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
#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()// configureing 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</pre>
data="chill";
noTone(BUZZER_PIN);
servoMotor.write(0);
// send alert
}
PublishData(flamelevel);
delay(1000);
if (!client.loop()) {
mqttconnect();
}
}
/*.....*/
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: ");
```

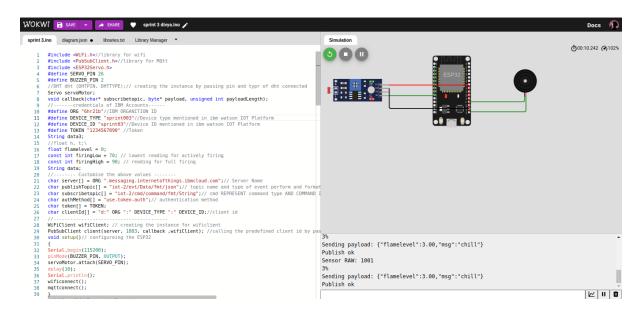
```
Serial.println(payload);
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial println("Publish ok");// if it sucessfully upload data on the cloud then it will
print publish ok in Serial monitor or else it will print publish failed
} else {
Serial.println("Publish failed");
}
}
void 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; <math>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=""; */
}
```

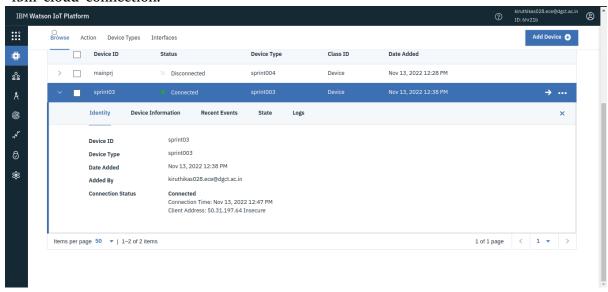
Daigram json:

```
{
"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:



ibm cloud connection:



IBM Watson iot platform cloud output:

