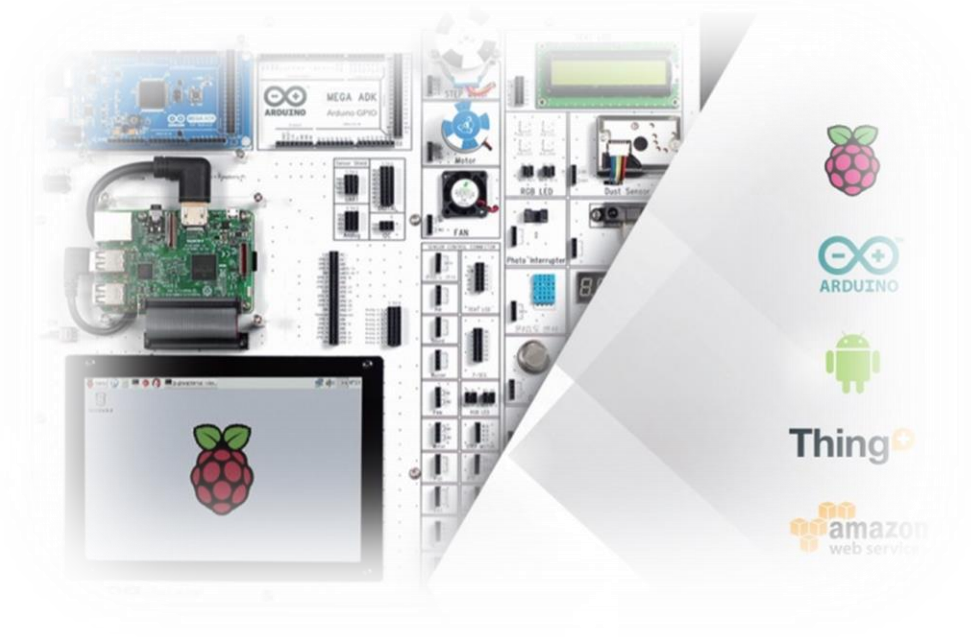
# 2021년도 2학기
# 휴먼스마트기기설계




교과목명 : 휴먼스마트기기설계 (01)

담당교수 : 이 수 형

E-mail : soohyong@uu.ac.kr

교재명 : 스마트기기 개발을 위한 사물인터넷, 이수형. (LINC+ 사업단 배포)

# 3부 : 네트워크

## 7. 원격 센서 측정

# 원격 센서 측정

- 사물인터넷 기기의 용도
  - 센서 측정된 값 ⇒ 인터넷 ⇒ 서버 ⇒ 원격에서 모니터링 (예: 기상 관측소)
  - 원격 제어 S/W ⇒ 서버 ⇒ 사물인터넷 기기 : 제어

- 원격 센서 측정 방법
  - 센서 → NodeMCU (서버의 역할) ⇒ 원격 모니터링
  - 센서 → NodeMCU (서버로 전송하는 역할 ) ⇒ 데이터 서버 ⇒ 원격 모니터링

# 7.1 원격 온/습도 측정장치 #1

- 온도/습도 센서
  - DHT11 : 아두이노에서 사용가능한 대표적인 온도/습도 센서
    - ➢ 온도사양 (Temperature Specification)
      - ✓ 분해능(Resolution) : 1 ℃
      - ✓ 정확도(Accuracy) : ± 2 ℃
      - ✓ 측정범위(Measuring Range) : 0~50 ℃
    - ➢ 습도사양 (Humidity Specification)
      - ✓ 분해능(Resolution) : 1% RH
      - ✓ 정확도(Accuracy) : ± 5% RH (0~50°C)
      - ✓ 측정범위(Measuring Range) : 20~90% RH (25°C)
    - ➢ 사양 (Specification) :
      - ✓ 동작전압(Operating Voltage) : 3.3 ~ 5V
      - ✓ 권장 보관 조건(Recommended storage conditions) : 온도 10~40 ℃ / 습도 60% RH 이하

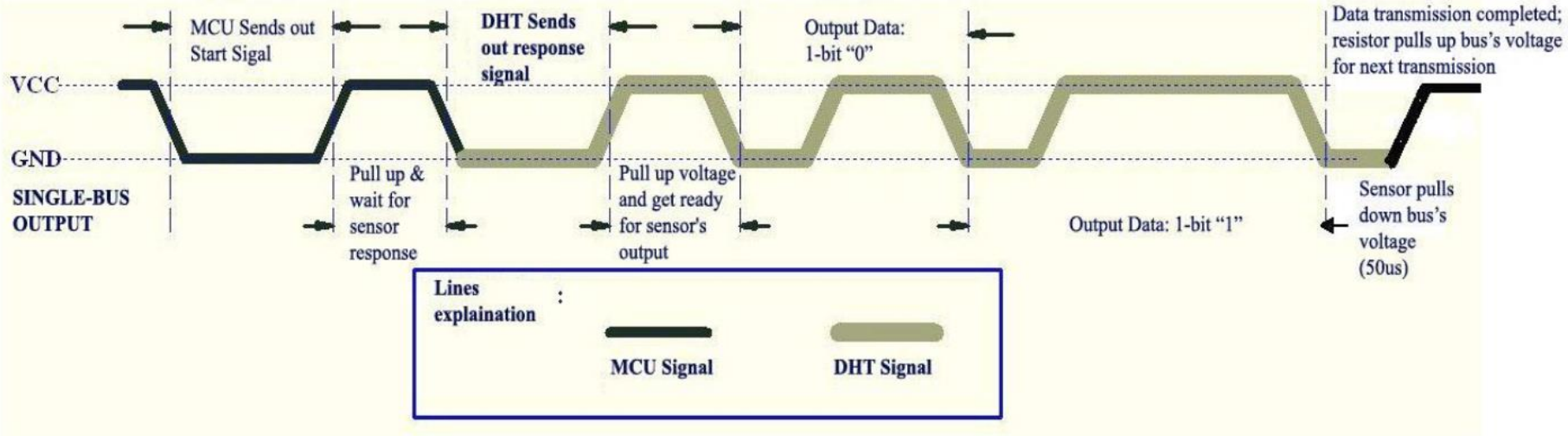
# 7.1 원격 온/습도 측정장치 #2

- AM2302센서 (DHT22)
  - DHT11에 비해서 고성능의 센서
    - ➢ 온도사양 (Temperature Specification) :
      - ✓ 분해능(Resolution) : 0.1 ℃
      - ✓ 정확도(Accuracy) : ± 0.5 ℃
        - 측정범위(Measuring Range) : -40~80 ℃
    - ➢ 습도사양 (Humidity Specification) :
      - 분해능(Resolution) : 0.1% RH
      - 정확도(Accuracy) : ± 2% RH
    - ➢ 단점 : 측정시간이 더 필요함


- 연결
  - DHT11/DHT22 : 4개의 핀이 있으며 추가 회로구성이 필요함
  - 센서 모듈을 사용함으로 포트에 직접 연결이 가능해짐

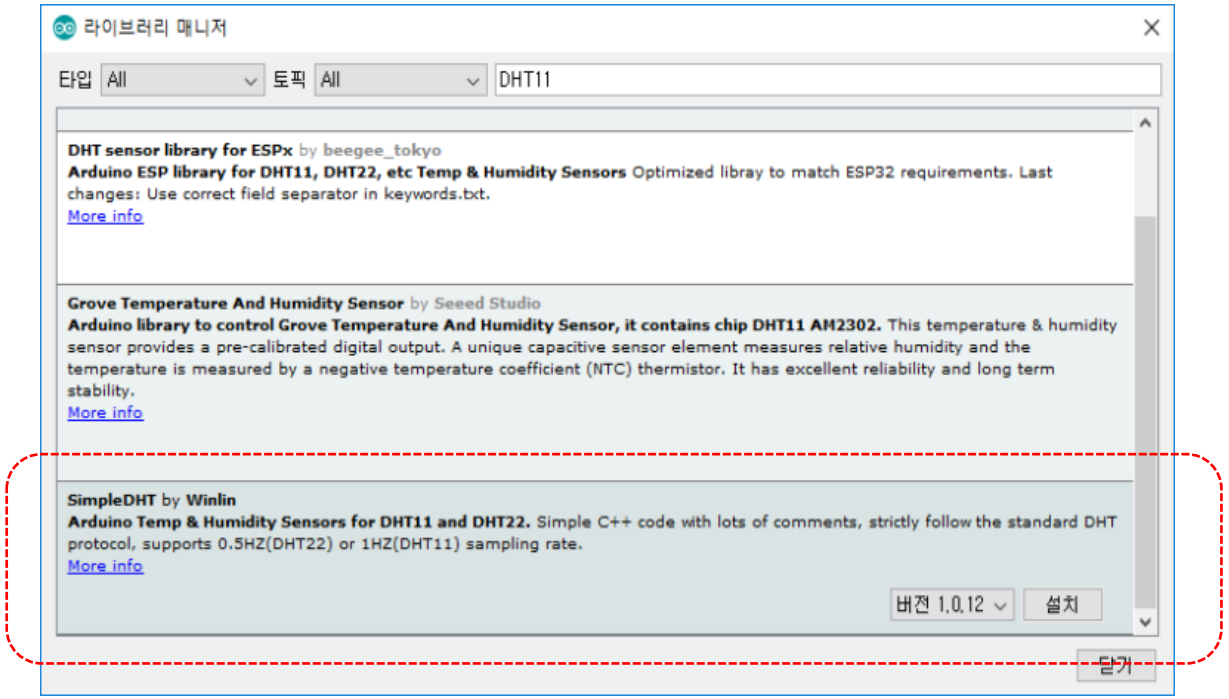
# 7.1 원격 온/습도 측정장치 #3

- 데이터 전송 방법
  - 1개의 데이터 선을 통해서 아두이노와 데이터를 주고 받는 방식 (Single Wire Two Way)
  - 아두이노에서 시작 신호를 전송한 후 대기 → 센서에서 온도와 습도 정보를 전송 (응답신호 + 40비트 이진 신호 전송)
  - 복잡함 ⇒ [SimpleDHT by Winlin] 라이브러리 사용



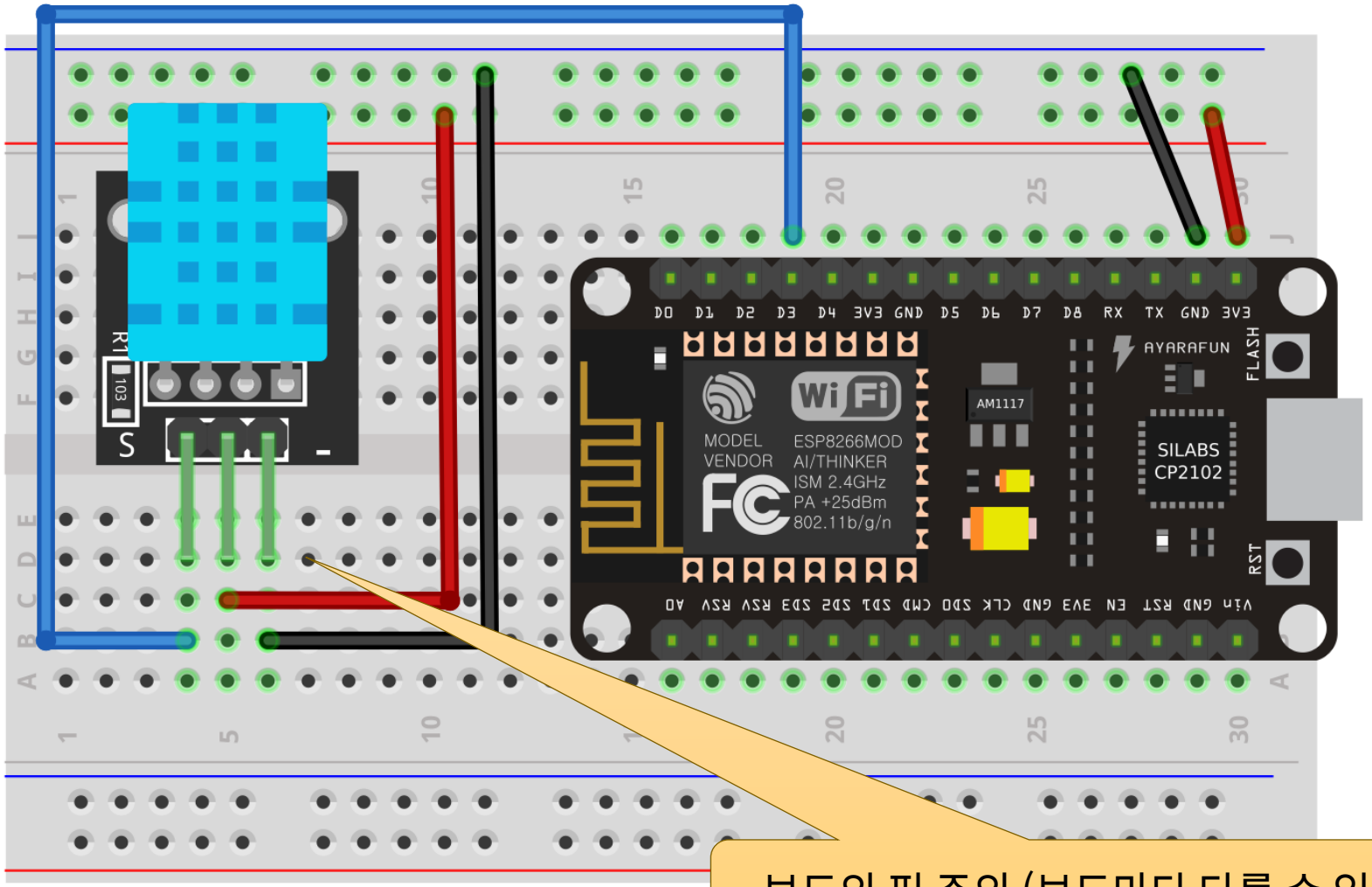
# 7.1 원격 온/습도 측정장치 #4

- DHT11/DHT22 라이브러리
  - 라이브러리 매니저를 이용하여 SimpleDHT 라이브러리 설치



# 7.1 원격 온/습도 측정장치 #5

- 회로 구성



# 7.1 원격 온/습도 측정장치 #6

- DHT11 측정 – 객체 생성, 센서 읽기 부분

```
#include <SimpleDHT.h>

// DHT 센서처리용
SimpleDHT11 dht11(D3);

// 온도/습도 저장용 변수
byte temp;
byte humi;

void loop() {

    int err;
    // DHT11 센서의 정보 읽어들이기
    if((err = dht11.read(&temp, &humi, NULL)) != SimpleDHTErrSuccess) {
        // 오류가 발생하는 경우
    }

}
```

# 7.1 원격 온/습도 측정장치 #7

- DHT11 구동 부분 – 최근 측정 후 2초가 지나는 경우마다 측정하기

```
unsigned long last;

void loop() {
  // 측정한지 2초가 지난 경우에 새로 센서로부터 측정치를 읽어들임
  unsigned long now = millis();
  // 현재 시간 - 이전의 측정 시간 > 2초
  if((now - last) > 2000) {
    // DHT11 센서의 정보 읽어들이기

    // 측정된 온도/습도 전송

    // 현재 측정 시각 저장
    last = now;
  }
}
```

# 7.1 원격 온/습도 측정장치 #8

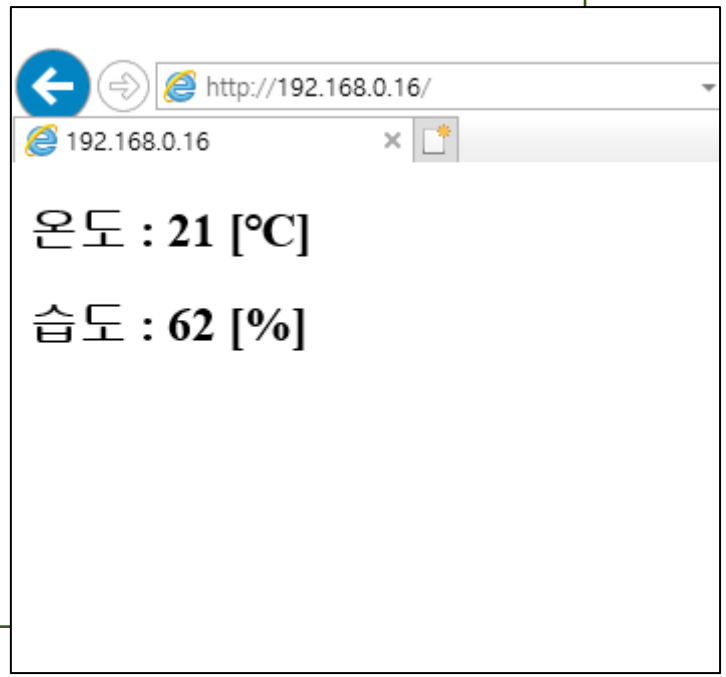
- 웹서버 : ESP8266WebServer 클래스 사용 – 예제 6.3 참고

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
#include "ssid.h"

ESP8266WebServer server(80);

void setup()
{
  // ... WiFi 접속

  // 서버 구현 부분
  server.on("/", procRoot);

  // 서버 시작
  server.begin();
}
```

# 7.1 원격 온/습도 측정장치 #9

- “/” 주소에 대한 처리 함수 : 온/습도 정보 전송

```
void procRoot() {
  char buffer[100]; // sprintf함수용 버퍼
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "<head>\r\n";
  ret += "<meta charset=\"UTF-8\">\r\n";
  ret += "</head>\r\n";
  // &#8451; 는 섭씨를 나타내는 기호
  sprintf(buffer, "<h2>온도 : %d [&#8451;]</h2>\r\n", temp);
  ret += buffer;
  sprintf(buffer, "<h2>습도 : %d [%%]</h2>\r\n", humi);
  ret += buffer;
  ret += "</body>\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}
```



# 7.1 원격 온/습도 측정장치 : BME280 #1

- BME280 : 온도/습도/고도/기압 측정용 센서 측정하기

```
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;
#define SEAPRESS 1013.25

float temp;
float humi;
float press;
float alti;

void setup()
{
  // 센서 초기화
  if(!bme.begin(0x76)) {
    Serial.println("Cannot initialize BME280");
  }
```

# 7.1 원격 온/습도 측정장치 : BME280 #2

- BME280 : 온도/습도/고도/기압 측정용 센서 측정하기

```
unsigned long last;

void loop() {
  // 측정한지 1초가 지난 경우에 새로 센서로부터 측정치를 읽어들임
  unsigned long now = millis();
  // 현재 시간 - 이전의 측정 시간 > 1초
  if((now - last) > 1000) {
    // BME280 센서 읽기
    temp = bme.readTemperature();
    humi = bme.readHumidity();
    alti = bme.readAltitude(SEAPRESS);
    press = bme.readPressure() / 100.0f;
    last = now;
  }
  // 클라이언트 접속 처리
  server.handleClient();
}
```

# 7.1 원격 온/습도 측정장치 : BME280 #3

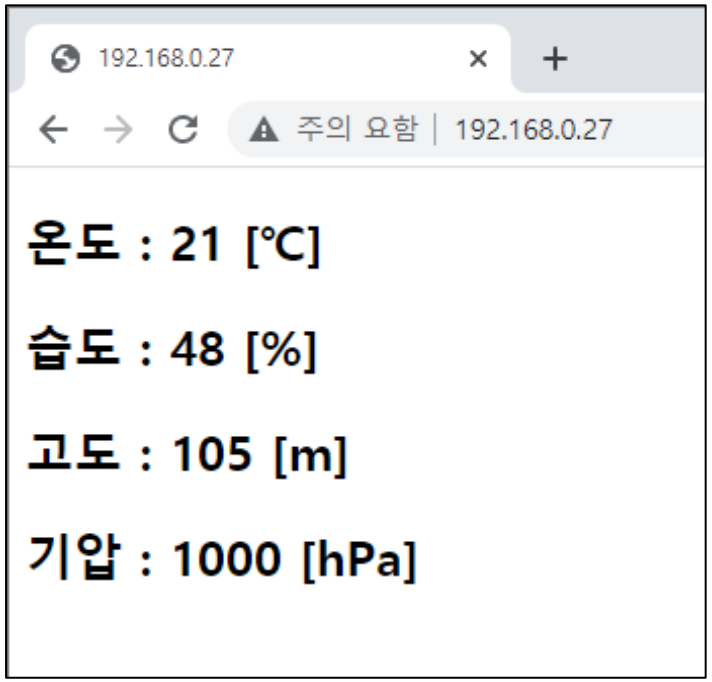
- “/” 주소에 대한 처리 함수 : 온/습도 정보 전송

```
void procRoot() {
  char buffer[100]; // sprintf함수용 버퍼
  String ret = "<!DOCTYPE HTML>\r\n";
  ret += "<html>\r\n";
  ret += "<head>\r\n";
  ret += "<meta charset=\"UTF-8\">\r\n";
  ret += "</head>\r\n";
  sprintf(buffer, "<h2>온도 : %d [&#8451;]</h2>\r\n", (int)temp);
  ret += buffer;
  sprintf(buffer, "<h2>습도 : %d [%%]</h2>\r\n", (int)humi);
  ret += buffer;
  sprintf(buffer, "<h2>고도 : %d [m]</h2>\r\n", (int)alti);
  ret += buffer;
  sprintf(buffer, "<h2>기압 : %d [hPa]</h2>\r\n", (int)press);
  ret += buffer;
  ret += "</body>\r\n";
  ret += "</html>\r\n";
  server.send(200, "text/html", ret);
}
```

# 7.1 원격 온/습도 측정장치 : BME280 #4

- HTML 페이지 소스

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
</head>
<h2>온도 : 21 [&#8451;]</h2>
<h2>습도 : 48 [%]</h2>
<h2>고도 : 105 [m]</h2>
<h2>기압 : 1000 [hPa]</h2>
</body>
</html>
```




- 페이지 자동 갱신 : HTML의 meta 태그 http-equiv 속성 사용

```
<head>
<meta charset="UTF-8">
<meta http-equiv="refresh" content="1">
</head>
```

1초마다 페이지 갱신

# 추가 실습

- BME280 센서 사용 예제에 조도 센서를 추가하도록 하라.

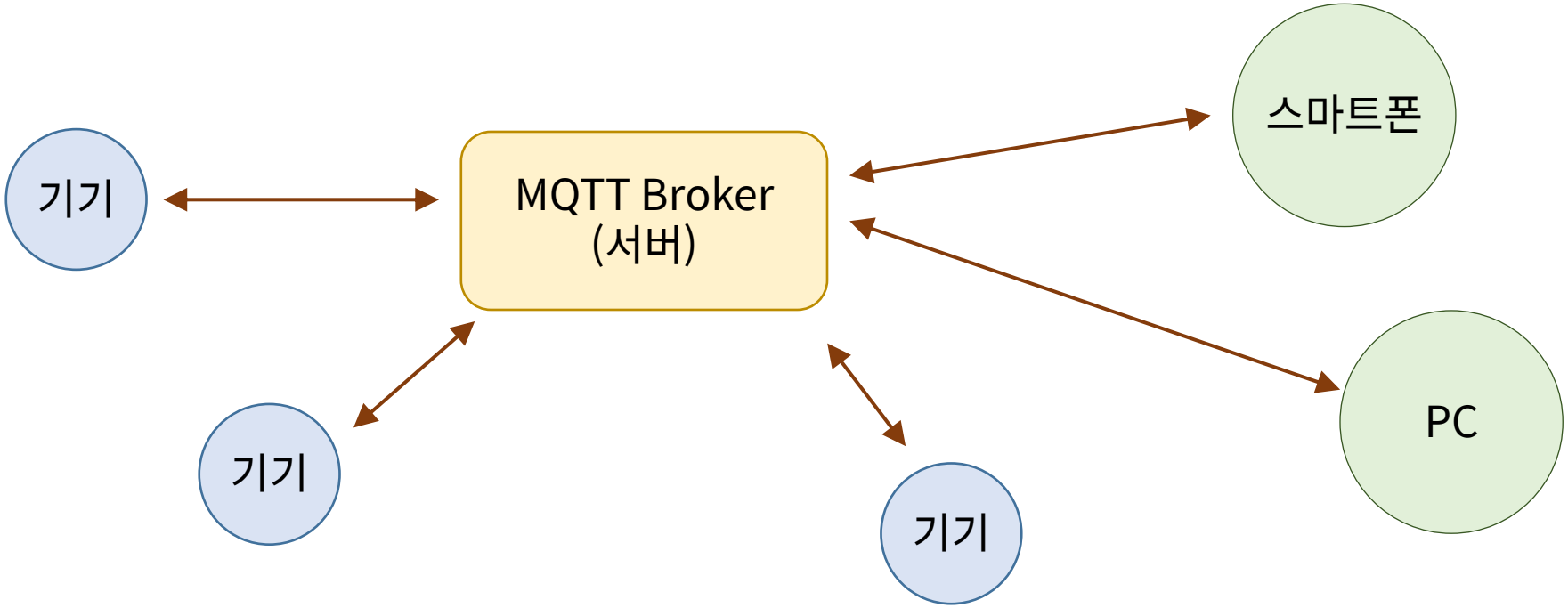
# 7.2 MQTT 서비스

- NodeMCU 서버의 단점
  - 공유기에 연결되는 경우 외부 네트워크망에서 접속 불가능 (보통 내부 사설망)
  - 모바일 기기 확인 시 웹 브라우저를 항상 수행해야 함

- 단점의 극복 방법
  - 공인 IP로 접속 가능한 외부 서버를 별도로 구성
  - NodeMCU에서 측정값을 외부 서버에 전송
  - 웹 프로토콜 외의 다른 프로토콜 (예: MQTT) 사용
    → 모바일 기기에서 해당 프로토콜을 지원하는 전용 앱 사용

# 7.2 MQTT 서비스

- MQTT (Message Queue for Telemetry Transport) 프로토콜
  - 소규모의 통신장치 사이의 통신을 위하여 개발
  - 소규모의 메시지 전송용 프로토콜
  - 저전력 장비 / 저속의 네트워크 환경에 적합
  - 동작 방식 : 구독 (subscribe) – 모니터링 , 발행 (publish) – 메시지 전송



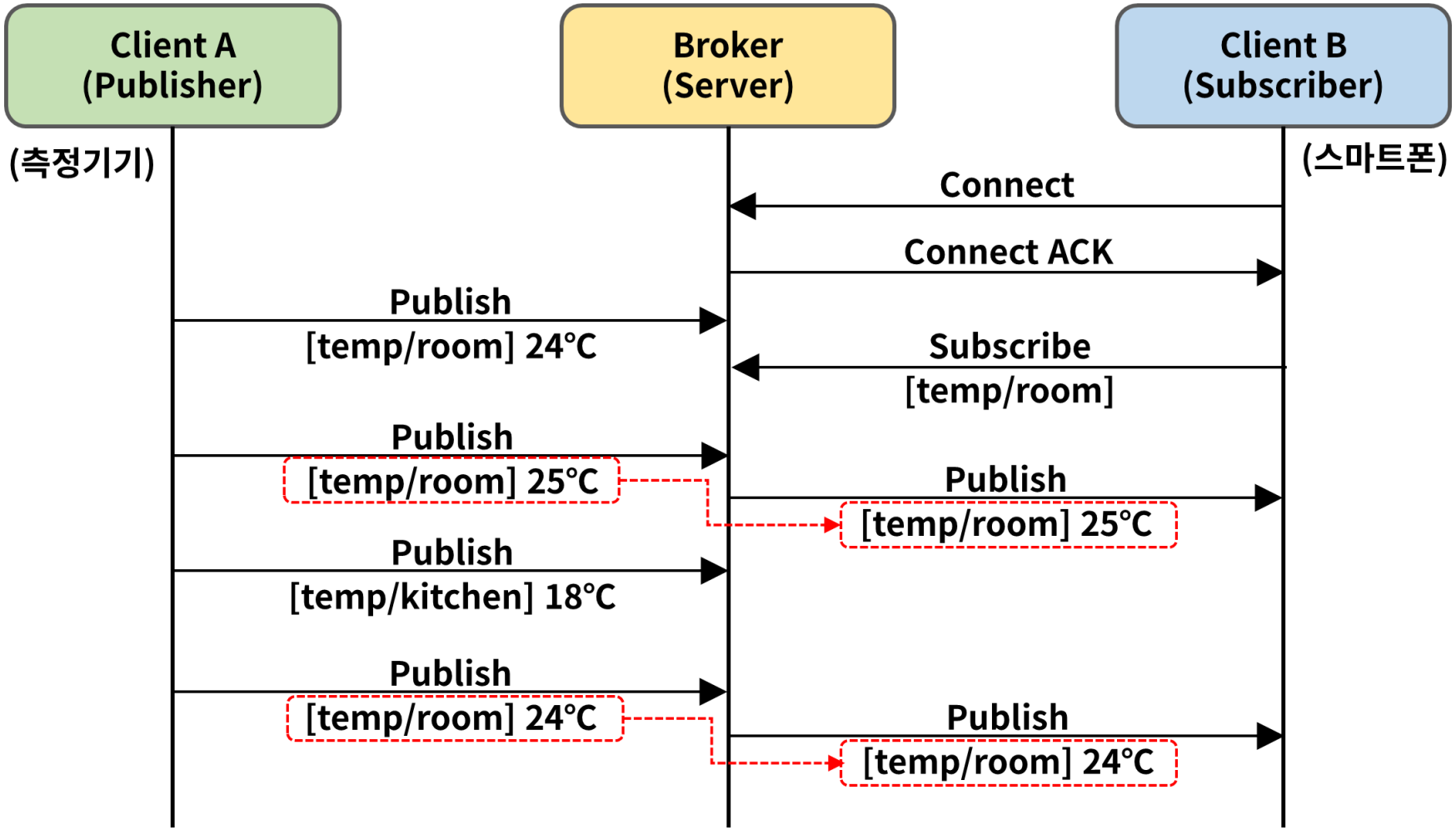
# 7.2 MQTT 서비스

- MQTT (Message Queue for Telemetry Transport) 프로토콜
  - 메시지 발행 → 서버 (Broker) → 구독하는 모든 기기
- 전송 신뢰도 설정 (QoS: Quality of Service)
  - QoS 0 : 메시지 한 번만 전달, 전달여부 확인 하지 않음
  - QoS 1 : 메시지는 한 번 이상 전달, 확인이 없으므로 중복 전달 가능.
  - QoS 2 : 메시지는 한 번만 전달, 과정을 추적한다. 가장 높은 품질
- 메시지의 구분 : 토픽(topic)
  - 메시지 발행 방식으로 계층구조를 사용하며 “/”로 구분하여 표시

```
home/bedroom/temperature
home/bedroom/humidity
home/kitchen/temperature
home/kitchen/humidity
```

# 7.2 MQTT 서비스

- MQTT 전송 과정



# 7.2 MQTT 서비스

- 메시지 구독(subscribe)하는 경우 토픽 지정 방식
  - 여러 개의 토픽을 지정할 수 있는 와일드 카드(wild card) 사용
  - “+” : 하나의 레벨을 임의로 지정
  - “#” : 여러 개의 레벨을 임의로 지정
  - 예]

```
home/bedroom/temperature
home/bedroom/humidity
home/kitchen/temperature
home/kitchen/humidity
```

  - ➢ home/+/temperature ⇒ home/bedroom/temperature 및 home/kitchen/temperature
  - ➢ home/bedroom/# ⇒ /home/bedroom/으로 시작하는 모든 토픽 구독

# 7.2 MQTT 서비스

- Broker 구축
  - 자신의 서버를 구축하여 사용 (Windows, Linux, Raspberry pi …)
  - 대표적인 서버 : mosquito (https://mosquitto.org)
  - 별도의 서버 필요

- 외부 MQTT Broker 서비스
  - 다양한 기업체에서 서비스 제공 (무료/유료)
  - HiveMQ : 회원가입 없음, 무료, 비공개 없음 ⇒ 외부에 공개
    - https://www.hivemq.com/public-mqtt-broker/
  - CloudMQTT : 비공개, 회원가입 필요, 무료 회원은 제한사항이 있음
    - https://www.cloudmqtt.com/plans.html

# 7.2 MQTT 서비스 – HiveMQ

- HiveMQ 사용
  - Public MQTT Broder & Browser Client
  - https://www.hivemq.com/public-mqtt-broker/

Our **Public HiveMQ MQTT broker** is open for anyone to use. Feel free to write an MQTT client that connects with this broker. We have a **dashboard** so you can see the amount of traffic on this broker. We also keep a list of **MQTT client libraries** that can be used to connect to HiveMQ.

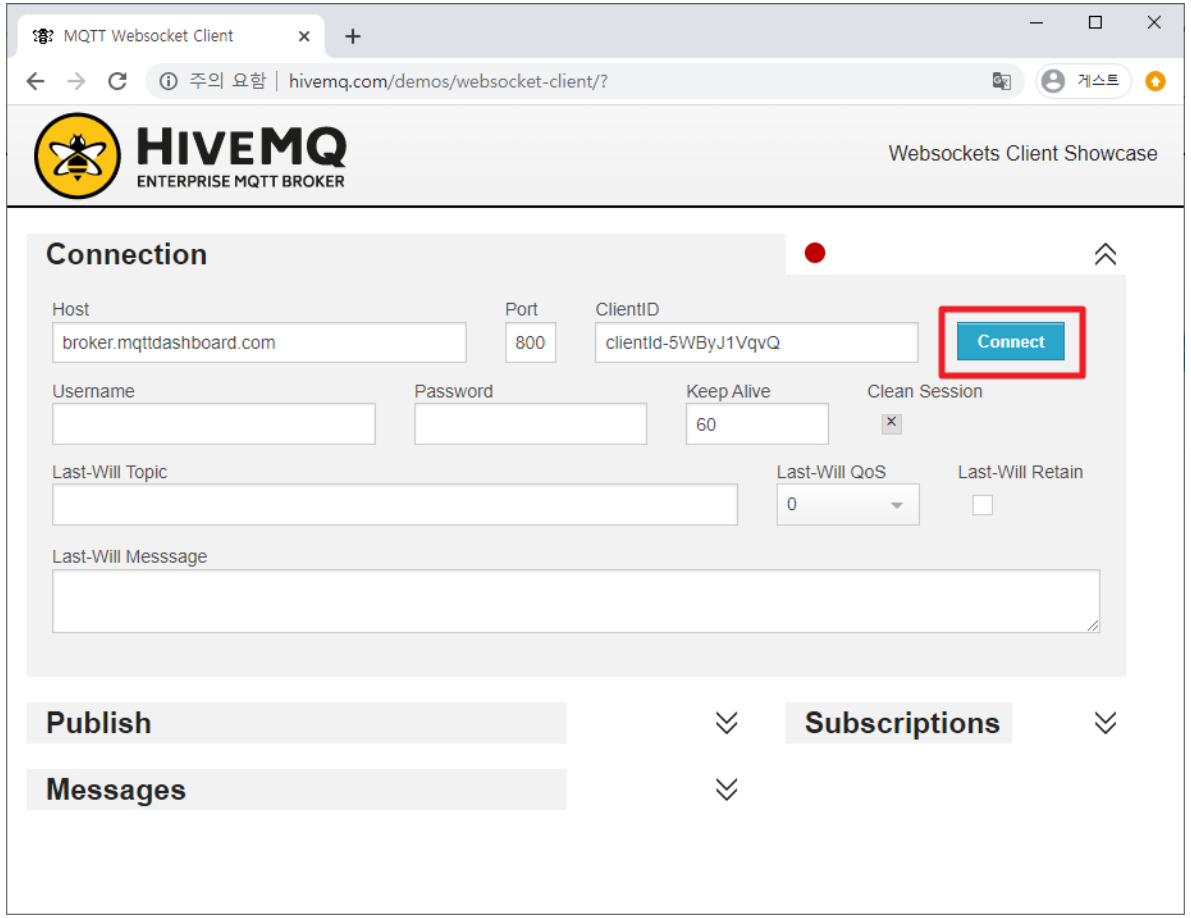
You can access the broker at:

Broker: `broker.hivemq.com`

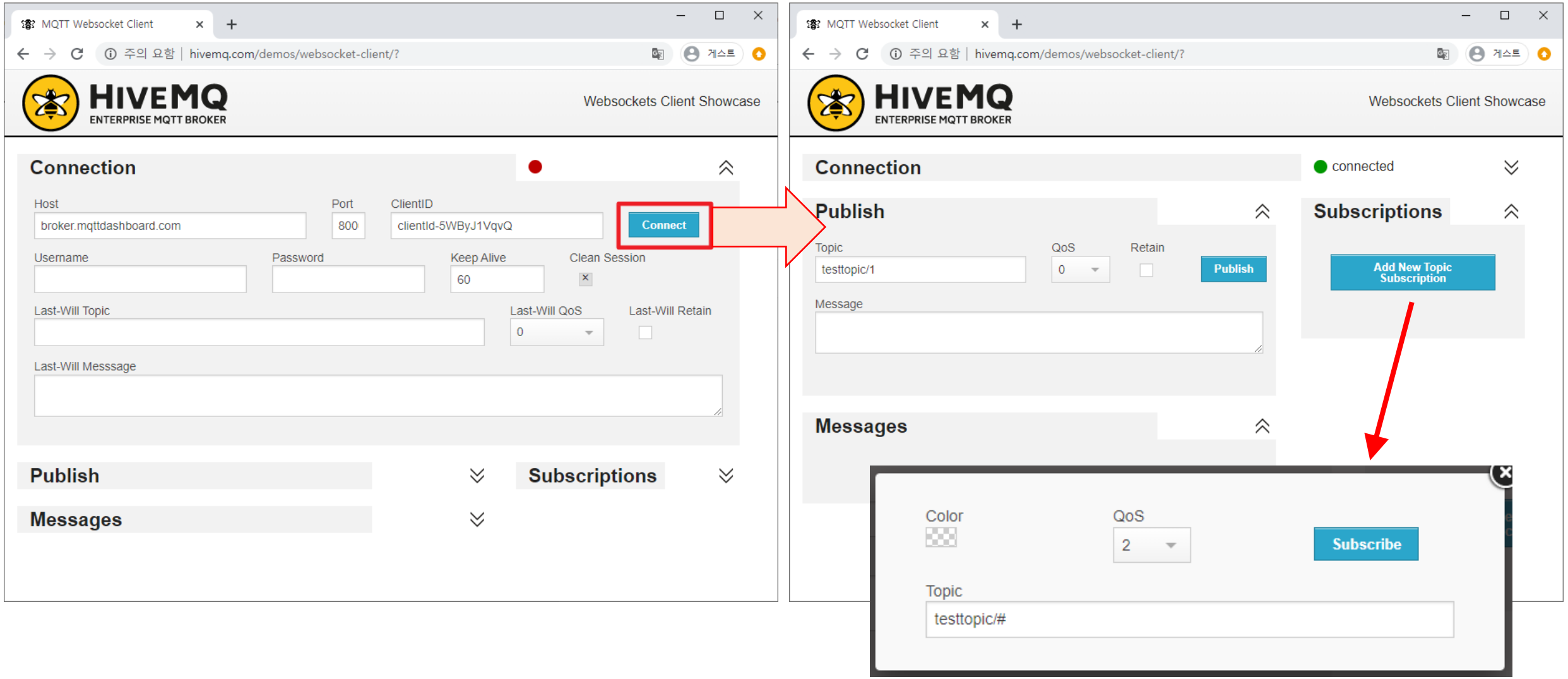
TCP Port: 1883

Websocket Port: 8000

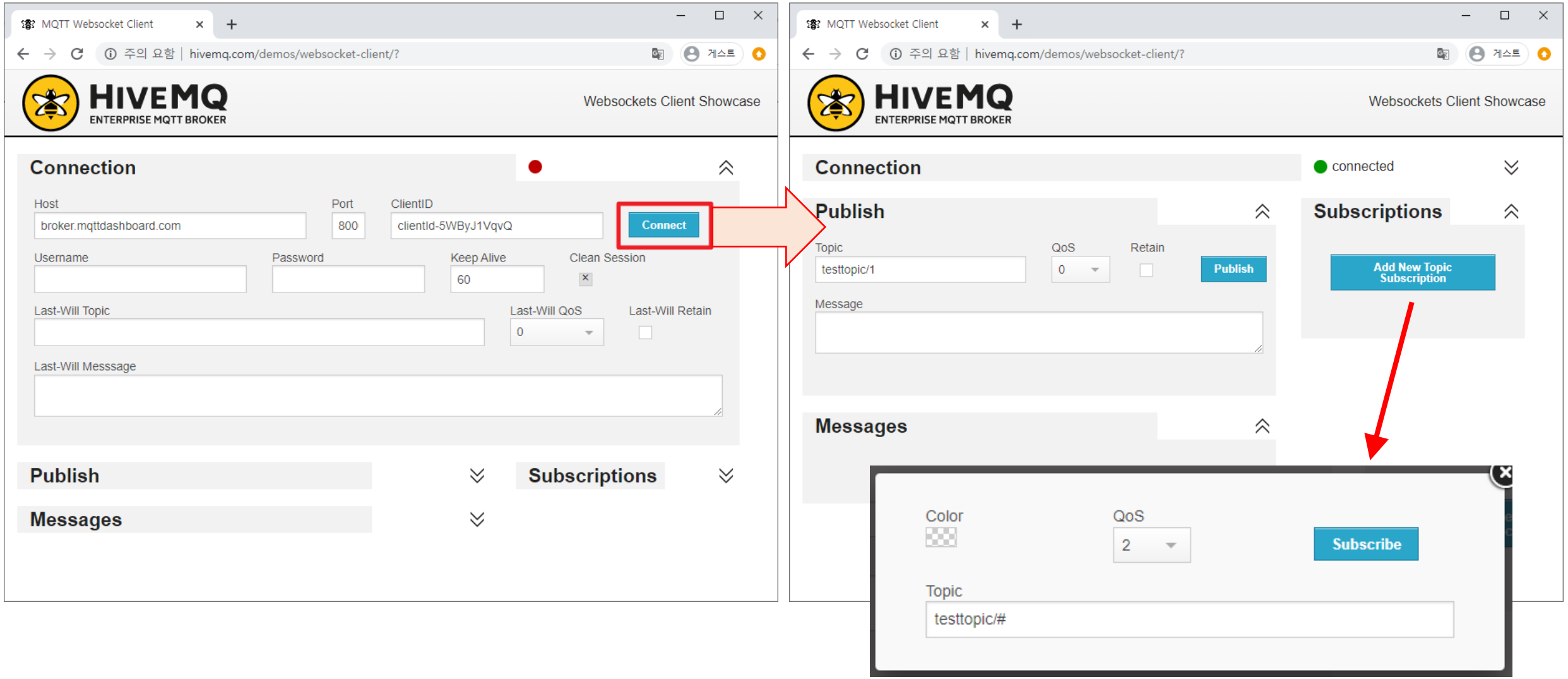
MQTT Broker



# 7.2 MQTT 서비스 – HiveMQ

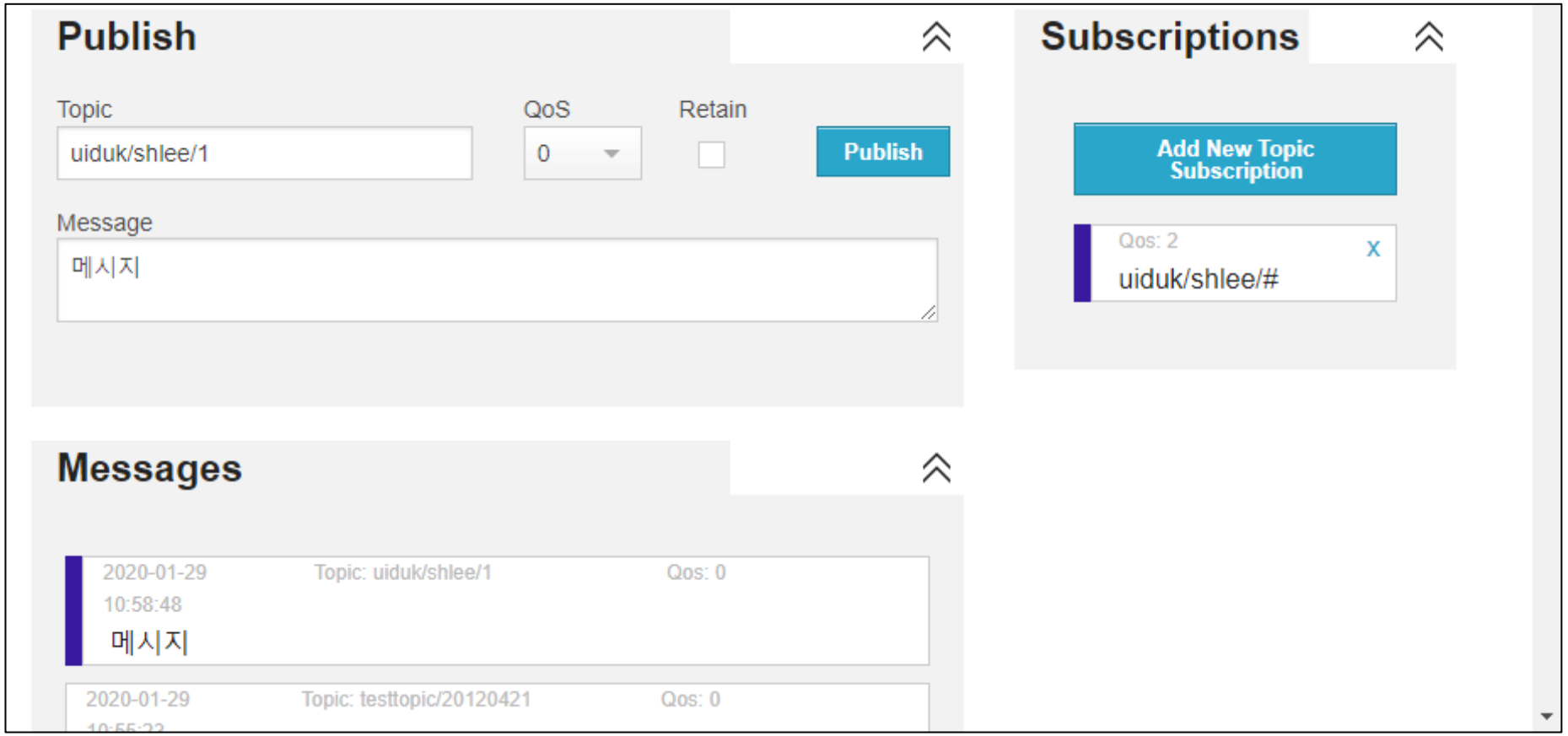


# 7.2 MQTT 서비스 – HiveMQ



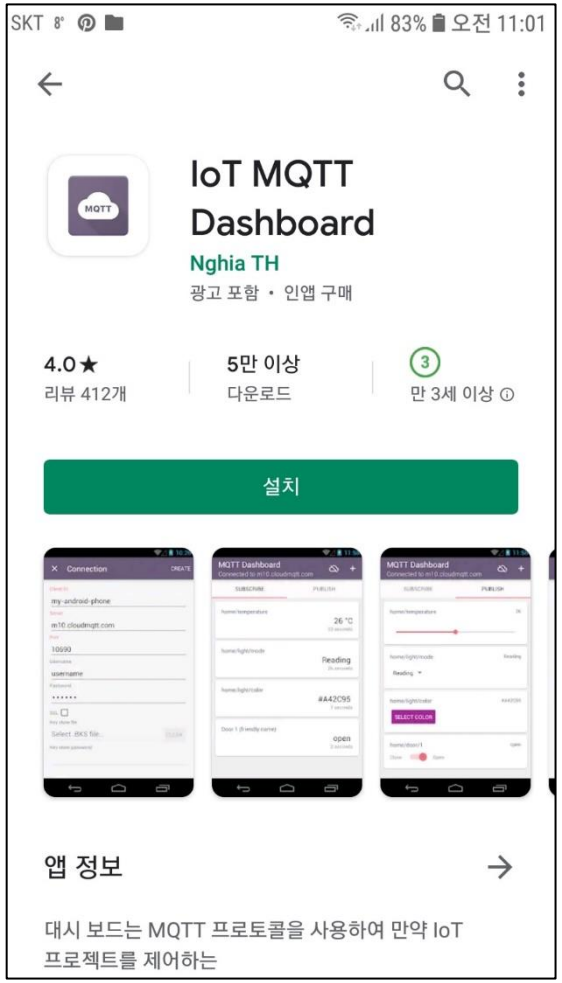
# 7.2 MQTT 서비스 – HiveMQ

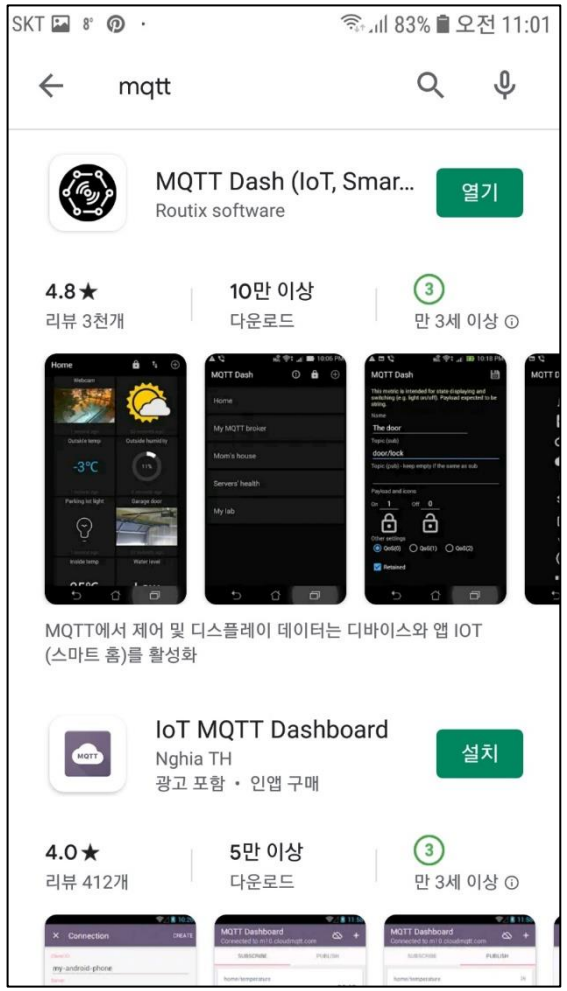
- 자신만의 토픽을 구성
  - Subscribe : 새로운 이름으로 topic 지정 (예: uiduk/shlee/#)
  - Publish : 지정한 topic에서 발행 (예: uiduk/shlee/1)



# 7.2 MQTT 서비스 – HiveMQ

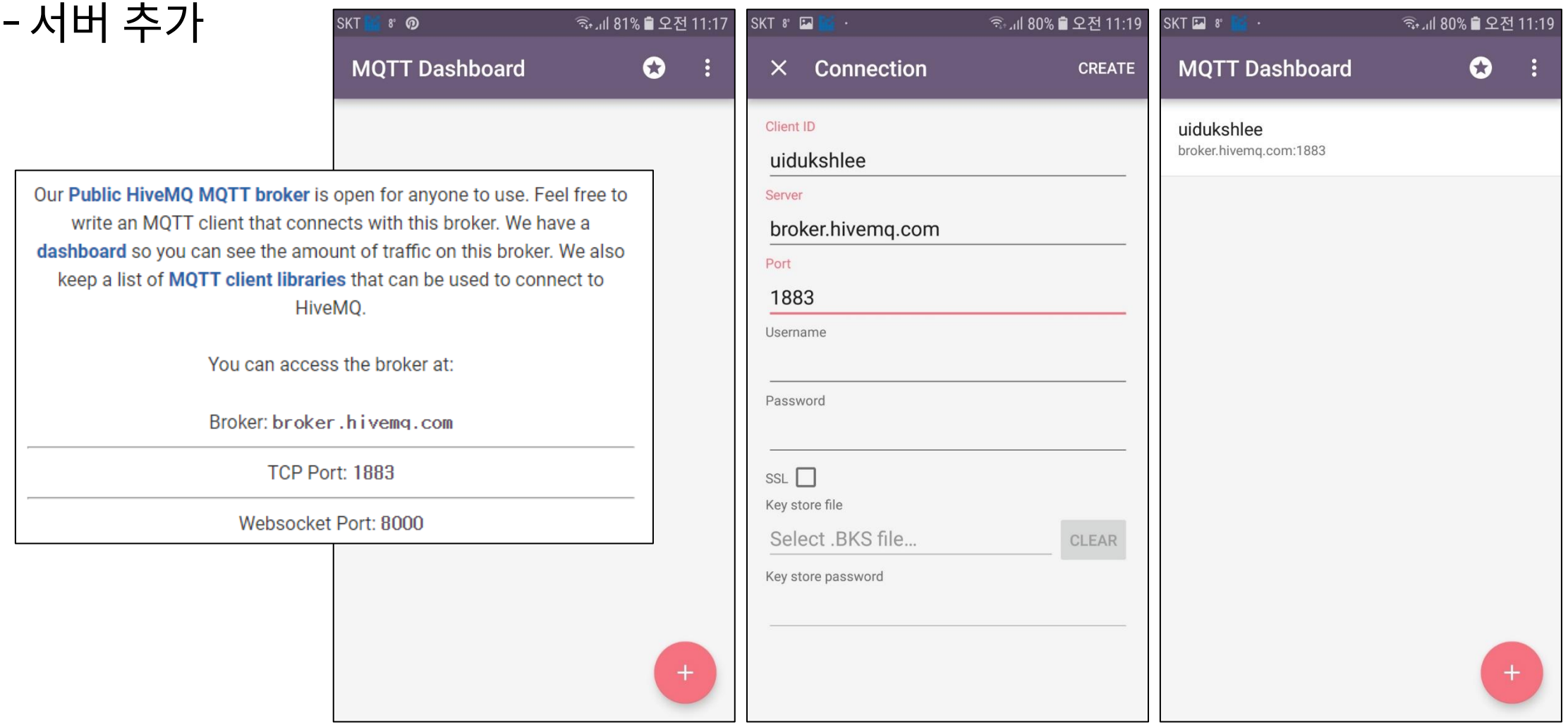
- 스마트폰과 연동
  - MQTT 지원 앱 설치
    - ➢ 예] Android
      MQTT Dashboard



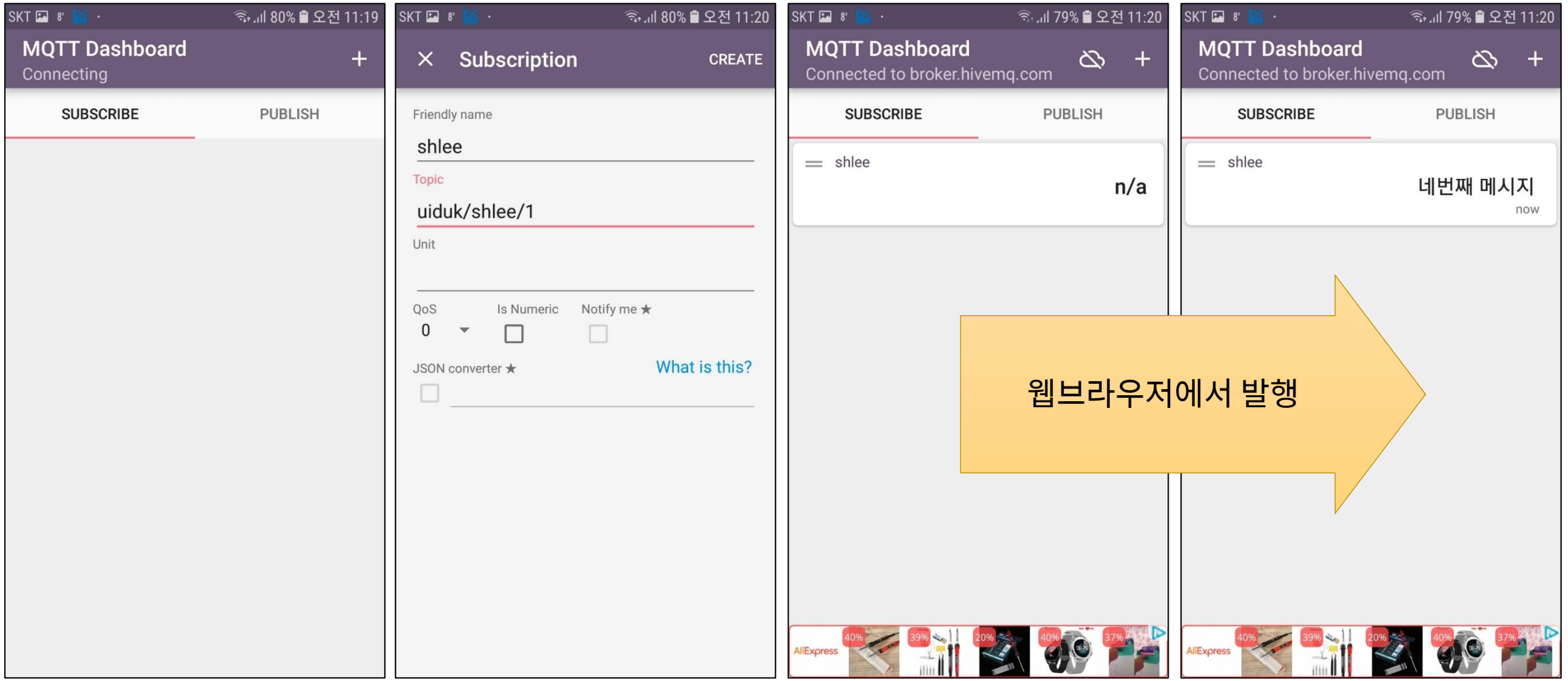
# 7.2 MQTT 서비스 – HiveMQ

- MQTT Dashboard 설정
  - 서버 추가



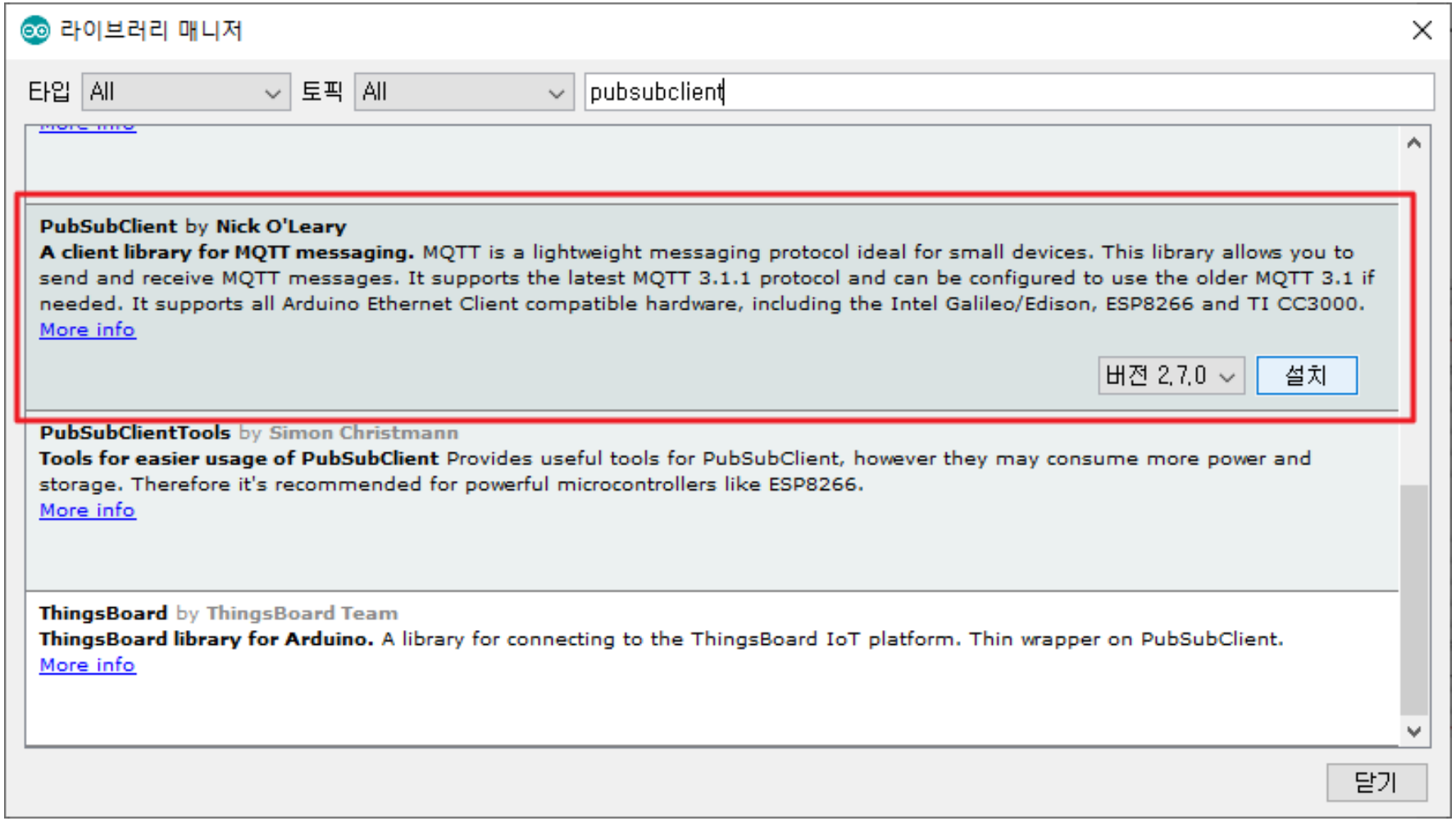
# 7.2 MQTT 서비스 – HiveMQ

- MQTT Dashboard 구독 (uiduk/shlee/1)



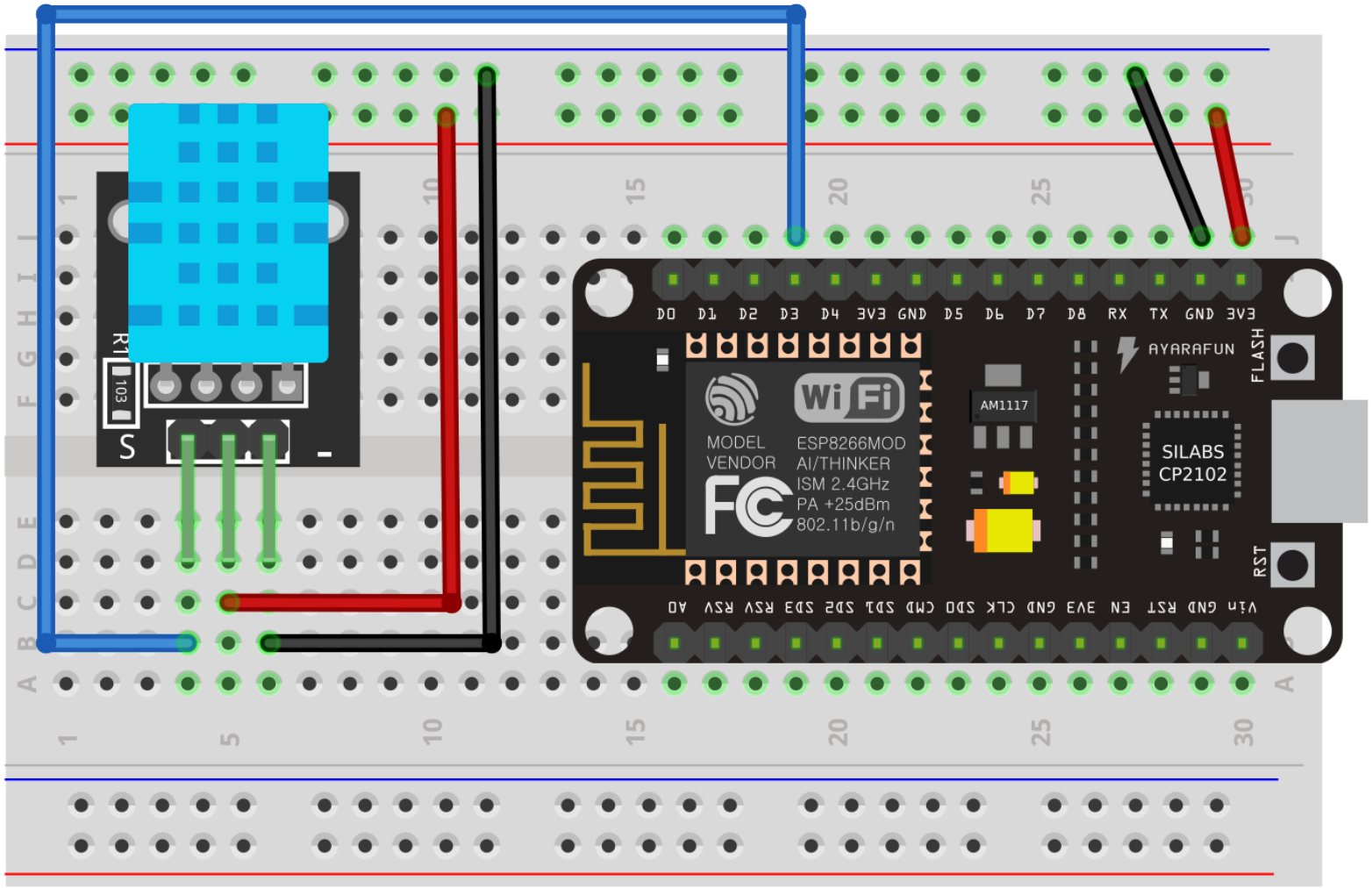
# 예제 7-2. MQTT를 이용한 원격 센서 측정

- 아두이노의 MQTT 라이브러리 : PubSubClient



# 예제 7-2. MQTT를 이용한 원격 센서 측정 #1

- 회로 구성 : 예제 7-1과 동일



# 예제 7-2. MQTT를 이용한 원격 센서 측정 #2

- DHT11 측정 – 객체 생성, 센서 읽기 부분 (예제 7-1과 동일)

```
#include <SimpleDHT.h>

// DHT 센서처리용
SimpleDHT11 dht11(D3);

// 온도/습도 저장용 변수
byte temp;
byte humi;

void loop() {

    int err;
    // DHT11 센서의 정보 읽어들이기
    if((err = dht11.read(&temp, &humi, NULL)) != SimpleDHTErrSuccess) {
        // 오류가 발생하는 경우
    }

}
```

# 예제 7-2. MQTT를 이용한 원격 센서 측정 #3

- DHT11 구동 부분 – 측정 후 2초가 지나는 경우마다 측정 (7-1과 동일)

```
unsigned long lastmeas; // 최근 측정한 시각
unsigned long lastpub;  // 최근 발행한 시각

void loop() {
  // 측정한지 2초가 지난 경우에 새로 센서로부터 측정치를 읽어들임
  unsigned long now = millis();
  // 현재 시간 - 이전의 측정 시간 > 2초
  if((now - lastmeas) > 2000) {
    // DHT11 센서의 정보 읽어들이기

    // 측정된 온도/습도 전송

    // 현재 측정 시각 저장
    lastmeas = now;
  }
}
```

# 예제 7-2. MQTT를 이용한 원격 센서 측정 #4

- MQTT용 라이브러리 사용 : 헤더파일

Our **Public HiveMQ MQTT broker** is open for anyone to use. Feel free to write an MQTT client that connects with this broker. We have a **dashboard** so you can see the amount of traffic on this broker. We also keep a list of **MQTT client libraries** that can be used to connect to HiveMQ.

You can access the broker at:

Broker: broker.hivemq.com

TCP Port: 1883

Websocket Port: 8000

```
#include <PubSubClient.h>

// MQTT용 객체
WiFiClient client;
PubSubClient mqtt(client);

#define MQTT_SERVER "broker.hivemq.com"
#define MQTT_PORT 1883

void callback(char *topic, byte *payload, unsigned int length);

void setup()
{
  // ...
  // MQTT용 클라이언트 초기화 (서버주소 및 포트)
  mqtt.setServer(MQTT_SERVER, MQTT_PORT);
  mqtt.setCallback(callback); // MQTT용 콜백함수 (메시지 수신시 처리함수)
}
```

# 예제 7-2. MQTT를 이용한 원격 센서 측정 #5

- MQTT용 라이브러리 사용 : loop() #1
  - 서버에 접속을 확인한 후 끊어진 경우 재 접속 & 구독

```
void loop() {
  // MQTT 서버 접속이 안되었거나 끊어진 경우 재접속
  if(!mqtt.connected()) {
    // 접속이 실패하는 경우 2초 후에 다시 시도
    if(!mqtt.connect("uidukshlee")) {
      Serial.println("Connect failed !!!");
      delay(2000);
      return;
    }
    if(!mqtt.subscribe("uiduk/shlee/#")) {
      Serial.println("Subscribe failed !!!");
    }
  }
```

# 예제 7-2. MQTT를 이용한 원격 센서 측정 #6

- MQTT용 라이브러리 사용 : loop() #2
  - 매 10초마다 broker로 발행

```
  // 온/습도 측정 (매 1초마다)

  // 매 10초마다 mqtt로 발행(publish)
  now = millis();
  if((now - lastpub) > 10000) {
    char str[10];
    sprintf(str, "%d", temp);
    mqtt.publish("uiduk/shlee/dht11/temperature", str);
    sprintf(str, "%d", humi);
    mqtt.publish("uiduk/shlee/dht11/humidity", str);
    lastpub = now;
  }

  // MQTT 처리
  mqtt.loop();
}
```

# 예제 7-2. MQTT를 이용한 원격 센서 측정 #7

- MQTT용 라이브러리 사용 : 구독 처리 함수
  - 서버 접속 후 구독한 토픽이 발행된 경우 구독 처리

```
// MQTT용 콜백함수, 구독하고 있는 토픽에 메시지 수신시 처리
void callback(char *topic, byte *payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i=0;i<length;i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
```

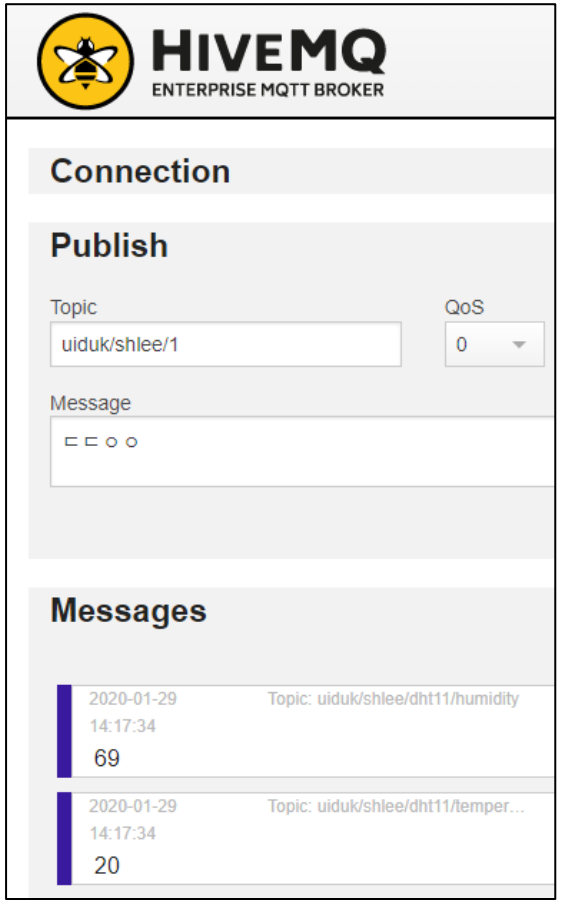
구독 토픽 수신 처리 결과

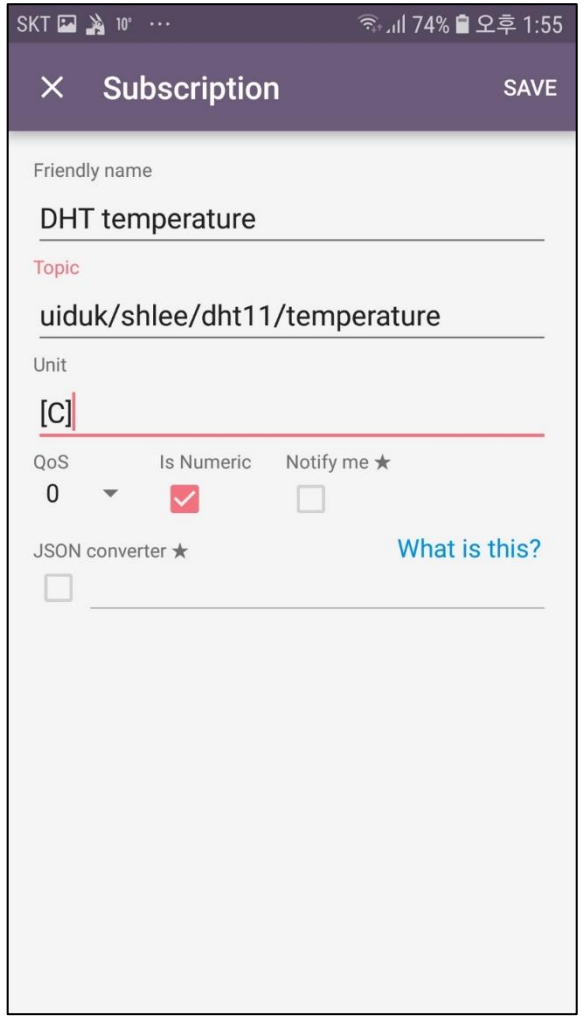
COM10

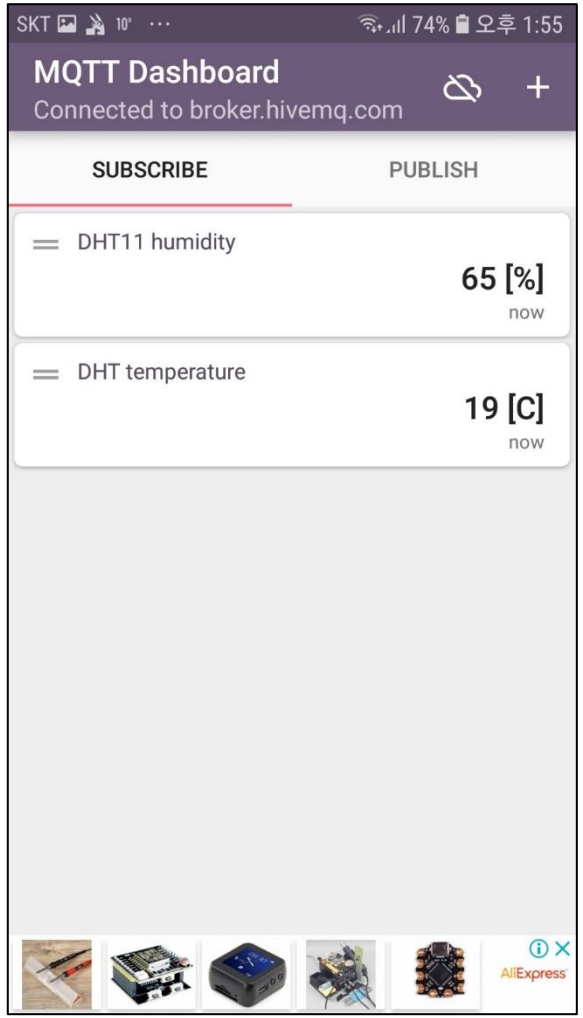
```
rld§§|§$§|└↑┘┘§└§1§┘b|§§┐§p└§§|‡c§↑B§§oo§loo§§b↑c└p§§bl`┐rlp
Connecting.
Temperature : 20
Humidity : 67
Temperature : 19
Humidity : 67
Temperature : 20
Humidity : 67
Temperature : 20
Humidity : 67
Temperature : 20
Humidity : 68
Message arrived [uiduk/shlee/dht11/temperature] 20
Message arrived [uiduk/shlee/dht11/humidity] 67
Temperature : 20
Humidity : 67
```

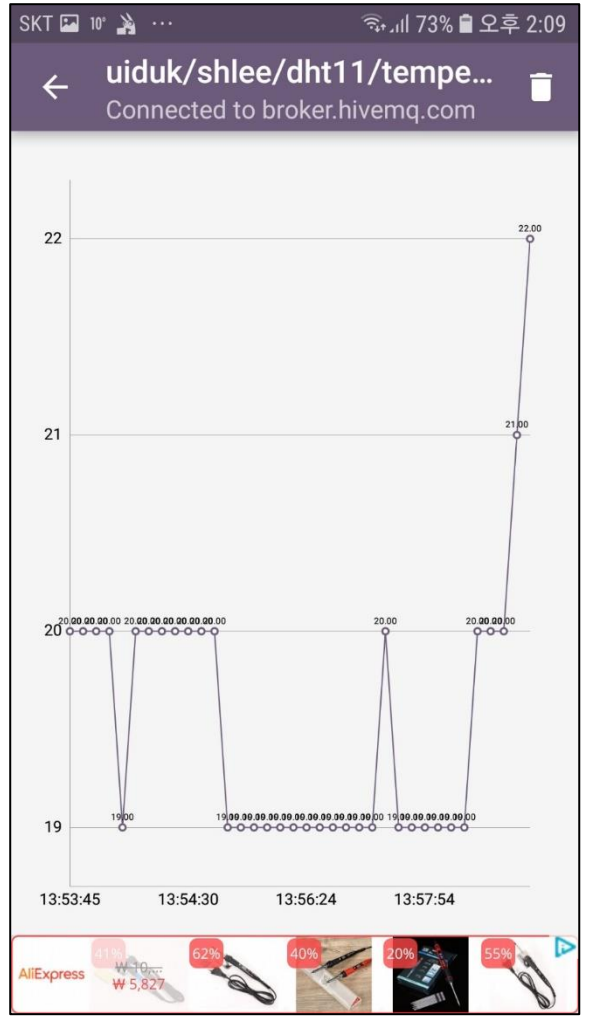
# 예제 7-2. MQTT를 이용한 원격 센서 측정 #8

- 측정 데이터 확인 (MQTT 브라우저 & 스마트폰)



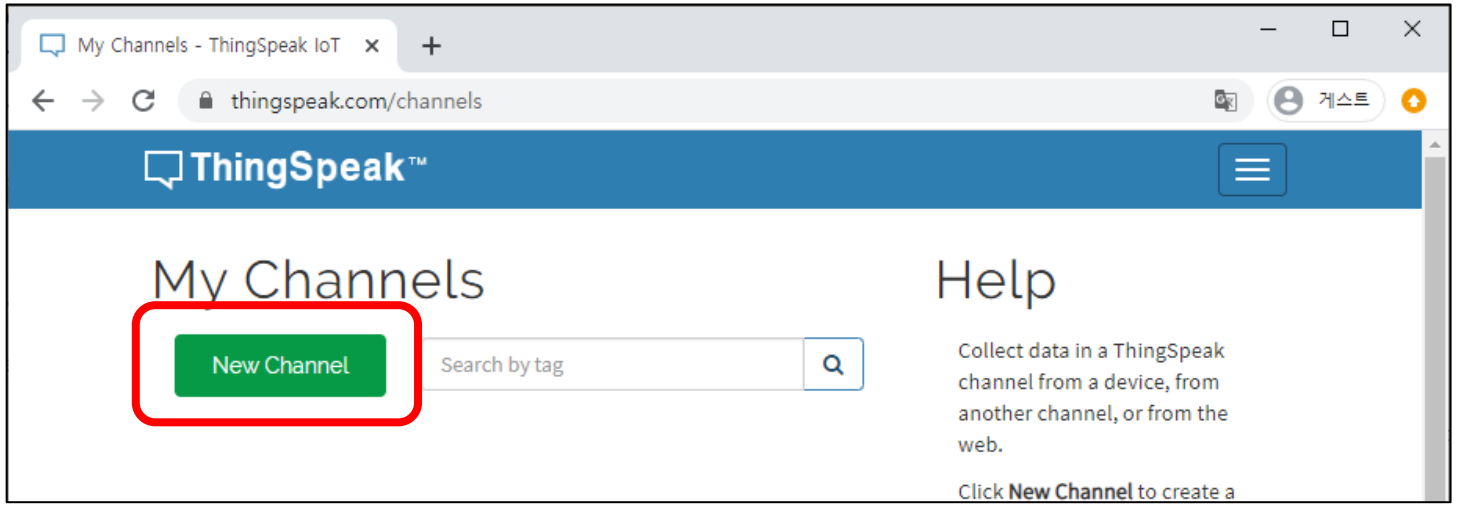



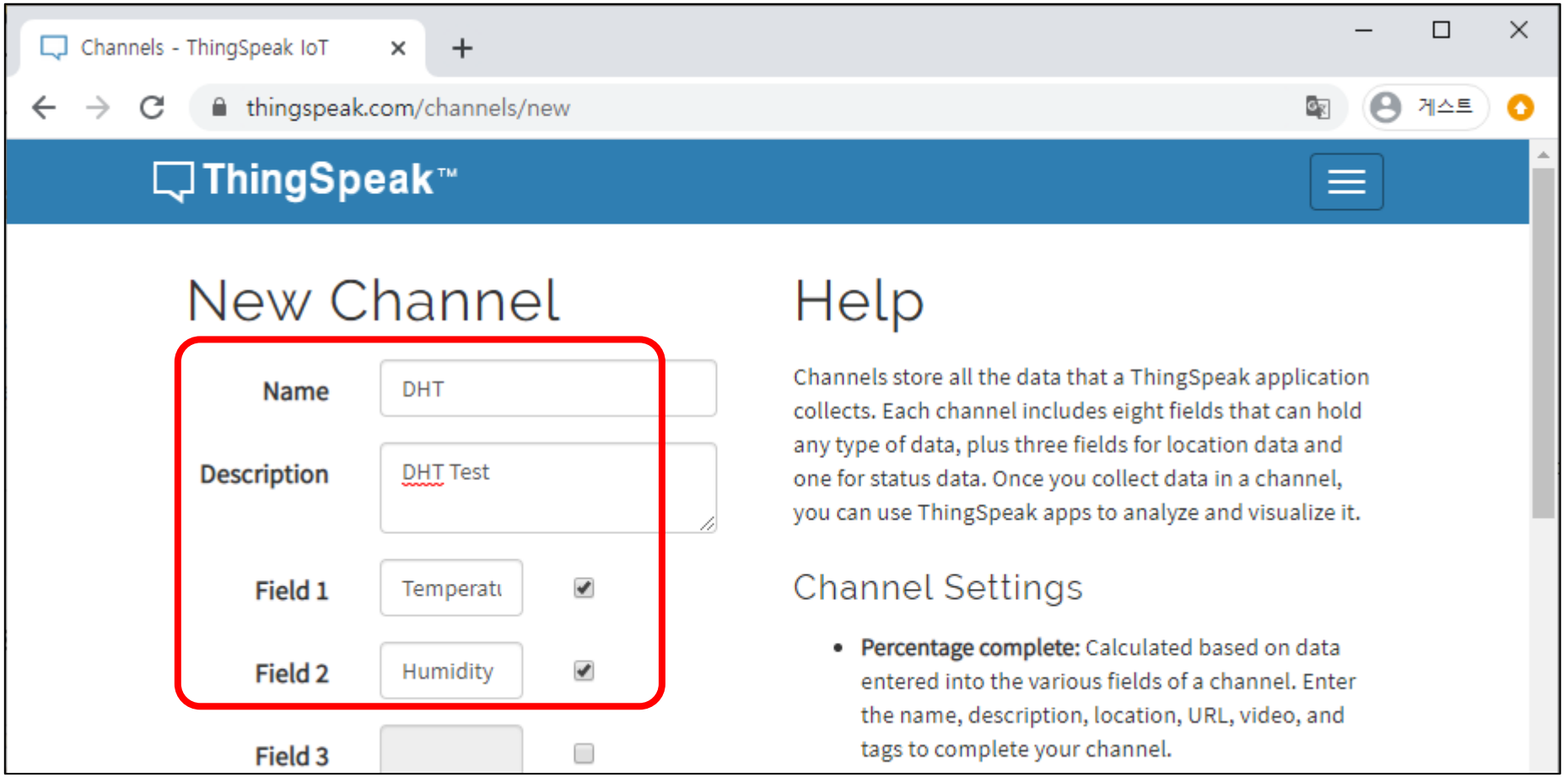
# 7.3 Thingspeak를 이용한 원격 데이터 수집

- Thingspeak
  - Matworks사에서 제공하는 사물인터넷 분석 사이트
  - 다양하게 획득한 데이터의 MATLAB를 이용한 분석 가능
  - 스마트폰을 통하여 위젯 형태를 통하여 언제나 확인 가능
- 사용 순서
  - 회원 가입 → 채널 생성 → 채널 접근 API Key 획득



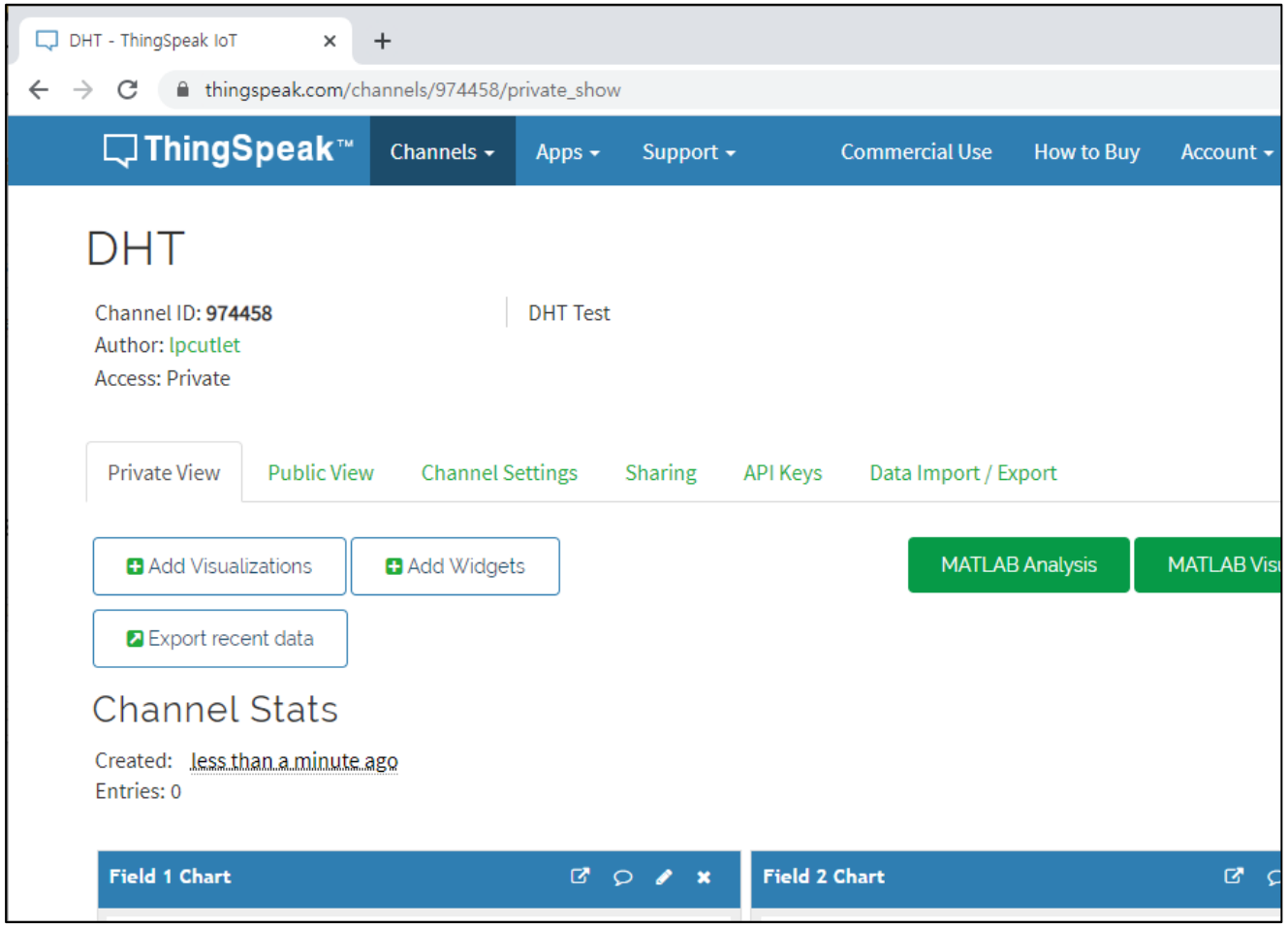
# 7.3 Thingspeak를 이용한 원격 데이터 수집

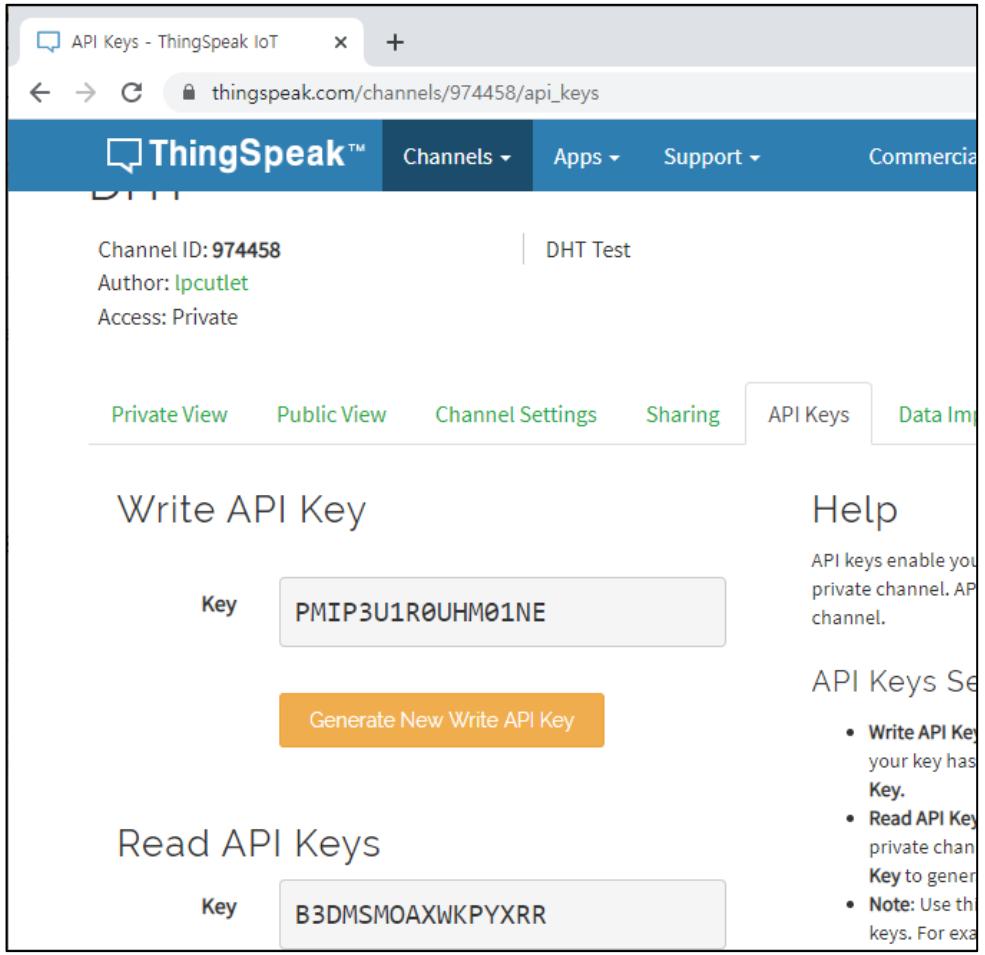
- 채널 생성 및 필드 추가



# 7.3 Thingspeak를 이용한 원격 데이터 수집

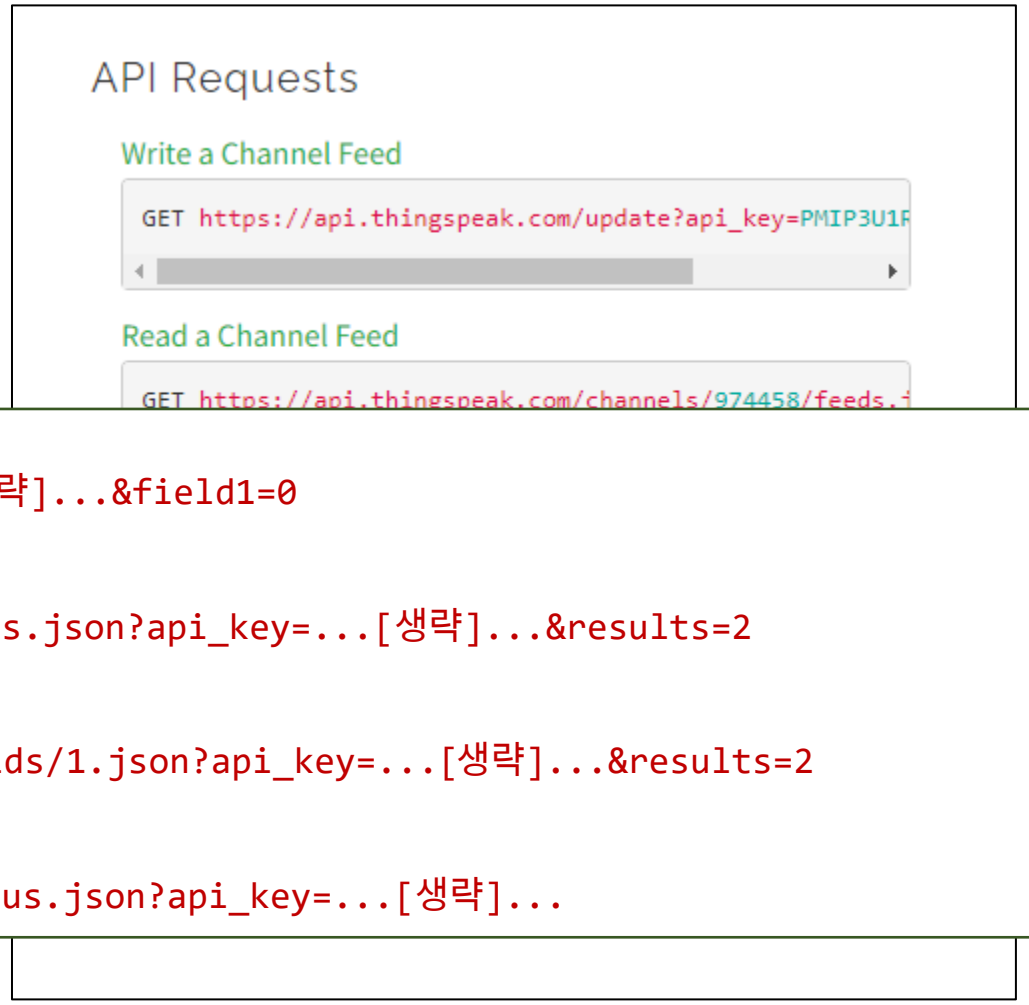
- 채널 정보 확인 및 API Key 획득



# 7.3 Thinkspeak를 이용한 원격 데이터 수집

- 채널 정보 확인 및 API Key 획득
  - API Keys 탭에서 오른쪽 아랫 부분
  - API Requests를 확인




```
Write a Channel Feed
GET https://api.thingspeak.com/update?api_key=...[생략]...&field1=0

Read a Channel Feed
GET https://api.thingspeak.com/channels/1380944/feeds.json?api_key=...[생략]...&results=2

Read a Channel Field
GET https://api.thingspeak.com/channels/1380944/fields/1.json?api_key=...[생략]...&results=2

Read Channel Status Updates
GET https://api.thingspeak.com/channels/1380944/status.json?api_key=...[생략]...
```

# 7.3 Thingspeak를 이용한 원격 데이터 수집

- Thingspeak의 API

```
Write a Channel Feed
GET https://api.thingspeak.com/update?api_key=...[생략]...&field1=0
```

- 교재와 달라진 점 : HTTP 를 사용하지 않고 HTTPS를 사용
- HTTPS : Secure HTTP 프로토콜이며, 암호화된 통신을 사용
- 교재의 스케치를 사용하는 경우 작동이 안됨

→ Mathworks 사의 라이브러리 사용
→ 라이브러리 매니저에서 “ThingSpeak”를 사용

# 7.3 Thingspeak : BME280 #1

- BME280 : 온도/습도/고도/기압 측정용 센서 측정 (앞 예제와 동일)

```
#include <Adafruit_BME280.h>

// Thingspeak 쓰기용 Key
#define KEY "PMIP3U1R0UHM01NE"

Adafruit_BME280 bme;
#define SEAPRESS 1013.25

float temp, humi, press,  alti;

void setup()
{
  // 센서 초기화
  if(!bme.begin(0x76)) {
    Serial.println("Cannot initialize BME280");
  }
```

교재의 DHT11 사용 예제와
비교해가면서 수행할 것

# 7.3 Thingspeak : BME280 #2

- BME280 : 온도/습도/고도/기압 측정용 센서 측정 (앞 예제와 동일)

```
unsigned long lastmeas; // 최근 측정한 시각
unsigned long lastsent; // 최근 전송한 시각

void loop() {
  // 측정한지 2초가 지난 경우에 새로 센서로부터 측정치를 읽어들임
  unsigned long now = millis();
  // 현재 시간 - 이전의 측정 시간 > 2초
  if((now - lastmeas) > 2000) {
    // BME280 센서 읽기
    temp = bme.readTemperature();
    humi = bme.readHumidity();
    alti = bme.readAltitude(SEAPRESS);
    press = bme.readPressure() / 100.0f;
    lastmeas = now;
  }
  // 매 15초마다 Thingspeak API 를 이용해서 전송
}
```

# 7.3 Thingspeak : BME280 #3

- BME280 : thingspeak 서버에 데이터 전송 (매 15초마다...)

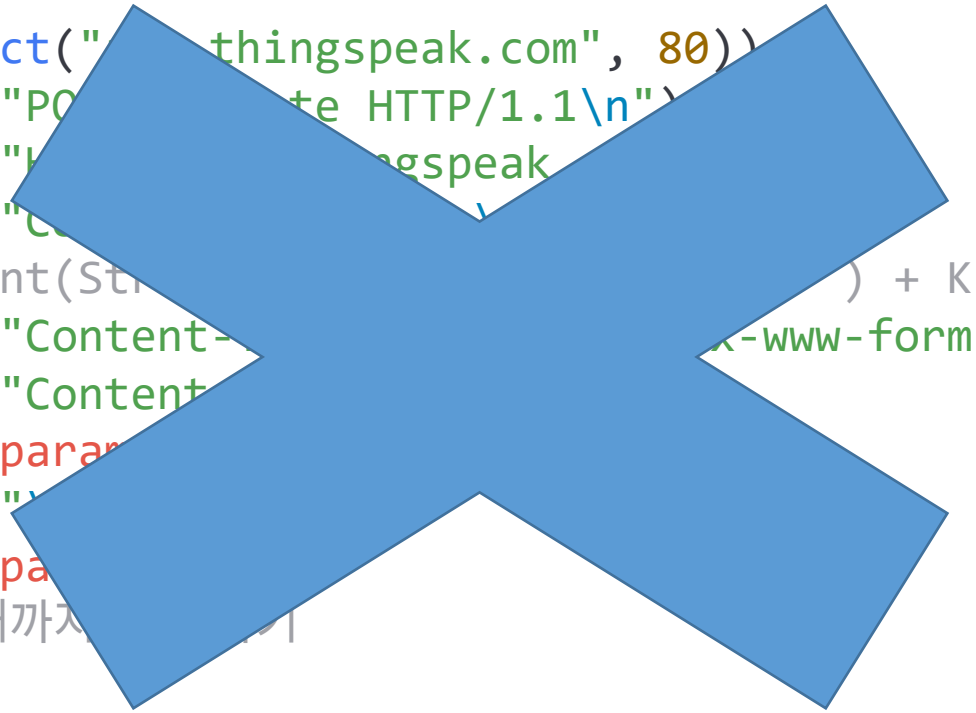
```
// 매 15초마다 Thingspeak API 를 이용해서 전송
now = millis();
if((now - lastsent) > 15000) {
  // thingspeak API 에 맞도록 데이터 필드
  char str[200], hs
  dtostrf(temp, 4,
  dtostrf(humi, 4, 1,
  sprintf(str, "api_key=%s
  String param = str;

  WiFiClient client
  // 서버에 접속
  if(client.connect(
    // 서버에 데이터 보
    // 응답이 올 때까지 대기하기
    // 서버로 부터 수신한 응답을 시리얼 모니터로 전송
  }
  lastsent = now;
}
```

float형 데이터를 문자열로
(4자리/소수점 1자리)

# 7.3 Thingspeak : BME280 #4

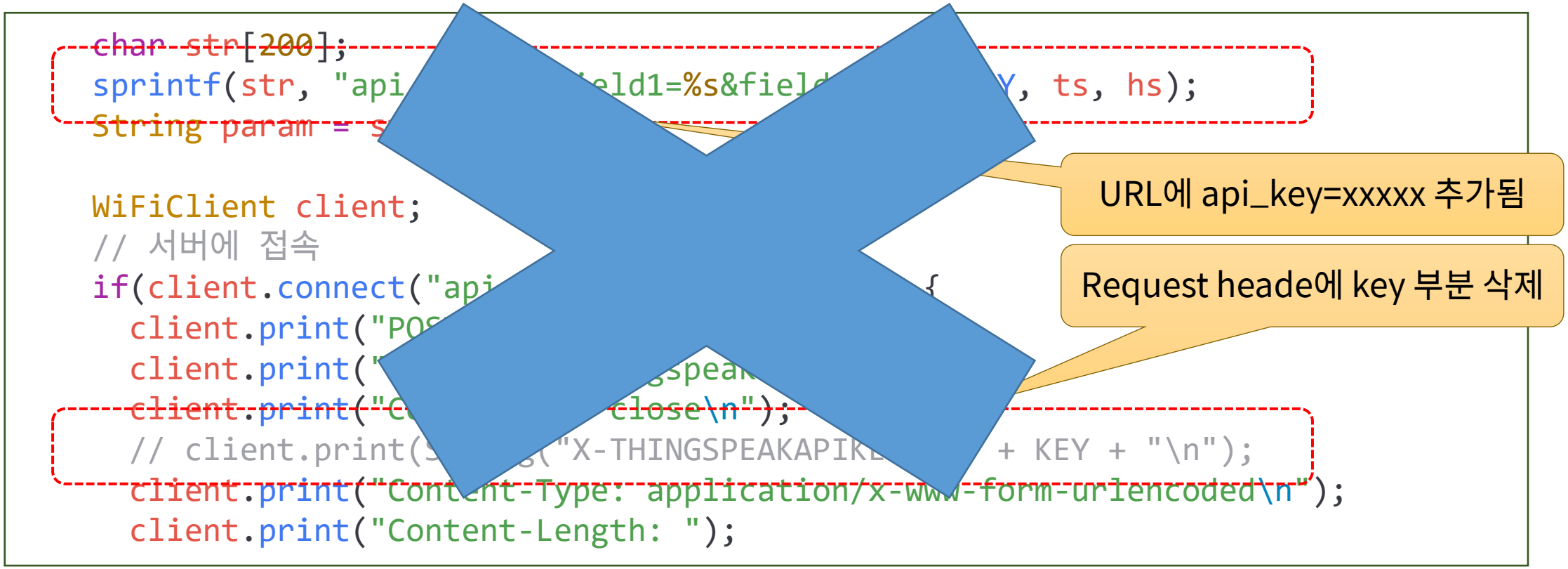
- BME280 : thingspeak 서버에 데이터 전송 (매 15초마다...)

```
WiFiClient client;
// 서버에 접속
if(client.connect("  thingspeak.com", 80))
  client.print("PO  te HTTP/1.1\n");
  client.print("H  gspeak
  client.print("C
  // client.print(St  ) + KEY + "\n");
  client.print("Content-  x-www-form-urlencoded\n");
  client.print("Content
  client.print(para
  client.print("
  client.print(pa
  // 응답이 올 때까지
  // 서버로 부터 수신한 응답을 시리얼 모니터로 전송
```



# 7.3 Thingspeak : BME280 #5

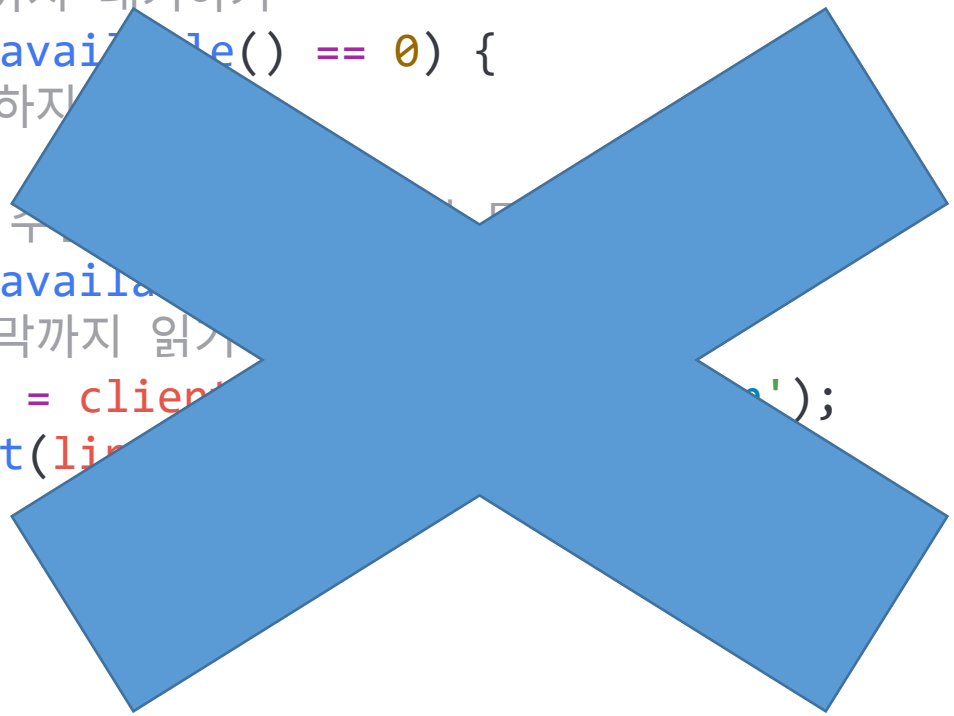
- BME280 : thingspeak 서버에 데이터 전송 (매 15초마다...)
  - 교재의 프로그램에서 달라진 점 V2 : 사용하지 못함



# 7.3 Thingspeak : BME280 #6

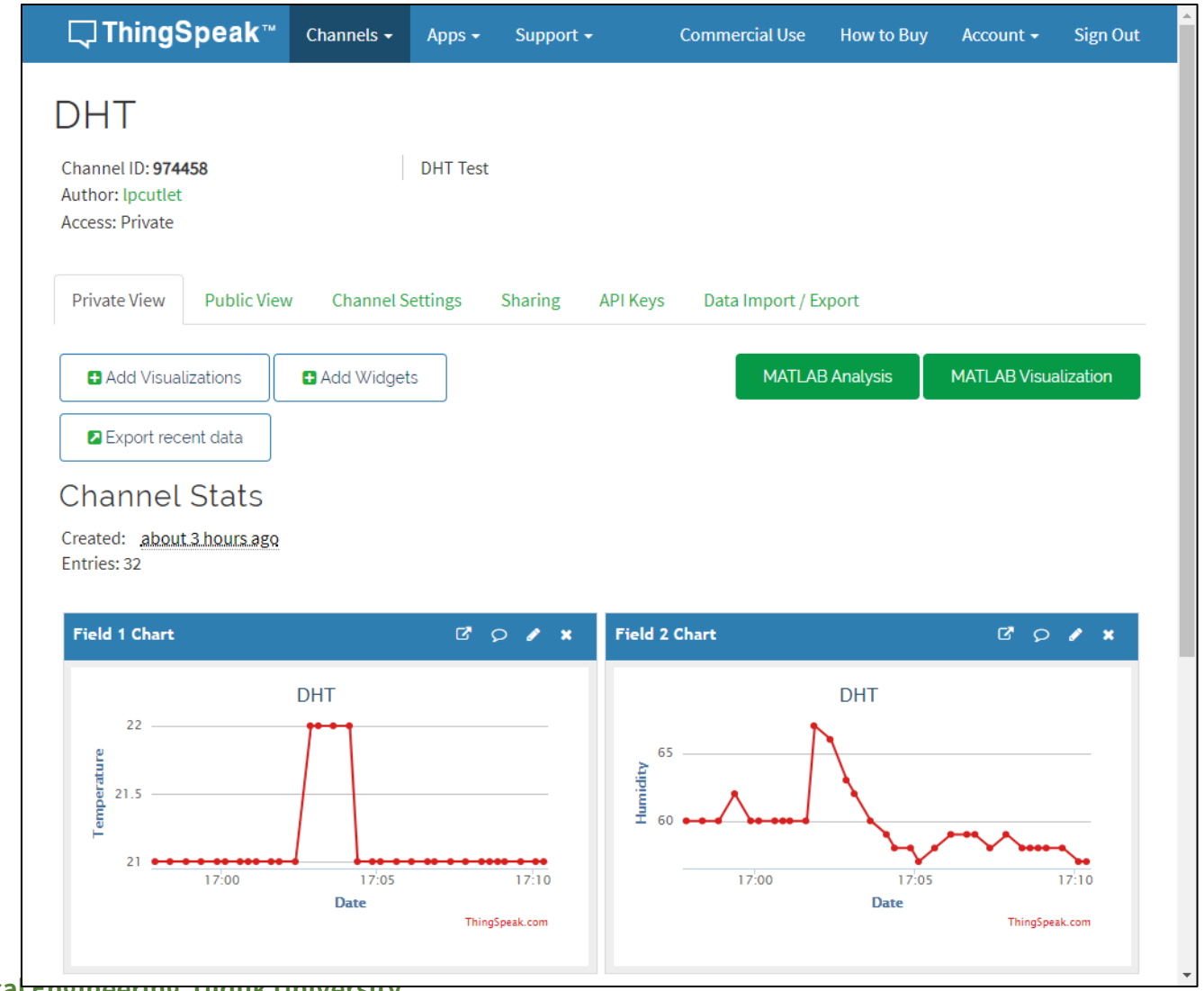
- BME280 : thingspeak 서버에 데이터 전송 (매 15초마다...)

```
      // 응답이 올때까지 대기하기
      while(client.avai   le() == 0) {
        // 아무것도 하지
      }
      // 서버로 부터 수
      while(client.availa
        // 줄의 마지막까지 읽기
        String line = clien                    n');
        Serial.print(li
      }
    }
    lastsent = now;
}
```



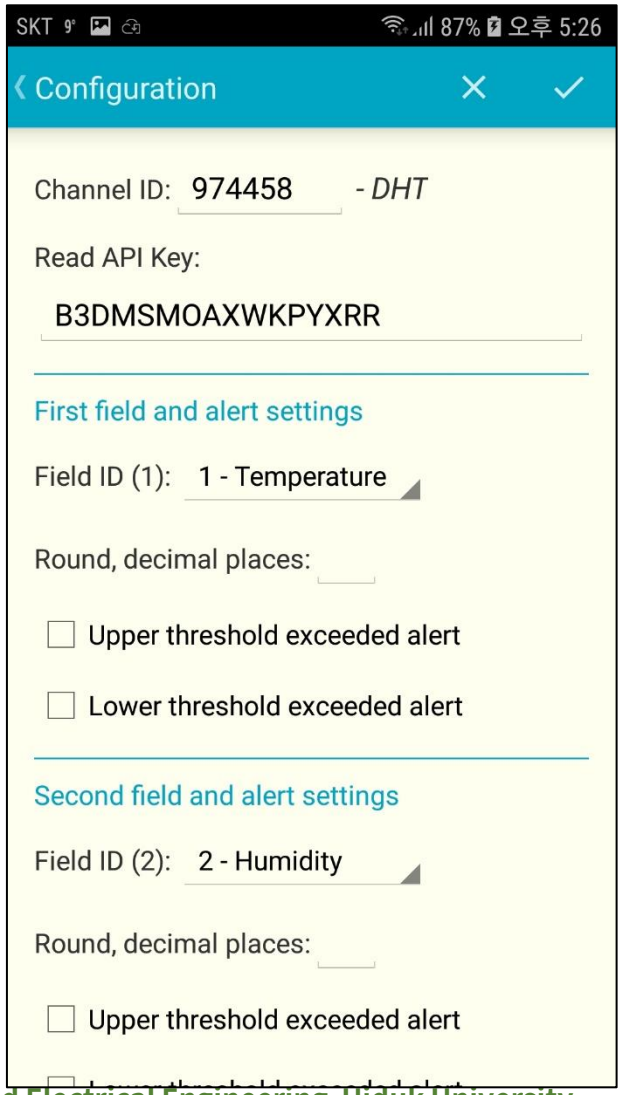
# 7.3 Thingspeak : BME280 #7

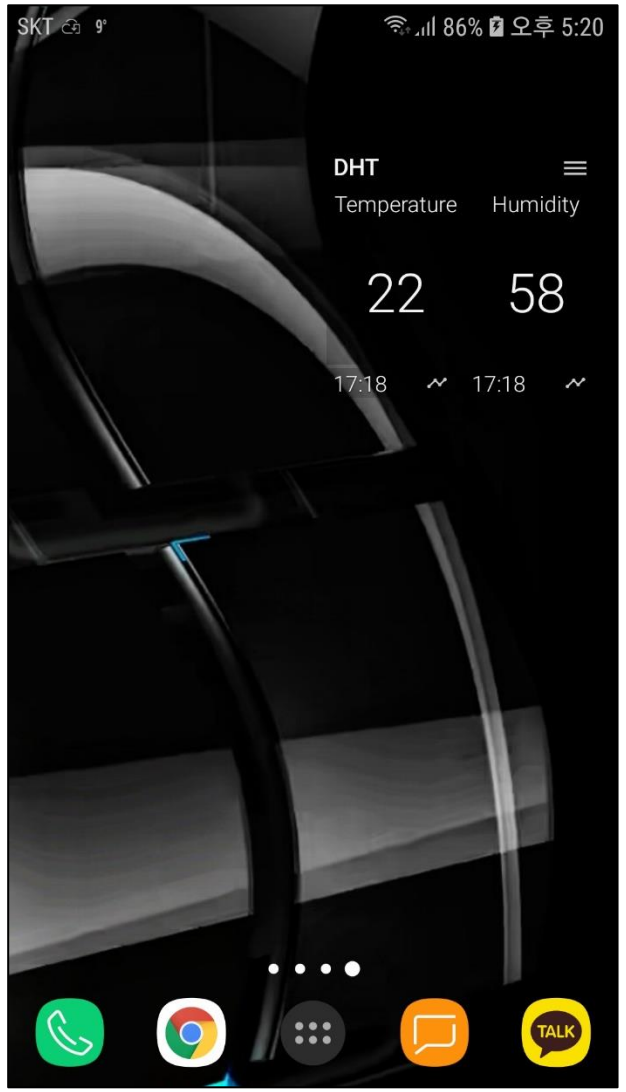
- 수행 결과 : thingspeak 홈페이지

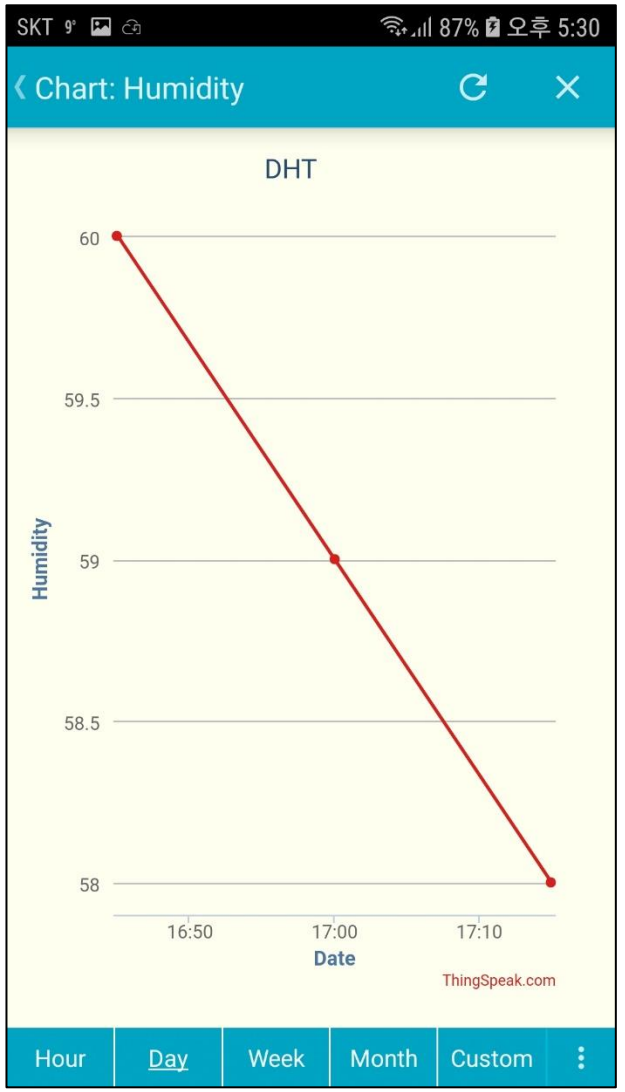


# 7.3 Thingspeak : BME280 #8

- 수행 결과 : 스마트폰 앱 (IoT ThingSpeak Monitor Widget)



# 추가 실습

- DHT11를 사용하여 thingspeak 사이트에 데이터를 전송하는 프로그램을 작성하시오.