7장. 메시지 통신과 MQTT

강의 목표

1. IoT의 표준 통신 기법 MQTT를 이해한다.
2. 라즈베리파이에서 MQTT를 활용하는 방법을 안다.
3. mosquitto 브로커를 활용한다.
4. paho mqtt 파이썬 라이브러리를 활용하여 MQTT 클라이언트들을 작성할 수 있다.
5. Windows와 라즈베리파이 모두에 paho mqtt를 다룰 수 있다.
6. MQTT를 활용하여 라즈베리파이에서 보낸 메시지를 Window에서 수신할 수 있다.

들어가기 전에

- 예제와 실습은 myenv 가상 환경에서 실행
  - 파이썬 코드 실행 전 반드시 가상 환경 활성화

```
pi@pi:~ $ source myenv/bin/activate
(myenv) pi@pi:~ $
```

- 예제 파이썬 코드는 ch07 디렉터리에 저장하고 실행

```
(myenv) pi@pi:~ $ mkdir ch07
(myenv) pi@pi:~ $ cd ch07
(myenv) pi@pi:~/ch07 $
```

- MQTT 프로그램 작성에는 paho.mqtt 버전2 사용
  - Client() 생성자와 on_connect()의 매개 변수에 유의

7.1 MQTT를 이용한 메시지 통신

MQTT 통신 개요

- MQTT
  - Message Queuing Telemetry Transport
  - 1999년 IBM에 의해 개발 메시지 기반의 통신 프로토톨
    - subscribe/publish 방식으로 메시지 전송
    - subscribe – 메시지를 받고자 **토픽**을 등록하는 행위
    - publish – **토픽**과 함께 메시지를 보내는 행위
- 활용
  - 센서 수집 및 제어, 자율자동차의 센서 수집 및 제어
  - 메시지 채팅, 페이스북은 MQTT 기반

6

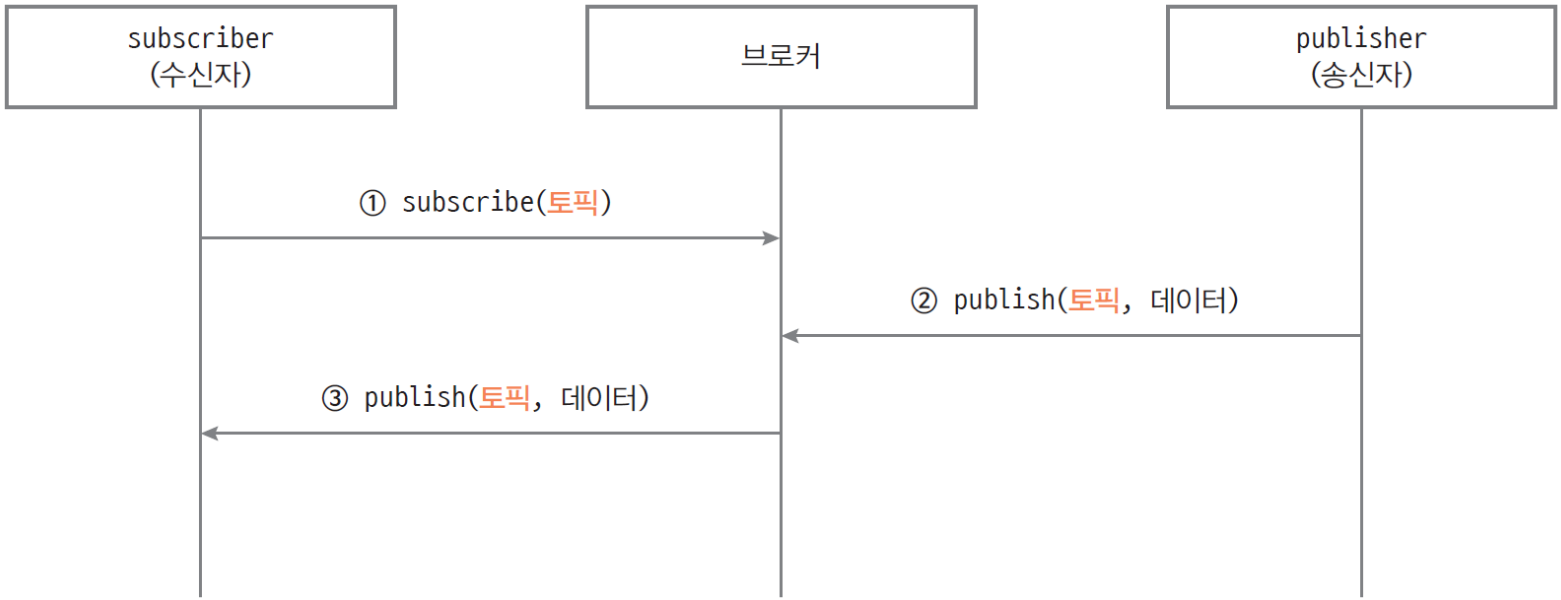
7.2 MQTT 통신의 구성 요소와 통신 과정

MQTT 통신의 구성 요소

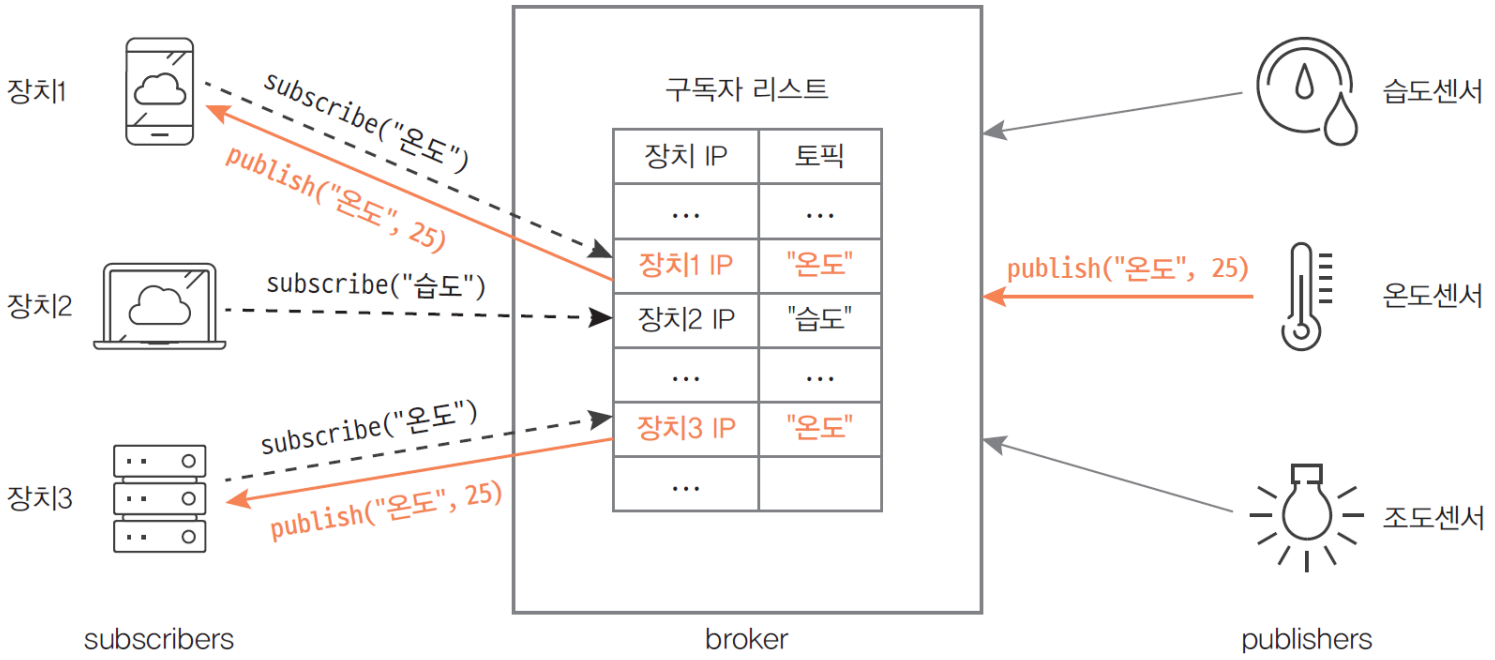
- MQTT 통신의 구성 요소
  - 메시지 송신자 - publisher 클라이언트
    - 토픽과 메시지를 브로커에게 전송
  - 메시지 수신자 - subscriber 클라이언트
    - 토픽을 브로커에 등록
    - 브로커로부터 대기하는 토픽의 메시지 수신
  - 메시지 중계자 - 브로커(MQTT Broker)
    - 토픽으로 등록한 subscriber 들 기억
    - 토픽과 메시지가 도착하면 subscribers들에게 메시지 전송

- Subscriber나 Publisher들은 직접 연결되지 않음
  - 브로커와 연결
  - 브로커를 경유하여 메시지 송수신

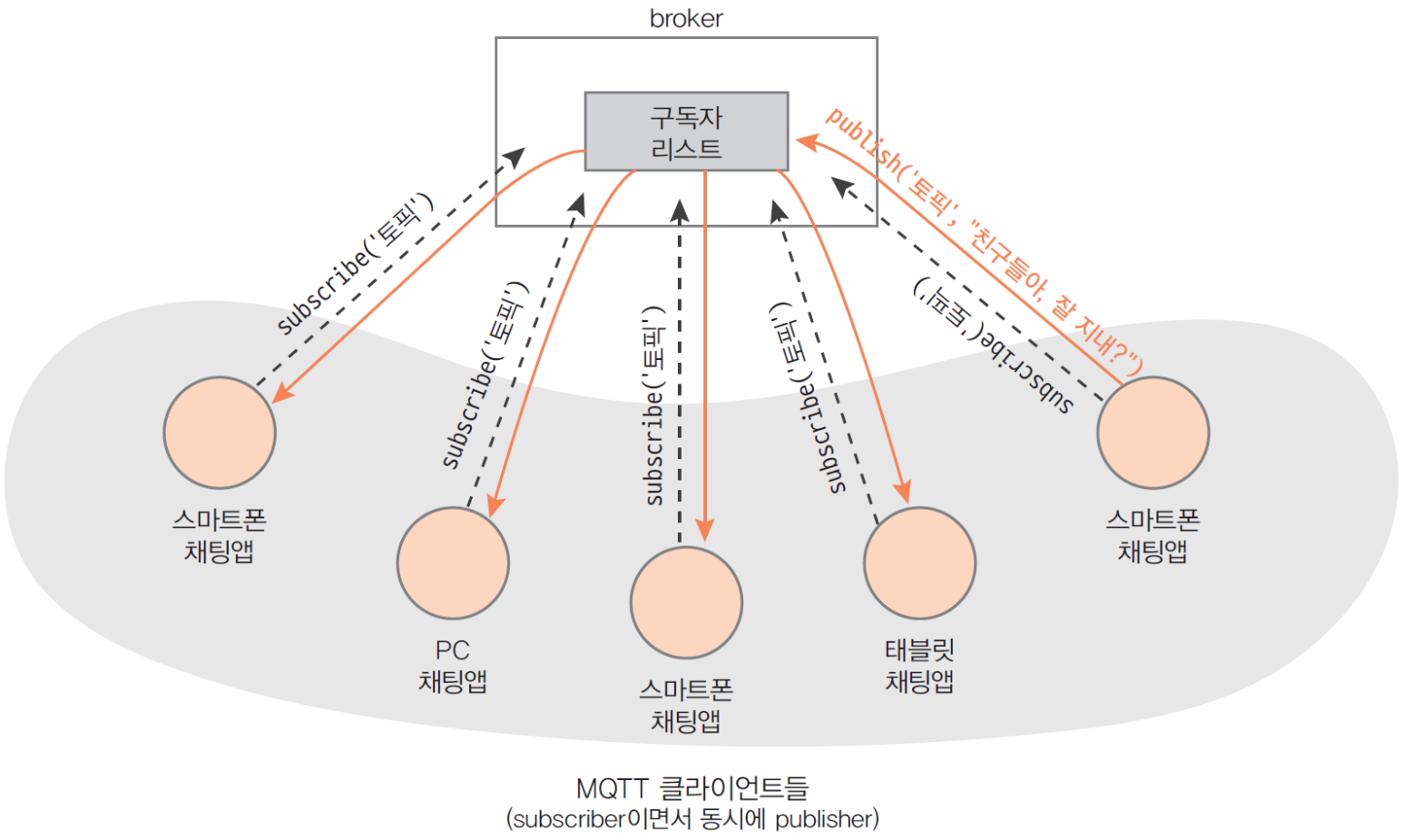
MQTT 통신 시스템 구성과 통신 과정



MQTT 응용 시스템 사례1 - 센서 데이터 수집



MQTT 응용 시스템 사례 2 - 채팅 앱



MQTT 통신 특징

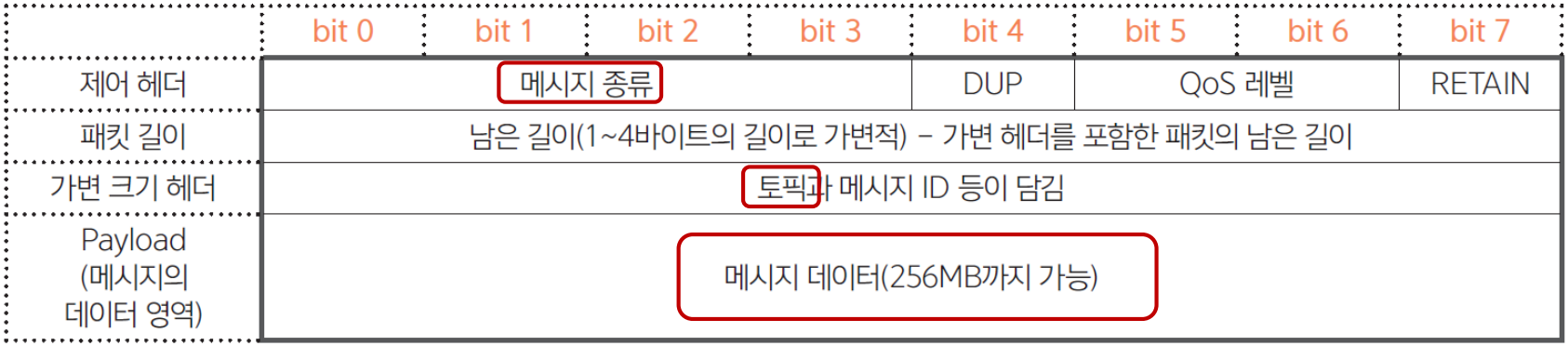
1. subscriber와 publisher들이 직접 연결되지 않음
   - 브로커에 의해 메시지 전송 중계
2. subscriber와 publisher들의 구성과 응용
   - 1:1 혹은 1:N, N:1, N:M으로 메시지를 주고받는 응용 시스템에 적합
3. 하나의 subscriber는 여러 개의 '토픽' 구독 가능
   - 하나의 publisher는 같은 토픽의 메시지를 여러 개 발행 가능
   - 하나의 publisher는 서로 다른 토픽의 메시지 발행 가능
4. 클라이언트는 subscriber와 publisher의 두 기능 모두 구현 가능
   - 클라이언트의 구현에 달려 있음
5. 메시지의 송수신은 토픽을 기준
6. MQTT는 TCP/IP를 활용하는 통신 프로토콜
   - publisher, subscriber, 브로커 모두 IP 주소를 가진 장치에서 실행
7. MQTT는 불안정한 상태의 열악한 네트워크 환경에서도 잘 작동

MQTT 활용 사례

- 원격 센서 데이터 수집 및 제어
- 스마트 홈과 웨어러블 장치
- 자동차
- SNS 모바일 애플리케이션

MQTT 패킷

- MQTT 패킷
  - 메시지 종류 16개
    - 제어 헤더의 메시지 종류 필드에 표기
  - 메시지 본체 - Payload 부분에 적재
    - 메시지 크기 256MB까지 가능
  - 토픽 문자열 65536 바이트까지



MQTT 메시지 종류 16가지

메시지 종류	번호	풀어쓰기	설명
Reserved	0		예약됨
CONNECT	1	connect	연결
CONNACK	2	connect ack	연결에 대한 응답
PUBLISH	3	publish	메시지 발행
PUBACK	4	publish ack	발행에 대한 응답
PUBREC	5	publish received	발행된 메시지 수신하였음
PUBREL	6	publish release	발행된 메시지 저장 해지 지시
PUBCOMP	7	publish complete	발행된 메시지를 구독자에게 전달 완료
SUBSCRIBE	8	subscribe	구독 신청
SUBACK	9	subscribe ack	구독에 대한 응답
UNSUBSCRIBE	10	unsubscribe	구독 취소
UNSUBACK	11	unsubscribe ack	구독 취소에 대한 응답
PINGREQ	12	ping request	PING 요청
PINGRESP	13	ping response	PING 응답
DISCONNECT	14	disconnect	연결 끊기
Reserved	15		예약됨

QoS

- QoS: Ouality of Service
  - publisher와 MQTT 브로커, MQTT와 subscriber 사이의 메시지 전송 품질

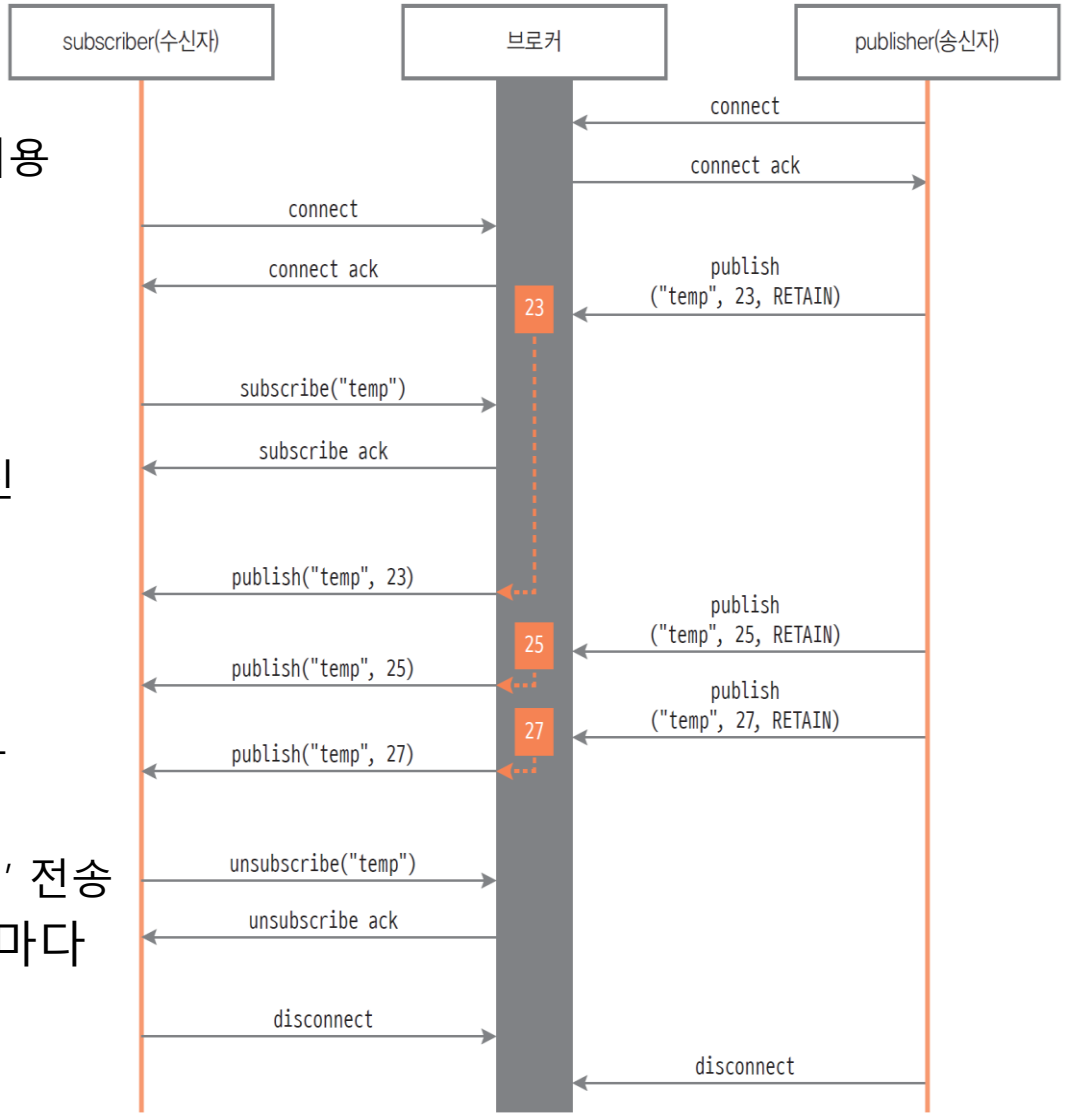
- 3가지 레벨
  - Level 0 (At most once)
    - 가장 낮은 품질
    - 메시지가 전달되었는지 확인하지 않는 낮은 수준
    - 메시지는 최대 한번 전달 – 전달되지 않을 수도 있음
  - Level 1 (At least once)
    - 메시지가 전달되었음을 확인하는 보통 수준
    - 일정시간 내에 보낸 메시지에 대한 수신 응답이 오지 않으면 메시지 다시 전송
      - 메시지가 여러 번 전송될 수 있음
  - Level 2 (Exactly once)
    - 가장 높은 수준의 전송
    - 메시지를 발행할 쪽에서 수신자에게 메시지가 전달되었음을 확인하는 절차 있음
    - 메시지는 반드시 한 번 전달

토픽

- 토픽
  - Publisher와 Subscriber는 토픽을 기준으로 메시지 구독/발행
  - 토픽은 문자열
    - 어떤 문자열도 가능
- 통신 시스템을 설계하는 사람이 마음대로 토픽 결정
  - subscriber와 publisher가 상호 알고 있어야 함
- 토픽 사례
  - meeting, newspaper, city 등
  - / 를 이용하여 토픽 계층화 가능
    - 예) "room/1/temperature", "room/2/temperature"
- 토픽의 중요성
  - 다수의 센서 기기들을 관리하는데 토픽은 매우 효과적

MQTT 프로토콜 사례

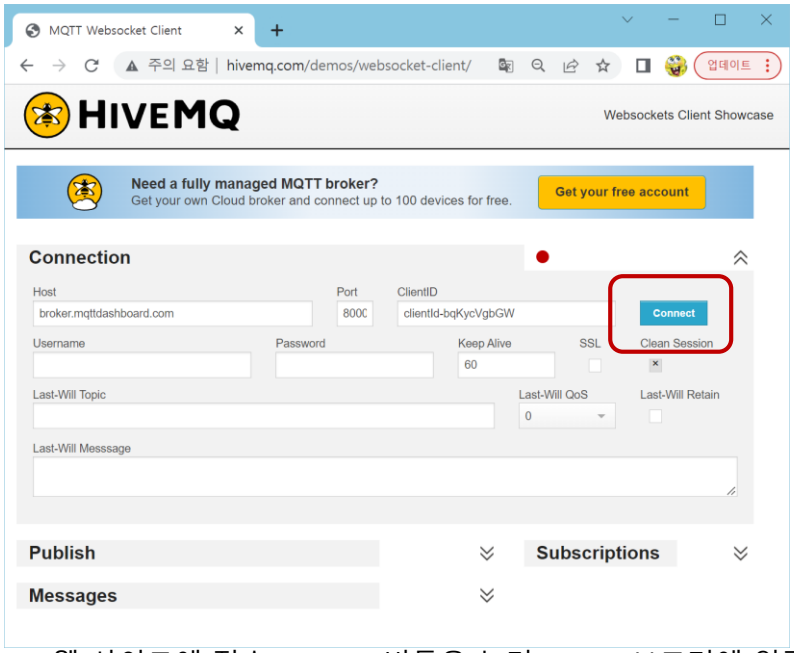
- subscriber
  - 브로커에 연결(connect)
    - 브로커의 IP, 포트 1883 이용
  - 구독 신청(subscribe)
    - 브로커에게 '토픽' 보내기
  - 메시지 수신 대기
  - 메시지 수신
    - 브로커로부터 메시지 수신
  - 수신 대기와 수신의 반복
- publisher
  - 브로커에 연결(connect)
    - 브로커 IP, 포트 1883 이용
  - 메시지 발행(publish)
    - 브로커에 '토픽'과 '메시지' 전송
  - 메시지를 보내고 싶을 때마다
    - 메시지 발행



MQTT 메시지 통신 실습

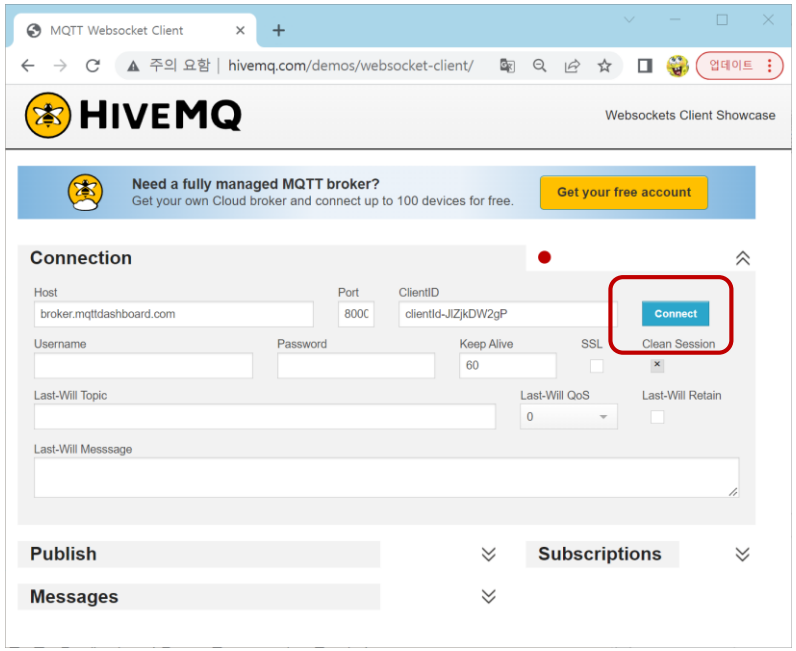
- HiveMQ의 데모 사이트 활용
  - 다음 사이트에 접속
  - http://www.hivemq.com/demos/websocket-client/
- 2개의 웹 브라우저를 열고
    - subscriber 1개
    - publisher 1개
  - 각각 HiveMQ의 데모 사이트에 접속

클라이언트 P(Publisher)

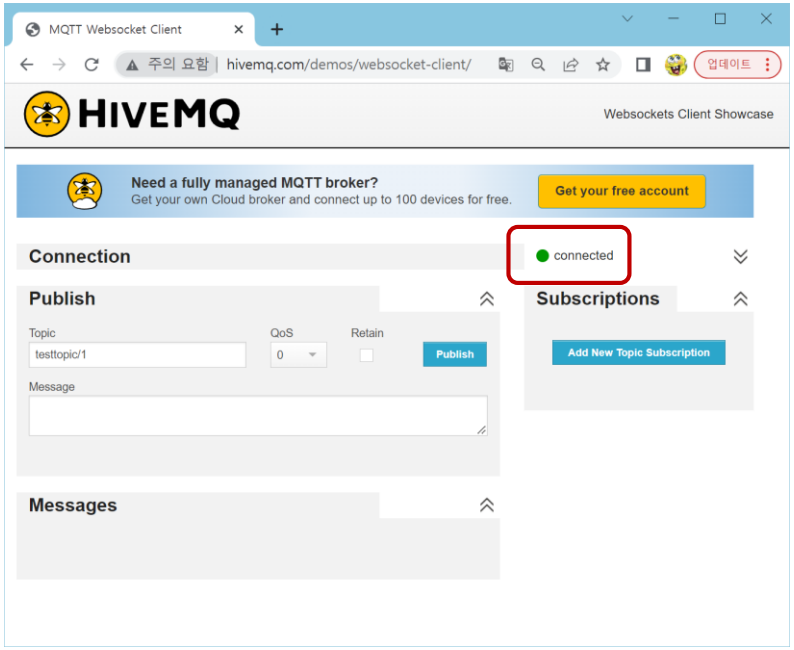


(a) 웹 사이트에 접속Connect 버튼을 눌러 MQTT 브로커에 연결

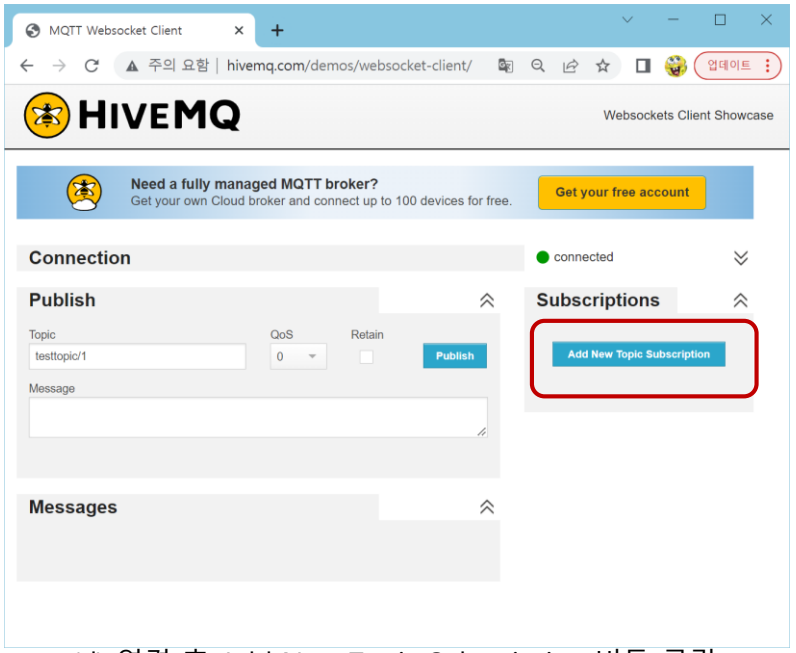
클라이언트 S(Subscriber)



(b) 웹 사이트에 접속Connect 버튼을 눌러 MQTT 브로커에 연결

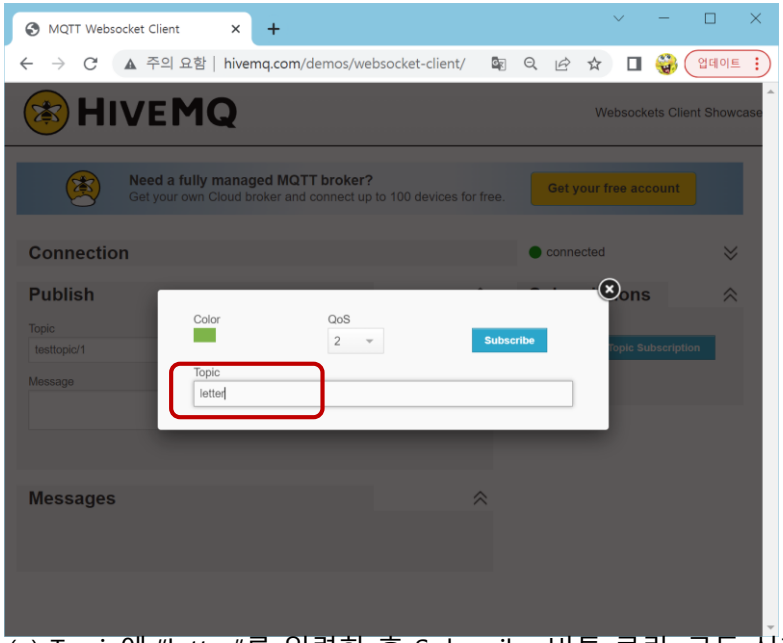


(c) 연결 완료

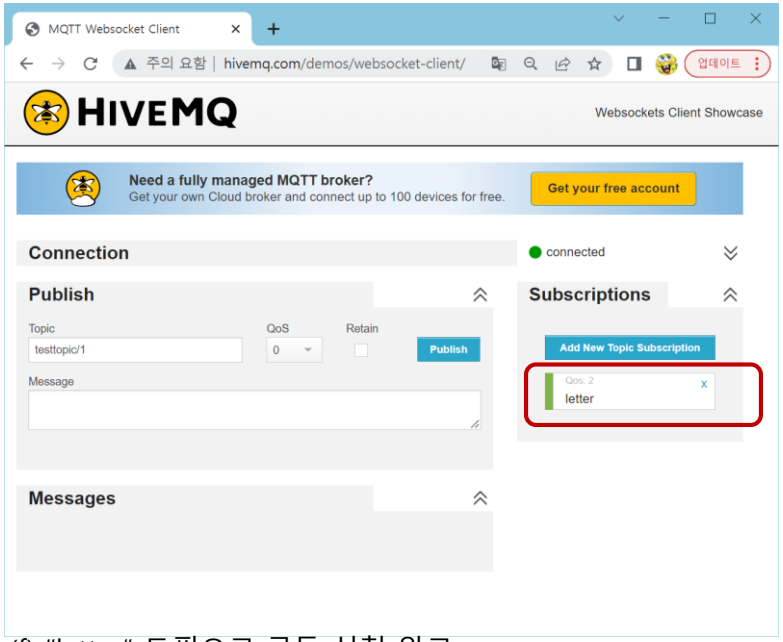


(d) 연결 후 Add New Topic Subscription 버튼 클릭

클라이언트 S

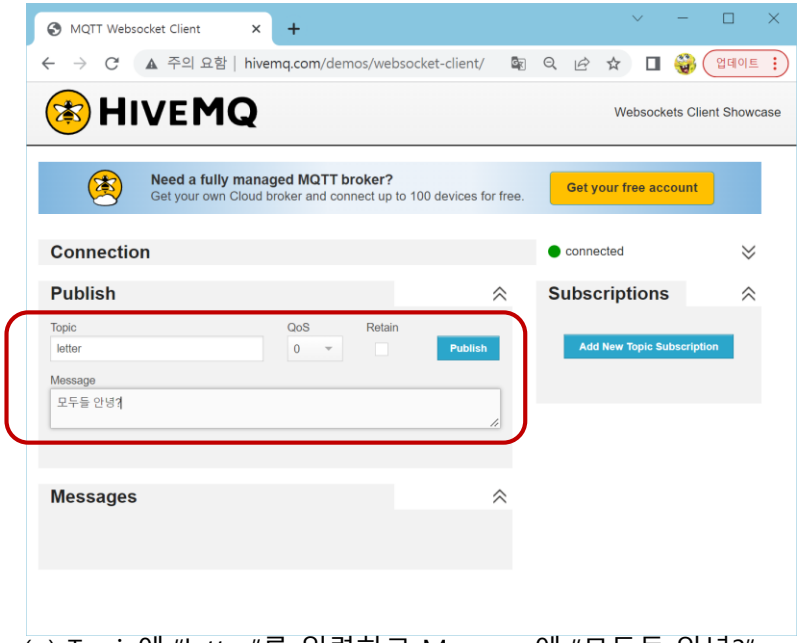


(e) Topic에 "letter"를 입력한 후 Subscribe 버튼 클릭. 구독 신청



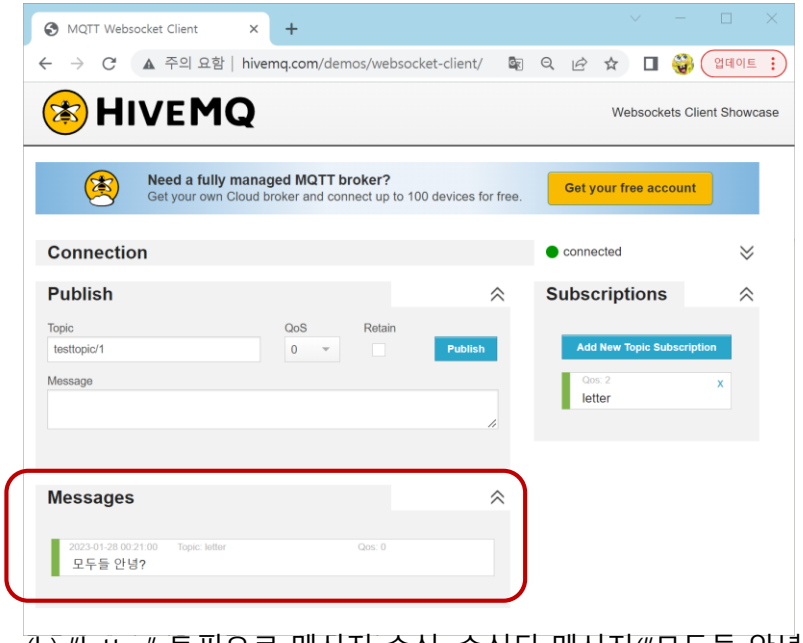
(f) "letter" 토픽으로 구독 신청 완료

클라이언트 P

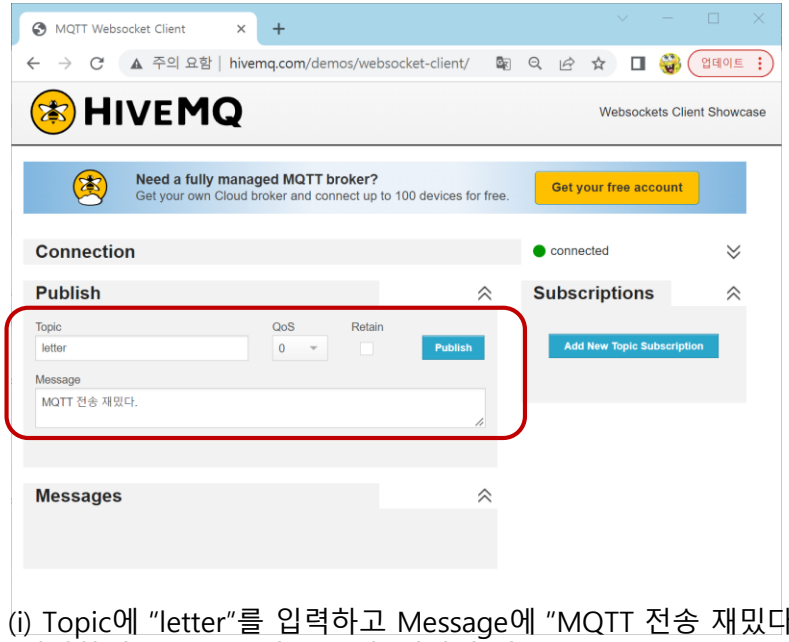


(g) Topic에 "letter"를 입력하고 Message에 "모두들 안녕?" 입력한 후, Publish 버튼 클릭. 메시지 전송

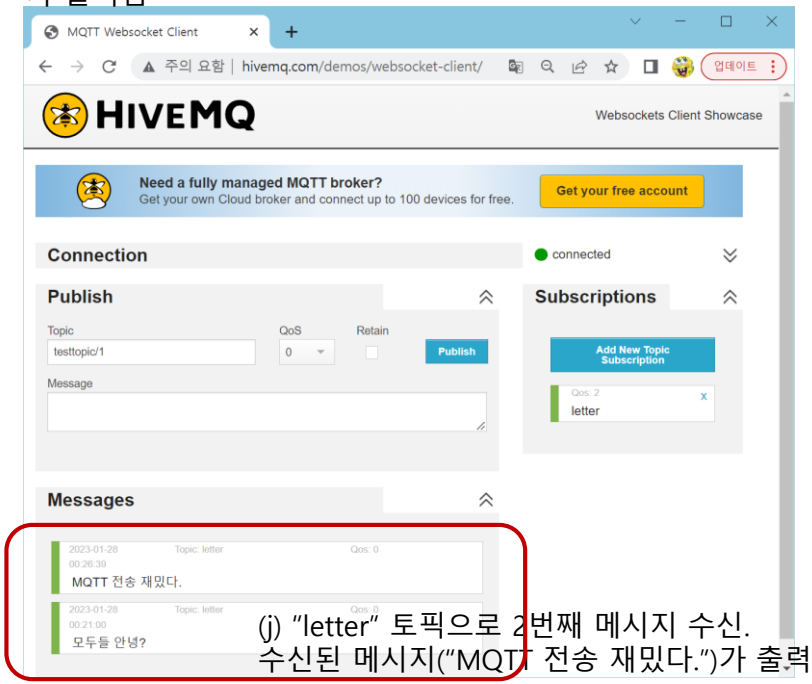
클라이언트 S



(h) "letter" 토픽으로 메시지 수신. 수신된 메시지("모두들 안녕?")가 출력됨



(i) Topic에 "letter"를 입력하고 Message에 "MQTT 전송 재밌다."를 입력한 후, Publish 버튼 클릭. 메시지 전송



(j) "letter" 토픽으로 2번째 메시지 수신. 수신된 메시지("MQTT 전송 재밌다.")가 출력됨

7.3 라즈베리파이에 MQTT 활용

MQTT 응용시스템 개발과 Mosquitto

- MQTT 브로커
  - MQTT는 IBM에 의해 개발된 통신 규약(Protocol)
  - 2014년 국제 민간 표준 기구인 오아시스에 의해서 표준으로 제정
  - MQTT 표준을 구현한 다양한 MQTT Brocker
    - HiveMQ, IbmMQ, RabbitMQ, vert.X, Mosquitto
  - 공개 브로커들
    - test.mosquitto.org, broker.hivemq.com, iot.eclipse.org
  - MQTT 응용시스템 개발자는 MQTT 브로커를 개발할 필요없음
    - MQTT 브로커 – 공개 소프트웨어 사용하면 됨
- Mosquitto
  - Eclipse foundation에서 개발한 오픈 소스 MQTT 브로커
  - 다양한 운영체제에서 실행
    - Linux, Mac, Windows

Mosquitto 설치

- mosquitto 설치
  - **$ sudo apt install -y mosquitto mosquitto-clients**
    - mosquito 브로커와 함께 2 개의 클라이언트 프로그램 설치
      - *mosquitto_sub*와 *mosquitto_pub* 도 함께 설치
  - 설치 후 자동으로 mosquito 브로커가 서비스로 실행됨
    - 확인(아래에서 현재 2180번 프로세스가 실행 중인 mosquito 임)

```
pi@pi:~ $ ps -ef | grep mos
mosquit+   2180      1   0 13:12 ?        00:00:00 /usr/sbin/mosquitto -c
/etc/mosquitto/mosquitto.conf
pi         2298   1318   0 13:14 pts/0    00:00:00 grep --color=auto mos
pi@pi:~ $
```

- mosquitto 설치 후 서비스 등록 해제
  - mosquitto 서비스 해제. 부팅 후 자동으로 실행되지 못하게 함
    - **$ sudo systemctl disable mosquitto**
  - 현재 실행 중인 mosquitto 서비스 실행 중단
    - **$ sudo systemctl stop mosquitto**
  - 이유
    - 필요할 때만 실행시키기 위해
  - 서비스가 중단되었는지 확인

```
pi@pi:~ $ ps -ef | grep mos
pi         2298   1318   0 13:14 pts/0    00:00:00 grep --color=auto mos
pi@pi:~ $
```

홈 디렉터리에 mos.conf 작성

- 사용자 홈 디렉터리(pi 디렉터리)에 mos.conf 작성

```
listener 1883                  # 외부 컴퓨터에서 1883 포트로 접속 허용
allow_anonymous true           # 어떤 사용자에게도 접속 허용
```

  - 목적
    - 디폴트로 설치된 /etc/mosquitto/mosquitto.conf 파일을 사용하지 않고, 사용자가 만든 설치 정보를 사용하기 위해

mosquitto 실행

- mosquitto 실행(mos.conf가 있는 디렉터리에서)
  - **$ mosquitto –v –c mos.conf**

```
pi@pi:~ $ mosquitto –v –c mos.conf
1664768323: mosquitto version 2.0.11 starting
1664768323: Config loaded from mos.conf.
1664768323: Opening ipv4 listen socket on port 1883.
1664768323: Opening ipv6 listen socket on port 1883.
1664768323: mosquitto version 2.0.11 running
```

    - -v 옵션
      - *mosquitto가 실행되는 동안 발생하는 상황들을 화면에 출력하도록 지시*
    - -c mos.conf 옵션
      - *mos.conf 파일에 설정 정보 있음*
- 이미 mosquitto가 실행되고 있는 경우, mosquitto를 실행시키면
  - 다음 오류 발생 - 이미 실행되고 있기 때문에 두 번 실행하면 안 됨

```
pi@pi:~ $ mosquitto –v –c mos.conf
1664768402: mosquitto version 2.0.11 starting
1664768402: Config loaded from mos.conf.
1664768402: Opening ipv4 listen socket on port 1883.
1664768402: Error: Address already in use
pi@pi:~ $
```

Mosquitto를 이용한 MQTT 통신 테스트

- 실행 방법
  - (putty 터미널 1) mosquitto 브로커 실행
    - **$ mosquitto –v –c mos.conf**
  - (putty 터미널 2) subscriber 응용프로그램 실행
    - *"temp" 토픽 구독 신청*
    - **$ mosquitto_sub -h localhost -p 1883 -d -t "temp"**
  - (터미널 3) subscriber 응용프로그램 실행
    - "temp" 토픽 구독 신청
    - **$ mosquitto_sub -h localhost -p 1883 -d -t "temp"**
  - (터미널 4) publisher 응용프로그램 실행
    - "temp" 토픽으로 36.5 값을 가진 메시지 발행
    - **$ mosquitto_pub –h localhost –p 1883 –t "temp" –m 36.5**
    - 2개의 subscrber들은 모두 36.5 값이 저장된 메시지 수신

- 앞서 mosquitto를 설치할 때 함께 설치된 2개의 클라이언트 프로그램 mosquitto_sub와 mosquitto_pub 활용

터미널 1

```
pi@pi:~ $ mosquitto -v -c mos.conf
1674628450: mosquitto version 2.0.11 starting
1674628450: Config loaded from mos.conf.
1674628450: Opening ipv4 listen socket on port 1883.
1674628450: Opening ipv6 listen socket on port 1883.
1674628450: mosquitto version 2.0.11 running
1674628486: New connection from ::1:46444 on port 1883.
1674628486: New client connected from ::1:46444 as auto-7D3D
3AAC0D073319 (p2, c1, k60).
1674628486: No will message specified.
1674628486: Sending CONNACK to auto-7D3DAD6B-3B2C-90C5-126D-3AAC0D073319 (0, 0)
1674628486: Received SUBSCRIBE from auto-7D3DAD6B-3B2C-90C5-126D-3AAC0D073319
1674628486:      temp (QoS 0)
1674628486: auto-7D3DAD6B-3B2C-90C5-126D-3AAC0D073319
1674628486: Sending SUBACK to auto-7D3DAD6B-3B2C-90C5-12
1674628522: New connection from ::1:51692 on port 1883.
1674628522: New client connected from ::1:51692 as auto-43FCB
(p2, c1, k60).
1674628522: No will message specified.
1674628522: Sending CONNACK to auto-43FCBE18-DCAA-6CA1-
1674628522: Received SUBSCRIBE from auto-43FCBE18-DCAA-6C
1674628522:      temp (QoS 0)
1674628522: auto-43FCBE18-DCAA-6CA1-949B-7D84F6C87E75 0
1674628522: Sending SUBACK to auto-43FCBE18-DCAA-6CA1-949B-7D84F6C87E75
1674628545: Received PINGREQ from auto-7D3DAD6B-3B2C-90C5-126D-3AAC0D073319
1674628545: Sending PINGRESP to auto-7D3DAD6B-3B2C-90C5-126D-3AAC0D073319
1674628560: New connection from ::1:53830 on port 1883.
1674628560: New client connected from ::1:53830 as auto-E9E506
(p2, c1, k60).
1674628560: No will message specified.
1674628560: Sending CONNACK to auto-E9E50679-C25C-532A-D
1674628560: Received PUBLISH from auto-E9E50679-C25C-532A-
'temp', ... (4 bytes))
1674628560: Sending PUBLISH to auto-7D3DAD6B-3B2C-90C5-12
'temp', ... (4 bytes))
1674628560: Sending PUBLISH to auto-43FCBE18-DCAA-6CA1-949B-7D84F6C87E75 (d0, q0, r0, m0,
'temp', ... (4 bytes))
1674628560: Received DISCONNECT from auto-E9E50679-C25C-532A-DAD7-4F9CC154C8DE
1674628560: Client auto-E9E50679-C25C-532A-DAD7-4F9CC154C8DE disconnected.
```

터미널 2

```
pi@pi:~ $ mosquitto_sub -h localhost -p 1883 -d -t "temp"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: temp, QoS: 0, Options: 0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
Client (null) received PUBLISH (d0, q0, r0, m0, 'temp', ... (4 bytes))
36.5
```

터미널 3

```
pi@pi:~ $ mosquitto_sub -h localhost -p 1883 -d -t "temp"
Client (null) sending CONNECT
Client (null) received CONNACK (0)
Client (null) sending SUBSCRIBE (Mid: 1, Topic: temp, QoS: 0, Options: 0x00)
Client (null) received SUBACK
Subscribed (mid: 1): 0
Client (null) received PUBLISH (d0, q0, r0, m0, 'temp', ... (4 bytes))
36.5
```

터미널 4

```
pi@pi:~ $ mosquitto_pub –h localhost –p 1883 –t "temp" –m 36.5
Client mosqpub|2664-raspberryp sending CONNECT
Client mosqpub|2664-raspberryp received CONNACK (0)
Client mosqpub|2664-raspberryp sending PUBLISH (d0, q0, r0, m1, 'hello', ... (5
bytes))
Client mosqpub|2664-raspberryp sending DISCONNECT
pi@pi:~ $
```

7.4 파이선으로 MQTT 클라이언트 작성

paho.mqtt 라이브러리 설치

- 가상 환경 활성화 후 paho.mqtt 설치

```
pi@pi:~ $ source myenv/bin/activate
(myenv) pi@pi:~ $ pip install paho.mqtt
```

  - myenv 가상환경 디렉터리에 paho.mqtt 라이브러리 설치

- 설치 확인

```
(myenv) pi@pi:~ $ cd myenv/lib/python3.9/site-packages/paho
(myenv) pi@pi:~/myenv/lib/python3.9/site-packages/paho $ ls
__init__.py  mqtt  __pycache__
(myenv) pi@pi:~/myenv/lib/python3.9/site-packages/paho $
```

```
(myenv) pi@pi:~/myenv/lib/python3.9/site-packages/paho $ cd mqtt
(myenv) pi@pi:~/myenv/lib/python3.9/site-packages/paho/mqtt $ ls
client.py     matcher.py      properties.py  __pycache__     subscribeoptions.py
__init__.py  packettypes.py  publish.py     reasoncodes.py  subscribe.py
(myenv) pi@pi:~/myenv/lib/python3.9/site-packages/paho/mqtt $
```

Client 객체

- Client 클래스 원형

```
Client(
    callback_api_version=CallbackAPIVersion.VERSION1 #API 버전
    client_id="",                # client id
    clean_session=True,          # disconnect 될 때 세션 유지 여부
    userdata=None,               # 콜백함수의 매개변수로 전달. userdata의 값
    protocol=MQTTv311,           # MQTT 버전
    transport="tcp"              # tcp or websockets
)
```

  - Client 클래스 기능
    - 브로커에 연결, 메시지 구독 및 수신, 메시지 발행 등 클라이언트의 **대부분의 기능 포함**

- Client 객체 생성

```
import paho.mqtt.client as mqtt
...
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
```

Client 클래스의 주요 함수와 콜백 속성

구분	종류	설명
함수	connect()	브로커에 연결. 연결 완료 후 리턴하는 블록킹 함수
	connect_async()	브로커에 연결. 연결 완료 전에 리턴하는 넌블록킹 함수
	disconnect()	브로커와 연결 종료
	subscribe()	토픽으로 메시지 구독 신청
	publish()	토픽과 함께 메시지 발행
	loop()	도착한 메시지가 있는 경우 처리하고 리턴. 메시지가 없으면 타임아웃 시간까지 기다렸다가 리턴. 블록킹 함수
	loop_forever()	loop() 함수를 영원히 반복 호출. 넌블록킹 함수
	loop_start()	메시지를 기다리면서 도착한 메시지를 처리하는 무한 루프 스레드를 생성하고 리턴
	loop_stop()	loop_start() 함수가 생성한 스레드를 종료시키는 함수
콜백 속성	on_connect	브로커와 연결 완료 시 실행되는 콜백 함수를 등록하는 속성
	on_disconnect	브로커와 연결 끊김 시 실행되는 콜백 함수를 등록하는 속성
	on_publish	발행한 메시지가 브로커에 완전히 전달된 후 호출되는 콜백 함수를 등록하는 속성
	on_subscribe	브로커에 구독 신청이 완료될 때 호출되는 콜백 함수를 등록하는 속성
	on_message	브로커로부터 메시지가 도착하였을 때 호출되는 콜백 함수를 등록하는 속성

브로커에 연결

- 브로커에 연결 2가지 방법
  - connect() - 연결이 완료될 때 리턴(블록킹 함수)
  - connect_async() - 연결을 지시 후 연결 확인 없이 리턴(넌블록킹 함수)
- connect() 함수

```
connect(
   host,              # 브로커 주소. ip 주소 혹은 호스트 이름
   port=1883,         # 브로커의 포트 번호
   keepalive=60       # 접속 유지 시간(단위는 초)
)
```

  - connect() 함수 활용 방법들

```
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2) # Client 객체 생성
....
client.connect("192.168.0.5", 1883) # 192.168.0.5의 브로커에 1883 포트로 연결
client.connect("192.168.0.5") # 192.168.0.5의 브로커에 1883 포트로 연결
client.connect("localhost") # 현재 컴퓨터에서 실행되는 브로커에 1883 포트로 연결
```

    - connect()는 브로커와 연결이 완료될 때 리턴하므로, 다음과 같이 활용

```
client.connect("192.168.0.5", 1883) # 브로커에 연결
# 브로커와 연결이 완료된 상태
client.subscribe("temp") # 연결이 완료되었으니 브로커에 구독 신청
```

브로커에 연결, connect_async() 함수 사용

- connect_async() 함수 원형

```
connect_async(
    host,              # 브로커 주소. ip 주소 혹은 호스트 이름
    port=1883,         # 브로커의 포트 번호
    keepalive=60       # 접속 유지 시간(단위는 초)
)
```

  - 브로커에 연결을 지시하고 연결을 확인하지 않은 채 바로 리턴
    - connect_async()로부터 리턴한 직후 브로커와 연결되었다고 보장할 수 없음
    - 반드시 on_connect() 콜백과 함께 사용
      - *on_connect() 함수는 연결이 완료될 때 호출되는 콜백 함수*
      - *on_connect()가 실행되면 연결이 완료됨이 확실함*
  - connect_async() 함수 활용 방법

```
client.connect_async("192.168.0.5", 1883) # 브로커에 연결
# 현재 브로커와 연결 완료를 보장할 수 없는 상태
client.subscribe("temp") # 브로커에 구독 신청. 구독 신청 실패 가능성 높음
```

on_connect() 콜백 함수

- on_connect() 콜백 함수
  - 연결 완료시 호출되는 함수

```
on_connect(
   client,          # connect를 요청한 client 객체의 레퍼런스
   userdata,        # client 생성시 저장한 userdata
   flags, rc        # connect 결과
   prop             # properties
)
```

  - 활용 사례

```
def on_connect(client, userdata, flags, rc, prop=None) :
   print("연결 완료")
...
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2) # 클라이언트 객체 생성
client.on_connect = on_connect      # 클라이언트에 콜백 등록
client.connect(...) # 브로커에 연결. client.connect_async()를 사용해도 됨
```

브로커와 연결 끊기

- Client 클래스의 disconnect() 함수 사용

```
client.disconnect()      # 브로커와 연결 끊기
```

- on_disconnect() 콜백 활용
  - 브로커와 연결이 끊긴 경우 on_disconnect 콜백 호출
    - 브로커가 갑자기 종료한 경우,
    - 브로커가 keepalive 시간 동안 메시지가 도착하지 않을 때 강제로 클라이언트와 연결을 끊은 경우

```
def on_disconnect(
   client,          # 콜백 함수를 소유한 클라이언트 객체에 대한 레퍼런스
   userdata,        # user_data_set() 함수로 설정하는 사적 사용자 데이터
   rc               # disconnect 결과
)
```

  - on_disconnect() 콜백 활용 사례

```
def on_disconnect(client, userdata, rc) :
   print("연결이 끊어졌습니다. ")
...
client.on_disconnect = on_disconnect
```

메시지 발행

- publish() 함수로 메시지 발행

```
publish(
   topic,              # 토픽 문자열
   payload=None,       # 전달 메시지 텍스트
   qos=0               # QoS 레벨
}
```

on_publish() 콜백 함수

- 메시지가 브로커에 전달되면 on_publish() 콜백함수 실행

```
def on_publish(topic, userdata, mid, granted_qos) :
   print("메시지가 전송되었습니다.")
...
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)   # 클라이언트 객체 생성
client.on_publish = on_publish     # on_publish 콜백 함수 등록
client.connect(...)                # 브로커에 연결
client.publish()                   # 메시지 발행
```

메시지 수신

- subscribe() 함수로 메시지 구독 신청
  - 메시지를 수신하기 위해 브로커에 토픽 등록

```
subscribe(
    topic,     # 수신 받고자 하는 메시지의 토픽 문자열
    qos=0      # QoS 레벨
)
```

- on_message() 콜백 함수
  - subscriber가 브로커로부터 메시지를 받았을 때 실행되는 함수

```
on_message(
    client,          # subscribe를 실행한 client
    userdata,        # client 생성시 저장한 userdata
    message          # 수신된 메시지
)
```

  - on_message() 콜백으로 메시지 수신

```
def on_message(client, userdata, msg):
    print("토픽 ", msg.topic, "의 메시지 ", str(msg.payload), "가 도착하였습니다.")
...
client.on_message = on_message                # 콜백 함수 등록
```

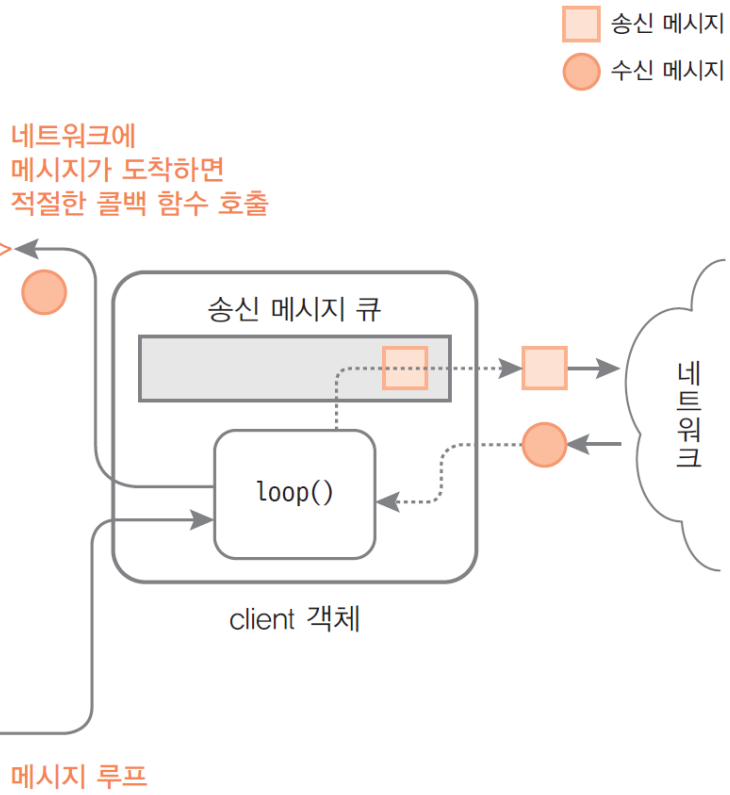
메시지 루프 만들기

- 메시지 루프
  - 클라이언트가 네트워크로 메시지를 받거나 보내는 코드
  - 메시지 송수신을 위해 반드시 작성 필요
  - 메시지 루프를 만들기 위해 **Client 클래스 활용**
- Client 클래스가 메시지 루프를 위해 제공하는 기능
  - 네트워크로 메시지를 송수신하는 3개의 함수 제공
    - loop()
    - loop_forever()
    - loop_start()/loop_stop()
  - 1개의 송신 메시지 큐 제공
    - 브로커로 전송할 메시지들을 담은 큐(outbound message queue)
    - publish() 함수에 의해 발행된 메시지가 일시적으로저장
    - 하나씩 순서대로 브로커로 전송

loop()

- loop() 함수
  - 네트워크로 보내야 하거나 네트워크에 도착한 메시지가 있는 경우 1개의 메시지만 처리하고 리턴
    - **무한 루프 아님**
    - 메시지가 도착하였거나, 메시지를 전송한 경우,
    - on_connect(), on_subscribe(), on_message(), on_disconnect() 등 호출

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
  print("연결되었습니다. ")
  client.subscribe("temp")

def on_subscribe(client, userdata, msg):
  print("구독 신청 완료되었습니다.")

def on_message(client, userdata, msg):
  print(msg.topic, " ", str(msg.payload), " 수신")

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect # 콜백함수 등록
client.on_subscribe = on_subscribe # 콜백함수 등록
client.on_message = on_message # 콜백함수 등록
client.connect("192.168.0.5", 1883)

# 메시지 루프 작성
while True :
  client.loop() # 한 번만 메시지 처리
  # 이곳에 사용자는 필요한 코드 작성
```

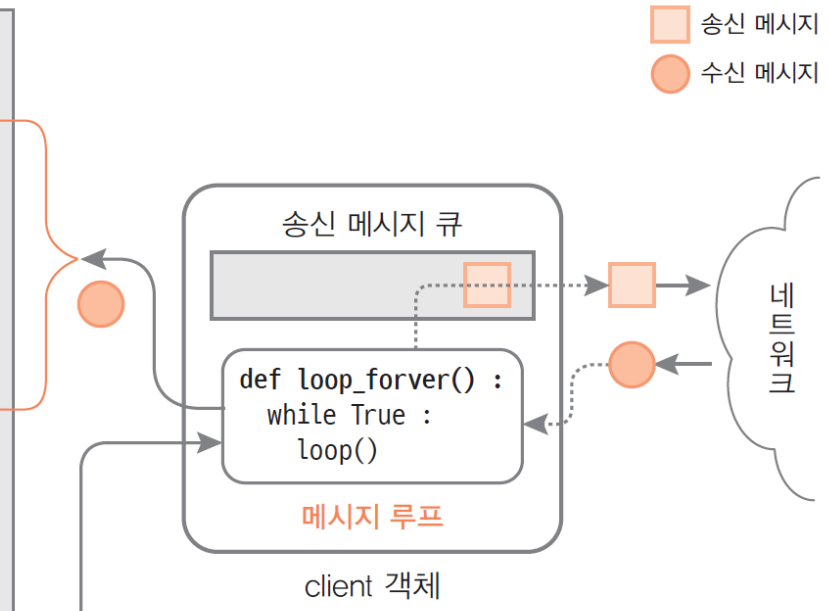


* 현재 라즈베리파이의 IP 주소는 192.168.0.5이고 이곳에 브로커가 실행 중이라는 가정을 하고 있으므로 client.connect()가 이 주소로 연결

loop_forever()

- loop_forever()
  - loop()를 반복 호출하는 무한 루프
    - 메시지가 도착하면 on_connect()나 on_message() 등 모든 메시지 처리, 송신할 메시지 있으면 송신

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
  print("연결되었습니다. ")
  client.subscribe("temp")

def on_subscribe(client, userdata, msg):
  print("구독 신청 완료되었습니다.")

def on_message(client, userdata, msg):
  print(msg.topic, " ", str(msg.payload), " 수신")

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect # 콜백함수 등록
client.on_subscribe = on_subscribe # 콜백함수 등록
client.on_message = on_message # 콜백함수 등록
client.connect("192.168.0.5", 1883)
client.loop_forever() # 메시지 루프
```



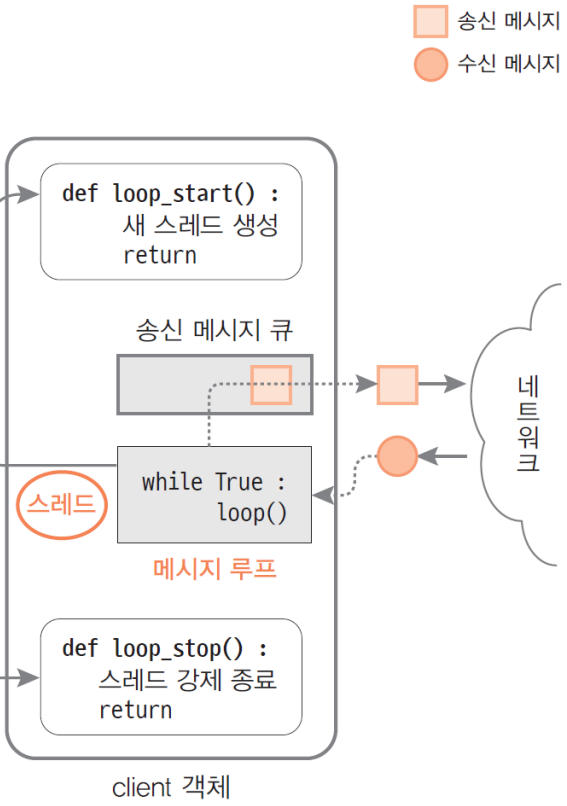
* 현재 라즈베리파이의 IP 주소는 192.168.0.5이고 이곳에 브로커가 실행 중이라는 가정을 하고 있으므로 client.connect()가 이 주소로 연결

loop_start()와 loop_stop()

- loop_start()와 loop_stop()
  - loop_start() 함수가 메시지 루프를 실행하는 스레드 생성
  - 이 스레드가 loop_stop()이 호출될 때까지 loop() 반복 호출

```
import time
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
  print("연결되었습니다. ")
  client.subscribe("temp")

def on_subscribe(client, userdata, msg):
  print("구독 신청 완료되었습니다.")

def on_message(client, userdata, msg):
  print(msg.topic, " ", str(msg.payload), " 수신")

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect # 콜백함수 등록
client.on_subscribe = on_subscribe # 콜백함수 등록
client.on_message = on_message # 콜백함수 등록
client.connect("192.168.0.5", 1883)
client.loop_start()

while True :
  temp = input("온도입력>>")
  client.publish("temp", temp)
  time.sleep(1)

client.loop_stop()
client.disconnect()
```

메시지 처리와 병렬적으로 처리하고자하는 작업



* 현재 라즈베리파이의 IP 주소는 192.168.0.5이고 이곳에 브로커가 실행 중이라는 가정을 하고 있으므로 client.connect()가 이 주소로 연결

keepalive와 메시지 루프

- keepalive 시간이란?
  - 브로커가 클라이언트로부터 아무 메시지를 수신하지 못하였을 때, 클라이언트와 연결이 끊어진 것으로 판단하는 시간
- connect() 함수에 keepalive 매개변수 있음
  - 클라이언트는 브로커에 접속할 때 connect() 함수의 매개변수에 keepalive 시간 전달
  - 클라이언트가 브로커에 연결된 후, keepalive 시간 내에 접속된 클라이언트로부터 아무 메시지가 없으면, 브로커는 클라이언트와 연결 강제 종료
- 응용프로그램이 브로커와 연결을 계속 유지하는 방법
  - 응용프로그램이 keepalive 시간 내에 메시지를 계속 보내야 브로커와 연결 유지 가능 -> 보낼 메시지가 없으면 어떡하지?
  - loop(), loop_forever() 혹은 loop_start() 힘수가 자동으로 ping 메시지를 keepalive 시간 내에 주기적으로 브로커로 전송, 브로커와 연결 유지
  - **응용프로그램에서 loop(), loop_forever() 혹은 loop_start()를 사용하면, 이들이 자동으로 ping 메시지 전송**

예제 7-1 텍스트 메시지 전송

사용자에게 문자열을 입력받아 "letter" 토픽으로 전송하는 응용 프로그램을 작성해보자.

- 토픽 : "letter"
- 메시지 : 사용자로부터 입력받은 문자열 텍스트
- mosquitto 브로커 : 라즈베리파이에서 실행. 1883 포트 이용
- subscriber : 라즈베리파이에서 실행. 브로커로부터 문자열을 수신하여 화면에 출력. 7-1sub.py
- publisher : 라즈베리파이에서 실행. 사용자로부터 문자열을 입력받아 전송. 7-1pub.py

7-1sub.py

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
    print("접속 결과: "+ str(rc)) # 접속 결과 출력, rc가 0이면 성공
    client.subscribe("letter") # letter 토픽으로 구독 신청

def on_message(client, userdata, msg):
    print(msg.topic, end=", ") # 토픽 출력
    print(str(msg.payload.decode("utf-8"))) # 메시지 출력

ip = input("브로커의 IP 주소>>")

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect # on_connect 콜백 함수 등록
client.on_message = on_message # on_message 콜백 함수 등록
client.connect(ip, 1883)
client.loop_forever() # 메시지 루프
```

7-1pub.py

```
import time
import paho.mqtt.client as mqtt

ip = input("브로커의 IP 주소>>")

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.connect(ip, 1883) # 브로커에 연결
client.loop_start() # 메시지 루프를 실행하는 스레드 생성

while True:
    message = input("문자메시지>>") # 사용자로부터 문자열 입력
    if message == "exit" :
        break
    client.publish("letter", message)
    print("메시지 전송: %s" % message)

client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()
```

예제 7-1 실행

putty 터미널 1

```
pi@pi:~ $ mosquitto –v –c mos.conf
1702049934: mosquitto version 2.0.11 starting
1702049934: Config loaded from mos.conf.
1702049934: Opening ipv4 listen socket on port 1883.
1702049934: Opening ipv6 listen socket on port 1883.
1702049934: mosquitto version 2.0.11 running
```

putty 터미널 2

```
(myenv) pi@pi:~/ch07 $ python 7-1pub.py
문자메시지>>Good Morning!
메시지 전송: Good Morning!
문자메시지>>즐거운 날입니다.
메시지 전송: 즐거운 날입니다.
문자메시지>>이제 공부하러 갈까요?
메시지 전송: 이제 공부하러 갈까요?
문자메시지>>너무 재미있어요. 랄랄랄
메시지 전송: 너무 재미있어요. 랄랄랄
문자메시지>>exit
(myenv) pi@pi:~/ch07 $
```

putty 터미널 3

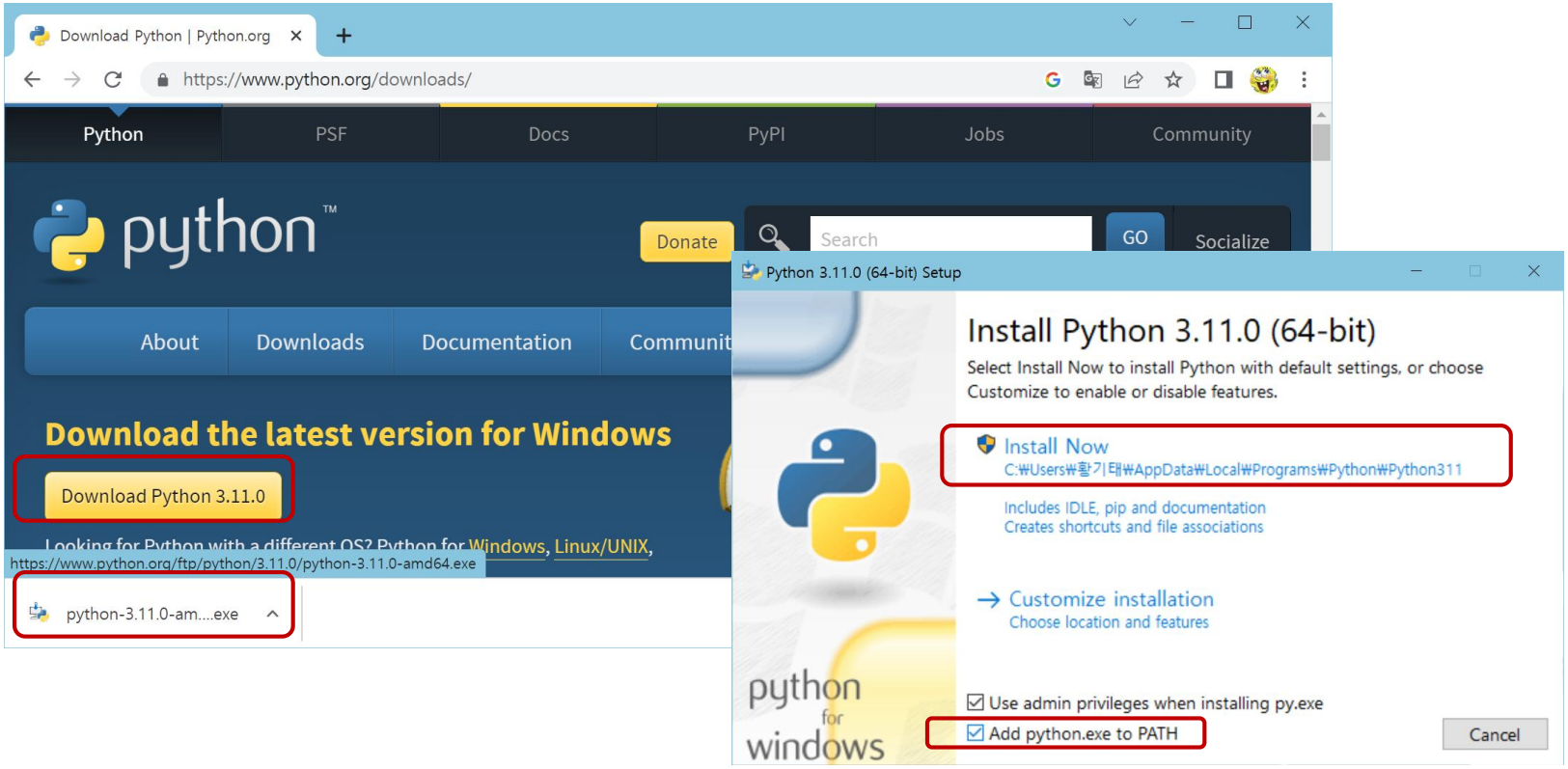
```
(myenv) pi@pi:~/ch07 $ python 7-1sub.py
브로커의 IP 주소>>192.168.0.5
접속 결과: 0
letter, Good Morning!
letter, 즐거운 날입니다.
letter, 이제 공부하러 갈까요?
letter, 너무 재미있어요. 랄랄랄
```

7.5 라즈베리파이와 윈도우 사이의 메시지 전송

1. 라즈베리파이와 윈도우 노트북은 반드시 동일한 무선 네트워크(무선 공유기 혹은 모바일 핫스팟) 에 연결되어 있어야 한다.

윈도우에 파이썬과 paho-mqtt 라이브러리 설치

- 파이선 설치
  - https://www.python.org/downloads/



파이선 설치 확인


```
C:₩>python
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

실행을 종료하려면 [Ctrl+Z] 키를 입력하고
다시 엔터키를 입력하면 된다.

paho-mqtt 라이브러리 설치

- paho-mqtt 라이브러리
  - MQTT 클라이언트를 작성할 때 필요한 라이브러리
- 설치
  - **C:₩>pip install paho-mqtt**
  - 설치 확인

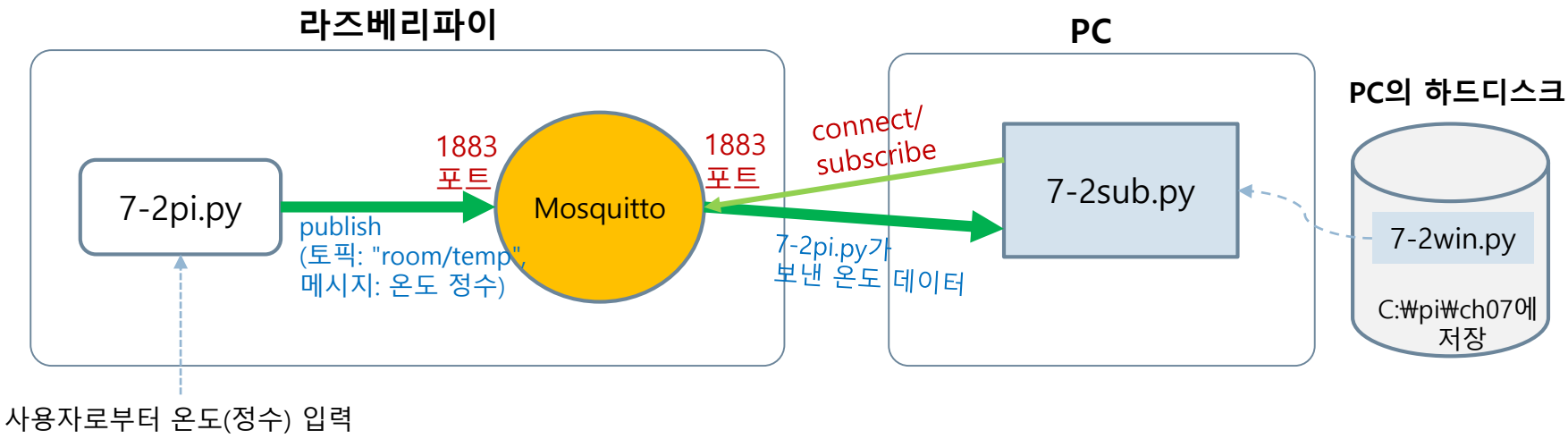
```
C:₩>python
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import paho.mqtt
>>>
```

import paho.mqtt 명령 입력 후 오류가 발생하지 않으면 설치 성공

예제 7-2 라즈베리파이에서 온도 값 보내고 윈도우에서 수신

라즈베리파이에서 사용자로부터 온도 값을 입력받아 브로커로 전송하고, 윈도우로 이를 수신하여 출력하는 MQTT 응용 시스템을 만들어 보자.

- 토픽 : "room/temp"
- 메시지 : 온도. 정수로 전송
- mosquitto 브로커 : 라즈베리파이에서 실행. 1883 포트 이용
- publisher : 라즈베리파이에서 실행. 사용자로부터 온도를 입력받아 전송. 7-2pi.py
- subscriber : 윈도우에서 실행되며, 온도를 수신하여 화면에 출력. 7-2win.py



* 라즈베리파이에서 실행되는 7-2pi.py는 pi 디렉터리의 ch07 디렉터리에 저장)

예제 7-2 코드

7-2pi.py(라즈베라파이의 pi/ch07 디렉터리에 저장)

```
import paho.mqtt.client as mqtt

broker_ip = "localhost"  # 이 컴퓨터의 브로커에 접속

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.connect(broker_ip, 1883) # 1883 포트로 브로커에 접속
client.loop_start() # 메시지 루프를 실행하는 스레드 생성

while True:
        temperature = input("온도>>>") # 사용자로부터 온도 입력
        temperature = int(temperature) # 문자열을 정수로 변환
        if temperature == 0 : # 온도에 0이 입력되면 while 문 벗어남
                break
        client.publish("room/temp", temperature) # 온도 전송

client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()
print("프로그램 종료")
```

7-2win.py(윈도우 노트북에서 실행, C:₩pi₩ch07에 저장)

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
    print("room/temp 토픽으로 메시지 구독 신청")
    client.subscribe("room/temp") # 메시지 구독 신청

def on_message(client, userdata, msg):
    print(msg.topic, end=", ") # 토픽 출력
    print(int(msg.payload)) # 메시지 데이터 출력

ip = input("브로커의 IP 주소>>") # 브로커 IP 주소 입력

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect=on_connect
client.on_message=on_message
client.connect(ip, 1883) # 1883 포트로 브로커에 접속
client.loop_forever() # 메시지 루프 실행
```

예제 7-2 실행

(1) putty 터미널 열고, mosquitto 실행

```
pi@pi:~ $ mosquitto –v –c mos.conf
1702180719: mosquitto version 2.0.11 starting
1702180719: Config loaded from mos.conf.
1702180719: Opening ipv4 listen socket on
port 1883.
1702180719: Opening ipv6 listen socket on
port 1883.
1702180719: mosquitto version 2.0.11 running
```

(2) putty 터미널 1개 더 열고, 라즈베리파이에 접속.
myenv 가상환경 활성화 후
pi/ch07 디렉터리로 이동 후 7-2pi.py 실행

```
(myenv) pi@pi:~/ch07 $ python 7-2pi.py
온도>>>23
온도>>>34
온도>>>66
온도>>>77
온도>>>0
프로그램 종료
(myenv) pi@pi:~/ch07 $
```

입력된 값은 윈도우로 전송

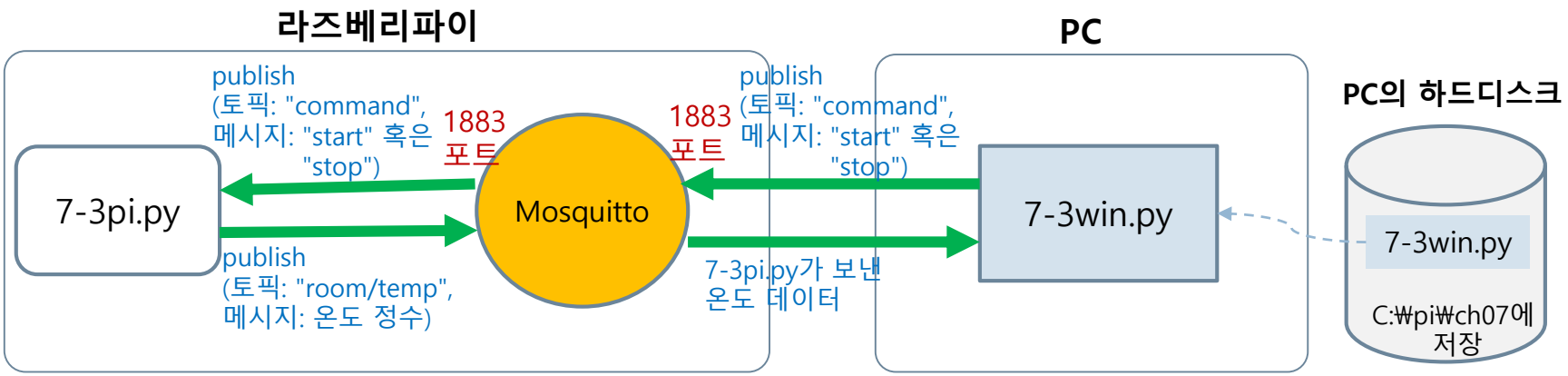
(3) 윈도우에서 명령창 열고
C:₩pi₩ch07로 이동하여 7-2win.py 실행

```
C:₩pi₩ch07>python 7-2win.py
브로커의 IP>> 192.168.0.5
room/temp 토픽으로 메시지 구독 신청
room/temp, 23
room/temp, 34
room/temp, 66
room/temp, 77
```

예제 7-3 윈도우에서 라즈베리파이에 온도 전송 시작/중지 지시

예제 7-2를 수정하여 윈도우에서 라즈베리파이로 온도 전송을 시작시키거나 중단시키는 명령을 내리도록 구현하라.

- 토픽 : "room/temp"와 "command"
- 메시지 : 온도와 "start" 혹은 "stop" 텍스트
- mosquitto 브로커 : 라즈베리파이에서 실행. 1883 포트 이용
- 라즈베리파이에서 실행되는 클라이언트(7-3pi.py) : 윈도우로부터 "command" 토픽의 메시지를 수신하고, "start" 메시지가 오면 2초 간격으로 랜덤하게 온도를 전송하고, "stop" 메시지의 경우 온도 전송 중지
- 윈도우에서 실행되는 클라이언트(7-3win.py) : 사용자의 입력에 따라 "command" 토픽으로 "start"나 "stop" 메시지를 보낸다. 라즈베리파이로부터 온도는 수신하기 위해 "room/temp" 토픽 구독



예제 7-3 코드

7-3pi.py(라즈베라파이의 pi/ch07 디렉터리에 저장)

```
import paho.mqtt.client as mqtt
import time
import random

flag = "stop" # 온도 전송을 하지 않음을 뜻함
def on_connect(client, userdata, flag, rc, prop=None):
    client.subscribe("command") # command 토픽으로 메시지 구독 신청

def on_message(client, userdata, msg):
    global flag # 전역변수 flag 사용
    command = msg.payload.decode('utf-8')
    if command == "start":
        flag = "start"
        print("온도 전송...")
    elif command == "stop":
        flag = "stop"
        print("₩n전송 중단...")
    else:
        print("명령 오류: ", command)

broker_ip = "localhost" # 이 컴퓨터의 브로커에 접속
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect
client.on_message = on_message
client.connect(broker_ip, 1883) # 1883 포트로 브로커에 접속
client.loop_start() # 메시지 루프를 실행하는 스레드 생성

print("시작 명령 대기 ...")
while True:
    if flag == "start":
        temperature = random.randint(0, 40) # 0에서 39사이의 랜덤 정수
        client.publish("room/temp", temperature) # 온도 전송
        print(temperature, end=" ", flush=True)
    time.sleep(2) # 2초 동안 잠자기

client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()
print("프로그램 종료")
```

7-3win.py(윈도우 노트북에서 실행, C:₩pi₩ch07에 저장)

```
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
    client.subscribe("room/temp") # 메시지 구독 신청

def on_message(client, userdata, msg):
    print(msg.topic, end=", ") # 토픽 출력
    print(int(msg.payload)) # 메시지 데이터 출력

ip = input("브로커의 IP 주소>>") # 브로커 IP 주소 입력
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect=on_connect
client.on_message=on_message
client.connect(ip, 1883) # 1883 포트로 브로커에 접속
client.loop_start() # 메시지 루프 시작
while True:

    command = input("전송 시작/중단(start/stop 입력)>>")
    if command == "start" or command == "stop":
        client.publish("command", command) # 토픽 "command"로 "start"
혹은 "stop" 메시지 전송

client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()
```

예제 7-3 실행

(1) putty 터미널을 열고, 라즈베리파이에 접속 후 mosquitto 실행

```
pi@pi:~ $ mosquitto –v –c mos.conf
1702180719: mosquitto version 2.0.11 starting
1702180719: Config loaded from mos.conf.
1702180719: Opening ipv4 listen socket on port 1883.
1702180719: Opening ipv6 listen socket on port 1883.
1702180719: mosquitto version 2.0.11 running
```

(2) putty 터미널 1개 더 열고, 라즈베리파이에 접속.
myenv 가상환경 활성화 후
pi/ch07 디렉터리로 이동 후 7-3pi.py 실행

```
(myenv) pi@pi:~/ch07 $ python 7-3pi.py
시작 명령 대기 ...
온도 전송...
34 36 19 13
전송 중단...
온도 전송...
32 13 21 16
전송 중단..
```

(3) 윈도우에서 명령창 열고
C:₩pi₩ch07로 이동하여 7-3win.py 실행

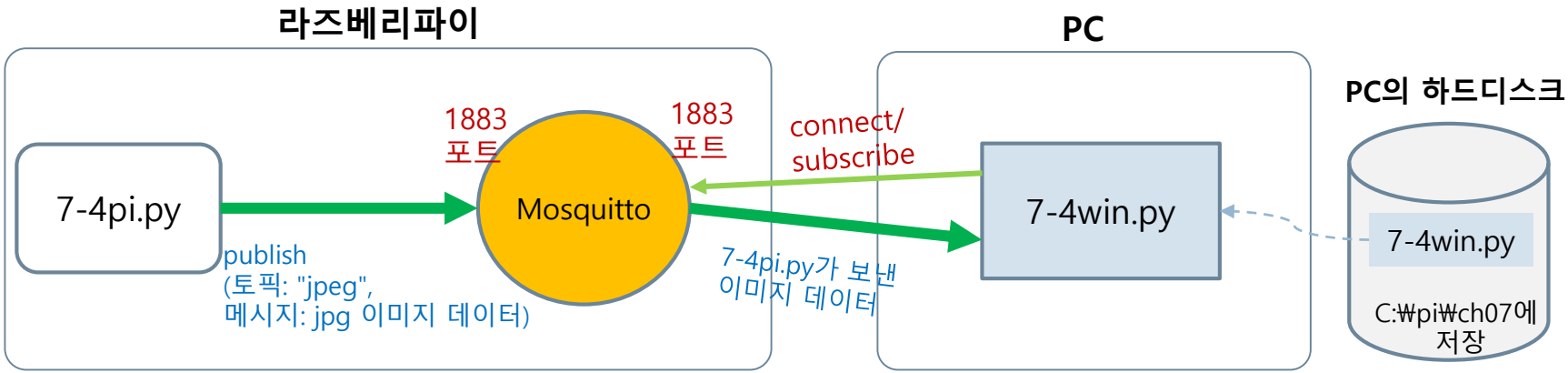
```
C:₩pi₩ch07>python 7-3win.py
브로커의 IP>> 192.168.0.5
전송 시작/중단(start/stop 입력)>>start
전송 시작/중단(start/stop 입력)>>
room/temp, 34
room/temp, 36
room/temp, 19
room/temp, 13
stop
전송 시작/중단(start/stop 입력)>>start
전송 시작/중단(start/stop 입력)>>
room/temp, 32
room/temp, 13
room/temp, 21
room/temp, 16
전송 시작/중단(start/stop 입력)>>stop
전송 시작/중단(start/stop 입력)>>
```

stop 입력되면 전송 중단

예제 7-4 윈도우에서 라즈베리파이가 촬영한 카메라 이미지 수신

라즈베리파이에서 카메라로 사진을 촬영하여 이미지를 전송하면 윈도우에서 이미지를 수신하여 jpeg 파일로 저장하는 응용 시스템을 만들어보자.

- 토픽 : "jpeg"
- 메시지 : 이미지 데이터(바이너리 데이터)
- mosquitto 브로커 : 라즈베리파이에서 실행. 1883 포트 이용
- publisher : 라즈베리파이에서 실행. 사진 촬영 후 이미지 바로 전송. 7-4pi.py
- subscriber : 윈도우에서 실행되며, 이미지를 받아 파일로 저장. 7-4win.py



예제 7-4 코드 - 라즈베리파이에서 실행되는 7-4pi.py

7-4pi. py

```
import io
import cv2
import paho.mqtt.client as mqtt

broker_ip = "localhost"

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.connect(broker_ip, 1883) # 1883 포트로 mosquitto에 접속
client.loop_start() # 메시지 루프를 실행하는 스레드 생성

# 카메라 객체를 생성하고 촬영한 사진 크기를 640x480으로 설정
camera = cv2.VideoCapture(0, cv2.CAP_V4L)
camera.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
camera.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# 프레임을 임시 저장할 버퍼 개수를 1로 설정
buffer_size = 1
camera.set(cv2.CAP_PROP_BUFFERSIZE, buffer_size)
```

```
while True:
    key = input("사진촬영계속?(종료는 stop 입력)>>")
    if(key == "stop"): # stop이 입력되면 while 문을 벗어남
        break
    # 버퍼에 저장된 모든 프레임을 버리고 새 프레임 읽기
    for i in range(buffer_size+1):
        ret, frame = camera.read()

    im_bytes = cv2.imencode('.jpg', frame)[1].tobytes() # 바이트 배열로 저장
    client.publish("jpeg", im_bytes, qos = 0) # 이미지 전송

camera.release() # 카메라 사용 끝내기
client.loop_stop() # 메시지 루프를 실행하는 스레드 종료
client.disconnect()
```

예제 7-4 코드 윈도우에서 실행될 7-4win.py (카메라 이미지 수신 프로그램)

프로그램 실행 전에 C:₩pi₩ch07 디렉터리에 다음 7-4win.py 코드를 작성하여 저장한다.
그리고 라즈베리파이로부터 받은 이미지를 저장할 data 디렉터리(C:₩pi₩ch07₩data)를 만든다.

7-4win. py

```
import io
import time
import paho.mqtt.client as mqtt

def on_connect(client, userdata, flag, rc, prop=None):
    print("jpeg 토픽으로 메시지 구독 신청")
    client.subscribe("jpeg")

def on_message(client, userdata, msg):
    filename = './data/image%d.jpg' % (time.time()*10)
    file = open(filename, "wb") # 파일 열기. 없으면 새로 생성
    file.write(msg.payload) # 수신한 이미지를 파일에 쓰기
    file.close() # 파일 닫기
    print("이미지수신 %s" % filename)

ip = input("브로커의 IP>>") # 브로커 IP 주소 입력

client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect
client.on_message = on_message
client.connect(ip, 1883)
client.loop_forever() # 메시지 루프 실행
```

수신한 이미지를 C:₩pi₩ch07₩data 디렉터리에 파일로 저장하기 때문에, 이 예제 실행 전 **C:₩pi₩ch07₩data 디렉터리를 만들어 두어야 한다.**

예제 7-4 실행 및 결과

(1) putty 터미널 열고 라즈베리파이에서 접속,
홈 디렉터리에서 mosquitto 실행

```
pi@pi:~ $ mosquitto –v –c mos.conf
1702180719: mosquitto version 2.0.11 starting
1702180719: Config loaded from mos.conf.
1702180719: Opening ipv4 listen socket on port 1883.
1702180719: Opening ipv6 listen socket on port 1883.
1702180719: mosquitto version 2.0.11 running
```

(2) putty 터미널 1개 더 열고 라즈베리파이에서 실행
myenv 가상환경 활성화 후
pi/ch07 디렉터리로 이동 후 7-4pub.py 실행

```
(myenv) pi@pi:~/ch07 $ python 7-4pi.py
사진촬영계속?(종료는 stop 입력)>>yes
사진촬영계속?(종료는 stop 입력)>>yes
사진촬영계속?(종료는 stop 입력)>>yes
사진촬영계속?(종료는 stop 입력)>>stop
(myenv) pi@pi:~/ch07 $
```

(3) 윈도우에서 실행

```
C:₩pi₩ch07>python 7-4win.py
브로커의 IP>>192.168.0.5
jpeg 토픽으로 메시지 구독 신청
이미지수신 ./data/image16747870188.jpg
이미지수신 ./data/image16747871061.jpg
이미지수신 ./data/image16747871229.jpg
```