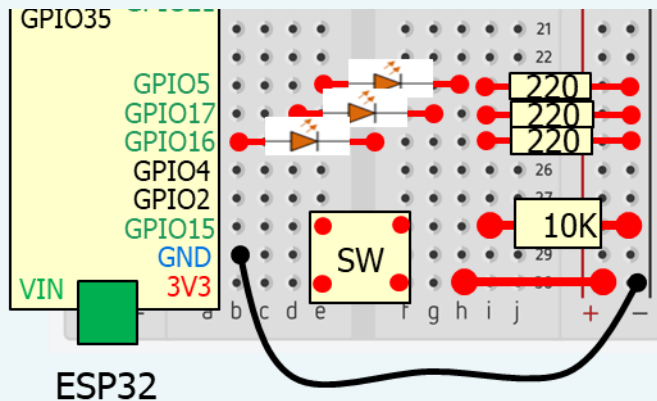


# Task09- Step B

Make thing (ESP32 button)

1. Connect ESP32Button to AWS-IoT  
(pub: esp32/button, sub: esp32/buttAck)
2. Test to Publish and Subscribe Topics with MQTT Test Client
3. Configure Rule-Engine as follows:  
<if Button Pressed>
  - send Email to yourself with Button ID
  - reply ACK to ESP32 {"ack":{"message": "ACK for button"}}



```
Connecting to WiFi..
Connected to wifi
Connected to AWS
Subscribe Successfull
Publish failed
Publish Message:ESP32-Button:001 Pressed!!
Received Message:{"ack":{"message":"ACK for button"}}
```

☒ 자동 스크롤 ☐ 타임스탬프 표시

mail to gen1223 3 받은편지함 AWS Notification Message - {"ack":{"message":"ACK for button"}} - If you wi

# Task09- Step A

- Install **MQTT-fx**.. from Internet..
- Thing을 만들기 전에 Topic/Payload 동작 확인
- Broker addr:  
Thing Rest-API endpoint
- Broker Port: 8883
- CA, Certificate,  
Private-key file 지정  
(for my-thing)

The screenshot shows the 'MQTT Broker Profile Settings' window. At the top, 'Profile Name' is 'myIoTButton' and 'Profile Type' is 'MQTT Broker'. The 'MQTT Broker Profile Settings' section includes 'Broker Address' (ats.iot.ap-northeast-2.amazonaws.com), 'Broker Port' (8883), and 'Client ID' (MQTT\_FX\_Client) with a 'Generate' button. Below this are tabs for 'General', 'User Credentials', 'SSL/TLS' (selected), 'Proxy', and 'LWT'. In the 'SSL/TLS' tab, 'Enable SSL/TLS' is checked, and 'Protocol' is 'TLSv1.2'. Under 'Self signed certificates', there are fields for 'CA File', 'Client Certificate File', and 'Client Key File', all pointing to files in the path 'G:\내 드라이브\@KAU\교육과정\2020-2\IoT\AWS-IoT\Amazon\_Roc'. The 'Client Key Password' field is empty. At the bottom, 'PEM Formatted' is checked.

# Task09- Step A

---

- Rule-Engine Setup:

1. 온도가 40 이상이면 자신의 email로 통보하고
2. 온도가 30 이상이고 습도가 40 이상일때 LED를 OFF
3. 온도가 20 이하이고 습도가 10 이하일때 LED를 ON

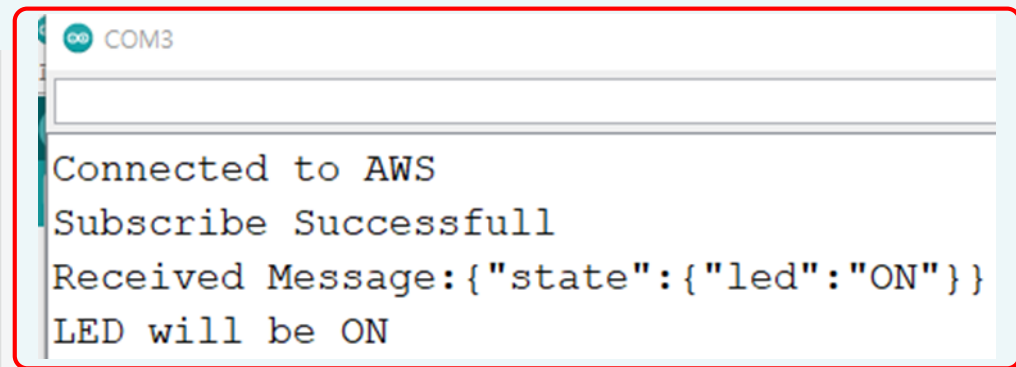
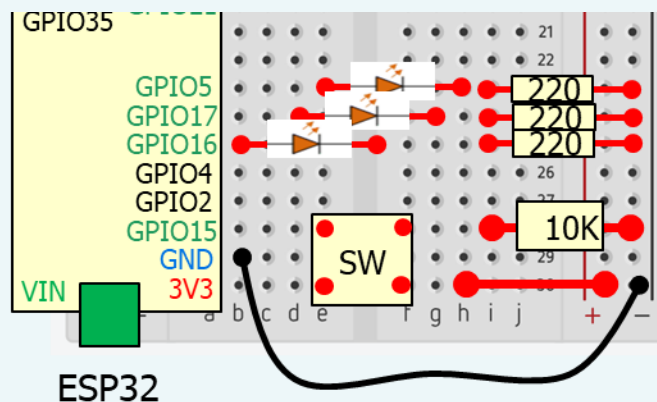
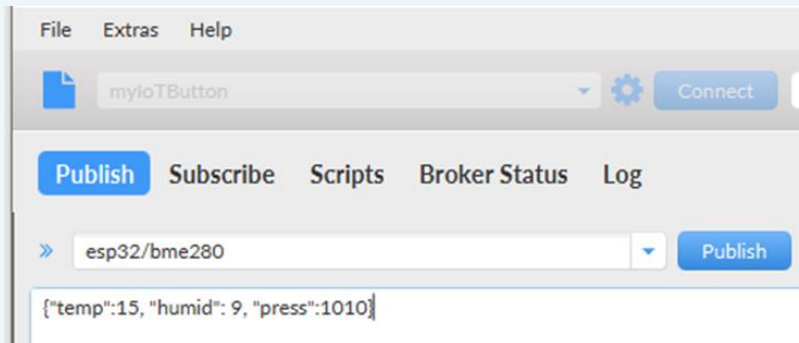
- Topic Setup:

Publish: [esp32/bme280](#) {"temp":n, "humid":n, "press":n}

Subscribe: [esp32/led](#) {"state": {"led": "ON" | "OFF"}}

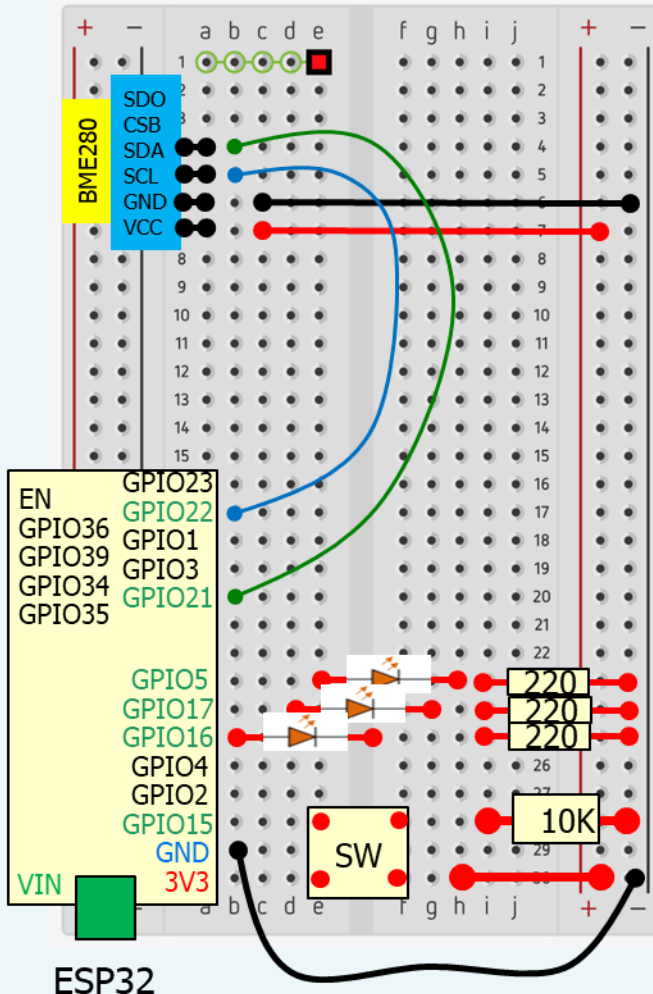
# Task09- Step A

- Make thing (ESP32\_led)
  - Publish 'esp32/bme280' by using MQTT-fx
  - Rule-Engine will publish 'esp32/led'
  - Your ESP32 should receive 'esp32/led' topic and control led



# Task09- Step C

- Make thing (ESP32\_bme280\_led)
  - Connect BME280 onto your ESP32 with I2C



- Publish 'esp32/bme280'

{“temp”:n, “humid”:n, “press”:n}

\*\* Publish period : 10 ~ 20 sec

or every time button pressed

- Rule-Engine will send email,  
publish 'esp32/led' on condition of Step-A
- Your ESP32 should change LED or  
send mail according to temp, humid..  
( Modify condition of your Rule-Engine  
for test )

ESP32