

Mosquitto (MQTT Broker)

→ <https://de.wikipedia.org/wiki/MQTT>

Installation

Broker („Server“): `sudo apt install mosquitto`

Client: `sudo apt install mosquitto-clients`

Mosquitto-Server starten:

`sudo /etc/init.d/mosquitto start`

und/oder

`systemctl enable mosquitto`

Test (ohne Verschlüsselung/Auth.)

→ <https://smarthome-blogger.de/tutorial/mqtt-raspberry-pi-einfuehrung/>

`mosquitto_sub -d -h localhost -t test`

`mosquitto_pub -d -h localhost -t test -m "Hello World"`

SSL/TLS Support (manpage, kurz)

→ <https://mosquitto.org/man/mosquitto-tls-7.html>

SSL Verschlüsselung benutzen mit selbstsigniertem Zertifikat

`openssl req -new -x509 -extensions v3_ca -out server.cert -keyout server.key -days 6500 -nodes`

→ Erzeugt das Zertifikat (selbstsignierter Public Key mit Informationen) als „server.cert“ und den dazu passenden geheimen Schlüssel (mit dem auch signiert wurde) als „server.key“.

Die Einstellungen in openssl.conf sollten in den AlternateNames alle Hostnamen und ggf. IP-Adressen des Servers enthalten, sonst KANN der Client den Verbindungsaufbau zurückweisen, wenn er nicht den angegebenen Hostname (-h) im Zertifikat sieht.

server.cert kann in diesem Beispiel auch als CA-Cert (zum Verifizieren der Signatur des Server-Certs) verwendet werden, wenn die entsprechenden Extensions in openssl.conf gesetzt sind.

/etc/mosquitto/mosquitto.conf wie folgt ergänzen:

```
# Normaler (unverschlüsselter) Port
port 1883

# Verschlüsselter Port (VOR cafile etc.)
listener 8883

# SSL Verschlüsselung: Zert. und Key
# Hier optional:
# cafile /home/pi/server.cert
keyfile /home/pi/server.key
certfile /home/pi/server.cert
```

Kontrolle: Zertifikat anschauen:

```
openssl x509 -in server.cert -text
```

Mosquitto-Dienst neu starten:

```
sudo /etc/init.d/mosquitto restart
```

Test der verschlüsselten Verbindung:

```
openssl s_client -connect mosquitto.pi:8883
```

→ Es sollte das Server-Zertifikat ausgegeben werden, das der mosquitto-Server schickt.

Benutzen der Verschlüsselung: Die Clients müssen, wenn oben cafile angegeben war, das Server-Zertifikat (hier identisch mit dem Server-Zertifikat) zum Verifizieren mit --cafile angeben:

```
mosquitto_sub -h mosquitto.pi -t test -p 8883
```

```
mosquitto_pub -h mosquitto.pi -t test -m "Hallo, Welt" -p 8883
```

Um ALLE Messages auf ALLEN Channels zu sehen, kann „#“ als Name verwendet werden. Unter Linux muss ein \ davor geschrieben werden (sonst wird # als Kommentar interpretiert).

```
mosquitto_sub -h 10.0.0.1 -t \#
```

Steckdose soll sich mit dem mosquitto-Server verbinden

In der Konfiguration der Steckdose den MQTT-Broker eintragen (hier: 10.0.0.1).

Wenn man den Server mit Zugangsdaten schützen will, kann in der `mosquitto.conf` eingetragen werde:

```
allow_anonymous false
```

```
password_file /home/pi/mosquitto-pass.txt
```

Die Datei `/home/pi/mosquitto-pass.txt` kann per

```
mosquitto_passwd /home/pi/mosquitto-pass.txt username password
```

mit User/Login-Paaren gefüllt werden.

Die Kommandos ändern sich dann dementsprechend, dass `-u username -P password` mit angegeben werden muss bei `mosquitto_pub` und `mosquitto_sub`.

Server: `mosquitto_passwd -b /home/pi/mosquitto-pass.txt username password`

Client:

```
mosquitto_sub -u gast -P gast -h 10.0.0.1 -t \#
```

OBI Socket 2 Modul

steckdose

Die Steckdose muss entsprechend auch einen erlaubten User/Passwort in der Konfiguration eintragen:

MQTT-Einstellungen	
Host ()	<input type="text" value="10.0.0.1"/>
Port (1883)	<input type="text" value="1883"/>
client (DVES_4863D6D6)	<input type="text" value="DVES_%06X"/>
Benutzer (DVES_USER)	<input type="text" value="steckdose"/>
Passwort	<input type="password" value="...."/>
topic = %topic% (sonoff)	<input type="text" value="test"/>
full topic (%prefix%/topic%/)	<input type="text" value="%prefix%/topic%"/>
<input type="button" value="Speichern"/>	

SSL Authentifizierung einschalten

→ <https://mosquitto.org/man/mosquitto-conf-5.html> (s. `require_certificate`)

„anonymous false“ und die Passwort-Date wieder aus der mosquitto-conf herausnehmen, dafür folgendes hinein:

```
# Clients NUR mit Zertifikaten erlauben
require_certificate true

# Alle Zertifikate, die mit dem Server-Key (!) signiert wurden,
werden akzeptiert
cafile /home/pi/server.cert
```

Für jeden Client ist nun ein Certificate Request auszustellen (Schlüsselpaar OHNE Signatur!):

```
openssl req -new -out client.cert -keyout client.key -nodes -days 3650
```

und dieser muss von dem CA-Zertifikat, das dem Server bekannt ist, unterschrieben werden.

```
openssl ca -keyfile server.key -cert server.cert -in client.req -out client.cert
-create_serial
```

Wichtig: CA und Client müssen den gleichen Organisationsnamen tragen (sonst braucht es noch ein übergeordnetes Zertifikat als Signierer!).

Nun muss der Client sein Zertifikat und den dazu passenden Key installieren, um sich zu authentifizieren. Ab sofort müssen ALLE Clients sich per SSL authentifizieren, und auch das Server-Zertifikat kennen (3 SSL-Dateien zu kopieren):

```
mosquitto_sub --cafile server.cert \  
  --cert client.cert \  
  --key client.key \  
  -h 10.0.0.1 -p 8883 -t \#
```

```
mosquitto_pub --cafile server.cert \  
  --cert client.cert \  
  --key client.key \  
  -h 10.0.0.1 -p 8883 -t test -m message
```