

SPRINT 2

Date	7 November 2022
Project Name	Smart Farmer - IOTEnabled Smart Farming Application

Simulation:

Sending temperature and humidity values from Wokwi to IBM Watson.

Program:

```
#include <WiFi.h> //library for wifi
#include <PubSubClient.h> //library for MQTT
#include "DHT.h" // Library for dht11
#define DHTPIN 15 // what pin we're connected to
#define DHTTYPE DHT22 // define type of sensor DHT 11
#define LED 5
DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and typr of
dht connected
void callback(char* subscribetopic, byte* payload, unsigned
int payloadLength);

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

#define ORG "93oivx" //IBM ORGANITION ID
#define DEVICE_TYPE "NodeMCU" //Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "12345" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3; float h, t;

//----- 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);
    dht.begin();
```

```

pinMode(LED,OUTPUT);
delay(10);    Serial.println();
wificonnect();
mqttconnect();
} void loop()// Recursive
Function
{    h = dht.readHumidity();
t = dht.readTemperature();
    Serial.print("temperature:");
    Serial.println(t);
    Serial.print("Humidity:");
Serial.println(h);
    PublishData(t, h);
delay(1000);    if
(!client.loop()) {
mqttconnect();
    }
}

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

void PublishData(float temp, float humid) {
mqttconnect();//function call for connecting to ibm
/*
    creating the String in in form JSon to update the data to ibm cloud
*/
    String payload = "{\"temperature\":";
payload += temp;    payload += ","
    "\"humidity\":";    payload += humid;
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="";

```

```

}

```

OUTPUT:

The screenshot shows the WOKWI simulation interface. On the left, the sketch code is displayed, which includes the necessary libraries, defines the DHT22 sensor pin and type, and sets up an MQTT client to send data to a specific topic. The code also includes a callback function to handle incoming data. On the right, the simulation shows an ESP32 microcontroller connected to a DHT22 sensor. The output window displays the sensor readings: temperature: 51.40 and humidity: 86.00.

The screenshot shows the IBM Watson IoT Platform dashboard. The device with ID 12345 is listed as connected. The 'Recent Events' tab is selected, showing a stream of data from the device. The table below lists the recent events, including the event type, value, format, and last received time.

Event	Value	Format	Last Received
IoTSensor	{"d":{"temp":51.4,"humidity":86}}	json	a few seconds ago
IoTSensor	{"d":{"temp":51.4,"humidity":86}}	json	a few seconds ago
IoTSensor	{"d":{"temp":51.4,"humidity":86}}	json	a few seconds ago
IoTSensor	{"d":{"temp":51.4,"humidity":86}}	json	a few seconds ago
IoTSensor	{"d":{"temp":51.4,"humidity":86}}	json	a few seconds ago

