Assignment - 4

IOT interfacing

Assignment Date	15 October 2022
Student Name	Siva Chokkalingam S
Student Roll Number	2019504588
Maximum Marks	2 Marks

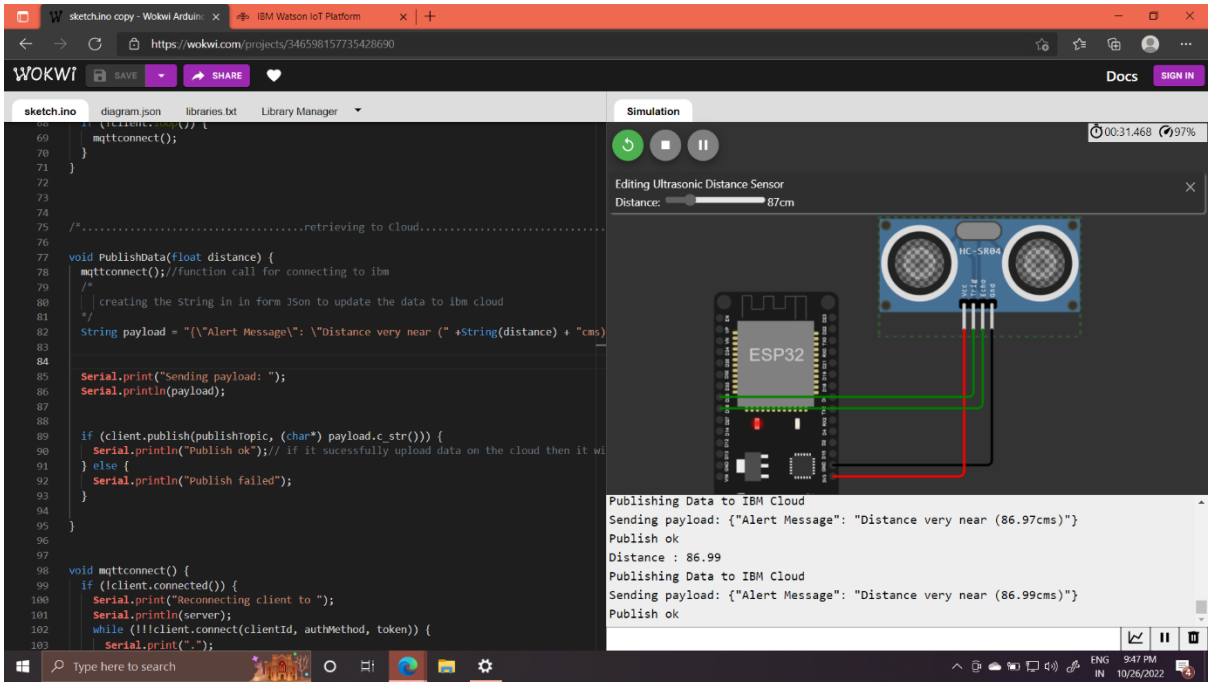
Question-1:

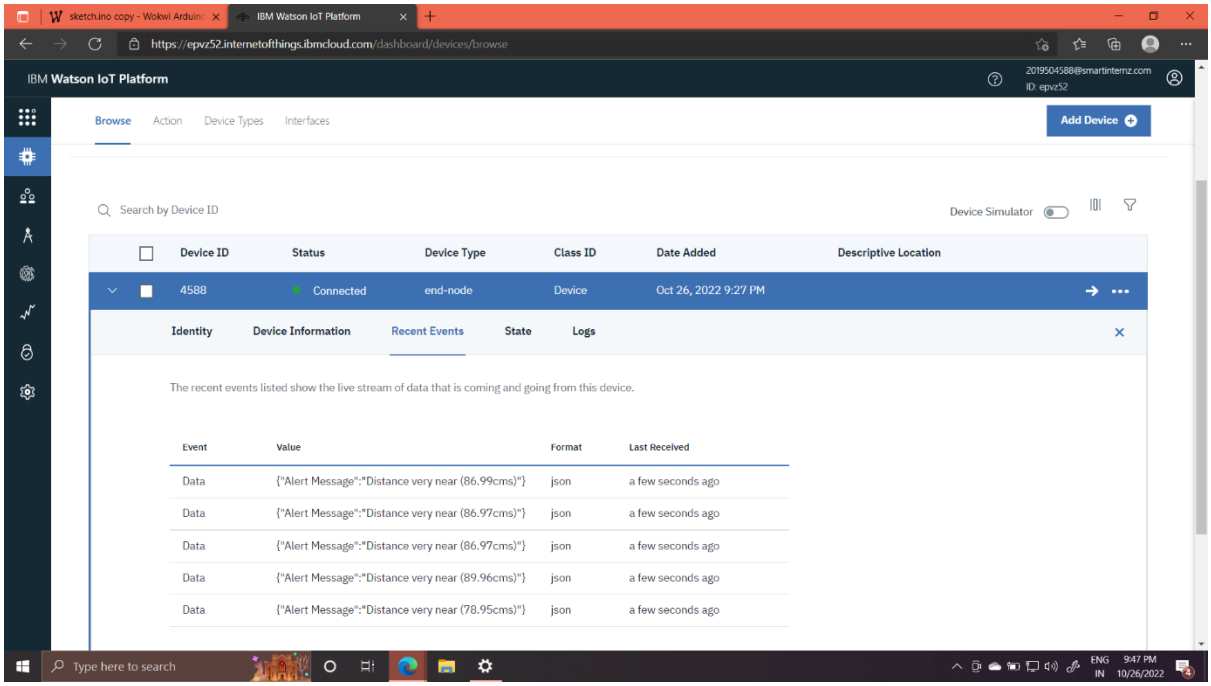
Write Code and connect in wokwi for ultrasonic sensor. Whenever distance is less than 100cms send “alert” to ibm cloud and display in device recent events

https://wokwi.com/projects/346598157735428690

Device credentials:

Organization ID	epvz52
Device Type	end-node
Device ID	4588
Authentication Method	use-token-auth
Authentication Token	12345678





Code:

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt

#define SOUND_VELOCITY 0.034

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

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

#define ORG "epvz52"//IBM ORGANITION ID
#define DEVICE_TYPE "end-node"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "4588"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token

//-------- 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


// -------------------------------------


const int trigPin = 25;

const int echoPin = 26;

float distance;

int duration;


void setup()// configureing the ESP32

{


  Serial.begin(115200);


  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);


  wificonnect();

  mqttconnect();

}
```

```
void loop()// Recursive Function
{
  digitalWrite(trigPin, HIGH);
  delay(150);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*SOUND_VELOCITY/2);

  Serial.println("Distance : "+String(distance));
  if(distance < 100) {
    Serial.println("Publishing Data to IBM Cloud");
    PublishData(distance);
  }

  delay(500);

  if (!client.loop()) {
    mqttconnect();
  }
}

/*.....retrieving to Cloud.....*/

void PublishData(float distance) {
  mqttconnect();//function call for connecting to ibm
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Alert Message\": \"Distance very near (" +String(distance) + "cms)\"}";
```

```
  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.println("Hello");
}
*/

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="";
}
```