

Personal Assistance for Seniors Who Are Self-Reliant

Assignment – IV

Name: Dharshanapriya V

RegisterNumber: 921019104012

E-Mail Id : dhharshanapriya56789@gmail.com

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

Code:

```
#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 "64yf7x" //IBM ORGANITION ID
#define DEVICE_TYPE "b11m3edevicetype" //Device type mentioned in ibm watson
IOT Platform
#define DEVICE_ID "b11m3edeviceid" //Device ID mentioned in ibm watson IOT
Platform
#define TOKEN "-&EMtr7l-v-Gz2G))e" //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="";
}

```

Simulation Output:

The screenshot shows the Wokwi web interface for a project. On the left, the sketch code is visible, which includes libraries for WiFi, MQTT, and the Ultrasonic sensor. It defines pins for the sensor and sets up an MQTT client to connect to the IBM Watson IoT Platform. The main loop calls the sensor's read function and publishes the distance data to a specific topic. On the right, the simulation window shows a 3D model of the ESP32 and the HC-SR04 sensor connected by wires. Below the model, the console output shows the device successfully connecting to WiFi (IP: 10.10.0.2) and sending data to the IoT platform, with a confirmation message 'subscribe to cmd OK'.

Cloud Output:

The screenshot displays the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, showing a table of data points received from the device. The table has columns for Event, Value, Format, and Last Received. The events show a series of distance measurements in centimeters, ranging from 18 to 84. The dashboard also includes a sidebar with navigation icons and a top bar with the user's profile and device ID.

Event	Value	Format	Last Received
event_1	["Alert Distance":84]	json	a few seconds ago
event_1	["Alert Distance":81]	json	a few seconds ago
event_1	["Alert Distance":72]	json	a few seconds ago
event_1	["Alert Distance":92]	json	a few seconds ago
event_1	["Alert Distance":18]	json	a few seconds ago

Link:

<https://wokwi.com/projects/346940553073525330>