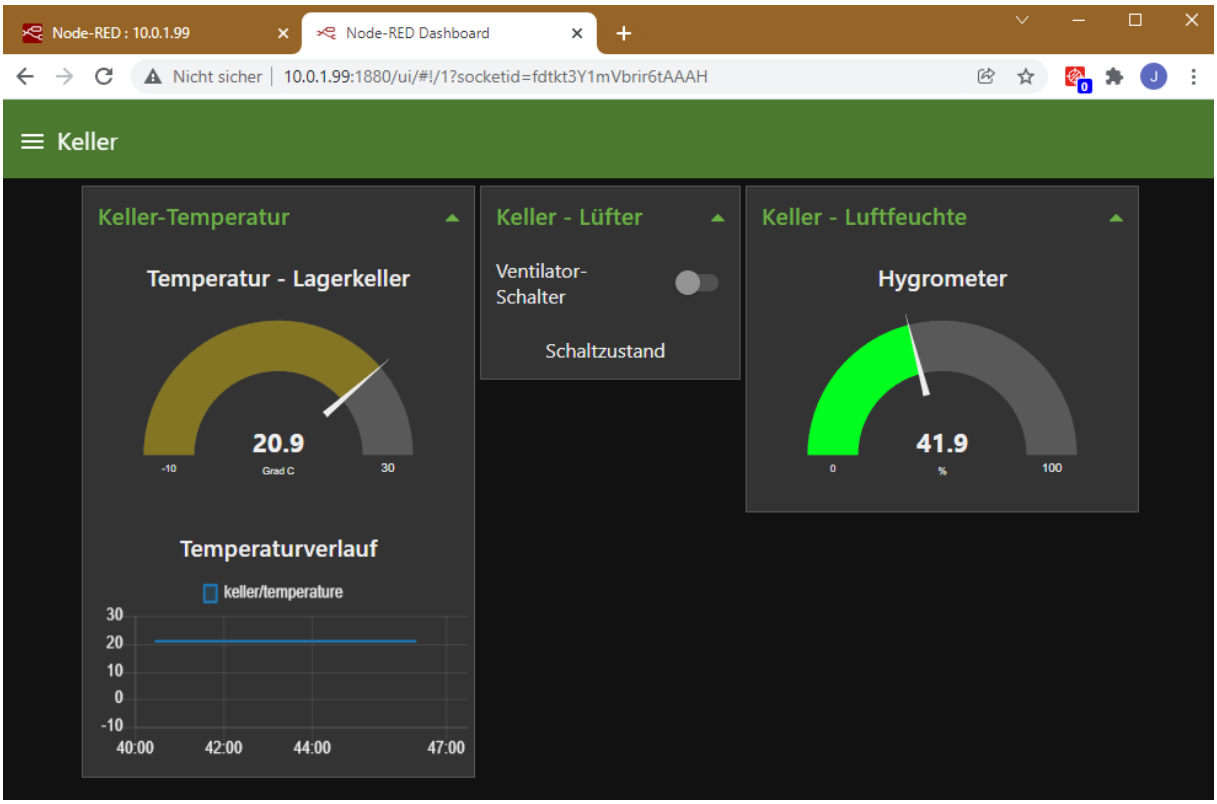


Abbildung 1: So soll es aussehen, wenn's fertig ist

Dieser Blogpost ist auch als PDF erhältlich.

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 zwei MQTT-Clients mit dem ESP8266 sowie ein Monitor mit einem ESP32 zum Steuern und Überwachen des Systems. Heute werden wir eine grafische Web-Oberfläche mit einem sehr interessanten Tool erstellen, das den Umgang mit MQTT erheblich vereinfacht. Schließlich macht es nicht viel Sinn, das Rad zum x-ten Mal neu zu erfinden. Damit willkommen zum 4. Teil der Reihe

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

4. Node-RED und MQTT

Benötigte Hard- und Software für diesen Teil 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 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
dht-client.json als Produkt des aktuellen Projekts

Bevor wir richtig loslegen können, werfen wir einen Blick auf die Oberfläche von Node-RED. Schalten wir also den Raspberry Pi ein, Node-RED sollte nach den Einstellungen in Blogpost 1 automatisch mit hochfahren. Ist das geschehen, dann starten wir auf der Windowskiste einen Browser. Ich habe mit Chrome gute Erfahrungen gemacht. Natürlich muss auch wenigstens einer unserer ESP8266-Clients gestartet werden, um den Brocker zu füttern. Bei mir laufen beide, und ich werde gleich als erstes den DHT-Client in Node-RED einbinden.

Das Fenster von Node-RED gliedert sich in drei Bereiche, die Nodeauswahl (Palette), den Arbeitsbereich, genannt Flow und den Bereich zum Anzeigen der Eigenschaften (Properties) in der rechten Seitenspalte. Nodes sind Icons, hinter denen man sich eine Funktion vorstellen kann, welche eine ganz bestimmte Aufgabe erfüllt. Manche Nodes haben Eingänge, sie entsprechen bei Funktionen den Eingabeparametern. Andere Nodes haben nur einen Ausgang, vergleichbar mit dem Rückgabewert einer Funktion. Viele Nodes haben Ein- und Ausgänge und alle haben mehr oder weniger Eigenschaften, die man editieren kann.

In der Seitenspalte findet man fünf Reiter, mit denen verschiedene Dinge angezeigt werden. Informationen über die Nodes im Flow, Hilfen zu den Nodes, das Debugfenster, Konfigurations-Nodes und Kontext-Daten.

Darüber verbirgt sich unter dem Symbol mit den drei Querbalken das Menü. Links daneben liegt der Button zum Veröffentlichen des Flows, quasi der Startknopf für das Programm.

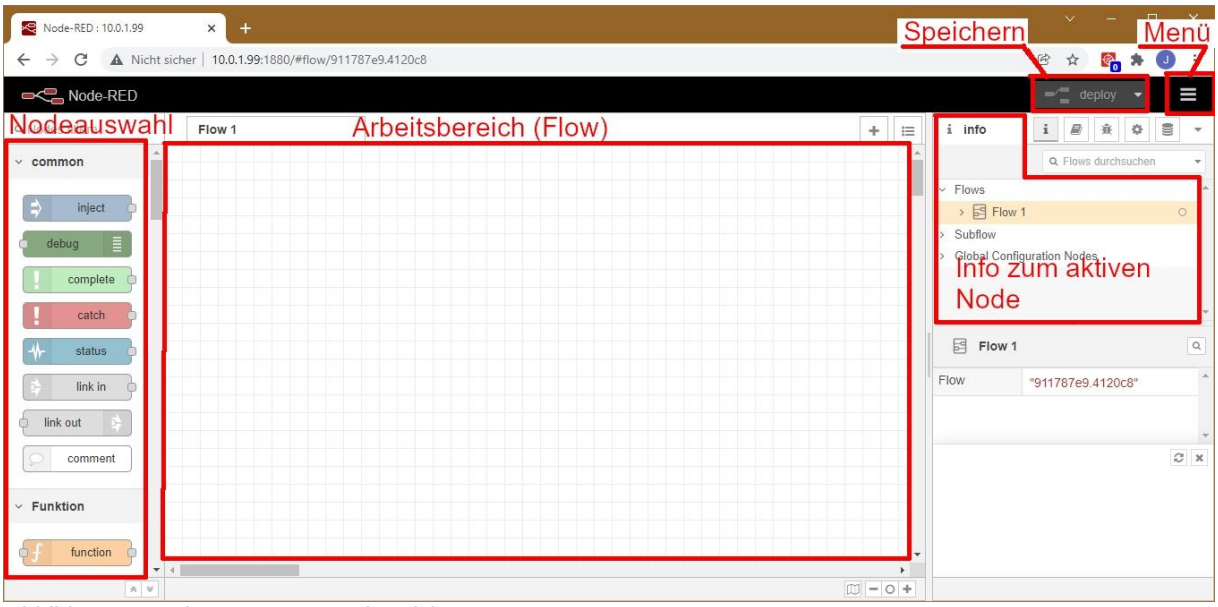


Abbildung 2: Node-RED - Fensterbereiche

Einen Flow erzeugt man dadurch, dass man Nodes (Knoten) aus der Palette in den Arbeitsbereich zieht und durch Leitungen verbindet. Probieren wir's gleich einmal aus. Wir klicken in der Property-Spalte auf das "**i**" für Info, dann ziehen wir einen **mqtt-in**-Node aus dem Reiter **network** in den Arbeitsbereich.

Abbildung 3: network-mqtt-in

Dann holen wir aus common einen Debug-Node dazu und ziehen rechts vom Ausgang des mqtt-Icons eine Verbindung zum Eingang des debug-Icons.

Abbildung 4: mqtt mit Debug-Node

Das Eigenschaftsfenster sieht jetzt so aus.

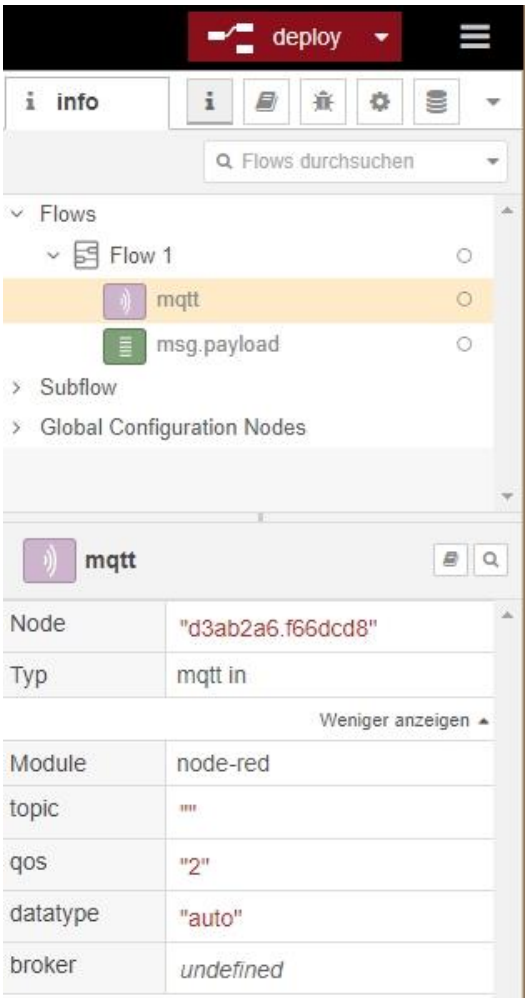


Abbildung 5: mqtt Properties

Dem mqtt-Node müssen wir natürlich sagen, von welchem Broker, welches Topic abonniert werden soll. Wir müssen also die Eigenschaften des Nodes richtig einstellen. Zum Editieren klicken wir doppelt auf das **mqtt-in**-Icon.

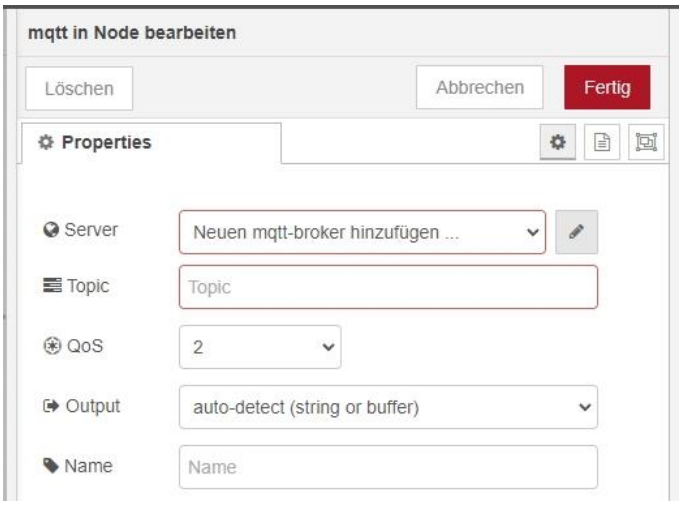


Abbildung 6: mqtt-Eigenschaften

Wir bearbeiten der Reihe nach die vier Felder **Server**, **Topic**, **Output** und **Name**. Zum Einstellen des Brokers klicken wir auf den Bleistift.

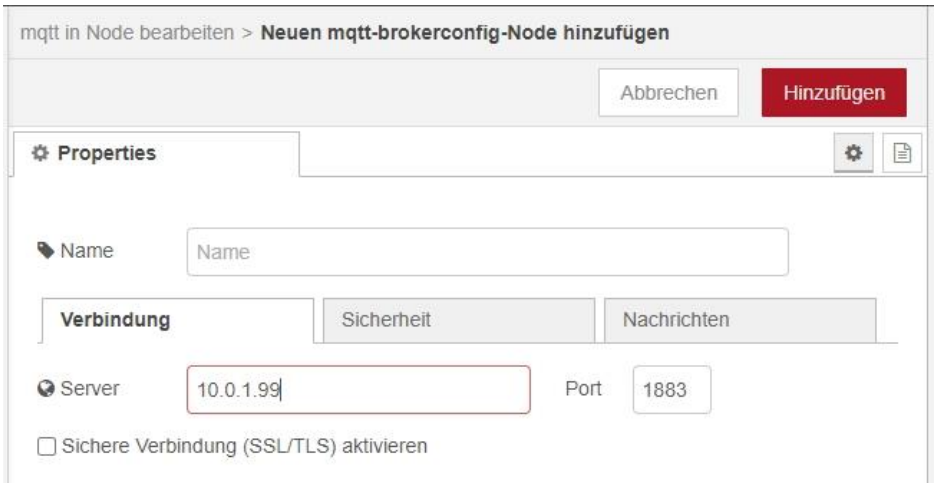


Abbildung 7: Hinzufügen des Brokers

In das Feld Server kommt die IP-Adresse unseres Raspi, der Port bleibt auf 1883. Als Name schreiben wir **Raspi**. Nach einem Linksklick auf Hinzufügen befinden wir uns wieder im Bearbeitungsfenster des mqtt-Nodes. Wir tragen das Topic ein, wählen aus der Liste bei Output **a string** und geben der Verbindung den Namen **Mosquitto**. – Fertig.

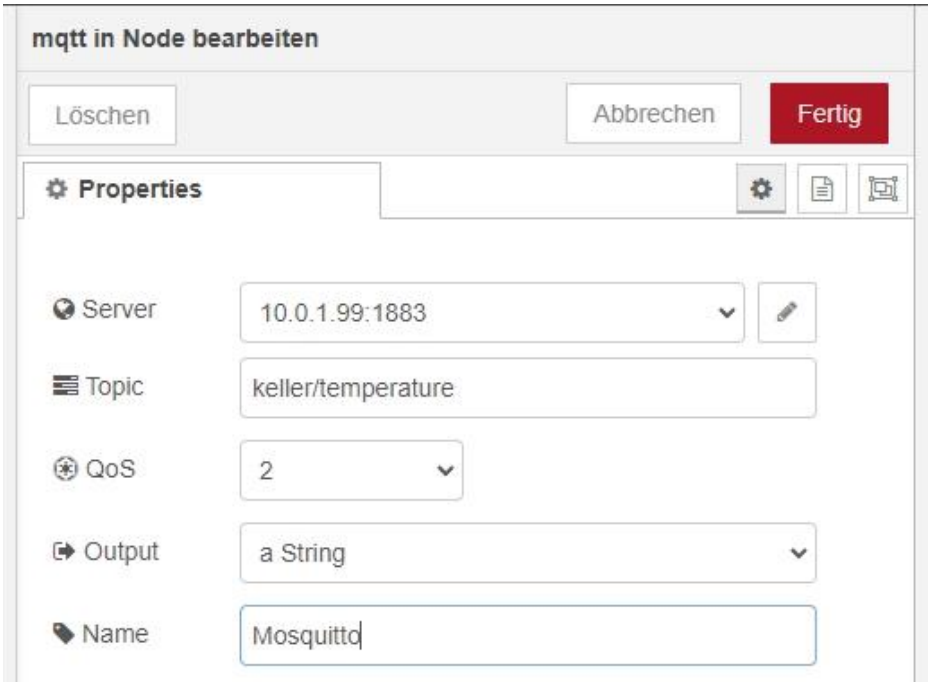


Abbildung 8: mqtt-Eigenschaften ausgefüllt

Ein Blick ins Eigenschaften-Fenster zeigt die letzten Einstellungen. Dann schalten wir durch Klick auf den Reiter mit der **Wanze** in der Seitenspalte auf das Debug-Fenster um und klicken auf den **deploy**-Button. Wenn dann **Erfolgreich implementiert** eingeblendet wird, haben wir alles richtig gemacht.

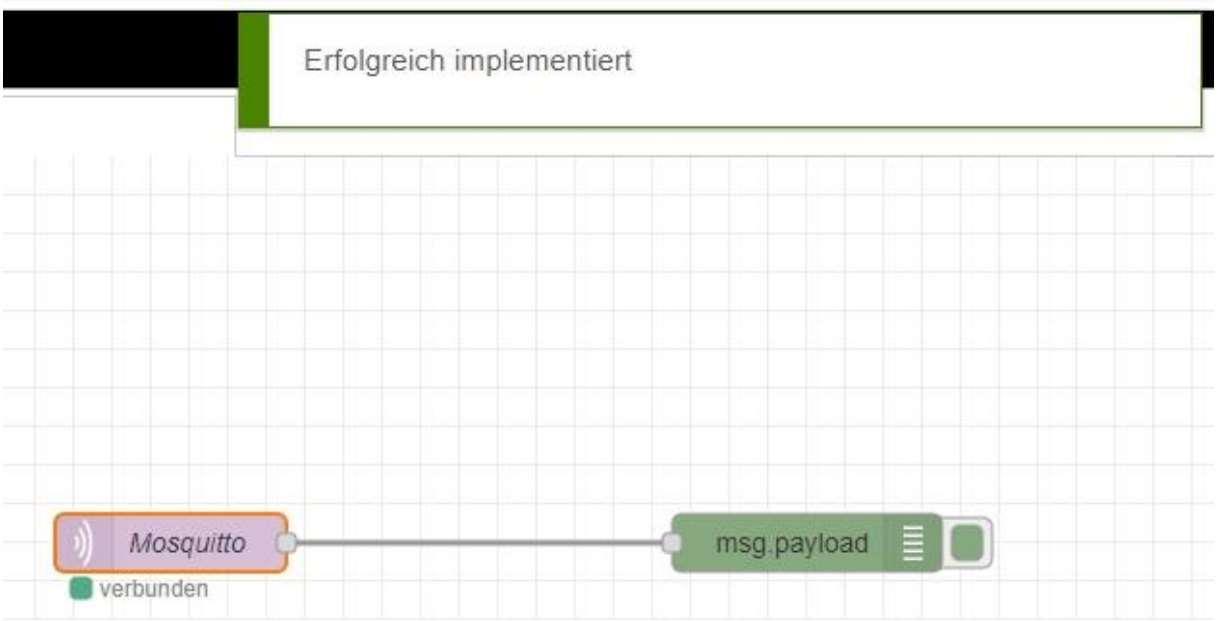


Abbildung 9: Erfolgreich implementiert

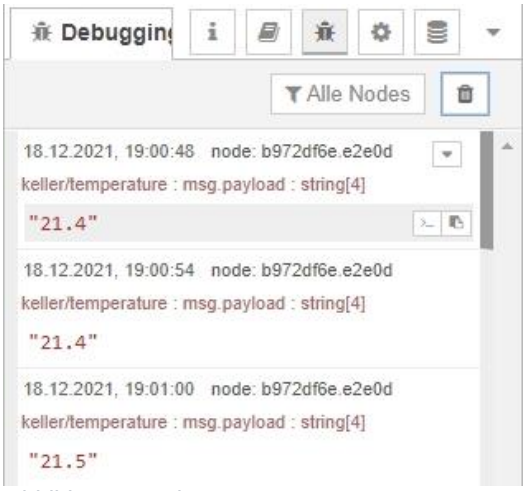


Abbildung 10: Die ersten Temperaturwerte vom dht-client

Sieh da, im Debugfenster erscheint alle 5 Sekunden ein neuer Temperaturwert vom DHT-Client. Sieht gut aus, das funktioniert schon mal!

Wir wollten aber doch eine schöne grafische Bedienoberfläche mit Anzeigegeräten und Schaltern. Gut, lassen Sie uns die gewünschten Nodes unserer Palette hinzufügen, denn im bisher bestehenden Fundus sind sie nicht enthalten.

Die Erweiterung führen wir über den Menüeintrag **Palette verwalten** durch.

Abbildung 11: Palette verwalten

Wir wählen den Reiter **Installieren**, geben im Suchfeld **node-red-dashboard** ein und starten, wenn das Paket angezeigt wird, rechts auf **installieren**. Es erscheint eine Meldung. Bevor sie erneut durch Klick auf **installieren** quittiert wird, ist es empfehlenswert, den Informationsbutton zu klicken. Er führt auf eine Webseite, wo die neuen Nodes erläutert werden. Nach ca. 15 bis 20 Sekunden ist die Einrichtung erledigt.

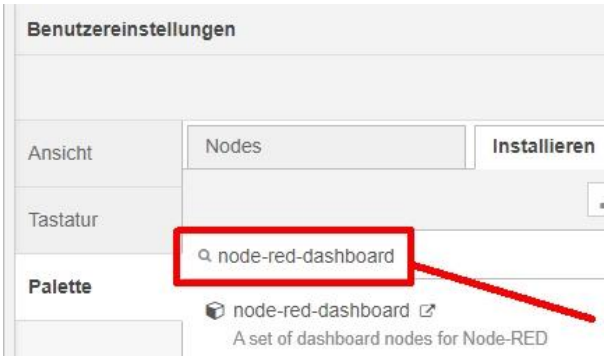


Abbildung 12: Node-RED - Dashboard installieren

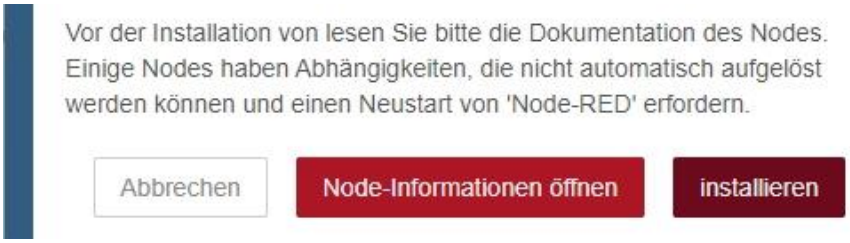
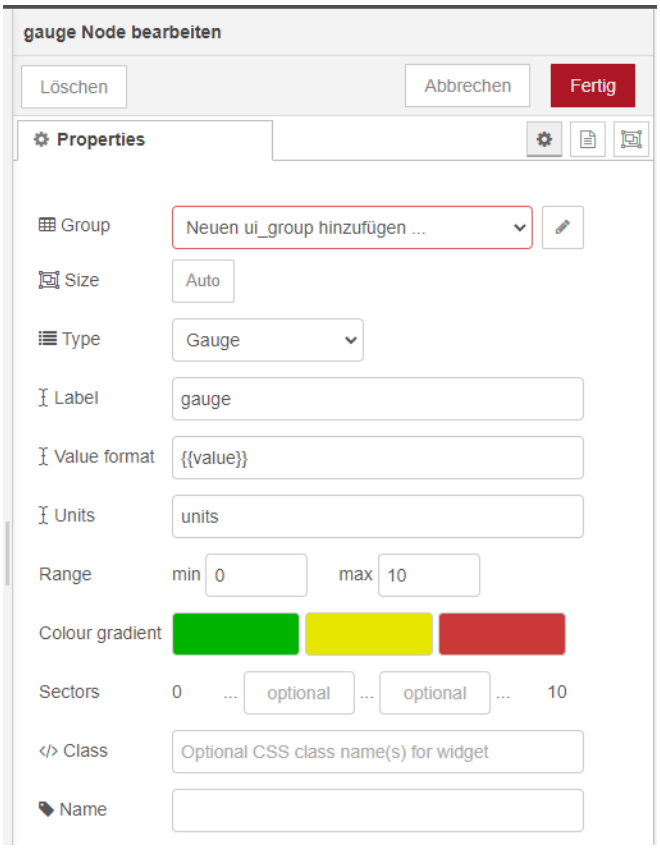


Abbildung 13: Dashboard-Installation starten

Wir beenden die Installation mit Linksklick auf **schließen**. In der Palette finden wir jetzt den neuen Reiter **Dashboard**. Hier drin ist viel mehr, als wir im Moment brauchen.

Abbildung 14: Node-RED - Dashboard Nodes

Ein paar Informationen zum Aufbau des Node-RED-Fensters vorneweg. Die Widgets oder Nodes werden im Flow am besten in Gruppen organisiert, die wiederum Tabs untergeordnet sind. Es können sowohl eigene Gruppen wie auch eigene Tabs erstellt werden. Durch die Angabe der Breite der Gruppen kann man beeinflussen wie viele davon nebeneinander dargestellt werden. Wir werden jetzt die Gruppe **keller** im Tab **home-assistant** im gleichen Zug mit dem ersten **gauge**-Widget, einer Zeigeranzeigeeinheit, anlegen.



Wir markieren den **debug-Node,** löschen ihn mit der **Entf**-Taste und ziehen einen **gauge-Node** aus dem Ordner **Dashboard** auf die Arbeitsfläche. Ein Doppelklick darauf öffnet das Editorfenster zum Festlegen der Properties.

Abbildung 15: gauge-Node bearbeiten - Gruppe hinzufügen

Bei **Group** öffnet ein Klick auf den **Stift** rechts neben dem Feld ein weiteres Fenster.

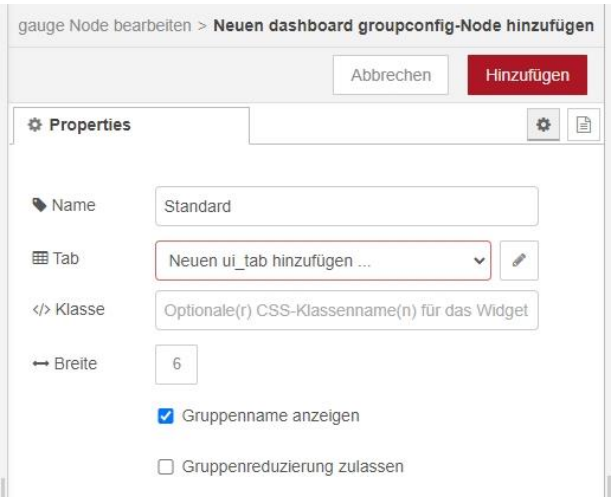


Abbildung 16: gauge-Node bearbeiten - group-Node hinzufügen

Und wieder klicken wir auf den Stift, denn es muss ein Tab eingerichtet sein, bevor wir eine Gruppe darin platzieren können. Wir überschreiben Home mit **home-assistant** und klicken auf **Hinzufügen**.

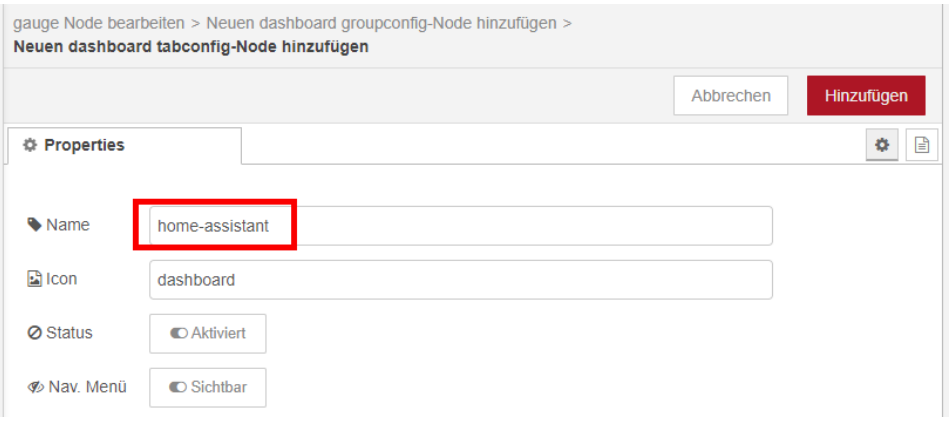


Abbildung 17: gauge-Node bearbeiten - Tab-Node hinzufügen

Wir landen wieder im Fenster zum Hinzufügen der Gruppe, der wir jetzt den Namen **Keller-Temperatur** geben. Mit einem Haken erlauben wir, dass die Gruppe zusammengefaltet, sprich ausgeblendet werden kann und schließen das Fenster mit **Hinzufügen**.

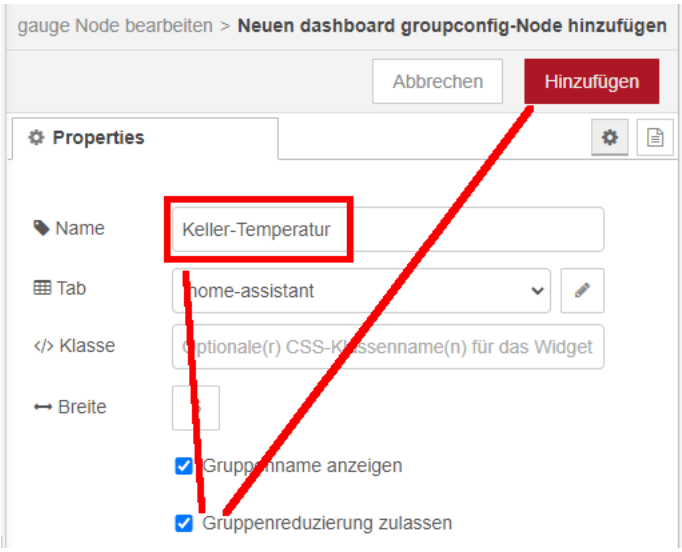


Abbildung 18: gauge-Node bearbeiten - Gruppenname hinzufügen

Zurück im Fenster gauge-Node, stellen wir als Erstes die Größe der Anzeige in Rastereinheiten ein. Die restlichen Angaben sind aus der nachfolgenden Abbildung ersichtlich.

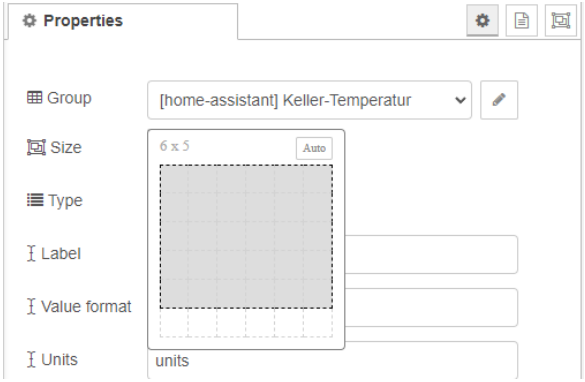


Abbildung 19: gauge-Node bearbeiten - Größe der Anzeige bestimmen

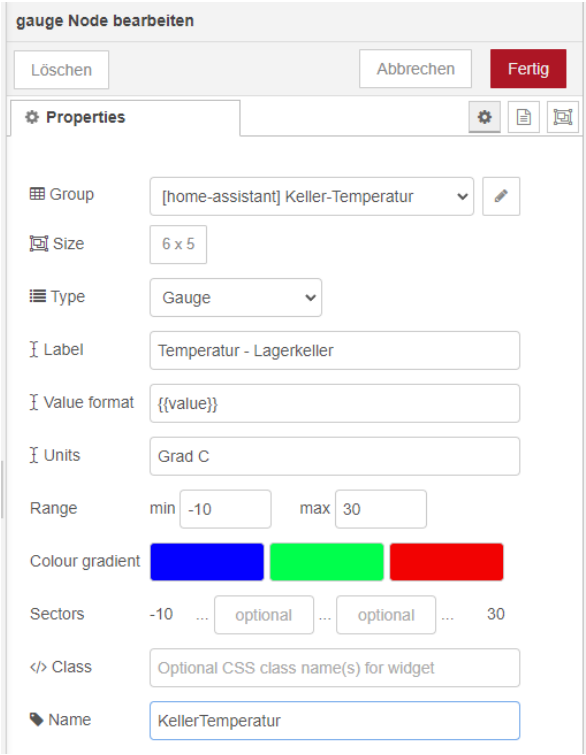
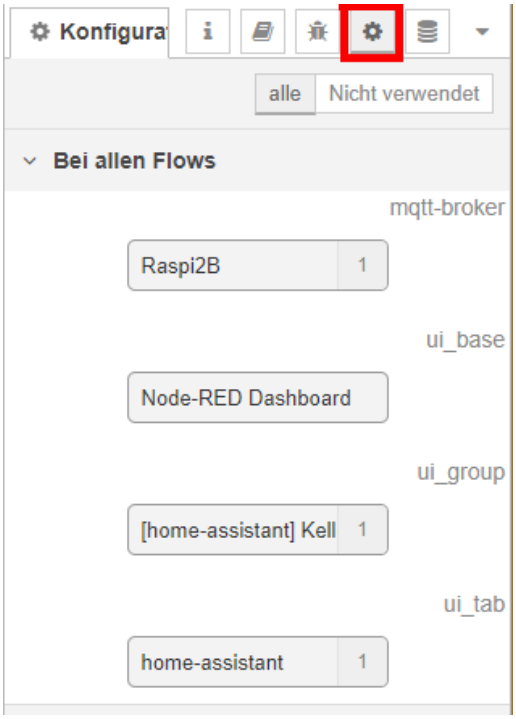


Abbildung 20: gauge-Node bearbeitet - Fertig

Klick auf **Fertig** beendet das Justieren der Eigenschaften. Im **Flow 1** ist jetzt auch das kleine rote Dreieck am gauge-Node verschwunden, das anzeigte, dass das Widget noch nicht konfiguriert war. Statt **gauge** ist jetzt auch der Name **KellerTemperatur** eingetragen.

Abbildung 21: Zurück im Flow 1



Das Fenster **Konfigurations-Node** zeigt uns eine Zusammenfassung unserer letzten Maßnahmen. Von hier aus kann man auch durch Doppelklick auf den entsprechenden Eintrag dessen Eigenschaften erneut editieren.

Abbildung 22: Die neuen Konfigurations-Nodes

Der Wechsel zur Dashboard-Spalte zeigt die Hierarchie der bisherigen Nodes.

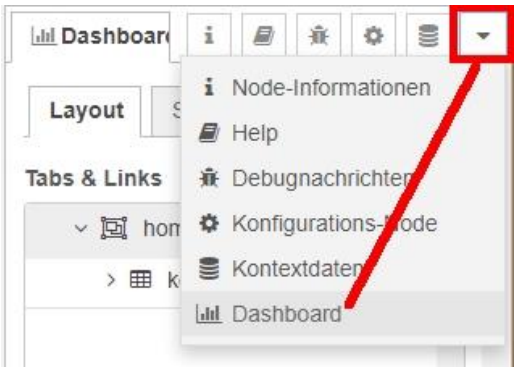


Abbildung 23: Zur Dashboard-Ansicht

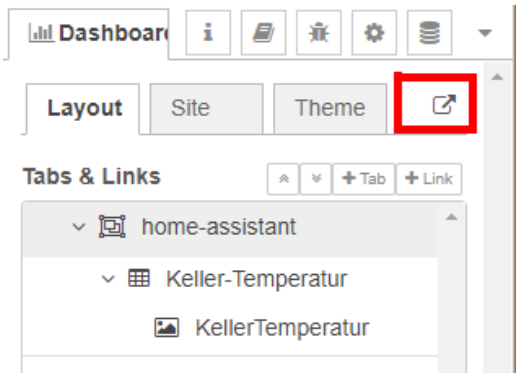


Abbildung 24: Dashboard Hierarchie

Na, da ist aber jetzt nach so viel Action ein Test angesagt. also – **deploy** und Klick auf die markierte Schaltfläche mit dem schräg nach oben weisenden Pfeil. – Wow!

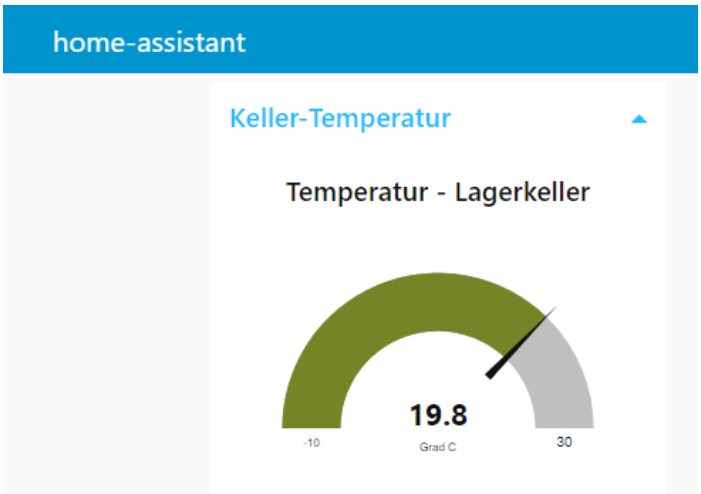


Abbildung 25: Anzeige der Kellertemperatur

Wenn wir das Verhalten der Anzeige für andere Temperaturwerte prüfen wollen, müssen wir nicht unbedingt den DHT-Client in die Gefriertruhe legen oder ein Feuerchen darunter entfachen, nein, es reicht, wenn wir ein Terminal zum Raspi öffnen und folgende Befehle abschicken. Beobachten Sie dabei die Temperaturanzeige.

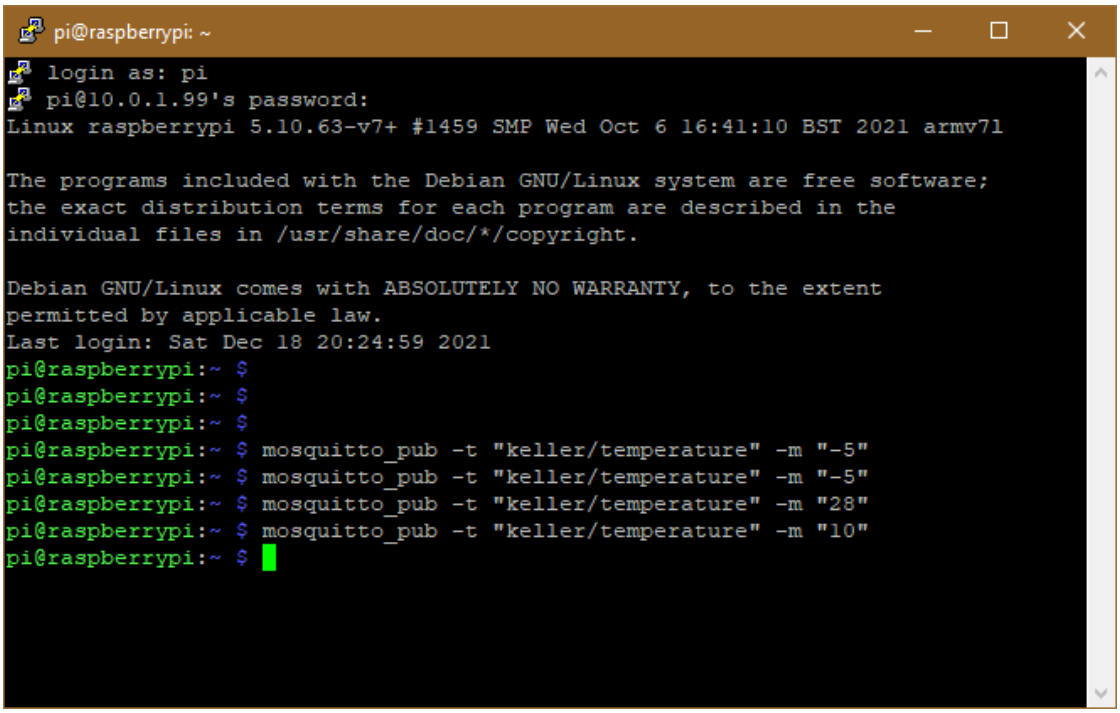


Abbildung 26: So testen wir andere Temperaturen

Bis zum nächsten automatischen Update der Temperatur durch den DHT-Client wird unser gefakter Wert angezeigt. Dem Broker ist es nämlich Wurscht was er empfängt und weitergibt, der hat ja keine Ahnung, wer die Nachricht veröffentlicht hat. Er bekommt die Nachricht und reicht sie an Node-RED weiter, fertig. Sie meinen, das ist eine ganz üble Sicherheitslücke? Wo Sie Recht haben, haben Sie Recht! Um die Lücke zu schließen gibt es mehrere Ansätze, aber das führt im Moment zu weit und ist das Thema für eine eigene Blogepisode. Wir haben ja noch viel zu erledigen, bis unsere Anwendung komplett ist.

Ein neuer **mqtt in**- Node und ein weiteres **gauge**-Widget fliegen auf der Arbeitsfläche ein und werden verbunden.

Abbildung 27: Die Feuchtigkeit kommt ins Spiel

Dieses Mal geht die Konfiguration schneller. Der Server ist bereits bekannt, Topic und Name werden ergänzt - **Fertig**.

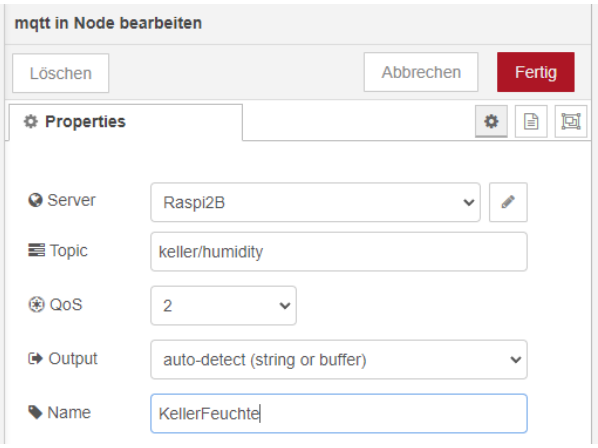


Abbildung 28: Relative Luftfeuchte - mqtt-Node Properties

Beim gauge-Node geht es nicht ganz so fix. Weil die beiden Anzeigen nebeneinander erscheinen sollen, müssen wir dem Feuchte-Node eine neue Gruppe zuordnen.

Doppelklick auf das gauge-Widget bringt uns in das Eigenschaftenfenster.

Abbildung 29: gauge-Node bearbeiten - neue Gruppe erzeugen

Um eine neue Gruppe anzulegen, klappen wir die Liste auf und klicken mit der Auswahl **Neuen ui_group hinzufügen** auf den **Stift**. Den Namen eintragen, **Gruppenreduzierung zulassen** und **Hinzufügen**.

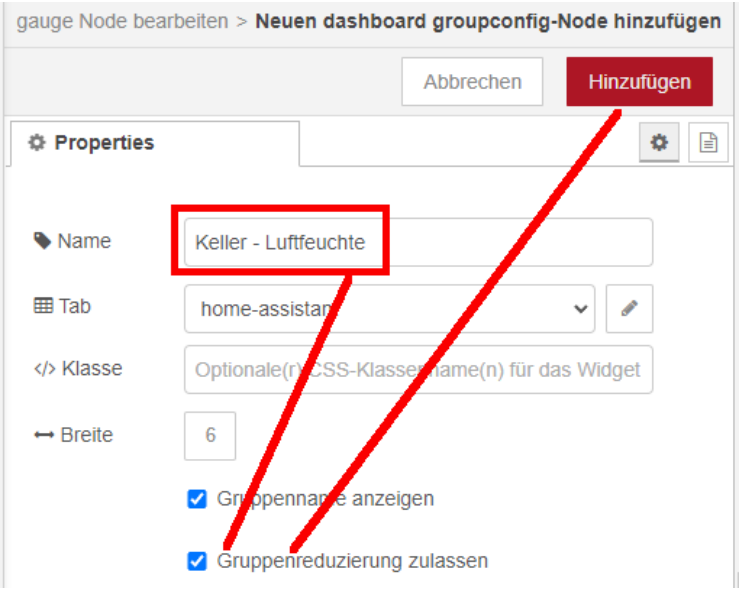


Abbildung 30: gauge-Node bearbeiten - Eigenschaften der neuen Gruppe

Die restlichen Properties folgen wieder aus der Abbildung. - **Fertig**

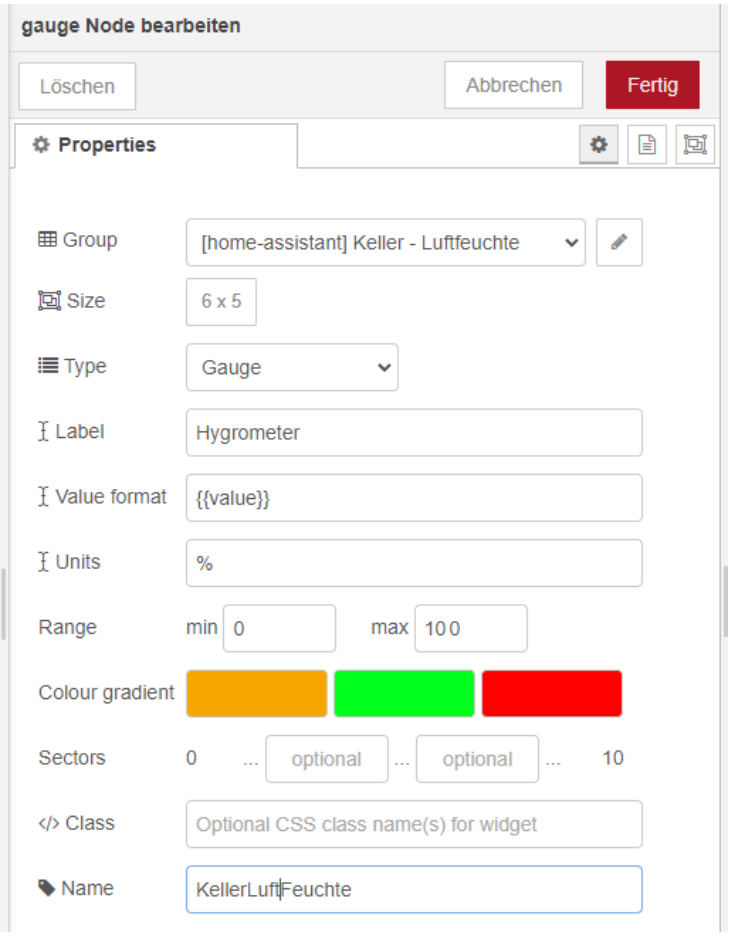


Abbildung 31: gauge-Node bearbeiten - Eigenschaften der Anzeige

Sie können auch einmal mit den Farben experimentieren. Mit **Sectors** kann man übrigens die Positionen angeben, an denen der Farbumschlag exakt stattfinden soll. Ohne diese Festlegung werden Mischfarben erzeugt, die Farbänderung erfolgt weich.

Zum Test muss der Flow erneut deployed werden. Die blauen Punkte an den Nodes deuten an, dass diese Nodes noch nicht oder nach einer Änderung der Eigenschaften nicht erneut deployed wurden. Wechseln wir jetzt einfach in das Dashboard-Fenster des Browsers, es müsste ja noch offen sein.

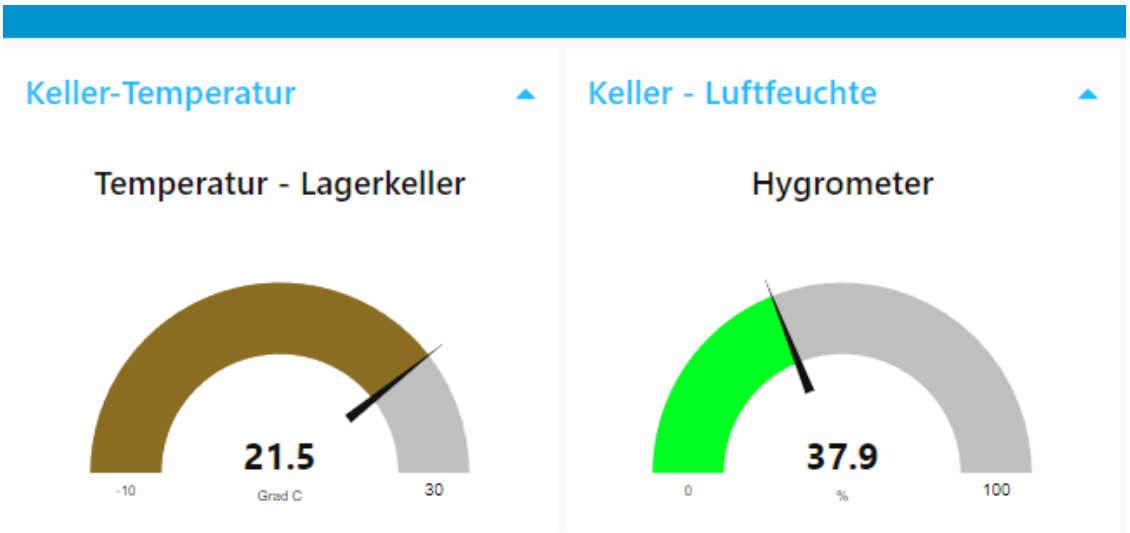


Abbildung 32: Beide Anzeigen im Dashboard

Natürlich kann auch das Hygrometer über das Terminalfenster getestet werden.

```
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "95"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "10"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "95"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "20"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "20"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "22"
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "21"
pi@raspberrypi:~ $
pi@raspberrypi:~ $ mosquitto_pub -t "keller/humidity" -m "22"
pi@raspberrypi:~ $
```

Abbildung 33: Hygrometer-Test

Jetzt fehlen noch das Schalten des Lüfters und die Rückmeldung über den Erfolg der Aktion.

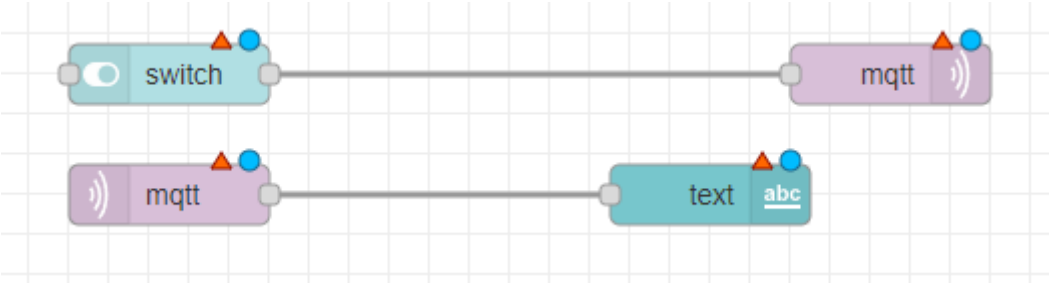


Abbildung 34: Schalten und Rückmeldung des Lüfters

Die Widgets **switch** und **text** wohnen in der Dashboard-Palette, **mqtt-in** und **mqtt-out** sind in **network** zu Hause. Wir holen sie auf die Arbeitsfläche. Die beiden Flows kommen in eine neue Gruppe mit dem Namen **Keller-Lüfter**, die wir so, wie beim zweiten gauge-Widget erzeugen. **On Payload** wird verschickt, wenn der Schalter angeschaltet wird. Wir versenden einen **String** und zwar "**an**". Analog behandeln wir **Off Payload**. Die Angabe des Topics ist wichtig und der Name rundet das Ganze ab – **Fertig**.

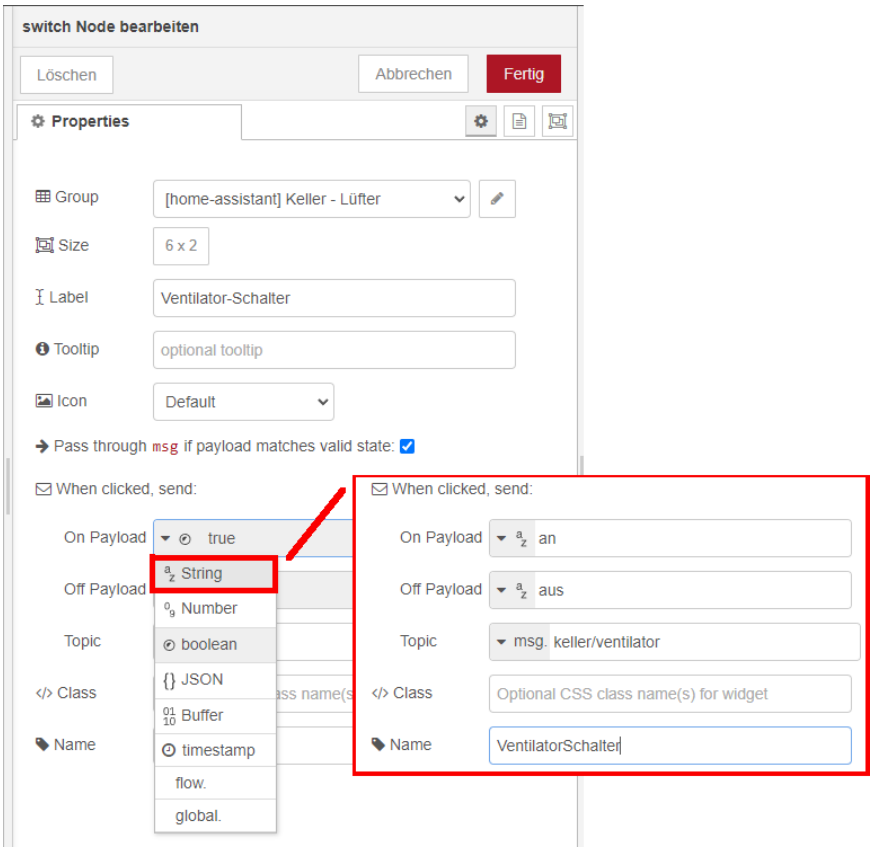


Abbildung 35: Switch-Node Properties

So sieht der Publisher der Schaltaktion aus. - **Fertig**

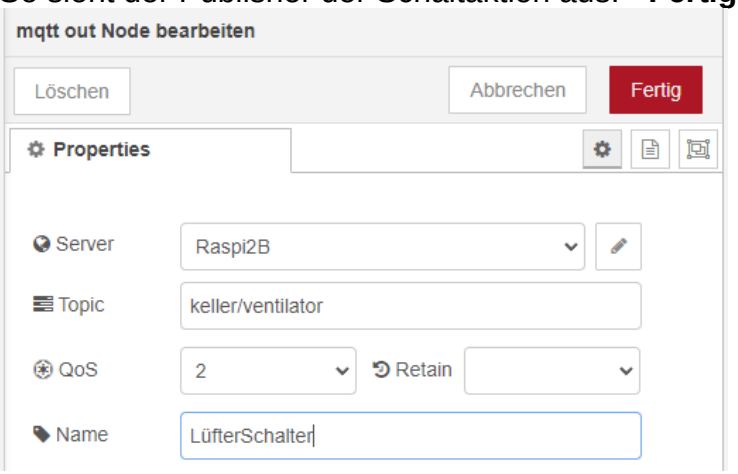


Abbildung 36: Publisher-Node für den Schalter

Mit dem **mqtt-in**- Node holen wir die Rückmeldung über die Schalterstellung vom DHT-Client ab.

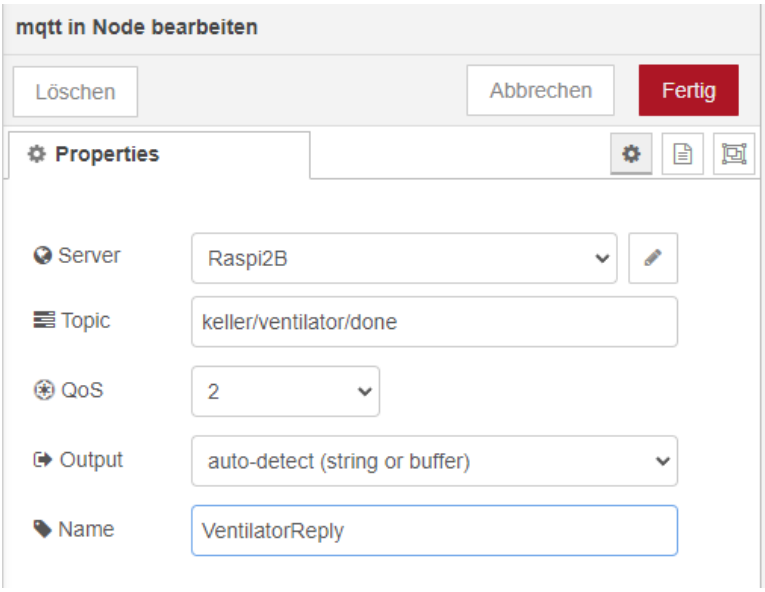


Abbildung 37: mqtt-switch Subscriber Properties

Und hier noch die Properties des Text-Nodes. - **Fertig**

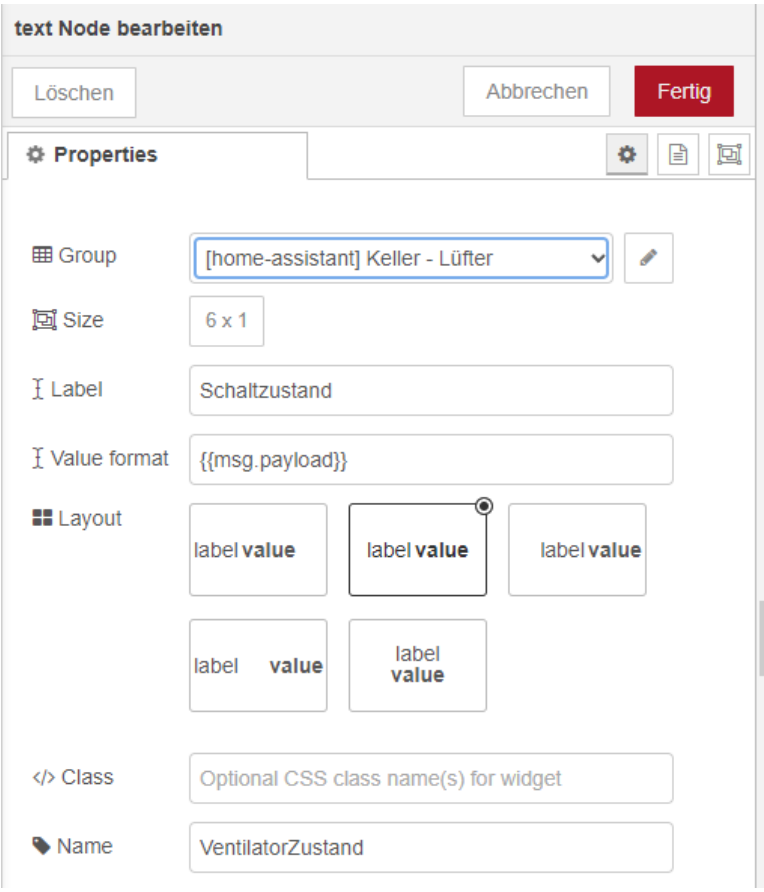


Abbildung 38: Text-Node Eigenschaften

Dann **deploy** und "schau mer mal, dann seh'n mer scho" (F. Beckenbauer), Wechsel zum Dashboard-Window.

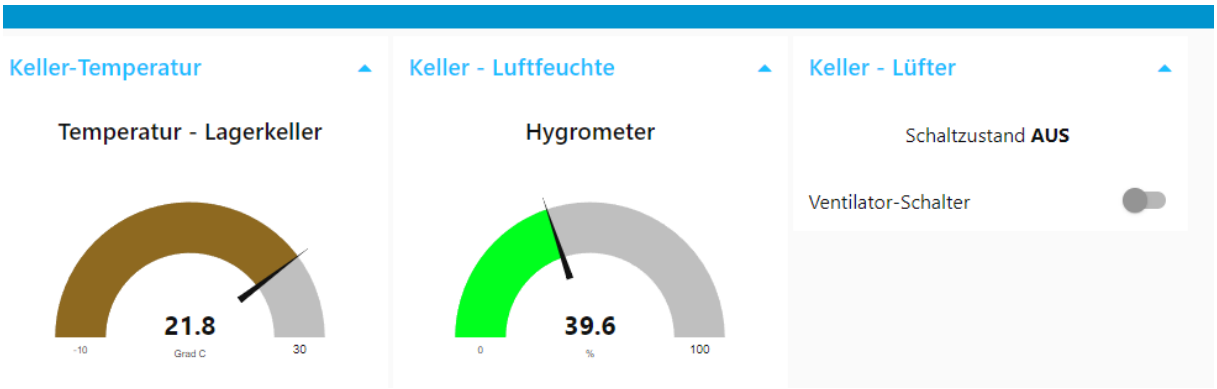


Abbildung 39: DHT-Client fertig eingebunden

Zurück zum node-RED-Fenster. Die Dashboard-Karte zeigt was wir bisher alles gemacht haben und sie verbirgt ein Geheimnis.

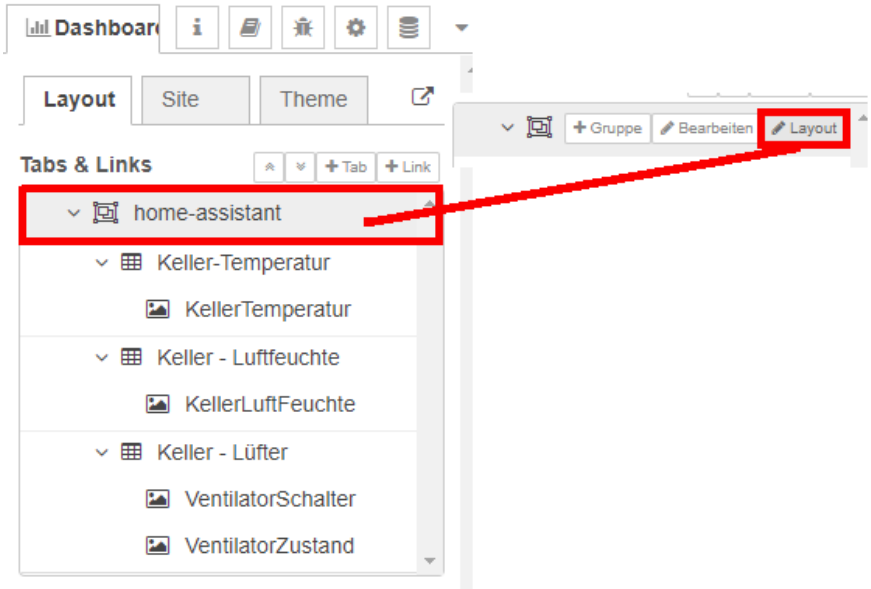


Abbildung 40: Die Dashboard-Karte zeigt den Zusammenhang

Fährt man nämlich mit der Maus über die Tab-Zeile, dann erscheinen weitere Schaltflächen. Wir können eine Gruppe hinzufügen, den bestehenden Tab bearbeiten und die **Anordnung der Gruppen** verändern. Das ist aber noch nicht alles. Die Einträge lassen sich mit der Maus per Drag and Drop verschieben. Damit ändert sich die Reihenfolge im Dashboard-Fenster. Auch die Verschiebung von Flows in andere Gruppen ist möglich, und über Layout kann die vertikale Position im Fenster verändert werden.

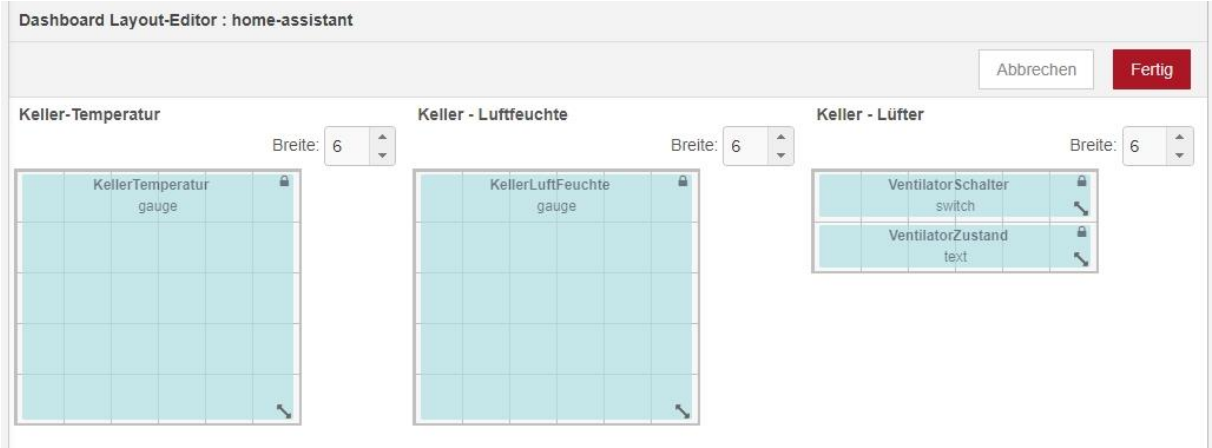


Abbildung 41: Gruppenanordnung vor der Anpassung

Interessant sind die Schaltflächen +Tab und +Gruppe, mit denen sich weitere Tabs und Gruppen anlegen lassen, auf die man dann bei der Auswahl in den Dashboard-Widgets zugreifen kann.

Hat man für eine Gruppe keine Verwendung mehr, lässt sie sich aus der Karte aber nicht löschen. Man muss auf bearbeiten klicken. Im Eigenschaftenfenster links oben ist der Button zum Löschen versteckt.

Damit sind wir fast am Ende der heutigen Folge. Eins fehlt noch der Export der gesamten Arbeit. Über das Menü wählen wir **Exportieren**.

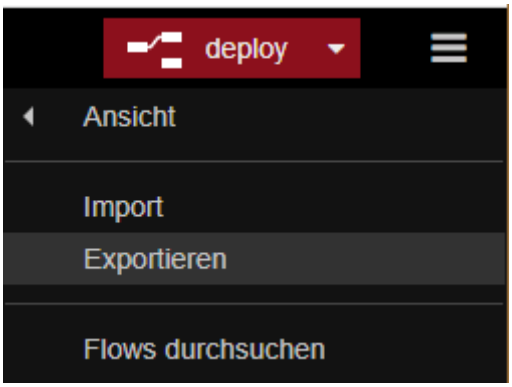


Abbildung 42: : Menü - Export

Wir kopieren den formatierten JSON-Code in die Zwischenablage und importieren ihn in ein neues, leeres Fenster in Thonny. Von hier speichern wir mit einem beliebigen Dateinamen ab.

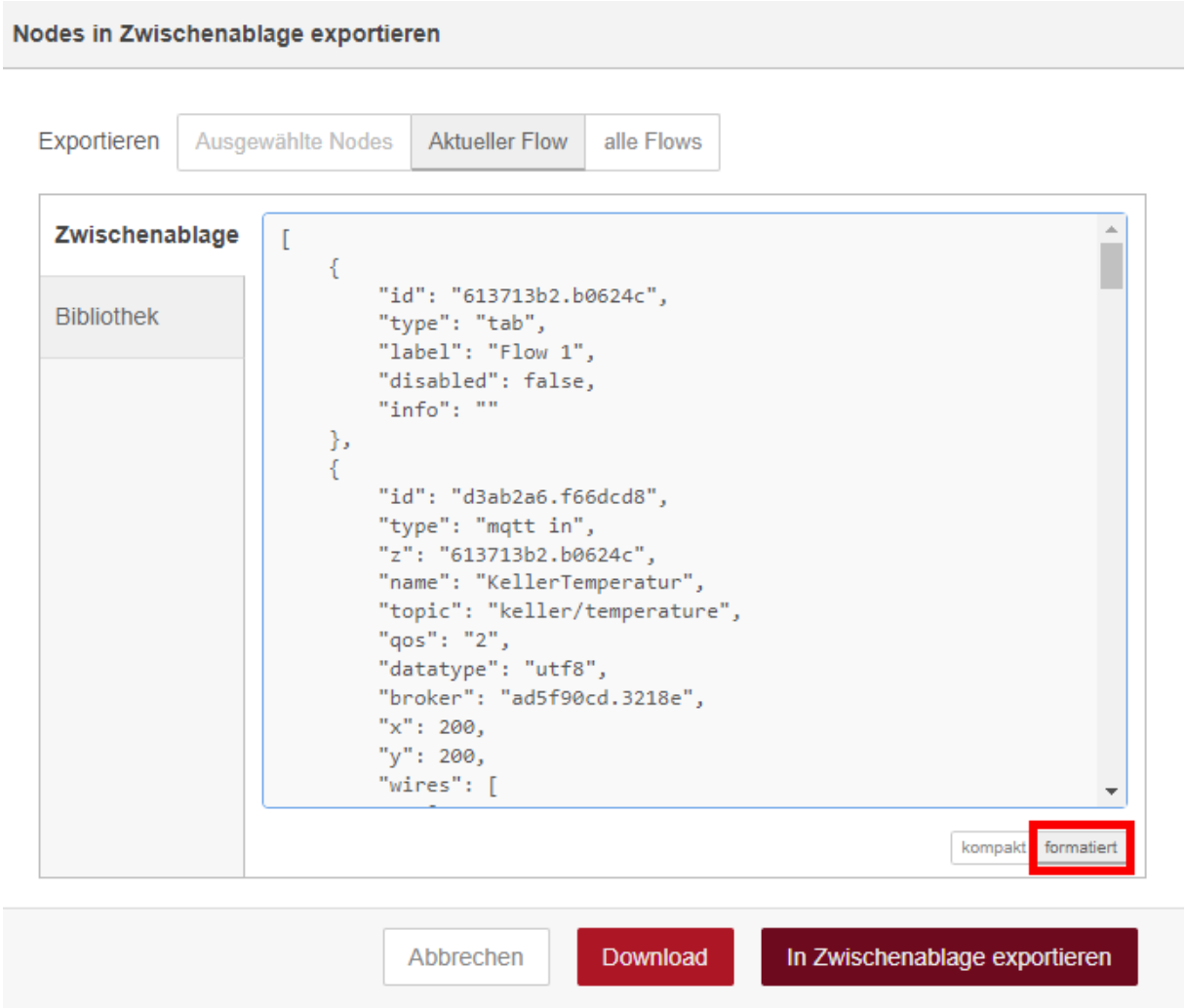


Abbildung 43: Kopie in die Zwischenablage

Man kann den Text auch als Datei mit Download herunterladen. Er landet dann in dem Ordner, der im Browser als Download-Ordner eingestellt ist.

Noch eine Kleinigkeit als Rausschmeißer. Ihnen gefallen die vorgegebenen Farben nicht? Nicht schlimm, vergeben Sie einfach selber für die diversen Objekte Farben Ihrer Wahl. Den Editor erreichen Sie über **Dashboard – Theme – Stil - Benutzerdefiniert**.

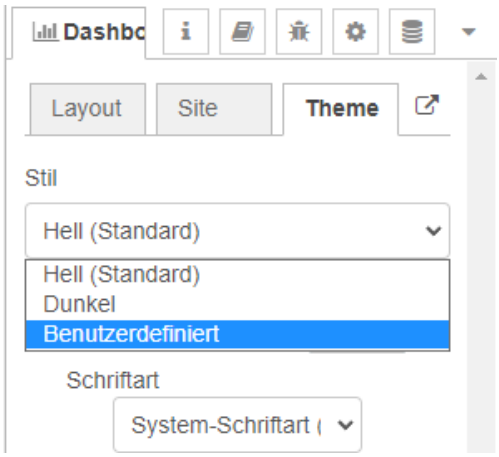


Abbildung 44: Ein benutzerdefiniertes Theme erstellen

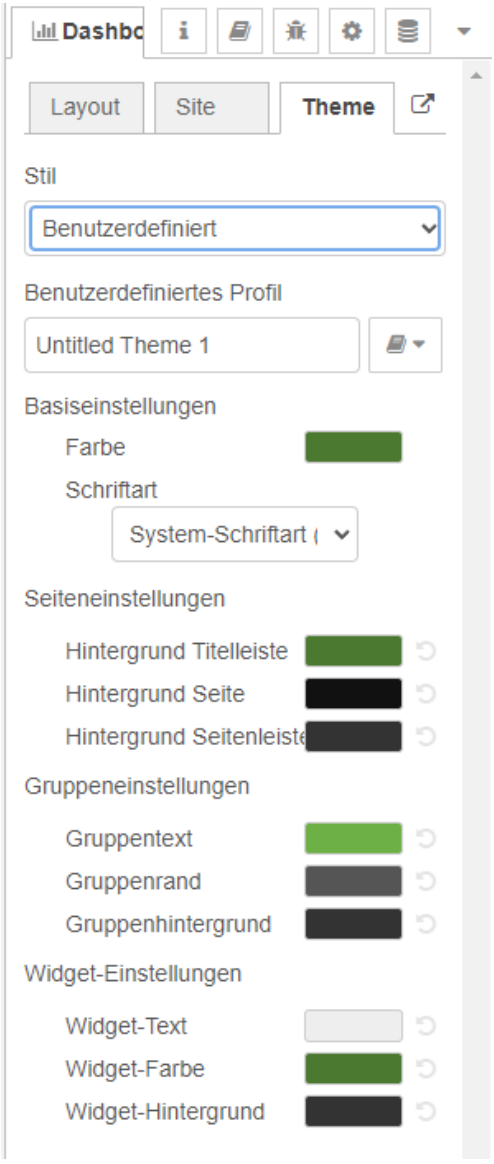


Abbildung 45: Benutzerdefinierte Farben

Als Hausaufgabe schaffen sie es jetzt sicher leicht, das Dashboard um folgende Features zu erweitern.

- Einen neuen Tab **Heizung** anlegen.
- Vor- und Rücklauf-Temperatur in einer neuen Gruppe **Temperaturen** in **Heizung** anlegen.
- Schalter für Brenner und Pumpe samt Rückmeldung in einer neuen Gruppe **Schalter** im Tab **Heizung** integrieren.
- In der Gruppe KellerTemperatur ein Chart-Widget hinzufügen, um die Temperatur über 24 Stunden zu Plotten

Ach ja, die JSON-Datei des heutigen Projekts steht zum Download bereit. Öffnen Sie die Datei zum Beispiel in Thonny, markieren Sie den Text (Strg + A) und kopieren Sie ihn in die Zwischenablage. In Node-RED importieren Sie ihn durch Strg + I und Strg + V. Viel Spaß beim Programmieren, Ausprobieren und Entdecken.

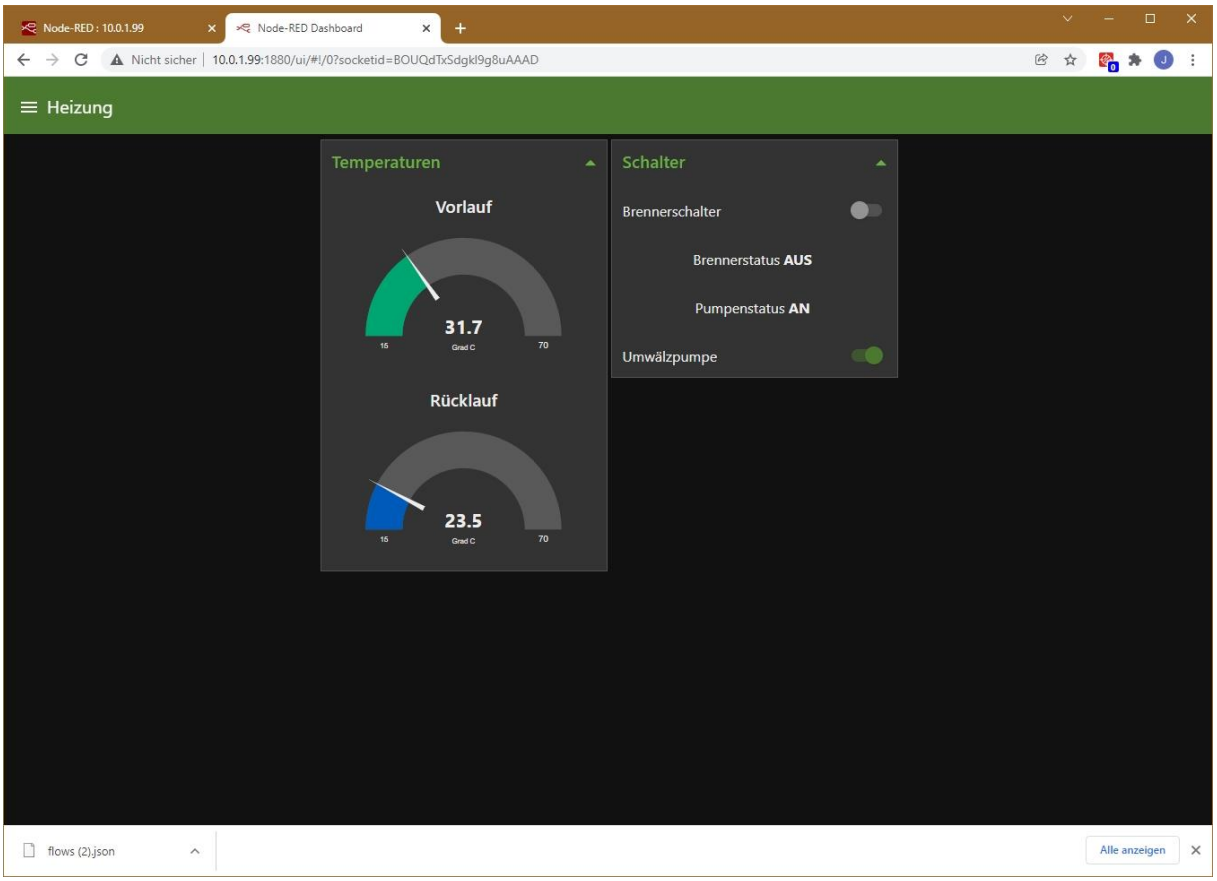


Abbildung 46: hausaufgabe-1