온습도 경고앱

온습도 센서

환경 모니터링

개요

라즈베리 파이에서 온도 데이터 수집 코드를 윈도우로로 배포하여 모니터링 하는 실습 제고

필요한 하드웨어

1. **PC** - 데이터 수신용
2. Azure IoT Edge용 Grove 스타터 키트 or **Arduino 센서 키트**
3. 라즈베리파이4 - 데이터 송신용
4. 마이크로 USB 인터페이스가 있는 5V 전원 공급 장치
5. 모니터 + HDMI 케이블
6. USB 키보드, 마우스
7. 기타 등등

부품 구성

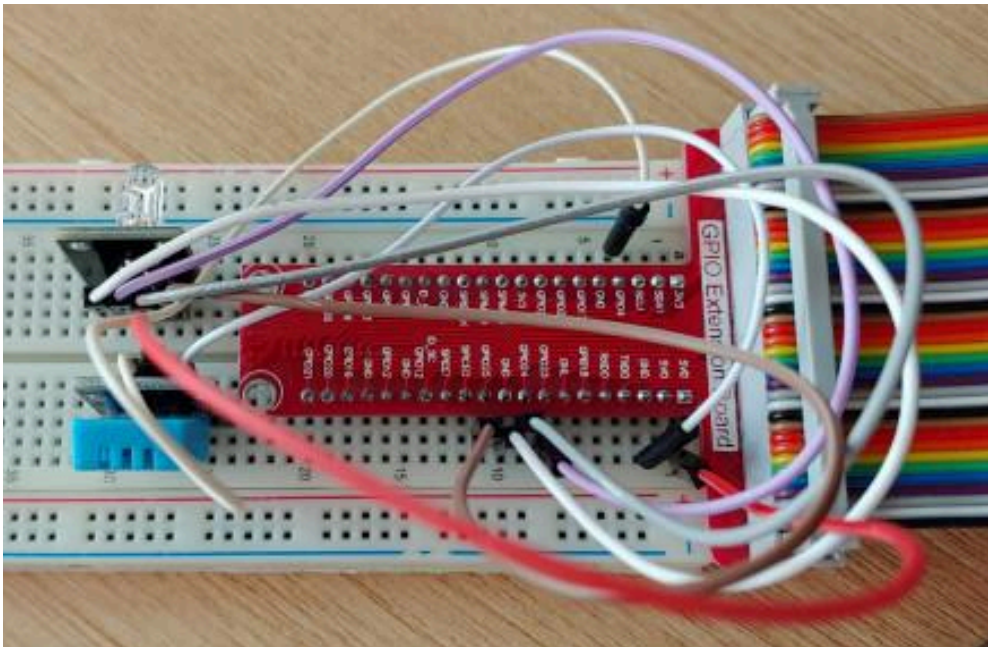
기본 부품

GPIO Extension Board 및 Bread Board 위에 온습도 센서 DHT11 및 RGB LED 구성
DHT11 (3개 단자)

- - : GND
- +(중앙) : 5V
- S : GPIO4

RGB LED 모듈 (4개 단자) - 단자 순서가 실제 RGB와 다를 수 있음

- - : GND
- R : GPIO23
- G : GPIO24
- B : GPIO25



개발 시작

아래의 내용은 여러 포스트에 나와 있으므로 생략가능합니다. 또한, 설치 방법도 변경되어 아래의 내용이 다를 수 있습니다. 주의하세요.

Visual Studio Code 설치

라즈비안 터미널에서 설치합니다.

콘솔 - GPG Key 설치

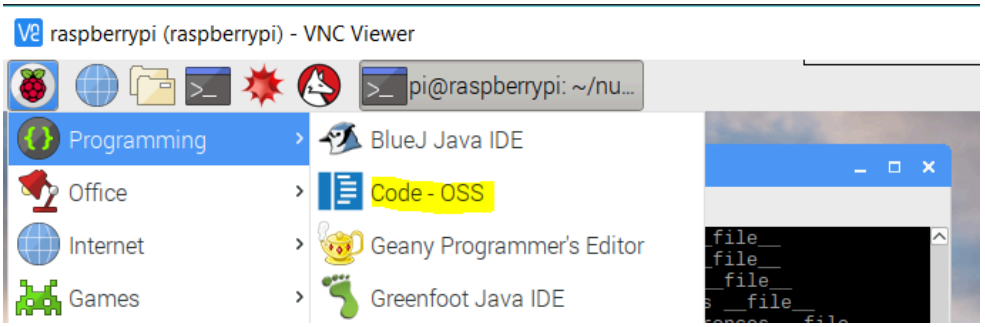
```
$ wget https://packagecloud.io/headmelted/codebuilds/gpgkey -O - | sudo apt-key add -
```

Visual Studio Code 설치

```
$ curl -L
https://raw.githubusercontent.com/headmelted/codebuilds/master/docs/installers/apt.sh |
sudo bash
```

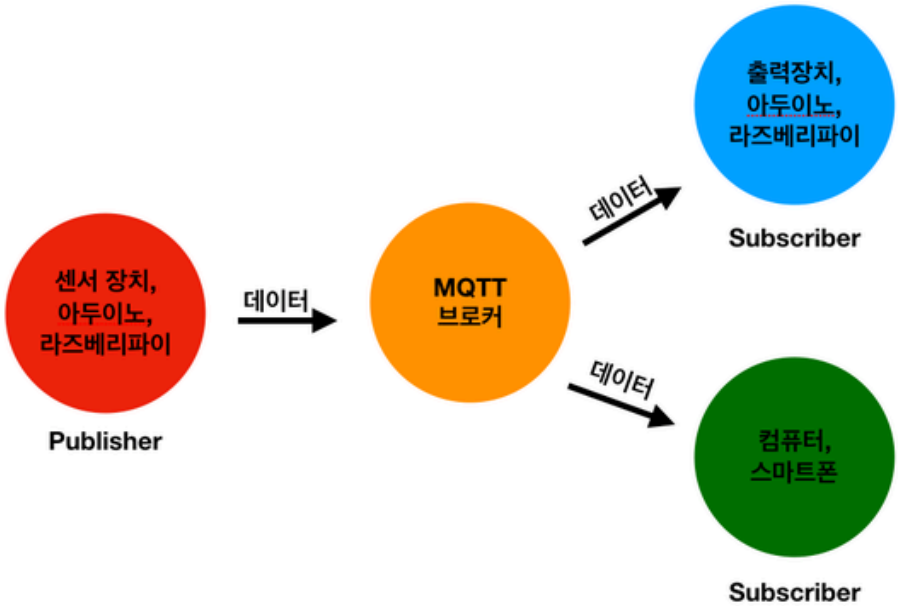
Code-OSS 실행

```
$ code-oss
```



데이터 전달 프로세스

라즈베리파이 디바이스 직접 관리 + MQTT

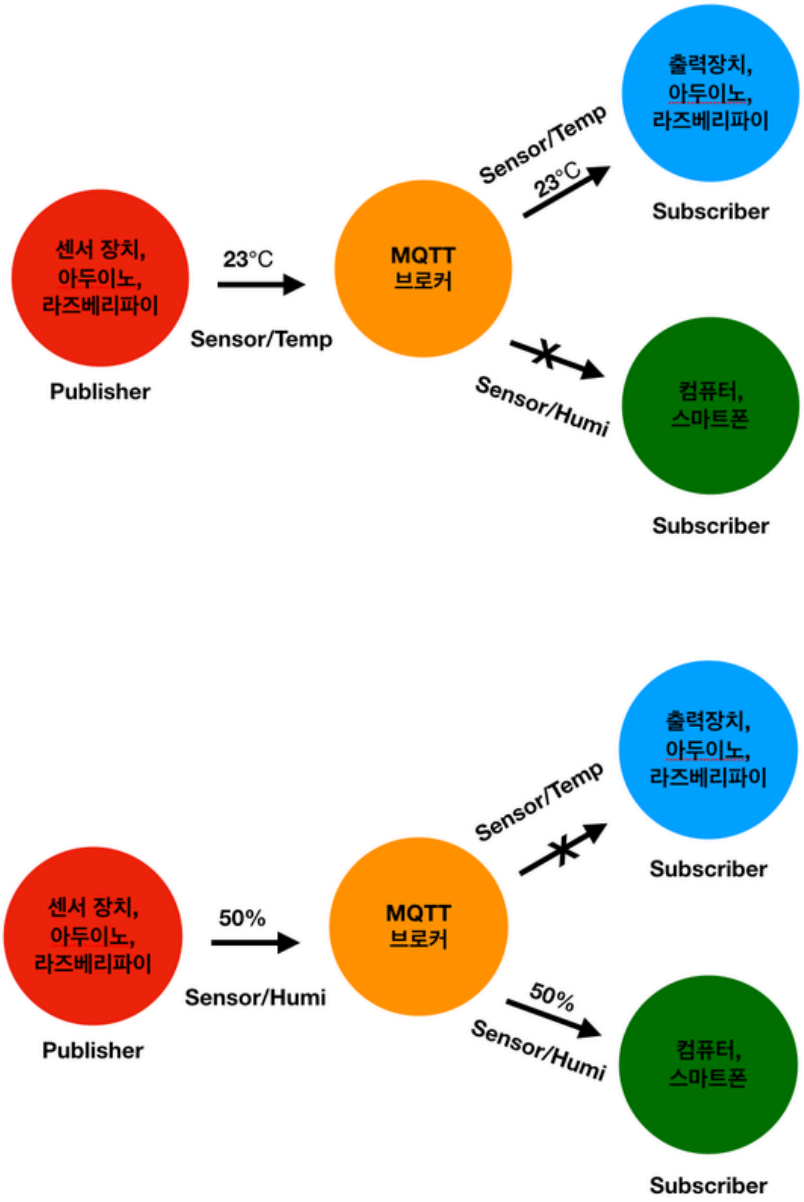


MQTT

- 사물 인터넷에서 사용하기 위해 개발된 TCP 기반 프로토콜 : Message Queuing Telemetry Transport, ISO/IEC PRF 20922 표준
- 낮은 전력, 낮은 대역폭, 저성능 환경에서도 가용성 높음
- 통신 publisher / subscriber 로 진행되며 중간에 Broker 존재

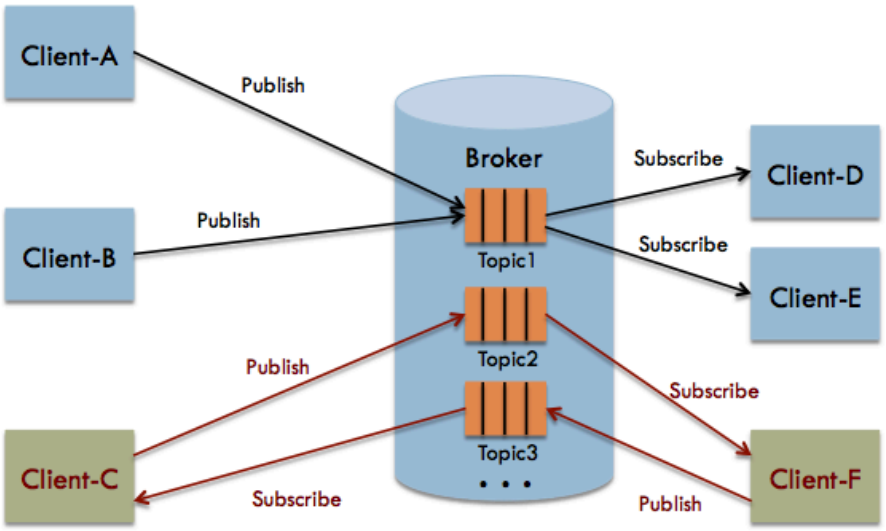
Publisher(출판자) / Subscriber(구독자) / Topic(주제) 라는 일반적인 개념의 이름을 용어로 사용하고 있습니다.

Publisher와 Subscriber 데이터 전달구조



기본 동작 순서

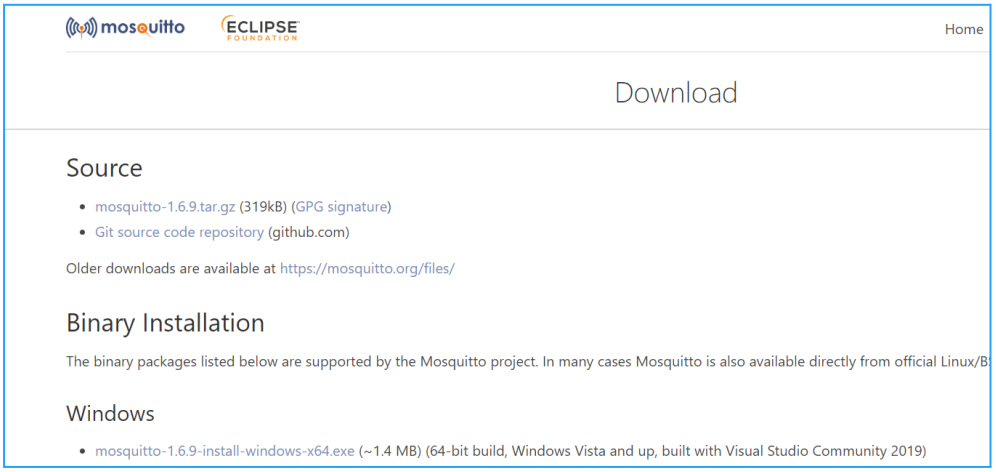
동작하는 Broker가 필요, Publisher와 Subscriber는 IP와 Port 정보를 알고 있어야 합니다. Subscriber가 Broker와 연결, 원하는 Topic(토픽)을 subscribe(구독)하는 겁니다. 예를 들면, 습도에 관한 토픽(Sensor/Humi)이 오면 A에게 주고, 다른 토픽(Sensor/Temp)이 오면 B에게 전달해달라는 의미입니다. Publisher는 Broker와 연결, 특정 토픽으로 message를 publish합니다.



Window MQTT Broker 설치

일반적으로 mosquitto라는 MQTT 브로커가 가장 유명합니다. Opensource (EPL) message broker solution 입니다.

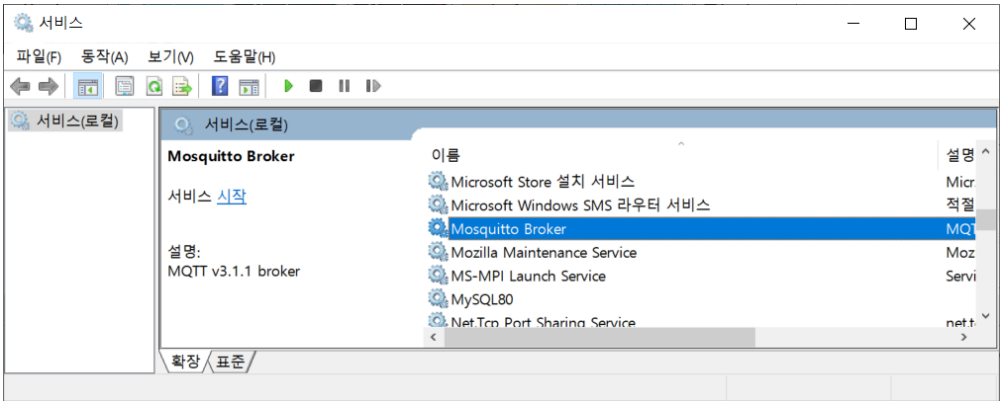
https://mosquitto.org/



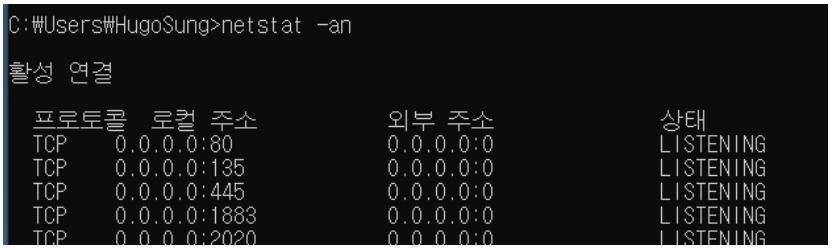
Dependency 확인 (실제로는 필요없음)

http://slproweb.com/products/Win32OpenSSL.html

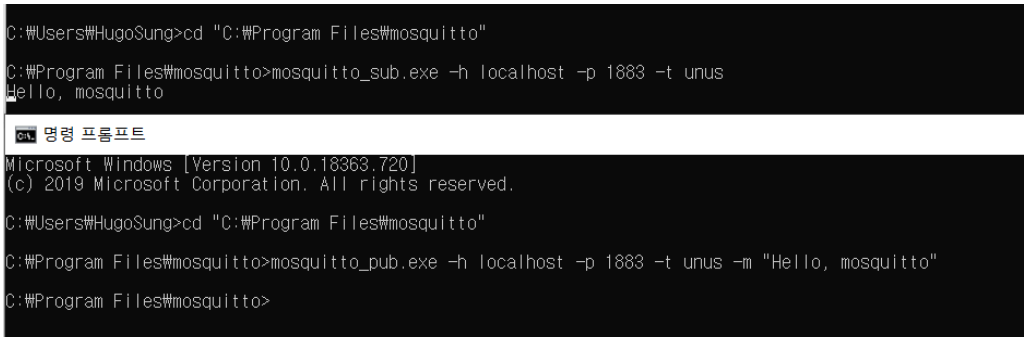
설치는 매우 쉽습니다. 모두 설치한 뒤에 윈도우 서비스를 시작합니다.



Mosquitto Broker 를 시작하는 겁니다. 디폴트 포트는 1883로 윈도우 방화벽 인바운드(Inbound) 설정을 추가해 주어야 합니다.



이후 MQTT 메시지 모니터링을 위하여 크롬 확장프로그램인 MQTTLens나 StandAlone 프로그램인 Mqtt explorer 중에 자신에게 맞게 설치합니다.
https://mqtt-explorer.com/ 를 추천합니다.



외부접속 허용

- Mosquitto 서비스 중지
- mosquitto.conf 파일에서 수정
    - listener 1883 추가
    - allow_anonymous true 추가
- 방화벽 및 네트워크 보호 > 고급 설정 > 인바운드 규칙 > 새 규칙 > 포트 > 1883 추가

- Mosquitto 서비스 재시작

Broker설치 컴퓨터 아이피를 확인합니다.
연결 확인 후, Subcribe와 Publish 토픽 설정합니다.

- Home/sensor1/data/

라즈베리파이 작업

온습도센서(DHT11) 테스트

Digital Humidity Temperature의 약어인것 같습니다.
앞서 라즈베리파이에 연결한 상태에서 파이썬 코드를 작성합니다.

Adafruit-DHT 센서는 2024-05 기준으로 변경되었습니다. 이전의 라이브러리 설치가 오류가 납니다. Raspbian Bulleye 까지 동작하던 기능은 bookworm에서 아예 동작 불가합니다.

처리 순서

apt-get 설치 불가 해제

```
> sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
```

이전 Adafruit 패키지 삭제

```
> sudo pip uninstall Adafruit-DHT
```

사용가능 라이브러리 설치

```
> sudo pip install adafruit-circuitpython-dht
```

변경 소스

```
# dht11_test.py
import adafruit_dht
import time
import RPi.GPIO as GPIO
import board
```

```
log_num = 0
sensor_pin = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN)
dhtDevice = adafruit_dht.DHT11(board.D18) # Problem!!!

while (True):
    try:
        temp = dhtDevice.temperature
        humid = dhtDevice.humidity
        print(f'{log_num} - Temp : {temp}C / Humid : {humid}%')
        log_num += 1
        time.sleep(2)
    except RuntimeError as ex:
        print(ex.args[0])
    except KeyboardInterrupt:
        break

dhtDevice.exit()
```

~~dht11_test_app.py 소스~~

```
# sudo pip3 install Adafruit_DHT
import Adafruit_DHT as dht
import time

sensor = dht.DHT11
pin = 4

try:
    while True:
        h, t = dht.read_retry(sensor, pin)
        if h is not None and t is not None:
            print("Temp = {0:0.1f}C Humidity = {1:0.1f}%".format(t, h))
        else:
            print("Read error")

        time.sleep(1)
    print("Terminated by Keyboard")

finally:
    print("End of program")
```

콘솔 결과는 아래와 같습니다. 아래의 메시지가 계속 출력됩니다.

```
Temp = 25.0C Humidity = 52.0%
Temp = 25.0C Humidity = 52.0%
Temp = 25.0C Humidity = 52.0%
Temp = 24.0C Humidity = 51.0%
```

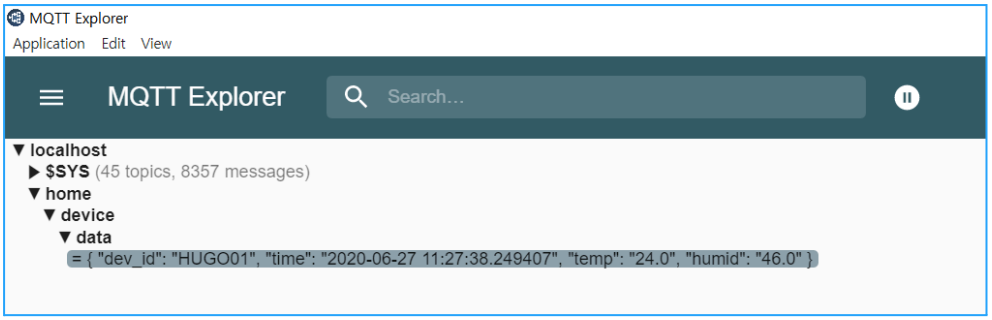
Publisher 구현

라즈베리파이에 Publisher를 구현합니다. MQTT 라이브러리로 paho-mqtt를 설치합니다.

```
$ sudo pip3 install paho-mqtt
```

mqtt_temp_monitoring.py 소스

```
import paho.mqtt.client as mqtt
import Adafruit_DHT as dht
import json
import time
import datetime as dt
import uuid
from collections import OrderedDict

sensor = dht.DHT11; pin = 4; count = 0

try:
    dev_id = "HUGO01"
    #mqtt publisher
    broker_address = "192.168.200.102"
    client2 = mqtt.Client("TempClient")
    client2.connect(broker_address)
    #dh11 init
    while True:
        count += 1
        currtime = dt.datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')
        h, t = dht.read_retry(sensor, pin)
        # groupdata 만들기
        raw_data = OrderedDict()
        raw_data["dev_id"] = dev_id
        raw_data["time"] = currtime
        raw_data["temp"] = "{0:0.1f}".format(t)
        raw_data["humid"] = "{0:0.1f}".format(h)

        pub_data = json.dumps(raw_data, ensure_ascii=False, indent="\t")
        #mqtt published
        print(dev_id, pub_data)
        client2.publish("home/device/data/", pub_data)
        time.sleep(5)

except Exception as ex:
    print('Error raised ', ex)
```

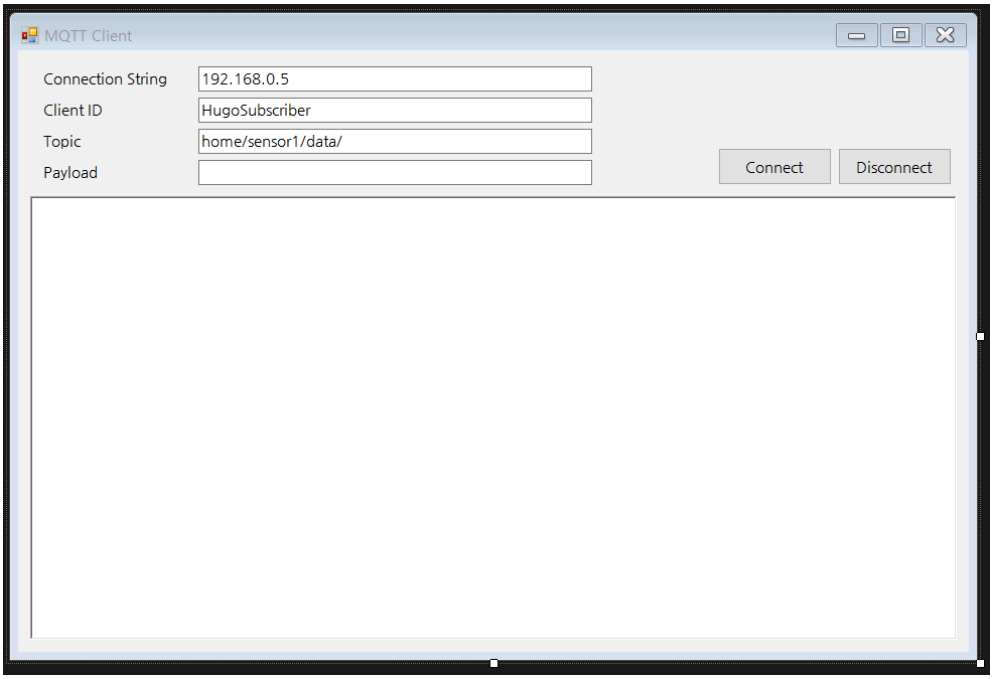


Windows에 Subscriber 구현

Winforms 프로젝트

Winforms 프로젝트 생성합니다. NuGet 패키지에서 M2Mqtt를 설치한 뒤, 아래와 같이 MainForm 화면을 구성합니다.

- TxtConnectionString
- TxtClientId
- TxtTopic
- TxtPayload
- BtnConnect
- BtnDisconnect
- RtbReceived

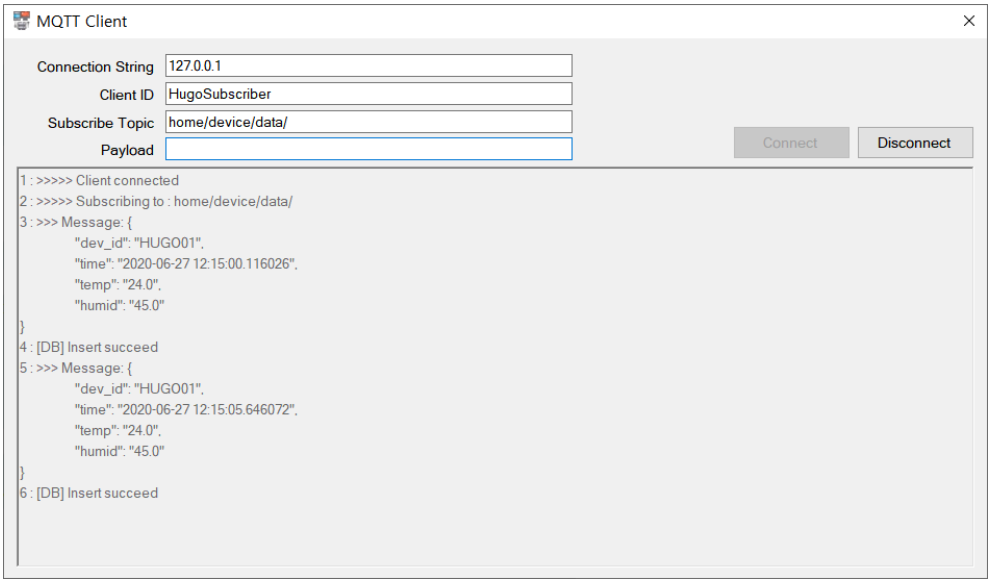


MainForm.cs 소스

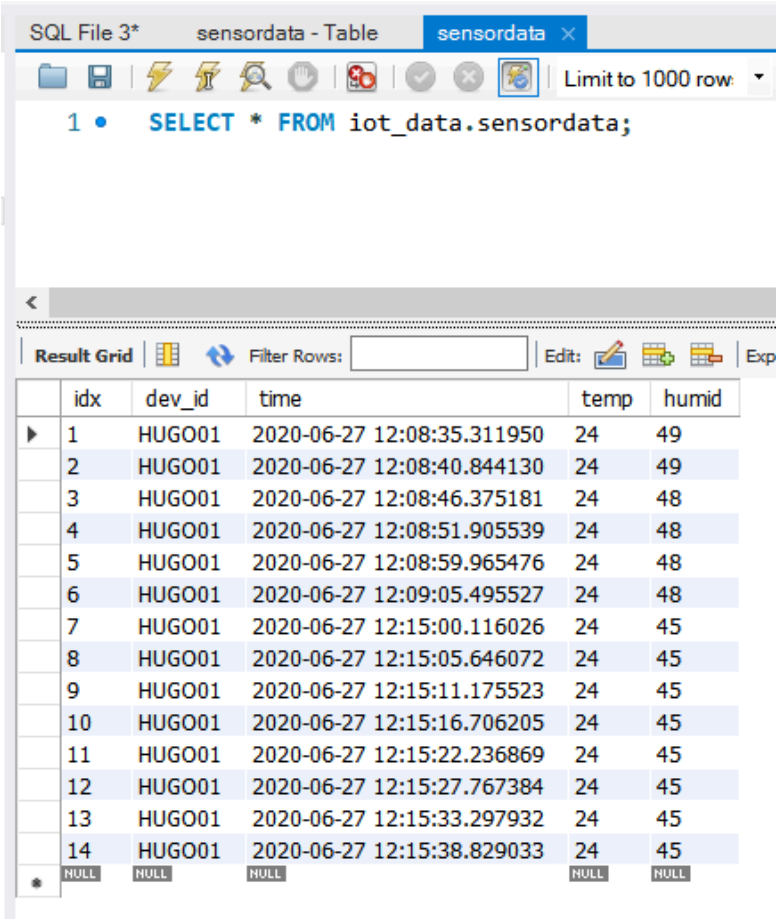
```
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

namespace MqttSubscriber {
    public partial class FrmMain : Form {
        MqttClient client;
        private string connectionString;
        private ulong line_count;
        delegate void UpdateTextCallback(string message);

        public FrmMain() {
            InitializeComponent();
            InitAllData();
        }

        private void InitAllData() {
            connectionString = "Server=localhost;Port=3306;"+
                               "Database=iot_data;Uid=root;Pwd=maria_p@ssw0rd!";
            line_count = 0;
            BtnConnect.Enabled = true;
            BtnDisconnect.Enabled = false;
        }

        private void FrmMain_Load(object sender, EventArgs e) {
            try {
                IPAddress hostIP;
                hostIP = IPAddress.Parse(TxtConnectionString.Text);
                client = new MqttClient(hostIP);
                client.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;
            } catch (Exception ex) {
                MessageBox.Show(ex.ToString());
            }
        }

        private void Client_MqttMsgPublishReceived(object sender,
MqttMsgPublishEventArgs e) {
            try {
                var message = Encoding.UTF8.GetString(e.Message);
                UpdateText(">>> Message: " + message);
                InsertData(message); // 메시지가 발생할 경우 DB에 저장
            } catch (Exception ex) {
                UpdateText("[ERROR] " + ex.Message);
            }
        }

        private void InsertData(string message) {
            var currentDatas = JsonConvert.DeserializeObject<Dictionary<string,
string>>(message);
            using (var conn = new MySqlConnection(connectionString)) {
                string strInsertQry = string.Format("INSERT INTO iot_data.sensordata " +
                                      " (idx, dev_id, time, temp, humid) " +
```

```
                                        "VALUES (NULL, '{0}', '{1}', {2}, {3}); ",
                                        currentDatas["dev_id"], currentDatas["time"],
                                        currentDatas["temp"], currentDatas["humid"]);
                try {
                    conn.Open();
                    MySqlCommand cmd = new MySqlCommand(strInsertQry, conn);
                    if (cmd.ExecuteNonQuery() == 1)
                        UpdateText("[DB] " + "Insert succeed");
                    else
                        UpdateText("[DB] " + "Insert failed");
                } catch (Exception ex) {
                    UpdateText("[DB] " + ex.Message);
                }
            }
        }

        private void UpdateText(string message) {
            if (RtbRecieved.InvokeRequired) {
                UpdateTextCallback d = new UpdateTextCallback(UpdateText);
                this.Invoke(d, new object[] { message });
            } else {
                line_count++;
                this.RtbRecieved.AppendText(line_count.ToString() + " : " + message +
"\n");
                this.RtbRecieved.ScrollToCaret();
            }
        }

        private void BtnConnect_Click(object sender, EventArgs e) {
            client.Connect(TxtClientId.Text + "_sub");
            UpdateText(">>>>> Client connected");
            client.Subscribe(new string[] { TxtTopic.Text },
                new byte[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
            UpdateText(">>>>> Subscribing to : " + TxtTopic.Text);
            BtnConnect.Enabled = false; BtnDisconnect.Enabled = true;
        }

        private void BtnDisconnect_Click(object sender, EventArgs e) {
            client.Disconnect(); UpdateText(">> Client disconnected");
            BtnConnect.Enabled = true; BtnDisconnect.Enabled = false;
        }
    }
}
```

실행화면



DB 확인화면



idx	dev_id	time	temp	humid
1	HUGO01	2020-06-27 12:08:35.311950	24	49
2	HUGO01	2020-06-27 12:08:40.844130	24	49
3	HUGO01	2020-06-27 12:08:46.375181	24	48
4	HUGO01	2020-06-27 12:08:51.905539	24	48
5	HUGO01	2020-06-27 12:08:59.965476	24	48
6	HUGO01	2020-06-27 12:09:05.495527	24	48
7	HUGO01	2020-06-27 12:15:00.116026	24	45
8	HUGO01	2020-06-27 12:15:05.646072	24	45
9	HUGO01	2020-06-27 12:15:11.175523	24	45
10	HUGO01	2020-06-27 12:15:16.706205	24	45
11	HUGO01	2020-06-27 12:15:22.236869	24	45
12	HUGO01	2020-06-27 12:15:27.767384	24	45
13	HUGO01	2020-06-27 12:15:33.297932	24	45
14	HUGO01	2020-06-27 12:15:38.829033	24	45
NULL	NULL	NULL	NULL	NULL

온습도 모니터링

개요

상태를 아래와 같이 나눕니다.

- 온도가 기준 온도 이상 올라갔을 경우 - RGB LED 빨간색 점멸 반복
- 온도가 기준 온도 이하로 내려갔을 경우 - RGB LED 녹색 항상 켜진 상태

Subscriber 수정

디자인 화면에 Publish Topic 추가 및 컨트롤 알림 및 정상 온도 작성 텍스트 박스를 추가합니다.

소스 추가 코딩

클래스 멤버 변수 추가

```
private float open_temp, close_temp;
private bool opened;
```

InitAllData 메서드 변경

```
private void InitAllData() {
    connectionString = "Server=localhost;Port=3306;" +
                       "Database=iot_data;Uid=root;Pwd=maria_p@ssw0rd!";
    line_count = 0;
    BtnConnect.Enabled = true;
    BtnDisconnect.Enabled = false;

    var temps = TxtControlTemp.Text.Split(',');
    opened = false; // 모터 컨트롤로 오픈했는지?
    open_temp = float.Parse(temps[0].Trim());
    close_temp = float.Parse(temps[1].Trim());
    UpdateText($"Initialized - Open Temperature {open_temp}°C / Close Temperature
{close_temp}°C");
}
```

Client_MqttMsgPublishReceived 메서드 내 신규 메서드 추가

```
UpdateText(">>> Message: " + message);
InsertData(message);        // 메시지가 발생할 경우 DB에 저장
SendToBroker(message);      // 알람이 발생시 디바이스로 재전송
```

SendToBroker(…) 메서드 추가

```
private void SendToBroker(string message) {
    var currentDatas = JsonConvert.DeserializeObject<Dictionary<string,
```

```
string>>(message);
    var dev_id = currentDatas["dev_id"];
    var currTemp = float.Parse(currentDatas["temp"]);
    Debug.WriteLine(currTemp);

    JObject json = new JObject();
    if (currTemp >= open_temp)     {
        if (opened == false)         {
            json.Add("dev_id", dev_id);
            json.Add("state", "ON");
            string strJson = JsonConvert.SerializeObject(json);
            client.Publish(TxtPayload.Text, Encoding.Default.GetBytes(strJson),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
            Debug.WriteLine(json);
            opened = true;
            UpdateText($"{dev_id} state Alarm");
        }
    } else if (currTemp <= close_temp) {
        if (opened) {
            json.Add("dev_id", dev_id);
            json.Add("state", "OF");
            string strJson = JsonConvert.SerializeObject(json);
            client.Publish(TxtPayload.Text, Encoding.Default.GetBytes(strJson),
MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
            Debug.WriteLine(json);
            opened = false;
            UpdateText($"{dev_id} state Normal");
        }
    }
}
```

MQTT 컨트롤러

소스코딩

Mqtt_temp_controller.py 소스

```
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
import json
import time

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("connected OK")
    else:
        print("Bad connection Returned code=", rc)
def on_disconnect(client, userdata, flags, rc=0):
    print(str(rc))
```

```
def on_subscribe(client, userdata, mid, granted_qos):
    print("subscribed: " + str(mid) + " " + str(granted_qos))

def on_message(client, userdata, msg):
    json_data = json.loads(str(msg.payload.decode("utf-8")))
    dev_id = json_data["dev_id"]
    state = json_data["state"]
    print(dev_id)
    print(state)
    if dev_id == 'HUGO01':
        process_alarm(state)

def process_alarm(state):
    if state == 'ON':
        GPIO.output(RED, 255)
        GPIO.output(GREEN, 0)
        GPIO.output(BLUE, 0)
    elif state == 'OFF':
        GPIO.output(RED, 0)
        GPIO.output(GREEN, 255)
        GPIO.output(BLUE, 0)

# RGB LED 모듈 초기화
GPIO.setmode(GPIO.BCM)
RED = 25; GREEN = 24; BLUE = 23
GPIO.setup(RED,GPIO.OUT)
GPIO.setup(GREEN,GPIO.OUT)
GPIO.setup(BLUE,GPIO.OUT)

GPIO.output(RED,0)
GPIO.output(GREEN,255)
GPIO.output(BLUE,0)

try:
    # 새로운 클라이언트 생성
    client = mqtt.Client()
    # 콜백 함수 설정 on_connect(브로커에 접속), on_disconnect(브로커에 접속중료),
on_subscribe(topic 구독),
    # on_message(발행된 메세지가 들어왔을 때)
    client.on_connect = on_connect
    client.on_disconnect = on_disconnect
    client.on_subscribe = on_subscribe
    client.on_message = on_message
    # address : localhost, port: 1883 에 연결
    client.connect('192.168.200.102', 1883)
    # common topic 으로 메세지 발행
    client.subscribe('home/device/control/', 1)
    client.loop_forever()
finally:
    GPIO.cleanup()
```

Mqtt_temp_monitoring.py 콘솔에서 실행하고 Mqtt_temp_controller.py를 따로 실행합니다.

동작 확인

1. 온습도 센서 앞에 뜨거운 물 컵과 차가운 물 컵으로 센서 변경
2. 뜨거운 물 온도가 오르면 RGB LED 빨간 색 점등
3. 차가운 물로 온도가 내려가면 RGB LED 녹색 점등