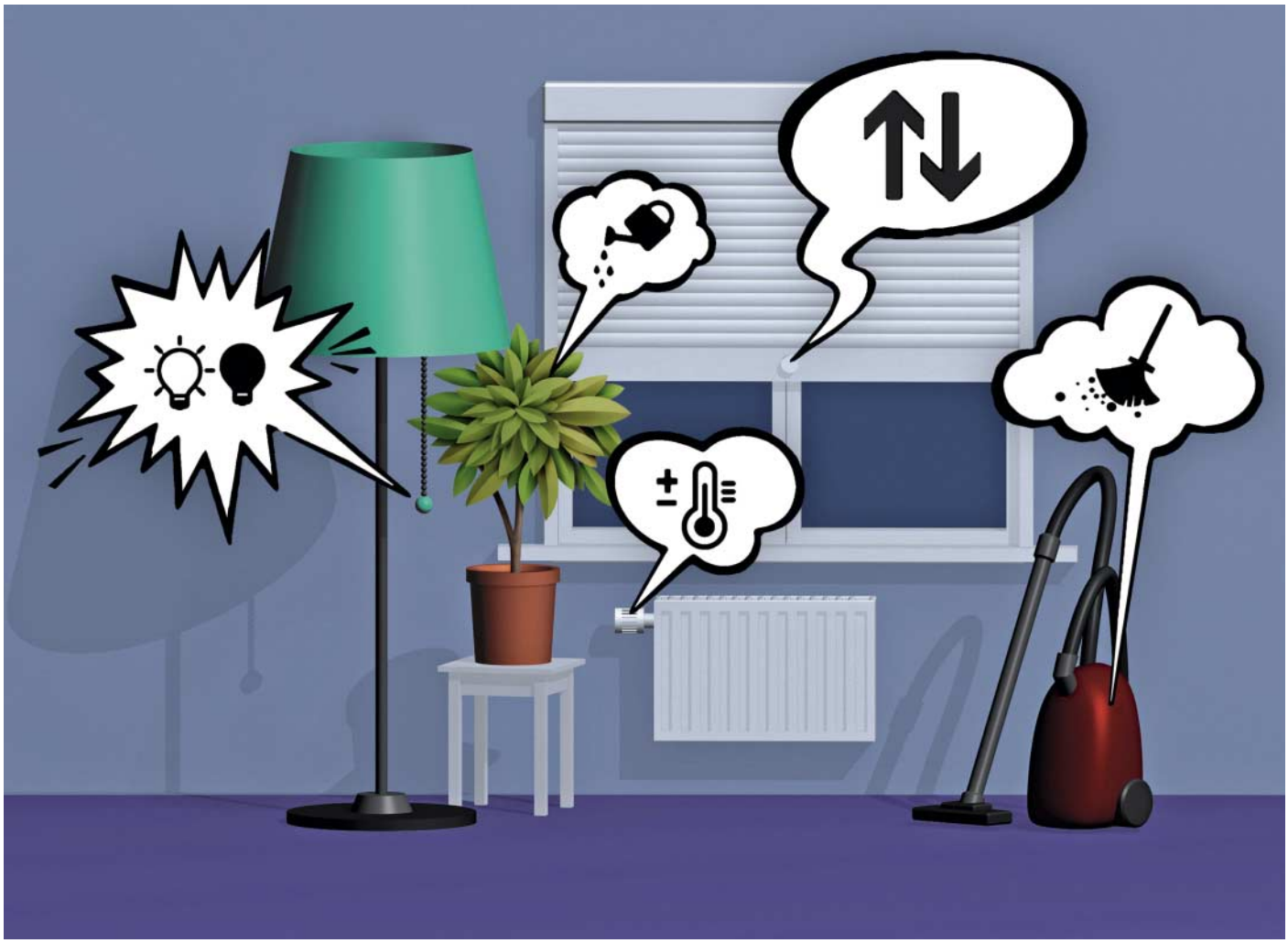


Bild: Jörg Niehage

Weltsprache

Das Protokoll MQTT für robusten Datenaustausch in Industrie und Hausautomation

Wenn Sensoren, Aktoren und Maschinen miteinander kommunizieren sollen, müssen sie eine gemeinsame Sprache sprechen. MQTT eignet sich für Industrieanlagen und das vernetzte Zuhause gleichermaßen und ist robust genug für unzuverlässige Verbindungen.

Von Jan Mahn

Ein Protokoll, das den Austausch zwischen verteilten Geräten regeln soll, muss vor allem mit Problemen umgehen können: Die Kommunikationspartner können aus unterschiedlichen Gründen nicht erreichbar sein, Nachrichten verloren gehen und Verbindungen während der Übermittlung abbrechen. MQTT (Message Queue Telemetry Transport) setzt auf TCP/IP auf und versucht, die Widrigkeiten der unzuverlässigen Verbindungswege durch einen zentralen Vermittler zu lösen: den MQTT-Broker. Dieser ist Schnittstelle für alle übertragenen Nachrichten. Die beteiligten Geräte unterhalten sich nur mit ihm und kennen sich untereinander nicht. Sie müssen weder die IP-Adressen noch technische Details anderer Teilnehmer kennen. Der Broker hat die Aufgabe, Nachrichten anzunehmen und an Interessierte weiterzugeben.

Der große Koordinator

Soll ein Sensor wie beispielsweise ein Mikrocontroller mit Temperaturfühler an einer MQTT-Kommunikation teilnehmen, muss er zunächst eine Verbindung mit dem Broker herstellen. Reserviert sind für MQTT die Ports 1883 für unverschlüsselte und 8883 für verschlüsselte Übertragung. MQTT ist, anders als zum Beispiel HTTP, ein zustandshaltendes Protokoll. Die Verbindung kann also auch dann bestehen bleiben, wenn keine Daten über-

tragen werden. Möchte der Sensor einen Temperaturwert übermitteln, sendet er eine Nachricht vom Typ `PUBLISH`. Jede Nachricht enthält ein möglichst beschreibendes Topic und einen Inhalt, die Payload. Das Topic ist ähnlich aufgebaut wie ein Unix-Dateipfad. Die Abschnitte werden durch `/` getrennt. Bei der Gestaltung kann sich der Betreiber der MQTT-Umgebung frei entfalten. Im vernetzten Zuhause könnte der Sensor das Topic `house/rooms/wc/sensors/temperature` mit dem Wert `"22.5"` an den Broker melden. Er hat damit seine Aufgabe erfüllt und muss sich keine Gedanken darüber machen, welche Geräte sich für diese Information interessieren könnten.

Ein Gerät, das Nachrichten empfangen möchte, verbindet sich mit dem Broker und bestellt mit dem Befehl `SUBSCRIBE` ein Abo für ein oder mehrere Topics. Um mehrere Werte einzusammeln, gibt es zwei Platzhalter (die daher nicht im Namen von Topics vorkommen dürfen): Mit `#` werden alle Nachrichten auf den niedrigeren Hierarchieebenen bestellt, das Zeichen darf daher nur am Ende stehen: `house/rooms/wc/#` abonniert alle Nachrichten, die das Badezimmer betreffen. Ein `+` dient als Platzhalter für eine Hierarchieebene: `house/rooms/+/sensors/temperature` abonniert alle Temperaturwerte. So könnte zum Beispiel der MQTT-fähige Heizkörperthermostat alle Sensorwerte des Hauses anfordern und darauf reagieren. Der Broker speichert die Abos ab und verteilt eingehende Nachrichten sofort an alle Abonnenten, die momentan eine Verbindung halten. Kommt ein Abonnent erst später dazu, würde er diese Information verpassen. Der Sender kann daher bei der Veröffentlichung das Retain-Flag setzen. Dieses zeigt an, dass diese Nachricht allen Abonnenten direkt nach dem `SUBSCRIBE` zugestellt werden soll. Der Broker speichert in diesem Fall den letzten Wert für das Topic zwischen und übermittelt diesen (und nicht etwa den gesamten Nachrichtenverlauf aus der Vergangenheit).

Drei-Klassen-System

MQTT enthält einen Mechanismus für Quality of Service. Dabei geht es jedoch nicht darum, Nachrichten vor anderen zu priorisieren, sondern um die Art der Empfangsbestätigung. Jede versendete Nachricht bekommt ein QoS-Level mit auf den Weg. Stufe 0 sorgt dafür, dass eine Nachricht ohne Bestätigung einmalig verschickt wird. Das geht schnell und ist ressourcenschonend. Für einfache Sensorwerte ist das ausreichend, aber nicht empfehlenswert, wenn von der Information der Betrieb einer Industrieanlage abhängt und die Netzwerkverbindung brüchig sein kann.

Eine Nachricht mit dem QoS-Wert 1 kommt mindestens einmal beim Empfänger an. Als Antwort auf ein `PUBLISH` sendet der Empfänger `PUBACK`. Bleibt diese Reaktion aus, versucht es der Sender erneut – bis der Empfang bestätigt wurde. Bei diesem Verfahren kann es vorkommen, dass eine Nachricht mehrmals ankommt, wenn die Bestätigung verloren geht. Der Entwickler muss sicherstellen, dass dadurch kein Schaden angerichtet werden kann. Bekommt ein Roboterarm unbeabsichtigt mehrmals den Auftrag, sich ein Stück geradeaus zu bewegen, kann das unangenehme Folgen haben.

Aufwendiger und dafür zuverlässiger ist QoS auf Stufe 2. Hier sorgen beide Gesprächspartner dafür, dass eine Nachricht nur genau einmal beim Gegenüber ankommt. Der Sender A verschickt per `PUBLISH` eine Nachricht mit einer Nachrichten-ID. Der Empfänger B bestätigt das per `PUBREC` und speichert die Nachricht erst einmal zwischen. Hat der Sender A die Bestätigung erhalten, sendet er `PUBREL` zurück, kann die Nachricht bedenkenlos löschen und wird sie unter keinen Umständen erneut senden. Er kann sich sicher sein, dass sein Gegenüber B die Nachricht jetzt kennt. Der Empfänger B sendet als letzte Mitteilung `PUBCOMP` an A. Erst jetzt beginnt B, die Nachricht zu verarbeiten. Handelt es sich bei B um einen Broker, würde er erst nach dem Versenden von `PUBCOMP` die Nachricht an die Abonnenten versenden. Erhält einer der beiden Partner eine Bestätigung außerhalb einer definierten Wartezeit, springt er einen Schritt zurück und versendet noch einmal seine letzte Bestätigung, aber nie die ursprüngliche Nachricht erneut.

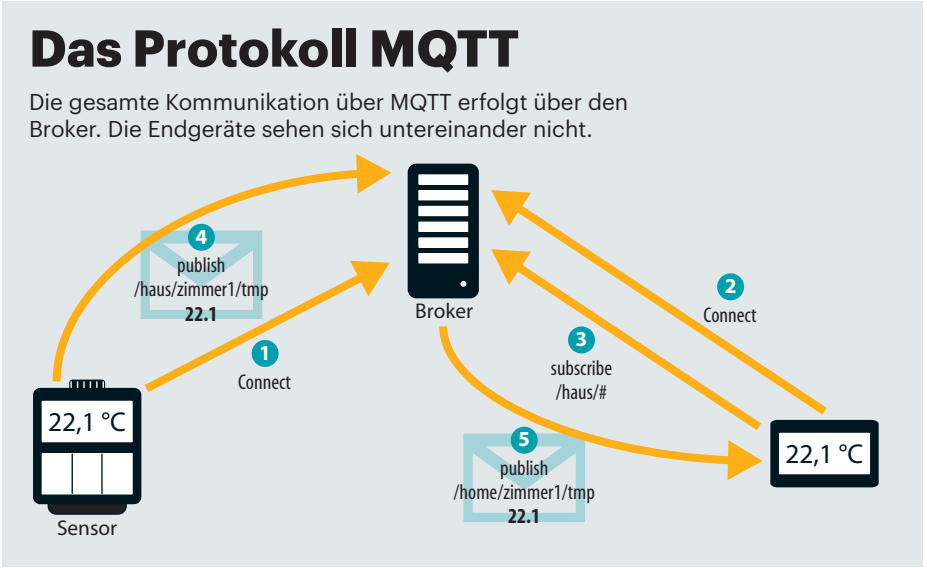
Der letzte Wille

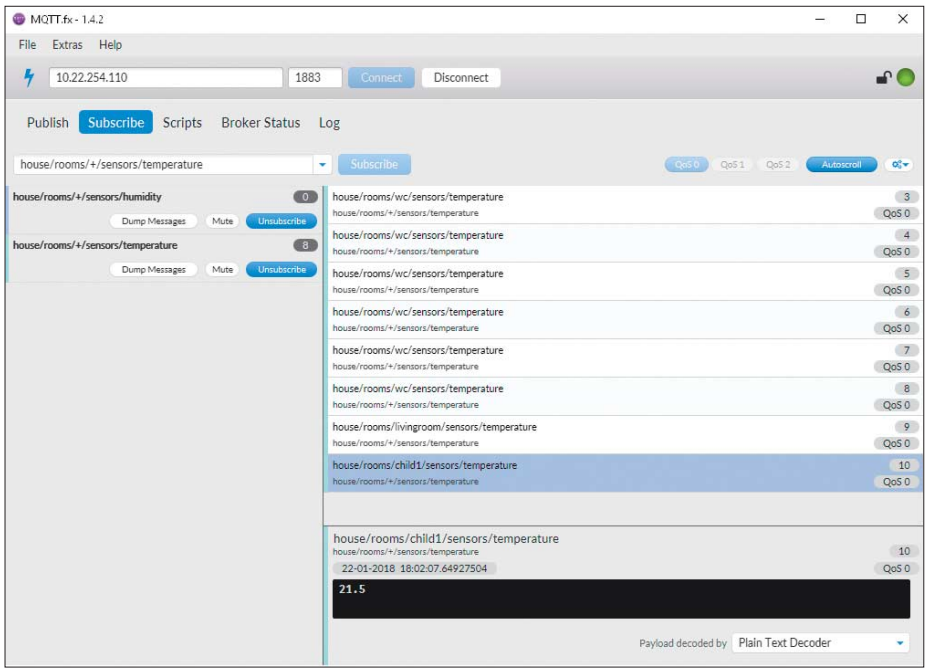
Ein Verbindungsabbruch kommt oft schneller als gedacht. Geht ein Gerät offline, kann es sich nicht mehr bei allen Weggefährten verabschieden, die vielleicht auf eine Nachricht warten. Für diesen Fall haben die Entwickler von MQTT vorgesorgt. Beim Verbindungsaufbau mit dem Broker kann jeder Client einen letzten Willen hinterlegen und für diesen die gleichen Eigenschaften festlegen, die auch für normale Nachrichten gelten: Topic, Payload, QoS und Retain-Flag. Der Broker merkt sich diesen letzten Willen und verteilt ihn an die Abonnenten, sobald die Verbindung getrennt wurde.

Broker zu Hause

Wer zu Hause mit MQTT experimentieren oder das Haus automatisieren möchte, kann auf verschiedene Open-Source-Broker zurückgreifen.

Die größte Verbreitung hat die Software Mosquitto, die von der Eclipse-Foundation entwickelt wird. Empfehlen kann man den Einsatz auf Linux – ein Raspberry Pi mit Raspbian Stretch oder Jessie wird mit einer Zeile zum Mosquitto-Server: `sudo apt install mosquitto`. Die Entwickler stellen auch eine Version für Windows bereit, dieser fehlen allerdings DLLs, die man per





Mit MQTT.fx bekommen Sie einen Überblick, welche Nachrichten über den Broker verschickt werden.

Hand aus einer speziellen OpenSSL-Installation herauskopieren muss. Um die Möglichkeiten von MQTT auszuprobieren, empfiehlt sich ein grafischer Client wie MQTT.fx, der auf Java basiert und unter macOS, Linux und Windows läuft. Den Download finden Sie über ct.de/ytzh.

Nach dem ersten Start wechseln Sie mit dem blauen Seiten-Symbol auf die freie Eingabe einer Server-Adresse. Verbinden Sie sich mit Ihrer Mosquitto-Instanz auf Port 1883. Im Reiter „Broker“ können Sie eine Statusübersicht des Servers auslesen. Wechseln Sie dazu im Drop-Down-Menü auf „Mosquitto“ und schließen Sie das Abo für die Statusmeldungen des Servers ab.

Im Reiter „Subscribe“ abonnieren Sie ein beliebiges Thema (oder gleich alle Themen mit #). Im Reiter „Publish“ versenden Sie selbst Nachrichten, die im Subscribe-Fenster wieder ankommen sollten. Wenn Sie einen zweiten Rechner zur Hand haben, der erst später eine Verbindung aufbaut, können Sie leicht die Funktionalität des Retained-Flag testen.

Logik zentralisieren

Der MQTT-Broker ist nur für das Annehmen und Austeilen von Nachrichten zuständig. Der Broker enthält daher keine Funktionen, um Regeln zu definieren oder Aktionen auszuführen. Hausautomationsprogramme wie openHAB oder Home Assistant arbeiten jedoch mit einem MQTT-Broker zusammen. Sie abonnieren den gesamten Verkehr, treffen auf Grundlage dieser und weiterer Daten Entscheidungen und veröffentlichen Nachrichten mit Steuerbefehlen. Ein MQTT-fähiger Lichtschalter sendet beispielsweise auf dem Topic `house/rooms/livingroom/switch` die Nachricht `"on"`. Die Hausautomation hat dieses Topic abonniert und führt eine Regel aus: „Schalte das Licht an, wenn der Schalter gedrückt wurde und es draußen dunkel ist.“ Über eine andere Quelle (aus dem Internet oder über eine Berechnung) besorgt sie sich die Information, ob die Sonne bereits untergegangen ist und gibt dann die Nachricht `home/rooms/livingroom/lamp` mit dem Wert `"on"` aus. Das Relais hat dieses Topic abonniert und schaltet. Überlässt man die gesamte Logik zwischen Sensor und Aktor einer Zentrale, macht man sich auf der einen Seite von dieser abhängig, kann aber schnell neue Funktionen einbauen, ohne die Endgeräte verändern zu müssen.

MQTT absichern

Ohne weitere Maßnahmen ist der gesamte MQTT-Verkehr so sicher oder unsicher wie das Netzwerk, über den er verschickt wird. Jedes Gerät darf alle Topics abonnieren und auf ihnen Nachrichten verschicken, die Nachrichten sind unverschlüsselt. Dankenswerterweise haben die Entwickler Sicherheitsfunktionen in MQTT vorgesehen und Mosquitto unterstützt sie.

Clients können dazu aufgefordert werden, sich per Benutzername und Kennwort beim Broker anzumelden. Dieser kann dann Lese- und Schreibrechte für alle Benutzer verwalten. Später sollten Sie die Verbindung verschlüsseln, damit das Kennwort nicht im Klartext übertragen wird. Öffnen Sie zum Einrichten der Anmeldung auf dem Server eine Kommandozeile und wechseln Sie in das Mosquitto-Verzeichnis:

```
cd /etc/mosquitto
```

Führen Sie dort den Befehl zum Anlegen einer Benutzerdatei aus und legen Sie den ersten Benutzer `admin` an:

```
sudo mosquitto_passwd -c pass.txt admin
```

Sie werden erst nach dem Root-Kennwort und anschließend zweimal nach einem Kennwort für den neuen MQTT-Benutzer gefragt. Einen weiteren Benutzer (mit dem Namen `sensor1` und dem Kennwort `secret`) erzeugt die Zeile:

```
sudo mosquitto_passwd -b pass.txt ↵
  ↳ sensor1 secret
```

Damit Mosquitto den Zugriff auf diese Benutzer beschränkt, öffnen Sie die Konfigurationsdatei

```
sudo nano mosquitto.conf
```

MQTT Dash für Android

Die App MQTT Dash spricht zwar kein Deutsch, dafür aber problemlos mit dem MQTT-Broker. Sie platzieren Schalter, Slider oder Color-Picker für verschiedene Räume frei auf der sehr schlichten Oberfläche und weisen diesen MQTT-Topics und Nachrichten zu. Über JavaScript können Sie Reaktionen programmieren und damit eine komfortable Bedienoberfläche zusammenbauen.

Dort fügen Sie zwei Zeilen hinzu:

```
allow_anonymous false
password_file: /etc/mosquitto/pass.txt
```

Starten Sie Mosquitto neu, damit das Programm die Einstellungen lädt:

```
sudo systemctl restart mosquitto
```

MQTT.fx wird jetzt beim Versuch scheitern, eine Verbindung herzustellen, weil der Benutzer nicht mehr berechtigt ist. Klicken Sie oben links auf das Blitz-Symbol und legen Sie ein Verbindungsprofil für den Server an. Im Reiter „User Credentials“ hinterlegen Sie Benutzername und Kennwort. Der Client kann sich jetzt wieder verbinden, die beiden Benutzer haben noch unbeschränkten Zugriff. Um das zu ändern, bearbeiten Sie erneut die Datei `mosquitto.conf` und ergänzen sie um die Zeile

```
acl_file /etc/mosquitto/acl.txt
```

In dieser Datei erlauben Sie dem Admin Vollzugriff, dem Sensor eingeschränkten Schreibzugriff auf einige Topics:

```
user admin
topic readwrite #
user sensor1
topic write house/rooms/+/sensors/#
```

Soll jeder Benutzer ein Topic bekommen, auf das nur er Zugriff hat, können Sie eine Regel mit dem Platzhalter `%u` für den Benutzernamen verwenden. Die folgende Zeile gewährt jedem Nutzer lesenden Zugriff auf einen Posteingang mit seinem Namen:

```
topic read postbox/%u/#
```

Um die neuen Regeln in Betrieb zu nehmen, starten Sie Mosquitto neu. Um TLS zu aktivieren, benötigen Sie ein TLS-Zertifikat, das Ihre Clients akzeptieren. Als Beispiel bieten sich Zertifikate von Let’s Encrypt an [1]. Fügen Sie dafür einen Listener für Port 8883 und die Pfade zu Zertifikat, Zertifikatskette und Schlüssel hinzu:

```
listener 8883
certfile /etc/letsencrypt/live/↵
↳iot.example.org/cert.pem
cafile /etc/letsencrypt/live/↵
↳iot.example.org/chain.pem
keyfile /etc/letsencrypt/live/↵
↳iot.example.org/privkey.pem
```

Um den Zugriff über den unverschlüsselten Kanal abzuschalten, können Sie den Listener für Port 1883 ganz entfernen oder mit `listener 1883 localhost` auf den lokalen Zugriff einschränken.

Scheunentore schließen

Die Hausautomation über MQTT zu steuern ist einfach und schnell eingerichtet. Für iOS und Android gibt es jeweils kostenlose Apps (siehe Kästen), mit denen Sie mit wenigen Klicks eine mobile Bedienoberfläche zusammengestellt haben und im lokalen Netzwerk Steuerbefehle versenden können. Damit das auch von unterwegs funktioniert, richten viele Anwender eine Portweiterleitung von Port 1883 in ihrem Router auf den Broker ein und vertrauen darauf, dass schon niemand die externe IP-Adresse erraten wird. Dass dieser Plan nicht aufgeht, können Sie leicht mithilfe der IoT-Suchmaschine shodan.io feststellen. Diese durchsucht systematisch das Internet nach offenen Ports und ermöglicht eine Filterung nach Protokollen. Eine Suche nach `mqtt country:"DE"` (nur nach kostenloser Registrierung möglich) liefert mehrere tausend Treffer allein in Deutschland – was grundsätzlich nicht schlimm wäre. Die meisten davon nutzen jedoch weder Verschlüsselung noch authentifizierten Zugang und Zugriffsbeschränkungen. Neben zahlreichen privaten Nutzern mit Hausautomations-Topics fanden wir auch Lasertag-Spielhallen und Taxi-Anbieter in der Trefferliste. Sollten Sie aus der Ferne auf Ihren MQTT-Broker zugreifen wollen, sind Verschlüsselung und Authentifizierung Pflicht. Wenn Sie ganz sichergehen wollen, sollten Sie nur über VPN aus der Ferne auf den Broker zugreifen.

Mehr als Automation

Die Anwendungsbereiche von MQTT gehen weit über Hausautomation und Steuerung von Industrieanlagen hinaus.

IoT OnOff für iOS

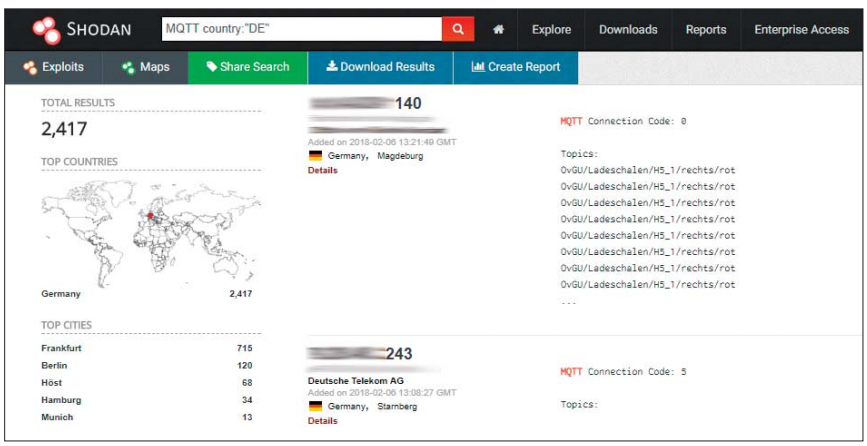
IoT OnOff kann mehr als nur ein- und ausschalten. Definieren Sie Schieberegler, Schalter, Color-Picker und Anzeigen für Werte und ordnen Sie diese nach Räumen. Bis zu zehn Elemente können Sie in der kostenlosen Version platzieren, die Vollversion kostet 3,99 Euro. Gut gemacht sind die zahlreichen Optionen für die einzelnen Steuerelemente.

So wurde durch einen Blog-Post einer Facebook-Entwicklerin bekannt, dass auch der Facebook-Messenger in Teilen mit MQTT arbeitet. *(jam@ct.de)* c't

Literatur

[1] Uli Ries, Let’s Encrypt!, SSL/TLS-Zertifikate gratis für alle, c’t 4/2018, S. 80

Mosquitto, MQTT.fx: ***ct.de/ytzh***



Die IoT-Suchmaschine Shodan zeigt, wer MQTT-Broker ohne Anmeldung betreibt. Die Treffer reichen von Heimautomation bis Industrie.