

Softwareentwurf und Anwendungen verteilter Systeme

BA Internet der Dinge – Gestaltung vernetzter Systeme

Semester 3

Hochschule für Gestaltung Schwäbisch Gmünd

Dozent: Yannick Schiele

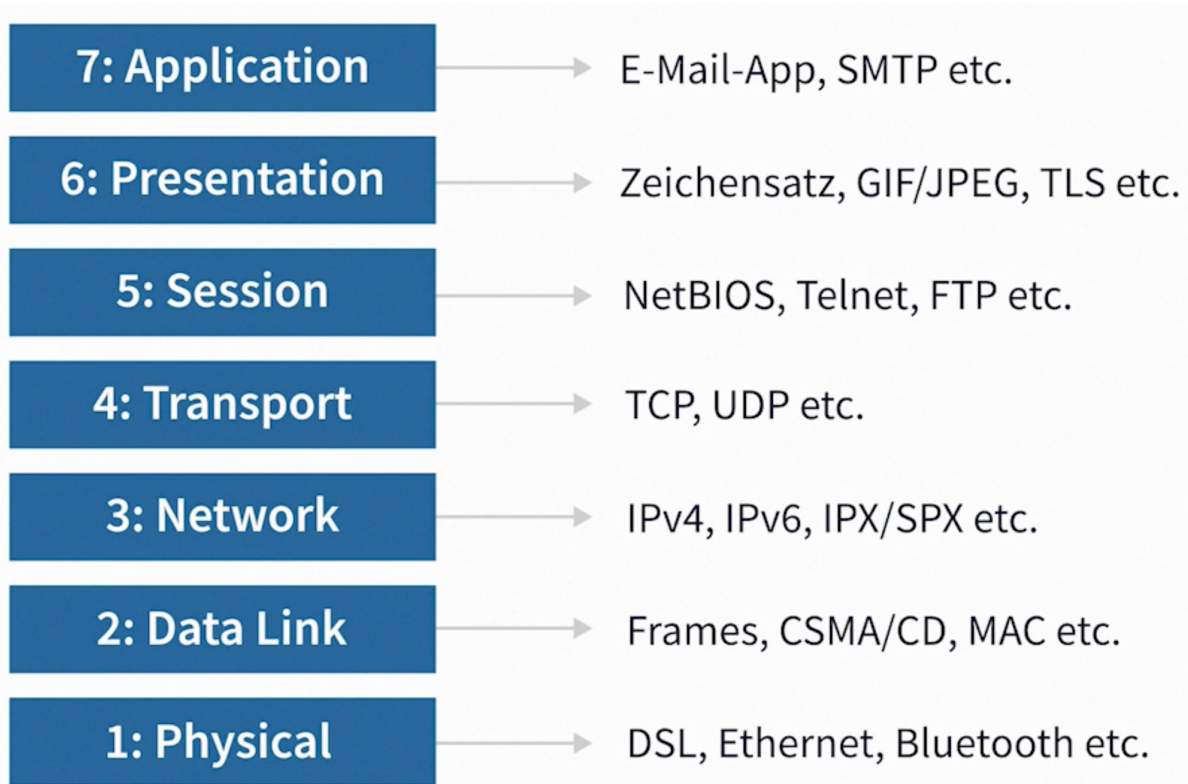
Azure Setup

Arduino

Webservertechnologien

Agenda

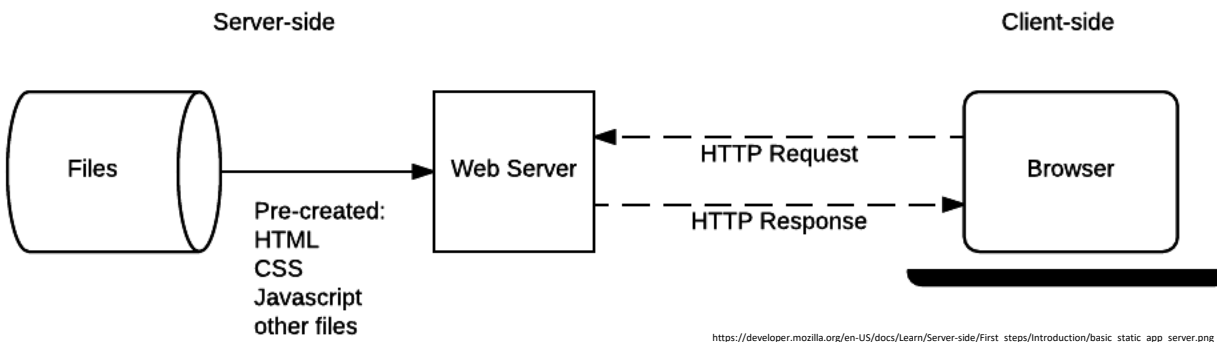
- Einführung
- NodeMCU ESP 8266
- Installation
- Programmierung
 - Sensoren
 - Aktoren
 - Webserver
- REST-APIs



<https://www.linkedin.com/learning/netzwerkgrundlagen-1-die-theorie-fur-die-praxis/das-osi-modell?autoplay=true&u=8226650>

OSI-Referenzmodell

Die verwendeten Protokolle der einzelnen Schichten beim Versenden einer E-Mail per Mail Client wie bspw. Outlook



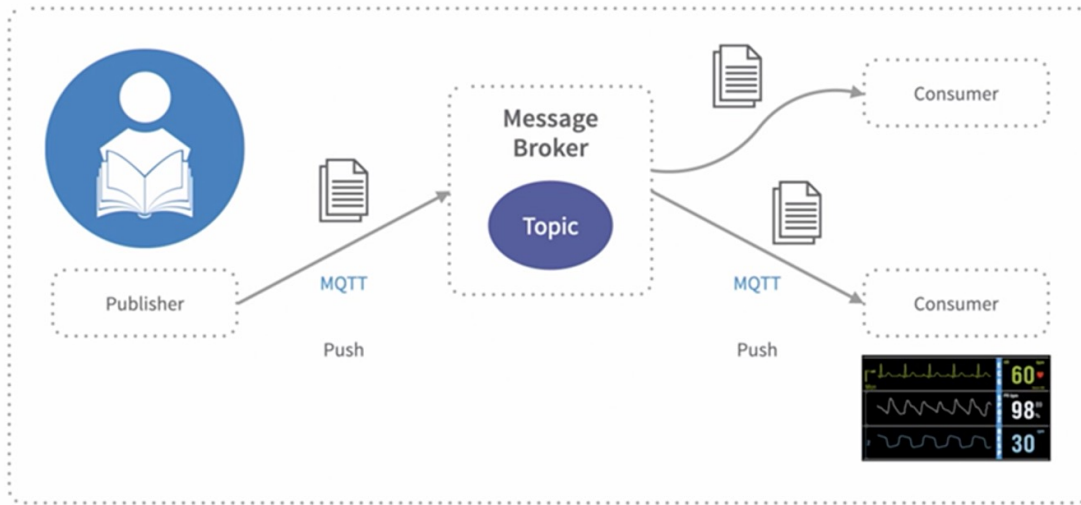
Webserver

- Als Webserver bezeichnet man jene Server, die zur Verbreitung von Webinhalten im Inter- oder Intranet dienen
- Als Teil eines Rechnernetzwerks übertragen sie Dokumente an sogenannte Clients– beispielsweise eine Webseite an einen Webbrowser
- Für die Übermittlung wird das Übertragungsprotokoll HTTP (OSI Layer 6) genutzt das auf den Netzwerkprotokollen IP und TCP beruht

<https://www.ionos.de/digitalguide/server/knowhow/webserver-definition-hintergruende-software-tipps/>

Netzwerkprotokolle

TCP, HTTP, MQTT, etc.

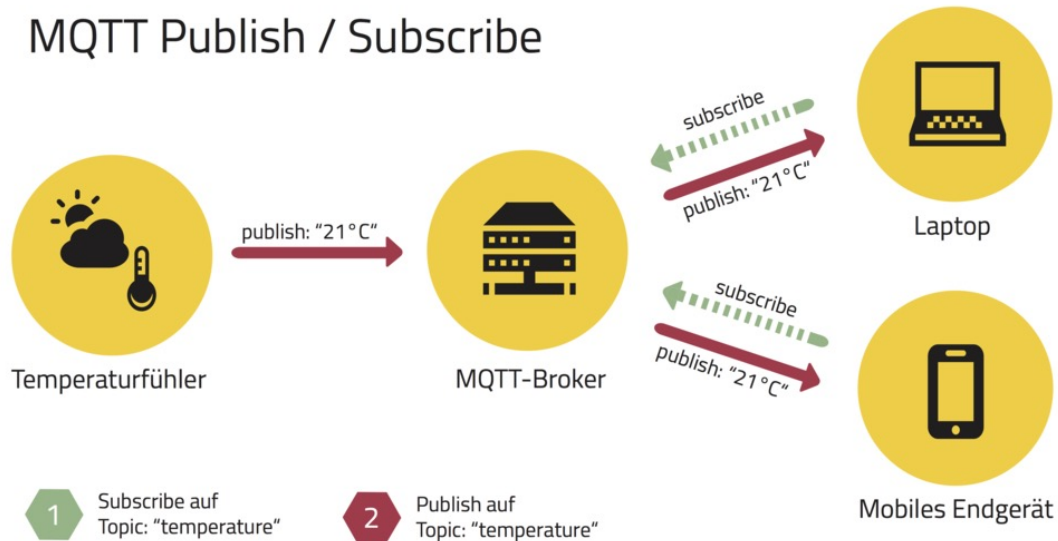


MQTT

- Message Queuing Telemetry Transport
- Von IBM und Eurotech entwickelt
- auf Umgebungen mit niedriger Bandbreite und hoher Latenz spezialisiert → ideales Protokoll für Machine-to-Machine-Kommunikation und IoT
- Funktioniert nach Publish/Subscribe-Modell
- OSI Schicht 7, meistens aufbauend auf TCP/IP

<https://www.linkedin.com/learning/search?keywords=mqtt&u=82266650>

MQTT Publish / Subscribe



https://www.informatik-aktuell.de/fileadmin/_processed_/8/e/csm_mqtt_abb1_florian_raschbichler_4703644d7b.png

MQTT

- Publisher- / Subscriber-Prinzip über einen zentralen Broker (Bspw. Azure IoT Hub)
- Sender und Empfänger haben keine direkte Verbindung
- Die Datenquellen melden ihre Daten über einen Publish nur an den Broker
- Broker verteilt die Nachricht an alle Empfänger mit Interesse an der Nachrichten (Subscribe auf "Topic")
- Bsp: Temperatursensor

<https://www.linkedin.com/learning/search?keywords=mqtt&u=82266650>



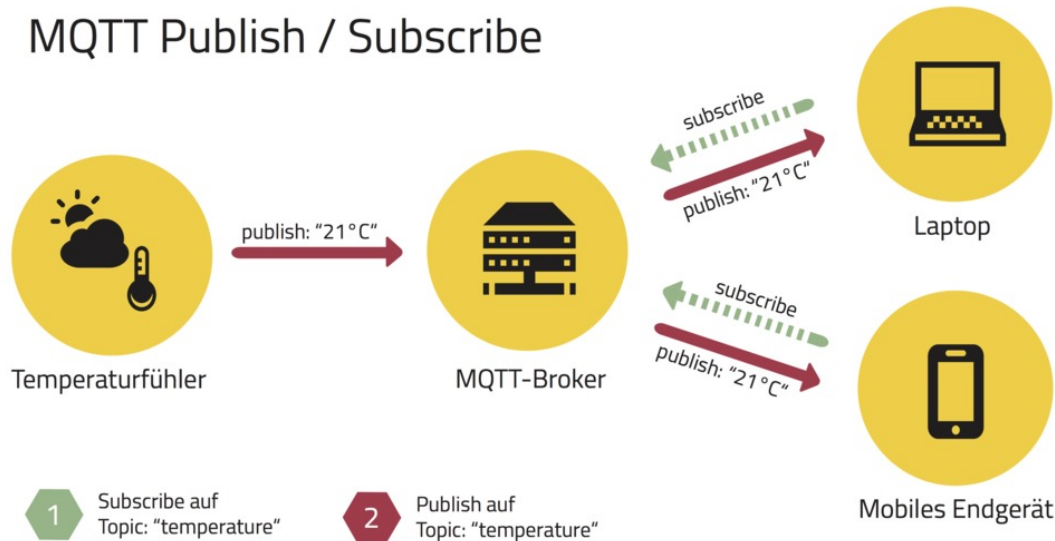
https://www.hivemq.com/img/blog/topic_wildcard_plus.png

MQTT Topic

- Topic sieht aus wie ein hierarchischer Dateipfad
- Clients können eine bestimmte Hierarchieebene eines Topics abonnieren
- ein Platzhalterzeichen (Wildcard) wird verwendet, um mehrere Stufen zu abonnieren

<https://www.linkedin.com/learning/search?keywords=mqtt&u=82266650>

MQTT Publish / Subscribe



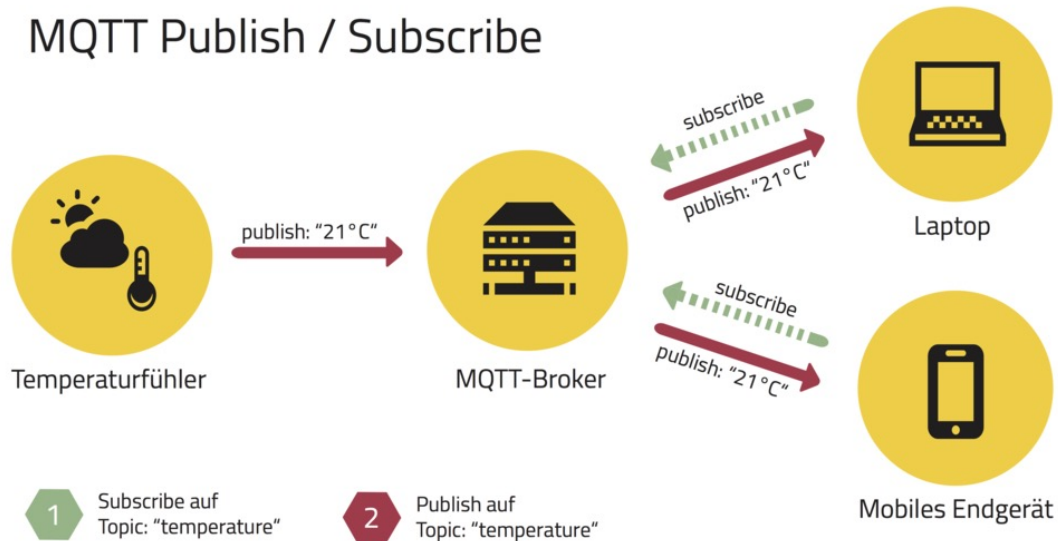
https://www.informatik-aktuell.de/fileadmin/_processed_/8/e/csm_mqtt_abb1_florian_raschbichler_4703644d7b.png

MQTT Broker

- zentrale Knotenpunkt
- kann bis zu Tausende gleichzeitig verbundene MQTT Clients verwalten (IoT)
- ist dafür verantwortlich, alle Nachrichten zu empfangen, zu filtern, und die Nachricht an die abonnierten Clients zu senden.
- Der Broker hält Sitzungen aller persistenten Clients ab, einschließlich Abonnements und entgangenen Nachrichten.
- weitere Aufgabe des Brokers ist die Authentifizierung und Autorisierung von Clients

<https://www.opc-router.de/was-ist-mqtt/>

MQTT Publish / Subscribe



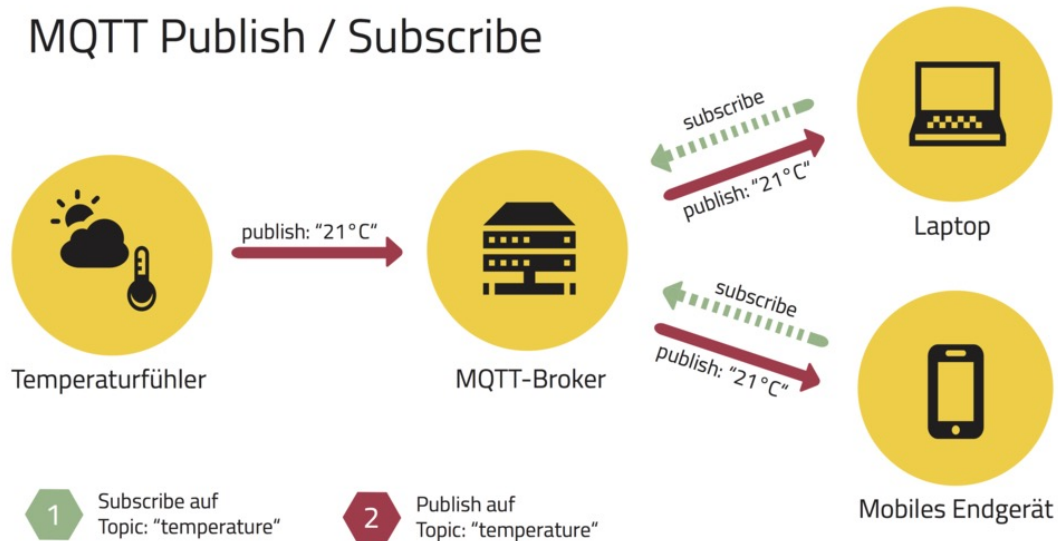
https://www.informatik-aktuell.de/fileadmin/_processed_/8/e/csm_mqtt_abb1_florian_raschbichler_4703644d7b.png

MQTT Message

- Jede Nachricht verfügt über ein Topic und sogenannten Payload.
- MQTT-Payload ist nicht an eine bestimmte Struktur gebunden und kann frei gestaltet werden
- Dennoch ist es sinnvoll den Nachrichteninhalt eine bestimmte Struktur vorzugeben, damit dieser von anderen Geräten oder Software gelesen werden kann
- Mögliche Nachrichtenstrukturen sind Json, XML, etc.

<https://www.opc-router.de/was-ist-mqtt/>

MQTT Publish / Subscribe



https://www.informatik-aktuell.de/fileadmin/_processed_/8/e/csm_mqtt_abb1_florian_raschbichler_4703644d7b.png

MQTT Client

- Als MQTT-Client werden alle Geräte und auch Software, bezeichnet, die in irgendeiner Art und Weise mit dem Broker verbunden sind
- Ein Client kann dem Broker Nachrichten senden (publish) und Nachrichten vom Broker erhalten (subscribe)
- Beim Senden einer Nachricht an den Broker, muss ein MQTT-Topic angegeben werden, anhand dessen die Nachricht weiter verarbeitet werden kann

<https://www.opc-router.de/was-ist-mqtt/>

```

const mqtt = require('mqtt')
const host = 'broker.emqx.io'
const port = '1883'
const clientId = `mqtt_${username}`
const connectUrl = `mqtt://${host}:${port}`
const client = mqtt.connect(connectUrl, {
  clientId,
  clean: true,
  connectTimeout: 4000,
  username: 'emqx',
  password: 'public',
  reconnectPeriod: 1000,
})
const topic = '/nodejs/mqtt'
client.on('connect', () => {
  console.log('Connected')
  client.subscribe([topic], () => {
    console.log(`Subscribe to topic '${topic}'`)
  })
  client.publish(topic, 'nodejs mqtt test', { qos: 0, retain: false },
(error) => {
  if (error) {
    console.error(error)
  }
})
})
client.on('message', (topic, payload) => {
  console.log('Received Message:', topic, payload.toString())
})

```

H f G

MQTT Example

npm install mqtt

Node mqttClient.js

Gemeinsame Programmierung

→ Eure Topics in Slack posten



MQTT Example

Online MQTT Client:



<http://www.emqx.io/online-mqtt-client/>

General

* Name

* Client ID  

* Host

* Port  

* Path

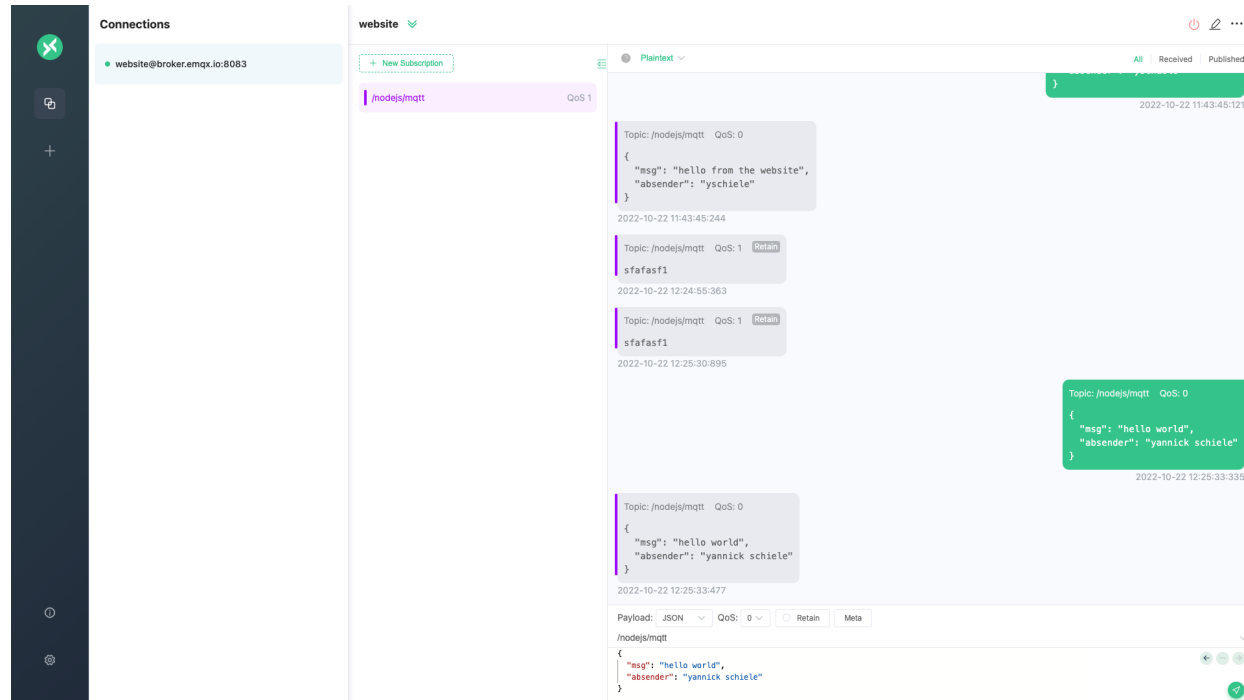
Username

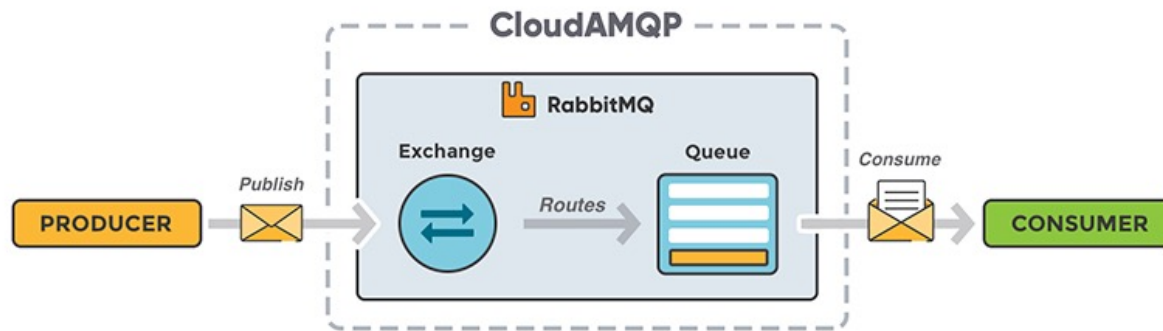
Password

SSL/TLS ☐

MQTT Client Example

<http://www.emqx.io/online-mqtt-client/>



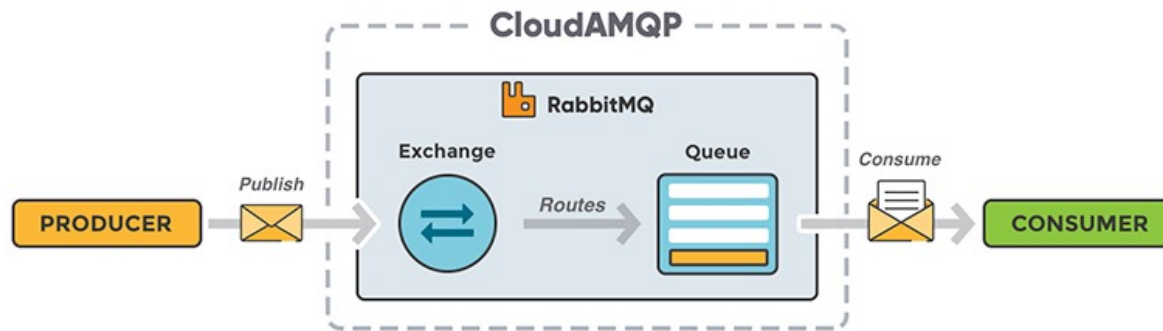


<https://www.ionos.de/digitalguide/websites/web-entwicklung/advanced-message-queuing-protocol-amqp/>

AMQP

- OSI Schicht 7 und Broker für die Übermittlung von Nachrichten wie MQTT
- Verwendung bspw. im Azure Service Bus

https://www.linkedin.com/learning/search?entityType=ALL&keywords=AMQP&language=en_US&spellcheck=true&ui=82266650

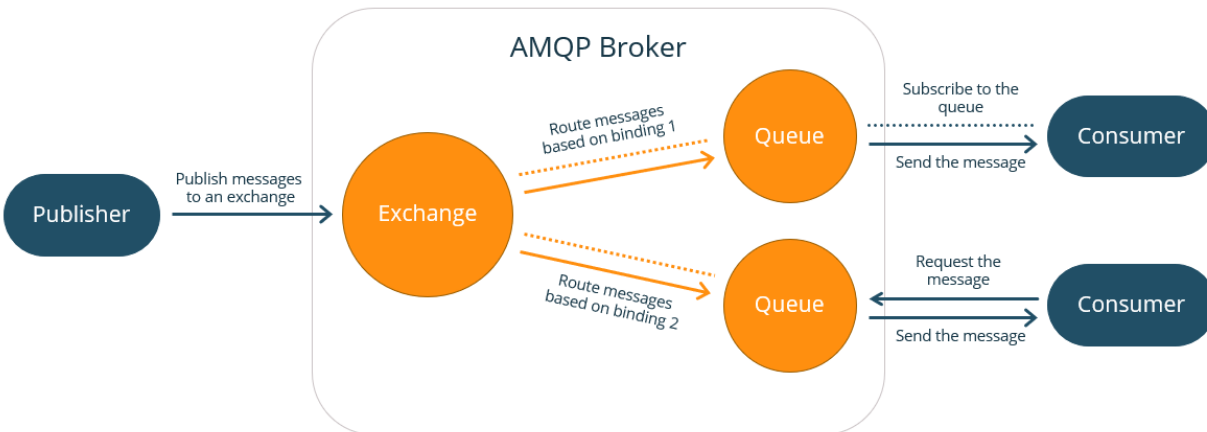


<https://www.ionos.de/digitalguide/websites/web-entwicklung/advanced-message-queuing-protocol-amqp/>

AMQP

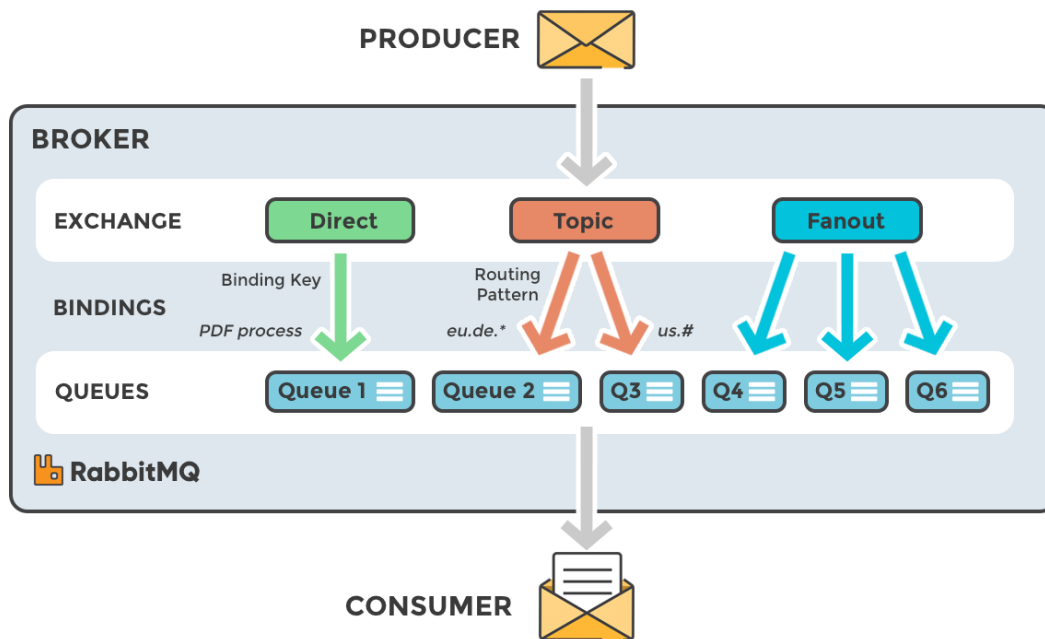
- Advanced Message Queuing Protocol
- sorgt für eine robuste Datenübertragung
- AMQP ermöglicht Nachrichten in einer Warteschlange zu lagern → asynchrone Kommunikation
- Der Empfänger der Nachricht muss die Information nicht direkt annehmen, verarbeiten und dem Sender den Empfang bestätigen
- Stattdessen holt er sich die Nachricht aus der Warteschlange, wenn er Kapazitäten dazu zur Verfügung hat.
- Das gibt dem Produzenten unterdessen die Möglichkeit weiterzuarbeiten

https://www.linkedin.com/learning/search?entityType=ALL&keywords=AMQP&language=en_US&spellcheck=true&un=82266650



AMQP

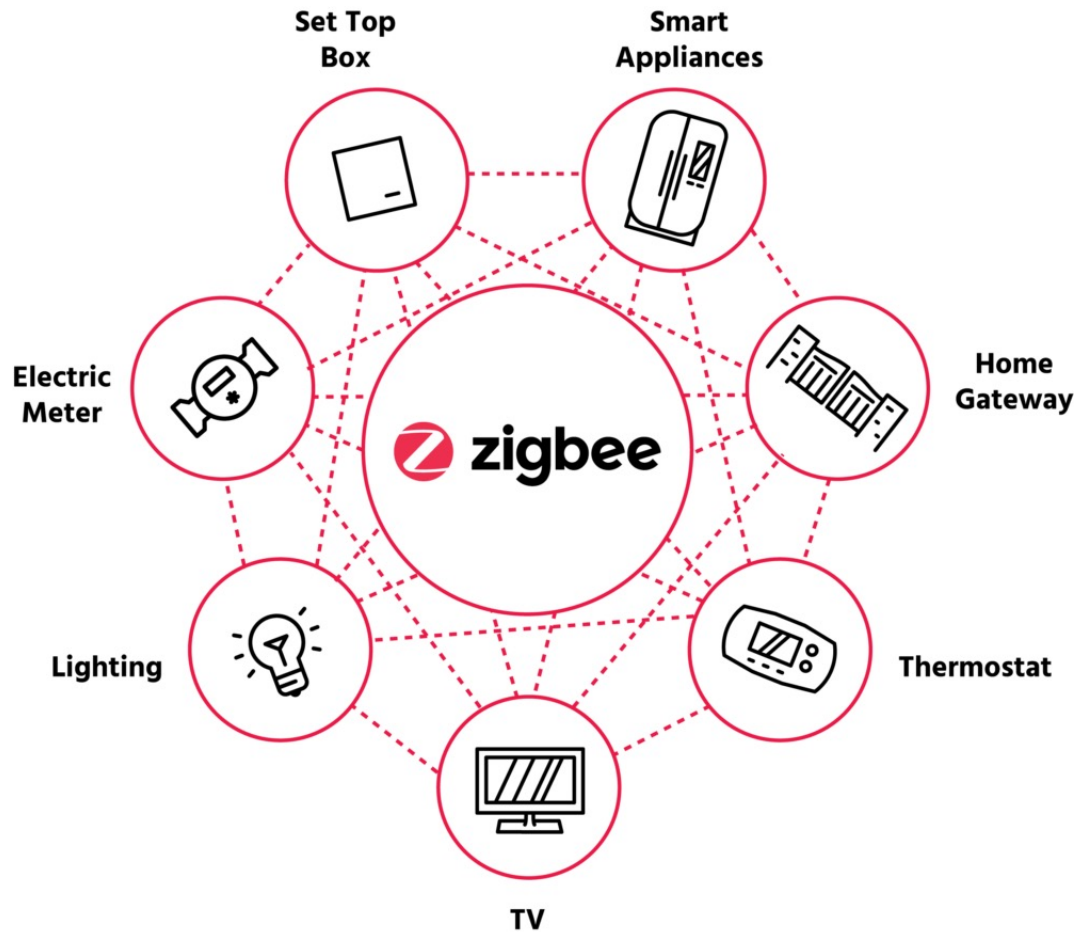
- Im Kosmos von AMQP gibt es drei Akteure und ein Objekt:
 - Die Nachricht ist das Kernelement der ganzen Kommunikation
 - Der Publisher erstellt eine Nachricht und versendet diese
 - Der Broker verteilt die Nachricht nach definierten Regeln in verschiedene Warteschlangen (Queue)
 - Der Konsument nimmt sich die Nachricht aus der Warteschlange, auf die er zugreifen kann, und bearbeitet sie
- Im Broker wird die Vermittlung der Nachricht noch einmal aufgebrochen:
 - Exchange empfängt die Nachrichten und routet die Daten in die korrekte Warteschlange
 - Die Queue wird über das „binding“ definiert. Es gibt vier Arten, wie eine Exchange Nachrichten weiterleitet



<https://www.ionos.de/digitalguide/websites/web-entwicklung/advanced-message-queuing-protocol-amqp/>

AMQP Exchange-Typen

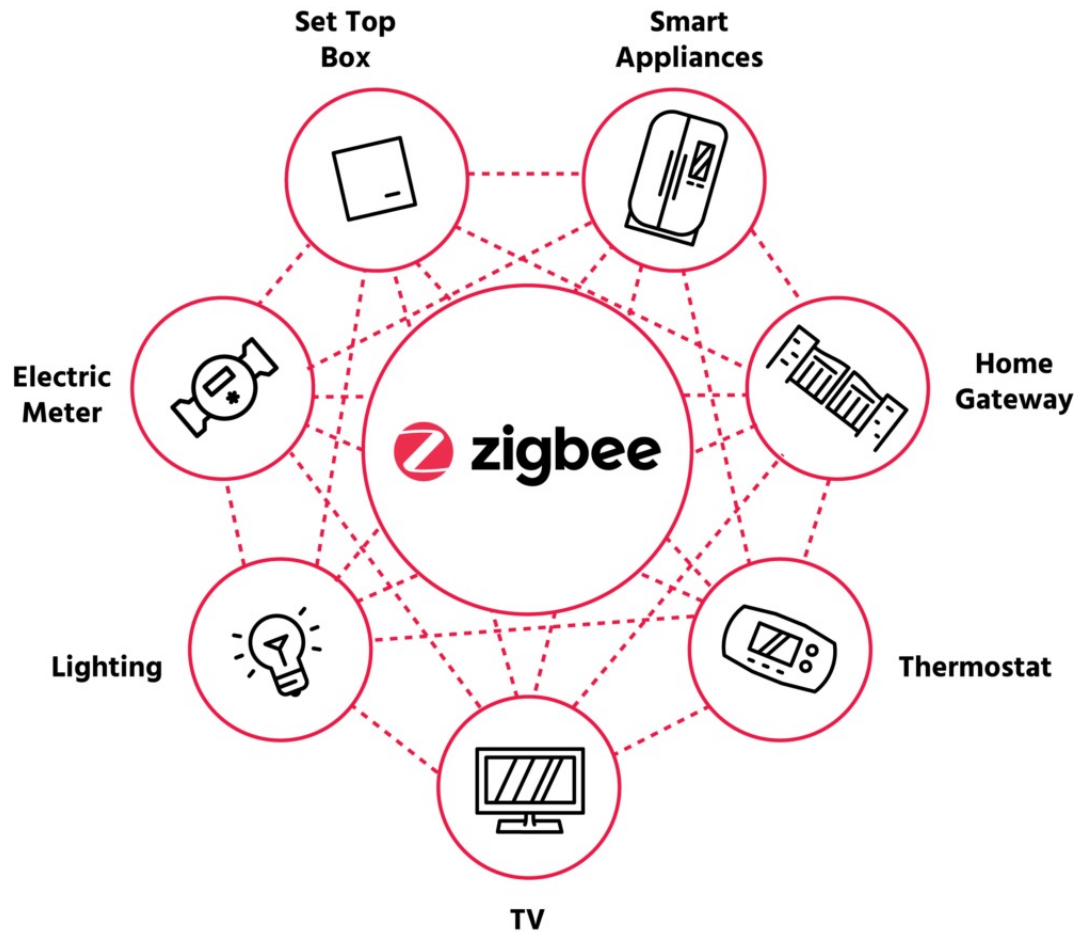
- Die erste Art, die **Direct Exchange**, schickt Nachrichten an genau einen Empfänger und arbeitet dafür mit Routing Keys.
- Ähnlich funktioniert die **Fanout Exchange**. Der Broker routet die Nachricht an alle verfügbaren Queues und vervielfältigt die Informationen dabei.
- Auf eine andere Weise funktioniert die **Topic Exchange**. Durch Platzhalter in den Keys können Nachrichten gezielt für mehrere Queues bereit gestellt werden.
- Die **Headers Exchange** schließlich agiert statt mit einem Routing Key mit dem Header einer Nachricht. Kann als Direct oder Topic Exchange genutzt werden.



<https://www.linkedin.com/learning/iot-grundlagen-fur-entwickler-innen-gerate-und-kommunikationsstandards/machine-to-machine-kommunikation-m2m-mit-mqtt?autoplay=true&un=82266650>

ZigBee

- Energieeffizientes Funkprotokoll auf OSI Layer 3 (Netzwerkschicht)
- Durch Energieeffizienz, Einfachheit in der Anwendung und hohe Flexibilität ideal für IoT
- Endgeräte verwalten das Netzwerk autark
 - Geräte treten dem Netzwerk nach ersten Einbindung selbständig bei, werden gefunden und kommunizieren darüber
- Reichweite: zwischen 10 und 20 Metern, unter idealen Bedingungen auch bis zu 100 Meter
- Bsp. Philips Hue



<https://www.linkedin.com/learning/iot-grundlagen-fur-entwickler-innen-gerate-und-kommunikationsstandards/machine-to-machine-kommunikation-m2m-mit-mqtt?autoplay=true&un=82266650>

ZigBee

- Bandbreite bis zu 250 kBit/s
- Zur Nutzung des Zigbee-Standards ist eine Bridge, Hub oder Gateway nötig
- Vorteil: Geräte funktionieren auch als Mesh Netzwerk
 - Jedes Gerät ist mit einem anderen Gerät verbunden
 - Ausfall eines Geräts kann durch Ausfallrouten abgefangen werden
 - Sehr große Reichweite durch Übertragung zwischen den Knoten
- Ermöglicht große Skalierbarkeit

Datenformate

XML	JSON	YAML
<pre><Servers> <Server> <name>Server1</name> <owner>John</owner> <created>12232012</created> <status>active</status> </Server> </Servers></pre>	<pre>{ Servers: [{ name: Server1, owner: John, created: 12232012, status: active, }] }</pre>	<pre>Servers: - name: Server1 owner: John created: 12232012 status: active</pre>

<http://www.gewi.uni-graz.at/zim/lehre/datenformat.html>

Übersicht

- Vorschrift, wie Daten in einer Datei, einem Speicherbereich oder beim Netzwerkverkehr strukturiert werden damit der Empfänger die Daten interpretieren kann

Es gibt

- binäre Datenformate (z.B. Word-Dokumente .doc)
- textbasierte Datenformate (z.B. XML, JSON, YAML, CSV, etc.)

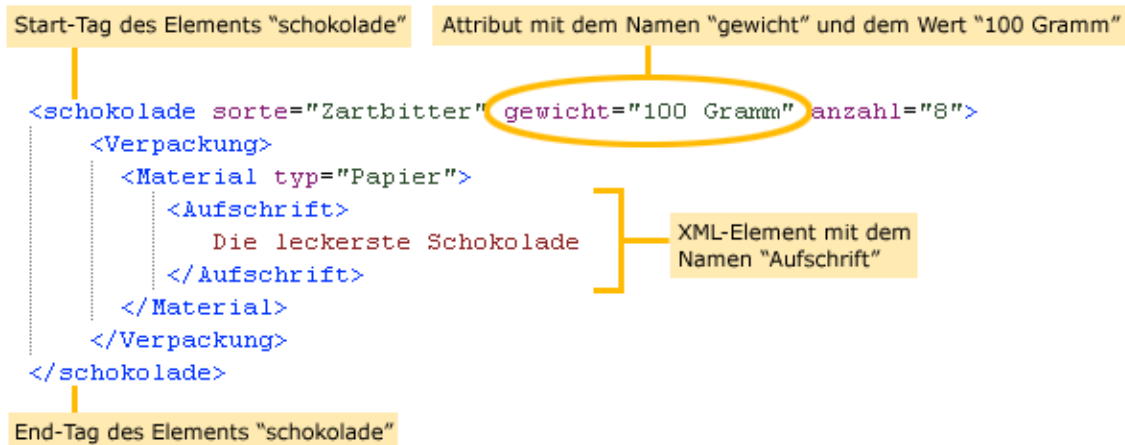

```

▼<vorlesung>
  <titel>XML und Datenbanken</titel>
  <semester>WS 2022/23</semester>
  ▼<thema>
    <name>XML-Grundlagen</name>
    <dozent>Yannick Schiele</dozent>
    <datum>26.10.2022</datum>
    <quelle>http://www.w3.org/XML/</quelle>
    <quelle>http://www.w3.org/TR/1998/REC-xml-19980210</quelle>
  </thema>
  ▼<thema>
    <name>XML-Verarbeitungsmodelle und Language Bindings</name>
    <vortragender>Christian Müller</vortragender>
    <quelle>http://www.w3.org/XML/</quelle>
  </thema>
</vorlesung>

```

XML

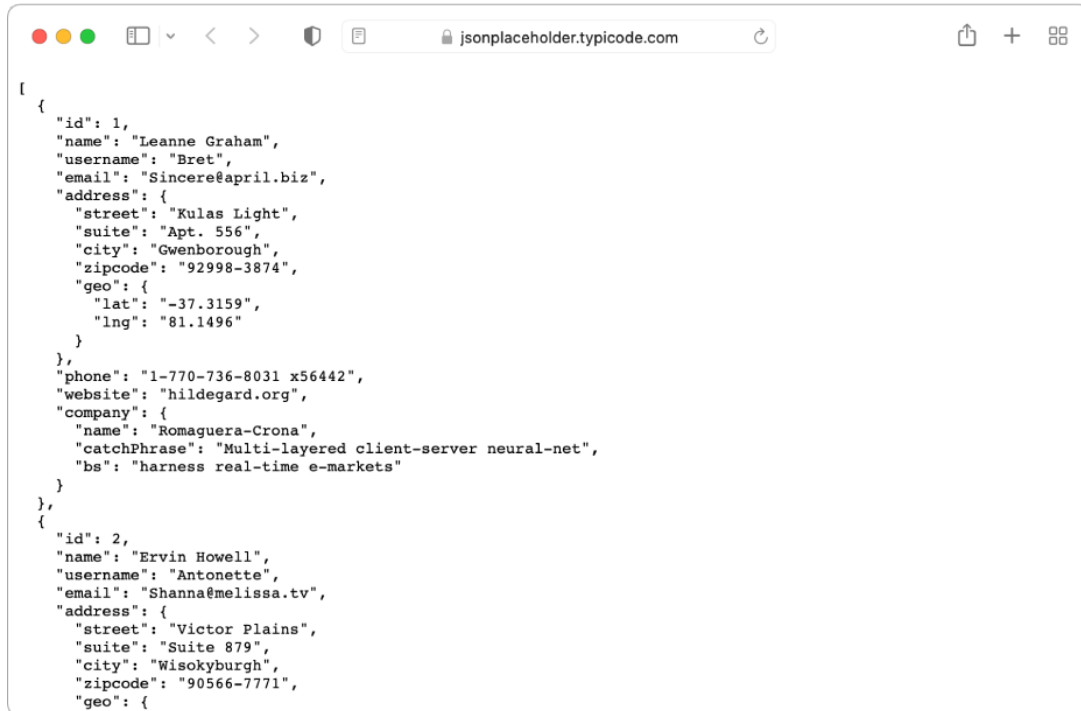
- Extensible Markup Language von 1998
- strukturiert Daten in einer Textdatei, die sowohl von Menschen als auch von Computern ausgewertet werden kann
- HTML und SVG bauen auf XML auf
- Wird häufig in der Webentwicklung verwendet



<https://www.ionos.de/digitalguide/websites/web-entwicklung/was-ist-xml>

Aufbau XML

- Elemente
 - Exakt ein Root/Start-Element
 - Beliebig viele Unterelemente und Verschachtelungen möglich
 - Namen können frei gewählt werden
 - Beginnen und enden mit Tags
- Attribut
 - Wert wird über '=' zugewiesen und muss in Anführungszeichen stehen



```
{
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    }
  }
}
```

<https://www.json.org/json-de.html>

JSON

- JavaScript Object Notation
- schlankes Datenaustauschformat,
- für Menschen einfach zu lesen und für Maschinen einfach zu parsen/generieren ist
- basiert auf einer Untermenge der JavaScript Programmiersprache
- Geringere Datenmenge im Vergleich zu XML

```

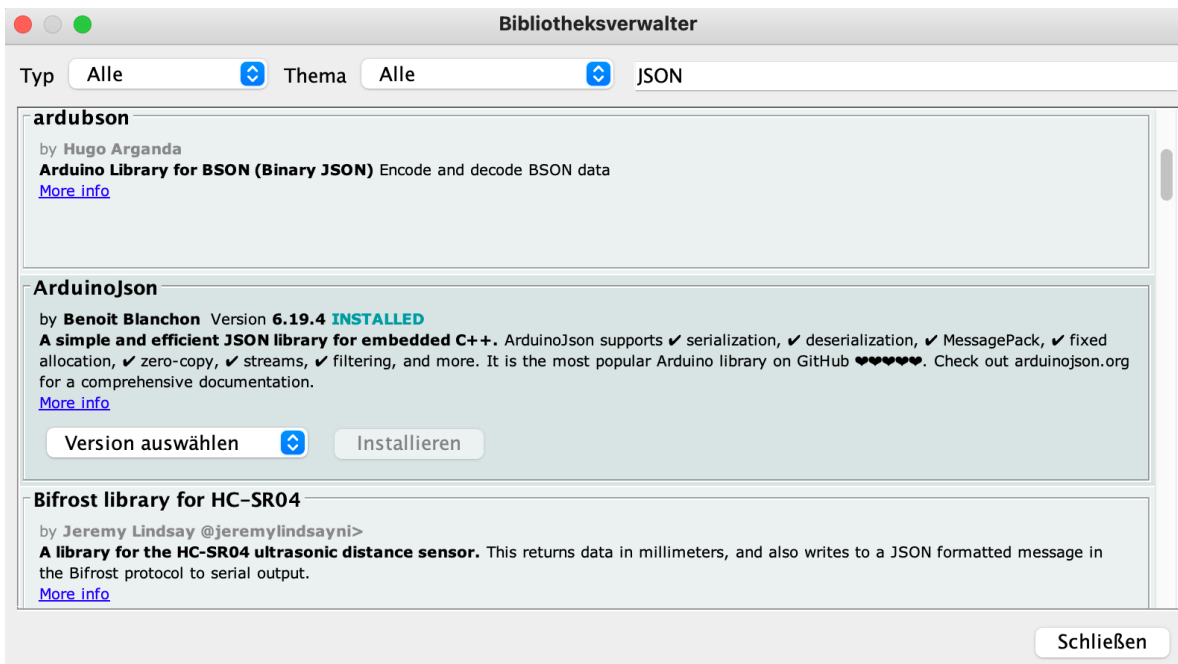
{
  "Herausgeber": "Xema",
  "Nummer": "1234-5678-9012-3456",
  "Deckung": 2e+6,
  "Waehrung": "EURO",
  "Inhaber":
    {
      "Name": "Mustermann",
      "Vorname": "Max",
      "maennlich": true,
      "Hobbys": ["Reiten", "Golfen", "Lesen"],
      "Alter": 42,
      "Kinder": [],
      "Partner": null
    }
}

```

<https://www.json.org/json-de.html>

JSON

- JavaScript Object Notation
- besteht aus Schlüssel-Wert-Paaren, die auch weiter verschachtelt werden können
- Getrennt durch einen Doppelpunkt (Schlüssel : Wert)
- Datentypen:
 - Nullwerte - null
 - Bool – true/false
 - Zahl – 0-9
 - String – “ ”
 - Array – []
 - Objekt – { }



JSON Example

<https://arduinojson.org/v6/example/>

Library: *Arduinojson* by Benoit Blanchon

- Zwei Beispiele:
 - JSON Parser
 - JSON Generator

Übung

Sensorwerte in JSON packen und auf Konsole ausgeben