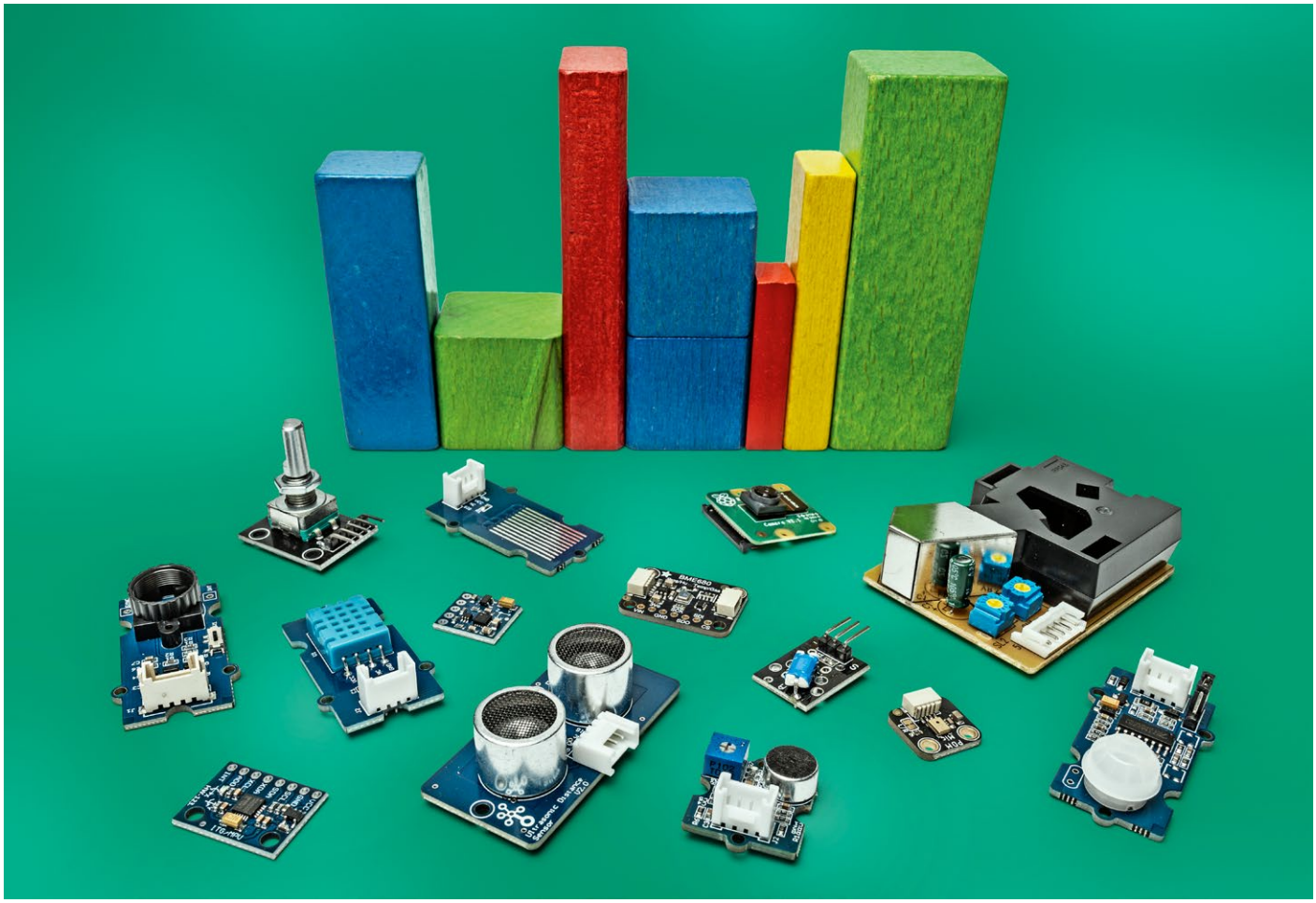


Daten greifen und begreifen

Daten sammeln, verteilen und visualisieren: MQTT, Node-Red, Influx, Grafana

Fertige oder selbstgebaute Kleingeräte mit Mikrocontroller darin, etwa Umweltsensoren, sind nicht dafür gemacht, Daten längerfristig aufzuheben. Wir zeigen, mit welchen Komponenten Sie deren Daten einsammeln, langfristig aufbewahren und auswerten.

Von Peter Siering

Wenn etwa in der Heimautomatisierung im Minutenrhythmus Messdaten anfallen und anderswo verarbeitet werden sollen, braucht es einen zuverlässigen Informationsvermittler. Der kann idealerweise über eine Standardschnittstelle Daten annehmen und abgeben. Mit dem „Message Queue Telemetry Transport“ (MQTT) gibt es genau das, ein Protokoll, mit dem Geräte nicht nur Daten austauschen, sondern auch Befehle empfangen können.

MQTT-fähige Geräte senden ihre Daten an einen MQTT-Server, der Broker heißt. Dort können andere MQTT-Geräte die Daten abonnieren. Treffen neue Daten ein, reicht der MQTT-Server sie an alle Abonnenten weiter. MQTT ist keine Einbahnstraße: Jedes Gerät kann selbst auch Daten empfangen. In der Praxis erhalten sie häufig Konfigurationsdaten oder Steuerbefehle auf diesem Weg.

Auch Software kann als MQTT-Client die Datenquellen über ein Abonnement anzapfen. So lassen sich zum Beispiel Dienste realisieren, die Temperaturen regelmäßig in eine Datenbank schreiben, um sie später auszuwerten. Auch im Smart Home verwenden manche Steuerzentralen MQTT, um Lampen anzusteuern.

MQTT entkoppelt

Unterm Strich hilft MQTT, Aufgaben zu trennen, etwa Sensoren und Steuerung, aber auch, Komponenten so zu isolieren, dass man sie austauschen kann, etwa die verwendete Steuerzentrale im Smart Home. Aber nicht nur das: Es ist auf Robustheit ausgelegt. Es kennt verschiedene Qualitätsanforderungen für die Zustellung von Nachrichten und funktioniert so auch über langsame und schlechte Verbindungen. Auf Wunsch speichert ein Broker stets die letzte Nachricht zwischen.

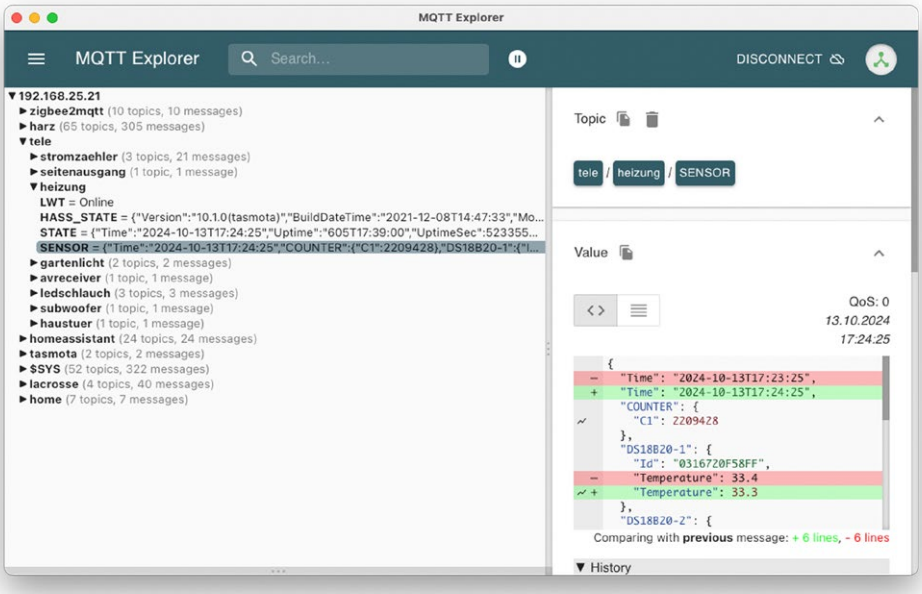
Nimmt sich einer der beteiligten Kommunikationspartner eine Auszeit, stört das die anderen nicht. Schlimmstenfalls gehen Nachrichten verloren. Da die normalerweise ohnehin regelmäßig übertragen oder abgefragt werden, fällt das oft nicht auf. Clients können vorbeugend für einen Verbindungsabbruch einem Broker den Auftrag erteilen, Abonnenten eine besondere Nachricht zu übermitteln.

MQTT-Nachrichten bestehen aus aneinander gehängten Topics, wie zum Beispiel „Zuhause/Bad/Temperatur“. So ergibt sich auf dem Broker eine Hierarchie ähnlich dem Verzeichnisbaum eines Dateisystems. An einem solchen Topic hängt der Nachrichteninhalt. Das können Binär- oder Klartextdaten sein; oft ist es JSON. Vorgaben für die Struktur und wie Topics zu heißen haben, gibt es nicht.

MQTT-Clients können mit Wildcards Nachrichten für einen ganzen Teilbaum abonnieren: Ein angehängtes „#“ liefert alle Nachrichten an den darunter liegenden Teilbaum, etwa aus „Zuhause/Bad/#“ die Topics „Temperatur“ und „Feuchte“. Mit einem „+“ kann ein Client gezielt nur die in einem Teilbaum erfassten Werte für die Temperatur abfragen, etwa mit „Zuhause/+/Temperatur“.

Nützlich für den Umgang mit MQTT ist Software, um die eingehenden Nachrichten zu beobachten. Hier empfiehlt sich der MQTT Explorer von Thomas Nordquist. Das Programm ist zwar etwas in die Jahre gekommen, aber nach wie vor sehr nützlich. Inzwischen hat der Entwickler angefangen, an der Version 0.4 zu arbeiten.

Für die Inbetriebnahme eines MQTT-Brokers lautet die Empfehlung nahezu einhellig, zu Eclipse Mosquitto zu greifen. Dieser Server-Dienst lässt sich am einfachsten per Docker einrichten. Wenn der Server auch aus dem Internet erreichbar sein soll, muss man akribisch darauf achten, die dafür eingerichteten Konten mit starken Passwörtern zu versehen und unbedingt nur TLS-gesicherte Verbindungen zu verwenden.



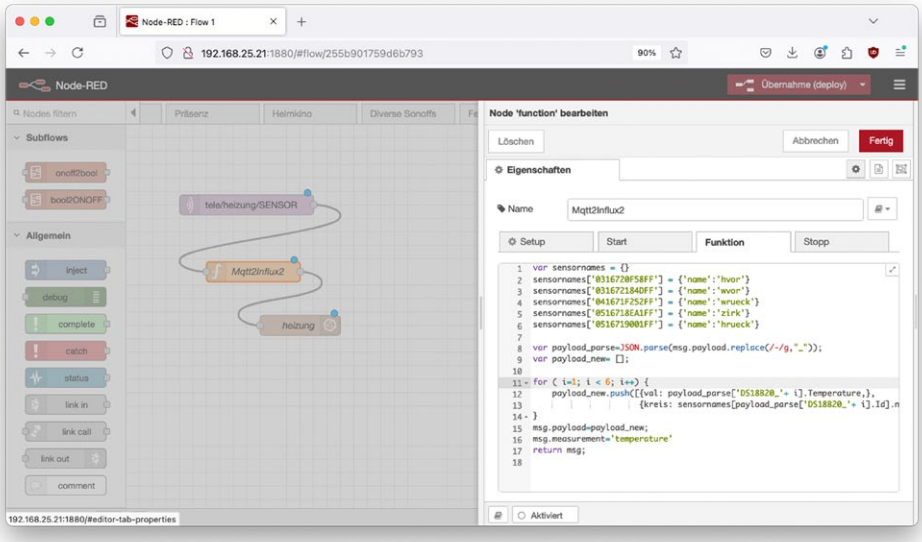
Um einen MQTT-Broker zu verstehen, hilft ein Programm, das ihn beobachtet. MQTT Explorer tut das und zeigt Änderungen an dorthin übermittelten Daten übersichtlich an, je nach Beschaffenheit sogar als Grafik.

Node-Red verbindet

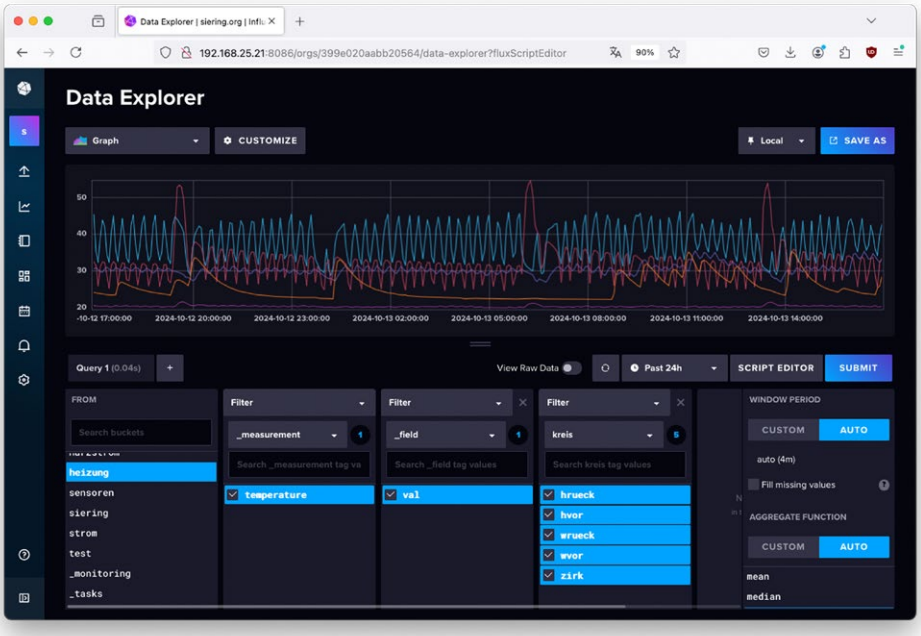
Wer selbst MQTT-Nachrichten generieren will, etwa um damit ein Gerät zu konfigurieren, kann Befehle auf der Kommandozeile einsetzen. Viel bequemer ist es aber, auf Software wie Node-Red zurückzugreifen: Dieses im Browser bediente und auf einem Server ausgeführte Werkzeug zur grafischen Programmierung spricht unter anderem MQTT.

Mit wenigen Mausklicks erstellen Sie in Node-Red eine Verbindung zu einem MQTT-Broker und abonnieren dort Topics. So können Sie Daten anzeigen oder weiterverarbeiten lassen. Nützlich ist Node-Red besonders dann, wenn Sie Daten von Geräten verarbeiten wollen, die selbst kein MQTT sprechen: Oft genügen einige Nodes oder ein paar Zeilen Code, um diese einzusammeln, passend aufzubereiten und dem Broker als MQTT-Topic zu übermitteln.

Für allerlei Funktionen und Erweiterungen stehen fertige Funktionen bereit,



Das grafische Entwicklungswerkzeug Node-Red vermittelt bei Bedarf zwischen per MQTT bereitstehenden Sensordaten und Zeitreihendatenbanken. Dafür eingesetzter JavaScript-Code hilft dabei, Sensor-IDs in lesbare Namen zu übersetzen, die als Tags in InfluxDB landen.



Mit dem Data Explorer in der Weboberfläche von InfluxDB stöpselt man seit Version 2 recht schnell Auswertungen der eingelaufenen Daten zusammen und erzeugt daraus ansehnliche Dashboards.

Knoten in der Node-Red-Diktion. Wenn die nicht genügen, um Daten von einem ins andere Format zu überführen, macht das nichts: Node-Red sieht vor, selbstgeschriebenes JavaScript in eigene Knoten zu verpacken.

Node-Red macht auch dann eine gute Figur, wenn es darum geht, per MQTT verfügbare Daten zum Beispiel an eine Zeitreihendatenbank weiterzugeben. Das ist so oft viel leichter zu realisieren als mit den umständlichen Werkzeugen, die solche Datenbanken selbst bereitstellen. Die naheliegende Lösung ist hier im Zweifel eine Ihnen vertraute Technik. Es finden sich im Netz Implementierungen, um mit Python Daten aus MQTT in eine Datenbank zu schieben.

Auf den ersten Blick klingt das alles nach einem sehr großen Aufwand für eine sehr kleine Aufgabe. Sensoren könnten doch schließlich auch direkt Daten in eine Datenbank schieben. Im Prinzip ja, aber: Dank MQTT verlassen die Daten ihr Silo und sind für so gut wie jeden gängigen Dienst zugänglich, etwa zur Heimautomatisierung mit Home Assistant, Homebridge, OpenHAB, ioBroker und so weiter – Sie lösen damit Ihre Daten und Sensoren aus der Abhängigkeit einzelner Dienste.

InfluxDB speichert

Als Aufbewahrungsort für Sensordaten bietet sich die Zeitreihendatenbank InfluxDB an. Sie bringt eine eigene Weboberfläche mit, um die Grundkonfiguration einzurichten und eingegangene Daten auch grafisch darzustellen. In der Weboberfläche kann man die Abfragen mehr oder minder zusammenklicken, um die Daten zu selektieren. InfluxDB kennt obendrein zwei eigene Abfragesprachen.

InfluxDB verwendet etwas andere Konzepte und Namen als andere Datenbanken. Messreihen (measurements) entsprechen einer Tabelle in einer SQL-Datenbank. Automatisch erzeugte Zeitstempel (points) sind wie Zeilen einer SQL-Tabelle. Werte (fields), zum Beispiel eine Temperaturangabe, lassen sich mit nicht indizierten Spalten vergleichen. Zusätzliche Merkmale für Werte (tags) sind verwandt mit indizierten Spalten einer SQL-Tabelle.

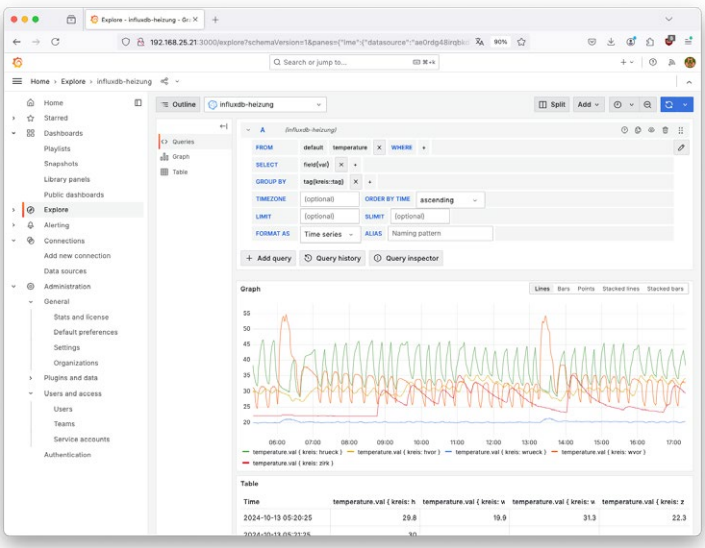
Außerdem begegnen dem Influx-Nutzer drei weitere Konzepte: Eine „Organisation“ stellt eine Verwaltungseinheit dar, innerhalb derer sich Rechte vergeben lassen. Hier genügt meist eine. Ein „Bucket“ könnte man als Container für Messreihen beschreiben. Wichtig ist die einem Bucket zugeordnete „Retention Policy“, also die Aufbewahrungsfrist; alte Daten löscht eine Influx-Installation auf Anforderung automatisch.

Für den interaktiven Zugriff auf die Weboberfläche setzt Influx auf herkömmliche Benutzernamen nebst Kennwort. Für den automatisierten Zugriff, also das Hinzufügen von Daten und deren Abruf, setzt Influx auf Tokens. Die generiert es als „API Tokens“ in seiner Weboberfläche mit unterschiedlichen Rechten. Achtung: Notieren Sie die generierten Tokens idealerweise in einem Passwortmanager, sie werden nur einmal angezeigt.

Viele Beispiele und Tutorials verwenden die an SQL orientierte Sprache „InfluxQL“, die mit InfluxDB in Version 1 eingeführt worden war. Version 2 ersetzt diese durch die im Vergleich dazu gewöhnungsbedürftige Sprache „Flux“, die völlig anders aufgebaut ist. Mit der nächsten Hauptversion wollen die Entwickler nach anhaltender Kritik zu InfluxQL zurückkehren. Kurzum: Ignorieren Sie Flux.

Grafana garniert

Wenn Ihnen die Möglichkeiten zur grafischen Auswertung in InfluxDB nicht genügen oder Sie InfluxQL benutzen möchten, landen Sie früher oder später bei der



Wer es hübscher mag, greift zu Grafana: Hier lassen sich die InfluxDB-Daten ebenfalls auswerten und vor dem Erstellen von dauerhaft erreichbaren Dashboards erforschen. Grafana verwendet auf Wunsch als Abfragesprache das wieder angesagte SQL-artige InfluxQL.

Analyse- und Visualisierungslösung Grafana. Die kann nämlich auch mit der wiederbelebten Abfragesprache arbeiten. Etwas trickreich ist die Einrichtung einer Verbindung zu InfluxDB mit InfluxQL zur Datenabfrage. Unserer Erfahrung nach sind viele Anleitungen im Netz unvollständig oder führen nicht zum Ziel.

Bei uns hat das Einrichten von InfluxDB als Datenquelle in Grafana geklappt, indem wir zunächst in Influx einen Zugriffs-Token erzeugt haben. Den haben wir dann in Grafana unter Datasources als Custom HTTP-Header „Authorization“ in das Feld „Value“ eingetragen und ihm das Wort „Token“ mit einem trennenden Leerzeichen vorangestellt. Als weitere Felder sind nur noch die URL einzutragen, der Name der Datenbank (nicht die ID des Buckets) und als HTTP-Methode „GET“ zu wählen.

So mächtig InfluxDB und Grafana auch sind, so prächtig kann man sich darin verzetteln. Deshalb unser Tipp: Schauen Sie, ob nicht Influx schon genügt oder vielleicht die Funktionen der ohnehin schon einsatzbereiten Heimautomationslösung weit genug reichen. Manchmal ist weniger auch mehr.

Wo man Node-Red, MQTT-Broker, InfluxDB und Grafana laufen lässt, ist nicht nur eine Geschmacksfrage: Den beiden erstgenannten genügt ein Raspberry Pi 4 oder 5. Für die Zeitreihendatenbank eher nicht, weil wegen der sehr häufigen Schreibzugriffe keine SD-Karte als Speichermedium verwendet werden sollte. Und InfluxDB und Grafana brauchen etwas Dampf, wenn sie über Wochen oder Monate gesammelte Daten auswerten sollen. Ein NAS oder ein Heimserver sind dafür gut geeignet.

Wer die Dienste nicht selbst betreiben will, kann auch auf fertige Cloud-Angebote zurückgreifen. Doch das kann ganz schnell teuer werden und birgt mitunter Überraschungen: So musste InfluxData neulich einräumen, in seinen Rechenzentren vorschnell die Daten zahlender Kunden gelöscht zu haben. Die Vorwarnungen hatten sie nicht erreicht und so wurden sie kalt erwischt.

Strich drunter

Es klang schon an: Jedes der empfohlenen Werkzeuge ist für sich bereits ein heftiges Kaliber und braucht einige Einarbeitung. InfluxDB als Datensammelbecken ist außerdem in Bewegung. Wer die Komponenten mit Docker benutzt, sollte von der eigentlich sinnvollen Strategie Abstand nehmen, die Images automatisiert mit Updates zu versehen. Sonst sitzt man womöglich plötzlich vor einem grundrenovierten GUI, auf dem man sich kaum mehr zurechtfindet. Solange die Dienste nicht von außen erreichbar sind, scheint das für manchen der bessere Kompromiss. *(ps@ct.de)* c't

Literatur

[1] Jan Mahn, Weltsprache, Das Protokoll MQTT für robusten Datenaustausch in Industrie und Hausautomation, c't 6/2018, S. 164

Erwähnte Tools: ***ct.de/ywsf***