

## Smart Farmer - IoT Enabled Smart Farming Application

Batch No	PNT2022TMID21910
Student Roll number	142219205045

### Question-1:

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 1 :

```
#include <WiFi.h>
#include <PubSubClient.h> void callback(char* subscribetopic, byte* payload,
unsigned int payloadLength);
#define ORG "92zbfc"
#define DEVICE_TYPE "esp32"
#define DEVICE_ID "12345"
#define TOKEN "12345678" 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!!\":"; payload += "\"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/347141309997253202>

Output and Simulation :

The screenshot shows the Wokwi IDE interface. On the left, the sketch code is displayed, which includes an ESP32 configuration and an Ultrasonic Distance Sensor (HC-SR04) connected to pins 18 (trigPin) and 19 (echoPin). The code sets up a Wi-Fi connection to IBM Cloud IoT Platform and publishes distance data as JSON payloads. On the right, the simulation window shows the physical components connected. The console output displays the following messages:

```

ALERT!!
Sending payload: {"Distance":72.96,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 72.96
ALERT!!
Sending payload: {"Distance":72.96,"ALERT!!":"Distance less than 100cms"}
Publish ok

```

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

The screenshot shows the IBM Watson IoT Platform dashboard. The 'Recent Events' tab for device ID 12345 is selected, displaying a list of events. The events are as follows:

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