

# SPRINT 2

Date	7 November 2022
Team ID	PNT2022TMID13187
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 WOKWI simulation environment displays an ESP32 microcontroller connected to a DHT22 sensor. The sketch code on the left configures the ESP32 as a NodeMCU device, sets up the DHT22 sensor, and publishes temperature and humidity data to an IBM Watson IoT Platform topic. The simulation results on the right show the device sending payloads containing temperature and humidity data.

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 5
7 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of dht connect
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadlength);
10
11 //-----credentials of IBM Accounts-----
12
13 #define ORG "93oivx" //IBM ORGANIZATION ID
14 #define DEVICE_TYPE "NodeMCU" //Device type mentioned in ibm watson IOT Platform
15 #define DEVICE_ID "12345" //Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "12345678" //Token
17 String data3;
18 float h, t;
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 perform a
24 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command type AND
25 char authMethod[] = "use-token-auth"; // authentication method
26 char token[] = TOKEN;
27 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
28
29 //-----
30
31 WiFiClient wifiClient; // creating the instance for wifiClient
32 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
33 void setup() // configuring the ESP32

```

Simulation Results:

```

Sending payload: {"temperature":51.40,"humidity":86.00}
temperature:51.40
Humidity:86.00
Sending payload: {"temperature":51.40,"humidity":86.00}
temperature:51.40
Humidity:86.00
Sending payload: {"temperature":51.40,"humidity":86.00}

```

The IBM Watson IoT Platform dashboard shows a list of devices. The device with ID 12345 is connected and is a NodeMCU type. The detailed view of this device shows recent events, which are JSON payloads containing temperature and humidity data.

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

