MQTT-Grunlagen-Kurs

Version 3.0 (2022/11)

Dieses Dokument soll den Video-Kurs *MQTT-Grundlagen* begleiten und Dich dabei unterstützen, das gelernte zu verinnerlichen und als Nachschlagewerk der Inhalte dienen. **Das Lesen von diesem Dokument ersetzt nicht das Anschauen der Videos des Kurses.**

Einleitung

MQTT steht für *Message Queuing Telemetry Transport* und ist ein Nachrichtenprotokoll zur Kommunikation zwischen verschiedenen Geräten. Der Begriff “Gerät” ist dabei natürlich sehr allgemein - und so ist es auch. Man kann (fast) jedes Gerät, welches Teilnehmer im Netzwerk (LAN) ist, und jede Programmiersprache überzeugen MQTT zu verstehen bzw. zu sprechen.

Da MQTT ein Netzwerkprotokoll ist, muss natürlich eine Netzwerkverbindung am Gerät zur Verfügung stehen - also LAN oder WLAN.

Grundlagen

In der “Mitte” steht ein Server zur Verfügung - der sogenannte “MQTT-Broker”. An diesen melden sich alle Geräte an - die Clients.

Jetzt sind schon die grundlegenden Vorraussetzungen geschaffen um mit MQTT zu starten: Jeder Client kann Nachrichten versenden (publish), welche von allen anderen Clients empfangen werden KÖNNEN. Ich schreibe bewusst können, da nicht jeder Client automatisch auch jede Nachricht bekommt. Um eine Nachricht erhalten zu können, muss der jeweilige Client das Topic (Thema) der Nachricht abonnieren (subscribe). Dies ist so gelöst worden, da auf diesem Wege der Broker nicht alle Nachrichten an jeden Client zustellen muss - so wird die Netzwerklast gering gehalten und die Clients müssen auch nicht tausende von Nachrichten auswerten, welche sie eigentlich gar nicht interessieren. Es ist also **kein** klassischer “Broadcast”, sondern **eine sehr gezielte Zustellung der relevanten Nachrichten an jeden einzelnen Client.**

Natürlich könnte man auch einfach ALLES abonnieren. Aber dazu später noch ein wenig mehr. Das macht aber nur in den wenigsten Fällen Sinn.

Nachrichten

Die angesprochenen Nachrichten (Messages) müssen dabei immer an ein definiertes Thema (Topic) gesendet werden. Das ist praktisch die “Adresse” der Nachricht. Klingt erstmal kompliziert, ist aber super einfach umgesetzt. Diese Topics sind auch einfache Zeichenketten, welche man durch Schrägstrich (Slash) trennen kann um eine Hierarchie aufzubauen. Hier mal einige Beispiele:

```
SmartHome/Haus/Keller/Waschmaschine
SmartHome/Haus/Keller/Trockner
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Stehlampe
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Fernseher
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Heizung
SmartHome/Haus/Erdgeschoss/Kueche/Spuelmaschine
SmartHome/Haus/Erdgeschoss/Kueche/Deckenlicht
SmartHome/GartenHaus/EingangsTuer
SmartHome/GartenHaus/Deckenlicht
```

Wie man sieht, bildet man einzelne Gruppen ab. Das Ganze könnte man mit einer Verzeichnisstruktur auf dem PC/Computer vergleichen. Natürlich ist das nicht unbedingt notwendig um MQTT zu betreiben, aber in jedem Fall sinnvoll. Genauso sinnvoll, wie es auch auf dem Computer ist, nicht alle Urlaubsfotos in ein Verzeichnis abzulegen, sondern nach Land, Ort und/oder Jahr zu sortieren um Sachen schnell wieder zu finden.

Hier sollte man sich also eine ordentliche Struktur überlegen, welche man später auch noch gut erweitern kann. Lieber eine Ebene mehr einbauen, als eine zu wenig. Denn die stärken des Protokolls liegen in sogenannten Wildcards - also Platzhaltern. Nehme ich also das oben gennante Beispiel, könnte man ganz einfach alle Themen aus dem Wohnzimmer abonnieren, indem man dem Broker mitteilt, dass man folgende Infos gerne zugestellt hätte:

```
SmartHome/Haus/Erdgeschoss/#
```

Die Raute (#) am Ende ist ein Platzhalter und sagt aus, dass ich gerne Nachrichten hätte, die mit diesem Topic **beginnen**. Also alles aus dem Erdgeschoss. So bekomme ich alle Nachrichten aus der Küche und auch aus dem Wohnzimmer zu allen darin befindlichen Geräten zugestellt. Logisch, oder? So spart man sich eine Menge Zeit und Schreibarbeit. Außerdem können mehr Geräte dazu kommen und

man muss z.B. ein Dashboard dann nicht noch einmal anpassen wenn neue Topics dazu kommen.

Dieses Beispiel würde also die Nachrichten der folgenden Topics empfangen:

```
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Stehlampe
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Fernseher
SmartHome/Haus/Erdgeschoss/Wohnzimmer/Heizung
SmartHome/Haus/Erdgeschoss/Kueche/Spuelmaschine
SmartHome/Haus/Erdgeschoss/Kueche/Deckenlicht
```

Ansonsten gibt es auch noch das Plus-Zeichen als Platzhalter. Dieses sagt aus, dass **nur eine Ebene mit einbezogen werden soll** (und nicht die darunter folgenden auch noch). Ich persönlich nutze dies aber wirklich extrem selten.

```
SmartHome/Haus/Erdgeschoss/+
```

Dieses Beispiel würde also die direkten Nachrichten darunter empfangen. Also zum Beispiel:

```
SmartHome/Haus/Erdgeschoss/Wohnzimmer
SmartHome/Haus/Erdgeschoss/Kueche
SmartHome/Haus/Erdgeschoss/Temperatur
```

Da ich in meinen Szenarien aber immer nur an die “Endpunkte” der Hierarchien Nachrichten sende, macht dieses Konstrukt nur in bestimmten Situationen Sinn. Je nachdem, wie man das Ganze strukturiert hat.

Tipps

Ich würde generell empfehlen, die **Struktur zu dokumentieren.** Sonst fügt man in einem halben Jahr ein neues Gerät hinzu und weiß nicht mehr ob man *Wohnzimmer* oder *WZ* als Topic genommen hat. So beugt man Missverständnissen und fehlgeleiteten Nachrichten direkt vor. Schließlich muss man die Topics nirgends registrieren, sondern kann einfach drauf lossenden. Das ist der einzige Knackpunkt von MQTT.

Natürlich gibt es Tools, welche automatisch dokumentieren, welche “Topics” in Benutzung sind. Aber dafür müssen auch erstmal Nachrichten auf diesen gesendet worden sein. Also: Von Anfang an Gedanken machen und alles dokumentieren, dann hat man am Ende viel weniger Stress!

Voraussetzungen

Genug der Worte - jetzt setzen wir das Ganze einmal in der Praxis um. Also ab an den Raspberry Pi und losgelegt.

Dieses Tutorial sollte auch auf jedem anderen System mit Debian oder Ubuntu funktionieren - ich habe es aber auf dem Raspberry Pi 4 dokumentiert, da es dort sicher am meisten von meinen Zuschauern und Lesern installiert wird.

Als Basis habe ich also einen Raspberry Pi 4 mit Raspberry Pi OS genutzt.

Raspberry Pi OS kann man hier herunterladen:

https://www.raspberrypi.org/software/

Da wir einen Server betreiben und keinen Desktop-Arbeitsplatz, würde ich die **Version ohne grafische Oberfläche empfehlen (also die 64-Bit Lite-Version).**

Wie genau man das Image aufspielt, findet man tausendfach im Internet. Genauso auf meinem YouTube-Kanal oder in meinem Online-Shop. Darauf möchte ich hier auch nicht näher eingehen.

Alle gezeigten Befehle und Aktionen führe ich als User “pi” aus, welchen ich während der Einrichtung des Images angelegt habe (dieser wird in den aktuellen Raspbian-Versionen nicht mehr im Standard erstellt). **Ich würde nie empfehlen als root zu arbeiten** - das macht meistens mehr kaputt als es hilft. Lieber gezielt über “sudo” die erforderlichen Rechte holen wenn man diese benötigt.

Das System sollte vor der Installation auf dem neuesten Stand sein. Dies erreichen wir durch das Absetzen der folgenden zwei Befehle:

```
sudo apt update
sudo apt full-upgrade
```

Installation von Mosquitto

Als Broker nutzen wir hier Mosquitto. Dies ist ein OpenSource-Projekt welches einen MQTT-Broker bereitstellt. Die ideale Basis für unser Vorhaben. Installiert wird dieser ganz einfach durch die Eingabe der folgenden Zeile:

```
sudo apt install mosquitto
```

Wenn Du wissen möchtest, welche Version Du von Mosquitto installiert bekommen hast, dann gib einfach folgendes ein:

```
mosquitto -h
```

In meinem Fall ist die **Version 2.0.11** installiert worden, das sehe ich an dieser Ausgabe (Zeile 1):

```
mosquitto version 2.0.11

mosquitto is an MQTT v5.0/v3.1.1/v3.1 broker.

Usage: mosquitto [-c config_file] [-d] [-h] [-p port]

 -c : specify the broker config file.
 -d : put the broker into the background after starting.
 -h : display this help.
 -p : start the broker listening on the specified port.
      Not recommended in conjunction with the -c option.
 -v : verbose mode - enable all logging types. This overrides
      any logging options given in the config file.

See https://mosquitto.org/ for more information.
```

Dent aktuellen Status des Prozesses (und eventuelle Fehler) bekommst Du mit folgendem Befehl heraus:

```
sudo service mosquitto status
```

MQTT Explorer

Weiterhin zeige ich Dir im Kurs viele Beispiele mit der Software *MQTT-Explorer*. Dies ist ein Desktop-Client, welcher sich ebenfalls mit dem MQTT-Broker verbinden kann. So ein Client ist ideal um zu schauen, ob und welche Nachrichten gerade so durch den Broker vermittelt werden. Den Download findest Du für alle gängigen Betriebssysteme unter der folgenden Adresse:

mqtt-explorer.com

Achtung: Ab Version 2.x von Mosquitto, kann man sich **NICHT** mehr einfach so (ohne Benutzerdaten) mit dem MQTT-Broker verbinden. Es ist zwingend notwendig, einen Benutzer anzulegen. Daher auf jeden Fall den nächsten Schritt durchführen - ansonsten wird die Verbindung abgelehnt!

Grundkonfiguration

Seit Version 2.x muss man einige Konfigurationen noch manuell vornehmen und Mosquitto lehnt alle eingehenden Verbindungen im Standard ab! Daher legen wir als nächstes eine neue Datei im dafür vorgesehenen Verzeichnis an. Wie diese Datei heißt, ist im Prinzip egal. Ich nenne sie einfach `custom.conf` .

```
sudo nano /etc/mosquitto/conf.d/custom.conf
```

Inhalt der neuen Datei:

```
listener 1883

allow_anonymous false
password_file /etc/mosquitto/mosquitto_passwords
```

Hier beschreiben wir zwei Dinge:

1. Wird Mosquitto mitgeteilt, auf dem MQTT-Standard-Port 1883 auf eingehende Verbindungen zu lauschen
2. Legen wir fest, dass kein anonymer Zugriff erlaubt ist. Die gültigen Benutzer und Passwörter sind in der Datei im hinterlegten Pfad zu finden. Diese Datei gibt es aber noch nicht, daher legen wir diese im nächsten Schritt an.

Absicherung mit Benutzername und Passwort

Um die Installation abzusichern, sollte man den MQTT-Broker mit Benutzername und Passwort schützen. Normalerweise solltest Du natürlich wissen, wer sich in einem privaten Netzwerk tummelt. Also auch ohne einen Passwort-Schutz wäre das Ganze sicher nicht so dramatisch - dennoch sollte man seine Dienste absichern (zumal sich der Aufwand dafür in Grenzen hält).

Dafür erstellen wir als erstes eine Datei, in welcher die Passwörter gespeichert werden sollen. Das Tool `mosquitto_passwd` hilft dabei. Allerdings wird ein Standard-Benutzername verlangt. Ich wähle `mkleine` und lasse die Datei erstellen:

```
sudo mosquitto_passwd -c /etc/mosquitto/mosquitto_passwords mkleine
```

Danach muss man 2x das Passwort für den Benutzer eingeben. Keine Sorge, die Eingaben sieht man nicht. Also fleißig tippen und Enter drücken, wenn das

Passwort eingeben wurde. Das ist überall unter Linux so, damit niemand zuschauen kann, wie lang das Passwort ist!

Ich wähle z.B. das zufällig generierte Passwort `9kfyrTtwqBzJqUvDAyqsDw` .

Damit die neuen Konfigurationsdateien geladen werden, muss der Dienst einmal neugestartet werden:

```
sudo service mosquitto restart
sudo service mosquitto status
```

Das wars auch schon. **Ab jetzt müssen sich ALLE Clients mit einem der Benutzer aus der neu angelegten Datei verbinden** - alle anderen Verbindungsanfragen werden direkt abgelehnt und es kann keine Verbindung aufgebaut werden. Dies können wir ganz einfach mit dem *MQTT Explorer* prüfen (wie im Video zu sehen).

FERTIG! Ab jetzt steht der MQTT-Broker für weitere Spielereien zur Verfügung!

Neue Benutzer hinzufügen / Benutzer löschen

Nachdem wir nun eine Datei angelegt haben, welche die Benutzer und Passwörter enthält, möchte man diese Liste wahrscheinlich irgendwann anpassen und erweitern.

Wenn Du nun einen neuen Benutzer hinzufügen möchtest, dann gib einfach die Passwort-Datei als Ziel an, wenn Du `mosquitto_passwd` ausführst (ohne irgendwelche Parameter!):

```
sudo mosquitto_passwd /etc/mosquitto/mosquitto_passwords tasmota
```

In diesem Beispiel fügen wir ein neuen Benutzer namens `tasmota` hinzu. Sobald Du diesen Befehl ausführst, wirst Du nach dem Passwort gefragt, welches Du dem Nutzer geben möchtest. Das wird dann mit Enter bestätigt und das Passwort zur Wiederholung noch ein zweites Mal eingegeben. Fertig.

Um das Passwort auch noch als Parameter mitzugeben, wird die Option `-b` verwendet. Dabei ist allerdings darauf zu achten, dass dieser Befehl dann komplett in der History landet und somit später einsehbar wäre:

```
sudo mosquitto_passwd -b /etc/mosquitto/mosquitto_passwords iobroker
```

Damit die Änderungen neu gelesen werden, muss Mosquitto neu gestartet werden. Das geht wie immer mit:

```
sudo service mosquitto restart
```

Um Benutzer zu löschen, kannst Du den Parameter `-D` (delete) verwenden. Gib einfach wieder den Pfad zur Passwort-Datei an und welcher Benutzer gelöscht werden soll. In diesem Beispiel lösche ich den `tasmota` Benutzer:

```
sudo mosquitto_passwd -D /etc/mosquitto/mosquitto_passwords tasmota
```

Ziemlich einfach, oder? Auch nach dieser Änderung muss der Dienst natürlich neu gestartet werden, damit die Benutzerliste erneut gelesen wird:

```
sudo service mosquitto restart
```

Pub- und Sub-Client

Wie schon angesprochen kann man aus nahezu jeder Programmiersprache mit dem Broker sprechen und Nachrichten senden und empfangen. Die hier gezeigten Schritte dienen nur zur Veranschaulichung dieser Möglichkeiten. Für den Betrieb des Brokers sind diese Schritte nicht relevant.

Als erstes installieren wir also die Client-Programme für MQTT unter Linux. Dann können wir von der Kommandozeile aus Nachrichten versenden und empfangen. Klingt cool? Ist es auch.

```
sudo apt install mosquitto-clients
```

Natürlich könnten wir diese Client-Programme auch auf jedem anderen Linux-System installieren. Also auch dort, wo der Broker nicht läuft. Wir verbinden uns dann eben nur zum Broker und setzen Nachrichten ab. So könnte man z.B. direkt von einem Raspberry Pi Daten von 1Wire-Sensoren abfragen und diese dann an den Broker senden. Alles mit wenigen Zeilen Code in einem Cron-Job.

Zum Veröffentlichen einer Nachricht kann man nun also das Programm mosquitto_pub verwenden. Möchte man Themen abonnieren und Nachrichten erhalten, ist das Programm mosquitto_sub zur Stelle.

```
mosquitto_pub -h localhost -t Bash/Test -m "Das ist ein Test" -u mkle
```

In diesem Beispiel senden wir nun also den Text “Das ist ein Test” an das Thema (Topic) “/Bash/Test”. Ob das klappt können wir natürlich mit dem oben genannten *MQTT Explorer* nachvollziehen.

Abonnieren von Themen geht genauso einfach.

```
mosquitto_sub -h localhost -t Bash/Test -u mkleine -P 9kfyrTtwqBzJqUv
```

Hier sagen wir einfach nur, welches Thema wir gerne hätten und übergeben Benutzername und Passwort. Das wars. Ab jetzt wartet das Programm auf eingehende Nachrichten. Wenn wir nun also dem *MQTT-Explorer* eine Nachricht an das Topic senden, können wir diese in der Shell sehen. Aber das zeige ich im Video auch genauer.

Wildcards

Um alle Topics in einem bestimmten Bereich zu abonnieren, kann beispielsweise mit der Raute `#` gearbeitet werden. Abonniere ich zum Beispiel `Bash/#`, bekomme ich die Nachrichten an alle Topics zugestellt, welche mit `Bash/` anfangen! Also auch `Bash/Kekse/Test/Hello`. Das kann in vielen Bereichen sehr hilfreich sein!

Möchte man ALLE Nachrichten vom Broker bekommen, welche jemals von irgend einem Client gesendet werden, kann man als Client auch einfach nur `#` abonnieren.

Ende

Das wars. Ab jetzt können wir also Nachrichten an unseren neuen MQTT-Broker senden und Geräte integrieren. Und wie schon oft erwähnt, könnte dies so gut wie alles sein. Also zum Beispiel auch ein Arduino an einer Powerbank, welcher die Temperatur des Kellers übermittelt. Ziemlich cool, oder?

Ich hoffe Du konntest Deinen MQTT-Broker erfolgreich installieren und mit einem Benutzernamen und Passwort absichern. Damit hast Du jetzt auf jeden Fall eine großartige Möglichkeit für die Kommunikation zwischen allen möglichen Geräten in deinem Netzwerk gelegt.

Viel Spaß bei der Umsetzung - lass mich gerne wissen ob Dir noch Themen fehlen

die den Kurs erweitern. Es gibt außerdem noch Bonus-Lektionen, welche nicht in diesem Dokument ausgeführt werden. Dazu einfach auf den in der Mail genannten Link klicken und diese auch noch genießen.

Bis dann,

Matthias Kleine

Trainer und Softwareentwickler

Weitere Inhalte

Das hier war natürlich nur der Anfang. Wenn Du tiefer in die einzelnen Systeme einsteigen möchtest, empfehle ich dir einen Kurs zu buchen.

Weiterhin würde mich sehr freuen, wenn Du mir auf den nachfolgenden Kanälen folgen würdest, damit ich Dich mit noch mehr hilfreichen Informationen wie diesen hier versorgen kann.

- YouTube: https://www.youtube.com/c/Hausautomatisierung-com/
- Twitter: https://twitter.com/haus_automation
- Facebook: https://www.facebook.com/HausAutomatisierungCom/
- Instagram: https://www.instagram.com/haus_automation/
- Patreon: https://www.patreon.com/haus_automation

Vielen Dank fürs Lesen bis hier hin. Weil Du das getan hast, lade ich Dich auch gerne in meine Insider-Gruppe auf Facebook ein. Diese findest Du unter diesem Link:

https://www.facebook.com/groups/HausAutomatisierungCom.Insider/

In der Gruppe tauschen sich extrem viele Smart-Home-Fans über alle möglichen Themen rund um ihre Probleme, Lösungen und Projekte aus. Du bist herzlich eingeladen, diesem exklusiven Club beizutreten.

Jetzt aber: Bis bald!