Table of contents

1 Kankun wifi plug MQTT + OpenHAB2로 제어하기

1.1 Mosquitto client(No-SSL 버전) 설치

다음의 두 명령어를 Kankun wifi plug에 SSH 접속 후 실행하면 MQTT client 중 하나인 mosquitto_pub과 mosquito_sub가 설치된다.

```
opkg update
opkg install mosquitto-client-nossl
```

1.2 메시지 송수신 테스트

1.2.1 Kankun --> MQTT

- MQTT 서버쪽에서 mosquitto_sub -h localhost -t /testsw/1/state -u user_1 -P password_1 실행해서 메시지 수신 대기
- Kankun 기기에서 mosquitto_pub -h mqtt_server_ip -t /testsw/1/state -m 1 -u user_1 -P password_1 실행해서 메시지 송신해서 MQTT서버측에 잘 송신되는지 확인

1.3 MQTT --> Kankun

- Kankun 기기에서 mosquitto_sub -h mqtt_server_ip -t /testsw/1/state -u user_1 -P password_1 실행해서 메시지 수신 대기
- MQTT 서버쪽에서 mosquitto_pub -h localhost -t /testsw/1/state -m 1 -u user_1 -P password_1 실행해서 메시지 송신해서 Kankun 스위치쪽으로 잘 송신되는지 확인

2번의 테스트를 통해 망이나 방화벽 상관없이 외부 MQTT 서버를 통해 메시지가 문제없이 송수신 됨을 확인했다. 즉, 스위치가 어떤 환경에 있던지 상관없이 외부망에 있는 브로커를 통해 어디서든 통신이 가능하다는 점을 확인한 것이다.

1.4 Kankun 스위치에서 현재 상태 publish

- mkdir /root/scripts
- vi /root/scripts/mqtt_publish.sh 해서 다음을 입력

```
# !/bin/sh

while true
do
 CURR_STATUS="₩`cat /sys/class/leds/tp-link:blue:relay/brightness₩`"
 mosquitto_pub -h mqtt_server_ip -t /testsw/1/state -u user_1 -P password_1 -m $CURR_STATUS
 sleep 5
done
```

- chmod 755 /root/scripts/mqtt_publish.sh

자동 시작되도록 vi /etc/profile 해서 /root/scripts/mqtt_publish.sh &을 추가한다.

이제 reboot 실행해서 재부팅하고 접속 후에 테스트해보자. * echo "1" > /sys/class/leds/tp-link:blue:relay/brightness 해서 안드로이드앱의 스위치 상태가 ON으로 변경되는지 확인

- echo 0; > /sys/class/leds/tp-link:blue:relay/brightness‘ 해서 안드로이드앱의 스위치 상태가 OFF으로 변경되는지 확인

1.5 MQTT 서버에서 Kankun 스위치 제어

MQTT를 통해 상태를 알 수 있게 되었으니 이제는 제어해보자.

Kankun 기기에서 mkdir /root/work 한 후 vi /root/scripts/mqtt_subscribe.sh 해서 다음을 입력한다.

```
#!/bin/sh
mosquitto_sub -h mqtt_server_ip-t /testsw/1/command -u user_1-P password_1 >> /root/work/mqtt.dat 2>/dev/null
```

chmod 755 /root/scripts/mqtt_subscribe.sh 한 다음 vi /etc/profile 해서 /root/scripts/mqtt_subscribe.sh & 을 추가해준다.

제어를 위해서 vi /root/scripts/mqtt_command.sh 해서 다음을 입력한다.

```
#!/bin/sh
while true
do
 if [ -e /root/work/mqtt.dat ]; then
  COMMAND="₩`tail -1 /root/work/mqtt.dat₩`"
 fi

 if [ "$COMMAND" = "1" ]; then
  echo "1" > /sys/class/leds/tp-link:blue:relay/brightness
 elif [ "$COMMAND" = "0" ]; then
  echo "0" > /sys/class/leds/tp-link:blue:relay/brightness
 fi
 echo "" > /root/work/mqtt.dat
 sleep 5
done
```

chmod 755 /root/scripts/mqtt_command.sh 후 vi /etc/profile 해서 /root/scripts/mqtt_command.sh & 을 추가해주면 준비가 끝이 난다. reboot 실행해서 재부팅 한 후 기기에 접속해서 테스트해보자.

OpenHAB앱에서 스위치 ON해보니 대략 5초가 소요된 후 Kankun 기기의 /sys/class/leds/tp-link:blue:relay/brightness가 1로 변경되고 스위치가 ON되었다.

1.6 스위치를 추가할 때 변경해야 하는 부분들

- /etc/openhab2/item/switch.items에 스위치별 topic(제어용/모니터링용) 선정 후 스위치 추가
- /etc/openhab2/sitemap/Test.sitemap에 추가된 스위치 추가
- /etc/mosquitto/mosquitto.acl에 추가된 topic을 사용자별로 추가

1.7 이제

Kankun 스위치에 위의 스크립트를 복사해 넣고 topic을 변경 기나긴 설정과 테스트의 끝에 드디어 Kankun wifi plug를 OpenHAB2 + MQTT 기반으로 제어 및 모니터링에 성공했다. 전용앱으로 쉽게 갈 수 있는 길을 어렵게 간 듯 하지만, 나중에 구현할 Follow Me 기능(특정 스마트폰이 집안 인터넷에 무선랜 접속시 뭔가를 하거나, 무선랜에서 접속이 사라지면 뭔가를 하거나 하는 등의 기능)을 위해서는 OpenHAB과 MQTT가 꼭 필요했다. 언제 다시 할 수 있을지는 모르겠지만.