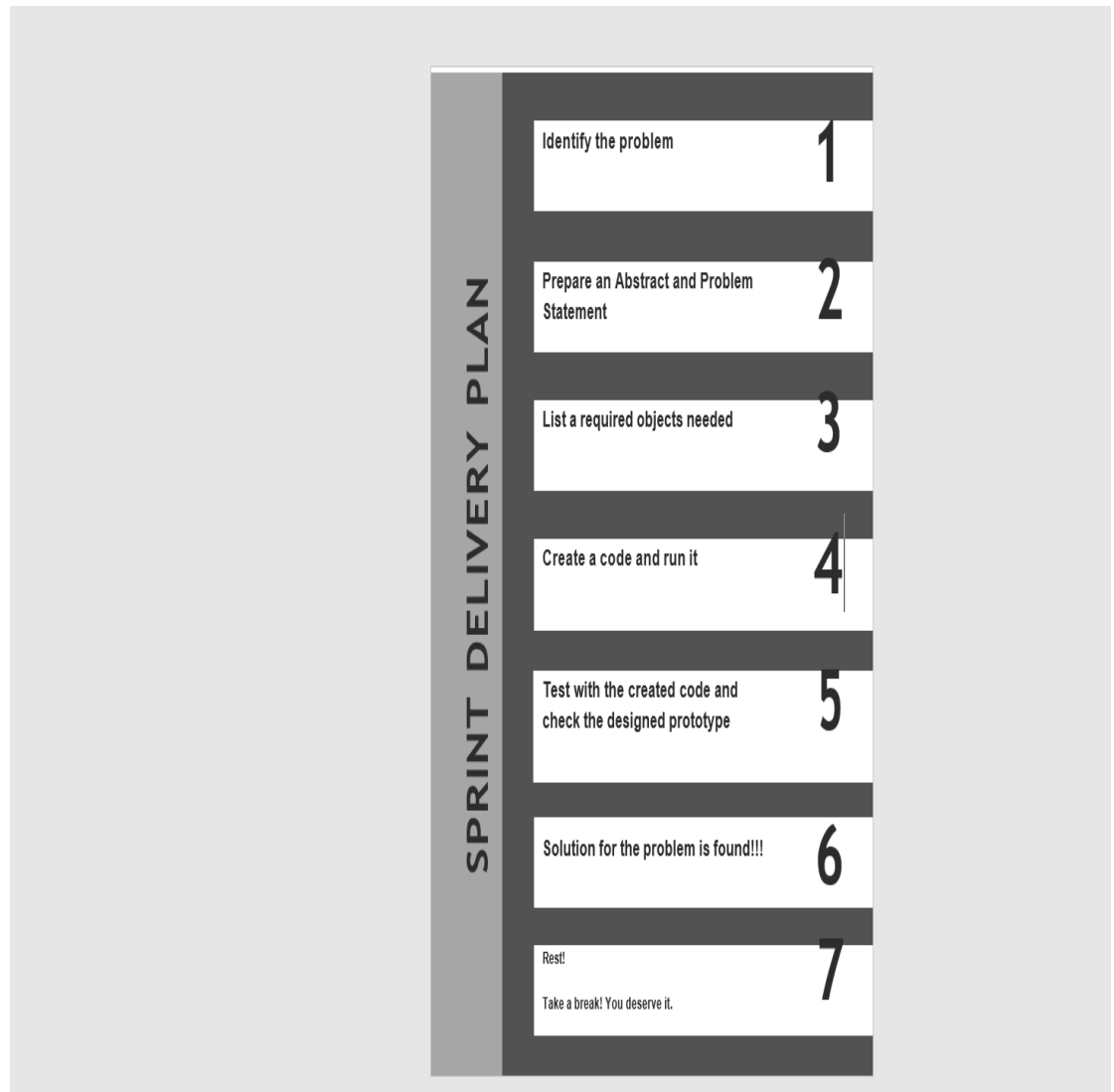
## 5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority
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
Customer(higher authority)	confirmation	USN-2	As a user I will receive confirmation message via email and once I received I can install the device in the owners place	I can receive confirmation email & click confirm	High
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
Customer (mobile user)	Mobile application	USN-4	As a user I can register by mobile application	I can register for gas detection device with owners mobile number and the alert message will be send by SMS	Medium
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
Customer (Web user)	Notification	USN-7	As a user when there is a critical situation regarding gas explosion the alert notification will be received through GSM module	Alert message is sent to owners mobile as an SMS	High
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

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

# 6 PROJECT PLANNING & SCHEDULING

## 6.1 Sprint Planning & Estimation



## 6.2. SPRINT DELIVERY SOLUTION

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	6Days	1Nov2022	5Nov2022	20	52022
Sprint-2	20	6Days	6Nov2022	10Nov2022	20	10Nov2022
Sprint-3	20	6Days	11Nov2022	15Nov2022	20	15Nov2022
Sprint-4	20	6Days	16Nov 2022	19Nov2022	20	19Nov2022

## 7. CODING & SOLUTIONING



## 7.1. FEATURE 1

IOT device

Wokwi software

Iot Watson platform

Node red

Cloudant db

Web UI

## 7.2 FEATURE 2

Registration

Login

Verification

Check the sensors

Buzzer the alarm

Fast sms



## DATA BASE SCHEME

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-
width, initial-scale=1">
<style>
body {font-family: Arial, Helvetica, sans-serif;}

/* Full-width input fields */
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
margin: 8px 0;
display: inline-block;
border: 1px solid #ccc;
box-sizing: border-box;
}

/* Set a style for all buttons */
button {
background-color: #04AA6D;
color: white;
padding: 14px 20px;
margin: 8px 0;
border: none;
cursor: pointer;
width: 100%;
}

button:hover {
opacity: 0.8;
}
```

**/\* Extra styles for the cancel button \*/**

```
.cancelbtn {  
    width: auto;  
    padding: 10px 18px;  
    background-color: #f44336;  
}
```

**/\* Center the image and position the close button  
\*/**

```
.imgcontainer {  
    text-align: center;  
    margin: 24px 0 12px 0;  
    position: relative;  
}
```

```
img.avatar {  
    width: 20%;  
    border-radius: 10%;  
}
```

```
.container {  
    padding: 16px;  
}
```

```
span.psw {  
    float: right;  
    padding-top: 16px;  
}
```

**/\* The Modal (background) \*/**

```
.modal {  
    display: none; /* Hidden by default */  
    position: fixed; /* Stay in place */
```

```
z-index: 1; /* Sit on top */  
left: 0;  
top: 0;  
width: 100%; /* Full width */  
height: 100%; /* Full height */  
overflow: auto; /* Enable scroll if needed */  
background-color: rgb(0,0,0); /* Fallback color */  
background-color: rgba(0,0,0,0.4); /* Black w/  
opacity */  
padding-top: 60px;  
}
```

```
/* Modal Content/Box */  
.modal-content {  
background-color: #fefefe;  
margin: 5% auto 15% auto; /* 5% from the top,  
15% from the bottom and centered */  
border: 1px solid #888;  
width: 80%; /* Could be more or less, depending  
on screen size */  
}
```

```
/* The Close Button (x) */  
.close {  
position: absolute;  
right: 25px;  
top: 0;  
color: #000;  
font-size: 35px;  
font-weight: bold;  
}
```

```
.close:hover,  
.close:focus {
```

```
    color: red;
    cursor: pointer;
}

/* Add Zoom Animation */
.animate {
    -webkit-animation: animatezoom 0.6s;
    animation: animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
    from {-webkit-transform: scale(0)}
    to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
    from {transform: scale(0)}
    to {transform: scale(1)}
}

/* Change styles for span and cancel button on
extra small screens */
@media screen and (max-width: 300px) {
    span.psw {
        display: block;
        float: none;
    }
    .cancelbtn {
        width: 100%;
    }
}
</style>
</head>
```

```
<body>
```

```
<h1 style="padding-top: 200px; text-align:
center;">Gas Leakage monitoring & Alerting
system for Industries</h1>
```

```
<button
onclick="document.getElementById('id01').style.d
isplay='block'" style="width:auto; margin-left:
600px;">Login</button>
```

```
<div id="id01" class="modal">
```

```
    <form class="modal-content animate"
method="post">
```

```
        <div class="imgcontainer">
```

```
            <span
onclick="document.getElementById('id01').style.d
isplay='none'" class="close" title="Close
Modal">&times;</span>
```

```
            
```

```
        </div>
```

```
        <div class="container">
```

```
            <label
for="uname"><b>Username</b></label>
```

```
            <input id="frm1" type="text"
placeholder="Enter Username" name="uname"
required>
```

```
            <label for="psw"><b>Password</b></label>
```

```
            <input type="password" placeholder="Enter
Password" name="psw" required>
```

```
        <button style="color: black"
onclick="window.location.href=('district.html')"
type="submit">signin</button>
        <label>
            <input type="checkbox" checked="checked"
name="remember"> Remember me
        </label>
    </div>
```

```
    <div class="container" style="background-
color:#f1f1f1">
        <button type="button"
onclick="document.getElementById('id01').style.d
isplay='none'"
class="cancelbtn">Cancel</button>
        <span class="psw">Forgot <a
href="#">password?</a></span>
    </div>
</form>
</div>
```

```
<script>
```

```
function myFunction() {
    var x = document.getElementById("frm1");
    var text = "";
    var i;
    for (i = 0; i < x.length ;i++) {
        text += x.elements[i].value + "<br>";
    }
}
```



```
document.getElementById("demo").innerHTML
= text;

}
// Get the modal action_page.php
var modal = document.getElementById('id01');

// When the user clicks anywhere outside of the
modal, close it
window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
    }
}
</script>

</body>
</html>
```

**PYTHON CODE:**

```
Test.py - C:/Users/thine/AppData/Local/Programs/Python/Python39/Test.py (3.9.6)
File Edit Format Run Options Window Help

#IBM Watson IoT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "0tuse0f",
        "typeId": "ESP32",
        "deviceId": "01"
    },
    "auth": {
        "token": "Gowth$@Bnk18"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    #cmd.data['command']

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s" % myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
```

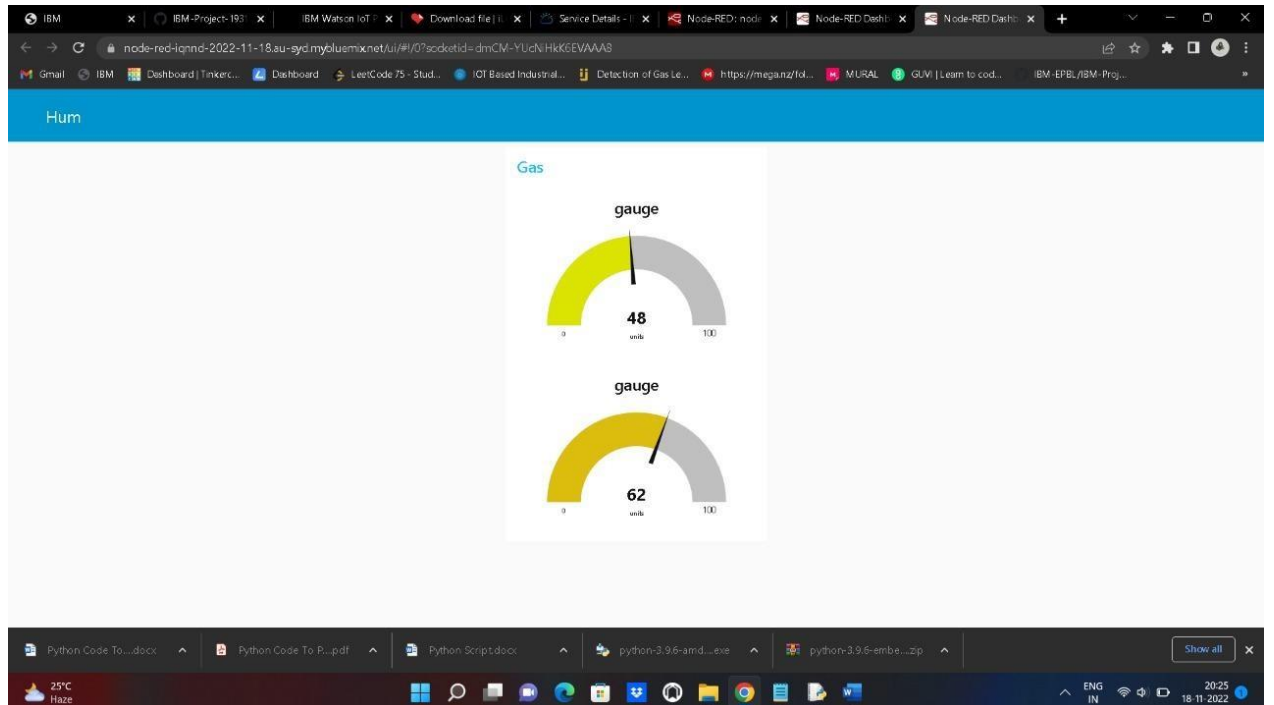
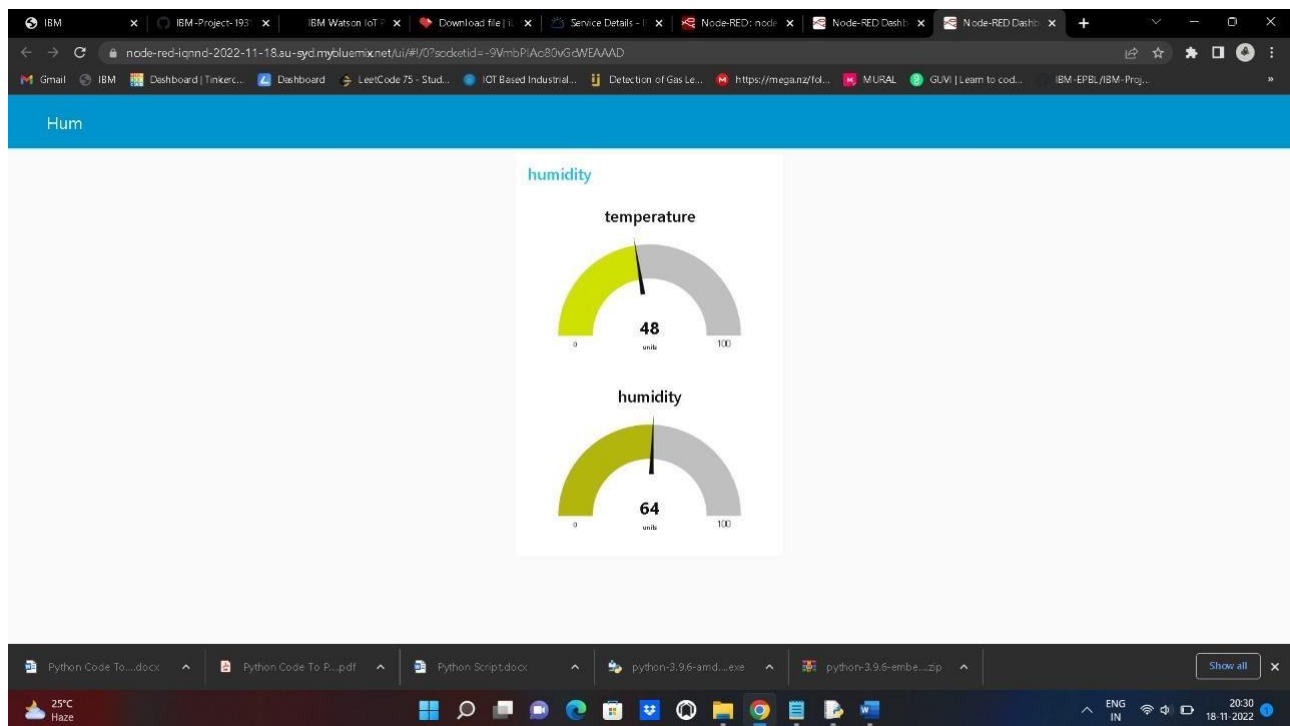
Ln: 33 Col: 0

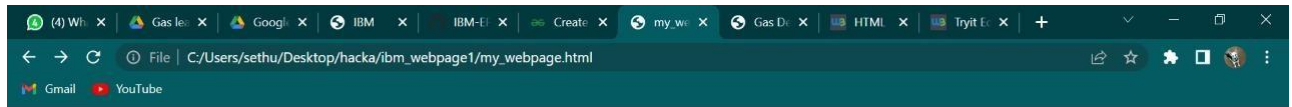
31°C Mostly sunny 03:28 PM 07-11-2022

## 7 TESTING

S NO	TEST CASE	FEATURE	STEPS TO EXECUTE	EXPECTED RESULT	ACTUAL RESULT	EXECUTED BY
1	FUNCTIONAL	LOGIN	LOGIN TO EXECUTE BY FILLING THE DETAILS	CORRECT LOGIN CREDENTIALS	WORKING AS EXPECTED	HARIPRASAD
2	FUNCTIONAL	REGISTRATION	REGISTRATION THROUGH FORMS	REGISTRATION FORM TO BE FILLED AND DISPLAYED	WORKING AS EXPECTED	HARIPRASAD & RAMYA
3	FUNCTIONAL	WOKWI	TO DEVELOP THE IOT DEVICE AND CODE THE IOT DEVICE	SENSE THE DATA	WORKING AS EXPECTED	RAMYA
4	FUNCTIONAL	IBM WATSON	PUSH THE SENSED DATA FROM WOKWI	SENSED DATA IN IBM WATSON	WORKING AS EXPECTED	SUREKA & SUREKA
5	FUNCTIONAL	NODE RED	TO CONNECT WITH THE IBM WATSON AND THEN COLLECT THE SENSED DATA AND DISPLAY IN NODE RED DASHBOARD	VISUAL REPRESENTATION OF SENSED DATA IN NODE RED DASHBOARD	WORKING AS EXPECTED	SUHITHA
6	TESTING	TEST THE ENTIRE WORK	TO CHECK ALL THE MENTIONED TESTCASE ARE WORKING PROPERLY	TEST CASE ARE WOKING PROPERLY	WORKING AS EXPECTED	SUREKA

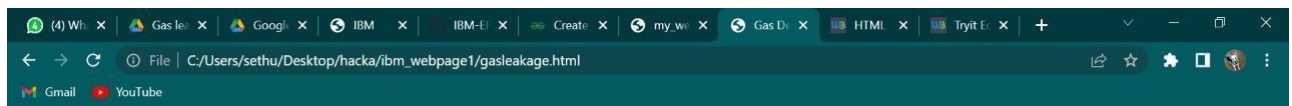
## 8 RESULTS



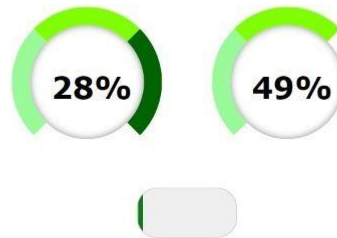


## Gas Leakage monitoring & Alerting system for Industries

Login



### Gas Leakage Detection system Analytics(PNT2022TMID08708)



Temperature : 28.00 C

Humidity : 49.50 g/m3

Gas : 287 ppm

## 9 ADVANTAGES AND DISADVANTAGES

### ADVANTAGES

It is a very useful system to implement in the industries or plant facilities to avoid catastrophic explosions. With the help of a gas monitoring solution, you can successfully measure temperature and humidity in the atmosphere, which results in improved plant facilities and ensures employee health safety.

## **DISADVANTAGES**

Poor stability and greater environmental impact; in particular, the selectivity of each sensor is not and the output parameters cannot be determined. Therefore, it should not be used in places where accurate measurement is required.

## **11. CONCLUSION**

This work presents the design and implementation of gas leakage detection system. Various works on gas leakages detection system was reviewed and presented. I was discovered that some of the existent research don't takes in to considerations the cost effectiveness for the purpose of implementation of gas leakages detection at individual/domestic uses, and not easy to be further modified. This research work had advanced in knowledge as it included an embedded system to alert users via multiple mobile phones for further action to be taken when leakage is detected. The device detects gas leakage using a highly sensitive MQ-2 gas sensor to activate a buzzer that alert people of leakages, and also sent an SMS with the information "Gas Leakage Detected" from the SIM800 GSM Module as a backup to alert the appropriate authority or facility owner of a gas leakage. By using various software such as node red, IBM Watson, cloudant we are collecting this information in the cloud and in the case of emergency it sends message to the user.

## **12. FUTURE SCOPE**

In this paper we use IOT technology for enhancing the existing safety standards. While making this prototype has been to bring a revolution in the field of safety against the leakage of harmful and toxic gases in environment and hence nullify any major or minor hazard being caused due to them. 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. This system will be able to detect the gas in environment using the gas sensors. This will prevent form the major harmful proble

## **13. APPENDIX**

### **Python Code:**

```
import random  
print('Hazardous Gas Level=',str(random.randint(0,100)))  
print('Temperature=',str(random.randint(0,100)))  
print('Humidity=',str(random.randint(0,100)))  
print('Pressure=',str(random.randint(0,100)))
```

```
Code.py - E:/PYTHON/Code.py (3.9.6)
File Edit Format Run Options Window Help

import random
print('Hazardous Gas Level=',str(random.randint(0,100)))
print('Temperature=',str(random.randint(0,100)))
print('Humidity=',str(random.randint(0,100)))
print('Pressure=',str(random.randint(0,100)))

IDLE Shell 396
File Edit Shell Debug Options Window Help
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:/PYTHON/Code.py =====
Hazardous Gas Level= 34
Temperature= 95
Humidity= 22
Pressure= 83
>>>
===== RESTART: E:/PYTHON/Code.py =====
Hazardous Gas Level= 27
Temperature= 11
Humidity= 14
Pressure= 1
>>>
===== RESTART: E:/PYTHON/Code.py =====
Hazardous Gas Level= 14
Temperature= 65
Humidity= 71
Pressure= 76
>>>|
```

```
Code.py - E:/PYTHON/Code.py (3.9.6)
File Edit Format Run Options Window Help

import random
print('Hazardous Gas Level=',str(random.randint(0,100)))
print('Temperature=',str(random.randint(0,100)))
print('Humidity=',str(random.randint(0,100)))
print('Pressure=',str(random.randint(0,100)))
```

## Watson Code:

```
{
    "Hazardous Gas": random(0,100),
    "Temperature": random(0,100),
    "Humidity": random(0,100),
    "Pressure": random(0,100)
}
```



The screenshot shows the IBM Watson IoT Platform dashboard. A device configuration window is open for a device named 'ESP32' (ID: 01). The window displays the 'Recent Events' tab, showing a list of events with columns: Event, Value, Format, and Last Received. The events are generated by a simulation running every minute. The payload for the events is shown in the 'Payload' section, containing random values for Hazardous Gas, Temperature, Humidity, and Pressure.

Event	Value	Format	Last Received
event_1	{"Hazardous Gas":28,"Temperature":19,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":94,"Temperature":28,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":68,"Temperature":65,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":85,"Temperature":77,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":66,"Temperature":58,"Humidit..."}	json	a few seconds ago

The payload configuration is as follows:

```

{
  "Hazardous Gas": random(0,100),
  "Temperature": random(0,100),
  "Humidity": random(0,100),
  "Pressure": random(0,100)
}
  
```

The screenshot shows the IBM Watson IoT Platform dashboard. A device configuration window is open for a device named 'ESP32' (ID: 01). The window displays the 'Recent Events' tab, showing a list of events with columns: Event, Value, Format, and Last Received. The events are generated by a simulation running every minute. The payload for the events is shown in the 'Payload' section, containing random values for Hazardous Gas, Temperature, Humidity, and Pressure.

Event	Value	Format	Last Received
event_1	{"Hazardous Gas":85,"Temperature":77,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":66,"Temperature":58,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":43,"Temperature":19,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":26,"Temperature":13,"Humidit..."}	json	a few seconds ago
event_1	{"Hazardous Gas":26,"Temperature":37,"Humidit..."}	json	a few seconds ago

The payload configuration is as follows:

```

{
  "Hazardous Gas": random(0,100),
  "Temperature": random(0,100),
  "Humidity": random(0,100),
  "Pressure": random(0,100)
}
  
```

1 Simulation running

#IBM Watson IOT Platform

#pip install wiotp-sdk import

wiotp.sdk.device import time

import random

```
myConfig = {  
    "identity": {  
        "orgId": "0tus0f",  
        "typeId": "ESP32",  
        "deviceId": "01"  
    },  
    "auth": {  
        "token": "Gowth@m@nk18"  
    }  
}
```

```
def myCommandCallback(cmd):  
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])  
    m=cmd.data['command']
```

```
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)  
client.connect()
```

```

while True: temp=random.randint(-
    20,125)

    hum=random.randint(0,100)

    myData={'temperature':temp, 'humidity':hum}

    client.publishEvent(eventId="status",  msgFormat="json",  data=myData,  qos=0,
onPublish=None)

    print("Published data Successfully: %s", myData)

    client.commandCallback  =  myCommandCallback

    time.sleep(2)

client.disconnect()

```

```

<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
body { font-family: Arial, Helvetica, sans-serif;}

```

```

/* Full-width input fields */
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
margin: 8px 0;
display: inline-block;
border: 1px solid #ccc;
box-sizing: border-box;
}

```

```

/* Set a style for all buttons */
button {
background-color: #04AA6D;
color: white;
padding: 14px 20px;
margin: 8px 0;
border: none;
cursor: pointer;
width: 100%;
}

```

```

button:hover {
opacity: 0.8;
}

```

```

/* Extra styles for the cancel button */
.cancelbtn {
width: auto;
padding: 10px 18px;
background-color: #f44336;
}

```

```

/* Center the image and position the close button */
.imgcontainer {
text-align: center;

```

```

margin: 24px 0 12px 0;
position: relative;
}

img.avatar {
width: 20%;
border-radius: 10%;
}

.container {
padding: 16px;
}

span.psw {
float: right;
padding-top: 16px;
}

/* The Modal (background) */
.modal {
display: none; /* Hidden by default */
position: fixed; /* Stay in place */
z-index: 1; /* Sit on top */
left: 0;
top: 0;
width: 100%; /* Full width */
height: 100%; /* Full height */
overflow: auto; /* Enable scroll if needed */
background-color: rgb(0,0,0); /* Fallback color */
background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
padding-top: 60px;
}

/* Modal Content/Box */
.modal-content {
background-color: #fefefe;
margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
border: 1px solid #888;
width: 80%; /* Could be more or less, depending on screen size */
}

/* The Close Button (x) */
.close {
position: absolute;
right: 25px;
top: 0;
color: #000;
font-size: 35px;
font-weight: bold;
}

.close:hover,
.close:focus {
color: red;
cursor: pointer;
}

/* Add Zoom Animation */
.animate {
-webkit-animation: animatezoom 0.6s;

```

```

    animation: animatezoom 0.6s
}

@-webkit-keyframes animatezoom {
    from {-webkit-transform: scale(0)}
    to {-webkit-transform: scale(1)}
}

@keyframes animatezoom {
    from {transform: scale(0)}
    to {transform: scale(1)}
}

/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
    span.psw {
        display: block;
        float: none;
    }
    .cancelbtn {
        width: 100%;
    }
}
</style>
</head>
<body>

<h1 style="padding-top: 200px; text-align: center;">Gas Leakage monitoring & Alerting system for
Industries</h1>

<button onclick="document.getElementById('id01').style.display='block'" style="width:auto; margin-
left: 600px;">Login</button>

<div id="id01" class="modal">

    <form class="modal-content animate" method="post">
        <div class="imgcontainer">
            <span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close
Modal">&times;</span>
            
        </div>

        <div class="container">
            <label for="uname"><b>Username</b></label>
            <input id="frm1" type="text" placeholder="Enter Username" name="uname" required>

            <label for="psw"><b>Password</b></label>
            <input type="password" placeholder="Enter Password" name="psw" required>

            <button style="color: black" onclick="window.location.href=('district.html')"
type="submit">signin</button>
            <label>
                <input type="checkbox" checked="checked" name="remember"> Remember me
            </label>
        </div>

        <div class="container" style="background-color:#f1f1f1">
            <button type="button" onclick="document.getElementById('id01').style.display='none'"

```

```
class="cancelbtn">Cancel</button>
    <span class="psw">Forgot <a href="#">password?</a></span>
</div>
</form>
</div>
```

```
<script>
```

```
function myFunction() {
    var x = document.getElementById("frm1");
    var text = "";
    var i;
    for (i = 0; i < x.length ;i++) {
        text += x.elements[i].value + "<br>";
    }
    document.getElementById("demo").innerHTML = text;
```

```
}
```

```
// Get the modal action_page.php
```

```
var modal = document.getElementById('id01');
```

```
// When the user clicks anywhere outside of the modal, close it
```

```
window.onclick = function(event) {
```

```
    if (event.target == modal) {
```

```
        modal.style.display = "none";
```

```
    }
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

## **PYTHON CODE:**

```
Test.py - C:/Users/thine/AppData/Local/Programs/Python/Python39/Test.py (3.9.6)
File Edit Format Run Options Window Help

#IBM Watson IoT Platform
#pip install wiotp-sdk
import wiotp.sdk.device
import time
import random
myConfig = {
    "identity": {
        "orgId": "0tus06f",
        "typeId": "ESP32",
        "deviceId": "01"
    },
    "auth": {
        "token": "Gowrh$mk18"
    }
}

def myCommandCallback(cmd):
    print("Message received from IBM IoT Platform: %s" % cmd.data['command'])
    m=cmd.data['command']

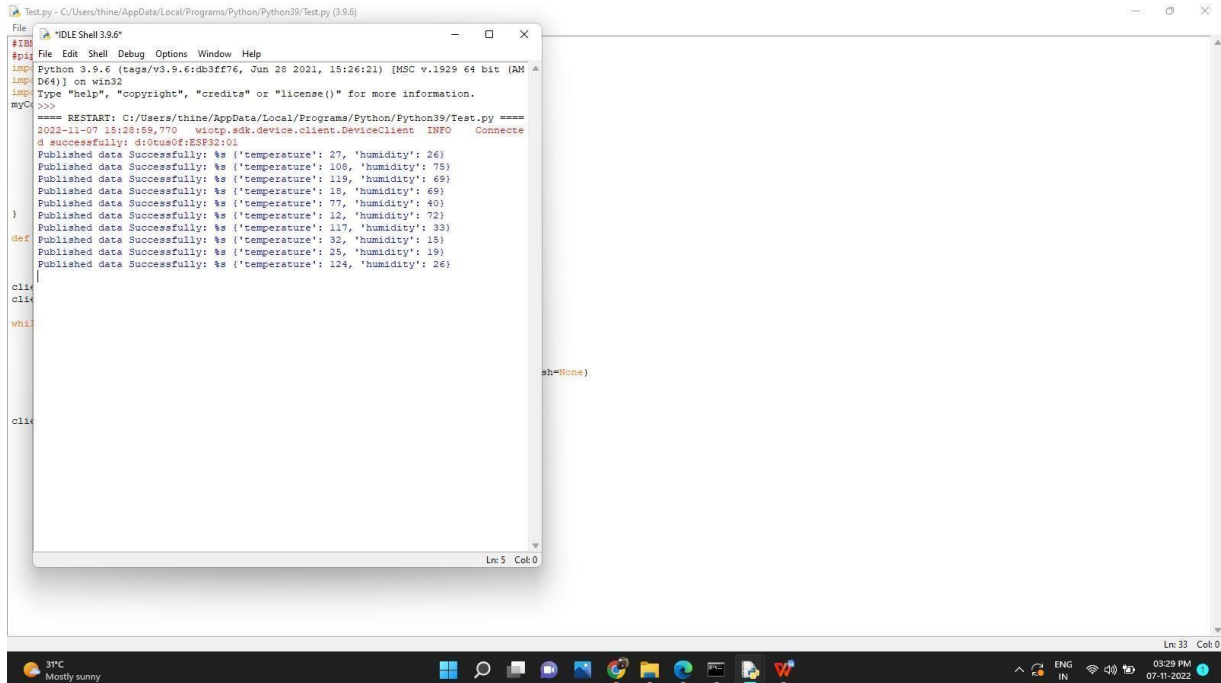
client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)
client.connect()

while True:
    temp=random.randint(-20,125)
    hum=random.randint(0,100)
    myData={'temperature':temp, 'humidity':hum}
    client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)
    print("Published data Successfully: %s" % myData)
    client.commandCallback = myCommandCallback
    time.sleep(2)
client.disconnect()
```

Ln: 33 Col: 0

31°C Mostly sunny 03:28 PM 07-11-2022

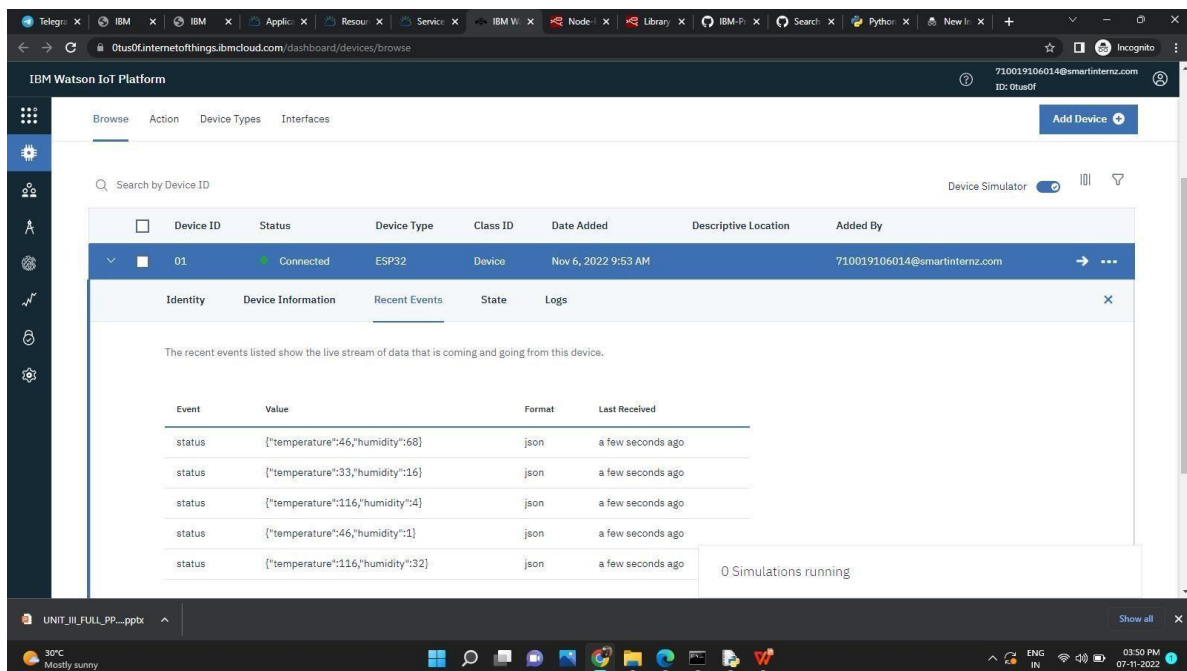
## PYTHON OUTPUT:



The screenshot shows a Python IDE window titled "Test.py - C:/Users/thine/AppData/Local/Programs/Python/Python39/Python39/Test.py (3.9.6)". The code defines a function `publish_data` that simulates an ESP32 device publishing temperature and humidity data. The output shows the device connecting successfully and publishing 10 data points.

```
#IBM
#p14
Python 3.9.6 (tags/v3.9.6:db3fff7e, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/thine/AppData/Local/Programs/Python/Python39/Test.py =====
2022-11-07 15:28:59,770 - mqtt.esk.device.client.DeviceClient - INFO - Connect
d successfully: d:0tusOf:ESP32:01
Published data Successfully: %s ('temperature': 27, 'humidity': 26)
Published data Successfully: %s ('temperature': 108, 'humidity': 75)
Published data Successfully: %s ('temperature': 119, 'humidity': 49)
Published data Successfully: %s ('temperature': 18, 'humidity': 69)
Published data Successfully: %s ('temperature': 77, 'humidity': 40)
Published data Successfully: %s ('temperature': 12, 'humidity': 72)
Published data Successfully: %s ('temperature': 117, 'humidity': 33)
Published data Successfully: %s ('temperature': 32, 'humidity': 15)
Published data Successfully: %s ('temperature': 25, 'humidity': 19)
Published data Successfully: %s ('temperature': 124, 'humidity': 26)
```

## Watson Cloud IBM:



The screenshot shows the IBM Watson IoT Platform dashboard. The main view displays a table of devices, with one device (ID: 01) selected. The selected device is in a "Connected" state and is an "ESP32" type. The dashboard shows the device's recent events, which are a stream of JSON data containing temperature and humidity values. A "Device Simulator" toggle is visible, and a notification indicates "0 Simulations running".

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By
01	Connected	ESP32	Device	Nov 6, 2022 9:53 AM		710019106014@smartinternz.com

Event	Value	Format	Last Received
status	["temperature":46,"humidity":68]	json	a few seconds ago
status	["temperature":33,"humidity":16]	json	a few seconds ago
status	["temperature":116,"humidity":4]	json	a few seconds ago
status	["temperature":46,"humidity":1]	json	a few seconds ago
status	["temperature":116,"humidity":32]	json	a few seconds ago



