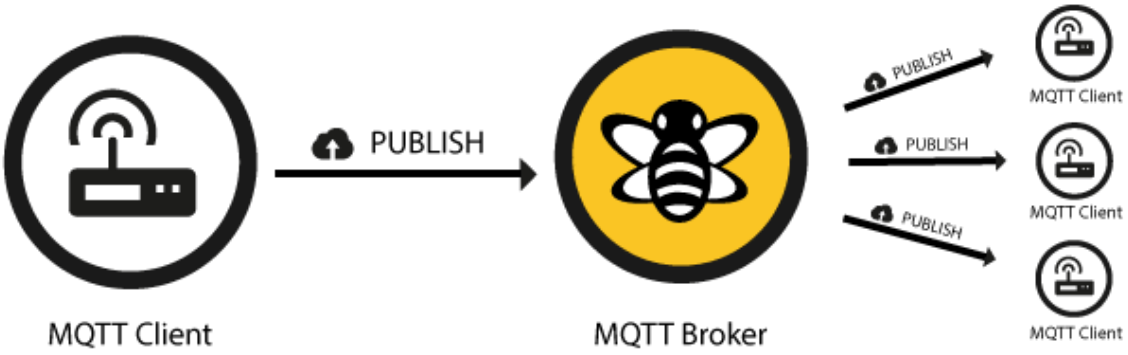
Übung MQTT und Visualisierung mit ioBroker

In dieser Übung sollen die grundlegenden Funktionen von ioBroker in Zusammenhang mit MQTT und Visualisierung verstanden und geübt werden.

Grundlagen MQTT-Protokoll

Um mit MQTT arbeiten zu können ist ein sogenannter MQTT-Server bzw. MQTT-Broker erforderlich.



Im gesamten System gibt es nur einen **Broker(Server)** alle anderen sind dann Clients. Die **Clients** können sowohl Nachrichten senden (**publish**) als auch Nachrichten empfangen (**subscribe**).

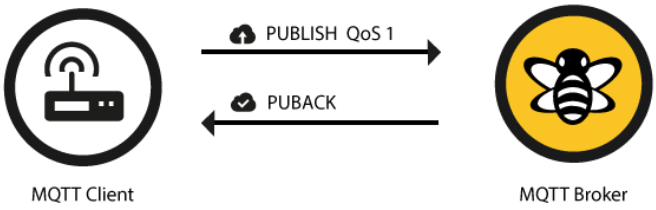
Die verschiedenen Nachrichten (z.B. Temperaturwerte) werden in sogenannten **Topics** „zusammengehalten“:

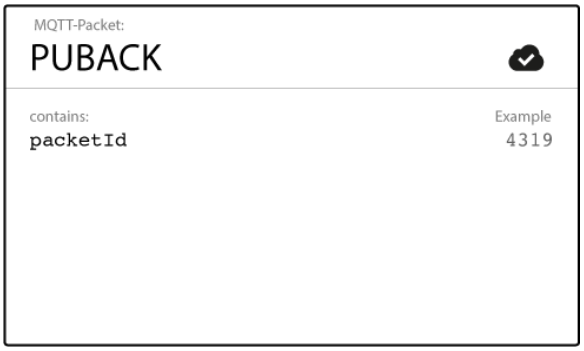
Ein Topic lässt sich als schwarzes Brett mit eindeutiger Inventarnummer interpretieren, wobei sich alle anmelden müssen, die Informationen auf dem Brett veröffentlichen wollen, ebenso diejenigen, die Nachrichten lesen wollen.

Darüber hinaus bietet dieses Protokoll auch die Möglichkeit **QoS** (**Q**uality **o**f **S**ervicezu definieren).

Bei der der niedrigsten Stufe 0 handelt es sich im Prinzip um eine *fire'n'forget*-Semantik. Es gibt also keine Garantie, dass die Nachricht überhaupt ankommt:

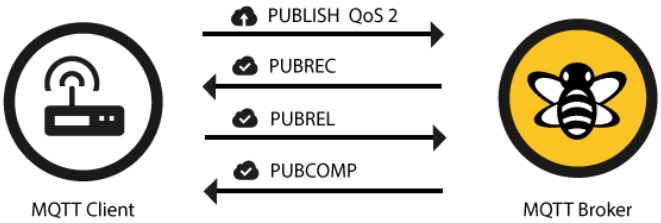
Bei QoS-Level 1 ist sichergestellt, dass die Nachricht mindestens einmal in der Topic-Queue landet (*At-least-once*-Semantik):

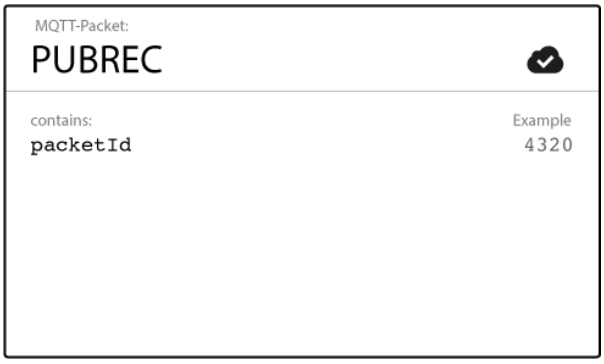


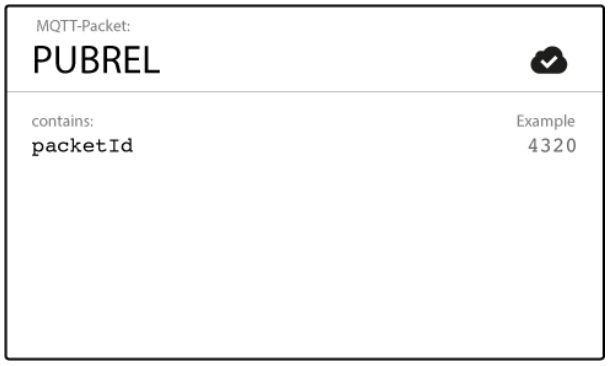


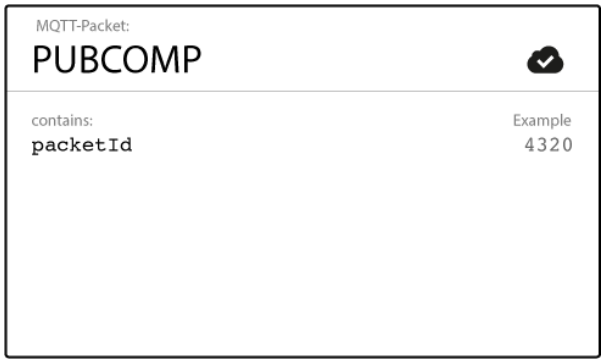
The association of PUBLISH and PUBACK is done by comparing the packet identifier in each packet. If the PUBACK isn't received in a reasonable amount of time the sender will resend the PUBLISH message.

In der höchsten Stufe 2 garantiert der Broker sogar *exactly-once*: die Nachricht wird also genau einmal abgelegt, nicht öfter und nicht weniger:









When the flow is completed both parties can be sure that the message has been delivered and the sender also knows about it.

Whenever a packet gets lost on the way, the sender is responsible for resending the last message after a reasonable amount of time. This is true when the sender is a MQTT client and also when a MQTT broker sends a message. The receiver has the responsibility to respond to each command message accordingly.

Clients können zudem ihren letzten Willen spezifizieren. Das ist eine Nachricht, die der Broker an ein Topic sendet, wenn die Verbindung mit dem betreffenden Client unvorhergesehen abgebrochen wurde. (**LWT L**ast **W**ill and **T**estament)

Das LWT wird beim ersten Verbindungsaufbau festgelegt:

MQTT-Packet:

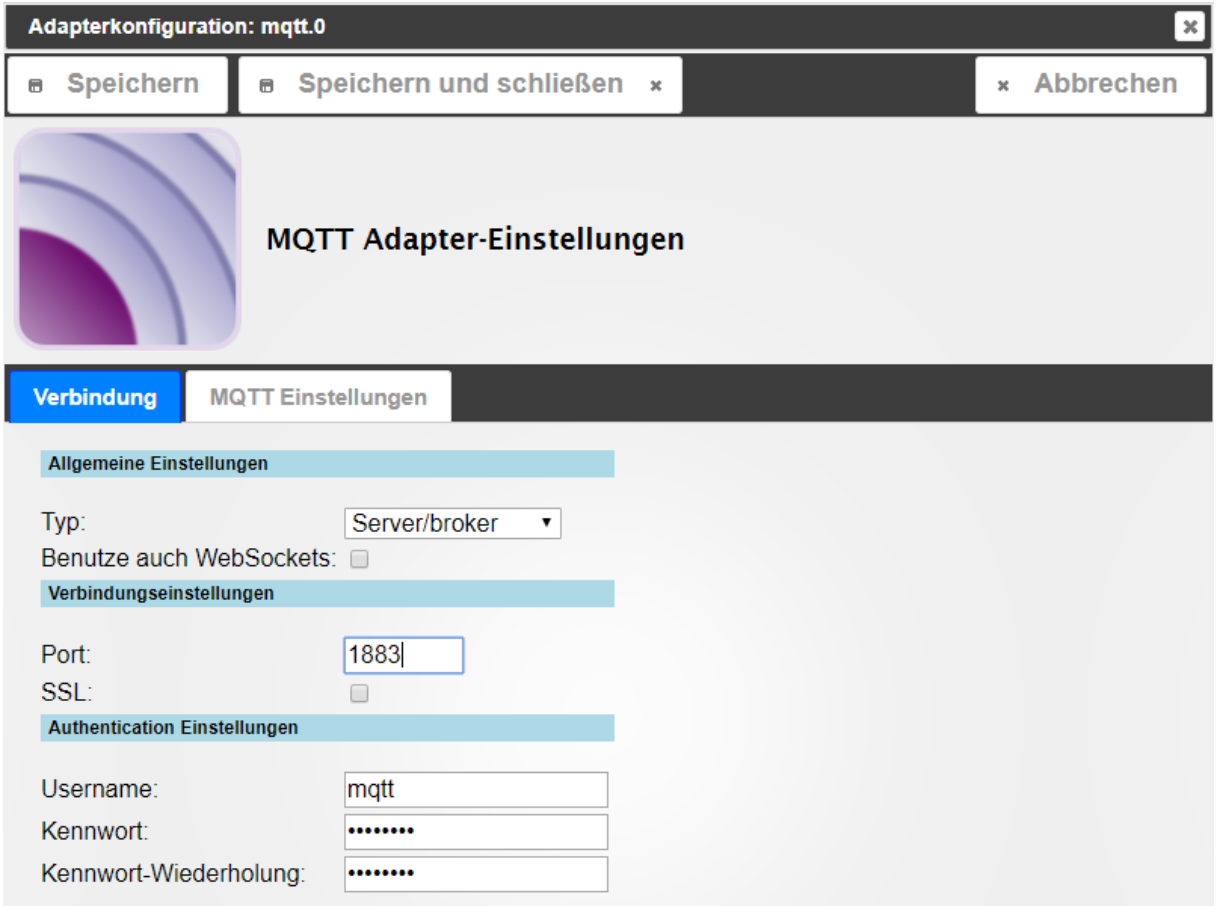
CONNECT

contains:	Example
clientId	"client-1"
cleanSession	true
username (optional)	"hans"
password (optional)	"letmein"
lastWillTopic (optional)	"/hans/will"
lastWillQos (optional)	2
lastWillMessage (optional)	"unexpected exit"
lastWillRetain (optional)	false
keepAlive	60

Verwendung in ioBroker

Generell wird in ioBroker nur QoS 0 unterstützt!
Der MQTT-Adapter von ioBroker kann sowohl als Server als auch al Client konfiguriert werden:

Server

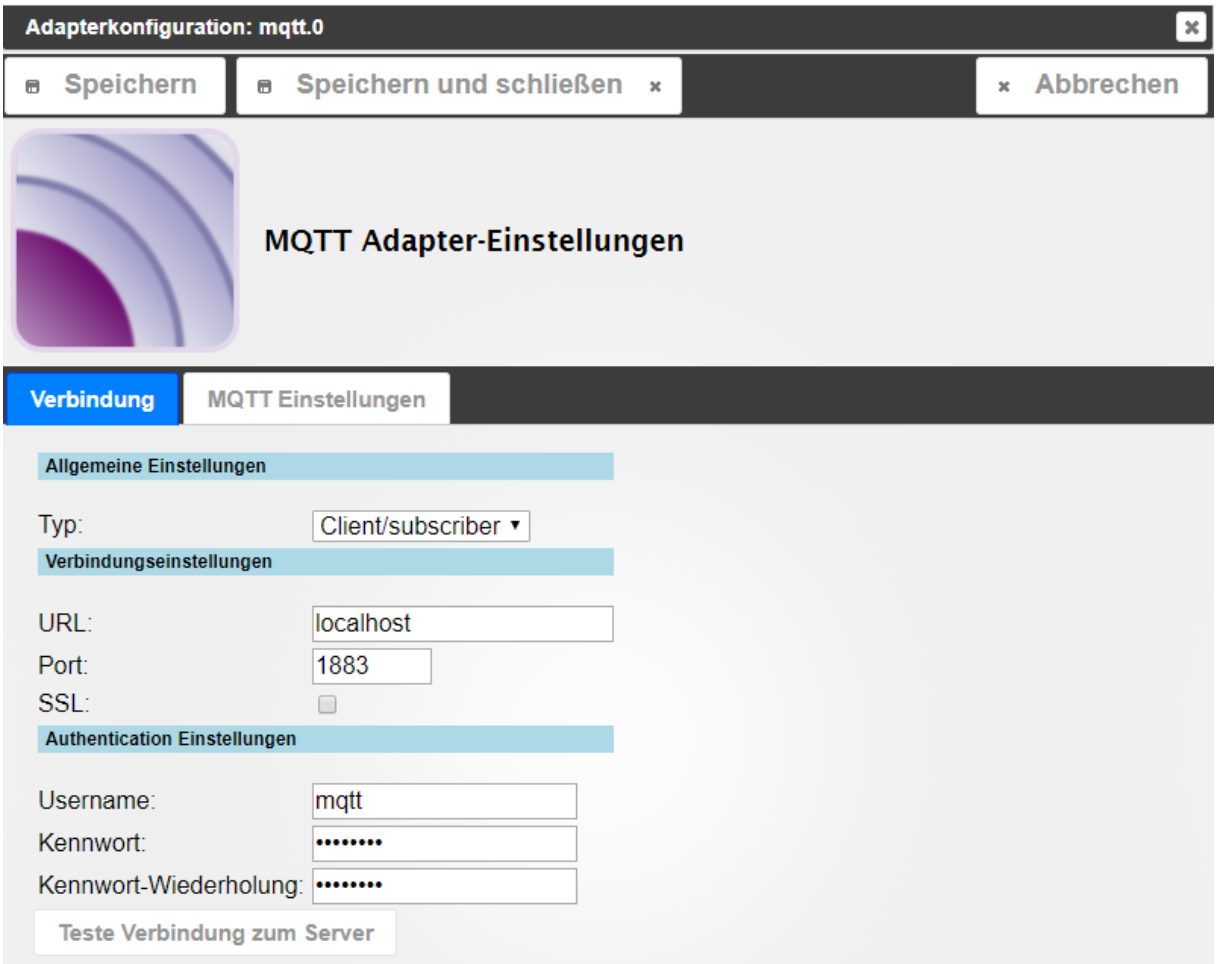


Den **Port** lässt man im Normalfall auf 1883, außer es ist bereits ein anderer Dienst auf diesem Port aktiv.
SLL (Verschlüsselte Übertragung) ist zwar prinzipiell sinnvoll, da aber die verwendeten ESP8266

Module diese Verschlüsselung aufgrund von Speicherknappheit nicht unterstützen, darf dies nicht ausgewählt werden.

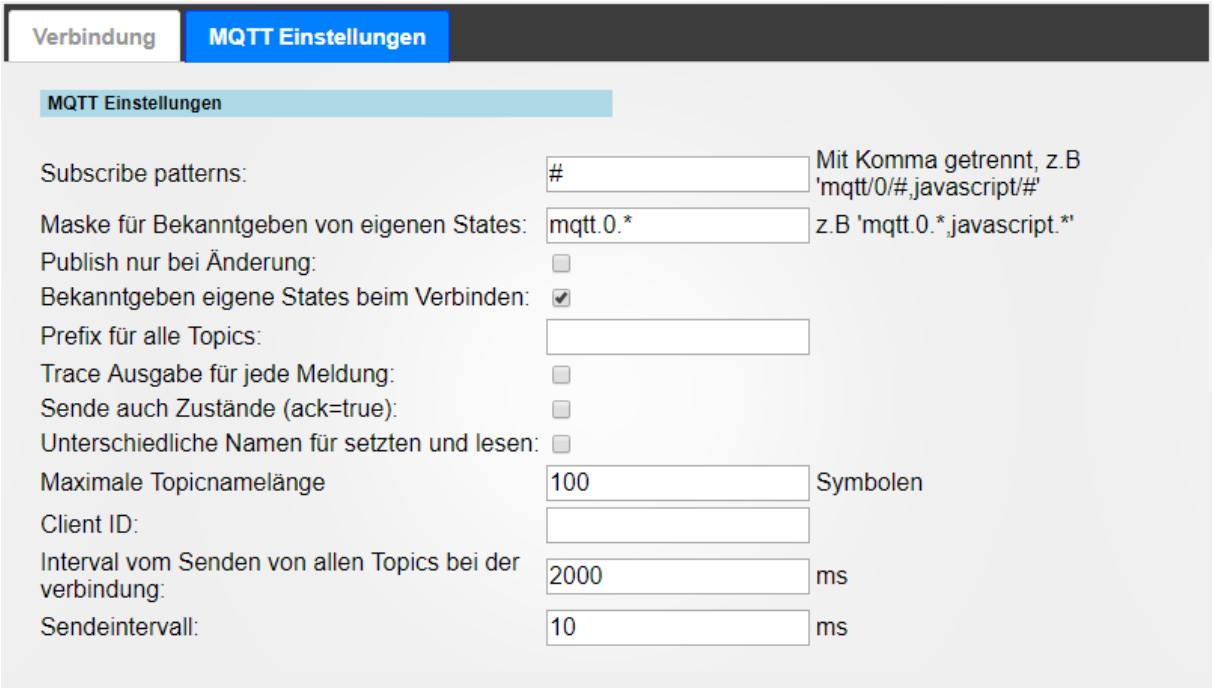
Username und **Kennwor**t können frei vergeben werden. (Diese Einstellung mach genaugenommen nur Sinn, wenn die Option SSL angewählt wurde)

Client



In dem, hier zusätzlichen Eintrag **URL** ist die Adresse des MQTT-Server (Brokers) einzutragen. (z.B.:192.168.66.200)

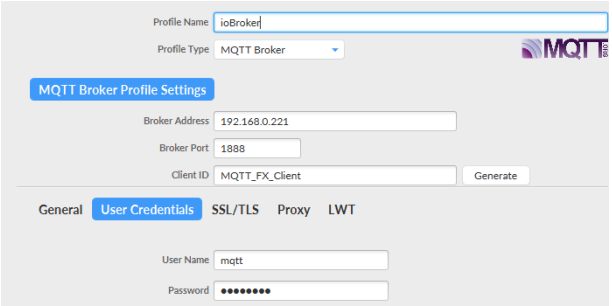
Unter der Registrierkarte MQTT Einstellungen kann man festlegen, was abonniert und gesendet werden soll:



Testbetrieb mit MQTT.fx

Um die grundlegende Funktionsweise zu testen, verwenden wir einen MQTT-Client.

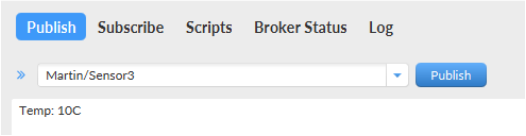
In den Verbindungseinstellungen tragen wir die Daten unseres MQTT-Brokers ein:



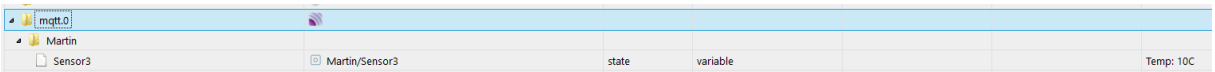
Danach verbinden wir uns mit dem Broker mittels „Connect“-Button:

In der Auflistung der Adapter unter ioBroker sollte nun der MQTT-Adapter von gelb auf grün wechseln – das ist das Zeichen, dass zumindest ein Client verbunden ist.

Wir schicken nun eine Nachricht an ioBroker:

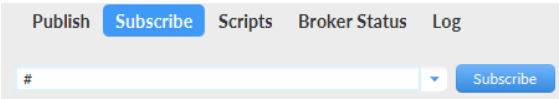


Diese Nachricht sollte nun in den Objekten von ioBroker aufscheinen:

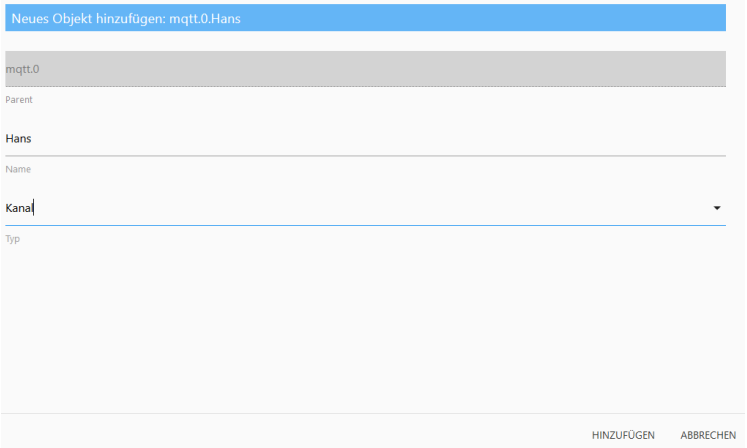


Überschreibt man in den Objekten nun diesen Wert, so wird dieser wieder an alle Subscriber gesendet:

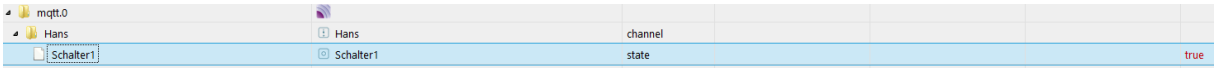
Um alle Nachricheten zu empfangen, geben wir ein „#“ in das Feld ein:



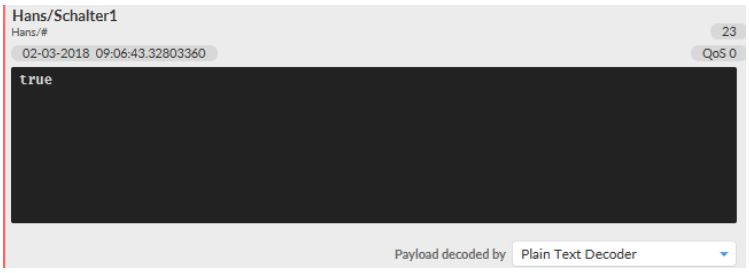
Um weitere Kanäle anzulegen, kann man diese in der Objektliste von ioBroker anlegen:



In diesem Objekt legen wir ein weiteres Objekt „Schalter1“ als Logikwert an:



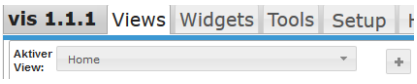
Wenn der Kanal in MQTT.fx abonniert (subscribed) wurde dann sollte nun der Wert in MQTT.fx angezeigt werden:



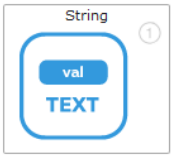
(Der Wert wird als Payload bezeichnet)

Verknüpfung mit der Visualisierung

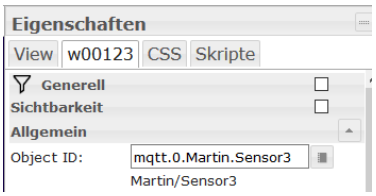
Als ersten Schritt erstellen wir einen neuen View:



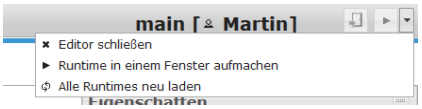
Um einen Text (String) anzuzeigen fügen wir das String Widget ein:



Unter „Object ID“ stellen wir eine Verknüpfung zu unserem Datenpunkt her:

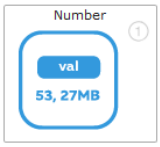


Um das Ergebnis auf der „Realen“ Webseite zu sehen klicken wir auf:

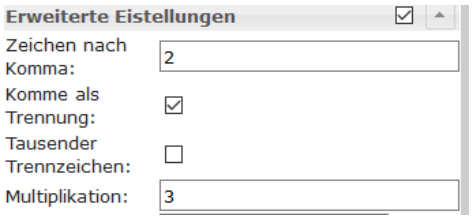


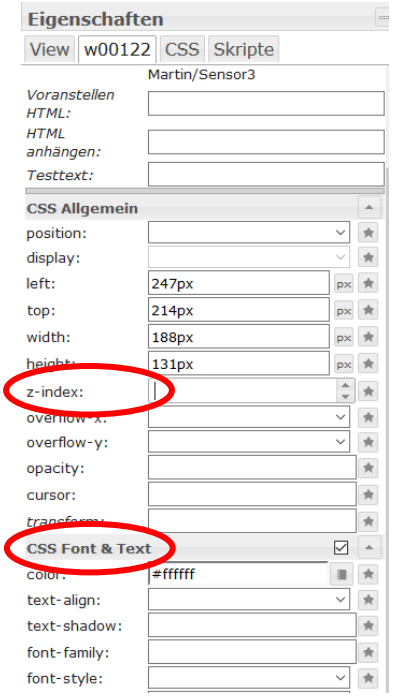
Runtime in einem Fenster aufmachen.

Zur Darstellung von Zahlenwerten eignet sich das Number-Widget besser:



… damit kann man unter „Erweiterte Einstellungen“ zusätzliche Formatierungen vornehmen:



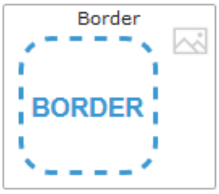


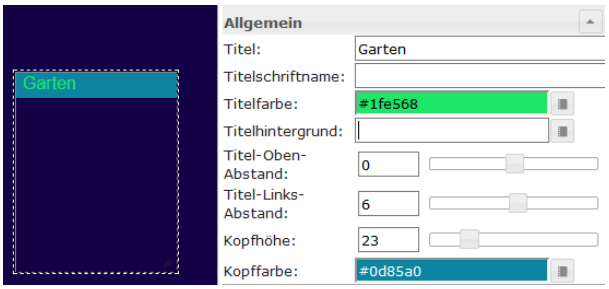
Weitere Formatierungs- und Layoutoptionen

Wenn man mehrere Widgets, die sich überlappen bzw. überdecken verwendet, kann über den Wert **z-index** die Anzeigeebene angeben. (Also welches Widget im Vorder- bzw. Hintergrund liegt).

Textfarbe, Ausrichtung usw. finden sich unter Font & Text.

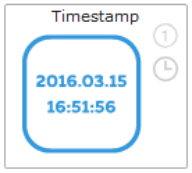
Rahmen können mit „Border“ erstellt werden:

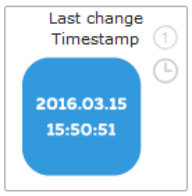




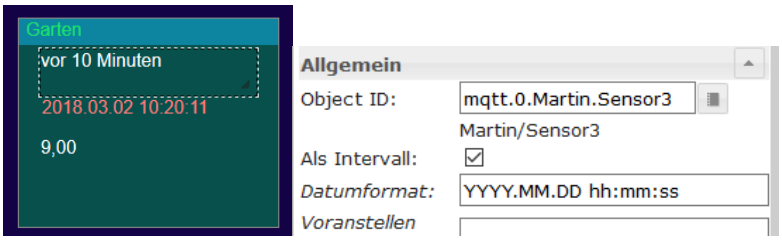
Aktualität der Daten anzeigen

Die beiden Widgets “Timestamp” und „Last change Timestamp” zeigen wann ein Datenpunkt geändert bzw. aktualisiert wurde:

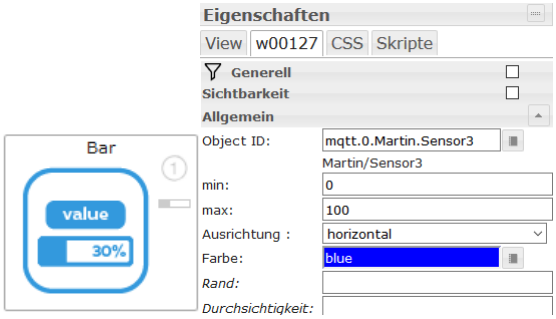




Es stehen verschieden Möglichkeiten der Anzeige zur Verfügung:

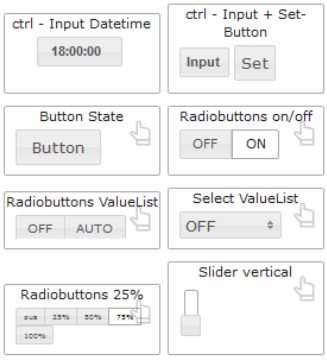


Darstellen von Messwerten mittels Balkendiagramm:



Steuernde Widgets

Sie sind in der Regel mit einem Handsymbol gekennzeichnet:



Aufgabe:

Auf einer ioBroker-Instanz wird der MQTT-Adapter als Server eingerichtet. Die anderen stellen diesen als Client ein und melden sich am Server an.

Jeder erstellt einen Channel mit seinem Namen und darin 3 States:

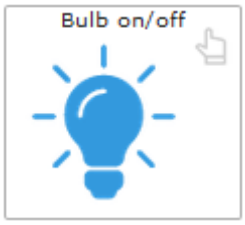
mqtt.0						
Hans	Hans	channel				
Schalter1	Schalter1	state				true
Temperatur	Temperatur	state				22.4
Text	Text	state				Testtext

! Richtige Datentypen verwenden!

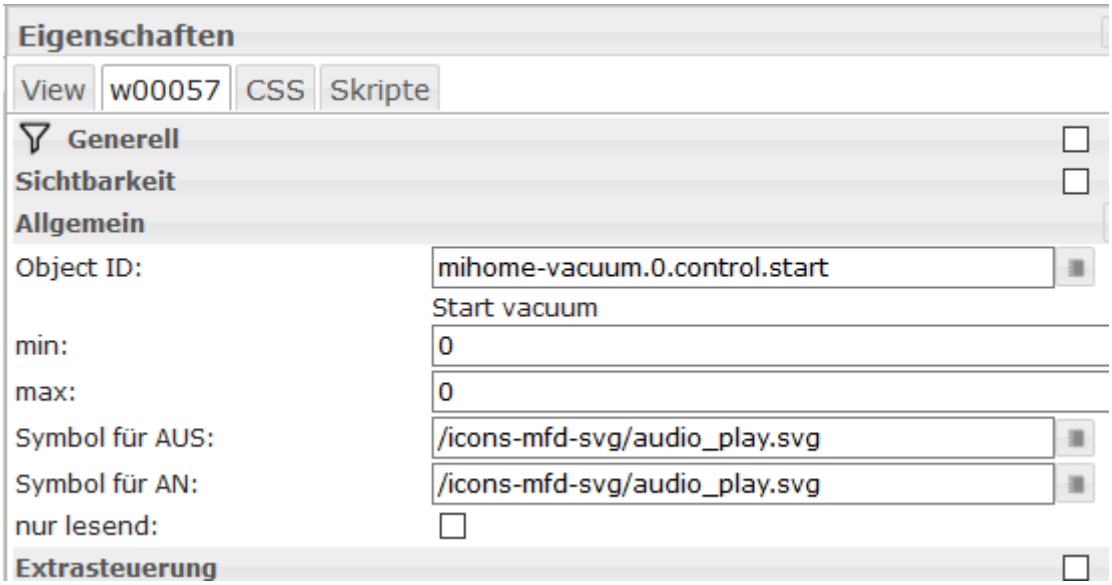
In der Visualisierung werden nun die eigenen States als steuerbar eingebaut. Danach sollen die States von anderen Mitschülern auch in die eigene Seite eingebaut werden.

Buttons mit Symbolen

Bulb on/off Widget einfügen



Soll der Datenpunkt nur aktualisiert werden (Taster) dann folgende Einstellung:

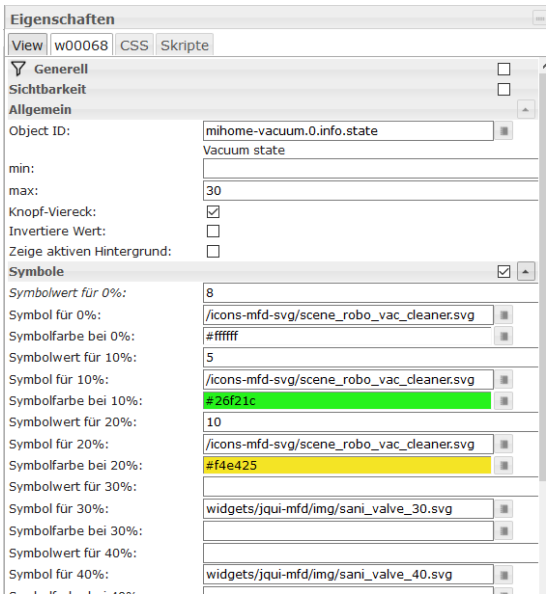


Zustände mit unterschiedlichen Farben darstellen:

Vis-jqui-mfd und icons-mfd-svg müssen installiert sein

Custom10 Widget einfügen:

Symbole und Farben zuweisen:



Button mit variablem Text einfärben

Vis-jqui-mfd muss installiert sein.

HTML Bool Widget einfügen: