Project Development Phase
Sprint Delivery - I

PROJECT TITLE	**Gas Leakage Monitoring and Alerting System**
TEAM ID	**PNT2022TMID06977**

Introduction:

Gas leakage detection systems are an integral part of a safety system, providing the first line of defense against the possible disasters of gas leakage. It detects the gas leakage and triggers an alert system to activate safety precautions. Some leakages are too small to be smelled or are of an unscented gas, so it's a necessary investment to install a gas leakage detection system.

Problem Statement:

Develop an efficient Gas Leakage Monitoring & Alerting System for Industries Such that The leakage of gases only can be detected by human nearby and if there are no human nearby, it cannot be detected. But sometimes it cannot be detected by human that has a low sense of smell. Thus, this system will help to detect the presence of gas leakage.

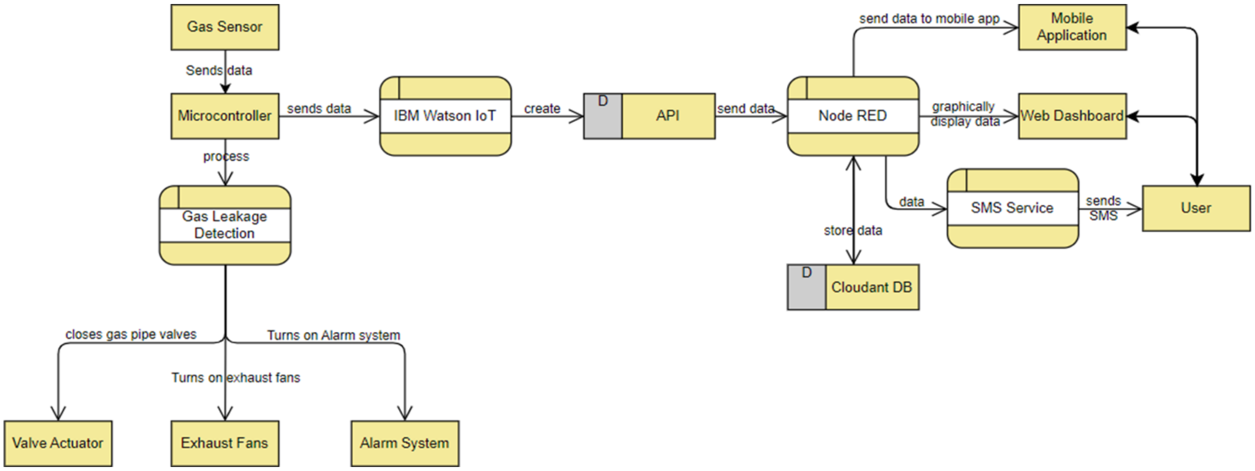
Proposed Solution:

Detect the presence of toxic gases and asphyxiants such as H2S, Methane, and CO in your industrial facilities or commercial buildings through gas sensor. Get alerts about presence of such gases or increasing temperatures through gas analyzers and monitors of our solution; and take immediate actions to curb fires and explosions.

Theoretical Analysis:

Block Diagram:

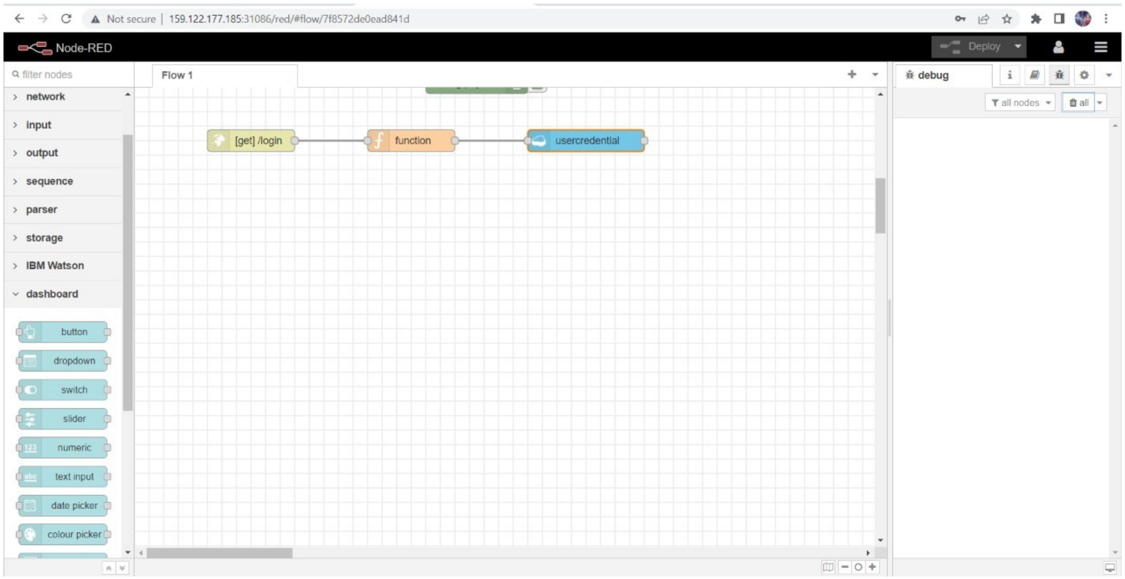
In order to implement the solution, the following approach as shown in the block diagram is used



Required Software Installation:

Node-Red

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services aspart of the Internet of Things. Node-RED provides a web browser-based flow editor,which can be used to create JavaScript functions.



Installation :

• First install npm/node.js

• Open cmd prompt • Type => npm install node-red

To run the application :

• Open cmd prompt

• Type=>node-red

• Then open http://localhost:1880/ in browser

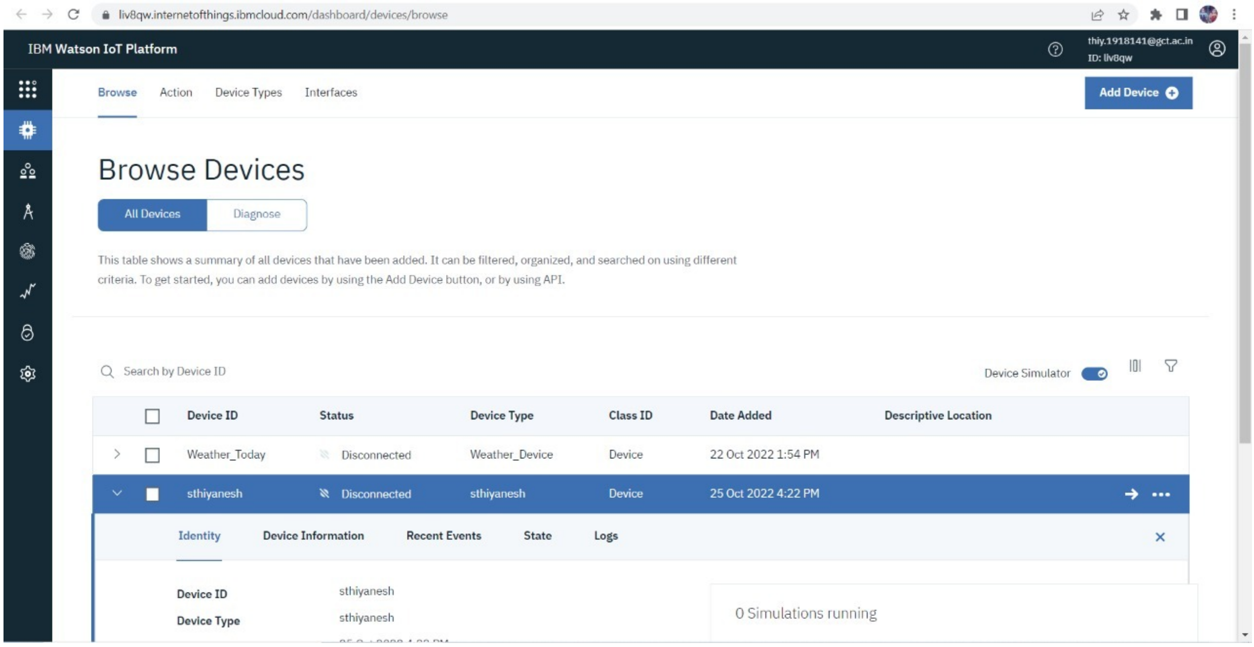
Installation of IBM IoT and Dashboard nodes for Node-Red

In order to connect to IBM Watson IoT platform and create the Web App UI these nodes are required 1. IBM IoT node

2. Dashboard node

IBM Watson IoT Platform

A fully managed, cloud-hosted service with capabilities for device registration, connectivity, control, rapid visualization and data storage. IBM Watson IoT Platformis a managed, cloud-hosted service designed to make it simple to derive value fromyour IoT Devices.



Browse Devices

This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
Weather_Today	Disconnected	Weather_Device	Device	22 Oct 2022 1:54 PM	
sthiyanesh	Disconnected	sthiyanesh	Device	25 Oct 2022 4:22 PM	

Device ID sthiyanesh

Device Type sthiyanesh

0 Simulations running

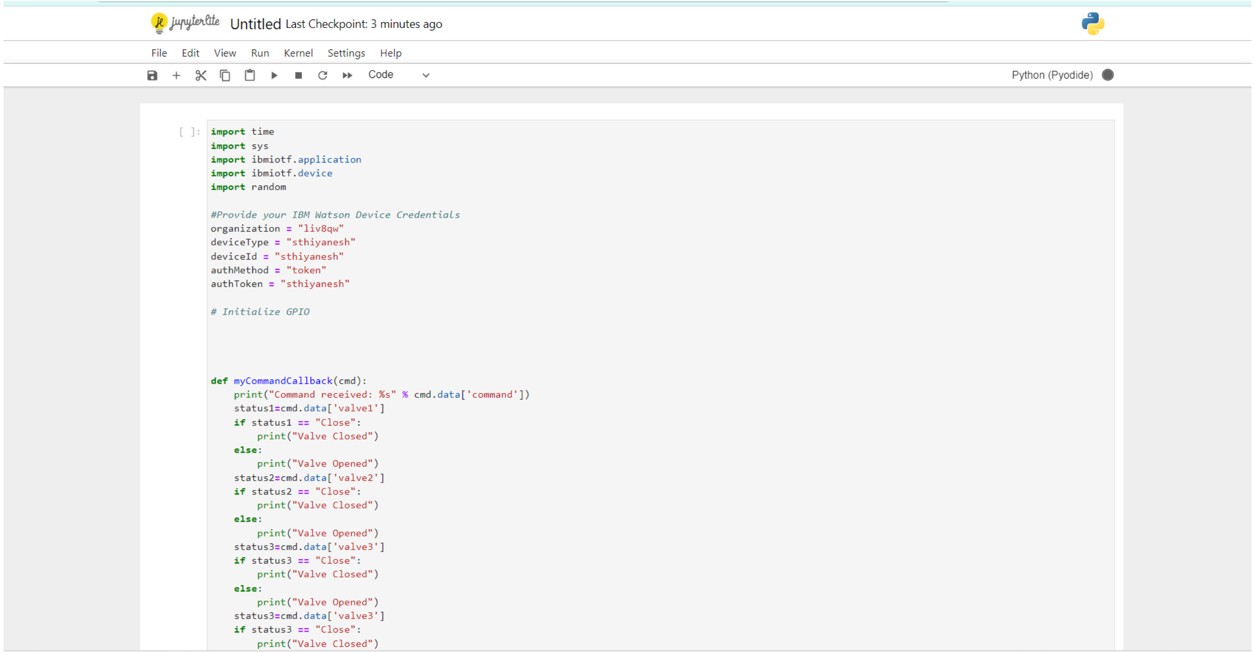
Steps to configure:

- Create an account in IBM cloud using your email ID
- Create IBM Watson Platform in services in your IBM cloud account
- Launch the IBM Watson IoT Platform
- Create a new device
- Give credentials like device type, device ID, Auth. Token
- Create API key and store API key and token elsewhere.

Python IDE

Install Python3 compiler

Install any python IDE to execute python scripts, in my case I used jupyter notebook to execute the code.

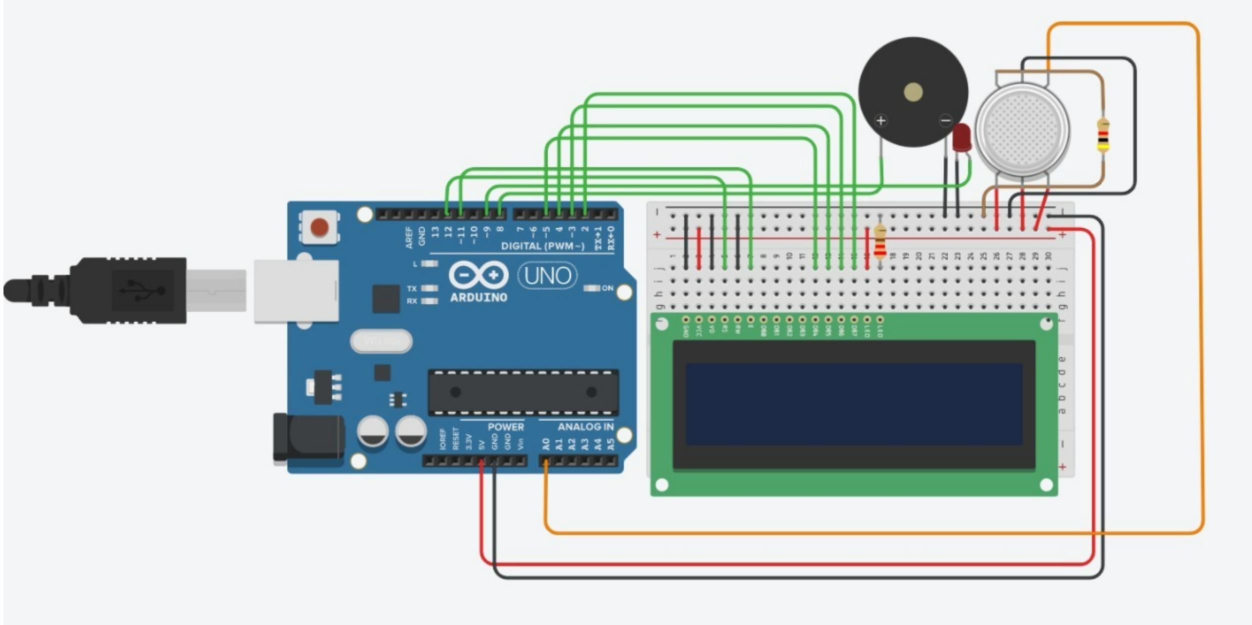


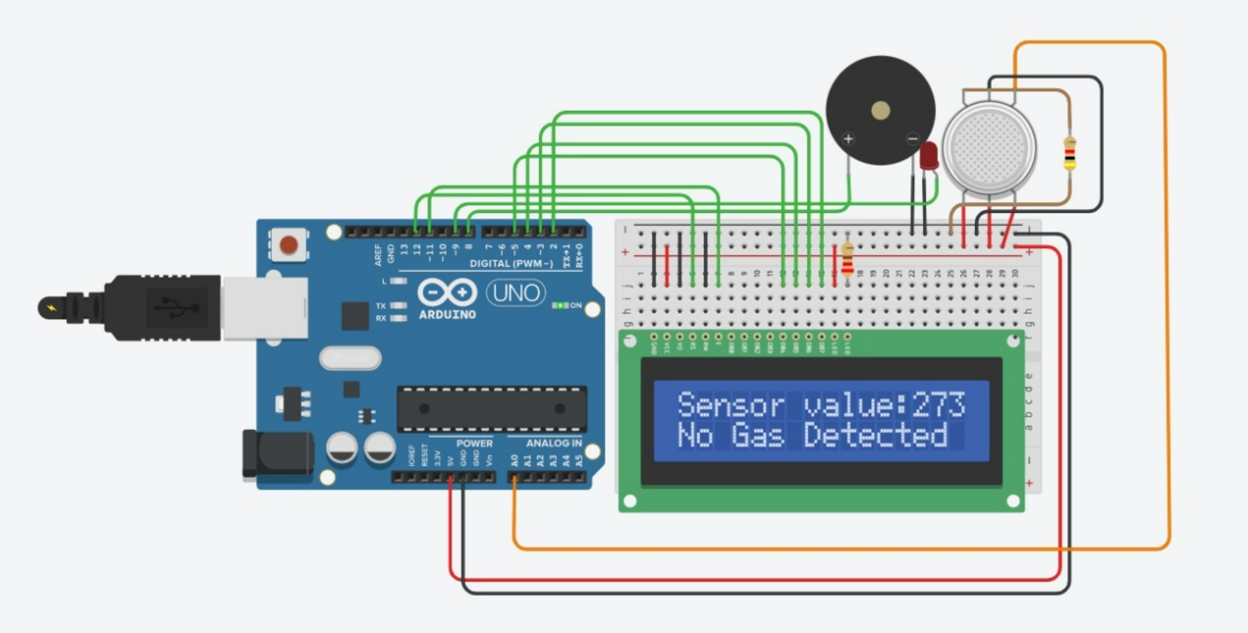
```python
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

#Provide your IBM Watson Device Credentials
organization = "liv8qw"
deviceType = "sthiyanesh"
deviceId = "sthiyanesh"
authMethod = "token"
authToken = "sthiyanesh"

# Initialize GPIO



def myCommandCallback(cmd):
    print("Command received: %s" % cmd.data['command'])
    status1=cmd.data['valve1']
    if status1 == "Close":
        print("Valve Closed")
    else:
        print("Valve Opened")
    status2=cmd.data['valve2']
    if status2 == "Close":
        print("Valve Closed")
    else:
        print("Valve Opened")
    status3=cmd.data['valve3']
    if status3 == "Close":
        print("Valve Closed")
    else:
        print("Valve Opened")
    status3=cmd.data['valve3']
    if status3 == "Close":
        print("Valve Closed")
```

Code:

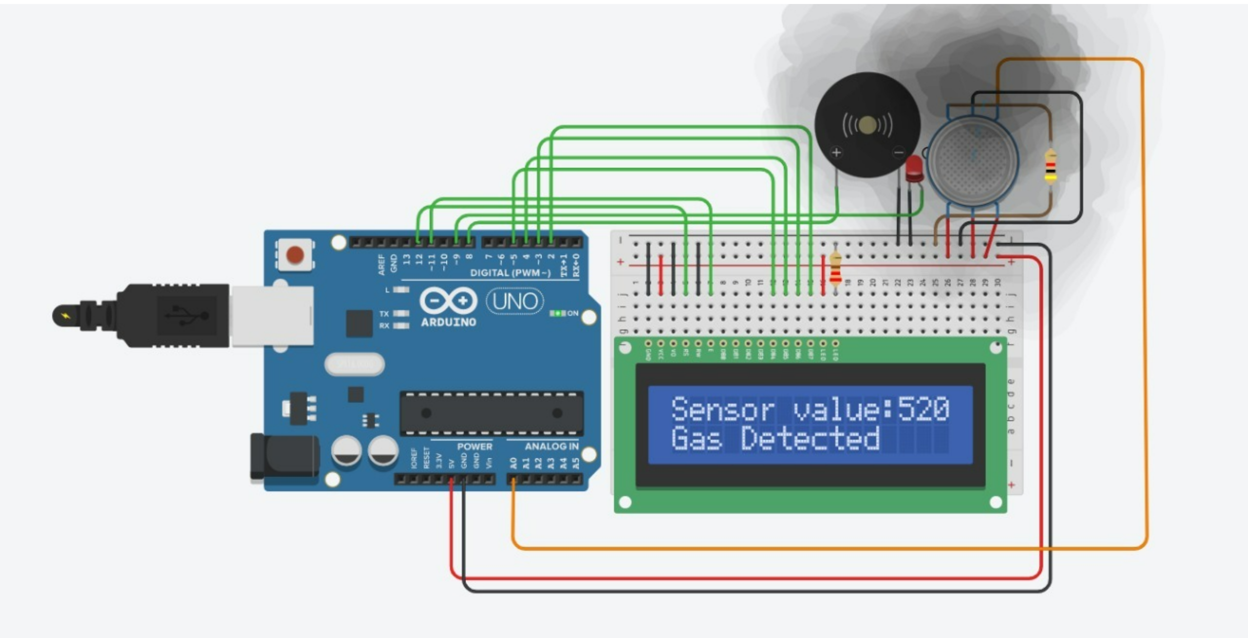
```
#include<LiquidCrystal.h>
int gasReading = 0;
int LED = 9;
int Buzzer = 8;
int gasSensor = A0;
int flag = 1;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
 lcd.begin(16, 2);
  pinMode(LED, OUTPUT);
  pinMode(Buzzer, OUTPUT);
  pinMode(gasSensor, INPUT);
}

void loop()
{
  gasReading = analogRead(gasSensor);
  String p = "Gas"+gasReading;
  lcd.setCursor(0,0);
  lcd.print(String("Sensor value:")+String(gasReading));
  if(gasReading>400){
    if(flag == 0) {
      lcd.setCursor(0,1);
      lcd.print("                ");
    }
    flag=1;
    lcd.setCursor(0,1);
    lcd.print("Gas Detected");
    digitalWrite(LED, HIGH);
    digitalWrite(Buzzer, HIGH);
  }else {
    if(flag == 1) {
      lcd.setCursor(0,1);
      lcd.print("                ");
    }
    flag=0;
    lcd.setCursor(0,1);
    lcd.print("No Gas Detected");
    digitalWrite(LED, LOW);
    digitalWrite(Buzzer, LOW);
  }
  delay(500);
}
```

Output:





```
Sensor value:273
No Gas Detected
```



```
Sensor value:520
Gas Detected
```