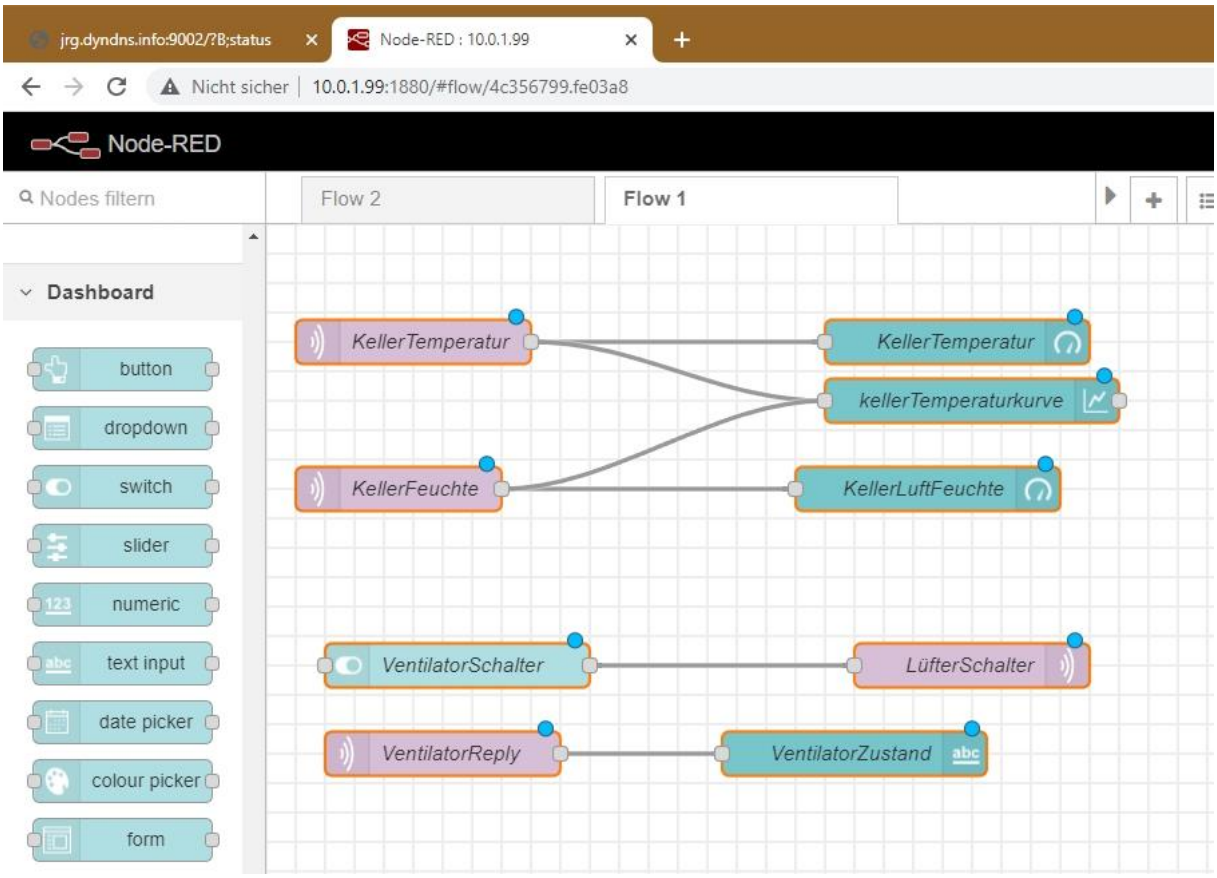

Abbildung 1: Das Node-RED-Dashboard - Schnittstelle zur Realwelt

In den Blogposts zur Reihe

Server und Clients unter MicroPython auf dem Raspi und der ESP-Familie

sind bereits wegweisende Dinge geschehen. Rückblickend wurden folgende Sachverhalte erklärt und drei MQTT-Clients gebaut. Hier eine kurze Zusammenstellung.

- In Folge 1 - MQTT - Raspberry Pi – Mosquitto und Node-RED haben wir Node-RED und den MQTT-Server Mosquitto auf dem Raspberry Pi installiert.
- In Folge 2 Ein ESP8266 als MQTT-Client und
- Folge 3 Erweiterung des MQTT-Home-Systems entstanden ein zweiter MQTT-Client mit dem ESP8266 sowie ein Monitor mit einem ESP32 zum Steuern und Überwachen des Systems.
- Um die übersichtliche Darstellung der erfassten Daten durch das Tool Node-RED ging es in Folge 4. Dort wurde auch genau der Umgang mit der Weboberfläche von Node-RED behandelt.

Heute wollen wir uns mit der Absicherung des Datenverkehrs beschäftigen, sofern uns das die beschränkten Möglichkeiten des ESP8266 erlauben. Damit willkommen zur aktuellen Episode:

5. Sicherheit beim Datentransfer zwischen ESP8266 und Mosquitto

Die Hard- und Software, die auch in diesem Teil benötigt wird, wurde in den vorangegangenen Episoden im Einzelnen aufgeführt und besprochen. Benutzen Sie daher bitte die eingangs aufgeführten Links für nähere Informationen. Im vorliegenden Post kommen insbesondere zum Einsatz

Hardware:

Der DHT-Client aus der 2. Folge
Der DS18B20-Client aus Folge 3
Ein Raspberry Pi mit Mosquitto- und Node-RED-Installation aus Folge 1
ein PC mit Putty-Installation aus Folge 1

Software:

dhtclient.py aus Folge 2
heizung.py aus Folge 3
Putty.exe aus Folge 1
Mosquitto und Node-RED auf dem Raspi aus Folge 1
dhtclient.json als Produkt des Projekts aus Folge 4

Zur Hausaufgabe aus der 4. Folge

Nach der ausführlichen Darstellung des Node-RED-Dashboards gab es eine Hausaufgabe. Deren Lösung werden wir uns jetzt gleich zu Beginn anschauen.

Es beginnt mit dem Einlesen des Status quo durch die Datei dht-client.json. Sie enthält alles, was letztes Mal in Node-RED an Nodes und Flows erstellt wurde. Nach dem Download in ein beliebiges Verzeichnis öffnen wir die Datei in einem Editor unserer Wahl. und kopieren ihn in die Zwischenablage, Strg+A, Strg+C.

Wir öffnen einen Browser, zum Beispiel Chrome, starten unseren Raspi mit Mosquitto und Node-RED. Laufen auch die beiden Clients dhtclient.py und heizung.py? Im Browser geben wir als URL die Adresse des Raspi, gefolgt von der Portnummer 1880 für den Aufruf von Node-RED ein.

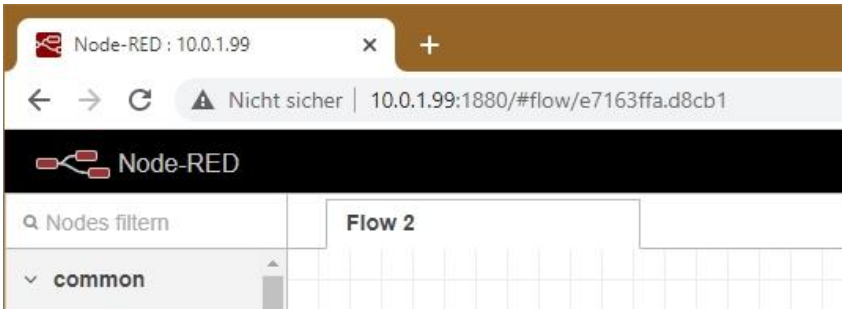

Abbildung 2: Node-RED starten

Über das **Menü** starten wir den **Import** des Inhalts der json-Datei aus der Zwischenablage.

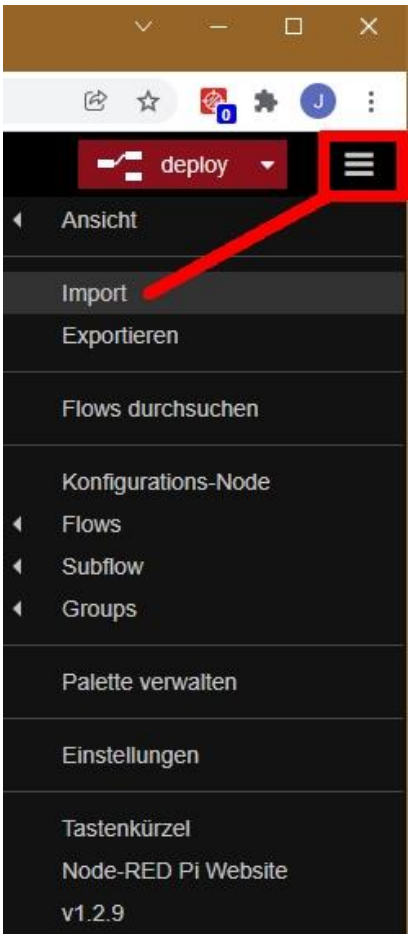

Abbildung 3: Flow importieren

Den json-Code kopieren wir in das rosa Fenster und klicken auf **Import**.

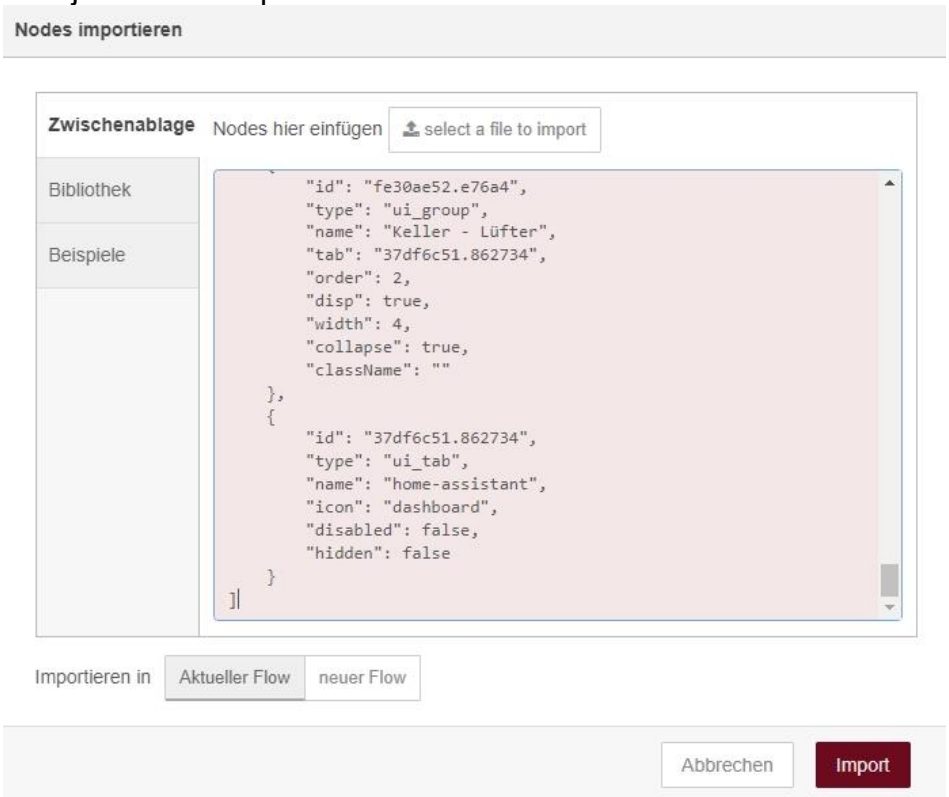

Abbildung 4: Der Inhalt von dhtclient.json wird importiert

Weil sich vom letzten Mal vermutlich noch Nodes im Speicher befinden, bringt Node-RED die folgende Meldung, die wir mit **Import copy** quittieren.

Abbildung 5: Meldung beim Import

In Flow 1 sehen wir jetzt die Nodes aus der vorangegangenen Folge, einmal auf der Arbeitsfläche und außerdem in der Seitenspalte unter dem Reiter Info.

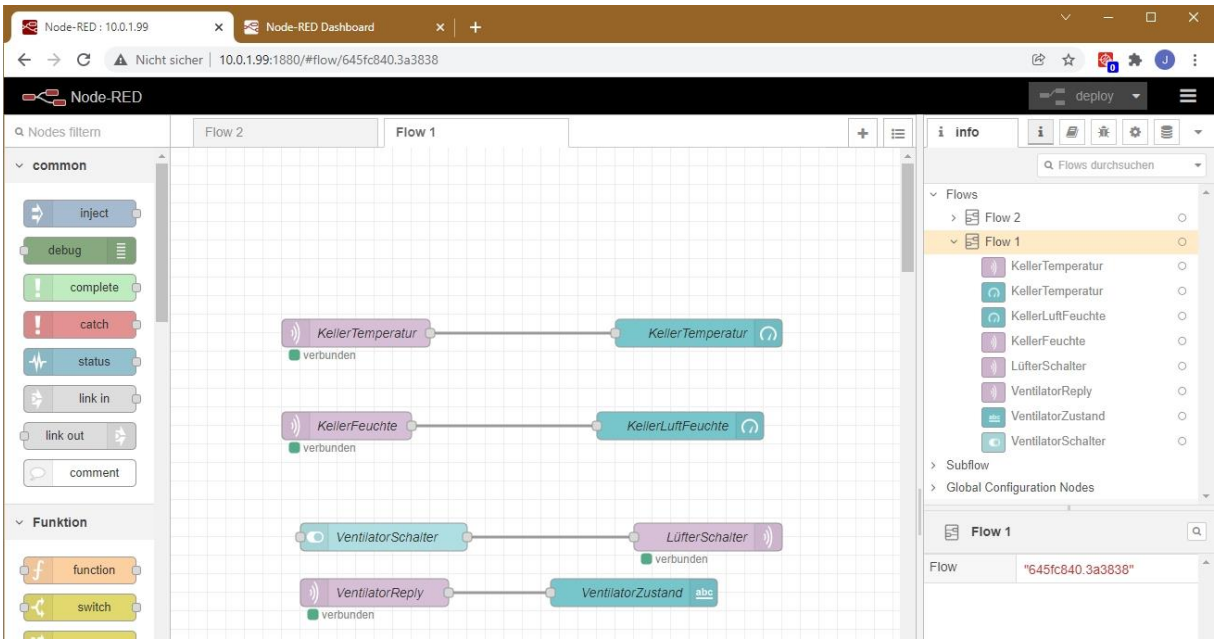

Abbildung 6: Der Flow aus der 4. Folge

Lassen Sie uns zunächst einen neuen Tab mit dem Namen **Heizung** erzeugen und konfigurieren. Über das kleine Dreieck rufen wir die Anzeige der Dashboard-Objekte auf und erzeugen durch Linksklick auf **+Tab** den neuen Tab, den wir sofort bearbeiten.

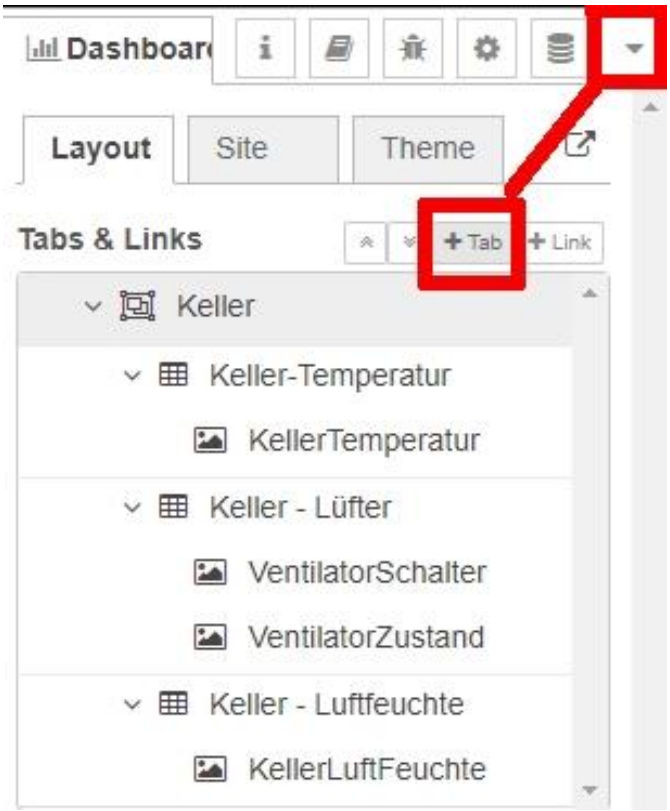

Abbildung 7: Dashboard-Objekte anzeigen und neuen Tab Erzeugen

Abbildung 8: Tab bearbeiten

Wir vergeben den Namen **Heizung** und **aktualisieren** den Zustand.

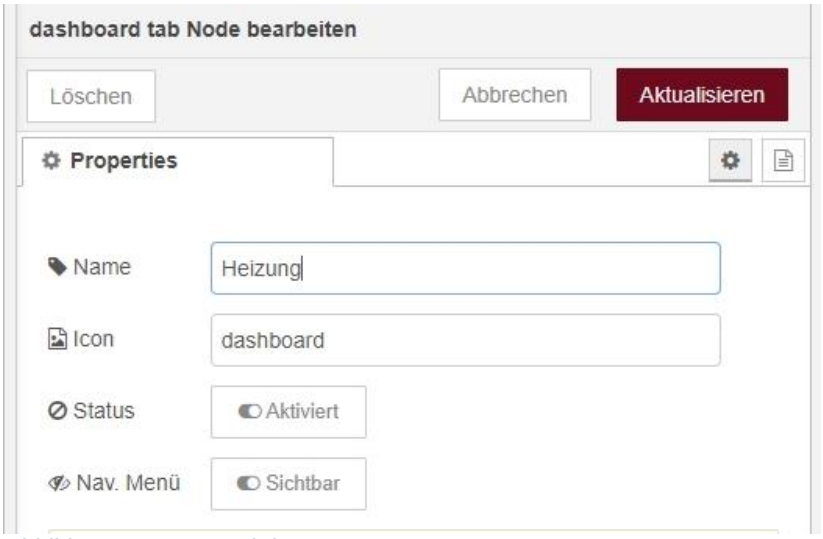

Abbildung 9: Neuen Tab benennen

Im neuen Tab erstellen wir zwei neue Gruppen, **Temperaturen** und **Schalter**, die wir auch gleich **bearbeiten**.

Abbildung 10: Neue Gruppen im Tab Heizung erzeugen

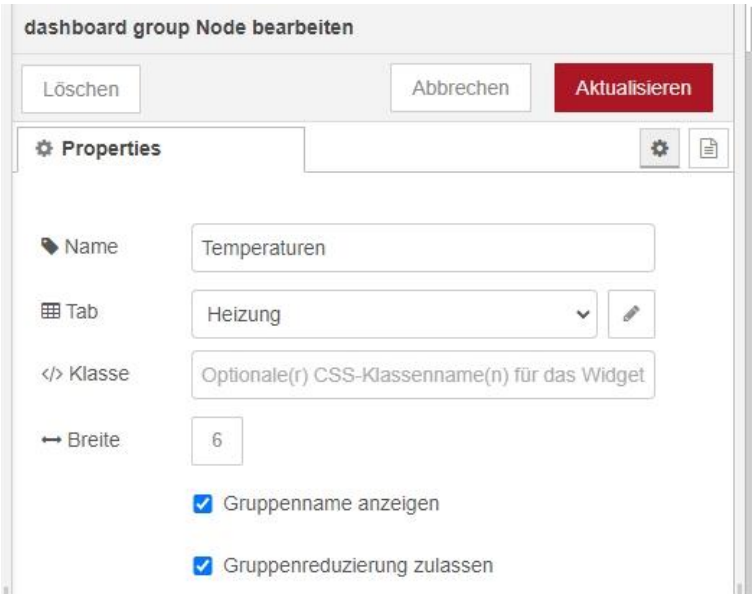

Abbildung 11: Gruppe Temperaturen bearbeiten

Analog zu den bereits bestehenden Nodes legen wir Nodes für die Erfassung von Vorlauf- und Rücklauftemperatur, deren Darstellung durch Messinstrumente sowie die Bedienung und Rückmeldung der Schalter fest. ein Doppelklick auf die Nodes öffnet, wie wir wissen, das Eigenschaftsfenster zur Bearbeitung.

Der mqtt-Node hat nur wenige Eigenschaften.

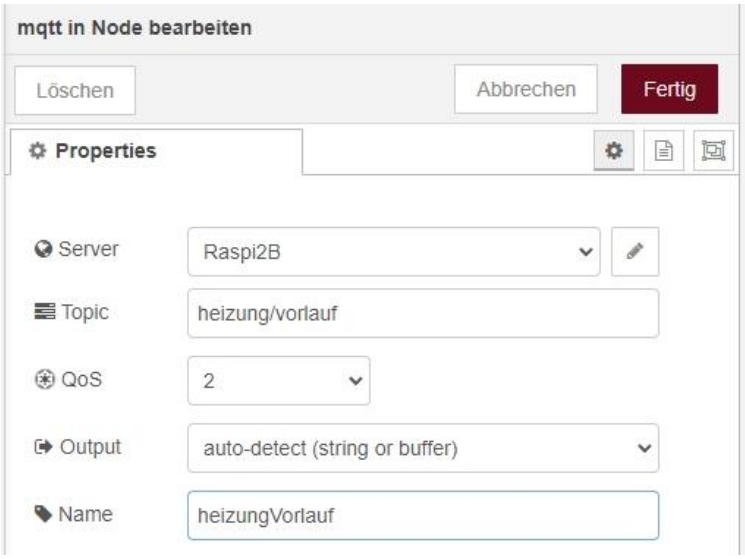

Abbildung 12: mqtt-Node Heizung-Vorlauftemperatur

So sieht jetzt der Bereich mit den neu angelegten Nodes aus.

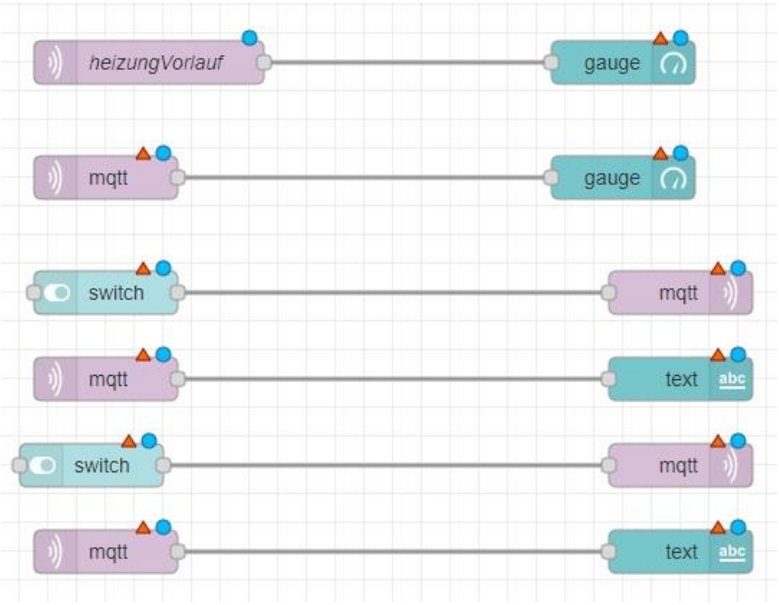

Abbildung 13: Die neuen Nodes

Bei der Anzeige der Vorlauftemperatur sind ein paar Einstellungen mehr nötig.

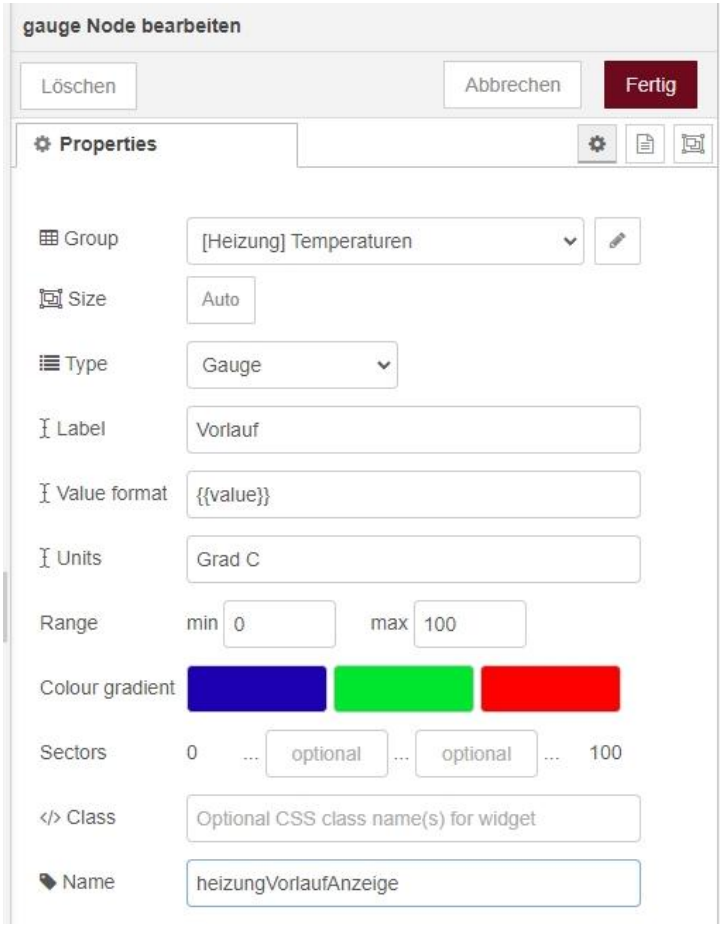

Abbildung 14: Anzeige Vorlauftemperatur

Für den Rücklauf gelten entsprechende Eigenschaftswerte. Es fehlen noch die Properties der beiden Schalter und ihrer mqtt-Nodes.

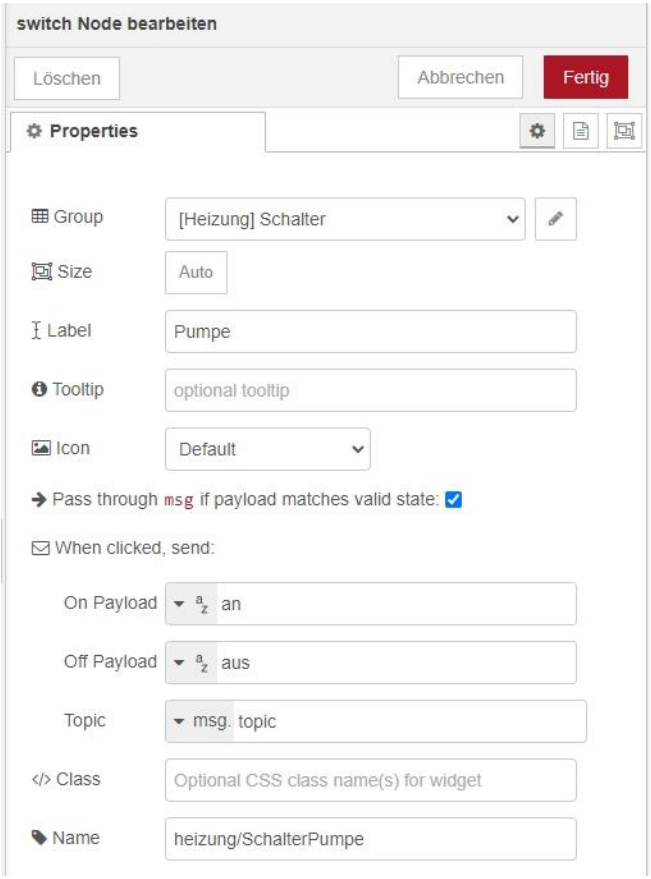

Abbildung 15: Eigenschaften des Pumpenschalters

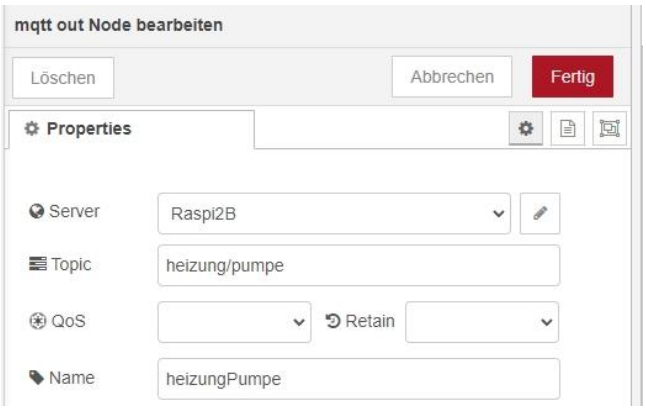

Abbildung 16: mqtt-Node der Pumpe

Analog werden die Daten des Schalters für den Brenner eingegeben. Der neue Flow ist damit fertig.

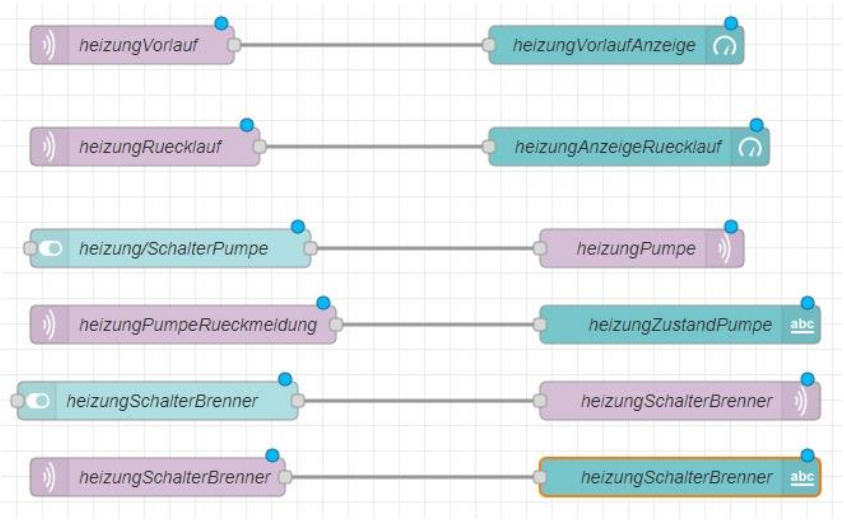

Abbildung 17: Der neue Flow ist fertig

Zum Veröffentlichen klicken wir auf **deploy** und schalten zum Dashboard weiter.

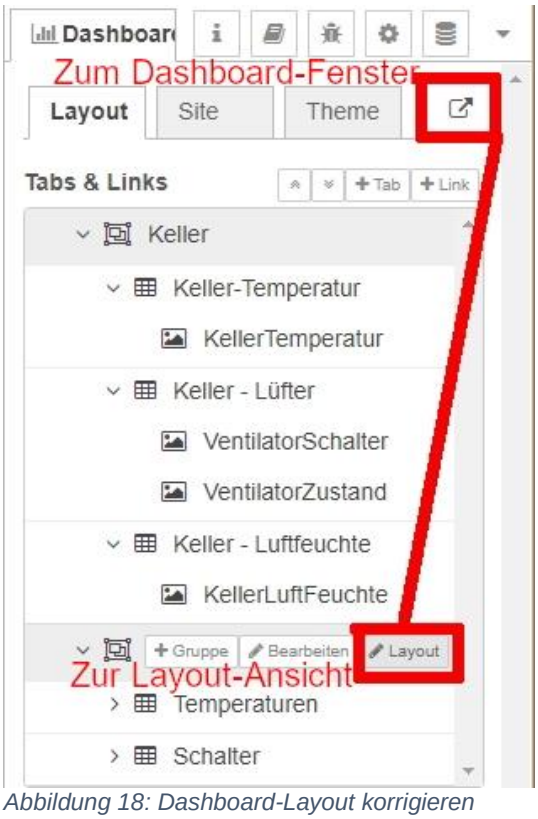

Abbildung 18: Dashboard-Layout korrigieren

Die Darstellung des Dashboards im Browser offenbart uns eine falsche Anordnung der Nodes in der Gruppe Schalter. Auch die Positionen von Vorlauf und Rücklauf stimmen nicht.

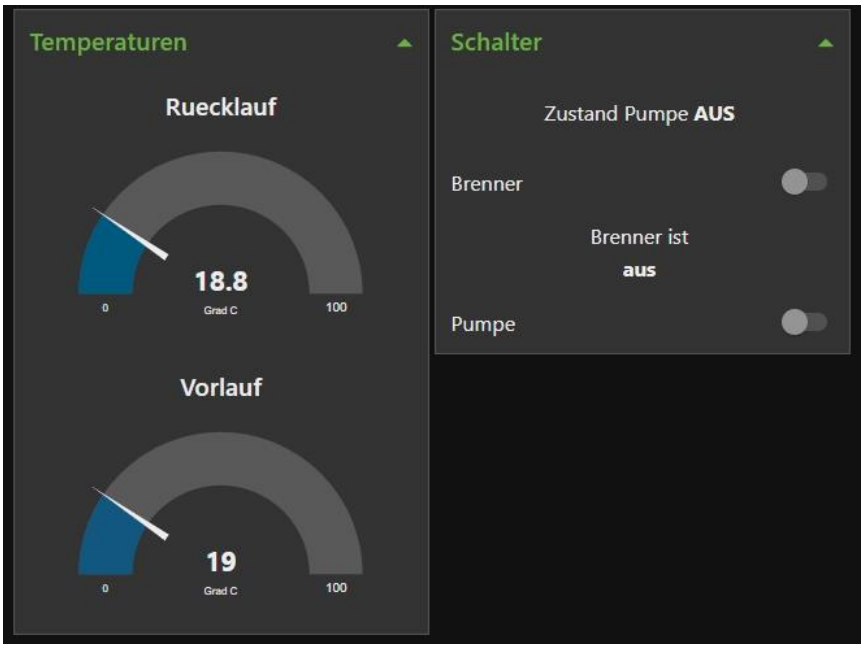

Abbildung 19: Die Schalteranordnung stimmt nicht

Wir schalten deshalb in die Layoutansicht und achten darauf, dass die Schlösser geöffnet sind, dann kann man die Nodes (hellblaue Felder) gegeneinander verschieben. Danach klicken wir die Schlösser zu, damit sich nichts mehr von selbst verändern kann.

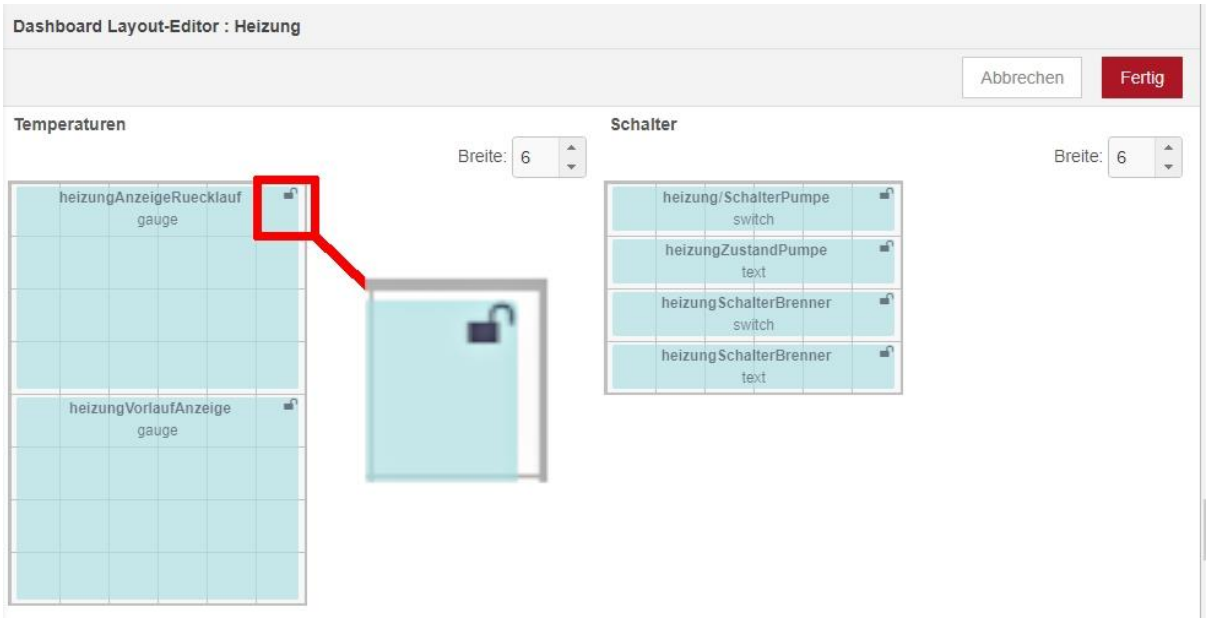

Abbildung 20: Anordnung durch Schieben verändern

Jetzt fehlt nur noch der Chart-Node für die Lagerkeller-Temperatur. Den holen wir aus dem Dashboard-Ordner in der Spalte links neben der Arbeitsfläche und verbinden seinen Eingang mit dem Ausgang des mqtt-Nodes **KellerTemperatur**.

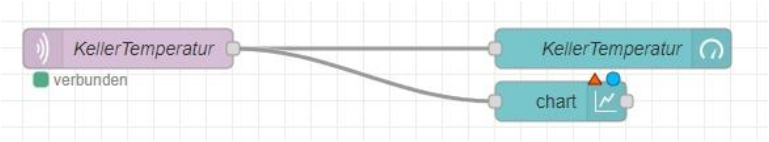

Abbildung 21: Chart einfügen

Doppelklick auf chart öffnet die Eigenschaften.

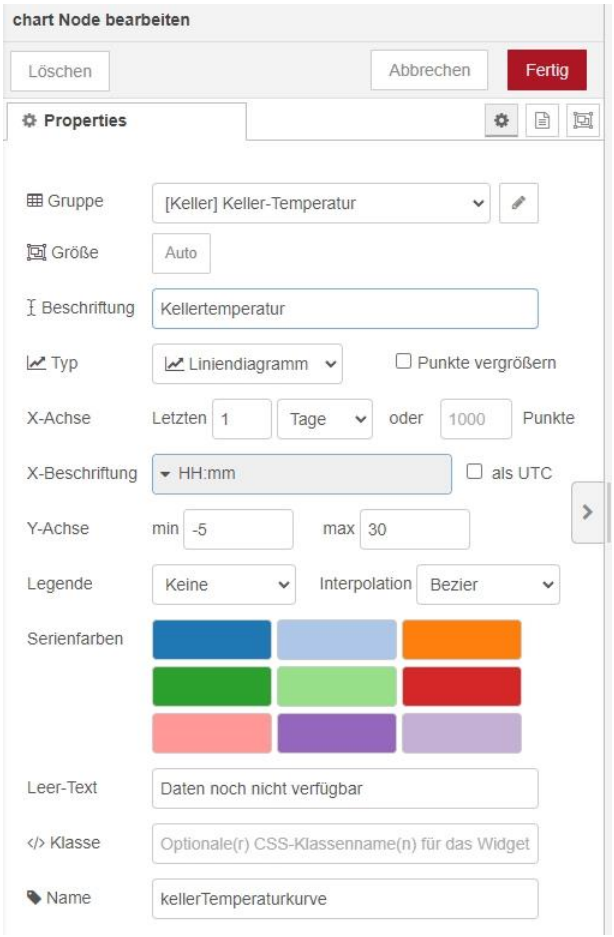

Abbildung 22: Chart- Node bearbeiten

Die drei waagrechten Linien verbergen das Menü zum Umschalten zwischen den Tabs **Keller** und **Heizung**.

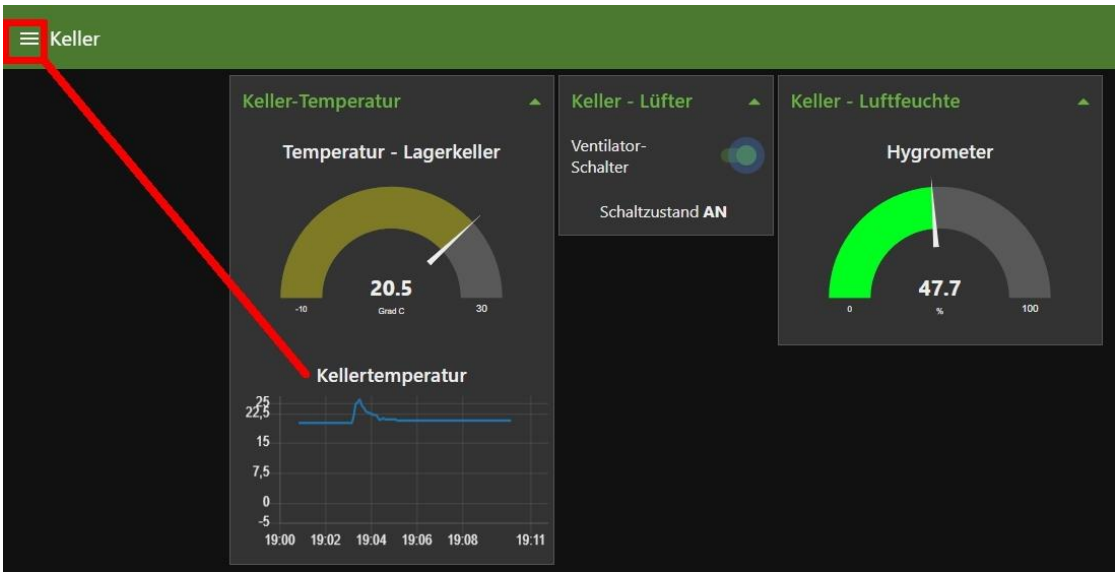

Abbildung 23: Umschalten zum Keller-Tab

Zum Sichern der Arbeit, erstellen wir eine neue json-Datei über den Menüpunkt **exportieren**.

Absicherung des Datentransfers

Wenden wir uns nun dem Thema Sicherheit zu. Bisher kann jeder Teilnehmer (oder Eindringling) in unserem Netz Nachrichten an den Broker senden oder von diesem abrufen. Wir hatten das benutzt, um von der Kommandozeile unseres Raspis aus, die Clients zu testen. Eines schicke ich voraus. Ein verschlüsselter Betrieb ist mit dem ESP8266 leider nicht möglich, weil dieser Controller keine Client-Zertifikate unterstützt und zu wenig Speicher aufweist. Das heißt, dass alle Daten, auch Passwörter, unverschlüsselt übertragen werden. Im lokalen Netz sollte das aber kein Problem darstellen. Außerdem besteht immer noch die Möglichkeit, die Payload der Nachrichten nicht im Klartext, sondern nach eigenen Verschlüsselungsmethoden codiert zu versenden und beim Empfänger entsprechend zu decodieren.

Auch ohne TLS werden wir nun die Zugriffe auf Mosquitto strenger regeln, indem wir Benutzer-Passwortpaare einführen und außerdem Zugriffsregeln in einer sogenannten Access Control List (ACL) aufstellen.

Ich habe zu diesen Zwecken zwei Terminals von Windows über putty als SSH-Verbindung geöffnet. Ebenso gut kann man direkt auf dem Raspi in Terminals arbeiten.

Kümmern wir uns als Erstes um die Benutzer, die sich mit einem Passwort an Mosquitto anmelden sollen. Wir könnten jeden Benutzer und damit meine ich jeden unserer Clients, **dhtclient**, **heizung** und **monitor** sowie den Chef-Benutzer, genannt **master,** einzeln von Hand anlegen. Es hat aber Vorteile, gleich eine Datei mit diesen Daten zu erstellen, wenn es sich um mehrere User handelt.

Also wechseln wir in das Verzeichnis, in dem diese Datei **mosquitto_users.txt** entstehen soll.

```
cd /etc/mosquitto/conf.d
```

```
sudo nano mosquitto_users.txt
```

Die folgenden Zeilen werden als Text erfasst. Sie können natürlich die User anders benennen und ihnen beliebige Passwörter zuordnen, aber **die letzte Zeile darf keinen Zeilenvorschub erhalten!**

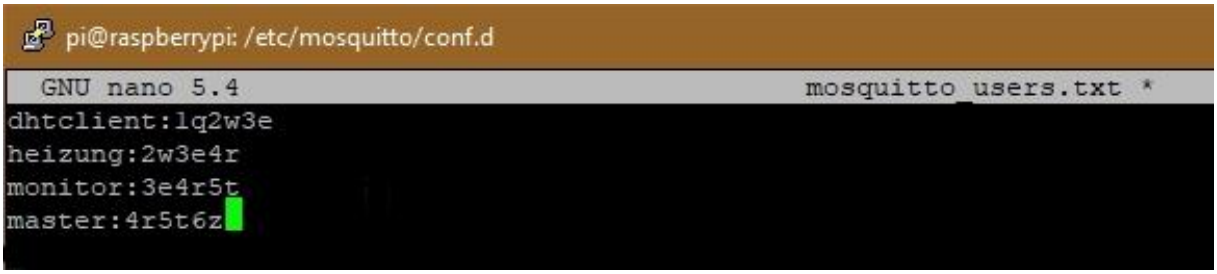

Abbildung 24: User-ZugriffsDatei

Dann speichern wir die Datei ab (Strg+O) und beenden den Editor (Strg+X). Ich empfehle dringend, von dieser Datei eine Sicherungskopie zu erstellen. Den Grund dafür verrate ich Ihnen gleich danach.

```
cp mosquitto_users.txt mosquitto_users.txt.sic
```

Mosquitto erwartet die Passwörter in verschlüsselter Form. Die Umwandlung von der lesbaren Form in einen Hash erledigt die folgende Zeile.

```
sudo mosquitto_passwd -U /etc/mosquitto/conf.d/mosquitto_users.txt
```

Danach sieht unsere Datei so aus.

```
cat mosquitto_users.txt
```

Ausgabe:

```
dhtclient:$7$101$eTNtaOGEedH1o1Ef$ZB3LOj2VII6Sz9S1KPvz4NCr0qvqOgxl1fH8S0TPlEAv6u9fayMLPz4l6ZwT8xdcVGobFijCaP7sCx+piNNmvQ==
heizung:$7$101$M5P1ddy9blnoMlAO$EoV2rGJgEiAHnBd02xRHL8Z+vAv9WpI8iChX2wRx5ayBk0W2nHpeR7bP4MtRqPg68YSH4UURR/xUwkS75XBvZQ==
monitor:$7$101$bXLLtf++LwBwQgnV$GFT70NirQesFlZ2RnBLQfotCpORRIQJuqH4gC6lXRRpNJNuE/bPoFbkDuM0r0P7imOr+l3Tq+Ezws5zarhMBHw==
master:$7$101$Jz4B8+DWEVa2SDmW$DH1TkfGr4al3nPtG5k6R5ATcLA2xTyLBF5b3fjY0dMI4bktsXDRGEsGJ8KY3/8c9s/27ioDrUtIONGQiKENyBA==
```

Für den Fall, dass Sie ein Passwort ändern möchten, können Sie jetzt immer noch auf die Originaldatei in der Sicherung zurückgreifen, um mit der Sicherung die Originaldatei zu überschreiben, die Änderungen durchzuführen, erneut zu sichern und dann auch die Umschreibung erneut auszulösen. Das Hinzufügen von Usern kann ähnlich gehandhabt werden. Man kann einzelne Benutzer aber auch von Hand hinzufügen.

```
sudo mosquitto_passwd /etc/mosquitto/conf.d/mosquitto_users.txt doorbell
```

Dieser Befehl fügt den User **doorbell** hinzu und fragt nach einem Passwort für ihn. Mit der Option -D entfernen Sie einen User.

```
sudo mosquitto_passwd -D /etc/mosquitto/conf.d/mosquitto_users.txt doorbell
```

Bitte beachten:
Es ist nicht ratsam, einen Passwort-Hash in der Datei durch ein neues Klartext-Passwort zu ersetzen und danach die Datei neu zu transscribieren. Der Befehl *mosquitto_passwd -U* unterscheidet nicht zwischen Klartext-Passwörtern und bereits verschlüsselten. Hashes werden bei der Prozedur also noch einmal verschlüsselt. Das hat zur Folge, dass die Datei unbrauchbar wird und auf den Broker kein Zugriff mehr möglich ist.

Als Nächstes unterrichten wir Mosquitto via seiner conf-Datei darüber, ab sofort nur noch bekannte Benutzer an sich heranzulassen. Die bestehende Konfigurationsdatei kopieren wir auf die neue, taufen die bestehende um und editieren die neue.

```
sudo cp -a 010-listener-anonymous.conf 020-restricted.conf

sudo mv 010-listener-anonymous.conf 010-listener-anonymous.conf.old

sudo nano 020-restricted.conf
```

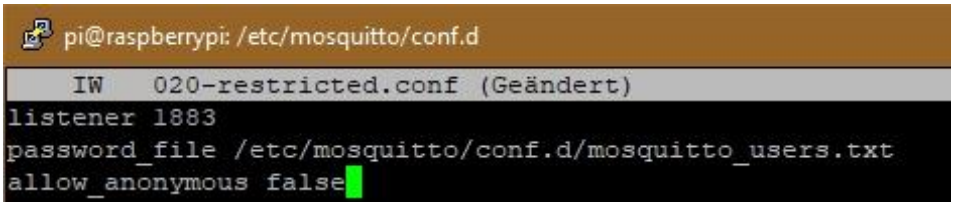

Abbildung 25: Neue Mosquitto-Config-Datei

Das config-Verzeichnis enthält jetzt folgende Dateien.

```
ls -lisa
```

Ausgabe:

```
insgesamt 28
258560 4 drwxr-xr-x 2 root root 4096 30. Dez 22:18 .
258554 4 drwxr-xr-x 5 root root 4096  5. Dez 06:07 ..
265774 4 -rw-r--r-- 1 root root   35  4. Dez 19:14 010-listener-
anonymous.conf.old
258110 4 -rw-r--r-- 1 root root   91 30. Dez 17:46 020-restricted.conf
258108 4 -rw-r--r-- 1 root root  607 30. Dez 20:11 mosquitto_users.txt
258561 4 -rw-r--r-- 1 root root  142  9. Jun 2021  README
```

Mosquitto erlaubt übrigens die Hintereinander-Ausführung mehrerer conf-Dateien im Verzeichnis /etc/mosquitto/conf.d. Die Nummern als Prefix steuern dann die Reihenfolge der Abarbeitung beim Start, ähnlich wie es mit den Links in den /etc/rcX.d-Verzeichnissen beim Booten des Systems geschieht.

Damit unsere Clients nun Kontakt mit dem Broker aufnehmen können, müssen sie sich bei der Kontaktaufnahme authentifizieren. Die entsprechenden Zeilen werden dementsprechend ergänzt.

monitor.py:

```
    client = MQTTClient(myID, myMQTTserver, user="monitor",\
                        password="3e4r5t")
```

heizung.py:

```
    client = MQTTClient(myID, myMQTTserver, user="heizung",\
                        password="2w3e4r")
```

dhtclient.py:

```
    client = MQTTClient(myID, myMQTTserver, user="dhtclient",\
                        password="1q2w3e")
```

Auch beim händischen Testen des Systems über ein Terminal muss jetzt eine Authentifizierung erfolgen.

```
mosquitto_pub -t "heizung/pumpe" -m "an" -u "master" -P "4r5t6z"
```

Jeder, der ein Auto fahren kann, kann auch in einen Bus einsteigen, aber es ist noch lange nicht gesagt, dass er diesen auch fahren darf. Gehen wir also noch einen Schritt weiter. Damit nicht jeder, der sich am System anmeldet, auch alles machen kann, werden wir jetzt Zugriffsbeschränkungen einführen, die so ähnlich wie Führerscheinklassen wirken.

Erzeugen wir also die Datei acl_liste.txt, die wir mit folgendem Inhalt füllen.

```
sudo nano acl_liste.txt
```

```
# Zugriffsliste
# alle duerfen gar nix
topic deny #

# Rechte der einzelnen Clients
user dhtclient
topic read keller/ventilator
topic write keller/temperature
topic write keller/humidity
topic write keller/ventilator/done

user heizung
topic read heizung/pumpe
topic read heizung/maschine
topic write heizung/vorlauf
topic write heizung/ruecklauf
topic write heizung/pumpe/done
topic write heizung/maschine/done

user monitor
topic read heizung/#
```

```
topic read keller/#
topic write heizung/pumpe
topic write heizung/maschine
topic write keller/ventilator

user master
topic readwrite #
```

Was bedeutet das alles?

Nun, dass alle erst einmal nichts dürfen, steht schon in dem Kommentar. Das "#" steht für alle untergeordneten Ebenen. Weil keine Ebene namentlich genannt ist, gilt das "#" für alles ab der obersten Ebene und **deny** heißt verweigern.

Der **dhtclient** darf die gemessenen Werte, Temperatur und Feuchte publishen, ferner darf er die Schalterstellung bekanntgeben. Er darf aber auch die Anforderung für die Schalterstellung entgegennehmen.

heizung darf die Schalteraktionen entgegennehmen sowie die Schalterstellung und die Temperaturwerte von Vor- und Rücklauf veröffentlichen.

monitor liest alle Veröffentlichungen von **dhtclient** und **heizung** und kann die Umwälzpumpe, den Brenner und den Ventilator im Lagerkeller schalten.

Natürlich braucht der Chef, genannt **master**, für die Überwachung und Steuerung sowie händische Tests alle nötigen Rechte.

Diese ACL machen wir ebenfalls dem Mosquitto-Broker bekannt.

sudo nano /etc/mosquitto/conf.d/020-restricted.conf

```
listener 1883
password_file /etc/mosquitto/conf.d/mosquitto_users.txt
allow_anonymous false
acl_file /etc/mosquitto/conf.d/acl_liste.txt
```

Die geänderte Software für die Clients

dhtclient_protected.py
heizung_protected.py
monitor_protected.py

wird in deren **boot.py** verfrachtet und hochgeladen – Neustart mit Reset.

Auch Mosquitto wird neu gestartet.

sudo service mosquitto restart

Und damit auch Node-RED wieder mitreden kann, braucht die Dashboard-App ebenfalls Generalrechte. Unseren Flow müssen wir also als **master** einloggen

lassen, weil eine Authentifizierung nach Diensten und Passwörtern nicht vorgesehen ist.

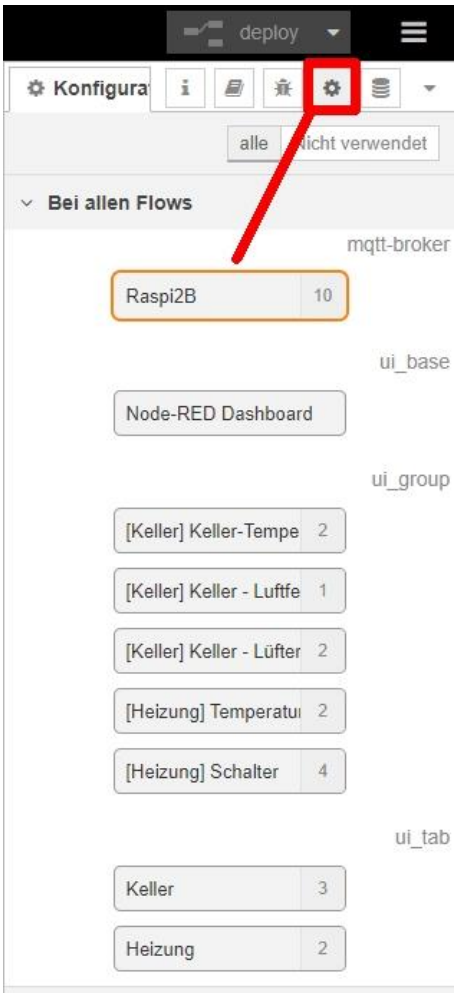

Abbildung 26: Node-RED-Applet mit Autentifizierungsdaten versehen

Doppelklick auf den Broker-Node öffnet dessen Property-Box. In der Karte Sicherheit geben wir die Credentials für **master** ein - **Aktualisieren**

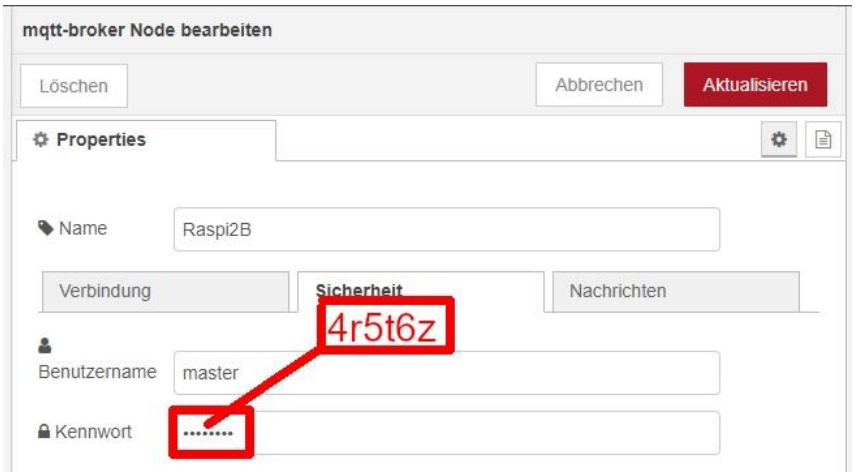

Abbildung 27: Credentials für den User master

Flow bereitstellen – **deploy**.

Bereit zum Test? Dann los! Wechseln wir ins Dashboard-Fenster. In beiden Tabs müssen jetzt die Werte alle 5 Sekunden nachgestellt werden. Die Relais an den Clients müssen von der Weboberfläche aus angesteuert werden können und die Rückmeldungen über die Schalterstellungen müssen ausgegeben werden. Wenn alles wunschgemäß funktioniert, folgt jetzt genussvolles Zurücklehnen.

Diesen Beitrag gibt es auch als PDF-Dokument zum Download.

Ausblick

In der nächsten Folge geht es, immer noch in der Reihe "Server und Clients unter MicroPython auf dem Raspi und der ESP-Familie" um eine Haustürklingel der etwas anderen Art. Natürlich wird dieses Projekt auch in die MQTT-Familie integriert. So resultiert daraus eine Anwendung, die nicht nur Besucher ankündigt, sondern auch Tag und Uhrzeit mitloggt und … Neugierig geworden, dann bis zum nächsten Mal.