

NAME : SAKTHIVEL K

REG NO : 210619205040

PROGRAM

Smart Waste Management System for Metropolitan Cities

ASSIGNMENT 4:

Write code and connections in wokwi for ultrasonic sensors. Whenever distance is less than 100 cms

send "alert" to ibm cloud and display in device recent events. Upload document with wokwi share link and images of ibm cloud.

CODE:

```
#include <WiFi.h>

#include <PubSubClient.h>

WiFiClient wifiClient;

String data3;

#define ORG "ztcz45"

#define DEVICE_TYPE "naveen"

#define DEVICE_ID "naveen123"

#define TOKEN "123456789"

#define speed 0.034 #define led 14 char server[] = ORG

".messaging.internetofthings.ibmcloud.com"; char

publishTopic[] = "iot-2/evt/Data/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);
  ...

```

[10:32 pm, 23/10/2022] Gogul B.E CSE:

```

} 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(
e(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 = "{\Normal
Distance\":"; payload += dist; payload
+= "}";
Serial.print("\n");
Serial.print("Sending payload: ");
Serial.println(payload); if (client.publish(publishTopic,
(char*) payload.c_str())) {
Serial.println("Publish OK");
}
}

```

```

}

if(dist>101 && dist<111){

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);

}else {

Serial.println("Publish 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++){ dist +=

(char)payload[i];

}

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

if(data3=="lighton"){

```

```

Serial.println(data3);

digitalWrite(led,HIGH);

}

data3="";

}

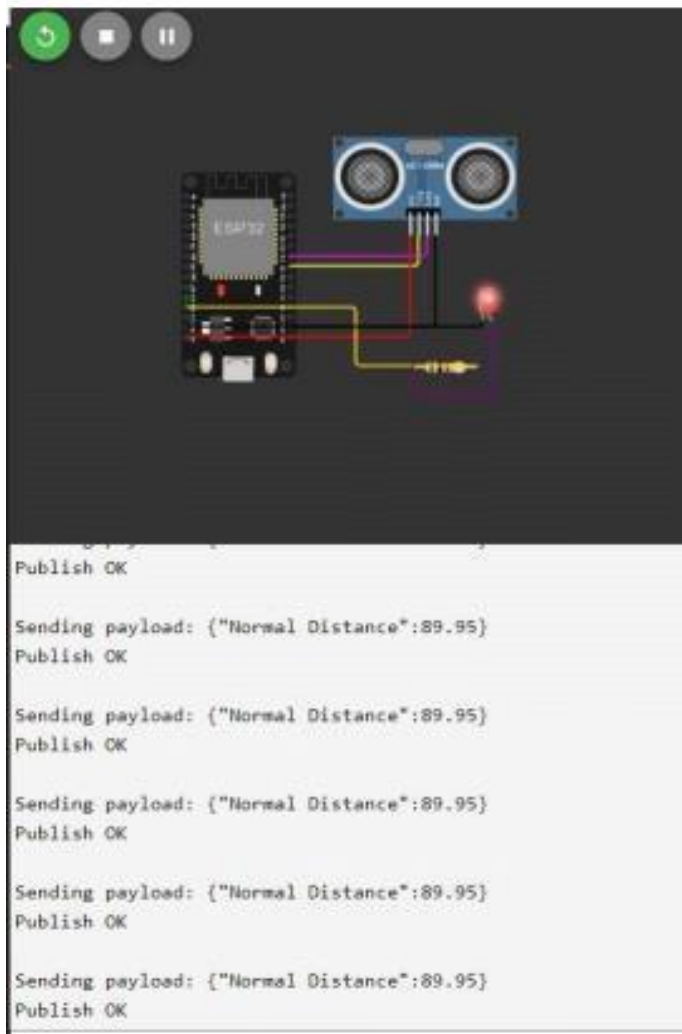
```

output:

The image shows a screenshot of the Wokwi IDE and the TTPM Watson IoT Platform. The Wokwi IDE on the left displays a C++ program for an Arduino Uno connected to an Ultrasonic Distance Sensor. The program sends distance data to the TTPM Watson IoT Platform via MQTT. The TTPM Watson IoT Platform on the right shows the device 'esp8266-133' in a 'Connected' state. Below the device information, there is a table of recent events showing distance readings.

Event	Value	Format	Last Received
Data	[{"Alert distance": 218.96}]	json	4 Feb 2020
Data	[{"Alert distance": 218.96}]	json	4 Feb 2020
Data	[{"Alert distance": 218.96}]	json	4 Feb 2020
Data	[{"Alert distance": 218.96}]	json	4 Feb 2020

1. When distance under 100 cm it will show normal distance.



2. When distance cross 100 cm it will show ALERT warning message distance

3. When it cross above 110 cm it today move to iff state once it
reduce to 110 it on again

The screenshot displays the Wokwi IDE interface. The main editor shows a C++ program for an ESP8266 module. The code includes headers for `WiFiClient`, `WiFiServer`, and `Ultrasonic`. It sets up a web server on port 80 and a distance sensor. The `setup` function initializes the sensor and the server. The `loop` function checks for incoming connections and sends the current distance reading to a web page. The right sidebar shows the 'Device Simulator' with a visual representation of the ESP8266 module and its connections.

Organization ID : ztcz45

Device ID : THOL123

Identity	Device Information	Recent Events	State	Logs
<p>The recent events listed show the live stream of data that is coming and going from this device.</p>				
Event	Value	Format	Last Received	
Data	{"Normal Distance":89.95}	json	a few seconds ago	
Data	{"Normal Distance":89.95}	json	a few seconds ago	
Data	{"Normal Distance":89.95}	json	a few seconds ago	
Data	{"Normal Distance":89.95}	json	a few seconds ago	
Data	{"Normal Distance":89.95}	json	a few seconds ago	