IOT ASSIGNMENT 3

```
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt

#define LED 5
#define LED2 4
#define LED3 2
int LDR = 32;
int LDRReading = 0;
int threshold_val = 800;
int lEDBrightness = 0;
int flag=0;

void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "stuloy"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678"     //Token
String data3;
float h, t;


//-------- 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 format in which data to be send
char subscribetopic[] = "iot-2/cmd/test/fmt/String";// cmd  REPRESENT command
type AND COMMAND IS TEST OF FORMAT STRING
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 passing parameter like server id,portand wificredential
void setup()// configureing the ESP32
```

```
{
  Serial.begin(115200);

  pinMode(LED,OUTPUT);
  pinMode(LED2,OUTPUT);
  pinMode(LED3,OUTPUT);
  delay(10);
  Serial.println();
  wificonnect();
  mqttconnect();
}

void loop()// Recursive Function
{



  //PublishData(t, h);
  //delay(1000);

  /* LDRReading = analogRead(LDR);
  Serial.print("LDR READING:");
  Serial.println(LDRReading);

  if (LDRReading >threshold_val){
  lEDBrightness = map(LDRReading, 0, 1023, 0, 255);
  Serial.print("LED BRIGHTNESS:");
  Serial.println(lEDBrightness);

  analogWrite(LED, lEDBrightness);
  analogWrite(LED2, lEDBrightness);
  analogWrite(LED3, lEDBrightness);
  }
  else{
  analogWrite(LED, 0);
  analogWrite(LED2, 0);
  analogWrite(LED3, 0);
  }

  delay(300);*/

  if (!client.loop()) {
    mqttconnect();
  }
}
```

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

/*void PublishData(float temp, float humid) {
  mqttconnect();//function call for connecting to ibm*/
  /*
     creating the String in in form JSon to update the data to ibm cloud
  */
  /*String payload = "{\"temperature\":";
  payload += temp;
  payload += "," "\"humidity\":";
  payload += humid;
  payload += "}";


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


  if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully 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());
}

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=="lighton1")
  {
Serial.println(data3);
digitalWrite(LED,HIGH);

  }
```

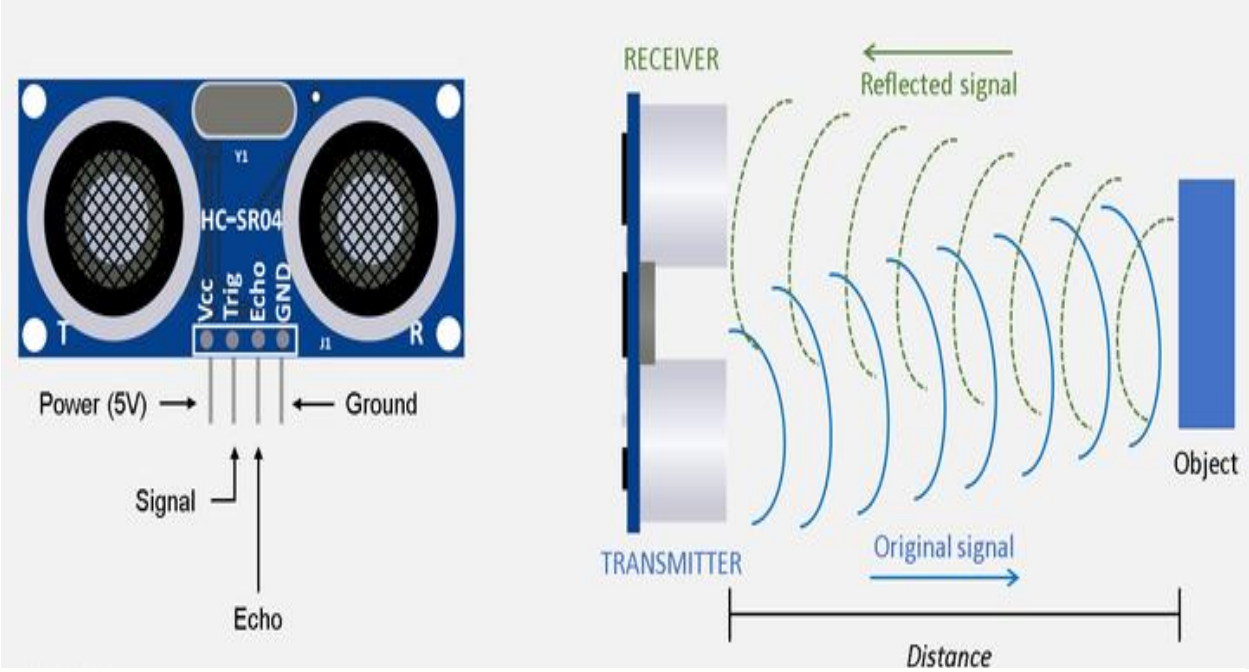
```
  else if(data3=="lightoff1")
  {
Serial.println(data3);
digitalWrite(LED,LOW);


  }
  else if(data3=="lighton2")
  {
Serial.println(data3);
digitalWrite(LED2,HIGH);

  }

  else if(data3=="lightoff2")
  {
Serial.println(data3);
digitalWrite(LED2,LOW);

  }
  else if(data3=="lighton3")
  {
Serial.println(data3);
digitalWrite(LED3,HIGH);

  }

  else if(data3=="lightoff3")
  {
Serial.println(data3);
digitalWrite(LED3,LOW);

  }
data3="";



}
```

DIAGRAM CODE:

```
  "version": 1,
  "author": "CHANDRIKA",
  "editor": "wokwi",
  "parts": [
```

```
{
  "type": "wokwi-esp32-devkit-v1",
  "id": "esp",
  "top": 23.61,
  "left": 45.52,
  "attrs": { "builder": "rust-std-esp" }
},
{
  "type": "wokwi-photoresistor-sensor",
  "id": "ldr1",
  "top": 182.59,
  "left": -351.38,
  "attrs": {}
},
{
  "type": "wokwi-led",
  "id": "led2",
  "top": 11.8,
  "left": 198.75,
  "attrs": { "color": "red" }
},
{
  "type": "wokwi-resistor",
  "id": "r1",
  "top": 94.9,
  "left": 177.43,
  "attrs": { "value": "1000" }
},
{
  "type": "wokwi-resistor",
  "id": "r2",
  "top": 136.26,
  "left": 219.28,
  "attrs": { "value": "1000" }
},
{
  "type": "wokwi-led",
  "id": "led1",
  "top": 41.97,
  "left": 278.55,
  "attrs": { "color": "yellow" }
},
{
  "type": "wokwi-led",
  "id": "led3",
```

```
      "top": 101.84,
      "left": 324.06,
      "attrs": { "color": "limegreen" }
    },
    {
      "type": "wokwi-resistor",
      "id": "r3",
      "top": 181.94,
      "left": 253.07,
      "attrs": { "value": "1000" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "esp:D32", "ldr1:AO", "green", [ "h0" ] ],
    [ "ldr1:DO", "esp:D15", "#8f4814", [ "h0" ] ],
    [ "ldr1:GND", "esp:D19", "gold", [ "h0" ] ],
    [ "ldr1:VCC", "esp:3V3", "red", [ "h0" ] ],
    [ "led2:C", "esp:GND.1", "magenta", [ "v0" ] ],
    [ "esp:D5", "r1:1", "green", [ "h0" ] ],
    [ "r1:2", "led2:A", "green", [ "v0" ] ],
    [ "esp:D4", "r2:1", "green", [ "h0" ] ],
    [ "led1:A", "r2:2", "green", [ "v0" ] ],
    [ "led1:C", "esp:GND.1", "magenta", [ "v0" ] ],
    [ "esp:D2", "r3:1", "green", [ "h0" ] ],
    [ "r3:2", "led3:A", "green", [ "v0" ] ],
    [ "led3:C", "esp:GND.1", "green", [ "v0" ] ]
  ],
  "serialMonitor": { "display": "terminal" },
  "dependencies": {}
}
```




```
[package]
name = "rust-project-esp32"
version = "0.1.0"
authors = ["Sergio Gasquez <sergio.gasquez@gmail.com>"]
edition = "2021"
resolver = "2"

[profile.release]
opt-level = "s"

[profile.dev]
debug = true    # Symbols are nice and they don't increase the size on Flash
opt-level = "z"

[features]
pio = ["esp-idf-sys/pio"]

[dependencies]
esp-idf-sys = { version = "0.32.1", features = ["binstart"] }
esp-idf-hal = "0.40.1"
esp-idf-svc = "0.45.0"

[build-dependencies]
embuild = "0.31.1"
```

ultrasonic_sesnor_range_calculation_using_arduino | Arduino 1.8.8

File Edit Sketch Tools Help

ultrasonic_sesnor_range_calculation_using_arduino

```
#include <Mouse.h>

const int trigpin= 8;
const int echopin= 7;
long duration;
int distance;
void setup(){
  pinMode(trigpin,OUTPUT);
  pinMode(echopin,INPUT);
  Serial.begin(9600);
}

void loop(){
  digitalWrite(trigpin,HIGH);
  delayMicroseconds(10);
  digitalWrite(trigpin,LOW);
  duration=pulseIn(echopin,HIGH);
  distance = duration*0.034/2;
  Serial.println(distance);
}
```

Done uploading.

```
Sketch uses 2968 bytes (9%) of program storage space. Maximum is 32256 bytes.
Global variables use 188 bytes (9%) of dynamic memory, leaving 1860 bytes for local variables. Maximum is 2048 bytes.
```

16

processing_ultrasonic_range_calculation | Processing 3.4

File Edit Sketch Debug Tools Help

processing_ultrasonic_range_calculation

```
import processing.serial.*;
Serial myPort;
String data="" ;
PFont  myFont;

void setup(){
  size(1366,900); // size of processing window
  background(0);// setting background color to black
  myPort = new Serial(this, "COM3", 9600);
  myPort.bufferUntil('\n');
}

void draw(){
  background(0);
  textAlign(CENTER);
  fill(255);
  text(data,820,400);
  textSize(100);
  fill(#4B5DCE);
  text("             Distance :        cm",450,400);
  noFill();
  stroke(#4B5DCE);
}

void serialEvent(Serial myPort){
   data=myPort.readStringUntil('\n');
}
```