

# Final Deliverables

Team ID : PNT2022TMID29941

IBM ID : IBM-Project-31889-1660205917

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 type of dht
connected
Servo servoMotor;
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "6hr21b" //IBM ORGANIZATION ID
#define DEVICE_TYPE "sprint004" //Device type mentioned in IBM Watson IOT Platform
#define DEVICE_ID "mainproject" //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 wifi client
PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client id by
passing parameter like server id, port and wifi credential
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 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);

```

```

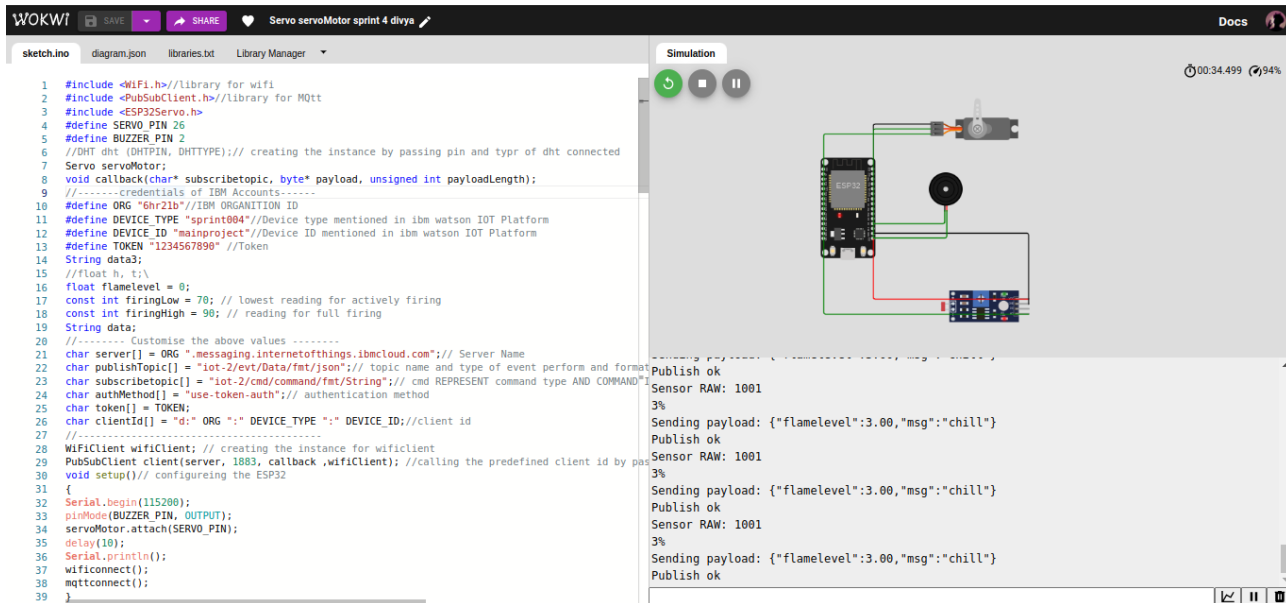
while (!client.connect(clientId, authMethod, token)) {
  Serial.print(".");
  delay(500);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect() //function defination for wificonnect
{
  Serial.println();
  Serial.print("Connecting to ");
  WiFi.begin("Wokwi-GUEST", "", 6);
  //passing the wifi 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++) {
    //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=""; */}

```

Diagram.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" }
    },
    { "type": "wokwi-servo", "id": "servo1", "top": -236.33, "left": -115.45, "attrs": {} }
  ],
  "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" ] ],
    [ "servo1:GND", "esp:GND.1", "black", [ "h0" ] ],
    [ "servo1:V+", "esp:3V3", "green", [ "h0" ] ],
    [ "servo1:PWM", "esp:D26", "green", [ "h0" ] ]
  ]
}
```

## Program&Output: Simulation:



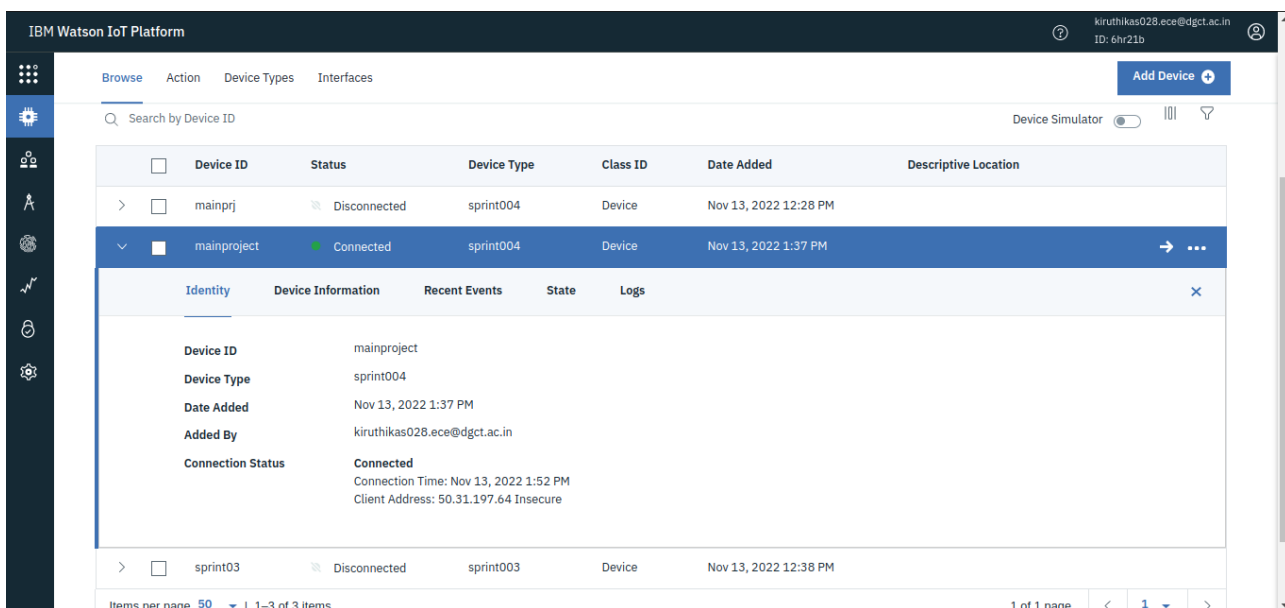
The screenshot displays the Wokwi IDE interface. On the left, the sketch editor shows the following code:

```
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 typr of dht connected
7 Servo servoMotor;
8 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
9 //-----Credentials of IBM Accounts-----
10 #define ORG "6hr21b" //IBM ORGANITION ID
11 #define DEVICE_TYPE "sprint004" //Device type mentioned in ibm watson IOT Platform
12 #define DEVICE_ID "mainproject" //Device ID mentioned in ibm watson IOT Platform
13 #define TOKEN "1234567890" //Token
14 String data;
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/fmt/json"; // topic name and type of event perform and format
23 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND COMMAND
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 passing server, port, callback, and wifiClient
30 void setup() // configuring the ESP32
31 {
32   Serial.begin(115200);
33   pinMode(BUZZER_PIN, OUTPUT);
34   servoMotor.attach(SERVO_PIN);
35   delay(100);
36   Serial.println();
37   wifiConnect();
38   mqttconnect();
39 }
```

On the right, the simulation window shows a visual representation of the ESP32 board connected to a servo motor and a buzzer. Below the simulation, the output console displays the following log:

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

## Ibm watson iot platform connection:



The screenshot shows the IBM Watson IoT Platform dashboard. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present with the text 'Search by Device ID'. The main content area displays a table of devices:

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location
mainprj	Disconnected	sprint004	Device	Nov 13, 2022 12:28 PM	
mainproject	Connected	sprint004	Device	Nov 13, 2022 1:37 PM	
sprint03	Disconnected	sprint003	Device	Nov 13, 2022 12:38 PM	

The 'mainproject' device is selected, and its details are shown in a modal window:

- Identity**: Device ID: mainproject
- Device Information**: Device Type: sprint004
- Recent Events**: Date Added: Nov 13, 2022 1:37 PM
- State**: Added By: kiruthikas028.ece@dgct.ac.in
- Logs**: Connection Status: **Connected**  
Connection Time: Nov 13, 2022 1:52 PM  
Client Address: 50.31.197.64 Insecure

The bottom of the dashboard shows 'Items per page 50' and '1-3 of 3 items'.

Watson output event:

[illegible]