

Project development phase

Sprint - III

Date	11 November 2022
Team ID	PNT2022TMID07157
Project Name	Project - Industry-specific intelligent fire management system
Maximum Marks	20 marks

OUTPUT:

The screenshot displays the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, showing the following code:

```
1 #include <time.h>
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4
5 #define ORG "alrnx"
6 #define DEVICE_TYPE "NodeMCU"
7 #define DEVICE_ID "mcu123"
8 #define TOKEN "12345678"
9
10 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
11 char publishTopic[] = "iot-2/evt/data/fmt/json";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15
16 WiFiClient wifiClient;
17 PubSubClient client(server, 1883, wifiClient);
18
19 float temperature = 0;
20 int gas = 0;
21 int flame = 0;
22
23 String flame_status = "";
24 String Gas_status = "";
25 String exhaust_fan_status = "";
26 String sprinkler_status = "";
27
28
29 void setup() {
```

On the right, the 'Simulation' window shows a 3D model of an ESP32 board. Below the model, a log window displays the following output:

```
Publish OK
Publish OK
Publish OK
Publish OK
Publish OK
Publish OK
```

The bottom of the image shows a Windows taskbar with various application icons and a system clock indicating 12:21 on 13-11-2022.

CODE:

```
#include <time.h>
#include <WiFi.h>
#include <PubSubClient.h>
#define ORG "alrorx"
#define DEVICE_TYPE "NodeMCU"
#define DEVICE_ID "mcu123"
#define TOKEN "12345678"
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/data/fmt/json";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, wifiClient);
float temperature = 0;
int gas = 0;
int flame = 0;
String flame_status = "";
String Gas_status = "";
String exhaust_fan_status = "";
String sprinkler_status = "";
void setup() {
  Serial.begin(99900);
  wifiConnect();
  mqttConnect();
}

void loop() {

  srand(time(0));

  //initial variables and random generated data

  temperature = random(-20,125);
  gas = random(0,1000);
  int flamereading = random(200,1024);
  flame = map(flamereading,200,1024,0,2);

  //set a flame status
```

```

switch (flame) {
case 0:
    flame_status = "No Fire";
    break;
case 1:
    flame_status = "Fire is Detected";
    break;
}

//send the sprinkler status

if(flame==1){
    sprinkler_status = "Working";
}
else{
    sprinkler_status = "Not Working";
}

//toggle the fan according to gas reading

if(gas > 100){
    Gas_status = "Gas Leakage is Detected";
    exhaust_fan_status = "Working";
}
else{
    Gas_status = "No Gas Leakage is Detected";
    exhaust_fan_status = "Not Working";
}

//json format for IBM Watson

String payload = "{";
payload+="\"gas\":";
payload+=gas;
payload+=",";
payload+="\"temperature\":";
payload+=(int)temperature;
payload+=",";
payload+="\"flame\":";
payload+=flamereading;

```

```
payload+=",";
payload+="\"fire_status\":"\""+flame_status+"\", ";
payload+="\"sprinkler_status\":"\""+sprinkler_status+"\", ";
payload+="\"Gas_status\":"\""+Gas_status+"\", ";
payload+="\"exhaust_fan_status\":"\""+exhaust_fan_status+"\"}";
```

```
if(client.publish(publishTopic, (char*) payload.c_str()))
{
    Serial.println("Publish OK");
}
else{
    Serial.println("Publish failed");
}
delay(1000);
```

```
if (!client.loop())
{
    mqttConnect();
}
}
```

```
void wifiConnect()
{
    Serial.print("Connecting to ");
    Serial.print("Wifi");
    WiFi.begin("Wokwi-GUEST", "", 6);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.print("WiFi connected, IP address: ");
    Serial.println(WiFi.localIP());
}
```

```
void mqttConnect()
{
    if (!client.connected())
    {
```

```
Serial.print("Reconnecting MQTT client to ");
Serial.println(server);
while (!client.connect(clientId, authMethod, token))
{
    Serial.print(".");
    delay(500);
}

Serial.println();
}
}
```