

Assignment -4

ESP32 Programming with IBM Cloud

Assignment Date	3 November 2022
Student Name	Suvetha S
Student Roll Number	717819F249
Maximum Marks	2 Marks

Question-1:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send "alert" to ibm cloud and display in device recent events.

Upload document with wokwi share link and images of ibm cloud.

Solution:

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

#define ECHO_GPIO 12
#define TRIGGER_GPIO 13
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"

Ultrasonic ultrasonic(13, 12);
int distance;

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

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

#define ORG "2melo1" //IBM ORGANITION ID
#define DEVICE_TYPE "Kruthika" //Device type mentioned in ibm Watson IOT
Platform
#define DEVICE_ID "0405" //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, port and wificredential
```

```

void setup()// configuring the ESP32 {
    Serial.begin(115200);
    delay(10); Serial.println();
    wificonnect(); mqttconnect();
}

void loop()// Recursive Function
{
    distance = ultrasonic.read(CM);
    if(distance< 100)
    {
        Serial.print("Distance in CM: ");
        Serial.println(distance);
        PublishData(distance);
        delay(1000);
        if (!client.loop())
        {
            mqttconnect();
        }
    }

    delay(1000);
}

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

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

```

The screenshot displays the Wokwi IoT simulator interface, which is used for developing and testing IoT applications on a virtual platform.

Top Section: Code Editor and Simulation View

- Code Editor:** Contains a C++ program for an ESP32 microcontroller. The program uses the `Ultrasonic` library to interface with an HC-SR04 ultrasonic sensor. It defines a constant `pin = 5` and a variable `distance` of type `int`. The `void setup()` function initializes the serial port at `115200` baud rate and the sensor at `pin`. The `void loop()` function reads the distance from the sensor, formats it into a JSON payload `{"Alert Distance:":distance}`, and publishes it to the MQTT broker `mqtt://192.168.1.1:1883` using the topic `ESP32`. The program includes the `Ultrasonic` library and the `WiFi` and `PubSubClient` libraries.
- Simulation View:** Shows a 3D model of the ESP32 microcontroller and the HC-SR04 ultrasonic sensor connected by wires. The sensor is labeled "HC-SR04" and the microcontroller is labeled "ESP32".
- Terminal Output:** Displays the output of the program, showing the distance measured by the sensor and the payload sent to the MQTT broker. The output is:


```
Distance in CM: 28
Sending payload: {"Alert Distance:":28.00}
Publish ok
Distance in CM: 28
Sending payload: {"Alert Distance:":28.00}
Publish ok
```

Bottom Section: Device Information

The bottom section shows the "Device Information" tab, which displays a table of device details. The table has columns for "Device", "Name", "ID", "Type", "Status", and "Location". The table contains one row of data for the device "ESP32".

Device	Name	ID	Type	Status	Location
ESP32	ESP32	ESP32	ESP32	Online	ESP32