Assignment -4

Using Ultrasonic sensors to detect distance

Assignment	Date 24 October 2022
Student Name	Bharath Kumar D P
Team ID	PNT2022TMID35177
Project Name Project	Smart Solution for Railways

Question

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

Wokwi simulation link
https://wokwi.com/projects/348771049151660628

Circuit:

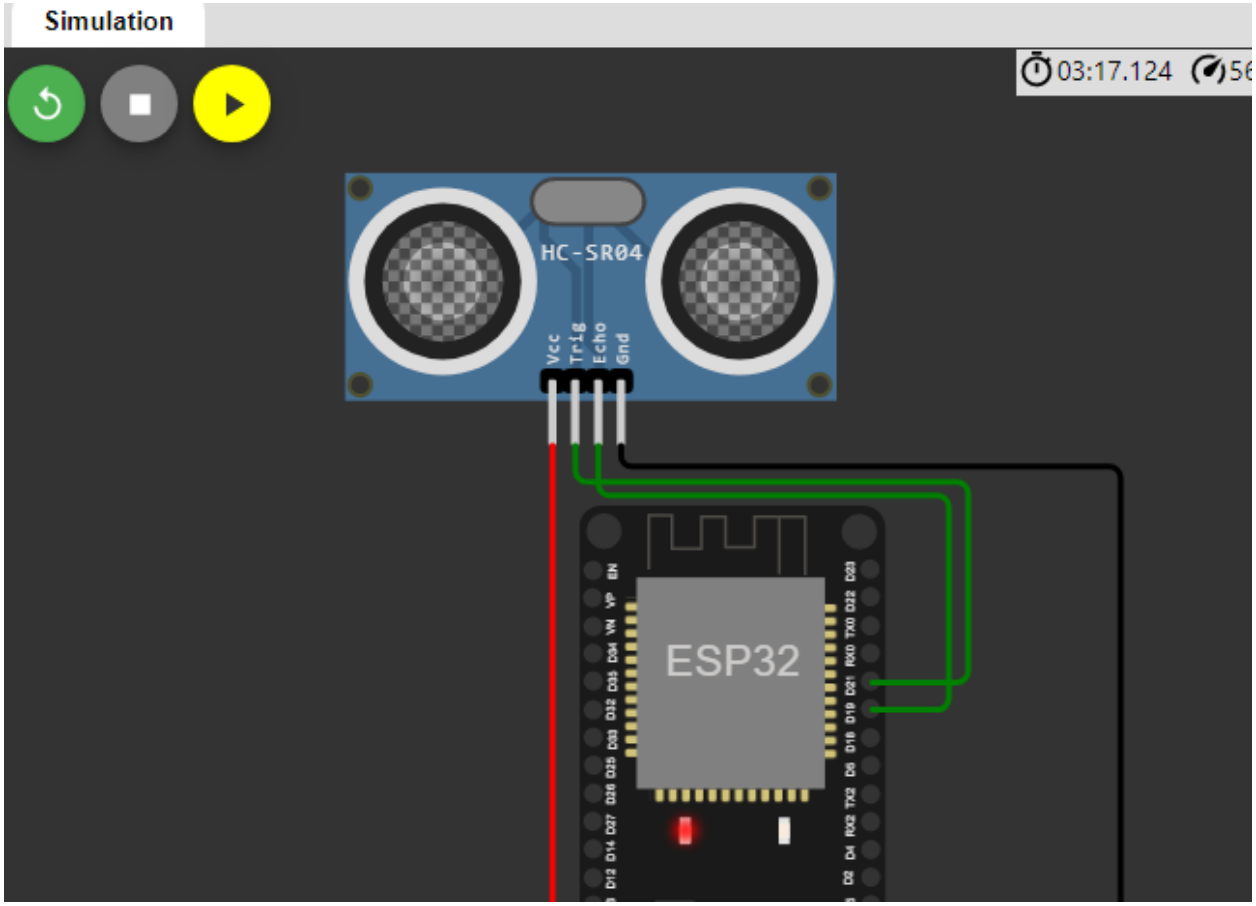


diagram.json

```
{
  "version": 1,
  "author": "Bharath kumar",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top":
70, "left": 40.67, "attrs": {} },
    { "type": "wokwi-hc-sr04", "id": "ultrasonic1", "top":
-43.37, "left": -40.17, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
```

```
    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],
    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v7.24",
"h172.55", "v159.33" ] ],
    [ "ultrasonic1:TRIG", "esp:D21", "green", [ "v12.57",
"h135.67", "v89.33" ] ],
    [ "ultrasonic1:ECHO", "esp:D19", "green", [ "v17.24",
"h121.11", "v74", "h-0.67" ] ]
  ]
}
```

Code:

```
#include <WiFi.h>
#include "PubSubClient.h"

#define ORG "1bfyv3"
#define DEVICE_TYPE "ESP32_Controller"
#define DEVICE_ID "Node29"

char deviceID[] = "d:"ORG":"DEVICE_TYPE":"DEVICE_ID;
char username[] = "use-token-auth";
char password[] = "XO_J!5Cx?@N0Bt1OkY";
char serverURL[] =
ORG".messaging.internetofthings.ibmcloud.com";
int port = 1883;

char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribeTopic[] = "iot-2/cmd/Sub/fmt/String";

String lineBreak = "-----------";

WiFiClient wifiClient;
```

```
PubSubClient pubSubClient(serverURL,
  port,
  [](char* topic, byte* payload, unsigned int length) {
    Serial.println("Callback Invoked!");
    for (int i = 0; i < length; ++i)
      Serial.print((char)payload[i]);
  },
  wifiClient
);

int trigPin = 21;
int echoPin = 19;

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  connectWiFi();
  connectMQTT();
}

void loop() {
  refreshMQTTConn();

  float distance = getUltraSonicDistance();
  if (distance < 100) {
    Serial.print("ALERT! Distance at: ");
    Serial.println(distance);
    publishData(distance);
  }
```

```
  delay(5000);
}

float getUltraSonicDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);
  return (float) pulseIn(echoPin, HIGH) / 58.0f;
}

void publishData(float distance) {
  refreshMQTTConn();
  String payload = "{";
  payload += "\"Message\": \"Distance less than 100cm\"";
  payload += ", ";
  payload += "\"Distance\": ";
  payload += distance;
  payload += "}";

  if (pubSubClient.publish(publishTopic,
(char*)payload.c_str())) {
    Serial.println("Publish OK");
  }
  else Serial.println("Publish FAILED");
}

void connectWiFi() {
  WiFi.begin("Wokwi-GUEST", "", 6);
```

```
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
  Serial.println(lineBreak);
}

void connectMQTT() {
  Serial.print("Connecting to IBM Watson @ ");
  Serial.print(serverURL);
  while (!pubSubClient.connect(deviceID, username, password))
{
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected to IBM Watson!");

  if (pubSubClient.subscribe(subscribeTopic)) {
    Serial.println("Subscribed to CMD");
  }
  else {
    Serial.println("Subscribe FAILED");
  }
  Serial.println(lineBreak);
}

void refreshMQTTConn() {
  if (!pubSubClient.loop()) {
```

```
    connectMQTT();
  }
}

#include <WiFi.h>
#include "PubSubClient.h"

#define ORG "1bfyv3"
#define DEVICE_TYPE "ESP32_Controller"
#define DEVICE_ID "Node29"

char deviceID[] = "d:"ORG":"DEVICE_TYPE":"DEVICE_ID;
char username[] = "use-token-auth";
char password[] = "XO_J!5Cx?@N0Bt1OkY";
char serverURL[] =
ORG".messaging.internetofthings.ibmcloud.com";
int port = 1883;

char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribeTopic[] = "iot-2/cmd/Sub/fmt/String";

String lineBreak = "-----------";

WiFiClient wifiClient;
PubSubClient pubSubClient(serverURL,
  port,
  [](char* topic, byte* payload, unsigned int length) {
    Serial.println("Callback Invoked!");
    for (int i = 0; i < length; ++i)
      Serial.print((char)payload[i]);
  },
  wifiClient
```

```
);

int trigPin = 21;
int echoPin = 19;

void setup() {
  Serial.begin(115200);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  connectWiFi();
  connectMQTT();
}

void loop() {
  refreshMQTTConn();

  float distance = getUltraSonicDistance();
  if (distance < 100) {
    Serial.print("ALERT! Distance at: ");
    Serial.println(distance);
    publishData(distance);
  }

  delay(5000);
}

float getUltraSonicDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
```

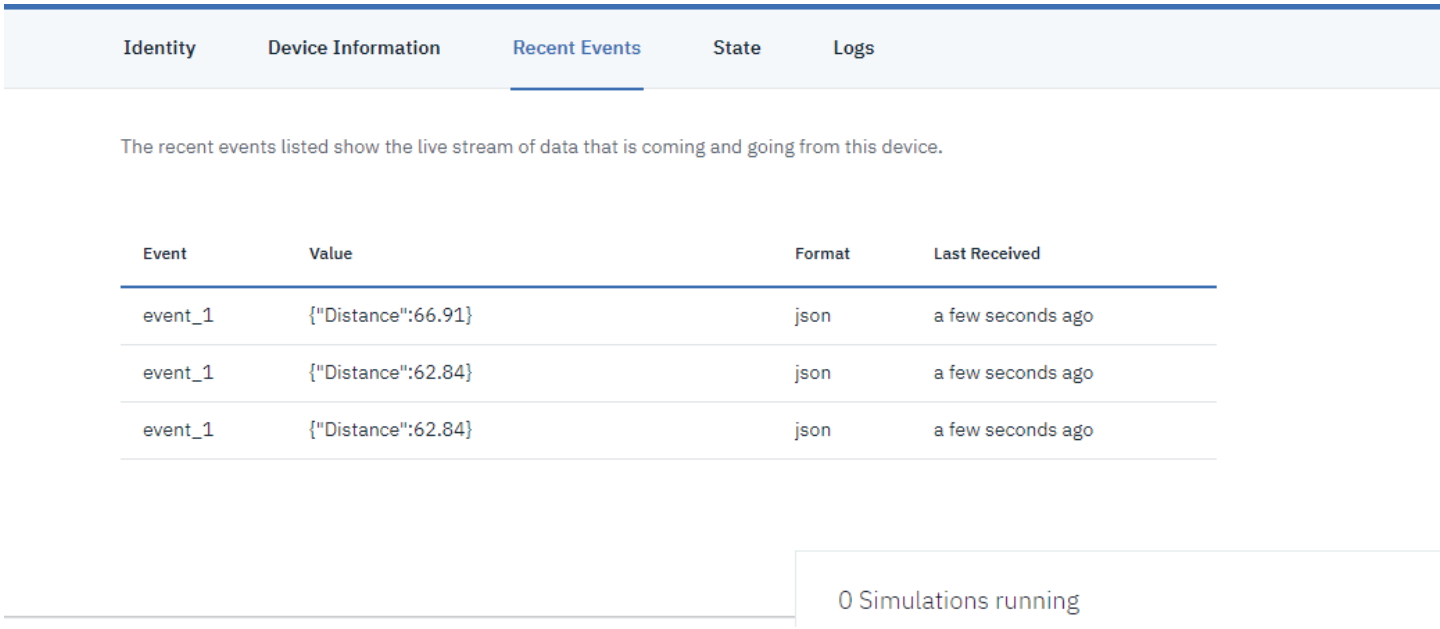
```
  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);
  return (float) pulseIn(echoPin, HIGH) / 58.0f;
}

void publishData(float distance) {
  refreshMQTTConn();
  String payload = "{";
  payload += "\"Message\": \"Distance less than 100cm\"";
  payload += ", ";
  payload += "\"Distance\": ";
  payload += distance;
  payload += "}";

  if (pubSubClient.publish(publishTopic,
(char*)payload.c_str())) {
    Serial.println("Publish OK");
  }
  else Serial.println("Publish FAILED");
}

void connectWiFi() {
  WiFi.begin("Wokwi-GUEST", "", 6);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
```

```
  Serial.println(lineBreak);
}

void connectMQTT() {
  Serial.print("Connecting to IBM Watson @ ");
  Serial.print(serverURL);
  while (!pubSubClient.connect(deviceID, username, password))
{
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected to IBM Watson!");

  if (pubSubClient.subscribe(subscribeTopic)) {
    Serial.println("Subscribed to CMD");
  }
  else {
    Serial.println("Subscribe FAILED");
  }
  Serial.println(lineBreak);
}

void refreshMQTTConn() {
  if (!pubSubClient.loop()) {
    connectMQTT();
  }
}
```

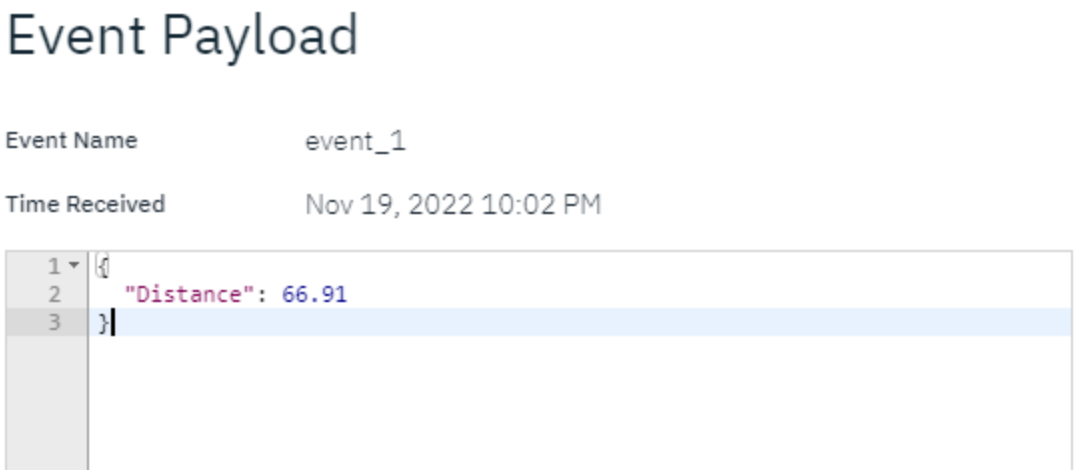
Output:

```
Connecting to IBM Watson @ 1bfyv3.messaging.internetofthings.ibmcloud.com
Connected to IBM Watson!
Subscribed to CMD
-----------
ALERT! Distance at: 48.64
Publish OK
ALERT! Distance at: 48.64
Publish OK
ALERT! Distance at: 48.66
Publish OK
```

IBM Watson output:



Data format



Connection logs