

ASSIGNMENT -3

QUESTION:

Build wowki product, use ultrasonic sensor and detect the distance from the object. whenever distance is less than 100cms upload the value to the ibm cloud. in recent device events upload the data from wowki.

CODE :

```
#include <WiFi.h> // library for WIFI

#include <PubSubClient.h> // library for MQTT

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

#define ORG "rwazv5" // IBM organisation id
#define DEVICE_TYPE "NodeRed" // Device type mentioned in
ibm watson iot platform #define DEVICE_ID "12345" // Device
ID mentioned in ibm watson iot platform #define TOKEN
"vC@S3TBre6(97jAOJ_" // Token
#define speed
0.034 #define
led 14 String
data3;
int LED = 4;

//.....customise above values .....

char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // server name
char publishTopic[] = "iot-2/evt/sreedhar/fmt/json"; // topic name and type
of event perform and format in which data to be send
char topic[] = "iot-2/cmd/led/fmt/String"; // cmd Represent type and
command is test format of strings
char authMethod[] = "use-token-auth"; //
authentication method char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client id

//.....

WiFiClient wifiClient; // creating instance for wificlient
PubSubClient client(server, 1883, wifiClient); // calling the predefined client id by
passing parameter like server id, port and wifi credential
```

```
const int
trigpin=5; const
int echopin=18;
String
command;
String
data="";
```

```
long
duration;
float dist;
```

```
void setup()
{
  Serial.begin(11520
0); pinMode(led,
OUTPUT);
pinMode(trigpin,
OUTPUT);
pinMode(echopin,
INPUT);
wifiConnect();
mqttConnect();
}
```

```
void loop() { bool isNearby
= dist <
  100;
digitalWrite(led,
isNearby);
```

```
publishD
ata();
delay(50
0);
```

```

if(!client.loop())
{
  mqttConnect();    // function call to connect to ibm
}
}

/*.....retrieving to cloud

.....*/
void wifiConnect()
{
  Serial.print("Connecti
ng 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(to
pic))
{
  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)
{
digitalWrite(LED,HIGH);
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())) // if data is uploaded to
cloud successfully,prints publish ok else prints publish failed
```

```

{
  Serial.println("Publish OK");
}
}
if(dist>100)
{
  digitalWrite(LED,HIGH);
  String payload =
  "{\"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");
  }
  else
  {
    digitalWrite(LED,LOW);
    Serial.println("Publish FAILED");
  }

}
}

```

OUTPUT :

Code simulation on wokwi

The screenshot shows the Wokwi IDE interface. On the left, the code for `esp32-dht22.ino` is displayed, including headers, credentials, and MQTT setup. The right side shows a 3D simulation of the ESP32-DHT22 module. The console output indicates successful WiFi connection and MQTT client reconnection.

```

1 #include <WiFi.h> // WiFi library
2 #include <PubSubClient.h> // MQTT library
3
4 //----- credentials of IBM Accounts -----
5
6 #define ORG "rwazv5" // IBM organisation id
7 #define DEVICE_TYPE "NodeRed" // Device type mentioned in IBM Cloud IoT
8 #define DEVICE_ID "12345" // Device ID mentioned in IBM Cloud IoT
9 #define TOKEN "vc@S3TBre6(97jAOJ_" // Token
10 #define speed 0.034
11 #define led 14
12 String data3;
13 int LED = 4;
14
15 //----- customise above values -----
16
17 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // server
18 char publishTopic[] = "iot-2/evt/sreedhar/fmt/json"; // publish topic
19 char topic[] = "iot-2/cmd/led/fmt/String"; // cmd Represen
20 char authMethod[] = "use-token-auth"; // authentication method
21 char token[] = TOKEN;
22 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //Client id
23
24 //-----
25
26 WiFiClient wifiClient; // creating instance for wifi client
27 PubSubClient client(server, 1883, wifiClient); // calling the predefined
28
29 const int trigpin=5;
30 const int echopin=18;

```

Simulation console output:

```

Connecting to Wifi..Wifi connected, IP address: 10.10.0.2
Reconnecting MQTT client to rwazv5.messaging.internetofthings.ibmcloud.com

```

Data sent to IBM Cloud with distance

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab is selected, displaying a table of live data streams from a device. The table lists events with their values, formats, and reception times.

Event	Value	Format	Last Received
distance	{"distance":141.32}	json	a few seconds ago
distance	{"distance":141.32}	json	a few seconds ago
distance	{"distance":141.32}	json	a few seconds ago
distance	{"distance":141.32}	json	a few seconds ago
distance	{"distance":141.32}	json	a few seconds ago

1 Simulation running