

# Smart Farmer - IoT Enabled Smart Farming Application

## ASSIGNMENT -4

Student Name	R SRIMATHI
Roll No	412519106151

To write code and connections in wokwi for ultrasonic sensor. Whenever distance is less than 100 CMS send "alert" to IBM cloud and display in device recent events.

Code:

```
#include <WiFi.h> // library for WIFI
#include <PubSubClient.h> // library for MQTT
//----- credentials of IBM Accounts -----
#define ORG "04gt4e" // IBM organisation id
#define DEVICE_TYPE "esp32" // Device type mentioned in ibm watson iot platform
#define DEVICE_ID "23456" // Device ID mentioned in ibm watson iot platform
#define TOKEN "zPS*0TV+fi0h)iq(sT" // 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/Data/fmt/json"; // topic name and type of event perform and format in which data to be send
char topic[] = "iot-2/cmd/test/fmt/String"; // cmd Represent type and command is test format of strings
char authMethod[] = "use-token-auth"; // authentication method char
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(115200);
  pinMode(led, OUTPUT);
  pinMode(trigpin, OUTPUT);
  pinMode(echopin, INPUT);
  wifiConnect(); mqttConnect();
}

void loop() { bool isNearby
= dist < 100;
digitalWrite(led, isNearby);
publishData();
delay(500);
if (!client.loop())
{
  mqttConnect(); // function call to connect to ibm
}
}

/* -----retrieving to cloud-----
-----*/

void wifiConnect()
{
  Serial.print("Connecting 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(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)
{
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");  
}  
}  
}
```

Simulation Output:

<https://wokwi.com/projects/347571602979816019>

The screenshot displays the Wokwi web IDE interface. On the left, the 'sketch.ino' file is open, showing a C++ program that initializes an ESP32, connects to a Wi-Fi network, and publishes distance data to an MQTT broker. The code includes comments for credentials and device information. On the right, the 'Simulation' window shows a virtual circuit with an ESP32 microcontroller and an HC-SR04 ultrasonic sensor. A dialog box for the sensor shows a distance of 87cm. Below the simulation, the serial output window shows three consecutive messages: 'Sending payload: {"Alert Distance":86.96}' followed by 'Publish OK'.

The screenshot displays the IBM Watson IoT Platform interface. At the top, the browser address bar shows the URL: <https://04gt4e.internetofthings.ibmcloud.com/dashboard/devices/browse>. The page header includes the text "IBM Watson IoT Platform" and a user profile section with the email "sec19ec035@sairamtap.edu.in" and ID "04gt4e".

The main navigation bar contains tabs: "Browse", "Action", "Device Types", and "Interfaces". An "Add Device" button is located on the right. The left sidebar features icons for various functions: a grid, a gear, a group of people, a person, a globe, a line graph, a clock, and a settings gear.

The central content area shows a device card for "esp32". The card header includes a dropdown arrow, a square icon, the ID "23456", a green "Connected" status, the device name "esp32", the label "Device", and the timestamp "6 Nov 2022 15:56". Below the header are tabs for "Identity", "Device Information", "Recent Events" (which is selected), "State", and "Logs".

Under the "Recent Events" tab, a message states: "The recent events listed show the live stream of data that is coming and going from this device." Below this is a table with the following data:

Event	Value	Format	Last Received
Data	{"Alert Distance":86.96}	json	a few seconds ago
Data	{"Alert Distance":86.96}	json	a few seconds ago
Data	{"Alert Distance":86.96}	json	a few seconds ago
Data	{"Alert Distance":86.96}	json	a few seconds ago
Data	{"Alert Distance":86.96}	json	a few seconds ago

At the bottom of the device card, it indicates "0 Simulations running". The Windows taskbar at the very bottom shows various application icons and the system clock displaying "16:18" on "06-11-2022".