

Project Development Phase Sprint-3

Date	12 November 2022
Team ID	PNT2022TMID29941
Project Name	Project - Industry-Specific Intelligent Fire Management System
Maximum Marks	20 Marks
IBM ID :	IBM-Project-31889-1660205917

US - 3 Develop a python script to publish random sensor data such as temperature, Flame level and Gas level to the IBM IoT platform

Code:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#include <ESP32Servo.h>
#define SERVO_PIN 26
#define BUZZER_PIN 2
//DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and typr of dht
connected
Servo servoMotor;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "6hr21b" //IBM ORGANITION ID
#define DEVICE_TYPE "sprint003" //Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "sprint03" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1234567890" //Token
String data3;
//float h, t;\
float flamelevel = 0;
const int firingLow = 70; // lowest reading for actively firing
const int firingHigh = 90; // reading for full firing
String data;
//----- Customise the above values -----
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/command/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
//-----
WiFiClient wifiClient; // creating the instance for wificlient
```

```

PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client
id by passing parameter like server id,portand wificredential
void setup()// configuring the ESP32
{
  Serial.begin(115200);
  pinMode(BUZZER_PIN, OUTPUT);
  servoMotor.attach(SERVO_PIN);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}
void loop()// Recursive Function
{
  float analogValue = analogRead(36);
  float flamelevel;
  Serial.print("Sensor RAW: ");
  Serial.println(analogValue, 0);
  flamelevel = map(analogValue, 0, 1024, 100, 0);
  Serial.print(flamelevel, 0);
  Serial.println("%");
  if (flamelevel >= firingHigh) { // stoker is fully firing
    tone(BUZZER_PIN,2000);
    servoMotor.write(180);
    delay(300);
    data="alert";
  }
  if (flamelevel < firingLow) { // fire out
    data="chill";
    noTone(BUZZER_PIN);
    servoMotor.write(0);
    // send alert
  }
  PublishData(flamelevel);
  delay(1000);
  if (!client.loop()) {
    mqttconnect();
  }
}
/* .....retrieving to Cloud.....*/
void PublishData(float flamelevel) {
  mqttconnect();//function call for connecting to ibm
/*
creating the String in in form JSon to update the data to ibm cloud
*/
  String payload = "{\"flamelevel\":";
  payload += flamelevel;
  payload += "," " \"msg\":";
  payload += data;
  payload += "\"}";
  Serial.print("Sending payload: ");

```

```

Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");// if it successfully 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 definition 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++) {
//Serial.print((char)payload[i]);
data3 += (char)payload[i];
}
}

```

```

/*Serial.println("data: "+ data3);
if(data3=="lighton")
{
Serial.println(data3);
digitalWrite(LED,HIGH);
}
else
{
Serial.println(data3);
digitalWrite(LED,LOW);
}
data3=""; */
}

```

Daigram json:

```

{
  "version": 1,
  "author": "Divya selvakumar",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -123.2, "left": -328.77, "attrs": {} },
    {
      "type": "wokwi-photoresistor-sensor",
      "id": "ldr1",
      "top": 137.8,
      "left": -96.65,
      "attrs": {}
    },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": -100.24,
      "left": -118.05,
      "attrs": { "volume": "0.1" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ldr1:VCC", "esp:3V3", "red", [ "h0" ] ],
    [ "ldr1:AO", "esp:VP", "green", [ "h0" ] ],
    [ "esp:GND.1", "ldr1:GND", "black", [ "h0" ] ],
    [ "bz1:2", "esp:3V3", "green", [ "v0" ] ],
    [ "bz1:1", "esp:D2", "green", [ "v0" ] ]
  ]
}

```

Diagram & simulation:

The WOKWI simulation interface displays a circuit diagram on the right and the corresponding Arduino code on the left. The circuit includes an ESP32 microcontroller, a servo motor, and a buzzer. The code is for a servo motor that can be controlled via MQTT. It includes comments for configuring the device type, ID, and token for the IBM Watson IoT Platform.

```
1 #include <WiFi.h> //Library for wifi
2 #include <PubSubClient.h> //Library for MQTT
3 #include <ESP32Servo.h>
4 #define SERVO_PIN 26
5 #define BUZZER_PIN 2
6 //DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connected
7 Servo servoMotor;
8 void callback(char* topic, byte* payload, unsigned int payloadLength);
9 //-----Credentials of IBM Accounts-----
10 #define ORG "6hr21b" //IBM ORGANIZATION ID
11 #define DEVICE_TYPE "sprint003" //Device type mentioned in ibm watson IOT Platform
12 #define DEVICE_ID "sprint03" //Device ID mentioned in ibm watson IOT Platform
13 #define TOKEN "1234567890" //Token
14 String data3;
15 //float h, t;
16 float flameLevel = 0;
17 const int firingLow = 70; // lowest reading for actively firing
18 const int firingHigh = 90; // reading for full firing
19 String data;
20 //----- Customise the above values -----
21 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
22 char publishTopic[] = "iot-2/evt/data/rmt/json"; // topic name and type of event perform and format
23 char subscribeTopic[] = "iot-2/cmd/command/rmt/String"; // cmd REPRESENT command type AND COMMAND I
24 char authMethod[] = "use-token-auth"; // authentication method
25 char token[] = TOKEN;
26 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
27 //-----
28 WiFiClient wifiClient; // creating the instance for wifiClient
29 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client id by pas
30 void setup() // configuring the ESP32
31 {
32   Serial.begin(115200);
33   pinMode(BUZZER_PIN, OUTPUT);
34   servoMotor.attach(SERVO_PIN);
35   delay(10);
36   Serial.println();
37   wifiConnect();
38   mqttConnect();
39 }
```

The simulation window shows the circuit diagram with an ESP32 microcontroller connected to a servo motor and a buzzer. The status bar indicates the simulation is running at 100% zoom.

Simulation Log:

```
3% Sending payload: {"flameLevel":3.00,"msg":"chill"}
Publish ok
Sensor RAW: 1001
3% Sending payload: {"flameLevel":3.00,"msg":"chill"}
Publish ok
```

ibm cloud connection:

The IBM Watson IoT Platform dashboard displays a list of devices. The device 'sprint03' is shown as 'Connected'. The dashboard includes a sidebar with navigation icons and a top bar with user information.

Device ID	Status	Device Type	Class ID	Date Added
mainprj	Disconnected	sprint004	Device	Nov 13, 2022 12:28 PM
sprint03	Connected	sprint003	Device	Nov 13, 2022 12:38 PM

Details for 'sprint03':

Identity	Device Information	Recent Events	State	Logs
Device ID	sprint03			
Device Type	sprint003			
Date Added	Nov 13, 2022 12:38 PM			
Added By	kiruthikas028.ece@dgct.ac.in			
Connection Status	Connected			
	Connection Time: Nov 13, 2022 12:47 PM			
	Client Address: 50.31.197.64 Insecure			

Items per page 50 | 1-2 of 2 items

1 of 1 page

IBM Watson iot platform cloud output:

The screenshot displays the IBM Watson IoT Platform interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A user profile is visible in the top right corner with the email 'kiruthikas028.ece@dgct.ac.in' and ID '6hr21b'. The left sidebar contains various icons for navigation. The main content area shows a modal window for a device named 'sprinkler', which is in a 'Connected' state. The 'Recent Events' tab is active, displaying a table of events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. It lists three data events, each with a JSON value: {"flamelevel":3,"msg":"chill"}. The 'Last Received' column indicates 'a few seconds ago' for each event. At the bottom of the modal, there is a pagination control showing 'Items per page 50' and '1-2 of 2 items'.

IBM Watson IoT Platform

Kiruthikas028.ece@dgct.ac.in
ID: 6hr21b

Browse Action Device Types Interfaces

Add Device

Identity Device Information Recent Events State Logs

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

Event	Value	Format	Last Received
Data	{"flamelevel":3,"msg":"chill"}	json	a few seconds ago
Data	{"flamelevel":3,"msg":"chill"}	json	a few seconds ago
Data	{"flamelevel":3,"msg":"chill"}	json	a few seconds ago

Items per page 50 | 1-2 of 2 items

1 of 1 page