**Assignment -4**

Embedded c Interface with IBM Cloud Programming

| Assignment Date | 27 September 2022 |
| --- | --- |
| Student Name | Mr.Praveen |
| Student Roll Number | 113019106060 |
| Maximum Marks | 2 Marks |

**Question-1:**

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.Upload document with wokwi share link and images.

**Solution:**

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

#include <PubSubClient.h>

const int trigPin = 5;

const int echoPin = 18;

//define sound speed in cm/uS

#define SOUND\_SPEED 0.034

#define CM\_TO\_INCH 0.393701

#define LED 2

#define b 13

long duration;

float distanceCm;

float distanceInch;

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

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

#define ORG "en9jhc"//IBM ORGANITION ID

#define DEVICE\_TYPE "roh"//Device type mentioned in ibm watson IOT Platform

#define DEVICE\_ID "12345678"//Device ID mentioned in ibm watson IOT Platform

#define TOKEN "GyAId1J?u!b-g(KQnx" //Token

String data3;

char h;

float 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() {

**Serial**.begin(115200); // Starts the serial communication

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT);

pinMode(b, OUTPUT);

wificonnect();

mqttconnect(); // Sets the echoPin as an Input

}

void loop() {

// Clears the trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculate the distance

t = duration \* SOUND\_SPEED/2;

// Convert to inches

distanceInch = t \* CM\_TO\_INCH;

if (t>100){

const char\* h="YOUR Distance more than 100";

digitalWrite(b,HIGH);

}

else

{ const char\* h="YOUR Distance less than 100";

digitalWrite(b,LOW);

}

// Prints the distance in the Serial Monitor

**Serial**.print("Distance (cm): ");

**Serial**.println(t);

**Serial**.print("Distance (inch): ");

**Serial**.println(distanceInch);

delay(1000);

PublishData(t, h);

delay(1000);

if (!client.loop()) {

mqttconnect();

}

}

void PublishData(float dis, char alert) {

mqttconnect();//function call for connecting to ibm

/\*

creating the String in in form JSon to update the data to ibm cloud

\*/

String payload = "{\"dis\":";

payload += dis;

payload += "," "\"Alert\":";

payload += alert;

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="";

}

<https://wokwi.com/projects/347449533495509588>