Assignment-4

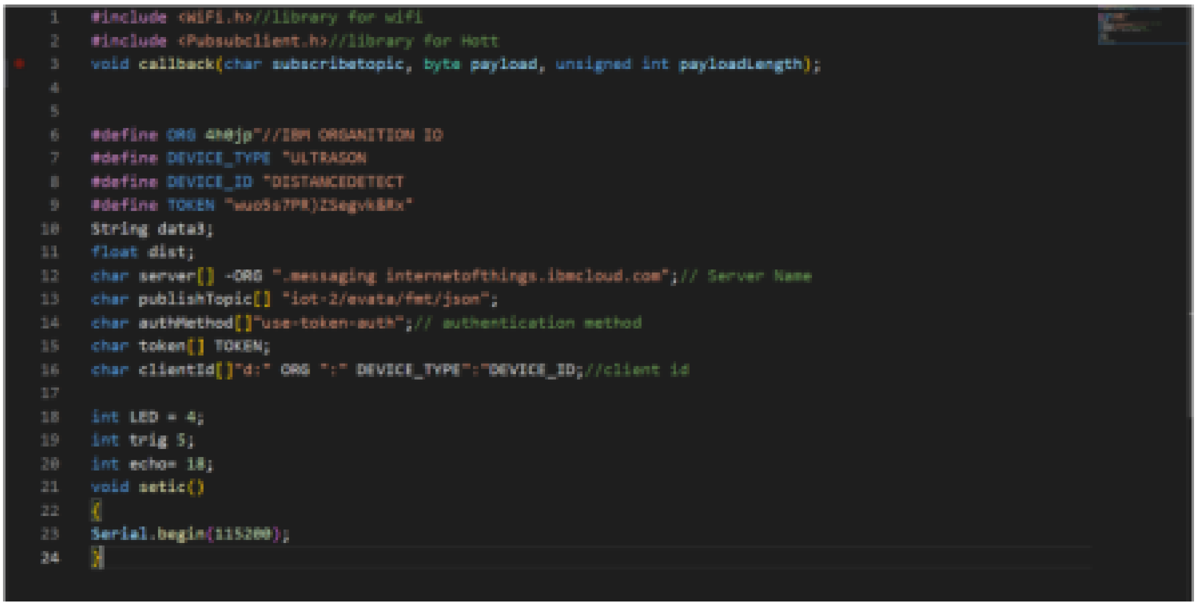
Distance Detection Using Ultrasonic Sensor

Date	26 October 2022
Team ID	PNT2022MID48179
Name	Manisha.S
Student Roll Number	912619106006
Maximum Marks	2 Marks

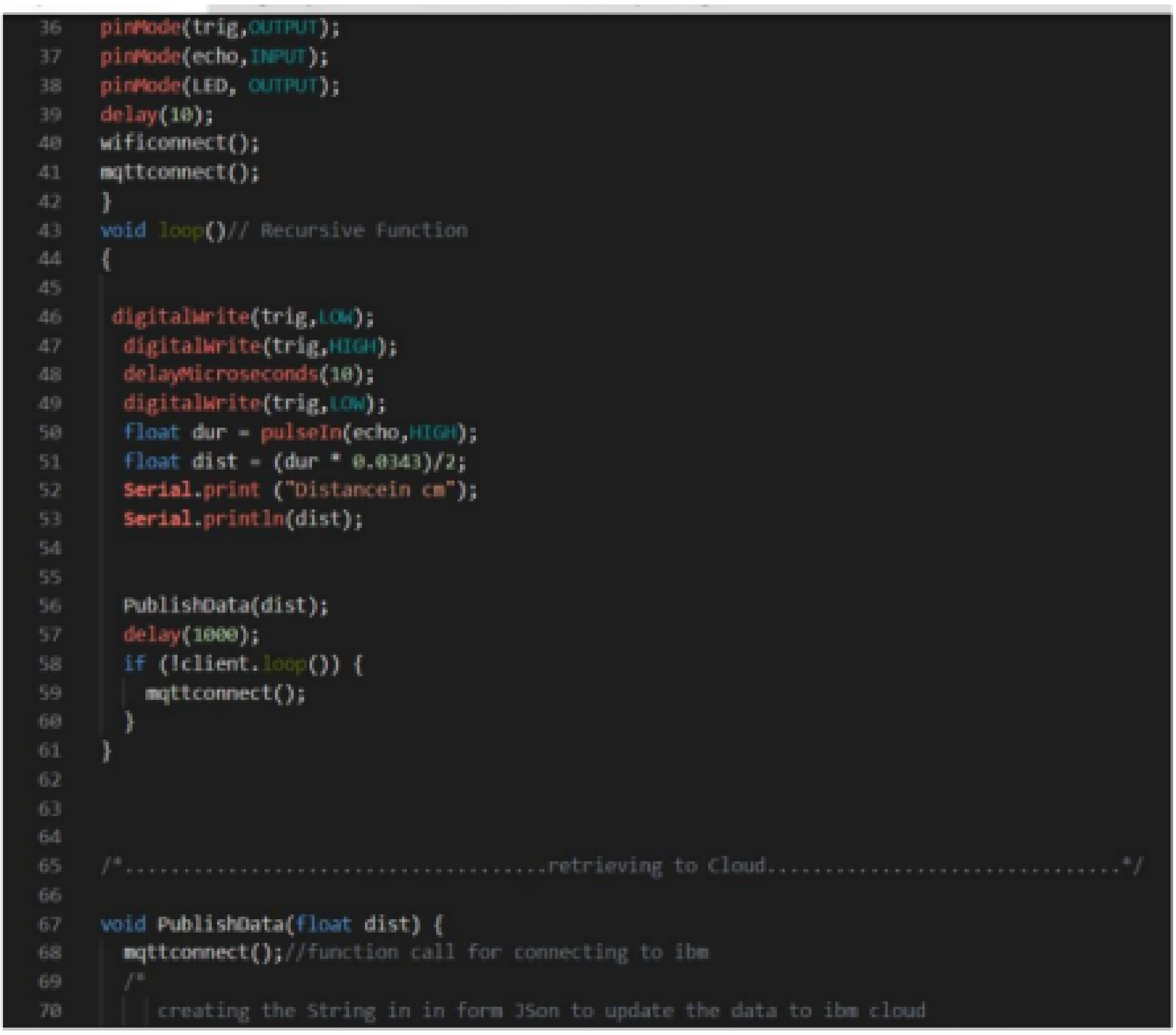
Question 1:

Write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 centimeters it should send "alert" to IBM cloud and display in device recent events.

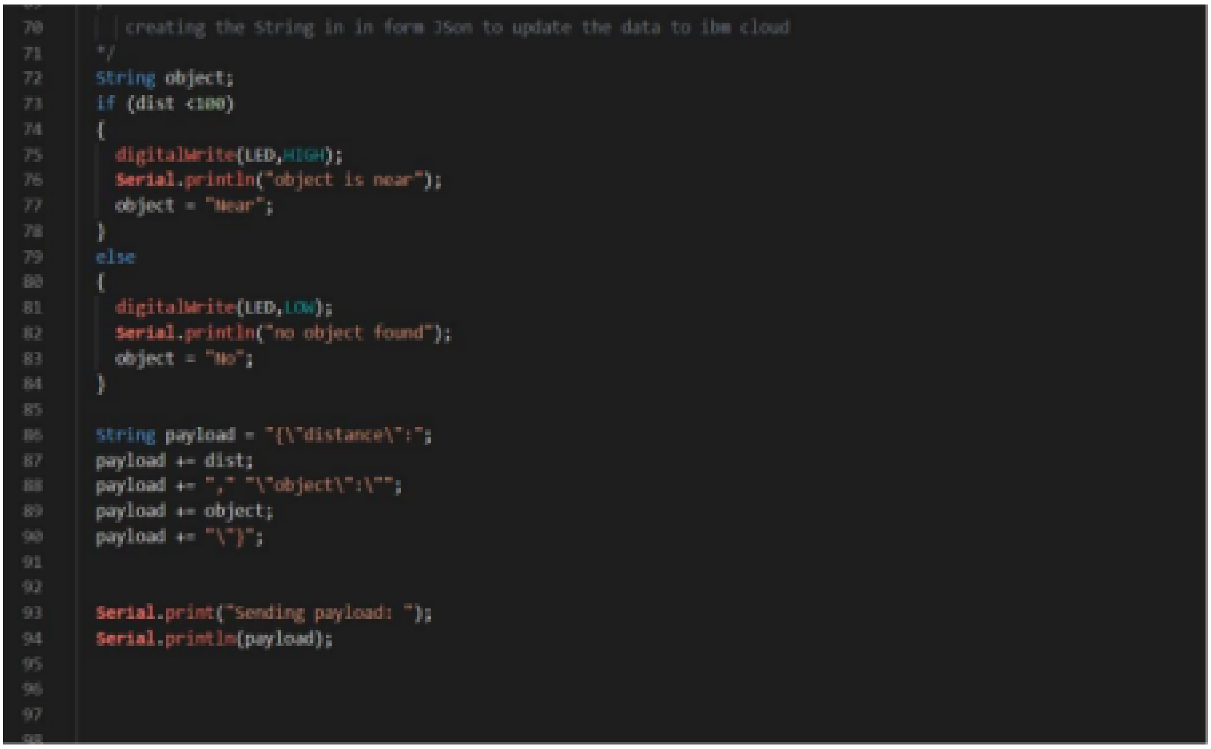
Code:



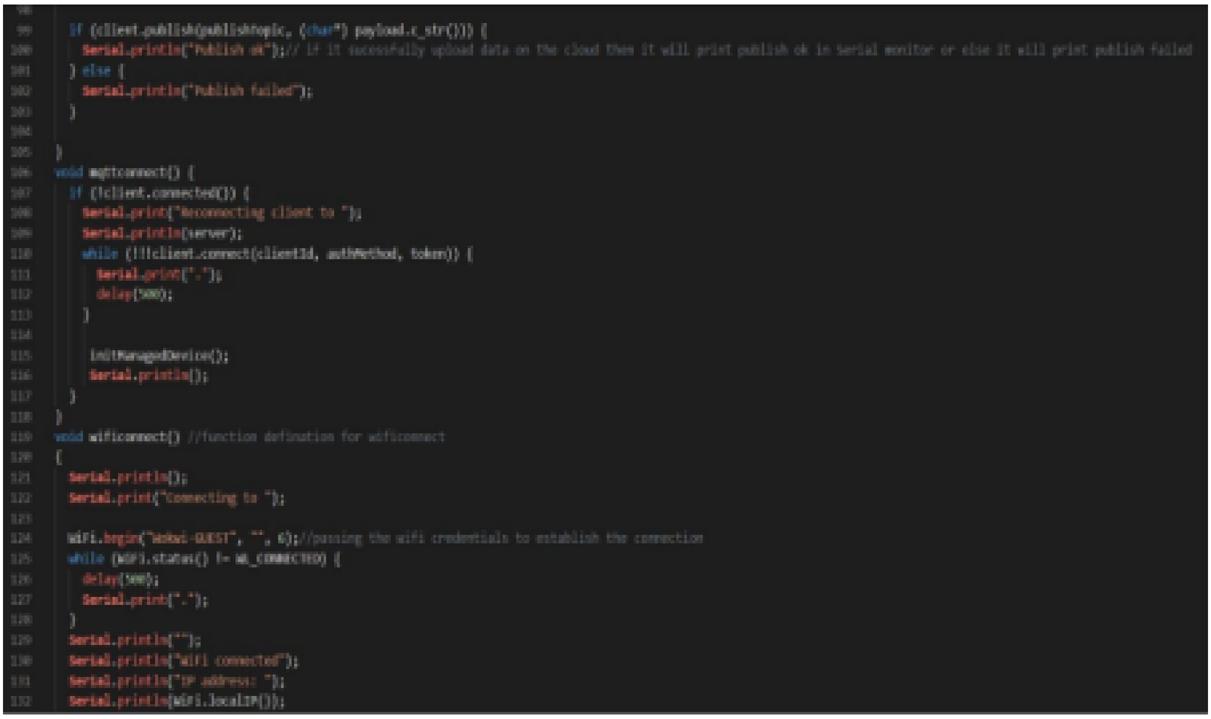
```
#include <WiFi.h>//library for wifi
#include <Pubsubclient.h>//library for Mqtt
void callback(char subscribetopic, byte payload, unsigned int payloadLength);


#define ORG "4h@jp"//IBM ORGANITION ID
#define DEVICE_TYPE "ULTRASON
#define DEVICE_ID "DISTANCEDETECT
#define TOKEN "wuo5s7PR)ZSegvk&Rx"
String data3;
float dist;
char server[] -ORG ".messaging internetofthings.ibmcloud.com";// Server Name
char publishTopic[] "iot-2/evets/fmt/json";
char authMethod[]"use-token-auth";// authentication method
char token[] TOKEN;
char clientId[]"d:" ORG ":" DEVICE_TYPE":"DEVICE_ID;//client id

int LED = 4;
int trig 5;
int echo= 18;
void setic()
{
Serial.begin(115200);
}
```



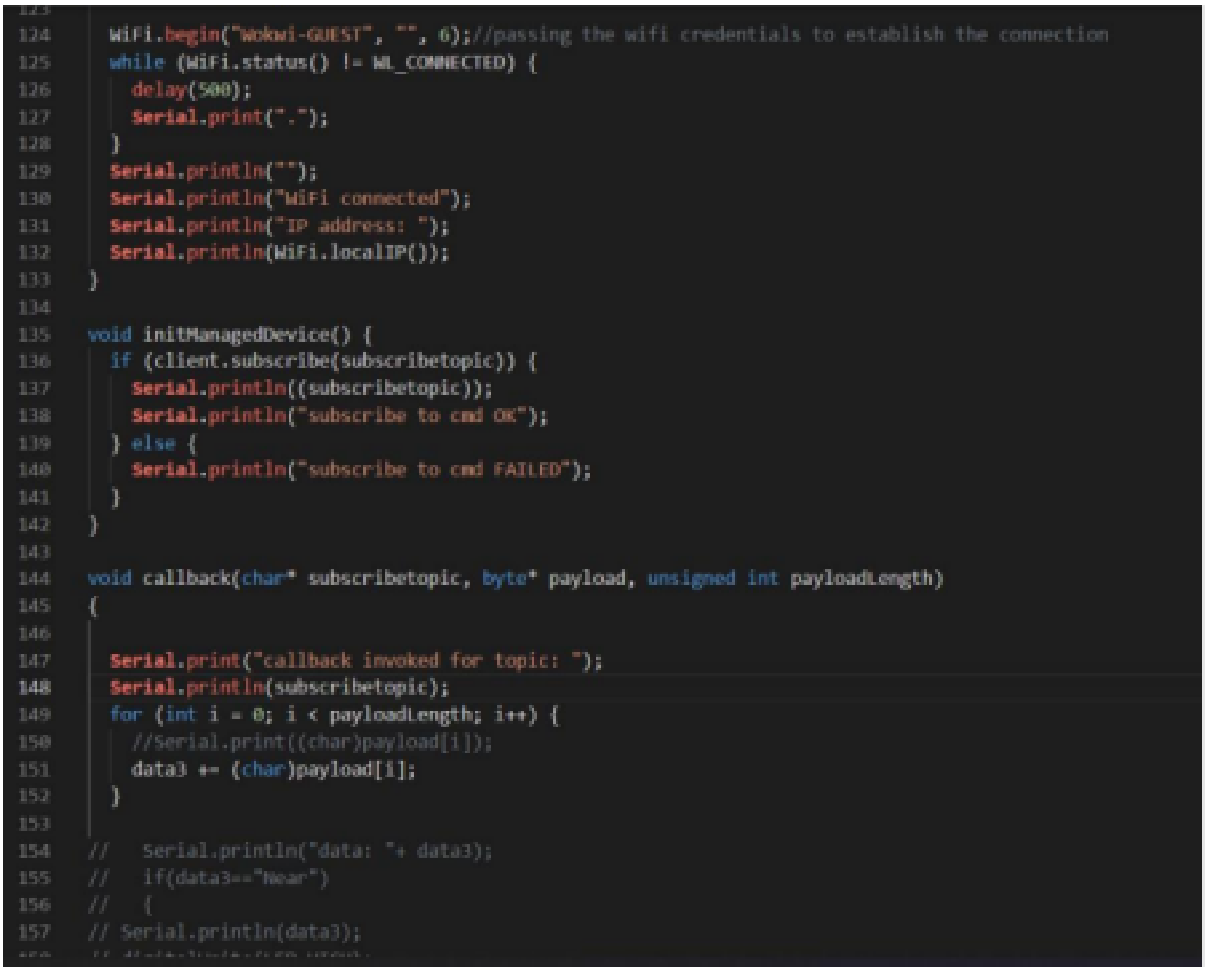
```
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(LED, OUTPUT);
  delay(10);
  wificonnect();
  mqttconnect();
  }
  void loop()// Recursive Function
  {

   digitalWrite(trig,LOW);
    digitalWrite(trig,HIGH);
    delayMicroseconds(10);
    digitalWrite(trig,LOW);
    float dur = pulseIn(echo,HIGH);
    float dist = (dur * 0.0343)/2;
    Serial.print ("Distancein cm");
    Serial.println(dist);


    PublishData(dist);
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
  }



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

  void PublishData(float dist) {
    mqttconnect();//function call for connecting to ibm
    /*
       creating the String in in form JSon to update the data to ibm cloud
```



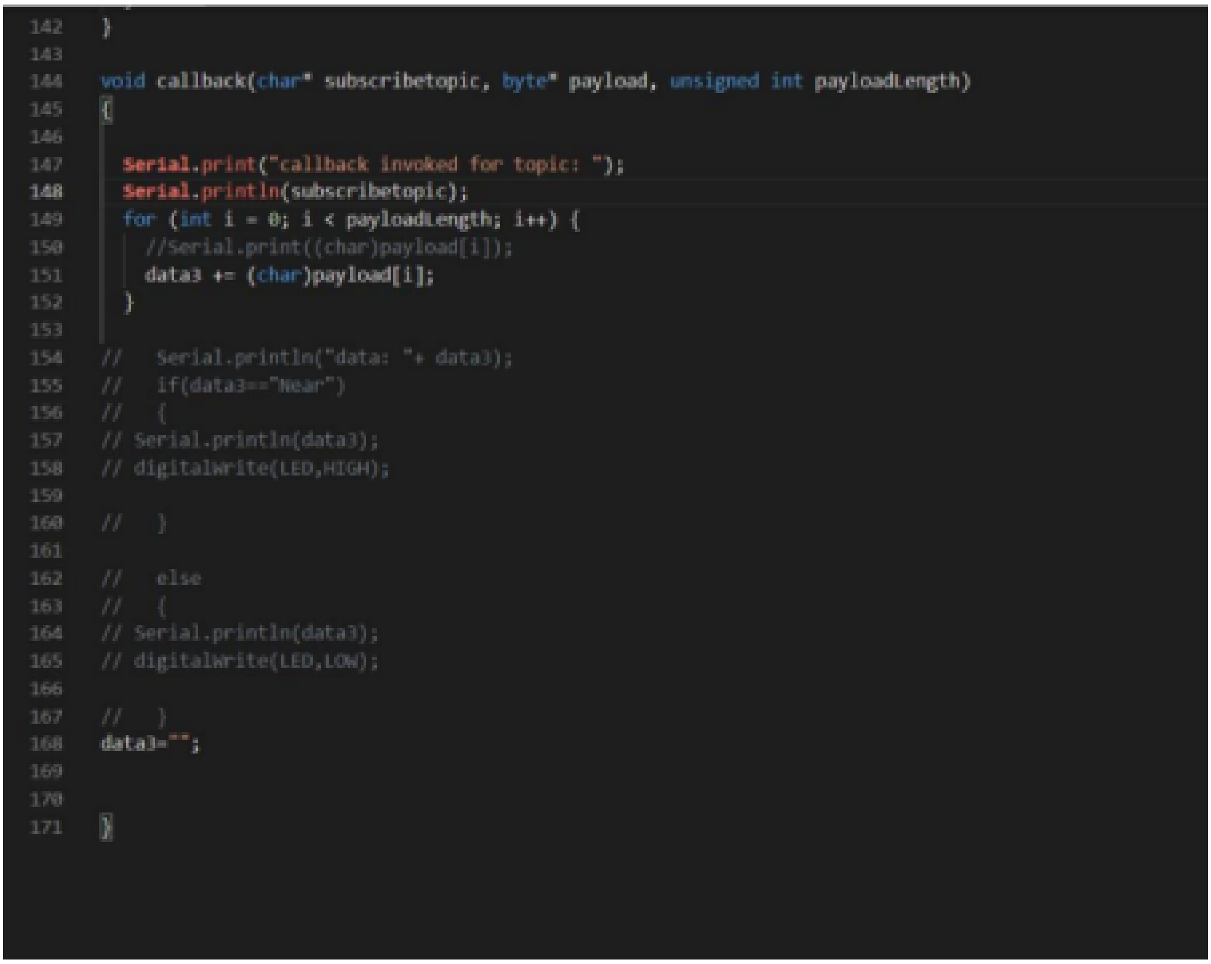
```
       creating the String in in form JSon to update the data to ibm cloud
    */
    String object;
    if (dist <100)
    {
      digitalWrite(LED,HIGH);
      Serial.println("object is near");
      object = "Near";
    }
    else
    {
      digitalWrite(LED,LOW);
      Serial.println("no object found");
      object = "No";
    }

    String payload = "{\"distance\":";
    payload += dist;
    payload += "," "\"object\":\"";
    payload += object;
    payload += "\"}";


    Serial.print("Sending payload: ");
    Serial.println(payload);
```



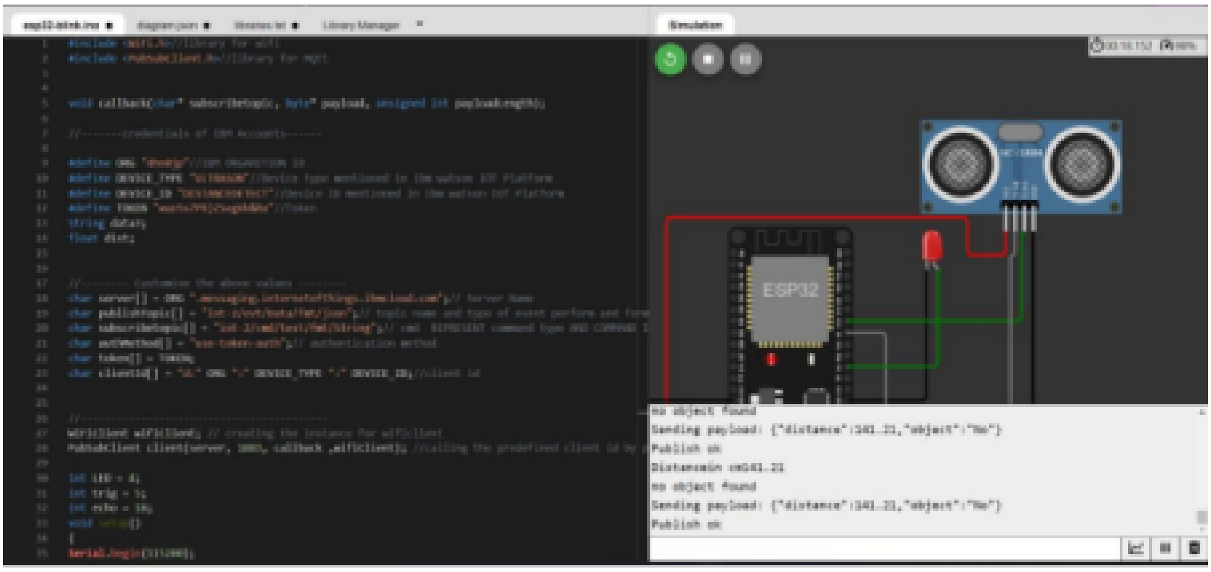
```
  if (client.publish(publishtopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it successfully upload data on the cloud then it will print publish ok in Serial monitor or else it will print publish failed
  } else {
    Serial.println("Publish failed");
  }

}
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());
```



```
  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=="Near")
//   {
// Serial.println(data3);
```



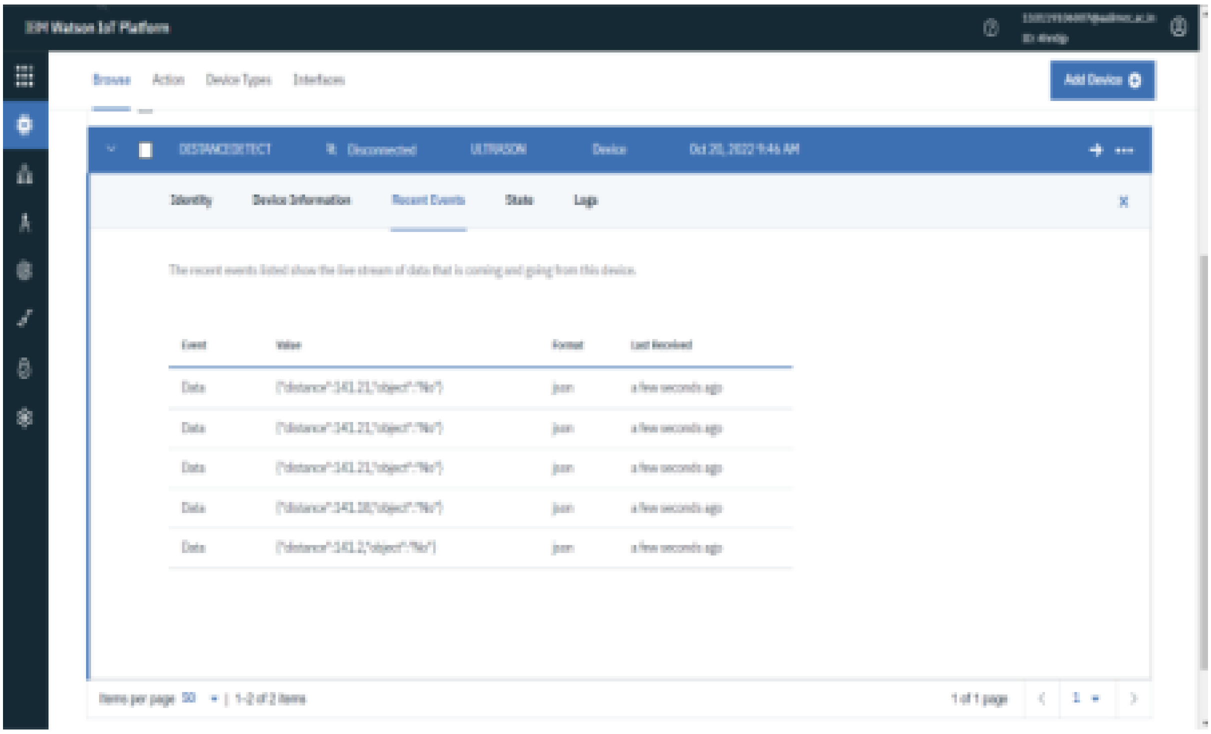
```
}

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=="Near")
//   {
// Serial.println(data3);
// digitalWrite(LED,HIGH);

//   }

//   else
//   {
// Serial.println(data3);
// digitalWrite(LED,LOW);

//   }
data3="";


}
```



```
#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 "[illegible]"//IBM ORGANITION ID
#define DEVICE_TYPE "[illegible]"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "[illegible]"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "[illegible]"//Token
String data3;
float dist;


//-------- 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 form
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command type AND COMMAND I
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 p

int LED = 4;
int trig = 5;
int echo = 18;
void setup()
{
Serial.begin(115200);
```

```
no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish ok
Distancein cm141.21
no object found
Sending payload: {"distance":141.21,"object":"No"}
Publish ok
```

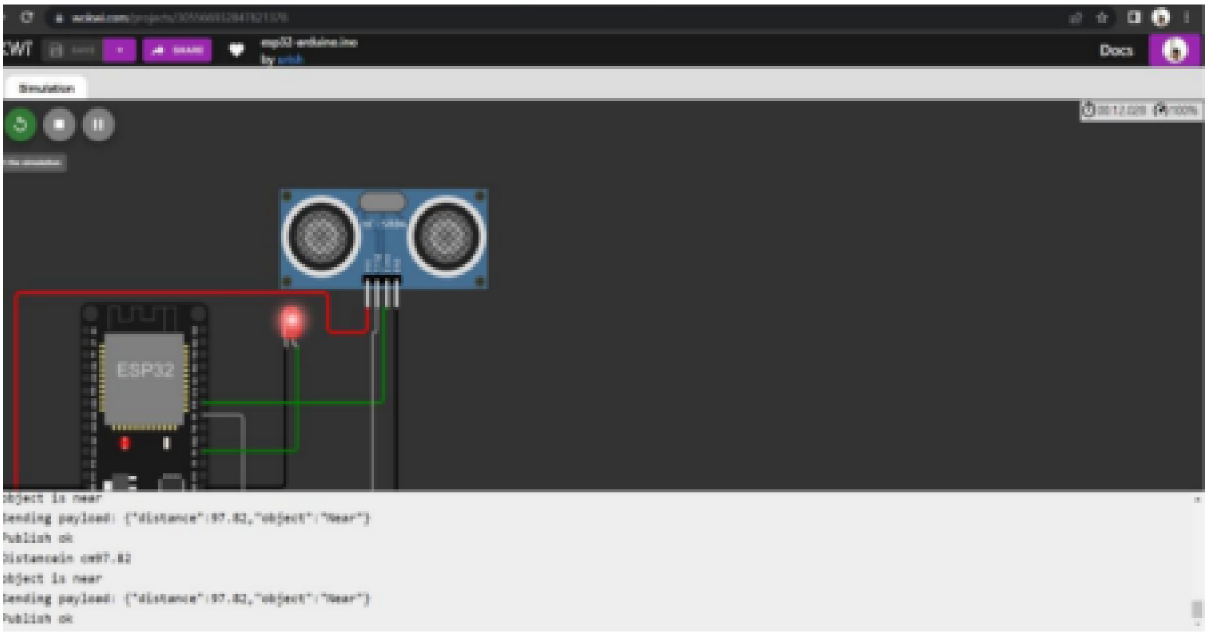
Data send to the IBM cloud service when the object is far



The recent events listed show the live stream of data that is coming and going from this device.

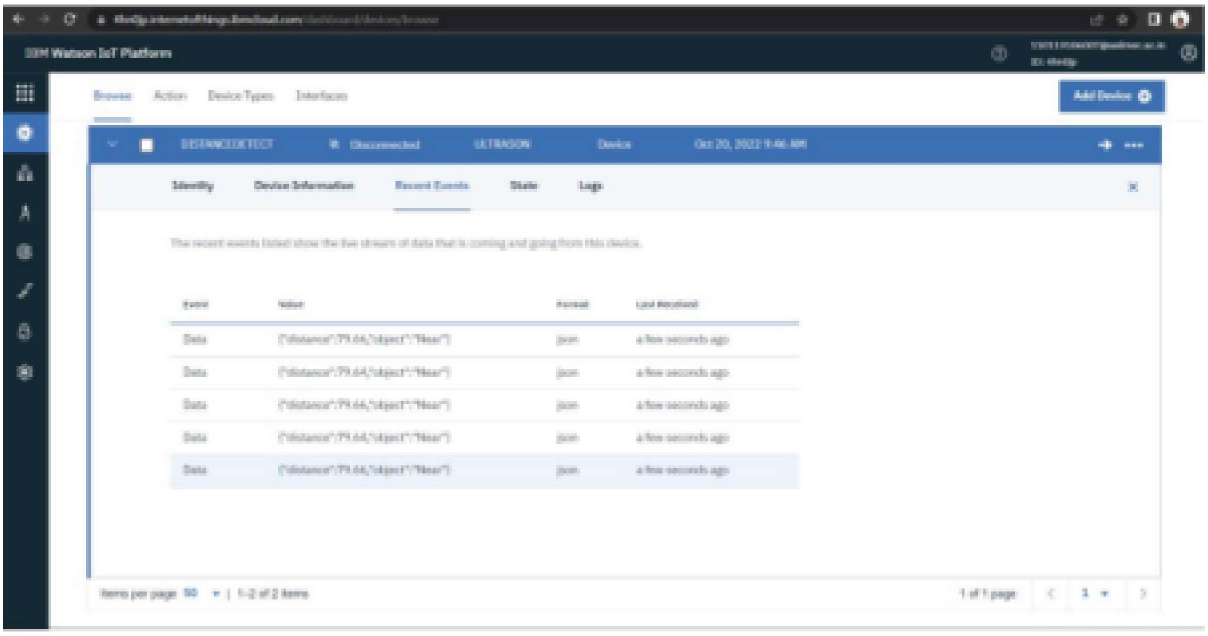
Event	Value	Format	Last Received
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.21,"object":"No"}	json	a few seconds ago
Data	{"distance":141.18,"object":"No"}	json	a few seconds ago
Data	{"distance":141.2,"object":"No"}	json	a few seconds ago

When object is near to the ultrasonic sensor



```
object is near
Sending payload: {"distance":97.82,"object":"Near"}
Publish ok
Distancein cm97.82
object is near
Sending payload: {"distance":97.82,"object":"Near"}
Publish ok
```

Data sent to the IBM Cloud Device when the objects is near



The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago
Data	{"distance":79.66,"object":"Near"}	json	a few seconds ago