

## Assignment -4

### Python Programming

Student Name	Varunkumar N
Student Roll Number	720319106026

#### Question-1:

#### Solution:

Link : <https://wokwi.com/projects/348131657358770771>

```
#include <WiFi.h>
#include <PubSubClient.h>
WiFiClient wifiClient;

#define trigpin    18
#define echopin    5

String data3;

#define ORG "kt3pfh"//IBM ORGANITION ID
#define DEVICE_TYPE "Water"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "24"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "79247924"

#define speed 0.034
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char topic[] = "iot-2/cmd/home/fmt/String";
char authMethod[] = "use-token-auth";
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;
PubSubClient client(server, 1883, wifiClient);
void publishData();

String command;
String data="";
long duration;
float dist;
```

```

void setup()
{
    Serial.begin(115200);
    wifiConnect();
    pinMode(trigpin, OUTPUT);
    pinMode(echopin, INPUT);

    mqttConnect();
}

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 loop()
{

    int pulseWidth = 0;
    digitalWrite(trigpin, HIGH);
    delayMicroseconds(100);
    digitalWrite(trigpin, LOW);
    pulseWidth = pulseIn(echopin, HIGH);
    Serial.print("AlertDistance: ");
    Serial.println(pulseWidth/58);

    publishData();
    if (!client.loop()) {
        mqttConnect();
    }
}

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(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){
        String payload = "{\"Normal 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");
        }
    }

    if(dist>101 && dist<111){
        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())) {
            Serial.println("Warning crosses 110cm -- it automaticaly of the loop");
        }
    }
}

```

```

    }else {
        Serial.println("Publish 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++){
        dist += (char)payload[i];
    }
    Serial.println("data:" + data3);
    if(data3=="lighton"){
        Serial.println(data3);
    }
    data3="";
}
}

```

The screenshot displays the IBM Watson IoT Platform dashboard. The top navigation bar includes tabs for 'Browse', 'Action', 'Device Types', and 'Interfaces'. The main content area is titled 'Recent Events' and shows a table of data events. The table has four columns: 'Event', 'Value', 'Format', and 'Last Received'. The data shows a series of 'Normal Distance' readings in JSON format, received 'a few seconds ago'.

Event	Value	Format	Last Received
Data	{"Normal Distance":68.95}	json	a few seconds ago
Data	{"Normal Distance":69.02}	json	a few seconds ago
Data	{"Normal Distance":69}	json	a few seconds ago
Data	{"Normal Distance":68.95}	json	a few seconds ago
Data	{"Normal Distance":68.99}	json	a few seconds ago

At the bottom of the dashboard, it indicates '0 Simulations running' and 'Items per page 50 | 1-1 of 1 item'.

Output : <https://kt3pfh.internetofthings.ibmcloud.com/dashboard/devices/browse>

The screenshot displays the Wokwi IDE interface. On the left, the code editor shows a C++ program for an ESP32. The code calculates distance using the formula  $dist = duration * speed / 2$  and implements two distance thresholds: a 'Normal' distance (100cm) and an 'Alert' distance (110cm). The code uses the Arduino `Serial` library for debugging and the `client.publish` function to send data to IBM Watson IoT. The right side of the IDE features a simulation window with a visual representation of the ESP32 and the HC-SR04 sensor connected by jumper wires. Below the simulation, a serial console window shows the output of the program, including the calculated distance and the status of the publish operation.

```
94 dist=duration*speed/2;
95 if(dist<100){
96   String payload = "{\"Normal Distance\": ";
97   payload += dist;
98   payload += "}";
99
100   Serial.print("\n");
101   Serial.print("Sending payload: ");
102   Serial.println(payload);
103   if (client.publish(publishTopic, (char*) payload.c_str())) {
104     Serial.println("Publish OK");
105   }
106 }
107
108 if(dist>101 && dist<111){
109   String payload = "{\"Alert distance\": ";
110   payload += dist;
111   payload += "}";
112
113   Serial.print("\n");
114   Serial.print("Sending payload: ");
115   Serial.println(payload);
116   if(client.publish(publishTopic, (char*) payload.c_str())) {
117     Serial.println("Warning crosses 110cm -- it automatically of the loop");
118   }
119 }else {
120   Serial.println("Publish FAILED");
121 }
122
123 }
```

Simulation

00:10.231 11%

Sending payload: {"Normal Distance":68.95}  
Publish OK  
AlertDistance: 69

Sending payload: {"Normal Distance":69.00}  
Publish OK