

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 interface. The top navigation bar includes 'WOKWI', 'SAVE', 'SHARE', and 'Docs' with a 'SIGN IN' button. Below the navigation bar, there are tabs for 'sketch.ino', 'diagram.json', 'libraries.txt', and 'Library Manager'. The 'sketch.ino' tab is active, showing the following code:

```

1 #include <WiFi.h> //library for wifi
2 #include <PubSubClient.h> //library for MQTT
3 #include "DHT.h" // Library for dht11
4 #define DHTPIN 15 // what pin we're connected to
5 #define DHTTYPE DHT22 // define type of sensor DHT 11
6 #define LED 2
7
8 DHT dht (DHTPIN, DHTTYPE); // creating the instance by passing pin and type of
9
10 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
11
12 //-----credentials of IBM Accounts-----
13
14 #define ORG "13869j" //IBM ORGANITION ID
15 #define DEVICE_TYPE "abcd" //Device type mentioned in ibm watson IOT Platform
16 #define DEVICE_ID "1234" //Device ID mentioned in ibm watson IOT Platform
17 #define TOKEN "12345678" //Token
18 String data3;
19 float h, t;
20
21
22 //----- Customise the above values -----
23 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
24 char publishTopic[] = "iot-2/evt/Data/fmt/json"; // topic name and type of event
25 char subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT comma
26 char authMethod[] = "use-token-auth"; // authentication method
27 char token[] = TOKEN;
28 char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
29

```

The 'Simulation' window on the right shows a visual representation of the hardware: an ESP32 microcontroller, a red LED, a resistor, and a DHT22 temperature and humidity sensor. Below the simulation, the output log shows the following sequence of events:

```

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

