

Assignment -4

SmartFarmer - IoT Enabled Smart Farming Application

| | |
|---------------------|--------------------|
| Assignment Date | 17 October 2022 |
| Student Name | S.L.Senthil Sachin |
| Student Roll Number | 910619106307 |
| Maximum Marks | 2 Marks |

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* subscribtopic, byte* payload, unsigned int payloadLength);
#define ORG "ovk5a1"
#define DEVICE_TYPE "ESP32"
#define DEVICE_ID "06307"
#define TOKEN "1234567890"
String data3;
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";
char publishTopic[] = "iot-2/evt/Data/fmt/json";
char subscribtopic[] = "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 :

Output and Simulation :

The screenshot shows the Wokwi IDE interface. On the left, the 'sketch.ino' file is open, displaying C++ code for an ESP32 microcontroller. The code sets up an MQTT client and uses an ultrasonic sensor to measure distance. It publishes distance data and an alert if the distance is less than 100cm. On the right, the 'Simulation' window shows a visual representation of the ESP32 and the HC-SR04 ultrasonic sensor connected. Below the simulation, the output console shows the following log:

```
Publish ok
Distance (cm): 12.99
ALERT!!
Sending payload: {"Distance":12.99,"ALERT!!":"Distance less than 100cms"}
Publish ok
Distance (cm): 12.99
ALERT!!
```

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 'Browse' tab is selected, and the 'Recent Events' tab is active for the device '06307'. The dashboard displays a table of recent events:

| Event | Value | Format | Last Received |
|-------|---|--------|-------------------|
| Data | {"Distance":12.99,"ALERT!!":"Distance less than ... | json | a few seconds ago |
| Data | {"Distance":65.94,"ALERT!!":"Distance less than ... | json | a few seconds ago |

The dashboard also shows the device's status as 'Connected' and provides navigation options for device management.