

Sprint Delivery – 4

Date	17 November 2022
Team ID	PNT2022TMID28572
Project Name	Hazardous Area Monitoring for Industrial Power plant powered by IoT
Maximum Marks	4 Marks

WOKWI CODE:

```
#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 2

DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
dhtconnected void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----

#define ORG "iagqzu" //IBM ORGANITION ID
#define DEVICE_TYPE "Deepak" //Device type mentioned in ibm watson IOT
Platform#define DEVICE_ID "123" //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 formatin
which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT command
typeAND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth"; // authentication method
chartoken[] = 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 predefinedclient
id by passing parameter like server id,portand wificredential
```

```

void setup()// configuring the ESP32
{
    Serial.begin(
    115200);
    dht.begin();
    pinMode(LED,OUTPUT);
    delay(10);
    Serial.println();

    wifi
    connect()
    ;
    mqtt
    connect()
    ;
} void loop()//
RecursiveFunction
{    h =
dht.readHumidity();t
=
dht.readTemperature
();
    Serial.print("tem
p:");
    Serial.println(t);
    Serial.print("Hu
mid:");
    Serial.println(h);
    PublishData(t, h);

    delay(10
00);
    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 =
    "{\"temp\":";payload +=
temp;      payload += ","
    "\"Humid\":";
            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 itwill
        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
forwificonnect
{
    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(L
ED,LOW);
} data3="";
}

```

WOKWI OUTPUT:

The screenshot shows the WOKWI IDE interface. On the left, the 'sketch.ino' file is open, displaying C++ code for an ESP32 connected to a DHT22 sensor and an LED. The code includes libraries for WiFi, PubSubClient, and DHT. It defines pins for DHT (15) and LED (2). The main logic involves creating a DHT instance, connecting to an MQTT server (IBM Watson IoT Platform), and publishing temperature and humidity data. On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 module, a red LED, and a DHT22 sensor. Below the simulation, the console output shows the device sending payloads: {"temp":24.00,"Humid":40.00} and {"temp":24.00,"Humid":40.00}, with status messages like 'Publish ok'.

IBM WATSON PLATFORM DEVICE EVENT LOG:

The screenshot shows the IBM Watson Platform Device Event Log interface. The top navigation bar includes 'Browse', 'Action', 'Device Types', and 'Interfaces'. A search bar is present, and a 'Device Simulator' toggle is visible. The main table lists devices, with the selected device '123' showing a 'Connected' status. Below the table, the 'Recent Events' tab is active, displaying a list of events. The events are categorized by 'Event' (IoT Sensor), 'Value' (JSON payload), 'Format' (json), and 'Last Received' (a few seconds ago).

Event	Value	Format	Last Received
IoT Sensor	{"temp":32,"Humid":33}	json	a few seconds ago
IoT Sensor	{"temp":58,"Humid":87}	json	a few seconds ago
IoT Sensor	{"temp":35,"Humid":6}	json	a few seconds ago
IoT Sensor	{"temp":86,"Humid":24}	json	a few seconds ago

DEVICE EVENT PAYLOAD:

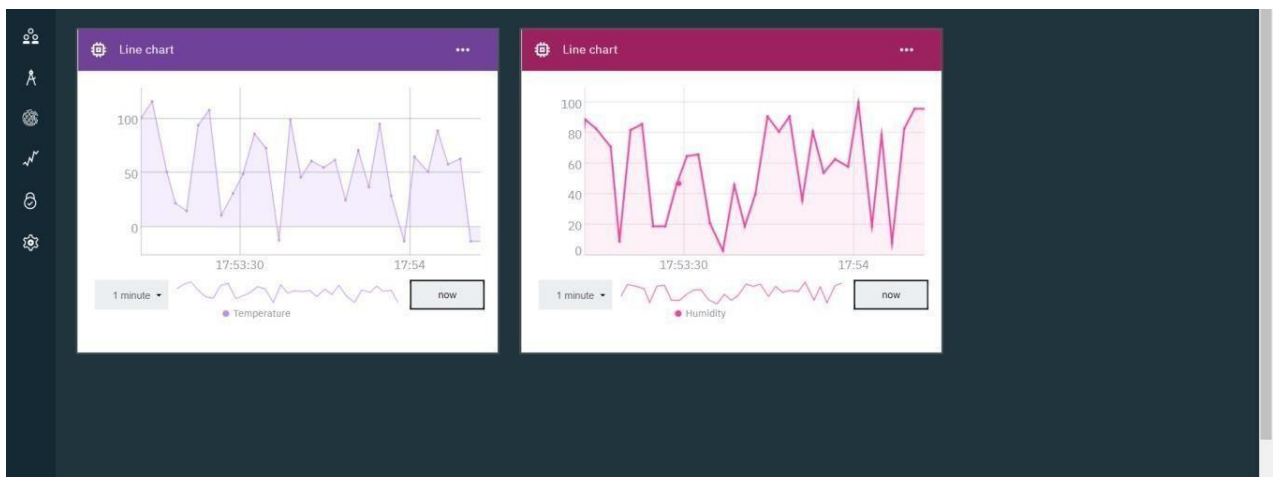
The screenshot displays a web application for managing IoT devices. A modal window titled "Event Payload" is open, showing details for an event from an "IoTSensor" received on "Nov 14, 2022 11:31 AM". The payload is a JSON object:

```
{ 1: { 2: { 3: "temp": 32, 4: "humid": 33 } }
```

. In the background, a table lists recent events for device ID 123, with columns for "Event" and "Value".

Event	Value
IoTSensor	{ "temp": 32, "humid": 33 }
IoTSensor	{ "temp": 58, "humid": 33 }
IoTSensor	{ "temp": 35, "humid": 33 }
IoTSensor	{ "temp": 86, "humid": 33 }
IoTSensor	{ "temp": 37, "humid": 33 }

DEVICE- BOARD:



IBM CLOUDANT DB LOG:

← noderedtglm202...

Document ID

Options

{ } JSON

All Documents

Query

Permissions

Changes

Design Documents

library

Table

Metadata

{ } JSON

Create Document

	_id	humidity	temperature
<input type="checkbox"/>	0096ab1244940360661f0bce73051181	9	79
<input type="checkbox"/>	0096ab1244940360661f0bce730520d0	68	122
<input type="checkbox"/>	0096ab1244940360661f0bce730d0d19	6	109
<input type="checkbox"/>	0096ab1244940360661f0bce730d1a9b	72	39
<input type="checkbox"/>	0096ab1244940360661f0bce730d380a	44	105
<input type="checkbox"/>	0096ab1244940360661f0bce731556d4	37	12
<input type="checkbox"/>	0096ab1244940360661f0bce73156b2b	18	-5
<input type="checkbox"/>	0096ab1244940360661f0bce731b7048	81	5
<input type="checkbox"/>	0096ab1244940360661f0bce731bbf33	25	90
<input type="checkbox"/>	0096ab1244940360661f0bce7320722f	87	11
<input type="checkbox"/>	0096ab1244940360661f0bce73207f8c	48	49
<input type="checkbox"/>	0096ab1244940360661f0bce7325d136	56	107

Showing 3 of 4 columns. ☐ Show all columns.

Showing document 1 - 20. Documents per page: 20

Log Out.