

SPRINT - 03

Date :	07 November 2022
Team ID :	PNT2022TMID47905
Project Name	SIGNS WITH SMART CONNECTIVITY FOR BETTER ROAD SAFETY

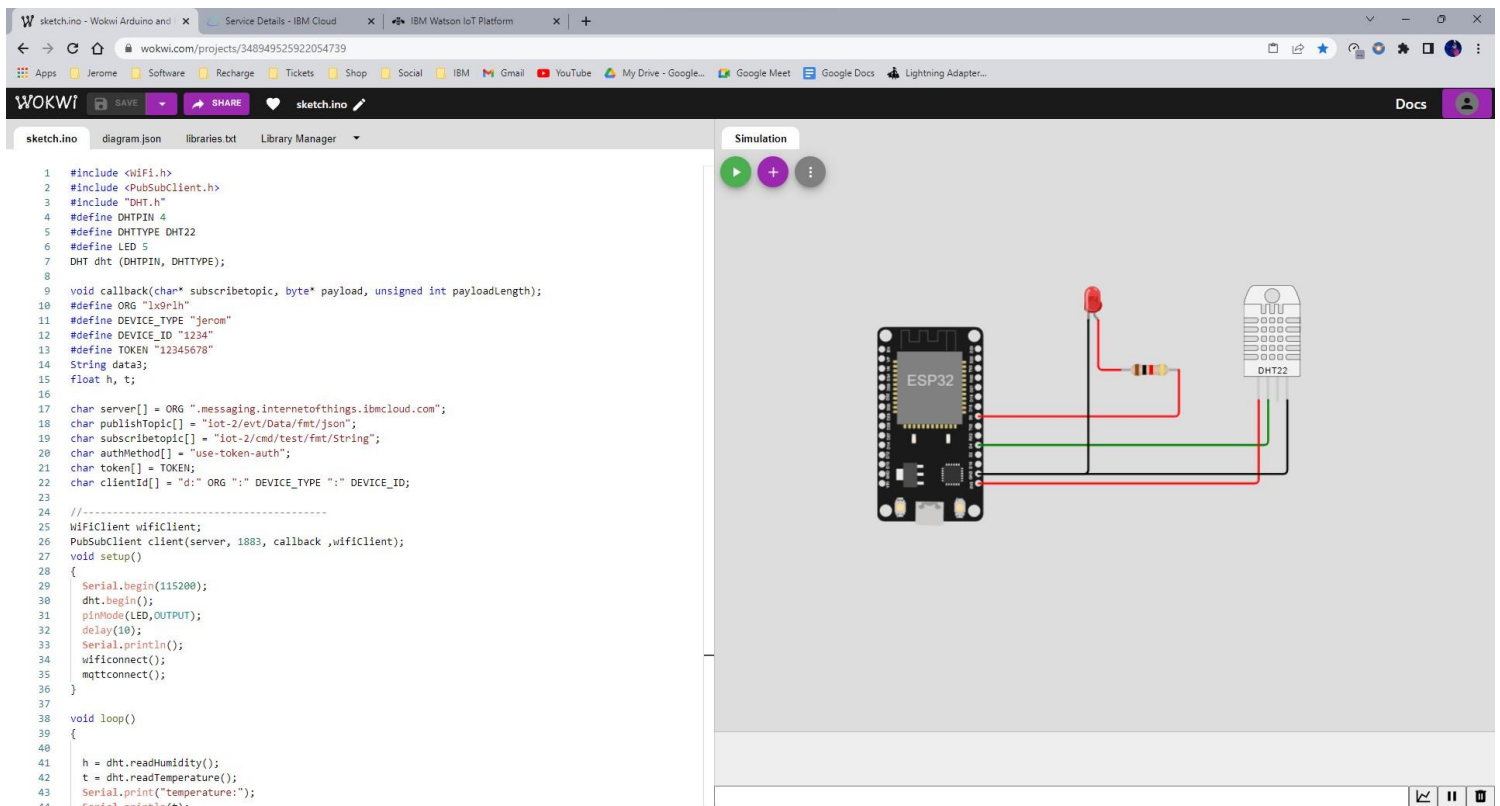
SPRINT GOAL:

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

POGRAM 01:

AIM: To find the Temperature and Humidity DHT22 and ESP32

PLATFORM: WOKWI



The screenshot displays the Wokwi IDE interface with a C++ sketch for an ESP32 microcontroller connected to a DHT22 temperature and humidity sensor. The sketch includes the following code:

```
1 #include <Wifi.h>
2 #include <PubSubClient.h>
3 #include "DHT.h"
4 #define DHTPIN 4
5 #define DHTTYPE DHT22
6 #define LED 5
7 DHT dht (DHTPIN, DHTTYPE);
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
10 #define ORG "1x9r1h"
11 #define DEVICE_TYPE "jerom"
12 #define DEVICE_ID "1234"
13 #define TOKEN "12345678"
14 String data3;
15 float h, t;
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
18 char publishTopic[] = "lot-2/evt/Data/fmt/json";
19 char subscribetopic[] = "lot-2/cmd/test/fmt/String";
20 char authMethod[] = "use-token-auth";
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
23
24 //-----
25 WiFiClient wifiClient;
26 PubSubClient client(server, 1883, callback ,wifiClient);
27 void setup()
28 {
29     Serial.begin(115200);
30     dht.begin();
31     pinMode(LED, OUTPUT);
32     delay(10);
33     Serial.println();
34     wifiConnect();
35     mqttconnect();
36 }
37
38 void loop()
39 {
40
41     h = dht.readHumidity();
42     t = dht.readTemperature();
43     Serial.print("Temperature:");
44     Serial.println(t);
```

The simulation window on the right shows the hardware setup: an ESP32 microcontroller board connected to a DHT22 sensor module. The sensor's VCC pin is connected to the ESP32's 5V pin, GND to GND, and the data pin to the ESP32's D4 pin. A red LED is connected to the ESP32's D5 pin (via a resistor) and its anode to the 5V pin.

Service Details - IBM Cloud

IBM Watson IoT Platform

sketchino - Wokwi Arduino and

+

wokwi.com/projects/34894952592054739

Apps Jerome Software Recharge Tickets Shop Social IBM Gmail YouTube My Drive - Google... Google Meet Google Docs Lightning Adapter...

WOKWI

SAVE SHARE sketchino

Docs

sketchino

diagram.json

libraries.txt

Library Manager

```

1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include "DHT.h"
4 #define DHTPIN 4
5 #define DHTTYPE DHT22
6 #define LED 5
7 DHT dht (DHTPIN, DHTTYPE);
8
9 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
10 #define ORG "ix9rlh"
11 #define DEVICE_TYPE "jerom"
12 #define DEVICE_ID "1234"
13 #define TOKEN "12345678"
14 String data3;
15 float h, t;
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
18 char publishTopic[] = "iot-2/evt/Data/fmt/json";
19 char subscribetopic[] = "iot-2/cmd/test/fmt/String";
20 char authMethod[] = "use-token-auth";
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
23
24 //-----
25 WiFiClient wifiClient;
26 PubSubClient client(server, 1883, callback, wifiClient);
27 void setup()
28 {
29   Serial.begin(115200);
30   dht.begin();
31   pinMode(LED, OUTPUT);
32   delay(10);
33   Serial.println();
34   wifiConnect();
35   mqttConnect();
36 }
37
38 void loop()
39 {
40
41   h = dht.readHumidity();
42   t = dht.readTemperature();
43   Serial.print("temperature:");
44   ..

```

Simulation

00:37.601 100%

Editing DHT22

Temperature: 30.0°C

Humidity: 44.0%

```

temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok
temperature:30.00
Humidity:44.00
Sending payload: {"temperature":30.00,"humidity":44.00}
Publish ok

```

Service Details - IBM Cloud

IBM Watson IoT Platform

+

ix9rlh.internetofthings.ibmcloud.com/dashboard/devices/browse

Apps Jerome Software Recharge Tickets Shop Social IBM Gmail YouTube My Drive - Google... Google Meet Google Docs Lightning Adapter...

IBM Watson IoT Platform

912019104013@smartinternz.com ID: lx9rlh

Browse Action Device Types Interfaces

Search by Device ID

Device Simulator

Device ID

Status

Device Type

Class ID

Date Added

Descriptive Location

Added By

Device Class

1234

Connected

jerom

Device

Nov 22, 2022 6:43 PM

912019104013@smartinternz.com

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":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":6.4,"humidity":27.5}	json	a few seconds ago
Data	{"temperature":30,"humidity":28.5}	json	a few seconds ago

Items per page 50 | 1-1 of 1 item

1 of 1 page

0 Simulations running

PYTHON CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT22
#define LED 5
DHT dht (DHTPIN, DHTTYPE);

void callback(char* subscribtopic, byte* payload, unsigned int payloadLength);
#define ORG "lx9r1h"
#define DEVICE_TYPE "jerom"
#define DEVICE_ID "1234"
#define TOKEN "12345678"
String data3;
float h, t;

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribtopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;

//-----
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
void setup()
{
    Serial.begin(115200);
    dht.begin();
    pinMode(LED,OUTPUT);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()
{
    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();
    }
}

void PublishData(float temp, float humid) {
    mqttconnect();

    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");
    } 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()
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);
    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++) {
        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/348949525922054739>

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

The screenshot displays the Wokwi IDE interface with the following components:

- Code Editor (Left):** Contains the Arduino sketch for the ESP32. The code includes headers for `WiFi.h` and `PubSubClient.h`, defines the server, topic, and device information, and implements a loop that triggers an alert when the distance is less than 100cm.
- Simulation (Right):** Shows a visual representation of the ESP32 and the HC-SR04 ultrasonic sensor connected by wires. The sensor is configured to trigger an alert when the distance is less than 100cm.
- Console (Bottom Right):** Displays the execution log, showing the connection to the IBM Cloud IoT Platform, the sending of the payload, and the resulting alert message.

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 void callback(char* topic, byte* payload, unsigned int payloadLength);
4 #define ORG "3ab6qp"
5 #define DEVICE_TYPE "jerom"
6 #define DEVICE_ID "12345"
7 #define TOKEN "12345678"
8 String data;
9 char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
10 char publishTopic[] = "iot-2/evt/Data/fmt/json";
11 char subscribeTopic[] = "iot-2/cmd/test/fmt/String";
12 char authMethod[] = "use-token-auth";
13 char token[] = TOKEN;
14 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
15 WiFiClient wificlient;
16 PubSubClient client(server, 1883, callback, wificlient);
17 const int trigpin = 5;
18 const int echopin = 18;
19 const int ledpin = 2;
20 long duration;
21 float distance;
22 #define sound_speed 0.034
23
24 void setup() {
25   Serial.begin(115200);
26   pinMode(trigpin, OUTPUT);
27   pinMode(echopin, OUTPUT);
28   pinMode(ledpin, OUTPUT);
29   wificlient.connect();
30   mqttconnect();
31 }
32
33 void loop() {
34   digitalWrite(trigpin, LOW);
35   digitalWrite(trigpin, HIGH);
36   delayMicroseconds(10);
37   digitalWrite(trigpin, LOW);
38
39   duration = pulseIn(echopin, HIGH);
40   distance = duration * sound_speed / 2;
41   if (distance <= 100) {
42     PublishData(distance);
43     delay(1000);
44     if (!client.loop()) {
45       mqttconnect();
46     }
47   }
48 }
```

Connecting to
WiFi connected
IP address:
10.10.0.2
Reconnecting client to 3ab6qp.messaging.internetofthings.ibmcloud.com
iot-2/cmd/test/fmt/String
subscribe to cmd OK

Sending payload: {"ALERT...!! ": 83.95}
Publish ok
ALERT...!!!
83.95
Sending payload: {"ALERT...!! ": 83.98}
Publish ok

IBM Watson IoT Platform

Search by Device ID

Device Simulator ☒

Device ID	Status	Device Type	Class ID	Date Added	Descriptive Location	Added By	Device Class
12345	Connected	jerom	Device	Oct 10, 2022 1:12 PM		912019104013@smartinternz.com	

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	{"ALERT...!! ":"63.97"}	json	a few seconds ago
Data	{"ALERT...!! ":"63.97"}	json	a few seconds ago
Data	{"ALERT...!! ":"63.97"}	json	a few seconds ago
Data	{"ALERT...!! ":"62.95"}	json	a few seconds ago

Items per page: 50 | 1-1 of 1 item

0 Simulations running

26°C Partly cloudy

1:36.6 KB/s 1:1.15 MB/s

ENG

10:27 PM 27/10/2022

PYHTON CODE:

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);
#define ORG "1x9r1h"
#define DEVICE_TYPE "jerom"
#define DEVICE_ID "12345"
#define TOKEN "12345678"
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribetopic[] = "iot-2/cmd/test/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883, callback ,wifiClient);
const int trigpin = 5;
const int echopin = 18;
const int ledpin = 2;
long duration ;
float distance;
#define sound_speed 0.034

void setup() {
    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);
    }
    delay(10); // this speeds up the simulation
}

void PublishData(float distance) {
    mqttconnect();

    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");
    } 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()
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);
    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/349037081204359762>

Conclusion:

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