Assignment-4

Name:Melodina Carnelian D Reg no:311119106029

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:

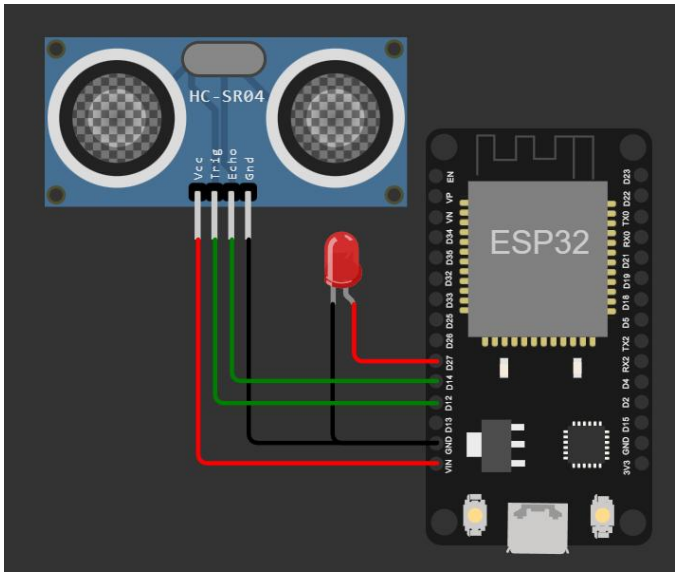
```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int payloadLength);
#define ORG "k0ly3z"
#define DEVICE_TYPE "ESPALPHA"
#define DEVICE_ID "ESP251"
#define TOKEN "dR8tqDUTxABgzQ?Fim"
String data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/distance/fmt/json";
char subscribeTopic[]="iot-2/cmd/test/fmt/String";
char authMethod[]="use-token-auth";
char token[]=TOKEN;
char clientID[]="d:"ORG":"DEVICE_TYPE":"DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server,1883,callback,wifiClient);
#define ECHO_PIN 14
#define TRIG_PIN 12
#define led 27
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  wificonnect();
  mqttconnect();
}
float readDistanceCM() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  int duration=random(1,200);
```

```
  //Serial.println(duration);
  //duration = pulseIn(ECHO_PIN, HIGH);
  return duration ;
  //Serial.println(duration);
}
void loop() {
  float distance = readDistanceCM();
  //Serial.println(distance);
  bool isNearby = distance < 100;
  digitalWrite(led, isNearby);
  Serial.print("Measured distance: ");
  Serial.println(distance);
  if(distance<100){
    PublishData2(distance);
  }else{
    PublishData1(distance);
```

```
  }
  //PublishData(distance);
  delay(1000);
  if(!client.loop()){
    mqttconnect();
  }
  //delay(2000);
}
void PublishData1(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 PublishData2(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 ok");
  } else{
    Serial.println("publish failed");
  }
}
void mqttconnect(){
  if(!client.connected()){
    Serial.print("Reconnecting 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(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++){
    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="";
}
```

CIRCUIT CONNECTION:



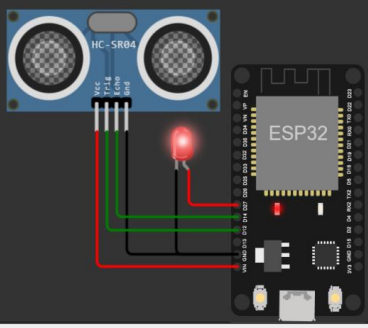
WOKWI OUTPUT:

sketch.ino | diagram.json | libraries.txt | Library Manager

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribetopic,byte* payload, unsigned int payloadLe
#define ORG "k0ly3z"
#define DEVICE_TYPE "ESPALPHA"
#define DEVICE_ID "ESP251"
#define TOKEN "dR8tqDUTxABgzQ?Fim"
String data3;

char server[]= ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[]="iot-2/evt/distance/fmt/json";
char subscribeTopic[]="iot-2/cmd/test/fmt/String";
char authMethod[]="use-token-auth";
char token[]=TOKEN;
char clientID[]="d:"ORG":"DEVICE_TYPE":"DEVICE_ID;

WiFiClient wifiClient;
PubSubClient client(server,1883,callback,wifiClient);

#define ECHO_PIN 14
#define TRIG_PIN 12
#define led 27

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  pinMode(led, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  wificonnect();
  mqttconnect();
}
```

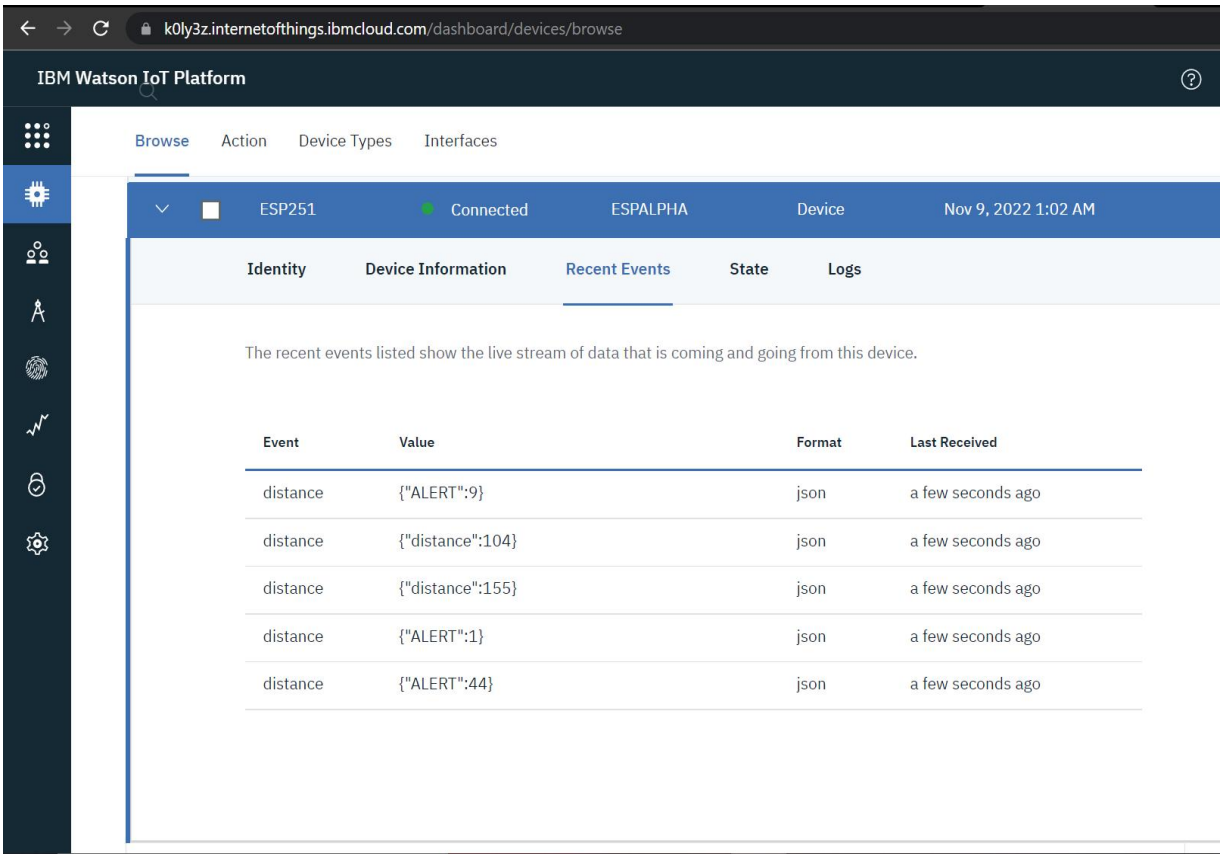
Simulation

02:03.776 99%




```
publish ok
Measured distance: 199.00
Sending payload:{"distance":199.00}
publish ok
Measured distance: 47.00
Sending payload:{"ALERT":47.00}
publish ok
```

LIVE DATA FEED SEEN ON IBM IoT:



INFERENCES:

- The schematics are designed in wokwi website and the code is written to interface the ESP32 connected ultrasonic sensor to IBM IoT platform.
- The simulated distance readings measured using ultrasonic sensor is displayed on the output terminal.
- The sensor readings are sent to and displayed in the IBM IoT platform.