

## Assignment 3

### Recent Events:

#### Recent Events

The recent events listed show the live stream of data that is coming and going from this device.

Event	Value	Format	Last Received
Data	{"distance":30}	json	a few seconds ago
Data	{"distance":81}	json	a few seconds ago
Data	{"distance":81}	json	a few seconds ago
Data	{"distance":81}	json	a few seconds ago
Data	{"distance":81}	json	a few seconds ago

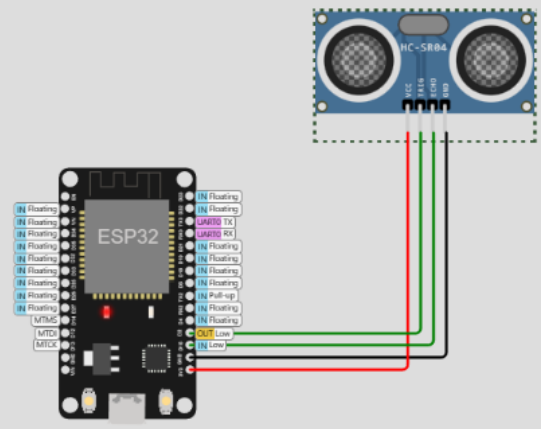
### Circuit Diagram:

Simulation

01:41.866 99%

Editing Ultrasonic Distance Sensor

Distance:  299cm



Sending payload: {"distance":30}

Publish ok

distance:30

Sending payload: {"distance":30}

Publish ok

distance:314

distance:314

## Code:

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

const int echo=15;
const int trig=2;

Ultrasonic ultrasonic(trig,echo);

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

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

#define ORG "51y8x8" //IBM ORGANITION ID
#define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "wokwi" //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 subscribtopic[] = "iot-2/cmd/command/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);
    delay(10);
    Serial.println();
}
```

```

wificonnect();
mqttconnect();
}

void loop()// Recursive Function
{

    long distance = ultrasonic.read();
    Serial.print("distance:");
    Serial.println(distance);

    if (distance<=100){

        PublishData(distance);
    }
    delay(1000);
    if (!client.loop()) {
        mqttconnect();
    }
}

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

void PublishData(long distance) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */
    String payload = "{\"distance\":";
    payload += distance;
    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=="lighton")
{
Serial.println(data3);
}
else
{
Serial.println(data3);
}
data3="";
}
```