MQTT

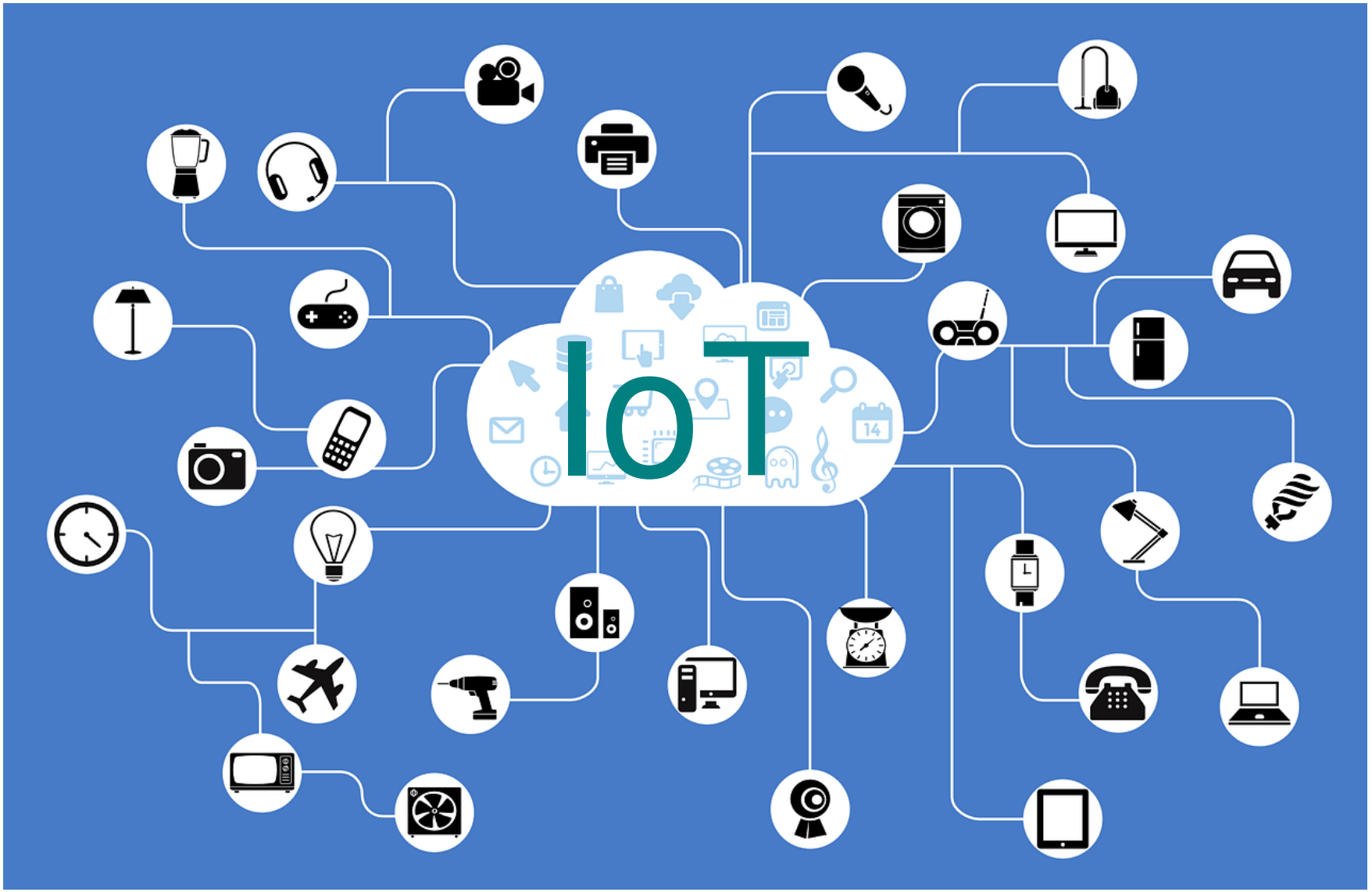
Die Sprache im Internet der Dinge (IoT)

Uwe Berger

bergeruw@gmx.net

Uwe Berger

- Beruf: Softwareentwickler
- Freizeit: u.a. mit Hard- und Software rumspielen
- Linux seit ca. 1995
- BraLUG e.V.
- bergeruw@gmx.net


IoT

Bildquelle: https://commons.wikimedia.org

Inhalt

- Was ist MQTT und wie funktioniert es?
- MQTT am Beispiel mit Mosquitto und -Clients
- Anwendungsbeispiele

MQTT allgemein

- **M**essage **Q**ueue **T**elemetry **T**ransport
- 1999 von IBM entwickelt
- Seit 2013 standardisiert über Organization of Structured Information Standards (OASIS) als IoT-Protokoll
  → es wird empfohlen MQTT ab Version 3.1 einzusetzen
- Machine-to-Machine-Kommunikation (M2M)
- Setzt auf TCP/IP auf (Port 1883 und 8883 reserviert durch Internet Assigned Numbers Authority (IANA))

MQTT allgemein

MQTT Eigenschaften:

- Einfach zu implementieren
- Leichtgewichtig und minimaler Protokoll-Overhead
- Push-Messaging
- Einstellbare QoS-Stufen (Quality of Service)
- Unabhängig von Dateninhalten (Data Agnostic)
- Nachrichten können für spätere Zustellung zwischengespeichert werden (Message Persistence)
- Verbindung zwischen Client/Server besteht ständig, auch ohne Datenverkehr (Persistent Sessions)

MQTT allgemein

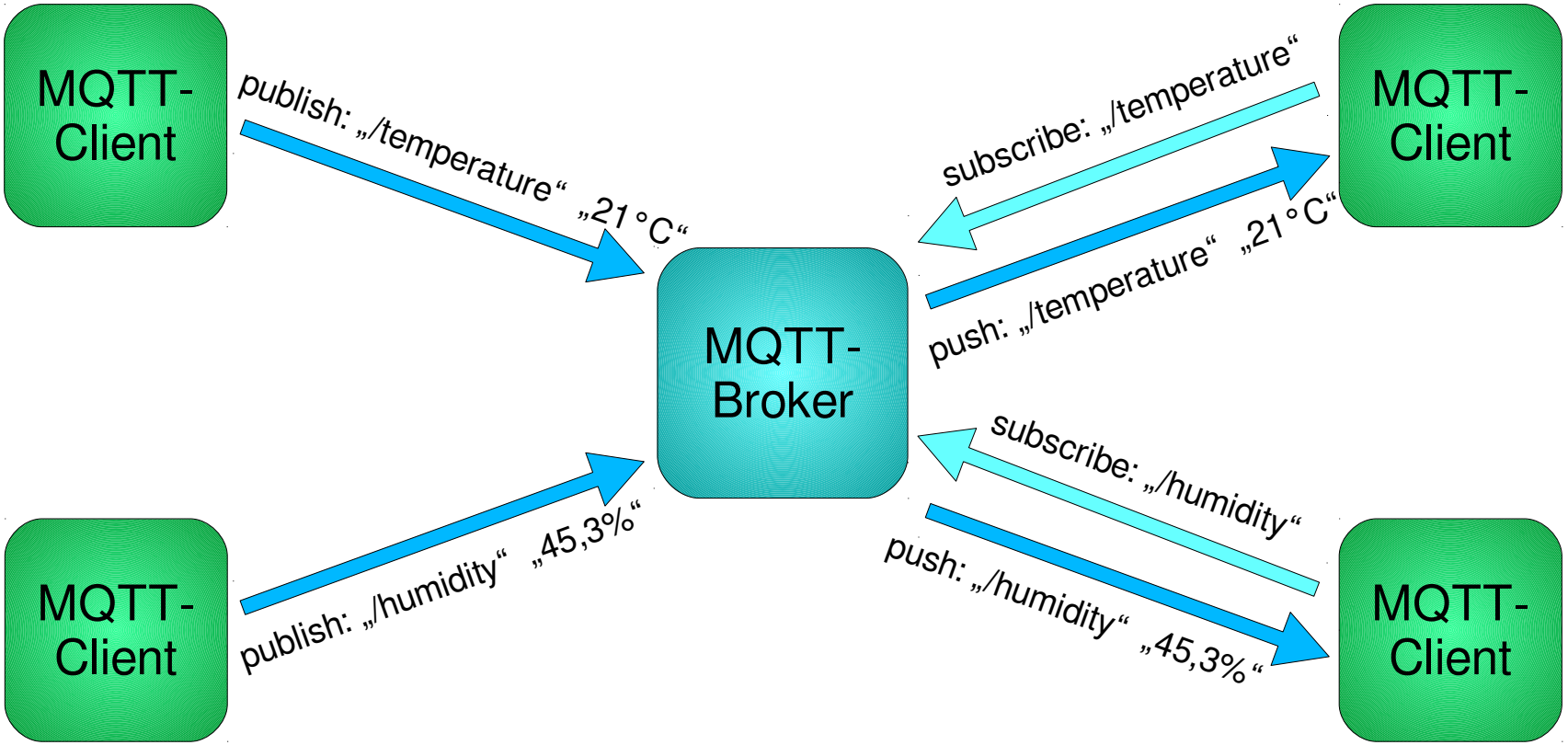
MQTT Einsatzgebiete:

- Eigentlich überall da, wo Daten zwischen Geräten unterschiedlichster Bauart ausgetauscht werden sollen...
- Aber prädestiniert:
  - auf ressourcenarmen Geräten (IoT...)
  - bei Kommunikation über (physisch) unsichere Netzwerke
  - sehr beliebt bei Hausautomatisierung o.ä.
  - ...
- Facebook-Messenger basiert auf MQTT...

MQTT Alternativen

- ...der Vollständigkeit halber, z.B.:
  - CoAP (Constrained Application Protocol)
  - XMPP (Extensible Messaging and Presence Protocol)
  - AMQP (Advanced Message Queuing Protocol)
- MQTT-SN → Abwandlung für non-TCP/IP-Netzwerke (z.B. ZigBee)

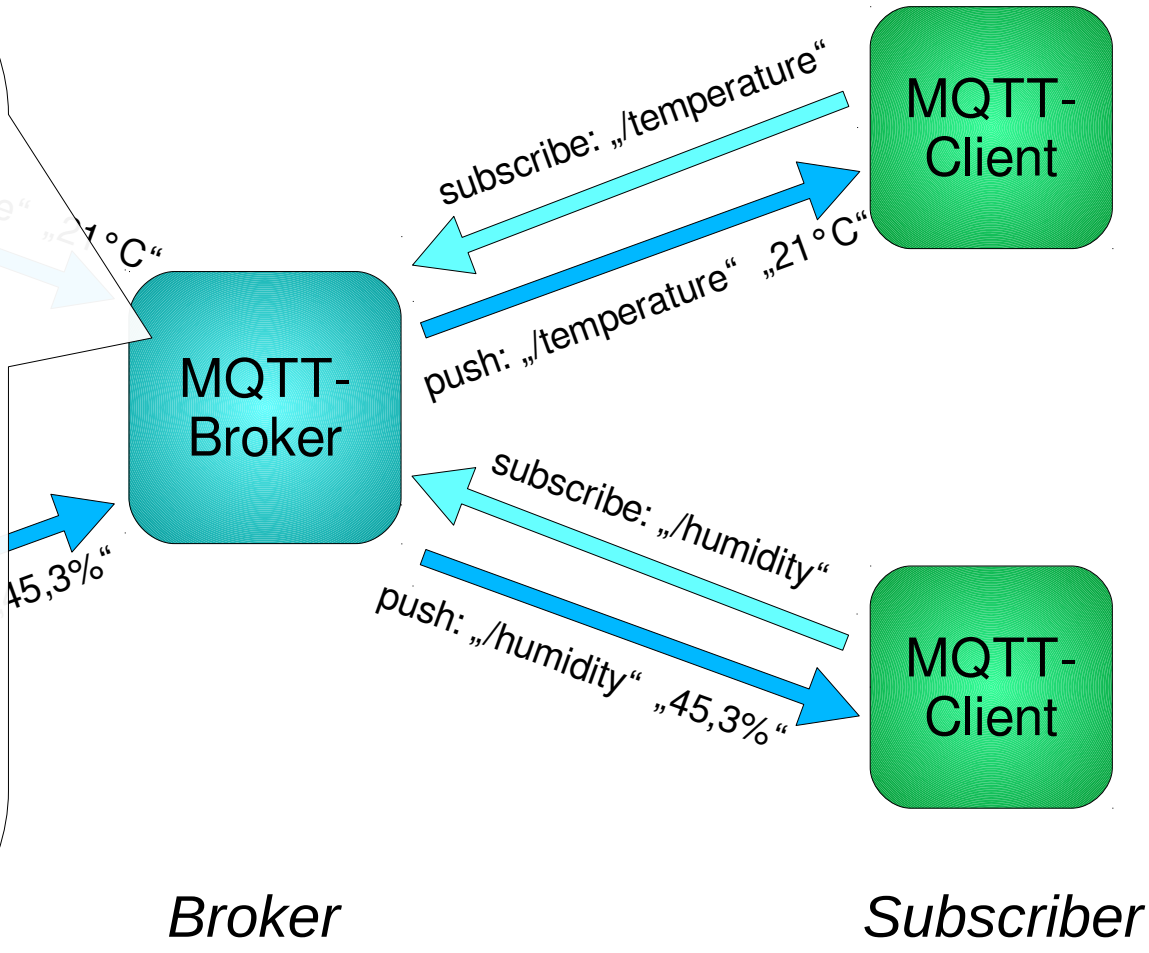
Wie funktioniert MQTT? (Rollensicht)



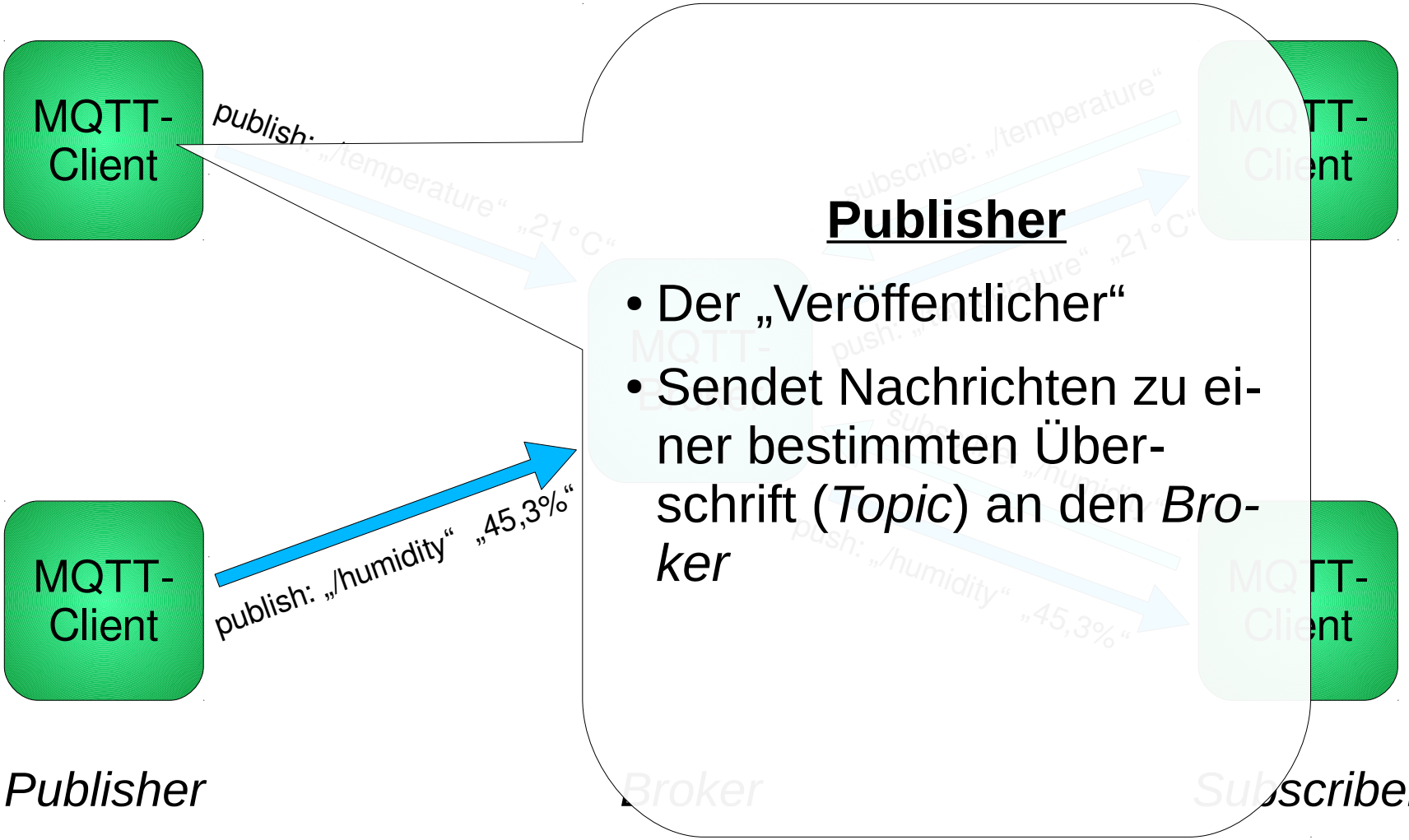
Wie funktioniert MQTT? (Rollensicht)

MQTT-Broker

- Der „Vermittler“
- Nimmt die Nachrichten der *Publisher* entgegen
- Verwaltet die Nachrichten und angemeldeten MQTT-Clients
- Verteilt die Nachrichten (entsprechend) an die *Subscriber*



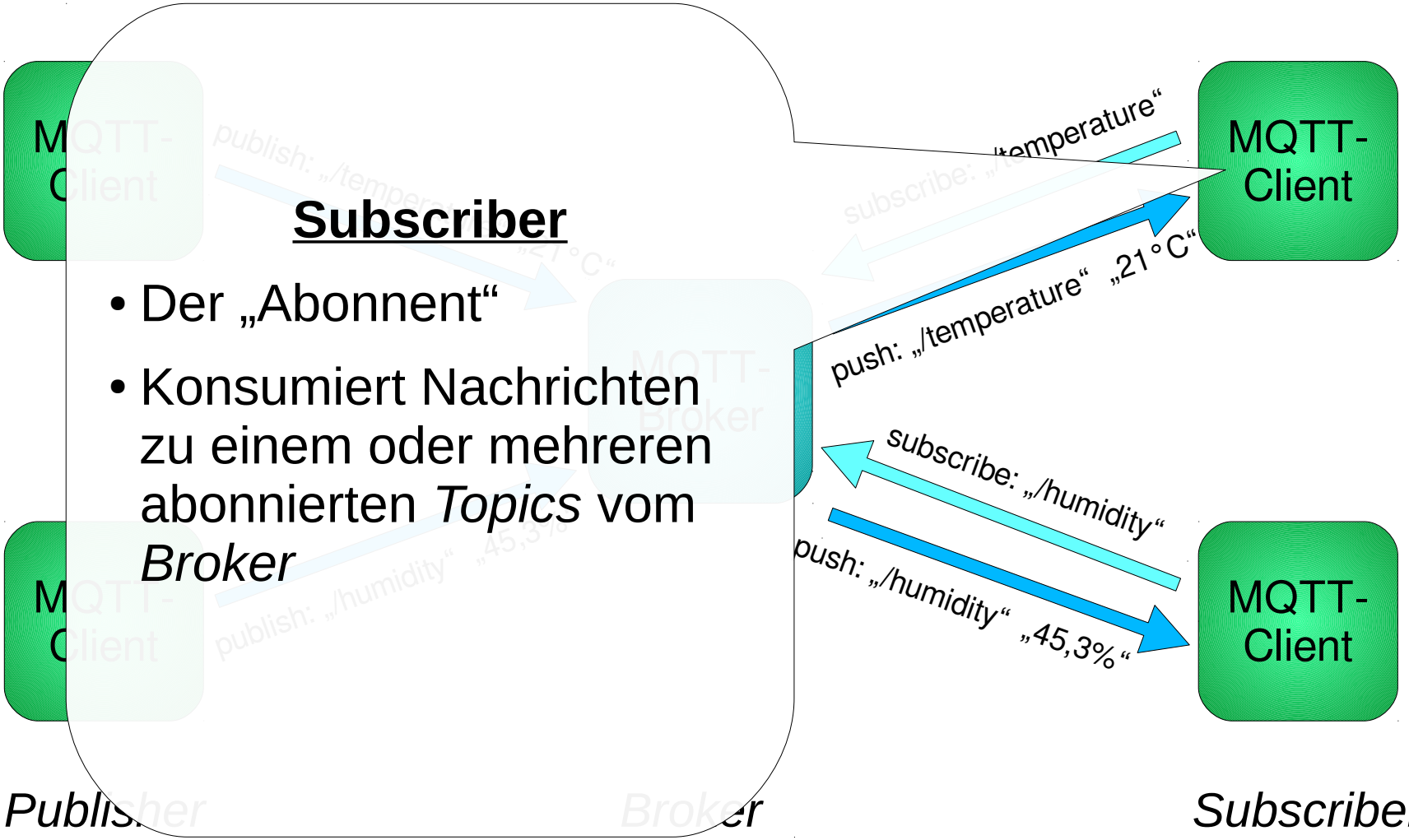
Wie funktioniert MQTT? (Rollensicht)



Publisher

- Der „Veröffentlicher“
- Sendet Nachrichten zu einer bestimmten Überschrift (*Topic*) an den *Broker*

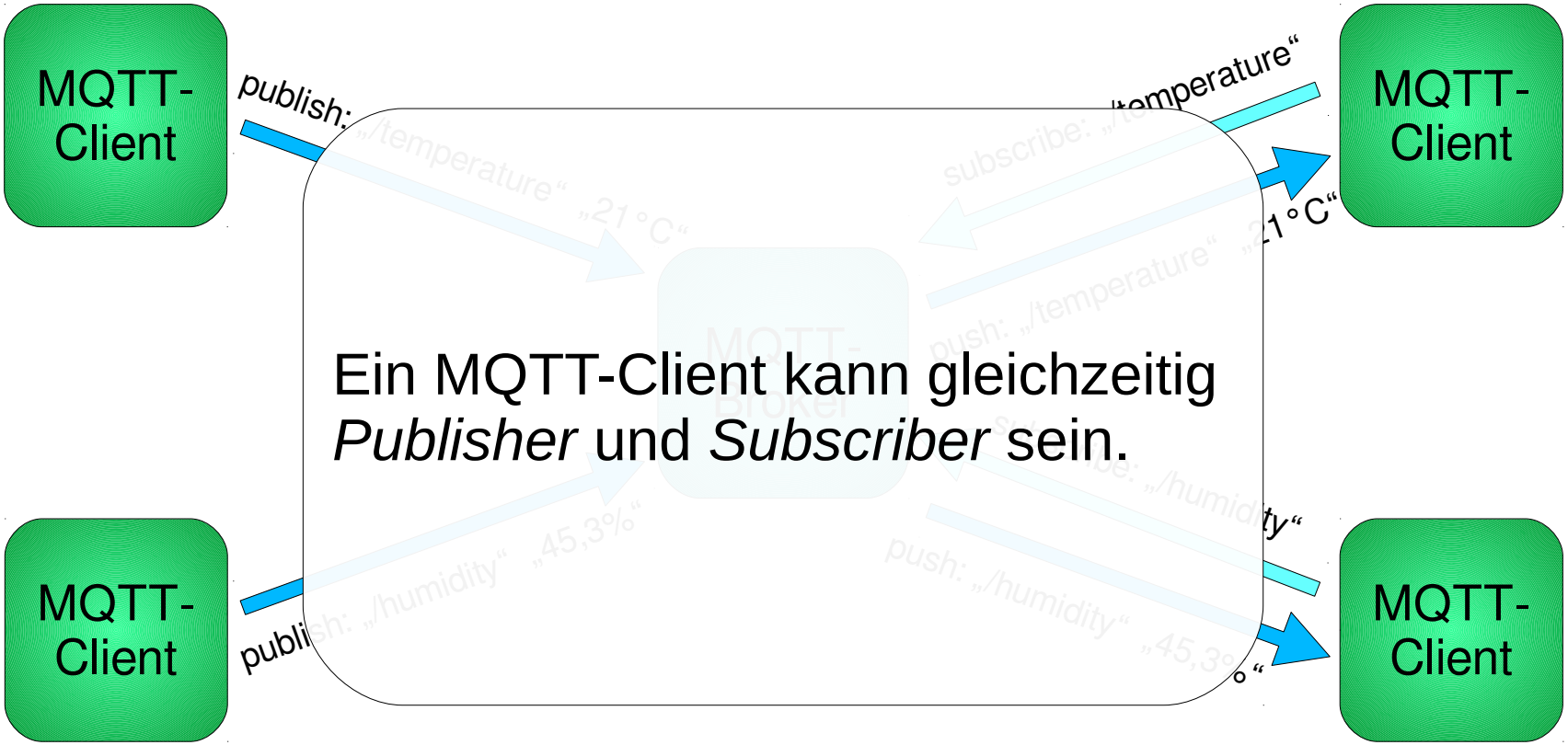
Wie funktioniert MQTT? (Rollensicht)



Subscriber

- Der „Abonnent“
- Konsumiert Nachrichten zu einem oder mehreren abonnierten *Topics* vom *Broker*

Wie funktioniert MQTT? (Rollensicht)



MQTT-Broker

Diverse Implementierungen
→ https://github.com/mqtt/mqtt.github.io/wiki/servers

U.a.:

- Moquette
- Mosquitto
- MQTTRoute
- Emqttd
- HiveMQ
- HBMQTT

Mosquitto

Ein MQTT-Broker
→ http://mosquitto.org/

- Installation (Debian, Ubuntu):

```
$ sudo apt-get install mosquitto
```

- Konfiguration

```
$ tree /etc/mosquitto/
/etc/mosquitto/
├── ca_certificates
│   └── README
├── certs
│   └── README
├── conf.d
│   └── README
└── mosquitto.conf
```

MQTT-Cients

Diverse MQTT-Tools (Subscriber, Publisher etc.) :
→ https://github.com/mqtt/mqtt.github.io/wiki/tools
→ https://www.hivemq.com/blog/category/mqtt-toolbox/

U.a.:

- mosquitto_sub, mosquitto_pub
- MQTTLens
- MQTT.fx
- mqttfs

Mosquitto-Tools

MQTT-Clients **`mosquitto_pub`**, **`mosquitto_sub`**
→ http://mosquitto.org/

- Installation (Debian, Ubuntu):

```
$ sudo apt-get install mosquitto-clients
```

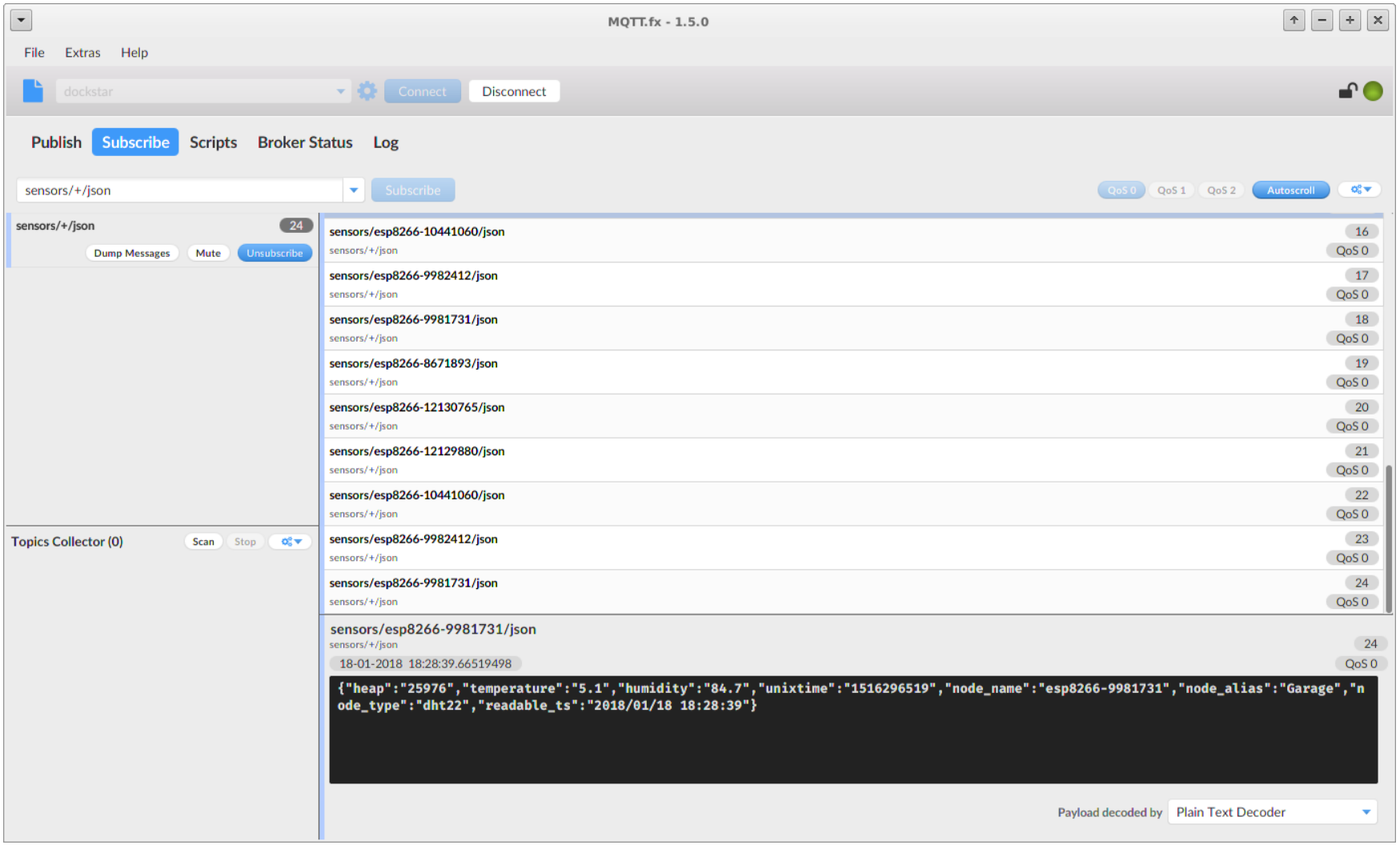
- Anwendung:

```
$ man mosquitto_sub
…
$ mosquitto_sub -h broker -t hallo/ -v


$ man mosquitto_pub
…
$ mosquitto_pub -h broker -t hallo/ -m MoinMoin
```

MQTT.fx

→ http://mqttfx.jensd.de



MQTT → Nachrichten?

- MQTT-Nachricht setzt sich zusammen aus:
  - einer Überschrift (Topic) und
  - dem eigentlichen Dateninhalt (Payload)

Beispiele:

```
sensor1/temperature   21.4°C
```

```
sensor1/humidity      42%
```

```
sensor1/json          {
                          "temp"  : "21.4",
                          "hum"   : "42",
                          "units" : ["temp" : "°C",
                                     "hum"  : "%"]
                      }
```

MQTT → Topic

- Ist praktisch der Kanal, die Mailingliste, die Usenet-Gruppe, die WhatsApp-Gruppe, über den/die eine bestimmte Art von Daten ausgetauscht wird:
  - Client (Publisher) sendet Daten zu einem Topic
  - Client (Subscriber) abonniert einen oder mehrere Topics
- Aufbau wie z.B. die Verzeichnisebenen im Filesystem:

```
/sensoren/sensor1/temperature
/sensoren/sensor1/humidity
/sensoren/sensor2/temperature
...
computername/sysinfo/mem/free
computername/sysinfo/mem/swap
...
```

MQTT → Topic-Filter

+ → Single-Level Wildcard: Platzhalter für eine Topic-Ebene

```
sensors/+/temperature    → sensors/sensor1/temperature
                         → sensors/sensor2/temperature
                         → sensors/sensor3/temperature
                         → ...
```

→ Multi-Level Wildcard: Platzhalter für mehrere Topic-Ebenen

```
sensors/sensor1/#        → sensors/sensor1/temperature
                         → sensors/sensor1/pressure
                         → sensors/sensor1/humidity/in
                         → sensors/sensor1/humitity/out
                         → ...
```

MQTT → Retained-Flag

- Normalerweise bekommen Subscriber erst dann Nachrichten zu einem abonnierten Topic zugestellt, wenn ein Publisher aktuell eine Nachricht zum Broker sendet...
- ...mit dem *Retained*-Flag gekennzeichnete Nachrichten werden vom Broker zwischengespeichert
- ...und werden vom Broker sofort ausgeliefert, wenn sich ein Subscriber entsprechend beim Broker anmeldet

MQTT → Last Will and Testament (LWT)

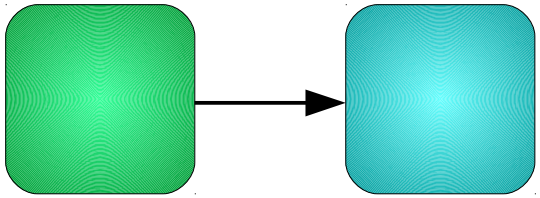
- Was soll passieren, wenn die Verbindung zwischen einem Client (Publisher/Subscriber) und Broker abbricht?
- Es kann eine Nachricht (Topic/Payload) vom Client definiert werden, die, bei Verbindungsabbruch, vom Broker publiziert wird;
- ...Beispiel:
  - Bei Verbindungsaufbau:
    - LWT definieren: `„sensor/status/“ „OFF“`
    - Publish: `„sensor/status/“ „ON“`
  - Bei Verbindungsabbruch:
    - Publish: `„sensor/status/“ „OFF“`

MQTT → Quality of Service (QoS)

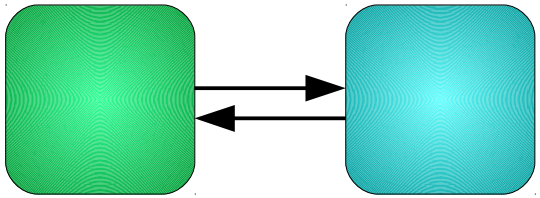
- QoS definiert/gewährleistet Garantien bezüglich der Zustellung von Nachrichten
- MQTT-Protokoll definiert 3 Stufen:
  - 0 → höchstens einmal (...aber auch keinmal!)
  - 1 → mindestens einmal (...aber auch mehrmals!)
  - 2 → exakt einmal
- QoS gilt in beide Richtungen:
  - publish: Client → Broker
  - subscribe (push): Broker → Client
- Der Client (Subscriber/Publisher) gibt den QoS-Level vor

MQTT → Wann welcher QoS?

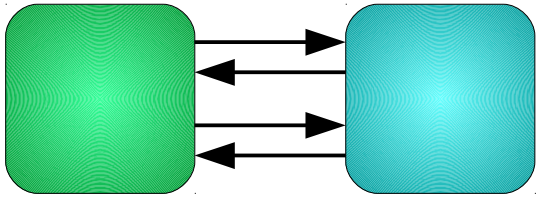
- QoS 0
  - verlorene Nachrichten sind tolerierbar, (...kommen aber selten vor, da Netzwerk stabil...)



- QoS 1
  - Nachricht muss ankommen
  - Netzwerk-Overhead von QoS 2 ist zu hoch
  - Duplikate sind tolerierbar



- QoS 2
  - Nachricht muss ankommen
  - Duplikate sind nicht tolerierbar



MQTT – interne Broker-Statistik

- $-Zeichen reserviert für Topics zu Informationen zum Broker
- $SYS → Broker-Statistiken
- Details teilweise abhängig von Broker-Implementierung; u.a.:

```
$SYS/broker/clients/…
$SYS/broker/messages/…
$SYS/broker/subscriptions/…
$SYS/broker/publish/…
$SYS/broker/bytes/…
$SYS/broker/load/…
...
```

MQTT - Sicherheit

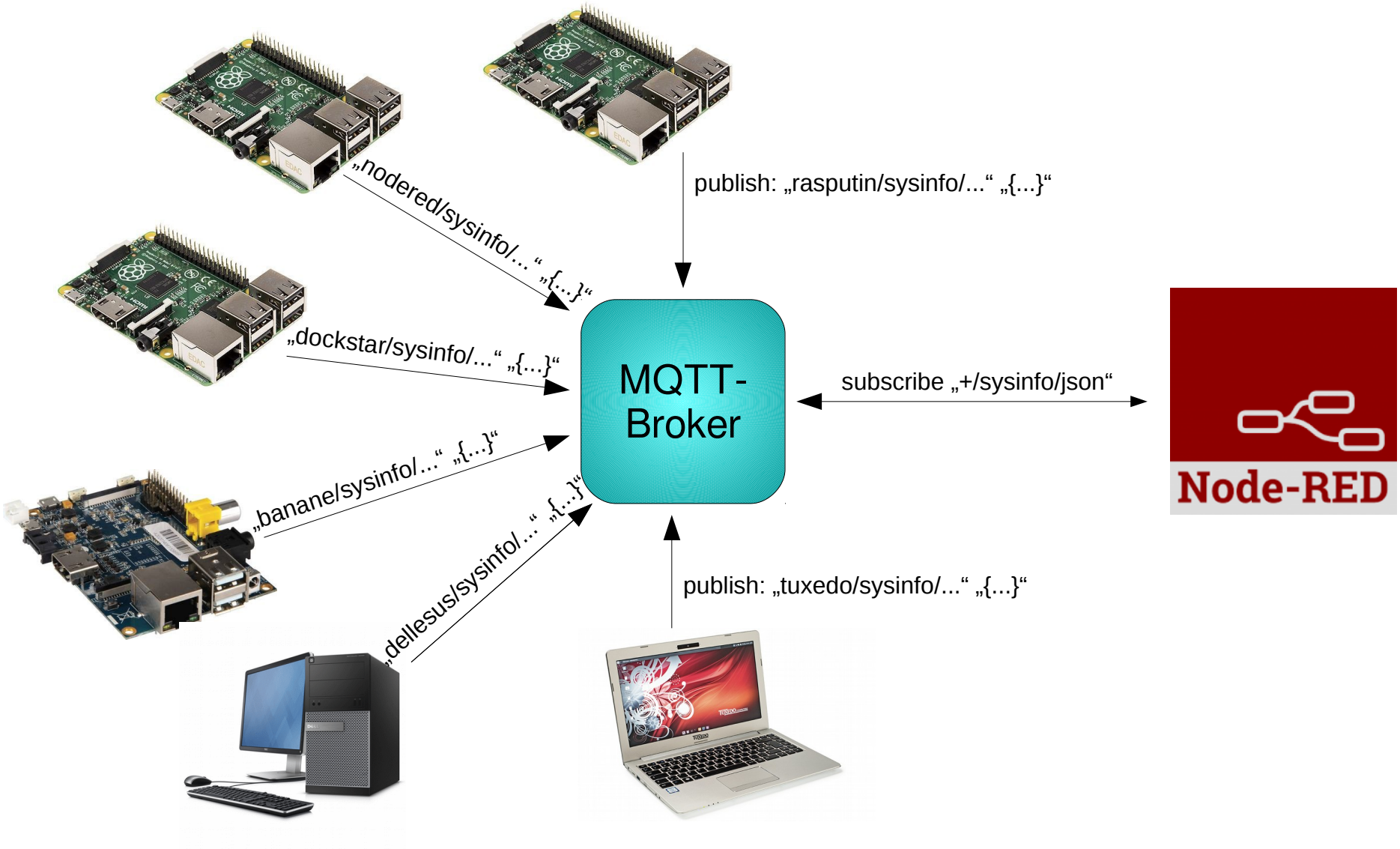
- Authentifizierung gegenüber Broker via User/Passwort einstellbar (...granulierbar bis auf Topic-Ebene)
- SSL/TLS-Support zwischen Clients und Broker konfigurierbar

MQTT für Softwareentwickler

MQTT-Bibliotheken, -Erweiterungen, -Module, -Firmware:
→ https://github.com/mqtt/mqtt.github.io/wiki/libraries

- U.a. für:
  - C/C++ (z.B. libmosquitto, libmosquittopp)
  - Java (z.B. Eclipse Paho Java, moquette)
  - Javascript, Node.js (z.B. mqtt.js)
  - PHP (z.B. phpMQTT, Mosquitto-PHP)
  - Python (z.B. Eclipce Paho Python)
  - Nodemcu (...Lua-Firmware für ESP-Chips)
  - Tcl (→ http://wiki.tcl.tk/48822)
- MQTT „unter der Haube“:
  → http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

Anwendungsbeispiel: sysinfo2mqtt

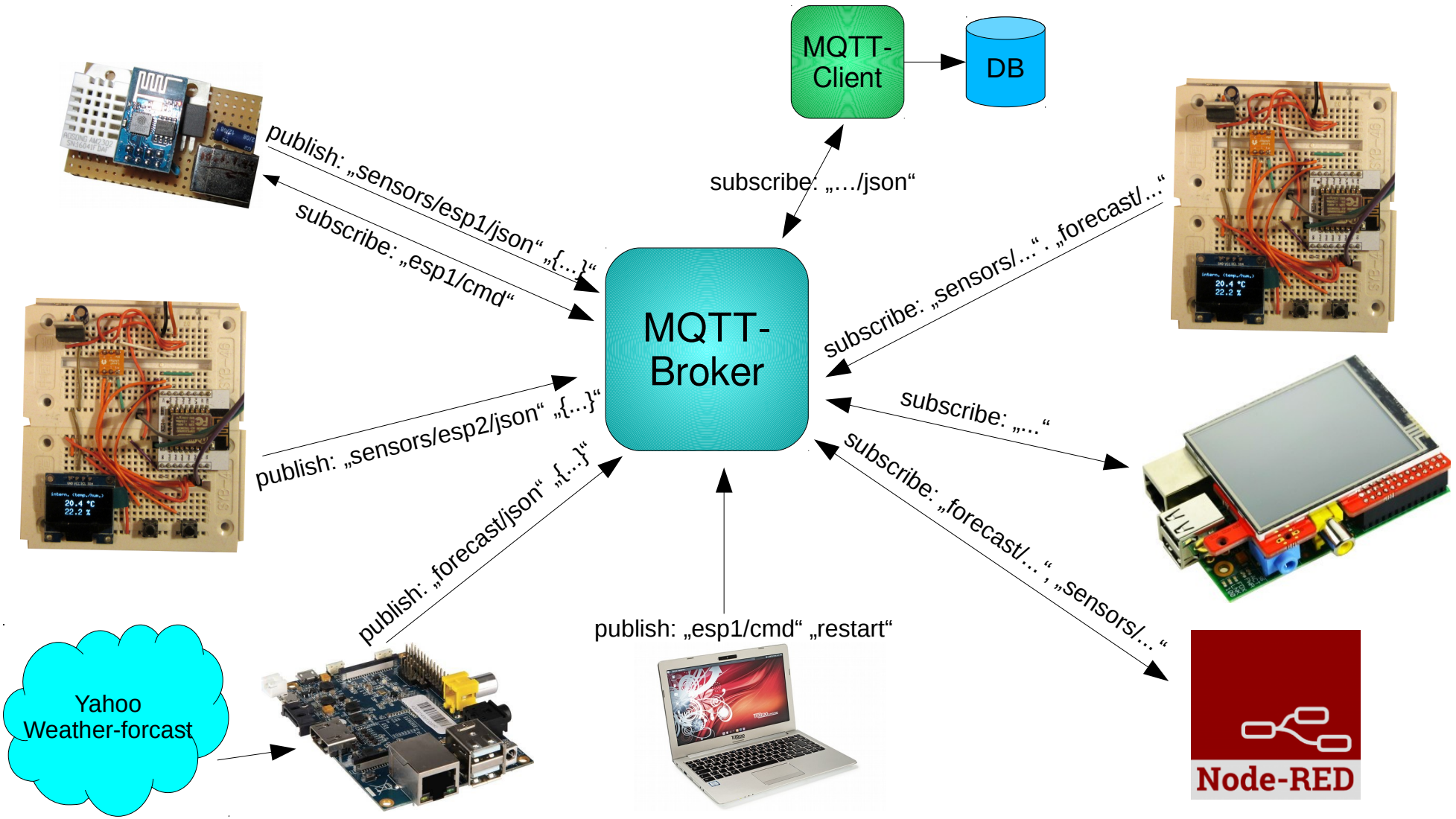


Anwendungsbeispiel: sysinfo2mqtt

computers

			load			ram [kByte]				swap [kByte]		
hostname	last seen	uptime	1min	5min	15min	free	share	buffer	total	free	total	processes
samsung	2018/01/15 07:39:01	0 days, 0:25:54	0.28	0.16	0.31	1038512	37036	154228	2051736	2988028	2988028	384
dockstar	2018/01/18 17:38:01	154 days, 0:31:39	0.01	0.04	0.05	294052	232	144136	996700	102396	102396	114
dellesus	2018/01/12 20:11:01	0 days, 0:07:57	1.76	3.06	1.80	1638072	14248	503336	3980264	7807552	7807552	417
nodered	2018/01/18 17:38:01	25 days, 19:00:17	0.02	0.01	0.00	587248	48424	88624	961752	102396	102396	130
rasputin	2018/01/18 17:13:01	3 days, 10:07:13	0.06	0.13	0.14	26064	0	95628	447416	102396	102396	83
banane	2018/01/18 17:38:01	106 days, 0:28:00	0.87	0.34	0.20	201084	0	205956	993780	524284	524284	82
tuxedo	2018/01/18 17:37:41	4 days, 2:03:07	0.55	0.28	0.25	5663664	591644	223092	16305104	8388604	8388604	697

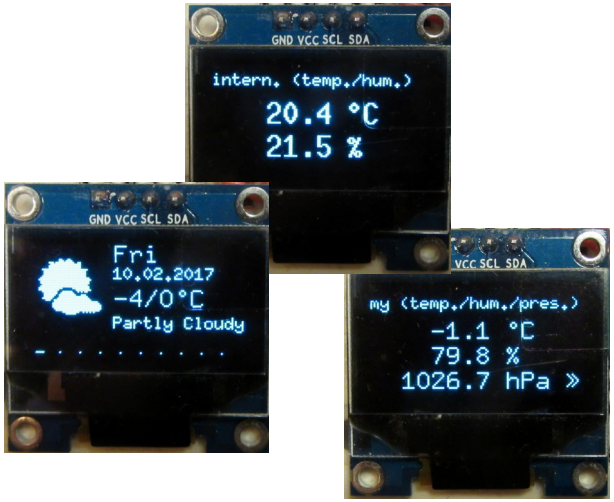
Anwendungsbeispiel: Sensornetzwerk

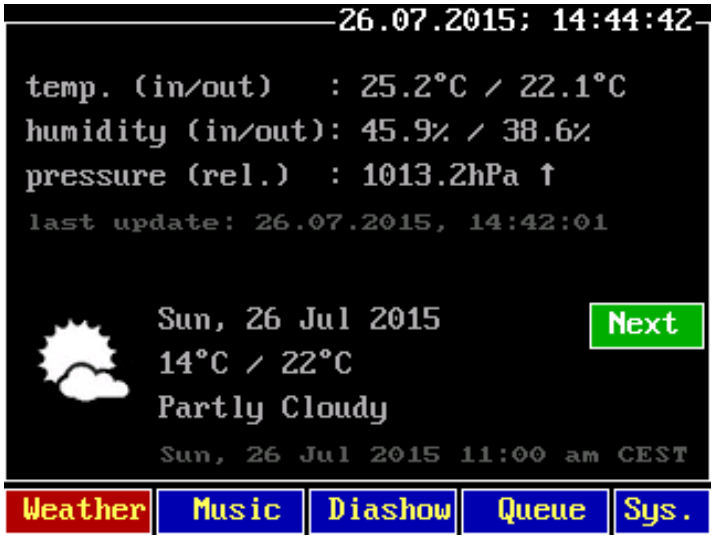


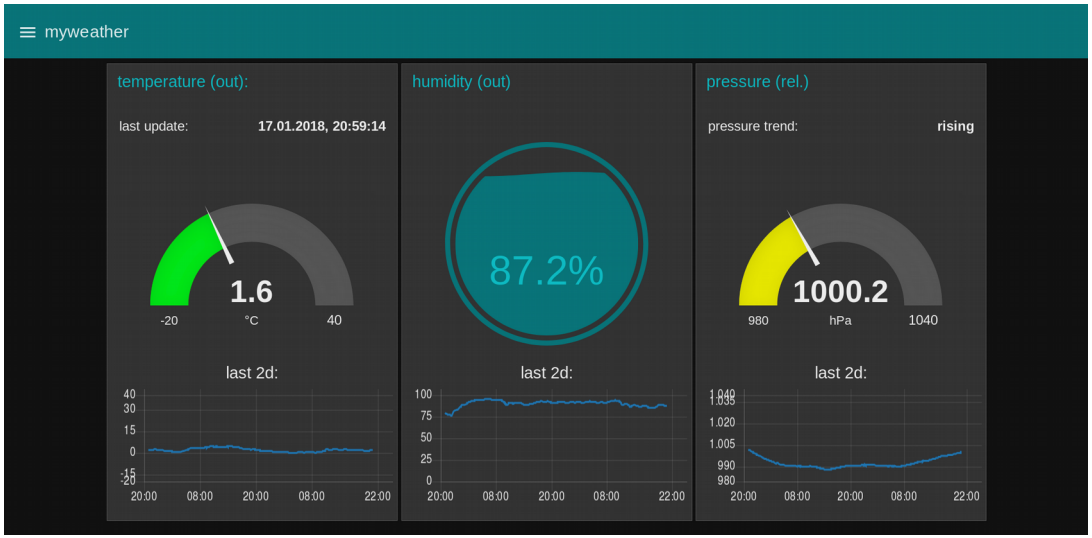
Anwendungsbeispiel: Sensornetzwerk

sensors

name	type	last seen	temp. [°C]	hum. [%]	press. [hPa]	drewp. [°C]
esp8266-9982412	dht22	2018/01/17 21:16:35	25.7	32.7		
esp8266-10441060	dht22	2018/01/17 21:17:32	19.7	25.2		
esp8266-8671893	dht22	2018/01/17 21:17:17	18.5	36.6		
esp8266-9981731	dht22	2018/01/17 21:16:35	4.6	81.2		
esp8266-12130765	dht22	2018/01/17 21:17:26	13.5	40.4		
esp8266-12129880	bme280	2018/01/17 21:17:27	21.8	33.2	1002.5	4.9







Weiterführende Informationen

- http://mqtt.org/
- http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
- http://mosquitto.org/
- https://github.com/mqtt/mqtt.github.io/wiki
- https://www.hivemq.com/mqtt-essentials/
- https://github.com/boerge42/mqtt_clients
- https://github.com/boerge42/nodemcu_scripts

Fragen...?