MQTT-Grunlagen-Kurs

Stand 02/2018

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

Einleitung

MQTT steht für "Message Queuing Telemetry Transport" und ist ein Nachrichtenprotokoll zur Kommunikation zwischen verschiedenen Geräten. Geräte ist dabei natürlich sehr allgemein gehalten - und das ist es auch. Man kann (fast) jedes Gerät, welches im Netzwerk ist, und jede Programmiersprache überzeugen MQTT zu verstehen.

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

Grundlagen

In der Mitte des Geschehens steht ein Server zur Verfügung - der sogenannte "Broker". An diesen melden sich alle Geräte an - die Clients.

Jetzt sind schon die grundlegenden Vorraussetzungen geschaffen um zu starten: Jeder Client kann Nachrichten versenden, 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 man das Topic (Thema) der Nachricht abonnieren (subscribe). Dies ist so gelöst, da auf diesem Wege der Broker nicht alle Nachrichten 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 angedacht.

Natürlich könnte man auch einfach ALLES abonnieren. Aber dazu später noch ein wenig mehr. Außerdem macht das 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 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 gliedern 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 beispielsweise 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.

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 irgendwo zu dokumentieren. Sonst fügt man in einem halben Jahr ein neues Gerät hinzu und weiß nicht mehr ob man Wohnzimmer oder Whz als Topic genommen hat. So beugt man Misverständnissen 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: Direkt richtig Gedanken machen und dokumentieren, dann hat man am Ende viel weniger Stress!

Installation

Genug der Worte - jetzt setzen wir das Ganze einmal in die Realität um. Also ab an den Raspberry Pi und losgelegt. Das Tutorial sollte auch auf jedem anderen System mit Debian funktionieren - ich habe es aber auf dem Raspberry Pi dokumentiert, da es dort sicher am meisten von meinen Zuschauern und Lesern installiert wird.

Als Basis habe ich also einen Raspberry Pi mit Raspbian (Debian Stretch Lite - Version November 2017) genutzt. In älteren Versionen (z.B. Jessie) funktioniert das genauso. Für die kommenden Versionen kann ich Stand heute natürlich noch nicht sprechen - aber viel wird sich dort auch nicht ändern. Wie auch bei Stretch kann es eben nur einige Zeit dauern bis die Installation reibungslos funktioniert. Das war bei Stretch bis vor kurzem auch nur über Umwege möglich - jetzt klappt aber alles einwandfrei.

Raspbian kann man hier herunterladen:

https://www.raspberrypi.org/downloads/raspbian/

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

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

Danach habe ich das Image ein wenig vorkonfiguriert. Also die Locale auf Deutschland umgestellt und auch die Zeitzonen. Auch das wurde schon hundertfach gezeigt und würde hier nur unnütz den Rahmen sprengen.

Alle gezeigten Befehle und Aktionen führe ich als User "pi" aus. Also dem Standard-Benutzer unter Raspbian. Ich würde nicht empfehlen direkt 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-get update
sudo apt-get upgrade
```

Außerdem installiere ich immer noch ein paar weitere Tools, welche mir die Arbeit auf dem Raspberry Pi erleichtern.

```
sudo apt-get install vim wget git
```

MQTT.fx

Weiterhin zeige ich im Video viel mit der Software MQTT.fx. Dies ist ein Desktop-Client, welcher sich ebenfalls mit dem Broker verbinden kann. Das ist ideal um zu schauen ob und welche Nachrichten gerade so durch den Broker gejagt werden. Den Download findest Du für alle gängigen Betriebssysteme unter der folgenden Adresse:

http://mqttfx.jensd.de/index.php/download

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-get install mosquitto
```

Absicherung mit Benutzername und Passwort

Um die Installation abzusichern, sollte man den 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 welche wir in jede Zeile einen Benutzername und ein Passwort schreiben. Getrennt wird der Benutzername vom Passwort durch einen Doppelpunkt.

```
cd ~
vi mosquitto_passwords
```

Inhalt wäre dann folgender (aus dem Video):

```
mkleine:Test1234
foo:bar
```

In diesem Beispiel existieren zwei Benutzer. Der Erste heißt mkleine und hat das Passwort Test1234, der Zweite hat als Benutzernamen foo und als Passwort bar. Bitte wähle entsprechend komplexe Passwörter - ich würde dennoch auf Sonderzeichen verzichten um Konflikte zu vermeiden. Also einfach eine längere Kette aus Buchstaben und Zahlen. z.B. 5yjZnH28kc7zhXdMJ oder 6cjBETzUsFSFbCRrr

Bitte die Passwörter notieren, da Du diese dann gleich nicht mehr im Klartext lesen können wirst.

Nun speichern wir diese Datei (ESC -> :wq -> ENTER unter vim). Nun müssen wir die Passwörter noch "hashen". Ein Hash ist eine Zeichenkette, welche nicht zurückgerechnet werden kann - ist also keine Verschlüsselung. Mehr dazu auf Wikipedia. Dies erreichen wir mit dem Befehl:

```
mosquitto_passwd -U mosquitto_passwords
```

Der Wert für den Parameter -U ist natürlich der Dateiname unserer gerade erstellten Datei. Wie diese heißt ist eigentlich egal, aber so weiß man wenigstens was darin enthalten ist.

Ob der Prozess geklappt hat, prüfen wir mit einem einfach "cat". Dieses Programm gibt unter Linux den Inhalt der übergebenen Datei einfach aus.

```
cat mosquitto_passwords
```

Jetzt sollten die zuvor eingegeben Passwörter nicht mehr im Klartext zu lesen sein (wie im Video gezeigt).

Als nächstes müssen wir Mosquitto nur noch mitteilen, dass diese Benutzer und Passwörter auch verwendet/geprüft werden. Bis hierhin haben wir einfach nur eine Datei mit ein wenig Text irgendwo im Dateisystem erstellt. Das heißt, dass Mosquitto noch nichts von der Existenz dieser Datei weiß. Dies ändern wir jetzt in den folgenden Schritten:

```
sudo mv mosquitto_passwords /etc/mosquitto/
cd /etc/mosquitto/
sudo chown root:root mosquitto_passwords
```

An der Situation von gerade hat sich jetzt noch nicht so viel geändert. Nun liegt die unbekannte Datei an einer anderen Stelle im System. Also teilen wir Mosquitto nun mit, wo er nach einer Passwort-Datei schauen soll.

```
cd conf.d/
sudo vi access.conf
```

Wie genau die Datei heißt ist ziemlich egal. Hauptsache sie liegt in diesem Verzeichnis, gehört root und endet mit ".conf". Dies wird durch den im Video gezeigten Include realisiert. Also werden einfach alle .conf-Dateien in diesem Verzeichnis geladen und berücksichtigt. Daher ist der Dateiname an dieser Stelle auch egal.

Inhalt der Datei ist folgender:

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

Die Datei wird dann gespeichert (ESC -> :wq -> ENTER unter vim) und Mosquitto muss einmal neugestartet werden, damit die Änderungen wirksam werden.

```
sudo service mosquitto restart
```

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 einfacht mit MQTT.fx prüfen (wie im Video zu sehen).

FERTIG! Ab jetzt steht unser Broker für weitere Spielereien zur Verfügung. Die Absicherung per Passwort sollte man am besten direkt nach der Installation realisieren. Stellt man diese erst später ein, muss man umständlich auf allen Clients diese Benutzerdaten nachreichen. Und das kann sehr anstrengend werden.

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-Bibliotheken für MQTT unter Linux. Dann können wir von der Bash aus Nachrichten absetzen und auch empfangen. Klingt cool? Ist es auch.

```
sudo apt-get 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 mkleine -P Test1
234
```

In diesem Beispiel senden wir nun also den Text "Das ist ein Test" an das Thema "/Bash/Test". Ob das klappt können wir natürlich wie immer mit dem oben genannten MQTT.fx nachvollziehen (wenn wir das Thema dort abonniert haben).

Abonnieren von Themen geht genauso einfach.

```
mosquitto_sub -h localhost -t /Bash/Test -u mkleine -P Test1234
```

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 aus MQTT.fx eine Nachricht an das Topic senden, können wir diese in der Shell sehen. Aber das zeige ich im Video auch genauer.

Ende

Das wars. Ab jetzt können wir also Nachrichten an unseren Broker senden. Und wie schon oft erwähnt könnte dies so gut wie alles sein. Also zum Beispiel auch ein Arduino an einer Powerbank, welcher die Temperatur im Keller übermitteln möchte. 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

Ich 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 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!