

IOT Based Smart Crop Protection System For Agriculture :

NAME: KANDRA BHANU PRASAD REDDY

ROLL.NO: 71119106301

TEAM.ID: PNT2022TMID44357

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..

Solution:

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

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

#define ECHO_GPIO 12
#define TRIGGER_GPIO 13
#define MAX_DISTANCE_CM 100 // Maximum of 5 meters
#include "Ultrasonic.h"

Ultrasonic ultrasonic(13, 12);
int distance;

void callback(char* subscribetopic, byte* payload, unsigned int
payloadLength);

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

#define ORG "wjgunn" //IBM ORGANITION ID
#define DEVICE_TYPE "ESP32_controller" //Device type mentioned in ibm watson
IOT Platform
#define DEVICE_ID "ESP32" //Device ID mentioned in ibm watson IOT Platform
#define TOKEN "1234567890" //Token
String data3;
float h, t;

//----- Customise the 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 subscribetopic[] = "iot-2/cmd/command/fmt/String"; // cmd REPRESENT
command type AND COMMAND IS TEST OF FORMAT STRING
char authMethod[] = "use-token-auth"; // authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID; //client id
```

```

//-----
WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback, wifiClient); //calling the
predefined client id by passing parameter like server id,portand
wificredential

void setup()// configureing the ESP32
{
    Serial.begin(115200);
    delay(10);
    Serial.println();
    wificonnect();
    mqttconnect();
}

void loop()// Recursive Function
{

    distance = ultrasonic.read(CM);
    if (distance < 100) {
        Serial.print("Distance in CM: ");
        Serial.println(distance);
        PublishData(distance);
        delay(1000);
        if (!client.loop()) {
            mqttconnect();
        }

    }

    delay(1000);

}

/*.....retrieving to
Cloud.....*/

void PublishData(float temp) {
    mqttconnect();//function call for connecting to ibm
    /*
        creating the String in in form JSon to update the data to ibm cloud
    */
    String payload = "{\"Alert Distance:\":";
    payload += temp;
    payload += "}";
}

```

```

Serial.print("Sending payload: ");
Serial.println(payload);

if (client.publish(publishTopic, (char*) payload.c_str())) {
    Serial.println("Publish ok");// if it sucessfully upload data on the cloud
    then it will print publish ok in Serial monitor or else it will print publish
    failed
} 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() //function defination for wificonnect
{
    Serial.println();
    Serial.print("Connecting to ");

    WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish
    the connection
    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++) {
        //Serial.print((char)payload[i]);
        data3 += (char)payload[i];
    }
    Serial.println("data: " + data3);
    if (data3 == "lighton")
    {
        Serial.println(data3);
    }
    else
    {
        Serial.println(data3);
    }
    data3 = "";
}

```

EXECUTION:

The screenshot displays the WOKWI IoT simulator interface. On the left, the 'sketch.ino' file is open in a code editor, showing the following code:

```

1  #include <WiFi.h> //library for wifi
2  #include <PubSubClient.h> //library for MQTT
3
4  #define ECHO_GPIO 12
5  #define TRIGGER_GPIO 13
6  #define MAX_DISTANCE_CM 100 // Maximum of 5 meters
7  #include "Ultrasonic.h"
8
9  Ultrasonic ultrasonic(13, 12);
10 int distance;
11
12 void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)
13
14 //-----credentials of IBM Accounts-----
15
16 #define ORG "wjgunn" //IBM ORGANIZATION ID
17 #define DEVICE_TYPE "ESP32_controller" //Device type mentioned in ibm watson IoT
18 #define DEVICE_ID "ESP32" //Device ID mentioned in ibm watson IOT Platform
19 #define TOKEN "1234567890" //Token
20 String data3;
21 float h, t;
22
23 //----- Customise the above values -----
24 char server[] = ORG ".messaging.internetofthings.ibmcloud.com"; // Server Name
25

```

On the right, the 'Simulation' tab shows a visual representation of the hardware. It includes an HC-SR04 ultrasonic sensor module connected to an ESP32 microcontroller board via four colored jumper wires (red, yellow, green, and blue). The sensor's VCC pin is connected to the ESP32's 5V pin, GND to GND, TRIG to GPIO 13, and ECHO to GPIO 12.

Below the simulation, the console output shows the following sequence of events:

```

Publish ok
Distance in CM: 57
Sending payload: {"Alert Distance":57.00}
Publish ok
Distance in CM: 57
Sending payload: {"Alert Distance":57.00}
Publish ok

```

IBM CLOUD OUTPUT :

Browse | Action | Device Types | Interfaces
Add Device +

▼ ● Connected ESP32_Controller Device Oct 30, 2022 4:42 PM → ...

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":1.99,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":125.95,"Status":"Normal"}	json	a few seconds ago
data	{"Distance":125.95,"Status":"Normal"}	json	a few seconds ago
data	{"Distance":34.97,"Status":"Alert"}	json	a few seconds ago
data	{"Distance":34.97,"Status":"Alert"}	json	a few seconds ago

0 Simulations running