<u>Sprint Delivery – 1</u>

Project	GAS LEAKAGE MONITORING AND ALERTING SYSTEM
Team ID	PNT2022TMID11539
Date	12 November 2022

1. Introduction

The main aim of this project is to help the industries in detecting the leakage of harmful gases along with monitoring and alerting the admins by notifying them using IOT. Internet of things aim towards making life simpler by automating every small task around us.

2. Problem Statement

Gas leakages are a common problem in homes and industries. If not detected and corrected at the right time, it can cause lost of lives and properties.. IOT powered gas detection solution uses gas sensors to identify the presence of toxic gases.

3. Proposed Solution

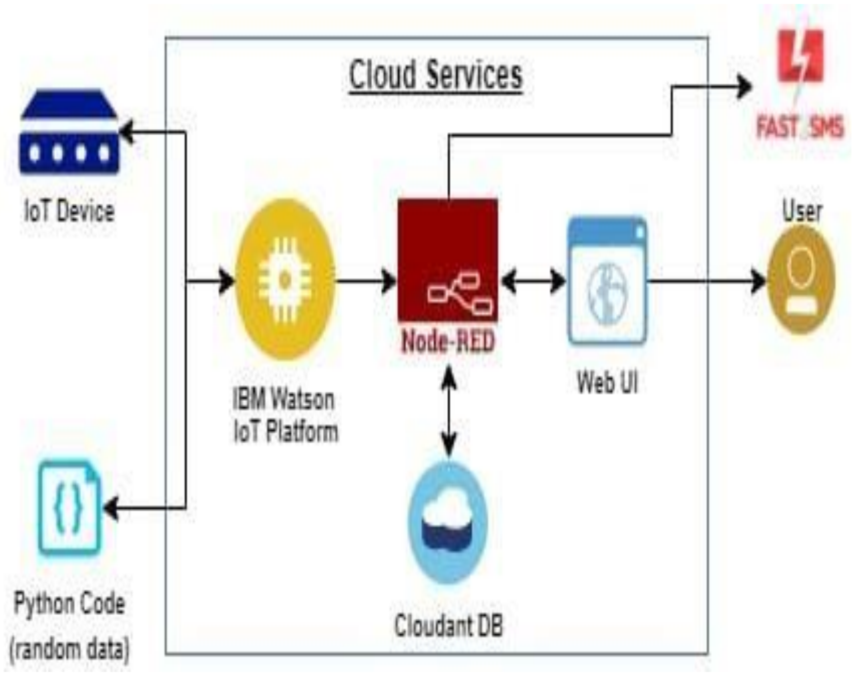
In order to improve the workers safety and make the working environment easier, we introduce IoT services to the industries in which we use cloud services and internet to enable industries to continuously monitor via internet. They can monitor the gas leakages and control the

Accidents and save human lives.

4. Theoretical Analysis

Block Diagram

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

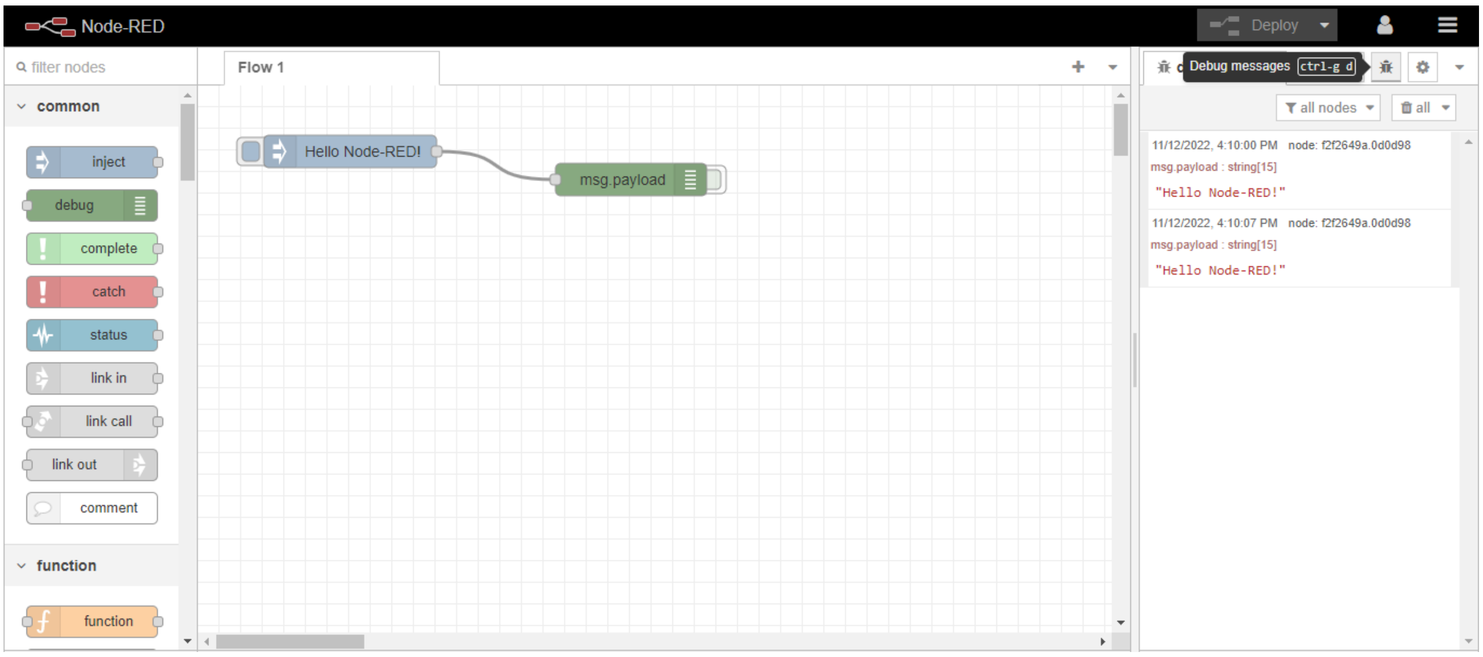


Required Software Installation

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 as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.

Installation:



```
11/12/2022, 4:10:00 PM   node: f2f2649a.0d0d98
msg.payload : string[15]
"Hello Node-RED!"
11/12/2022, 4:10:07 PM   node: f2f2649a.0d0d98
msg.payload : string[15]
"Hello Node-RED!"
```

- 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

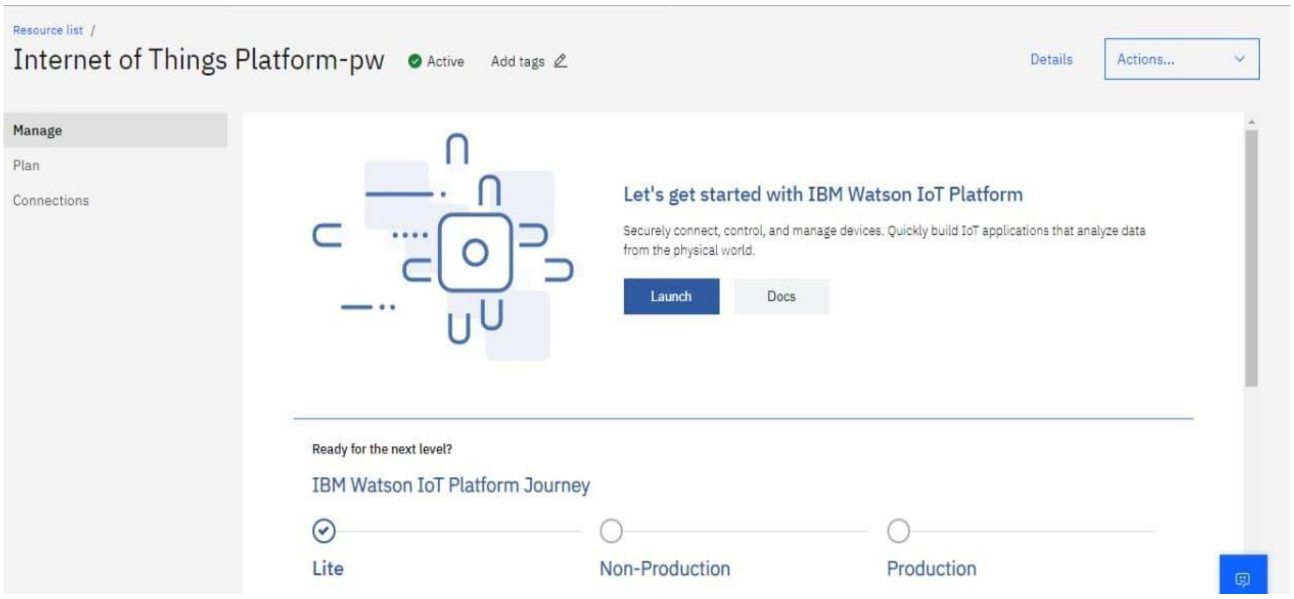
Installation of IBM IoT and Dashboard nodes for Node-Red

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 Platform is a managed, cloud-hosted service designed to make it simple to derive value from your IoT devices.



Let's get started with IBM Watson IoT Platform

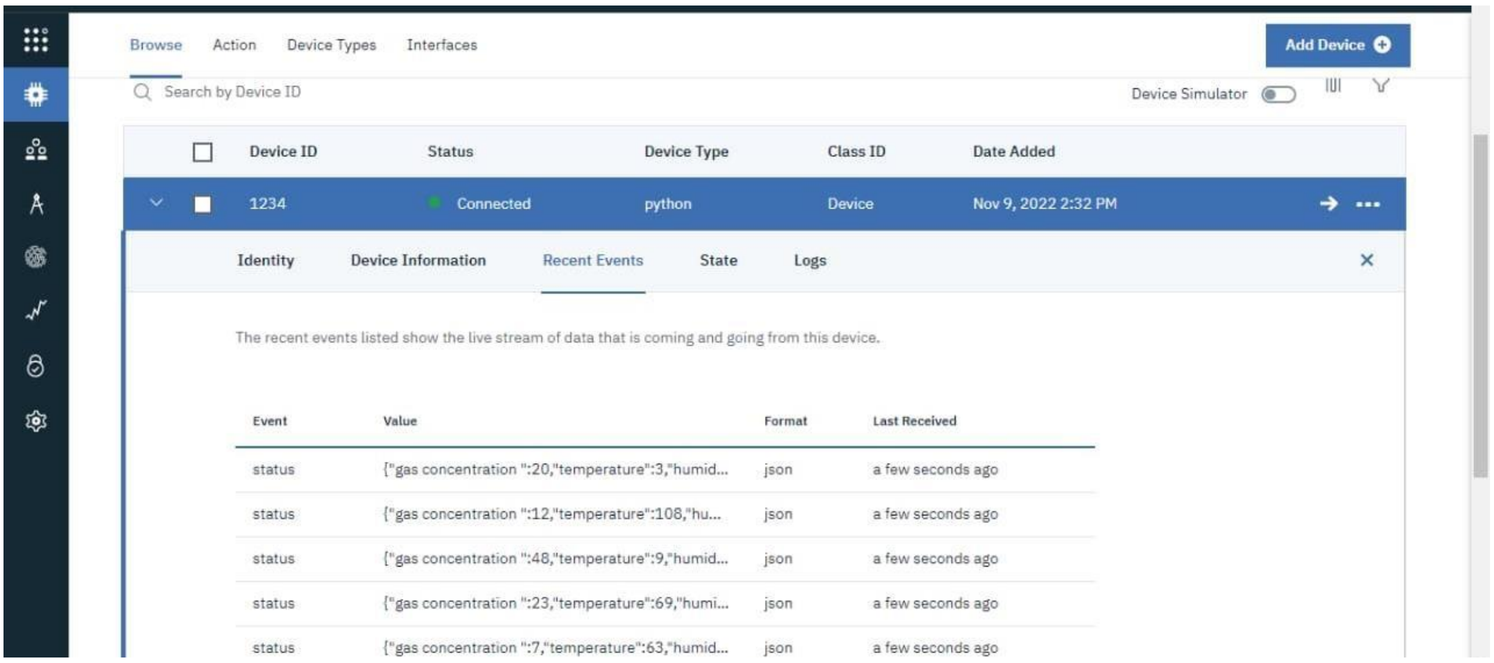
Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.

Ready for the next level?

IBM Watson IoT Platform Journey

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.



Device ID	Status	Device Type	Class ID	Date Added
1234	Connected	python	Device	Nov 9, 2022 2:32 PM

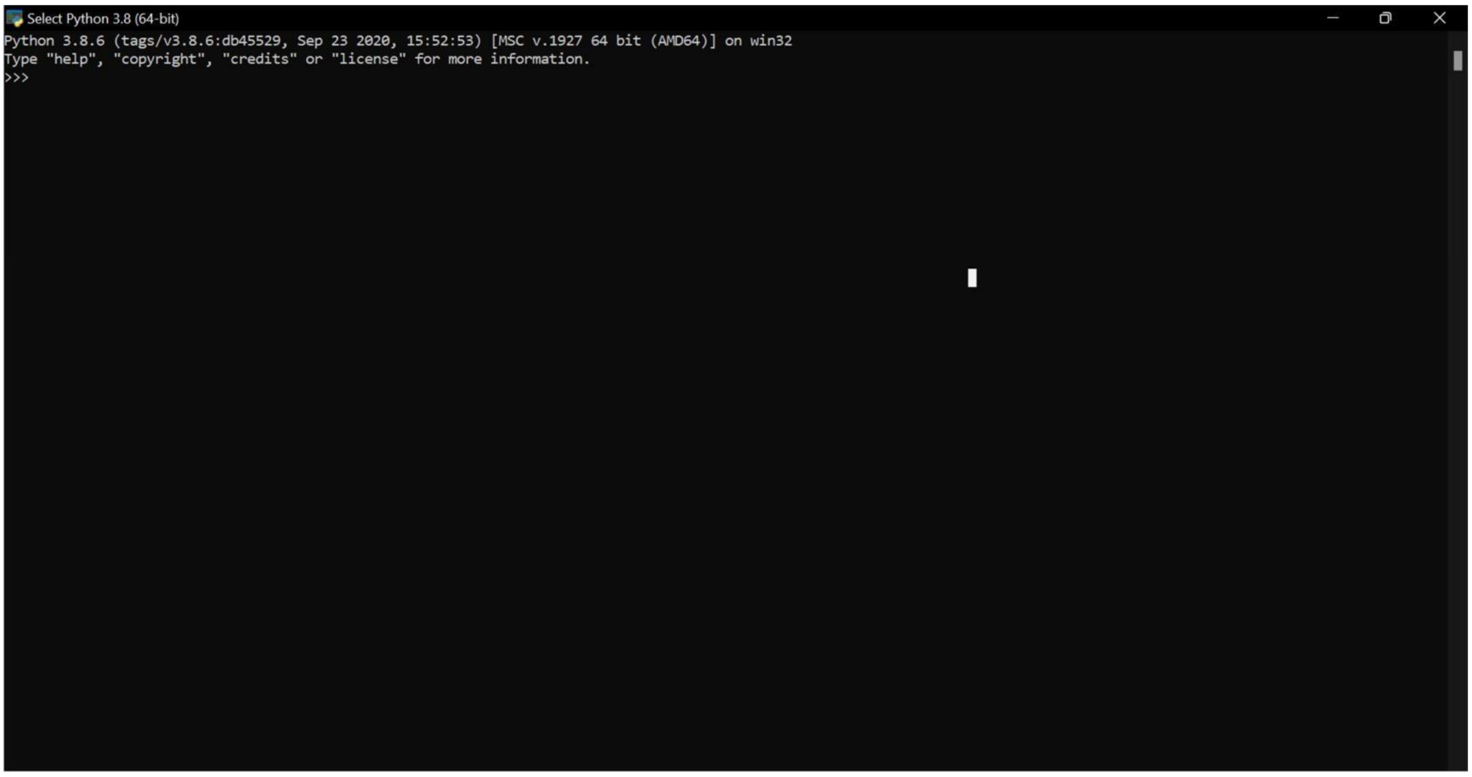
The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
status	{"gas concentration ":20,"temperature":3,"humid...	json	a few seconds ago
status	{"gas concentration ":12,"temperature":108,"hu...	json	a few seconds ago
status	{"gas concentration ":48,"temperature":9,"humid...	json	a few seconds ago
status	{"gas concentration ":23,"temperature":69,"humi...	json	a few seconds ago
status	{"gas concentration ":7,"temperature":63,"humid...	json	a few seconds ago

Python IDE

Install Python3 compiler

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



```
Select Python 3.8 (64-bit)
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Code:

```
import time

import sys

import ibmiotf.application

import ibmiotf.device

import random



#Provide your IBM Watson Device Credentials

organization = "jjrtf7"

deviceType = "ESP32"

deviceId = "1234"

authMethod = "token"
```

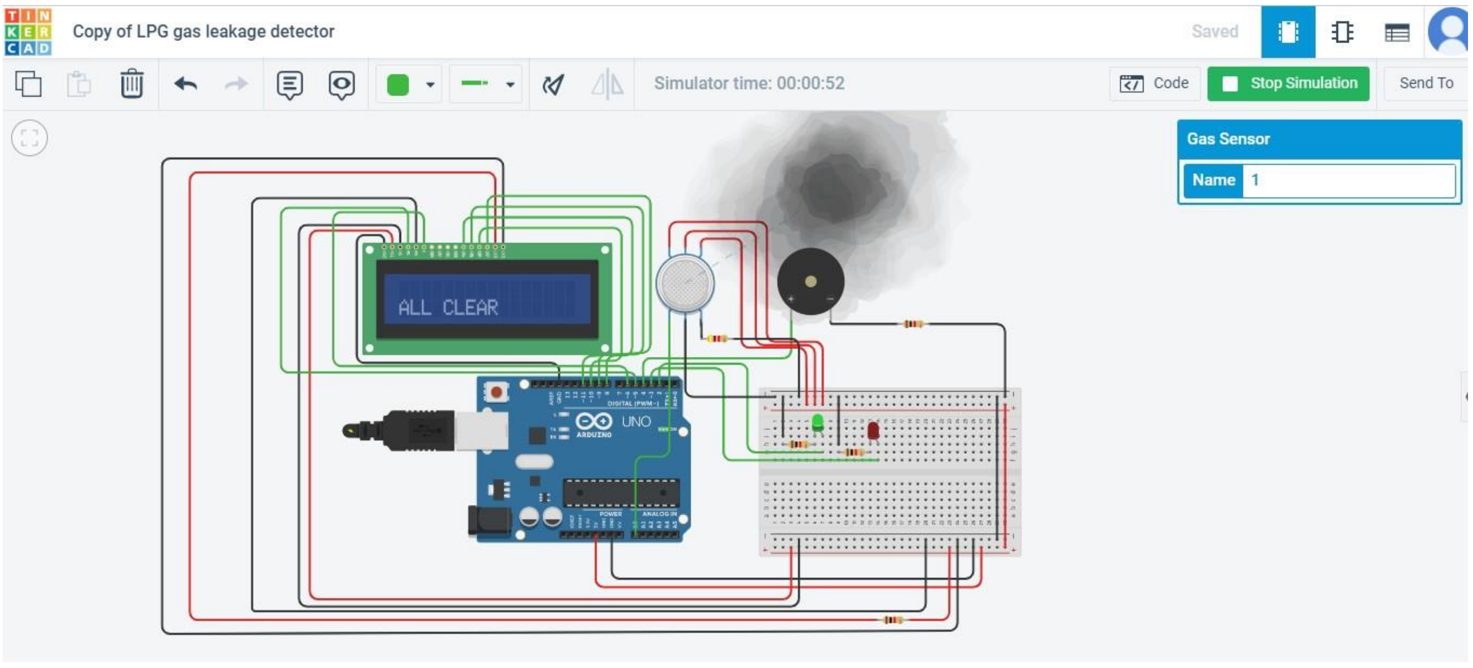
```
authToken = "12345678"

# Initialize GPIO

def myCommandCallback(cmd):
   print("Command received: %s" % cmd.data['command'])
   status=cmd.data['command']
   if status=="switchon":
      print ("Switch is on")
   else :
      print ("Switch is off")

   #print(cmd)

try:
   deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-
method": authMethod, "auth-token": authToken}
   deviceCli = ibmiotf.device.Client(deviceOptions)
   #..............................................

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

# Connect and send a datapoint "hello" with value "world" into the cloud as an
event of type "greeting" 10 times
deviceCli.connect()
```

```
while True:
        #Get Sensor Data from DHT11

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

        data = { 'temp' : temp, 'Humid': Humid, "gasconcentration": 
gasconcentration}

        #print data
        def myOnPublishCallback():
            print ("Published Temperature = %s C" % temp, "Humidity = %s %%" % 
Humid, "gasconcentration = %s %%" % gasconcentration, "to IBM Watson")

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

        deviceCli.commandCallback = myCommandCallback

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

Arduino code for C :

```
 #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 = 200;

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");
    delay(700);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("EVACUATE");
    delay(700);
  }
  else
  {
    digitalWrite(greenled,HIGH);
    digitalWrite(redled,LOW);
    noTone(buzzer);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("SAFE");
    delay(700);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("ALL CLEAR");
    delay(700);
  }
}
```

https://www.tinkercad.com/things/5irDkBFIFQq



```
#include <LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
#include<SoftwareSerial.h>
SoftwareSerial mySerial(9, 10);
int gasValue = A0; // smoke / gas sensor connected with analog pin A1 of the arduino / mega.
int data = 0;
int buzzer = 13;
int G_led = 8; // choose the pin for the Green LED
int R_led = 9; // choose the pin for the Red Led

void setup()
{
pinMode(buzzer,OUTPUT);
pinMode(R_led,OUTPUT); // declare Red LED as output
pinMode(G_led,OUTPUT); // declare Green LED as output
randomSeed(analogRead(0));
mySerial.begin(9600); // Setting the baud rate of GSM Module
Serial.begin(9600); // Setting the baud rate of Serial Monitor (Arduino)
lcd.begin(16,2);
pinMode(gasValue, INPUT);
lcd.print (&quot; Gas Leakage &quot;);
lcd.setCursor(0,1);
lcd.print (&quot; Detector Alarm &quot;);
delay(3000);
lcd.clear();
}
void loop()
{
data = analogRead(gasValue);
Serial.print(&quot;Gas Level: &quot;);
Serial.println(data);
lcd.print (&quot;Gas Scan is ON&quot;);
lcd.setCursor(0,1);
lcd.print(&quot;Gas Level: &quot;);
lcd.print(data);
delay(1000);
if ( data &gt; 90) //
{

digitalWrite(buzzer, HIGH);
digitalWrite(R_led, HIGH); // Turn LED on.
digitalWrite(G_led, LOW); // Turn LED off.
SendMessage();
Serial.print(&quot;Gas detect alarm&quot;);
lcd.clear();
lcd.setCursor(0,0);
lcd.print(&quot;Gas Level Exceed&quot;);
lcd.setCursor(0,1);
lcd.print(&quot;SMS Sent&quot;);
delay(1000);
}
else
{
digitalWrite(buzzer, LOW);
digitalWrite(R_led, LOW); // Turn LED off.
digitalWrite(G_led, HIGH); // Turn LED on.
Serial.print(&quot;Gas Level Low&quot;);
lcd.clear();
lcd.setCursor(0,0);
lcd.print(&quot;Gas Level Normal&quot;);
delay(1000);
}
lcd.clear();
}
void SendMessage()
{
Serial.println(&quot;I am in send&quot;);
mySerial.println(&quot;AT+CMGF=1&quot;); //Sets the GSM Module in Text Mode
delay(1000); // Delay of 1000 milli seconds or 1 second
mySerial.println(&quot;AT+CMGS=\&quot;+91xxxxxxxxxx\&quot;\r&quot;); // Replace x
with mobile number
delay(1000);
mySerial.println(&quot;Excess Gas Detected.&quot;);// The SMS text you want to send
mySerial.println(data);
delay(100);
mySerial.println((char)26);// ASCII code of CTRL+Z
delay(1000);
}
```