

Assignment -4
ESP32 Programming with IBM Cloud

Date	14 NOVEMBER 2022
Team ID	PNT2022TMID14654
Project Name	Hazardous area monitoring for Industrial plant powered by IoT
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 "iagzqu"//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 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);
  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 shows the 'Recent Events' tab selected. A text message states: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table with the following columns: Event, Value, Format, and Last Received. The table contains five rows of data, all with the value '["West Distance":28]' and the format 'json'. The 'Last Received' column shows 'a few seconds ago' for each entry. At the bottom, there is a pagination bar showing 'Items per page: 10' and '1 of 1 page'.

Event	Value	Format	Last Received
Data	["West Distance":28]	json	a few seconds ago
Data	["West Distance":28]	json	a few seconds ago
Data	["West Distance":28]	json	a few seconds ago
Data	["West Distance":28]	json	a few seconds ago
Data	["West Distance":28]	json	a few seconds ago

Items per page: 10 | 1 of 1 page