

SPRINT -2

| | |
|--------------|--|
| Date | 9 NOV 2022 |
| Team ID | PNT2022TMID35909 |
| Project Name | GAS LEAKAGE MONITORING AND ALERTING SYSTEMS FOR INDUSTRIES |

SIMULATION:

```
#include <LiquidCrystal.h>
#include <WiFi.h> //library for wifi
#include <PubSubClient.h>
#include "DHTesp.h"
#define BUZZER_PIN 19

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

#define ORG "cknbr5" //IBM ORGANITION ID
#define DEVICE_TYPE "123" //Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "252725" //Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "27252527"      //Token
String data3;
int trigger;
float h, t;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; //
Server Name

char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and
type of event perform and format in which data to be send

char subscribetopic[] = "iot-2/cmd/test/fmt/String"; //
cmd REPRESENT command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth"; // authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client
id

const int DHT_PIN = 25;
DHTesp dhtSensor;
LiquidCrystal lcd(4,15,5,18,21,22);
int ThreshHold = 60;
```

```
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);

void setup() {
  Serial.begin(9600);
  dhtSensor.setup(DHT_PIN, DHTesp::DHT22);
  lcd.begin(16,2);
  pinMode(BUZZER_PIN, OUTPUT);
  wificonnect();
  mqttconnect();
}

void loop() {
  delay(2000);
  TempAndHumidity data = dhtSensor.getTempAndHumidity();
  Serial.println("Temperature: " + String(data.temperature, 2) +
"°C");
  Serial.println("Humidity: " + String(data.humidity, 1) + "%");

  float gassensor=random(0,100);
  Serial.print(F("Gas Concentration: "));
  Serial.println(gassensor);

  if (gassensor>ThreshHold)
  {
    trigger=1;
    Serial.println(F("GAS LEAKED ALERT!"));
    Serial.println();
    lcd.clear();
    lcd.print ("GAS LEAKAGE :(");
    tone(BUZZER_PIN,31);
    delay (1000);
    lcd.clear();
    lcd.print ("ALERT!!!!");
    delay(1000);
    noTone(BUZZER_PIN);

  }
  else
  {
    trigger=0;
    Serial.println(F("SAFE!"));
    Serial.println();
    lcd.clear();
    lcd.print ("ALL GOOD :)");
    delay(1000);
  }
}
```

```

        lcd.clear();
        lcd.print ("SAFE!");
        delay(1000);
    }

PublishData(data.temperature,data.humidity,gassensor,trigger);
delay(1000);
if (!client.loop()) {
    mqttconnect();
}

/*
.....retrieving to
Cloud.....*/
void PublishData(float temp, float humid , float sensorvalue ,int trigger) {
    mqttconnect();//function call for connecting to ibm
/*
    creating the String in in form JSON to update the data to ibm
cloud
*/
    String payload = "{\"Temperature\":";
    payload += temp;
    payload += "," +"\"Humidity\":";
    payload += humid;
    payload += "," +"\"Gas Concentration\":";
    payload += sensorvalue;
    payload += "," +"\"Status\":";
    payload += trigger;
    payload += "}";
}

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data
on the cloud then it will print publish ok in Serial monitor or
else it will print publish failed
} 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() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6); //passing the wifi credentials
    to establish the connection
    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");
    }
}

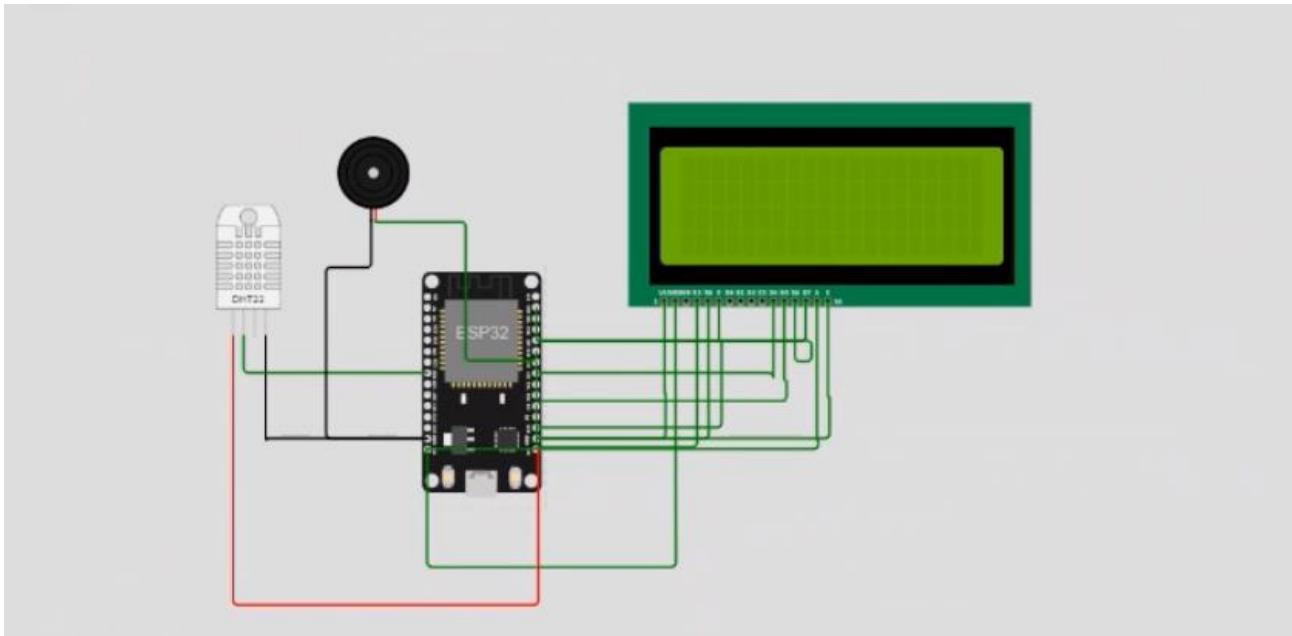
void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength)
{
    Serial.print("callback invoked for topic: ");
    Serial.println(subscribetopic);
    for (int i = 0; i < payloadLength; i++) {

        data3 += (char)payload[i];
    }
}

```

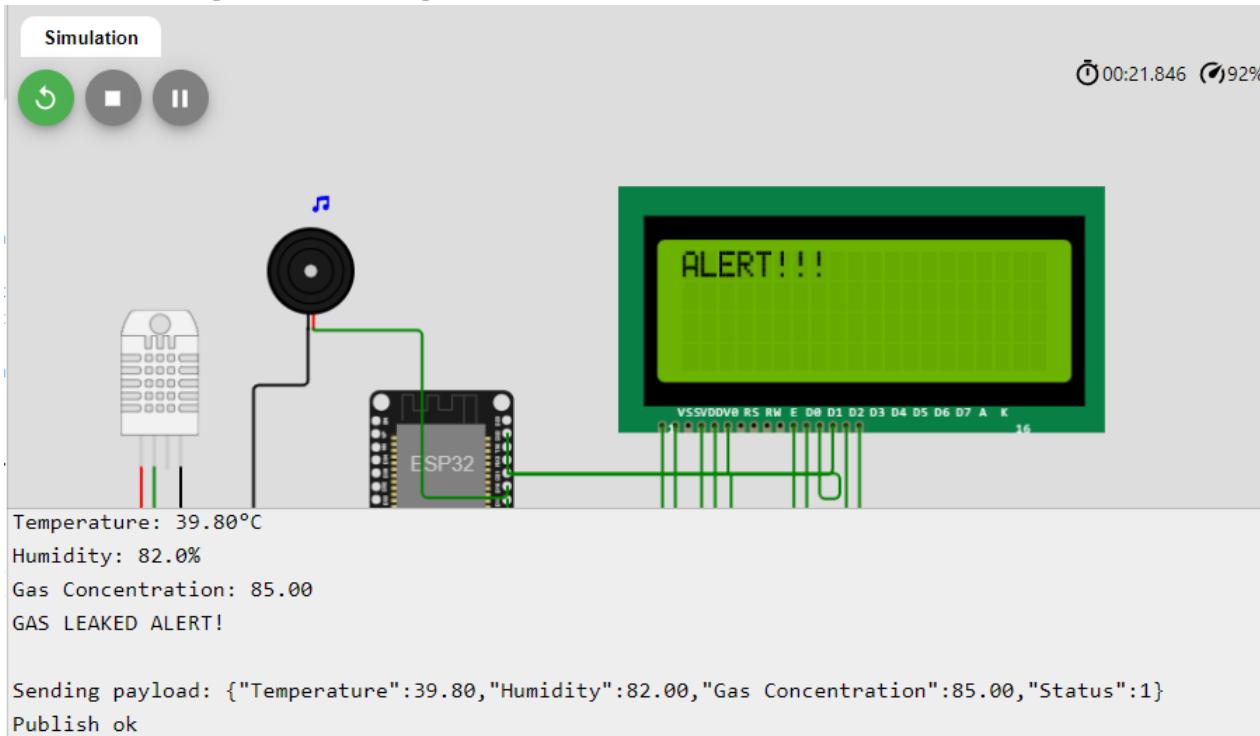
}

CIRCUIT DIAGRAM:

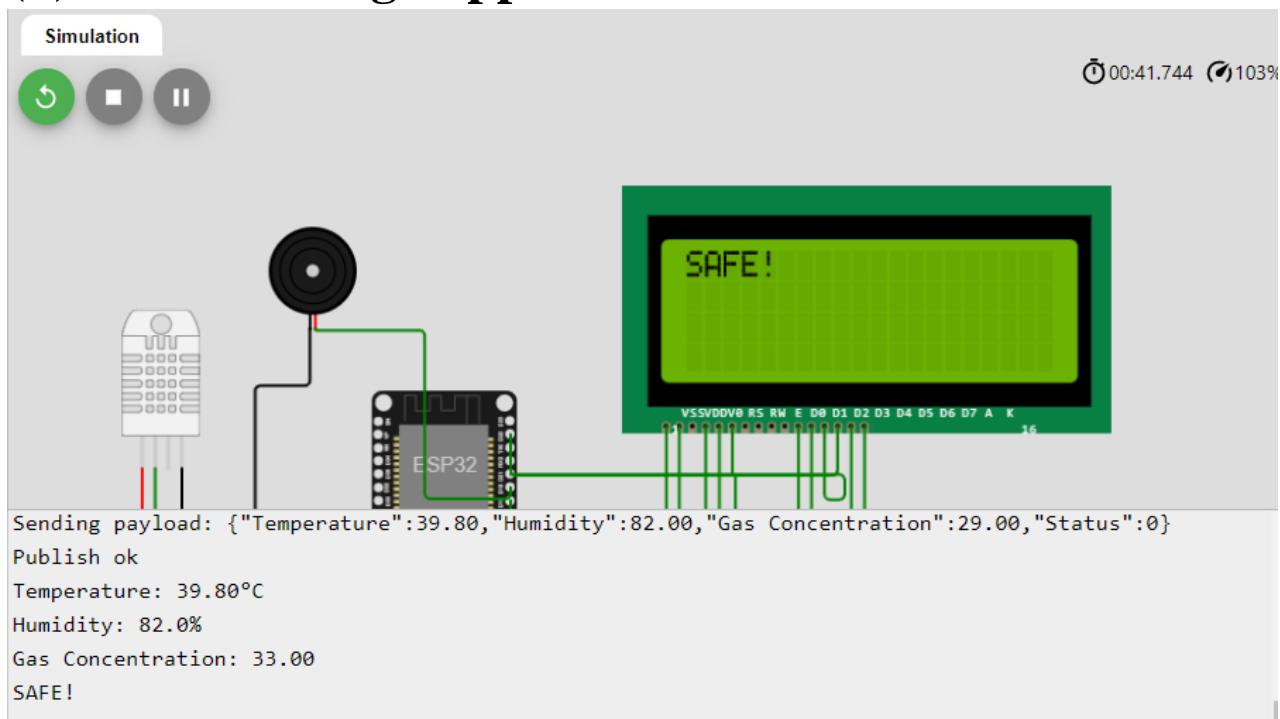


OUTPUT:

(i) When gas leakage occurs:



(ii) When nothing happens:



IBM DEVICE INFORMATION:

Device Drilldown - 252725

| | | Connection Information |
|------------------------|--|---|
| Connection Information | | Basic connection information about this device. |
| Recent Events | Device ID | 252725 |
| State | Device Type | 123 |
| Device Information | Date Added | Oct 6, 2022 10:31 PM |
| Metadata | Added By | 2019504575@student.annauniv.edu |
| Diagnostics | Connection Status | Connected |
| Connection Logs | Connection Time: Nov 9, 2022 11:37 PM Client Address: 50.31.197.64 Insecure | |
| Device Actions | | |

OUTPUT IN IBM CLOUD:

The screenshot shows the IBM Cloud Device Management interface. At the top, there are tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. On the right, there is a blue button labeled 'Add Device' with a plus sign. Below the tabs, a device card is displayed with the ID '252725', status 'Disconnected', and serial number '123'. The card also shows the last update time 'Oct 6, 2022 10:31 PM' and a three-dot menu icon. The card has tabs for 'Identity', 'Device Information', 'Recent Events' (which is selected), 'State', and 'Logs'. A note below the tabs states: 'The recent events listed show the live stream of data that is coming and going from this device.' A table below lists five recent events, all of which are 'Data' type and formatted as JSON. Each event includes its value, format (json), and the time it was received (a few seconds ago).

| Event | Value | Format | Last Received |
|-------|--|--------|-------------------|
| Data | {"Temperature":39.8,"Humidity":82,"Gas Concentration":10,"Status":0} | json | a few seconds ago |
| Data | {"Temperature":39.8,"Humidity":82,"Gas Concentration":10,"Status":0} | json | a few seconds ago |
| Data | {"Temperature":39.8,"Humidity":82,"Gas Concentration":10,"Status":0} | json | a few seconds ago |
| Data | {"Temperature":39.8,"Humidity":82,"Gas Concentration":10,"Status":0} | json | a few seconds ago |
| Data | {"Temperature":39.8,"Humidity":71.5,"Gas Concentration":10,"Status":0} | json | a few seconds ago |

Event Payload

This screenshot shows the details of a specific event payload. It includes fields for 'Event Name' (Data) and 'Time Received' (Nov 9, 2022 8:00 PM). Below these, a code editor displays the JSON data for the event. The JSON object contains five properties: Temperature, Humidity, Gas Concentration, Status, and a timestamp. The Gas Concentration value is highlighted in blue.

```
1 | {
2 |   "Temperature": 39.8,
3 |   "Humidity": 82,
4 |   "Gas Concentration": 10,
5 |   "Status": 0
6 | }
```

LINK:

<https://wokwi.com/projects/347830805902393938>