

# SPRINT - 03

Date :	07 November 2022
Team ID :	PNT2022TMID37707
Project Name	SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY

## SPRINT GOAL:

Integrate the hardware to be able to access the cloud functions and provide inputs to the same.

## PROGRAM 01:

**AIM:** To find the Temperature and Humidity DHT22 and ESP32

**PLATFORM:** WOKWI

The screenshot displays the Wokwi web IDE interface. On the left, the 'sketch.ino' file is open, showing C++ code for an ESP32 microcontroller. The code includes the following components:

- Libraries:** `<WiFi.h>` for WiFi, `<PubSubClient.h>` for MQTT, and `<DHT.h>` for the DHT22 sensor.
- Constants:** `DHTPIN` is defined as 4, `DHTTYPE` as `DHT22`, and `LED` as 5.
- Instance Creation:** A `DHT` instance is created with `DHTPIN` and `DHTTYPE`.
- MQTT Configuration:** Constants for `ORG` ("7kb3es"), `DEVICE_TYPE` ("deepi\_reshpeery"), `DEVICE_ID` ("56789"), and `TOKEN` ("CUsp45PBu8EsQeJ\_9N") are defined. A `server` is set to `"messaging.internetofthings.ibmcloud.com"`.
- Client Setup:** A `wifiClient` is created, and a `PubSubClient` is initialized with the server, port 1883, and the callback function.

On the right, the 'Simulation' window shows a 3D model of the ESP32 board. It is connected to a DHT22 sensor module and a red LED with a resistor. The connections are as follows:

- ESP32 GND to DHT22 GND and LED cathode.
- ESP32 VCC to DHT22 VCC and LED anode.
- ESP32 D4 to DHT22 SDA.
- ESP32 D5 to DHT22 SCL.

The bottom of the image shows the Windows taskbar with the search bar and various application icons. The system clock indicates 6:53 AM on 11/18/2022.

WOKWI

sketch.ino

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 4 // 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
8
9 void callback(char* topic, byte* payload, unsigned int payloadLength) {
10 //-----credentials of IBM Accounts-----
11
12
13 #define ORG "7kb3es" //IBM ORGANIZATION ID
14 #define DEVICE_TYPE "deepi_resheery" //Device type mentioned in ibm watson
15 #define DEVICE_ID "56789" //Device ID mentioned in ibm watson IOT Platform
16 #define TOKEN "CUsp45PBu8EsQeJ_9N" //Token
17 String data3;
18 float h, t;
19
20
21 //----- Customise the above values -----
22 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server
23 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of
24 char subscribeTopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT coming
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 pre

```

Simulation

Editing DHT22

Temperature: 28.5°C

Humidity: 44.0%

Humidity: 91.00

Sending payload: {"temperature": 28.50, "humidity": 91.00}

Publish ok

temperature: 28.50

Humidity: 91.00

Sending payload: {"temperature": 28.50, "humidity": 91.00}

Publish ok

IBM Watson IoT Platform

410119106009@smartinternz.com  
ID: 7kb3es

Browse Action Device Types Interfaces

Add Device

Device ID	Status	Device Name	Type	Last Seen
4321	Disconnected	assign4	Device	18 Nov 2022 02:07
56789	Connected	deepi_resheery	Device	22 Oct 2022 05:28

Identity Device Information Recent Events State Logs

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"temperature": 6.1, "humidity": 81}	json	a few seconds ago
Data	{"temperature": 6.1, "humidity": 81}	json	a few seconds ago
Data	{"temperature": 26.2, "humidity": 64.5}	json	a few seconds ago
Data	{"temperature": 26.2, "humidity": 64.5}	json	a few seconds ago
Data	{"temperature": -9.2, "humidity": 64.5}	json	a few seconds ago

## PYTHON CODE:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQTT
#include "DHT.h"// Library for dht11
#define DHTPIN 4    // 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 type of dht connected

void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
\
//-----credentials of IBM Accounts-----

#define ORG "7kb3es"//IBM ORGANITION ID
#define DEVICE_TYPE "deepi_reshepeery"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "56789"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "CUsp45PBuBEsQeJ_9N" //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 subscribtopic[] = "iot-2/cmd/test/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 link :**

<https://wokwi.com/projects/346292886858367572>

By using this Wokwi we determined the Temperature and Humidity for better road safety.

## POGRAM 02

**AIM:** Write code and connection in Wowki for ultrasonic sensor. Whenever distance is less than 100 cms send “Alert” to IBM cloud and display in device recent events by using ESP32

### PLATFORM: WOKWI

Wokwi IDE interface showing the code for the ESP32 project. The code includes headers for WiFi and PubSubClient, defines credentials and device information, and sets up a callback function to send data to IBM Cloud IoT Platform. The simulation window shows the ESP32 board connected to an HC-SR04 ultrasonic sensor and an LED.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
5 \
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "7kb3es" //IBM ORGANITION ID
9 #define DEVICE_TYPE "suba" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "636903" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "987654321" //Token
12 String data3;
13
14
15 //----- Customise the above values -----
16 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
17 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform
18 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND
19 char authMethod[] = "use-token-auth"; // authentication method
20 char token[] = TOKEN;
21 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
22
23
24 //-----
25 WiFiClient wifiClient; // creating the instance for wificlient
26 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
27 const int trigpin = 5;
28 const int echopin = 18;
29 const int ledpin = 2;
30
31
32
```

Wokwi IDE interface showing the code for the ESP32 project. The code includes headers for WiFi and PubSubClient, defines credentials and device information, and sets up a callback function to send data to IBM Cloud IoT Platform. The simulation window shows the ESP32 board connected to an HC-SR04 ultrasonic sensor and an LED. A dialog box shows the distance measured by the sensor is 62cm. The console output shows the payload being sent to IBM Cloud IoT Platform.

```
1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3
4 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
5 \
6 //-----credentials of IBM Accounts-----
7
8 #define ORG "7kb3es" //IBM ORGANITION ID
9 #define DEVICE_TYPE "suba" //Device type mentioned in ibm watson IOT Platform
10 #define DEVICE_ID "636903" //Device ID mentioned in ibm watson IOT Platform
11 #define TOKEN "987654321" //Token
12 String data3;
13
14
15 //----- Customise the above values -----
16 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
17 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event perform
18 char subscribetopic[] = "iot-2/cmd/test/fmt/String"; // cmd REPRESENT command type AND
19 char authMethod[] = "use-token-auth"; // authentication method
20 char token[] = TOKEN;
21 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
22
23
24 //-----
25 WiFiClient wifiClient; // creating the instance for wificlient
26 PubSubClient client(server, 1883, callback, wifiClient); //calling the predefined client
27 const int trigpin = 5;
28 const int echopin = 18;
29 const int ledpin = 2;
30
31
32
```

Editing Ultrasonic Distance Sensor  
Distance: 62cm

Sending payload: {"ALERT...!! ": 61.97}  
Publish ok  
ALERT...!!!  
61.97  
Sending payload: {"ALERT...!! ": 61.97}  
Publish ok

The screenshot shows the IBM Watson IoT Platform interface. At the top, there's a navigation bar with tabs like 'Browse', 'Action', 'Device Types', and 'Interfaces'. A sidebar on the left contains various icons for navigation. The main content area displays details for a specific device. The device ID is '636903', it's connected, and its name is 'suba'. Below this, there's a tabbed interface with 'Identity', 'Device Information', 'Recent Events', 'State', and 'Logs'. The 'Recent Events' tab is active, showing a table of events. The table has columns for 'Event', 'Value', 'Format', and 'Last Received'. The events are JSON strings representing alerts, such as '{"ALERT...!! ":57.97}'. The bottom of the image shows a Windows taskbar with various application icons and the system clock indicating 7:03 AM on 11/18/2022.

Event	Value	Format	Last Received
Data	{"ALERT...!! ":57.97}	json	a few seconds ago
Data	{"ALERT...!! ":22.98}	json	a few seconds ago
Data	{"ALERT...!! ":3.99}	json	a few seconds ago
Data	{"ALERT...!! ":85.99}	json	a few seconds ago
Data	{"ALERT...!! ":85.99}	json	a few seconds ago

## PYHTON CODE:

```
#include <WiFi.h>//library for wifi
```

```
#include <PubSubClient.h>//library for MQTT
```

```
void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
```

```
\
```

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

```
#define ORG "7kb3es"//IBM ORGANITION ID
```

```
#define DEVICE_TYPE "suba"//Device type mentioned in ibm watson IOT Platform
```

```
#define DEVICE_ID "636903"//Device ID mentioned in ibm watson IOT Platform
```

```
#define TOKEN "987654321" //Token
```

```
String data3;
```

```
//----- 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/test/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
```

```
const int trigpin = 5;
```

```
const int echopin = 18;
```

```
const int ledpin = 2;
```

```
long duration ;
```

```
float distance;
```

```
#define sound_speed 0.034
```

```
void setup() {
```

```
    // put your setup code here, to run once:
```

```
    Serial.begin(115200);
```

```
    pinMode(trigpin, OUTPUT);
```

```
    pinMode(echopin, OUTPUT);
```

```
    pinMode(ledpin, OUTPUT);
```

```
    wificonnect();
```

```
    mqttconnect();
```

```
}
```

```
void loop() {
```

```
    digitalWrite(trigpin, LOW);
```

```
    digitalWrite(trigpin, HIGH);
```

```
    delayMicroseconds(10);
```

```
    digitalWrite(trigpin, LOW);
```

```
    duration= pulseIn(echopin,HIGH);
```

```
    distance = duration * sound_speed /2;
```

```
    if(distance<=100){
```

```
        PublishData(distance);
```

```
    delay(1000);
```

```
    if (!client.loop()) {
```

```
        mqttconnect();
```

```
    }
```

```
        digitalWrite(ledpin, HIGH);
```

```
        Serial.println("ALERT....!!!");
```

```
        Serial.println(distance);
```

```
    }
```

```
    else
```

```
    {
```

```
        digitalWrite(ledpin, LOW);
```

```
    }
```

```
    // put your main code here, to run repeatedly:
```

```
    delay(10); // this speeds up the simulation
```

```
}
```

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



```

void PublishData(float distance) {
  mqttconnect();//function call for connecting to ibm

  // creating the String in in form JSon to update the data to ibm cloud
  String payload = "{\"ALERT...!! \": ";
  payload += distance;
  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);
  }
  else
  {
    Serial.println(data3);
  }
  data3="";
}

```

## **Output link :**

<https://wokwi.com/projects/343963122450563668>

## **Conclusion:**

Here I showed the ALERT and DISTANCE in IBM cloud when vehicle has the distance is less than 100 cms

**THANK YOU...!!**