

Sprint 4

Code for Simulation:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include <LiquidCrystal_I2C.h>
#include "DHT.h"// Library for dht11
#define DHTPIN 15      // what pin we're connected to
#define DHTTYPE DHT11  // define type of sensor DHT 11
#define LED 2
DHT dht (DHTPIN, DHTTYPE);// creating the instance by passing pin and type of
dht connected
void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);

//-----credentials of IBM Accounts-----

#define ORG "9a7os9"//IBM ORGANIZATION ID
#define DEVICE_TYPE "ESP"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "ESP32"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "LC!x?+V9etumdVMaSR"      //Token
String data3="";
int buzz= 13;

//----- 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 subscribtopic[] = "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
LiquidCrystal_I2C lcd(0x27,16,2);

//-----
```

```

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(LED,OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{
    if (!client.loop()) {
        mqttconnect();
    }
}

/*.....retrieving to
Cloud.....*/

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("Please take "+ data3);
    if(data3 != "")
    {
        lcd.init();
        lcd.print("Take"+ data3);

        digitalWrite(LED,HIGH);
        delay(20000);
        digitalWrite(LED,LOW);

    }

    else
    {
        digitalWrite(LED,LOW);

    }
    data3="";

}

```


OUTPUT :

The screenshot displays the Wokwi IoT simulator interface. On the left, the code for `esp32-dht22.ino` is shown, which includes libraries for WiFi, MQTT, and DHT11. It defines the DHT11 sensor pin (15) and type (DHT11), and sets up an ESP32 device connected to an IBM Watson IoT platform. The code includes credentials for the IBM account and configures the device type and ID. The right side of the interface shows a simulation of the hardware setup, including an ESP32, a DHT11 sensor, and an LCD display. The LCD display shows the text "Takedolo 650". Below the simulation, the console output shows the device subscribing to the topic `iot-2/cmd/command/fmt/String` and receiving a command `iot-2/cmd/command/fmt/String` with the payload `Please take dolo 650`.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include <LiquidCrystal_I2C.h>
4 #include <DHT.h> // library for dht11
5 #define DHTPIN 15 // what pin we're connected to
6 #define DHTTYPE DHT11 // define type of sensor DHT 11
7 #define LED 2
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
9 void callback(char* topic, byte* payload, unsigned int payloadLength)
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "9a7os9" //IBM ORGANITION ID
14 #define DEVICE_TYPE "ESP" //Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "ESP32" //Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "tCkx?V0etumdVhaSR" //Token
17 String data3="";
18 int buzz= 13;
19
20 //----- Customise the above values -----
21
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
23 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
24 char subscribeTopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
28 LiquidCrystal_I2C lcd(0x27,16,2);
29
```

Simulation

iot-2/cmd/command/fmt/String
subscribe to cmd OK
callback invoked for topic: iot-2/cmd/command/fmt/String
Please take dolo 650
callback invoked for topic: iot-2/cmd/command/fmt/String
Please take dolo 650

