**ASSIGNMENT - 4**

|  |  |
| --- | --- |
| **Date** | 06 September 2022 |
| **Team ID** | PNT2022TMID41907 |
| **Project Name** | Personal Assistance for Seniors Who Are Self-Reliant |
| **Team Member** | Jayavarshini  Guruprakash  Pavithra  Samsonjayakumar |

**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. Upload document with wokwi share link and images of IBM cloud.

**Code:**

#include <WiFi.h>

#include <WiFiClient.h>

#include <PubSubClient.h>

const int trigPin = 5;

const int echoPin = 18;

//define sound speed in cm/uS

#define Speed 0.034

#define cm\_to\_inch 0.393701

long duration;

float distance;

float distanceInch;

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

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

#define ORG "  08foek "//IBM ORGANITION ID

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

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

#define TOKEN "123456789"     //Token

String data3;

//-------- 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/test/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);

void setup() {

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

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

  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

**Serial**.println();

  wificonnect();

  mqttconnect();

}

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

  distance = duration \* Speed/2;

  // Convert to inches

  distanceInch = distance \* cm\_to\_inch;

  // Prints the distance in the Serial Monitor

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

**Serial**.println(distance);

  PublishData(distance);

  delay(1000);

  if (!client.loop()) {

    mqttconnect();

  }

}

  void PublishData(float centimeter) {

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

  /\*

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

  \*/

  String payload = "{\"Distance in Centimeter\":";

  payload += centimeter;

  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... ");

  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);

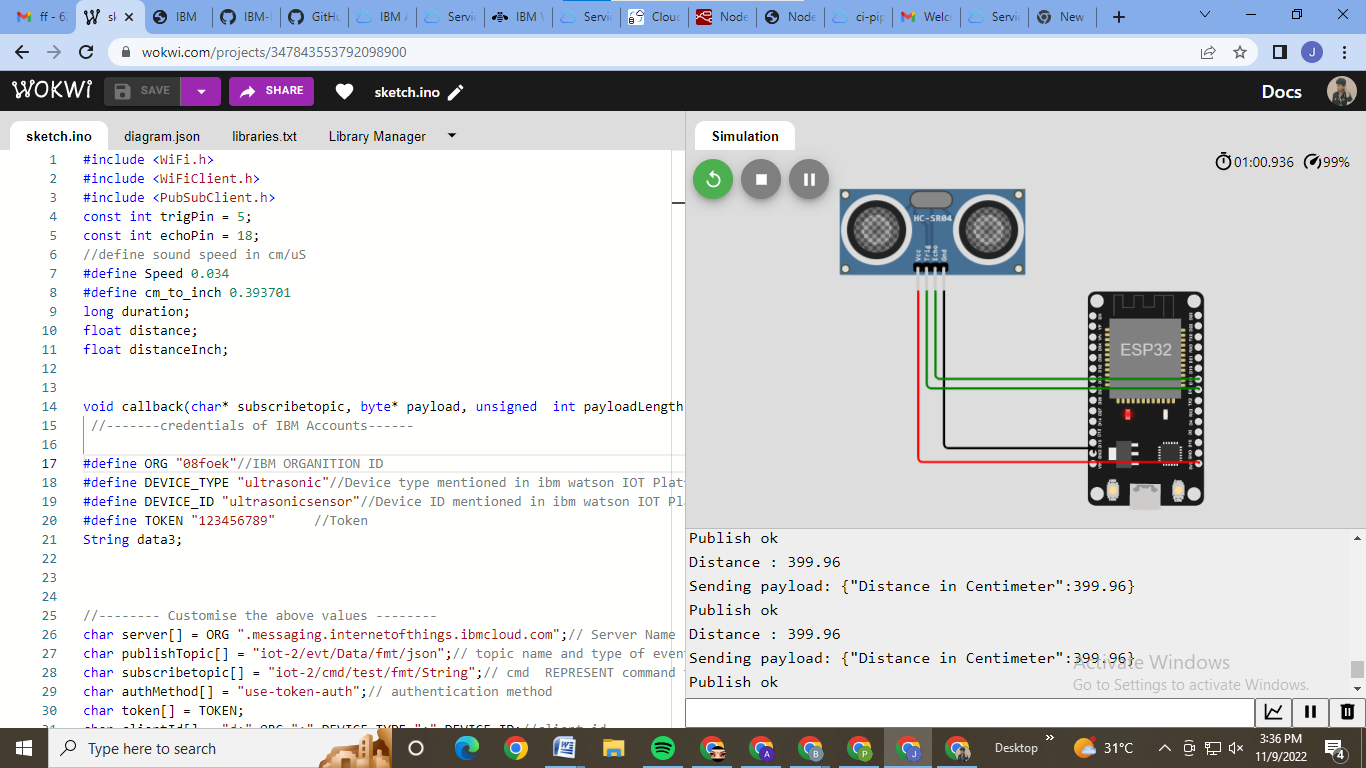
  for (int i = 0; i < payloadLength; i++) {

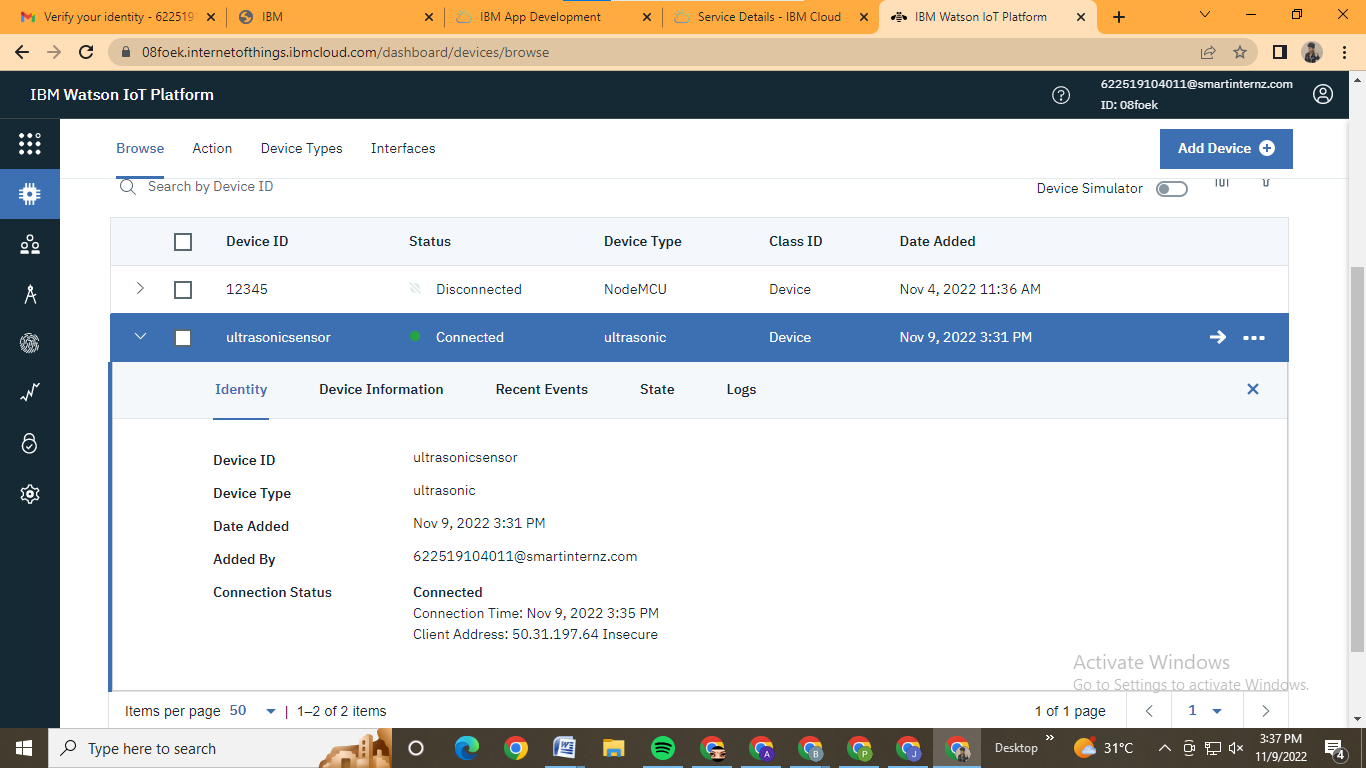
    //Serial.print((char)payload[i]);

    data3 += (char)payload[i];

  }

}



****