SPRINT-1

TEAM ID: PNT2022TMID29935

GAS LEAKAGE DETECTION

CODE:

```
#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>



// Watson IoT connection details

#define MQTT_HOST "x6troc.messaging.internetofthings.ibmcloud.com"
//Organization ID.messaging.internetofthings.ibmcloud.com

//change  3xr4l4

#define MQTT_PORT 1883

#define MQTT_DEVICEID "6374679606"  //d:Organization ID:Device Type:Device ID

//change  3xr4l4

#define MQTT_USER "use-token-auth"

#define MQTT_TOKEN "eSJ(wSv_kaOwuZ?yIl" // change your auth_id :

#define MQTT_TOPIC "iot-2/evt/status/fmt/json"

#define MQTT_TOPIC_DISPLAY "iot-2/cmd/display/fmt/json"


// Add WiFi connection information

char ssid[] = "raspberr";     //  your network SSID (name)

char pass[] = "dayo2022";  // your network password

// MQTT objects

void callback(char* topic, byte* payload, unsigned int length);

WiFiClient wifiClient;

PubSubClient mqtt(MQTT_HOST, MQTT_PORT, callback, wifiClient);
```

```
// variables to hold data
StaticJsonDocument<100> jsonDoc;
JsonObject payload = jsonDoc.to<JsonObject>();
JsonObject status = payload.createNestedObject("d");
static char msg[50];
float h;
void callback(char* topic, byte* payload, unsigned int length) {
  // handle message arrived
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] : ");
payload[length] = 0; // ensure valid content is zero terminated so can treat as c-string
  Serial.println((char *)payload);
}

void setup() {
  // Start serial console
  Serial.begin(115200);
  Serial.setTimeout(2000);
  while (!Serial) { }
  Serial.println();
  Serial.println("ESP8266 IBM Cloud Application");
  // Start WiFi connection
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, pass);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
```

```
 }
  Serial.println("");
  Serial.println("WiFi Connected");


  // Start connected devices
 // Connect to MQTT - IBM Watson IoT Platform
  if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN)) {
    Serial.println("MQTT Connected");
    mqtt.subscribe(MQTT_TOPIC_DISPLAY);


  } else {
    Serial.println("MQTT Failed to connect!");
    ESP.reset();
  }
}
void loop() {
  mqtt.loop();
  while (!mqtt.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (mqtt.connect(MQTT_DEVICEID, MQTT_USER, MQTT_TOKEN)) {
      Serial.println("MQTT Connected");
      mqtt.subscribe(MQTT_TOPIC_DISPLAY);
      mqtt.loop();
    }
else {
      Serial.println("MQTT Failed to connect!");
      delay(5000);
    }
```

```
 }
  h = random(1, 400);
  // uncomment this line for centigrade
  // t = dht.readTemperature(true); // uncomment this line for Fahrenheit


  // Check if any reads failed and exit early (to try again).
  if (h > 400) {
    // Send data to Watson IoT Platform
    status["gas_level"] = h;
    serializeJson(jsonDoc, msg, 50);
    Serial.println(msg);
    if (!mqtt.publish(MQTT_TOPIC, msg)) {
      Serial.println("MQTT Publish failed");
    }
  }


 // Pause - but keep polling MQTT for incoming messages
  for (int i = 0; i < 10; i++) {
    mqtt.loop();
    delay(1000);
  }
}
```

OUTPUT: