

IBM NALAIYATHIRAN
**SMART FARMER-IOT ENABLED SMART FARMING
APPLICATION**

ASSIGNMENT -4

Title	Smart farmer-IoT enabled smart farming application
Domain	Internet of Things
Team ID	PNT2022TMID44170
Project Name	Project – Smart Farmer-IoT Enabled smartFarming Application

Question:

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 display in the device recent events. Upload document with wokwi share link and images of IBM cloud

CODE :

```
#include<WiFi.h>
#include<PubSubClient.h> void callback(char* subscribetopic, byte* payload,
unsigned int payloadLength);

#define ORG "48uqbr"
#define DEVICE_TYPE
"hasnarahah1009" #define
DEVICE_ID "hasna09"
#define TOKEN "BMqqN8HR9t9" String data3; char server[]
= ORG ".messaging.internetofthings.ibmcloud.com"; char
publishTopic[] = "iot-2/evt/Data/fmt/json"; char
subscribetopic[]
= "iot-2/cmd/test/fmt/String"; char authMethod[] = "use-token-
auth"; char token[] = TOKEN; char clientId[] = "d:" ORG ":"
DEVICE_TYPE ":" DEVICE_ID;
WiFiClient wifiClient;
PubSubClient client(server, 1883,
callback, wifiClient); const int trigPin = 5; const int
echoPin = 18; #define SOUND_SPEED 0.034 long
duration; float distance; void setup(){
Serial.begin(115200); pinMode(trigPin, OUTPUT);
```

```

pinMode(echoPin, INPUT); wifiConnect();
mqttConnect();
}
void loop() {
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW); duration
=pulseIn(echoPin, HIGH); distance
=duration* SOUND_SPEED/2;
Serial.print("Distance (cm):");
Serial.println(distance);
if(distance<100)
{
Serial.println("ALERT!!");
delay(1000);
PublishData(distance);
delay(1000); if
(!client.loop()) {
mqttConnect();
}}
delay(1000)
;}
void PublishData(float dist) { mqttConnect();
String payload="{\"Distance\":"; payload+=dist; payload
+=",\"ALERT!!\": \"\"Distance less than
100cms\""; payload+="}";
Serial.print("Sending payload:");
Serial.println(payload);

if(client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");
} 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()
{
Serial.println();
Serial.print("Connecting to "); WiFi.begin("Wokwi-GUEST", "", 6); 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++)
{
data3 += (char)payload[i];
}
Serial.println("data: "+ data3); data3="";
}

```

Wokwi Link :

<https://wokwi.com/projects/345395196387656275675>

Output and Simulation :

The screenshot displays the Wokwi web IDE interface. On the left, the 'sketch.ino' file is open, showing a C++ program. The code includes the following components:

- Headers:** `<WiFi.h>` for WiFi, `<PubSubClient.h>` for MQTT, and `"DHT.h"` for the DHT11 library.
- Constants:** `DHTPIN` is defined as 15, `DHTTYPE` as DHT22, and `LED` as 2.
- Instance:** A `DHT` object is created with `DHTPIN` and `DHTTYPE`.
- Callback Function:** `void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)` is defined to handle incoming MQTT messages.
- Credentials:** Constants for IBM Watson IoT credentials are defined: `ORG` ("13869j"), `DEVICE_TYPE` ("abcd"), `DEVICE_ID` ("1234"), and `TOKEN` ("12345678").
- Variables:** `String data3;` and `float h, t;` are declared.
- Setup:** A comment indicates where to customize values, followed by the definition of `server`, `publishTopic`, `subscribetopic`, `authMethod`, `token`, and `clientId`.

On the right, the 'Simulation' window shows a visual representation of the hardware: an ESP32 microcontroller, a red LED, and a DHT22 temperature and humidity sensor. Below the simulation, the serial output is visible, showing the program's execution:

```

Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok
temp:24.00
Humid:40.00
Sending payload: {"temp":24.00,"Humid":40.00}
Publish ok

```

Whenever the distance is less than 100 cms send an "alert" to the IBM cloud and display in the device recent events.

Identity

Device Information

Recent Events

State

Logs

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":72.96,"ALERT!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":72.96,"ALERT!":"Distance less than ...	json	a few seconds ago
Data	{"Distance":72.96,"ALERT!":"Distance less than ...	json	a few seconds ago

2001

Disconnected

raspberrypi

Device

Oct

0 Simulations running

Barre car route

50

1 4 3 of 3 Barre

