

ASSIGNMENT 4

Write a code and connections in wokwi for the ultrasonic sensor. Whenever the distance is less than 100cms send an "Alert" to IBM cloud and display in the device recent events ?

Name:-B Bavin Kumar

Register Number:- 961619104015

College:-Marthandam College Of Engineering and Technology

Code

```
#include <WiFi.h>
#include <PubSubClient.h>
void callback(char* subscribtopic, byte* payload, unsigned int
payloadLength);
//-----credentials of IBM Accounts-----
#define ORG "ib3zmz"//IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_Controller"//Device type mentioned in ibm watson IOT
Platform
#define DEVICE_ID "ESP32"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribtopic[] = "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);
const int trigPin = 5;
const int echoPin = 18;
#define SOUND_SPEED 0.034
long duration;
float distance;
void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  wificonnect();
  mqttconnect();
```

```

}
void loop()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * SOUND_SPEED/2;
  Serial.print("Distance (cm): ");
  Serial.println(distance);
  if(distance<100)
  {
    Serial.println("ALERT!!");
    delay(1000);
    PublishData(distance);
    delay(1000);
    if (!client.loop()) {
      mqttconnect();
    }
  }
  delay(1000);
}

void PublishData(float dist) {
  mqttconnect();
  String payload = "{\"Distance\": ";
  payload += dist;
  payload += ", \"ALERT!!\": \"\" \"Distance less than 100cms\"";
  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 client to ");
    Serial.println(server);
    while (!!!client.connect(clientId, authMethod, token)) {

```

```

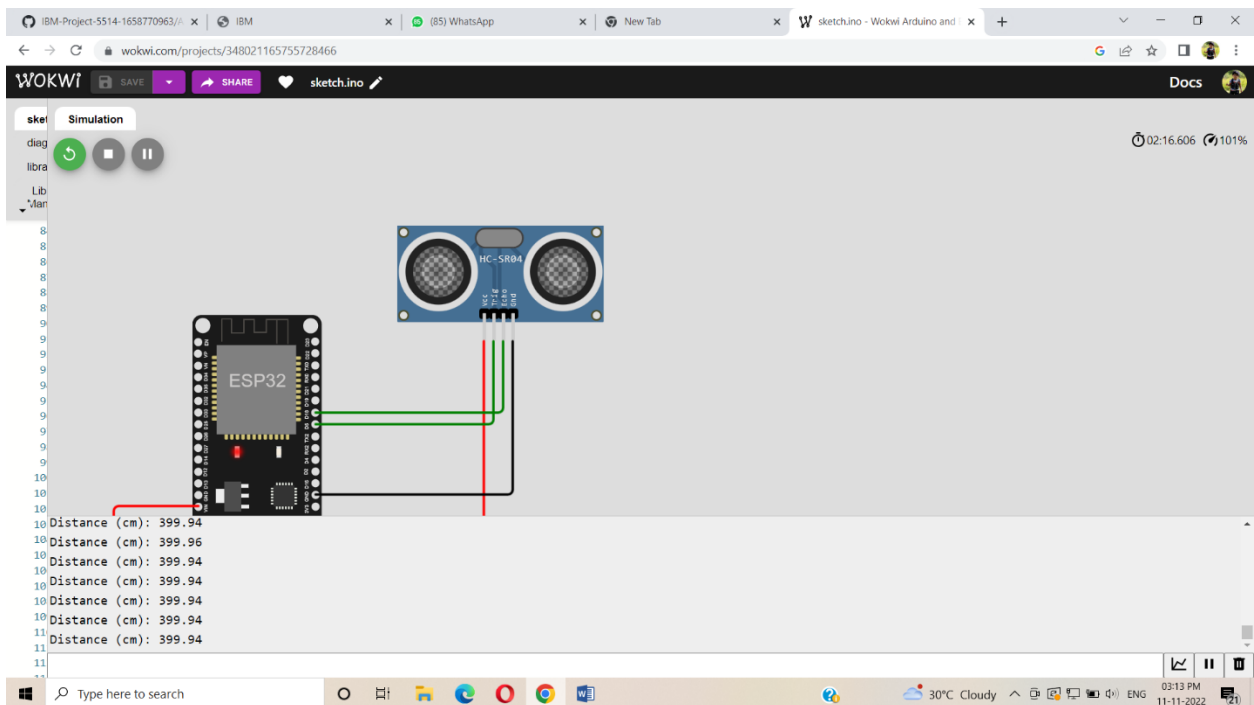
Serial.print(".");
delay(500);
}
initManagedDevice();
Serial.println();
}
}
void wificonnect()
{
Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
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);
data3="";
}

```

Diagram.json:

```
{
  "version": 1,
  "author": "Bavin",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 2.59, "left": -112.14,
    "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top": -68.86, "left": 55.7,
    "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v198.61", "h-304.12", "v-60.29" ]
  ],
  [ "ultrasonic1:TRIG", "esp:D5", "green", [ "v0" ] ],
  [ "ultrasonic1:ECHO", "esp:D18", "green", [ "v0" ] ],
  [ "esp:GND.1", "ultrasonic1:GND", "black", [ "h0" ] ]
  ]
}
```

OUTPUT



CLOUD OUTPUT

The screenshot displays the Wokwi Cloud Output interface. On the left is a dark sidebar with icons for various functions. The main area has a top navigation bar with tabs: 'Browse', 'Action', 'Device Types', and 'Interfaces'. A blue 'Add Device +' button is in the top right. Below the tabs, there's a sub-navigation bar with 'Identity', 'Device Information', 'Recent Events' (which is selected), 'State', and 'Logs'. A close button 'X' is on the far right of this bar. The 'Recent Events' section contains a text description: 'The recent events listed show the live stream of data that is coming and going from this device.' Below this is a table with four columns: 'Event', 'Value', 'Format', and 'Last Received'. The table lists four events, all labeled 'event_1', with JSON values containing distance and alert information, all in 'json' format, and all received 'a few seconds ago'.

Event	Value	Format	Last Received
event_1	{"distance":7,"Alert":"","Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"","Distance less than 10"}	json	a few seconds ago
event_1	{"distance":8,"Alert":"","Distance less than 10"}	json	a few seconds ago
event_1	{"distance":9,"Alert":"","Distance less than 10"}	json	a few seconds ago

Link: <https://wokwi.com/projects/347923547949105746>