**Assignment - 4**

Ultrasonic Sensor

|  |  |
| --- | --- |
| Assignment Date | 22 October 2022 |
| Student Name | Sudarshan A R |
| Student Roll Number | 2019504594 |
| Maximum Marks | 2 Marks |

**Question-1:**

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 cm send “alert” to IBM cloud and display in device recent events.

## Solution:

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

#include <PubSubClient.h>//library for MQtt

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

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

#define ORG "lqca59"//IBM ORGANITION ID

#define DEVICE\_TYPE "raspberrypi"//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;

//-------- 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); //calling the predefined client id by passing parameter like server id,portand wificredential

#define ECHO\_PIN 2

#define TRIG\_PIN 4

#define LED 15

void setup() {

**Serial**.begin(115200);

  randomSeed(analogRead(0));

  pinMode(LED, OUTPUT);

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

**Serial**.println();

  wificonnect();

  mqttconnect();

}

float readDistanceCM() {

  digitalWrite(TRIG\_PIN, LOW);

  delayMicroseconds(2);

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);

  digitalWrite(TRIG\_PIN, LOW);

  int duration = pulseIn(ECHO\_PIN, HIGH);

  float num = random(255);

  return duration \* num \* 0.0001 / 2;

}

void loop() {

  float dist = readDistanceCM();

  bool near = dist < 100;

  digitalWrite(LED, near);

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

**Serial**.println(dist);

  delay(100);

  if (dist < 100) {

    PublishAlert(dist);

  }

  else {

    PublishData(dist);

  }

  delay(1000);

  if (!client.loop()) {

    mqttconnect();

  }

}

void PublishData(float dist){

  mqttconnect();

  String payload = "{\"Distance\":";

  payload += dist;

  payload += "}";

**Serial**.print("Sending payload: ");

**Serial**.println(payload);

  if (client.publish(publishTopic, (char\*) payload.c\_str())){

**Serial**.println("Publish Ok");

  }

  else {

**Serial**.println("Publish failed");

  }

}

void PublishAlert(float dist){

  mqttconnect();

  String payload = "{\"Alert!\":";

  payload += dist;

  payload += "}";

**Serial**.print("Sending payload: ");

**Serial**.println(payload);

  if (client.publish(publishTopic, (char\*) payload.c\_str())){

**Serial**.println("Publish Alert Ok");

  }

  else {

**Serial**.println("Publish Alert 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() {

**Serial**.println();

**Serial**.print("Connecting to ");

  WiFi.begin("Wokwi-GUEST", "", 6);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(100);

**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++) {

    data3 += (char)payload[i];

  }

**Serial**.println("data: "+data3);

  if (data3 == "lighton")

  {

**Serial**.println(data3);

    digitalWrite(LED, HIGH);

  }

  else

  {

**Serial**.println(data3);

    digitalWrite(LED, LOW);

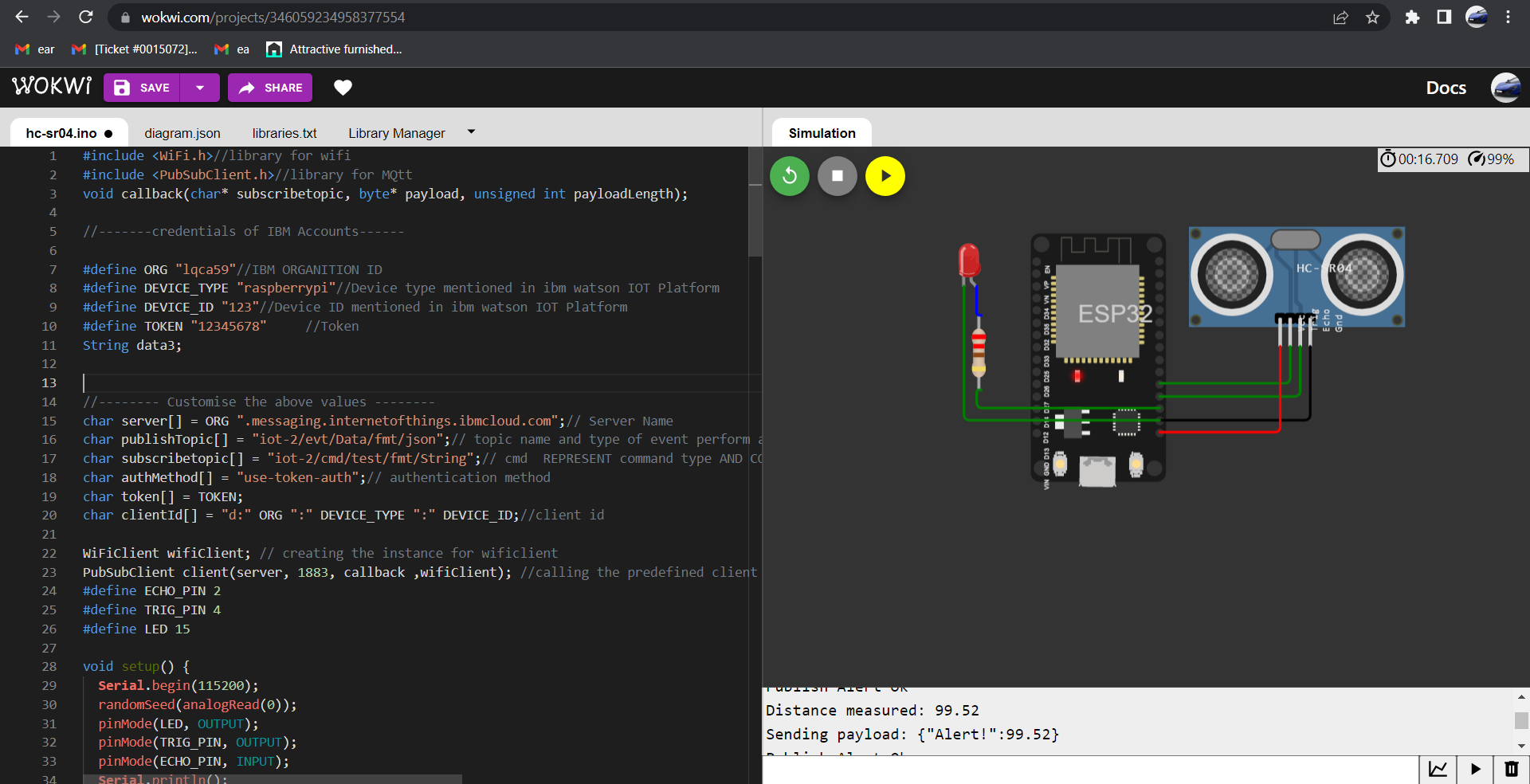
  }

  data3 = "";

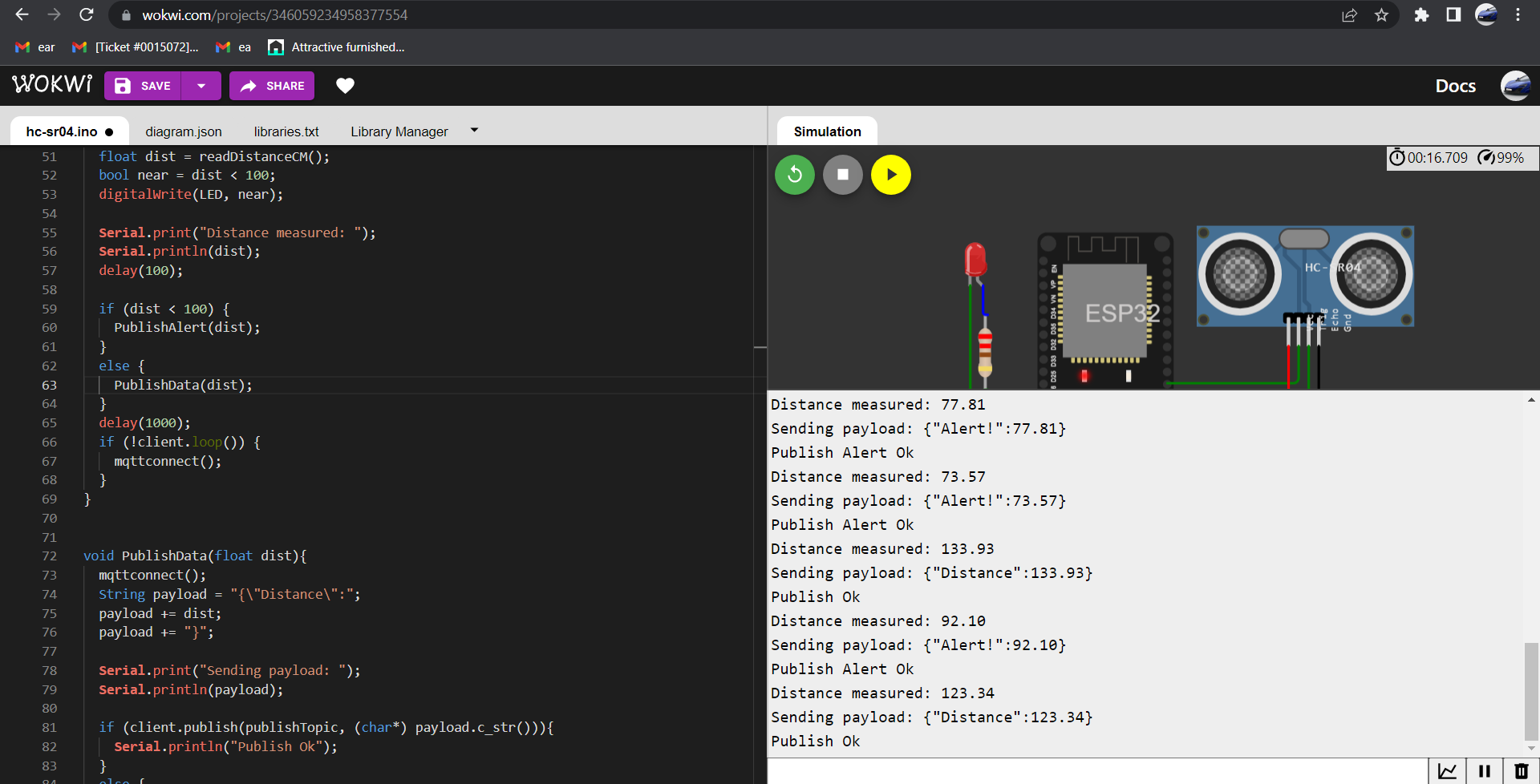
}

## Output:

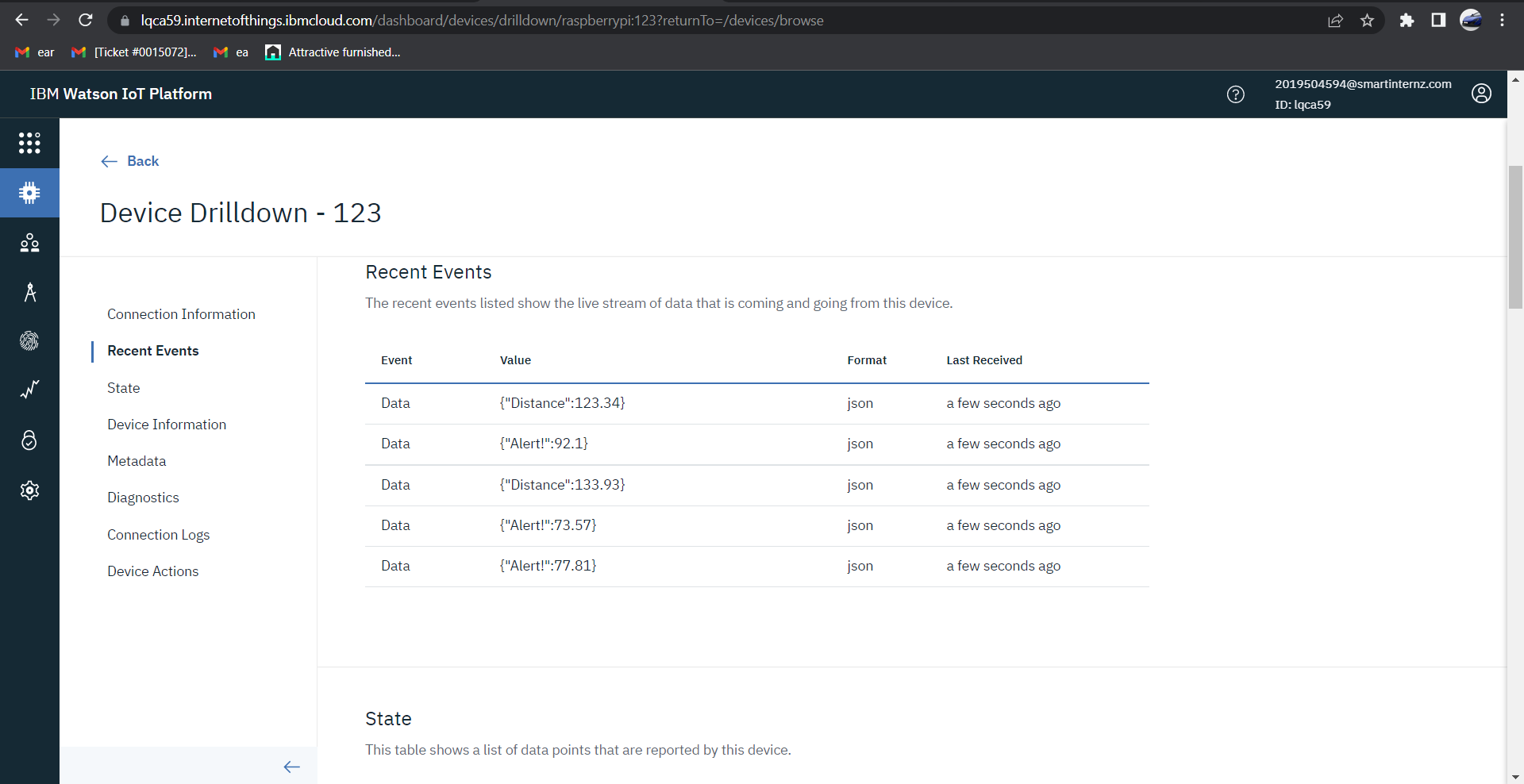
**Code and Diagram:**

****

**Wokwi Output:**

****

**Output in IBM cloud for both normal and alert case:**

****

## Link to wokwi project: https://wokwi.com/projects/346059234958377554