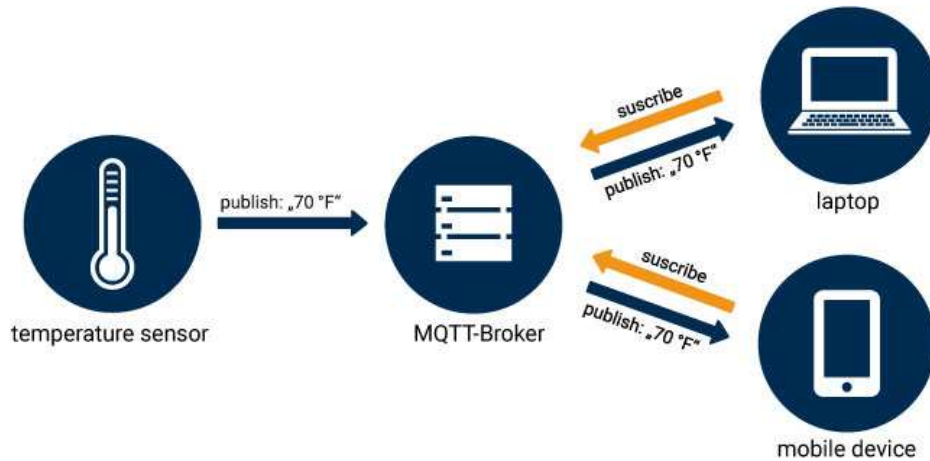


[MQTT : Message Queuing Telemetry Transport]

홍지우



[Message Queuing Telemetry Transport : 메시지 큐잉 텔레메트리 트랜스포트]

- ISO 표준 발행-구독 방식 경량 메시징 프로토콜
- 사물통신, 사물인터넷과 같이 대역폭이 제한된 통신 환경에 최적화하여 개발된 푸시 기술 기반의 경량 메시지 전송 프로토콜
- TCP/IP 프로토콜 위에서 사용 <- TCP/IP 프로토콜 스택에서 작동
- 모바일 기기나 낮은 대역폭의 소형 디바이스들에 최적화
- 느리고 품질이 낮은 네트워크에서도 메시지를 안정적으로 전송할 수 있도록 설계됨
- 저전력 설계
- 가장 작은 메시지는 2byte까지 가능
- 세 가지의 QoS 레벨 제공

MQTT 자체는 메시지를 어떻게 보낼 것인지를 정의하는 규약일 뿐이다. 따라서 실제 “MQTT”를 동작시키기 위해서는 서버 역할을 해주는 장치와 프로그램이 필요한데 이를 **MQTT 브로커(Broker)**라고 한다. **MQTT 브로커**는 각종 장치들(MQTT Client)이 보내주는 메시지를 수집하고 이걸 다시 필요한 장치들에게 전송해주는 중계서버 역할을 한다.

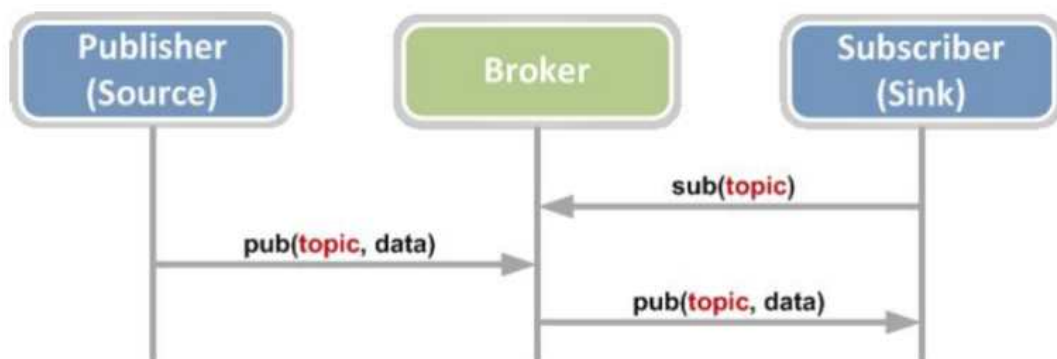


Figure 1: The publish/subscribe communication model

MQTT 시스템에 참여하는 MQTT 클라이언트는 메시지 발행 (publish, 트윗에 해당), 메시지 구독(subscribe, follow에 해당) 두 가지 동작을 할 수 있다. MQTT 클라이언트가 메시지를 특정 채널(Topic, 토픽)에 발행하면 이 채널을 구독한 모든 클라이언트에게 메시지가 전달된다. 중간에서 메시지를 수집, 재분배 하는 작업은 MQTT 브로커가 담당한다.

[토픽 (Topic)]



메시지를 발행/구독(pub/sub) 하는 행위는 채널 단위로 일어난다.

이를 MQTT에서는 토픽(Topic)이라고 한다. 토픽은 슬래시(/)로 구분된 계층구조를 갖는다.



메시지를 구독/발행 할 때 여러개의 토픽을 한번에 지정할 수 있도록 wild card 문자를 지원한다. [+] 문자는 단 1개의(1 레벨) 토픽을 임의의 토픽으로 대체하는 와일드카드 문자이다.



반면 [#] 문자는 여러 레벨의 토픽을 대체할 수 있다. [myhome/#] 처럼 사용하면 myhome 의 하위 토픽들 모두를 지칭하게 된다. [#] 문자는 토픽 문자열 끝에만 사용할 수 있으며 하위 토픽 모두를 지칭한다. (2단계 이상 하위 토픽도 포함)

[\$] 문자로 시작하는 토픽은 시스템에 의해 사용되는 특수한 토픽을 의미한다. 이 토픽들은 [#] 문자열로 지정해도 포함되지 않는 토픽이 된다. 주로 [\$SYS/] 로 시작하는 토픽들이 브로커의 내부 메시지를 위해 사용된다. (이에 대한 명확한 표준은 없는 상태인 듯)

[QoS (Quality of Service)]

MQTT는 시스템에 참여하는 장치들의 처리 능력, 네트워크 대역폭, 메시지 오버헤드 등 주변상황에 맞게 시스템이 동작할 수 있도록 3단계 QoS (Quality of Service) 를 제공한다.

- 0 : 메시지는 한번만 전달하며, 전달여부를 확인하지 않는다. Fire and Forget 타입이다.
- 1 : 메시지는 반드시 한번 이상 전달된다. 하지만 메시지의 핸드셰이킹 과정을 엄밀하게 추적하지 않기 때문에, 중복 전송될 수도 있다.
- 2 : 메시지는 한번만 전달된다. 메시지의 핸드셰이킹 과정을 추적한다. 높은 품질을 보장하지만 성능의 희생이 따른다.

0에 가까울수록 메시지 처리에 대한 부하가 적은 대신 메시지 손실 위험이 높아집니다. 2에 가까울수록 메시지 손실 위험은 줄어들지만 메시지 처리 부하가 급격히 늘어납니다.

(보통 0~1 정도의 QoS를 사용하면서 메시지 손실 등의 위험은 상위 어플리케이션 차원에서 관리하도록 하는 듯.)

[MQTT Arduino 구현]

[Mosquitto Broker 구현]

MQTT에서는 발행자와 구독자 사이에 브로커가 존재한다.

아두이노로 모스키토 통신을 구현하기 전에 서버와 같은 역할을 하는 **MQTT 브로커**를 먼저 만들어 줘야 한다. 그리고 MQTT 브로커는 (일정한 금액을 지불하고...) 클라우드 서버에서 역할을 수행하기도 하지만, 자체적으로 브로커를 사용할 때에는 **모스키토라는 MQTT 브로커**를 가장 많이 사용한다. 여러 브로커 프로그램들이 많이 개발 되었지만, 많은 사람들이 모스키토를 사용하는 이유는 무료이기 때문이다..

<https://mosquitto.org/download/>

지원하는 운영체제는 윈도우,리눅스,MAC이 있다. 컴퓨터 환경이 윈도우이기 때문에 구현은 윈도우에서 진행하였다. 윈도우에서는 모스키토 서버를 셋팅하는 작업이 필요하다.

[windows 모스키토 서버 셋팅]

1. 메모장 또는 워드패드 관리자 권한 실행 후 mosquitto.conf
 2. 변경: listener 1883 , allow_anonymous true;
 3. 윈도우 방화벽 -> 인바운드 규칙 1883 포트 추가.
 4. 윈도우 서비스 -> mosquitto -> 재시작
 5. cmd 관리자 실행 후 모스키토 경로에서
mosquitto -c mosquitto.conf -v
 6. 또 다른 cmd에서 netstat -an으로 내부 네트워크의 1883 포트 열렸는지 확인.
- MQTT에서 사용하는 기본 TCP 포트는 1883번이다.

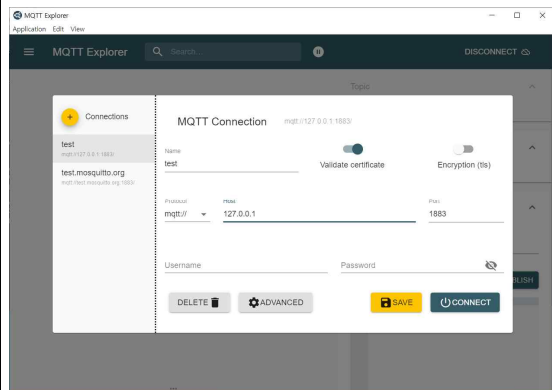
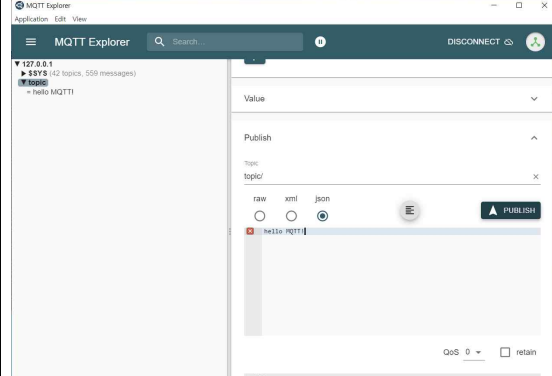
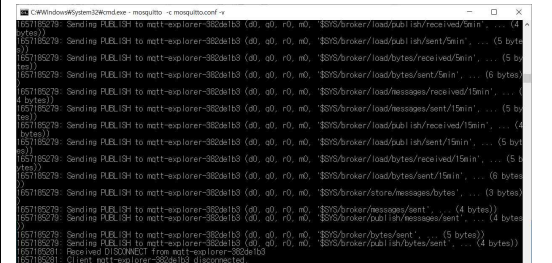
```
C:\Windows\System32\cmd.exe - mosquitto -c mosquitto.conf -v
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd ..
C:\Windows>cd ..
C:\>cd #Program Files#mosquitto
C:\#Program Files#mosquitto>mosquitto -c mosquitto.conf -v
1657184174: mosquitto version 2.0.14 starting
1657184174: Config loaded from mosquitto.conf.
1657184174: Opening ipv6 listen socket on port 1883.
1657184174: Opening ipv4 listen socket on port 1883.
1657184174: mosquitto version 2.0.14 running
```

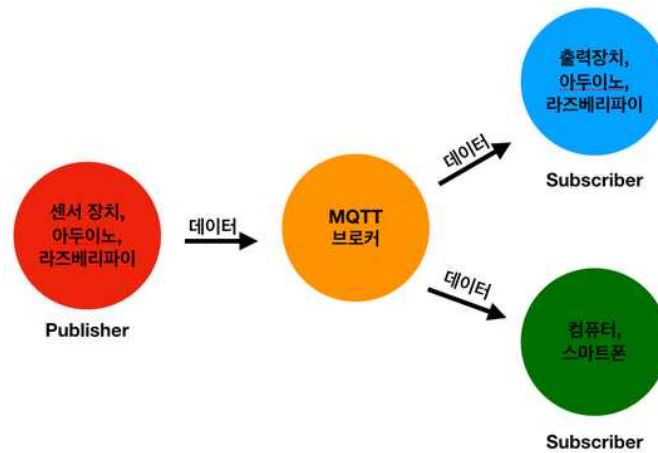
```
선택 명령 프롬프트
TCP 192.168.100.103:64754 13.107.42.254:443 TIME_WAIT
TCP 192.168.100.103:64755 13.107.213.49:443 TIME_WAIT
TCP 192.168.100.103:64756 13.107.21.200:443 TIME_WAIT
TCP 192.168.100.103:64757 13.107.213.49:443 TIME_WAIT
TCP 192.168.100.103:64758 13.107.4.254:443 TIME_WAIT
TCP 192.168.100.103:64761 204.79.197.200:443 ESTABLISHED
TCP 192.168.100.103:64762 13.107.18.254:443 ESTABLISHED
TCP 192.168.100.103:64763 117.18.232.200:443 ESTABLISHED
TCP 192.168.100.103:64764 13.107.4.254:443 ESTABLISHED
TCP 192.168.100.103:64765 204.79.197.222:443 ESTABLISHED
TCP [*]:1883 [*]:0 LISTENING
TCP [*]:445 [*]:0 LISTENING
TCP [*]:1883 [*]:0 LISTENING
TCP [*]:5357 [*]:0 LISTENING
TCP [*]:49664 [*]:0 LISTENING
TCP [*]:49665 [*]:0 LISTENING
TCP [*]:49666 [*]:0 LISTENING
TCP [*]:49667 [*]:0 LISTENING
TCP [*]:49668 [*]:0 LISTENING
TCP [*]:49669 [*]:0 LISTENING
UDP 0.0.0.0:3702 *:*
UDP 0.0.0.0:3702 *:*
UDP 0.0.0.0:3702 *:*
UDP 0.0.0.0:5050 *:*
UDP 0.0.0.0:5353 *:*
UDP 0.0.0.0:5353 *:*
UDP 0.0.0.0:5353 *:*
UDP 0.0.0.0:5353 *:*
UDP 0.0.0.0:5353 *:*
```

로컬 네트워크의 TCP 포트가 열린 것을 확인 하였다면, 브로커로 접속하여 발행과 구독이 가능하다. 내부네트워크에서만 열려있으므로 외부에서 이 브로커에 접속하려면 현재 로컬 IP 주소의 1883번 포트를 포트포워딩 해야한다.

[MQTT Explorer]

	<p>MQTT 브로커에 접속을 해야한다.</p> <p>127.0.0.1은 현재 로컬 네트워크를 의미하는 주소이다. 포트는 1883으로 한다.</p> <p>MQTT에서 사용하는 포트는 기본적으로 1883번 포트로 맞춰져 있다.</p>
	<p>접속하면 토픽을 구독하고 발행할 수 있다.</p> <p>해당 사진은 토픽을 직접 하나 발행하고 hello MQTT!라는 메시지를 보내었다.</p>
	<p>해당 MQTT 브로커를 확인해 보면 발행 내역을 확인해 볼 수 있다.</p>

[Arduino 구현]



ESP는 MQTT 브로커에 온습도 값을 계속해서 **발행(Publisher)** 한다.

따라서 어떠한 장치이든 MQTT 브로커에 접속하여 온습도 토픽을 **구독(Subscriber)**하면 ESP에서 발행한 메시지를 확인 할 수 있다.

ESP:/hum, /tem 토픽에 센서 값 발행 -> 브로커 <- 디바이스:/hum, /tem 토픽에 구독.

```

10:17:18.933 -> Connecting to WiFi
10:17:19.964 -> ....
10:17:22.966 -> Connected to the WiFi network
10:17:22.966 -> Connecting to MQTT...
10:17:23.200 -> connected
10:17:26.252 -> Publish message:
10:17:26.252 -> Temperature : 26.70°C
10:17:26.252 -> Humidity : 52.00%
10:17:27.236 -> Publish message:
10:17:27.236 -> Temperature : 26.70°C
10:17:27.236 -> Humidity : 52.00%
10:17:28.268 -> Publish message:
10:17:28.268 -> Temperature : 26.70°C
10:17:28.268 -> Humidity : 52.00%
10:17:29.253 -> Publish message:
10:17:29.300 -> Temperature : 26.70°C
          
```

```

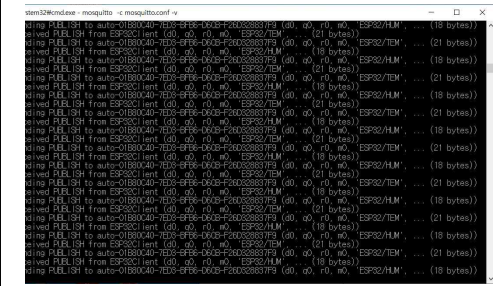
09:41:03.923 -> Publish message:
09:41:03.970 -> Temperature : 26.90°C
09:41:03.970 -> Humidity : 51.00%
09:41:04.954 -> Publish message:
09:41:04.954 -> Temperature : 26.90°C
09:41:04.954 -> Humidity : 51.00%
09:41:05.986 -> Publish message:
09:41:05.986 -> Temperature : 26.90°C
09:41:05.986 -> Humidity : 51.00%
09:41:07.018 -> Publish message:
09:41:07.018 -> Temperature : 26.90°C
09:41:07.018 -> Humidity : 51.00%
09:41:08.002 -> Publish message:
09:41:08.002 -> Temperature : 26.90°C
09:41:08.002 -> Humidity : 51.00%
          
```

1. 아두이노 : WiFi 연결 후, MQTT 브로커에 접속한다. MQTT 브로커에 접속 되면, 해당 브로커로 발행 메시지를 2초마다 보낸다. Publish message 메시지는 ESP32/TEM 토픽에 온도값. ESP32/HUM 토픽에 습도 값 메시지를 보낸다.

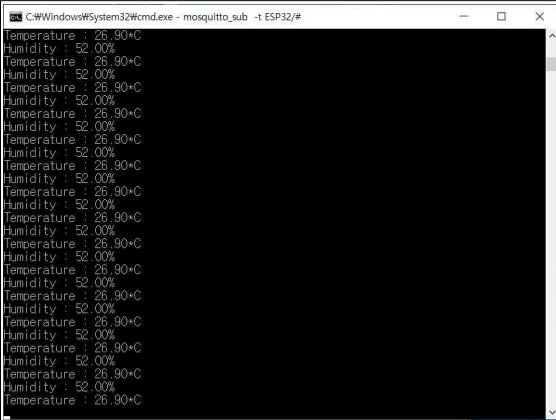
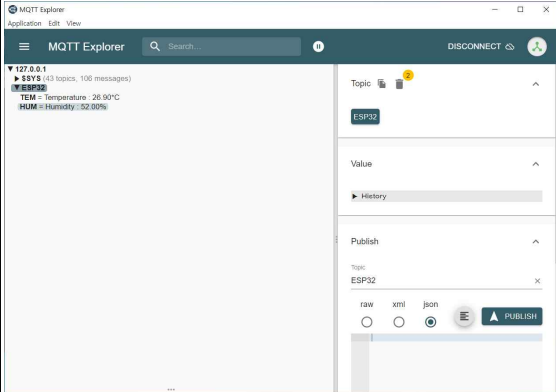
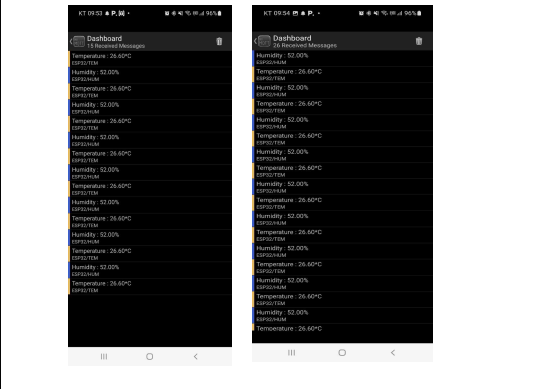
* MQTT Broker 접속 환경, 즉 내부 네트워크로 접속해야한다.

외부 네트워크에서 현재 Broker에 접속하려면 해당 IP 주소의 1883번 포트를 공유기에서 포트포워딩 해서 접속 해야한다.

2. 모스키토 브로커 확인

	<p>브로커의 상태를 보면 ESP32/TEM 토픽과 ESP32/HUM 토픽에 계속해서 발행 메시지가 오는 것을 확인 할 수 있다.</p> <p>이제 어떤 장치든 이 브로커에 접속하여 토픽을 구독하게 되면 메시지를 볼 수 있다.</p>
---	--

3. 브로커에 접속하고 구독(Subscrib)하여 메시지 확인

	<p>Mosquitto_sub 프로그램을 활용하여 ESP32/#으로 구독하였다.</p> <p>여기서 #은 ESP32의 토픽에 해당하는 메시지를 전부 포함하겠다는 뜻이다. (ESP32/의 하위토픽을 전부 지칭한다.) 즉, ESP32/TEM과 ESP32/HUM을 전부 지칭한다.</p>
	<p>MQTT Explorer로 MQTT 브로커에 접속하여 ESP32 토픽의 TEM 토픽과 HUM 토픽에 해당하는 메시지를 확인 할 수있다.</p>
	<p>MQTT 클라이언트 어플로 스마트폰에서도 브로커에 접속하여 구독이 가능하다.</p> <p>브로커에 접속하여 ESP32/#을 구독하여 ESP32/# 토픽에 해당하는 메시지를 전부 확인 할 수 있다.</p>

4. 패킷 검증

<

와이어샹크로 192.168.103 (브로커IP) <-> 192.168.100.101 (ESP)와 주고받은 패킷을 확인하였다. MQTT는 TCP/IP 스택위에서 동작하므로 TCP 패킷과 MQTT 패킷을 확인 할 수 있다. 그러나 두 번째 사진을 보면 유실되는 패킷들도 많은 편이다. MQTT 프로토콜 자체가 메시지를 위한 간단한 프로토콜이므로 전송제어에 조금 민감한 편이라고 생각한다. 또한 QoS는 0을 사용했으므로 메시지가 유실되어도, 재전송을 하지 않는다.