**Assiggnment 4**

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 diplay in the device recent events.

Upload document with wokwi share link and images od IBM cloud.

**code:**

**esp32-dht22.ino**

#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;

String data3;

#define ORG "s8ov1q"

#define DEVICE\_TYPE "gayathri"

#define DEVICE\_ID "gayathri123"

#define TOKEN "123456789"

#define speed 0.034

#define led 14

char server[] = ORG ".messaging.internetofthings.ibmcloud.com";

char publishTopic[] = "iot-2/evt/Gayathri/fmt/json";

char topic[] = "iot-2/cmd/home/fmt/String";

char authMethod[] = "use-token-auth";

char token[] = TOKEN;

char clientId[] = "d:" ORG ":" DEVICE\_TYPE ":" DEVICE\_ID;

PubSubClient client(server, 1883, wifiClient);

void publishData();

const int trigpin=5;

const int echopin=18;

String command;

String data="";

long duration;

float dist;

void setup()

{

**Serial**.begin(115200);

  pinMode(led, OUTPUT);

  pinMode(trigpin, OUTPUT);

  pinMode(echopin, INPUT);

  wifiConnect();

  mqttConnect();

}

void loop() {

  bool isNearby = dist < 100;

  digitalWrite(led, isNearby);

  publishData();

  delay(500);

  if (!client.loop()) {

    mqttConnect();

  }

}

void wifiConnect() {

**Serial**.print("Connecting to "); **Serial**.print("Wifi");

  WiFi.begin("Wokwi-GUEST", "", 6);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

**Serial**.print(".");

  }

**Serial**.print("WiFi connected, IP address: "); **Serial**.println(WiFi.localIP());

}

void mqttConnect() {

  if (!client.connected()) {

**Serial**.print("Reconnecting MQTT client to "); **Serial**.println(server);

    while (!client.connect(clientId, authMethod, token)) {

**Serial**.print(".");

      delay(500);

    }

    initManagedDevice();

**Serial**.println();

  }

}

void initManagedDevice() {

  if (client.subscribe(topic)) {

    // Serial.println(client.subscribe(topic));

**Serial**.println("IBM subscribe to cmd OK");

  } else {

**Serial**.println("subscribe to cmd FAILED");

  }

}

void publishData()

{

  digitalWrite(trigpin,LOW);

  digitalWrite(trigpin,HIGH);

  delayMicroseconds(10);

  digitalWrite(trigpin,LOW);

  duration=pulseIn(echopin,HIGH);

  dist=duration\*speed/2;

  if(dist<100){

    String payload = "{\"Alert Distance\":";

    payload += dist;

    payload += "}";

**Serial**.print("\n");

**Serial**.print("Sending payload: ");

**Serial**.println(payload);

      if(client.publish(publishTopic, (char\*) payload.c\_str())) {

**Serial**.println("Warning crosses 110cm -- it automaticaly of the loop");

      digitalWrite(led,HIGH);

    }

  }

    if(dist>101 && dist<111){

    String payload = "{\"Normal Distance\":";

    payload += dist;

    payload += "}";

**Serial**.print("\n");

**Serial**.print("Sending payload: ");

**Serial**.println(payload);

    }

  }

  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++){

    dist += (char)payload[i];

  }

**Serial**.println("data:"+ data3);

  if(data3=="lighton"){

**Serial**.println(data3);

    digitalWrite(led,HIGH);

  }

  data3="";

}

**diagram.json**

{

  "version": 1,

  "author": "Uri Shaked",

  "editor": "wokwi",

  "parts": [

    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 92.67, "left": 45.33, "attrs": {} },

    {

      "type": "wokwi-led",

      "id": "led1",

      "top": 194.54,

      "left": 309.26,

      "attrs": { "color": "red" }

    },

    {

      "type": "wokwi-hc-sr04",

      "id": "ultrasonic1",

      "top": 60.71,

      "left": 185.64,

      "attrs": { "distance": "90" }

    },

    {

      "type": "wokwi-resistor",

      "id": "r1",

      "top": 269.89,

      "left": 260.39,

      "attrs": { "value": "100" }

    }

  ],

  "connections": [

    [ "esp:TX0", "$serialMonitor:RX", "", [] ],

    [ "esp:RX0", "$serialMonitor:TX", "", [] ],

    [ "ultrasonic1:TRIG", "esp:D5", "yellow", [ "v0" ] ],

    [ "ultrasonic1:ECHO", "esp:D18", "magenta", [ "v0" ] ],

    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v0" ] ],

    [ "ultrasonic1:GND", "esp:GND.1", "black", [ "v0" ] ],

    [ "esp:D12", "r1:2", "gold", [ "h156.9", "v62.96" ] ],

    [ "led1:C", "esp:GND.2", "black", [ "v0" ] ],

    [ "r1:1", "led1:A", "purple", [ "v28.12", "h94" ] ],

    [ "esp:D12", "esp:D14", "green", [ "h0" ] ]

  ]

}

**libraries.txt**

# Wokwi Library List

# See https://docs.wokwi.com/guides/libraries

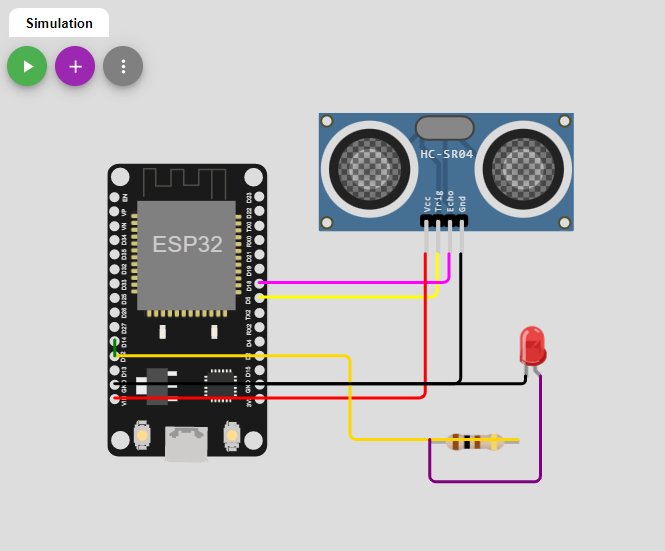
DHT sensor library for ESPx

PubSubClient

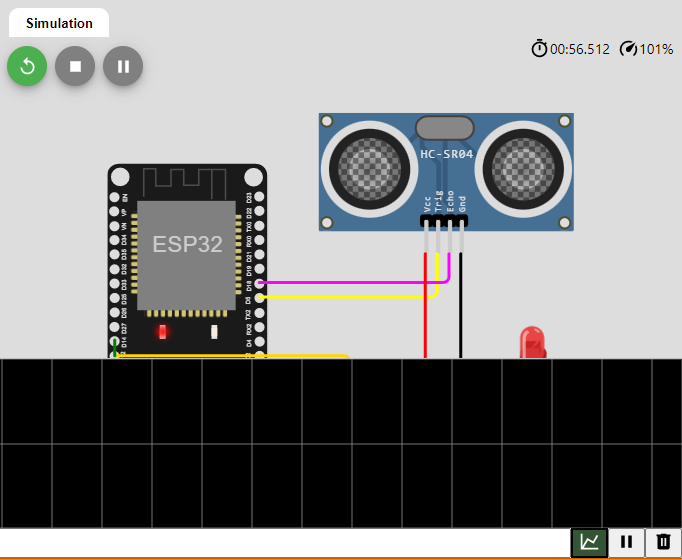
**libraries manager**

Installed Libraries

* DHT sensor library for ESPx
* PubSubClient



**output:**

****