

Grundübersicht

Flows und Nodes

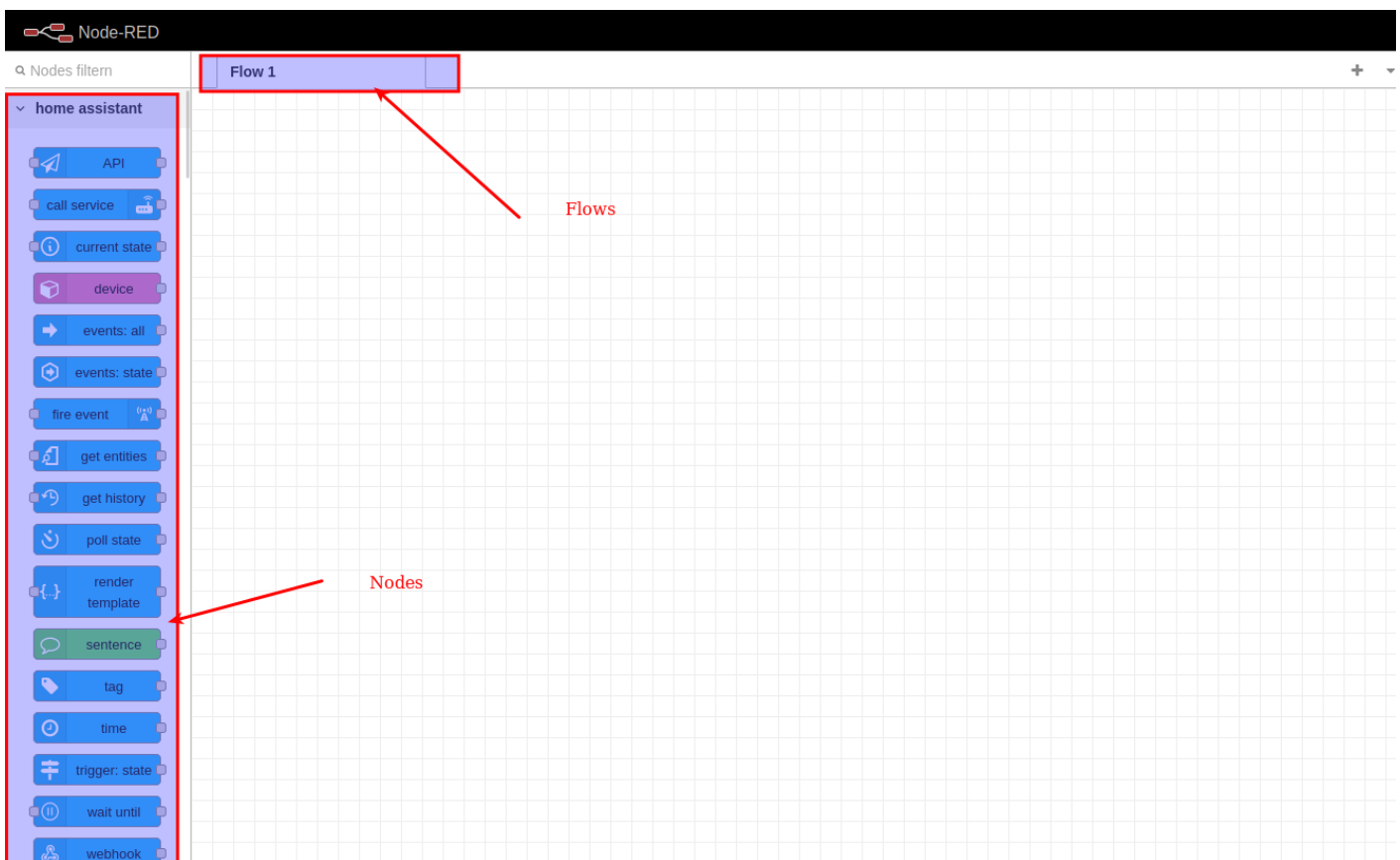
Übersicht

Flows sind die Seiten auf den Nodes plaziert werden. Mann kann mehrere Flows haben. Vergleichbar mit Szenen.

Somit wird es übersichtlicher. Man kann natürlich auch alles auf einen Flow packen. Aber der Übersichthalber kann man halt mehrere Flows anlegen.

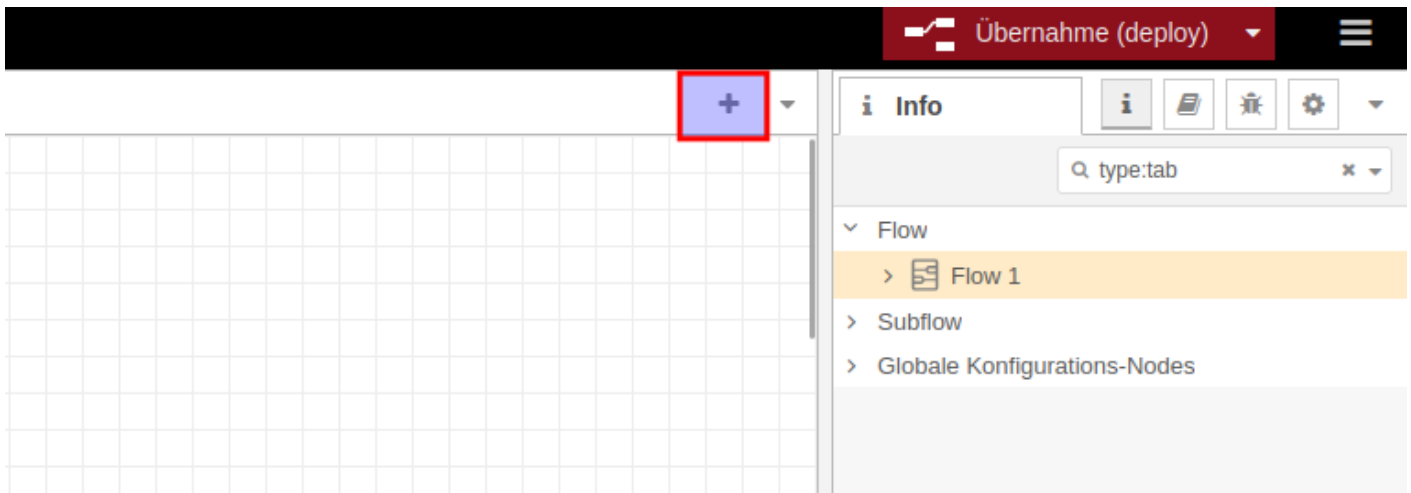
Die Nodes sind auf der Linke Seite.

Nodes sind unsere Bausteine mit den wir unseren Flow bauen.

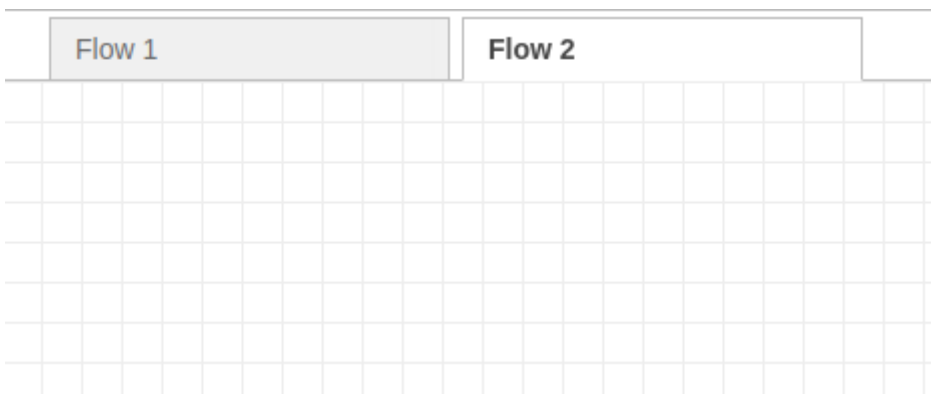


Flow hinzufügen

Um ein weiteren Flow hinzu zu fügen klicken wir auf das + Symbol



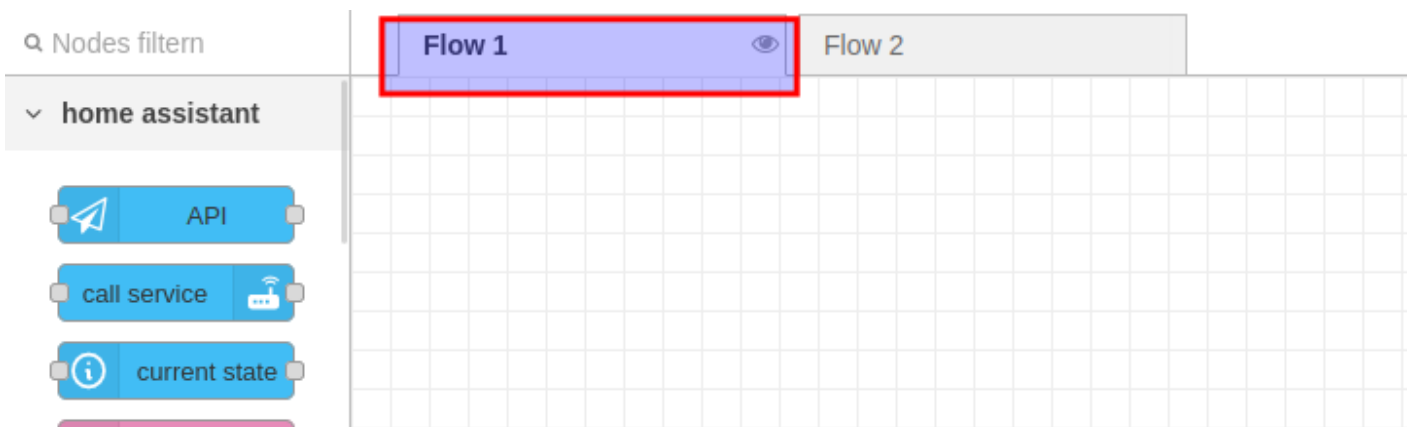
Nun haben wir einen zweiten Flow im Tab daneben.



Im Rechten Fenster sehen wir auch nochmal unsere Flows.

Flow umbenennen

Um einen Flow umzubenennen, doppelklick auf den Tab des Flows den man umbenennen möchte



Nun dort den Text ändern. Wir haben auch die Möglichkeit eine Beschreibung hinzuzufügen. Darin kann man zum Beispiel noch mal seine Abläufe erläutern, damit man nach einem Jahr auch

noch weiß was womit gemeint war, so ne Art Doku.

Flow bearbeiten: Flow 1

Löschen Abbrechen **Fertig**

⚙️ **Eigenschaften** ⚙️ ☰

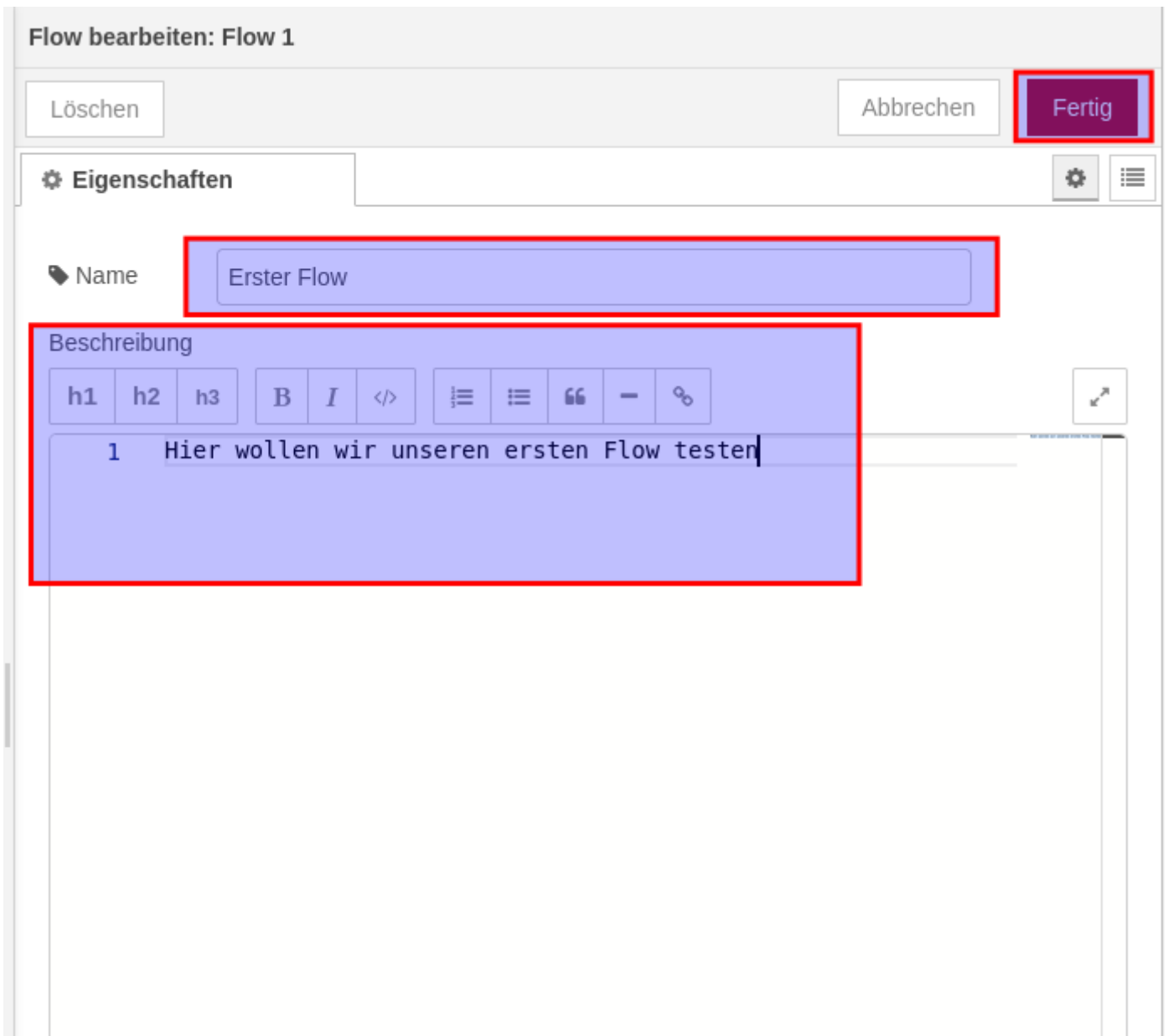
👤 **Name** Flow 1

Beschreibung

h1 h2 h3 **B** *I* </> ☰ ☰ “ - 🔗 ↗

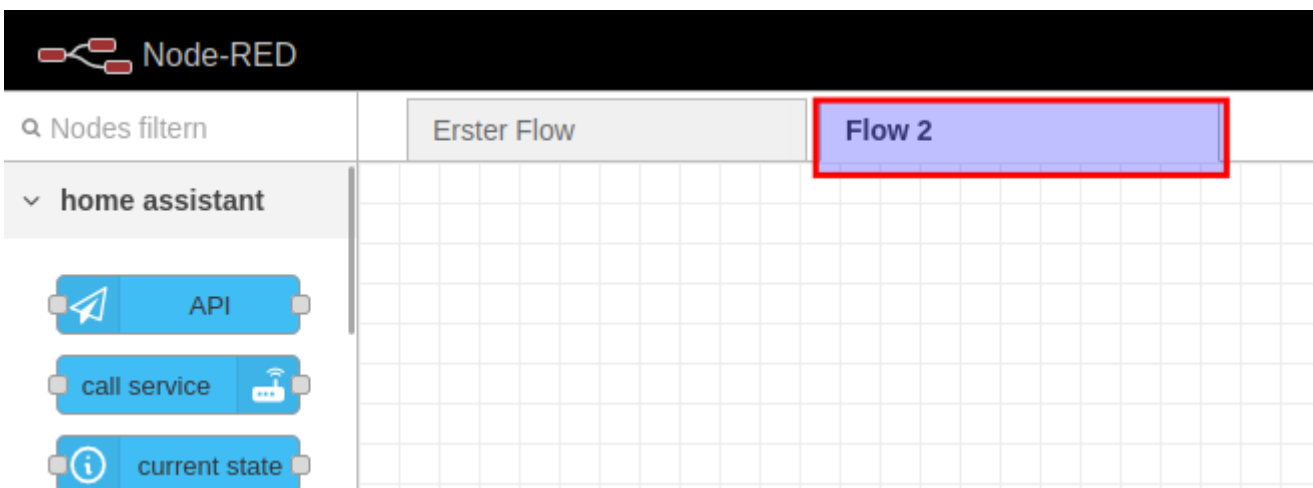
1

Geändert und eine Beschreibung eingefügt. Nun auf Fertig klicken.

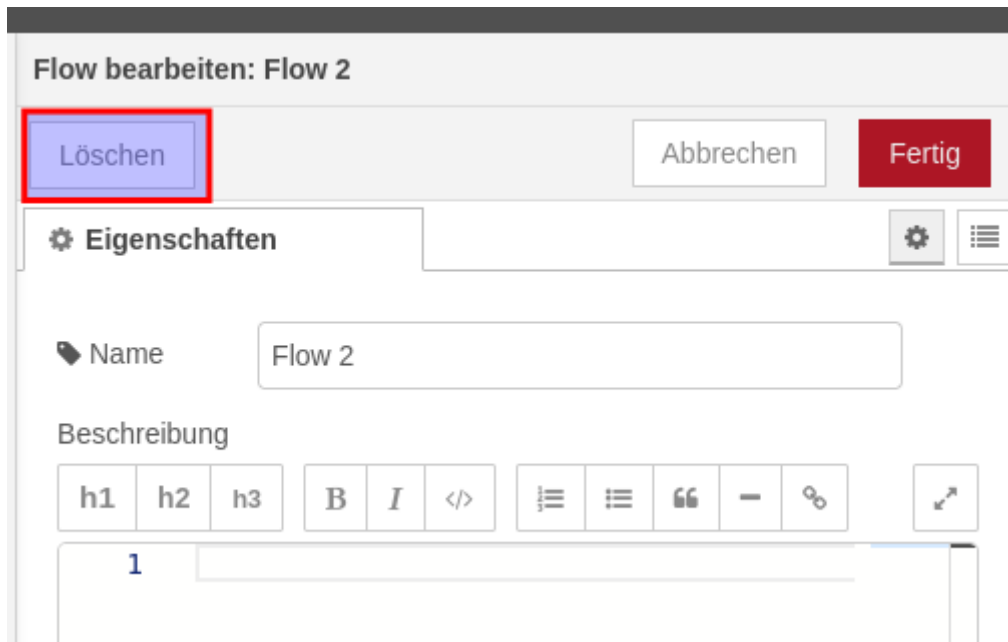


Flow löschen

Doppelklick wieder auf den Reiter des zu löschenden Flows



Und dort dann löschen anklicken



Nodes

Plazieren von Nodes

Um ein Node auf den Flow zu ziehen, den node den man haben möchte aus der Liste Links mit der linken Muastatse gedrückt halten und auf den Flow ziehen

Node-RED

Nodes filtern

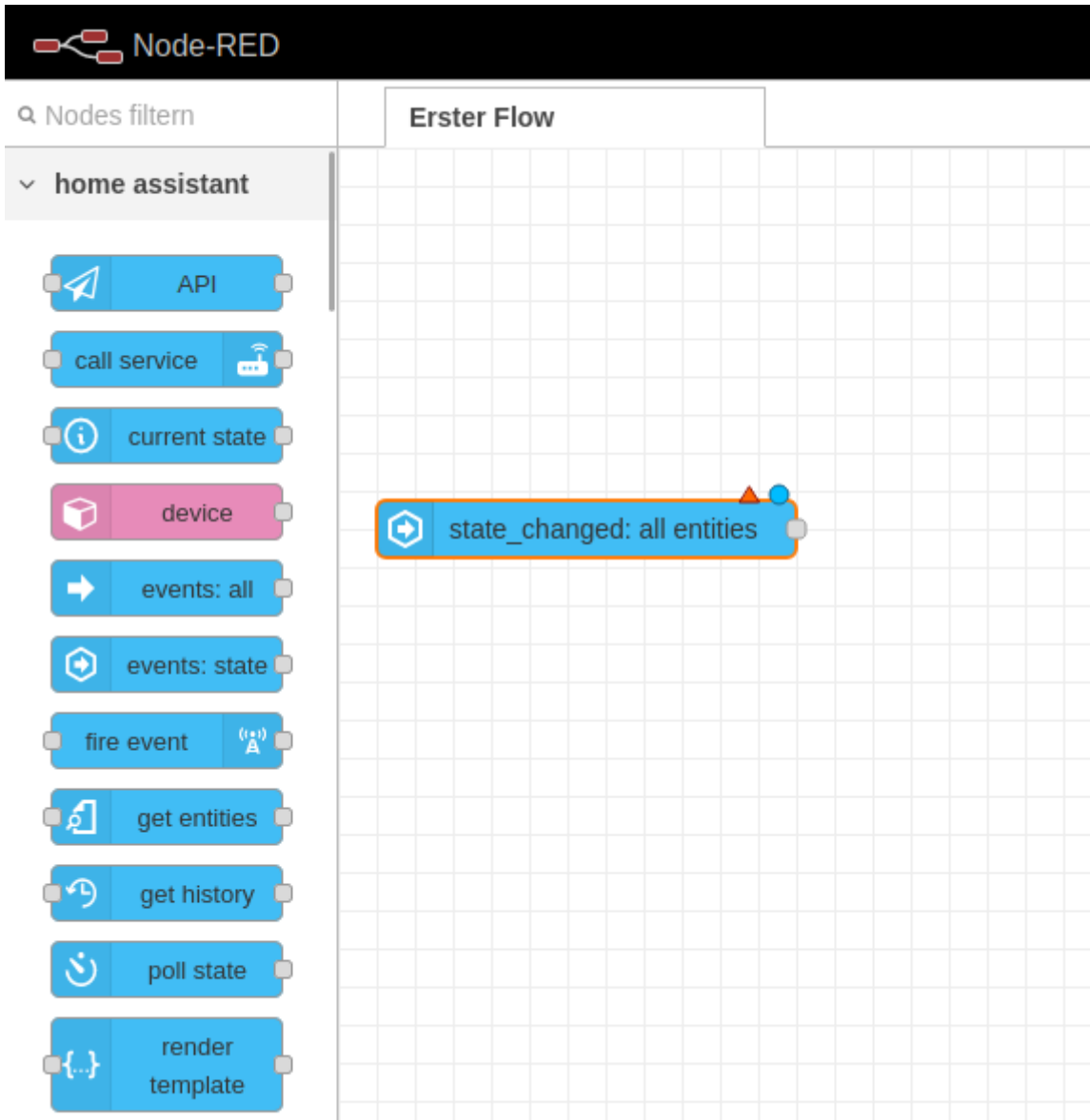
Erster Flow

home assistant

- API
- call service
- current state
- device
- events: all
- events: state**
- fire event
- get entities
- get history
- poll state
- render template
- sentence
- tag
- time
- trigger: state
- wait until

Links auf den Node gedrückt halten und auf die Fläche ziehen

Nun ist die Node plaziert.



Eigenschaften der Node anzeigen, umbenennen und löschen

Hierrüber kann die Node auch gelöscht werden, analog wie beim Tab.

Dazu doppelklick auf die Node.

Nun können wir Name vergeben oder aber über den löschen Button die Node löschen.

Node 'events: state' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name

Server Home Assistant

Entity exact

If State is a_z

For 0 minutes

State Type string

Output properties

msg. payload = entity state

msg. data = event data

msg. topic = entity id

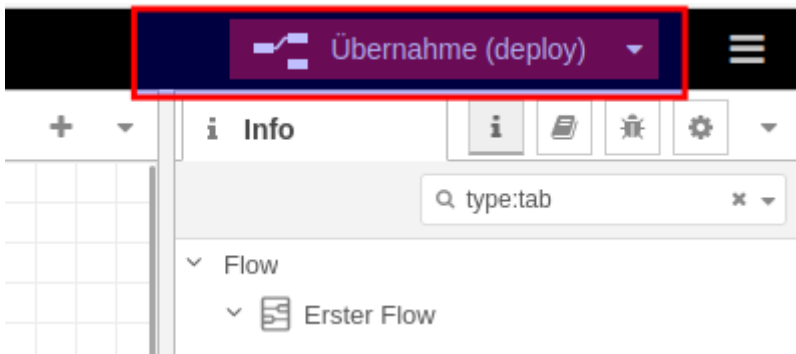
+ hinzufügen

Ignore state change event when:

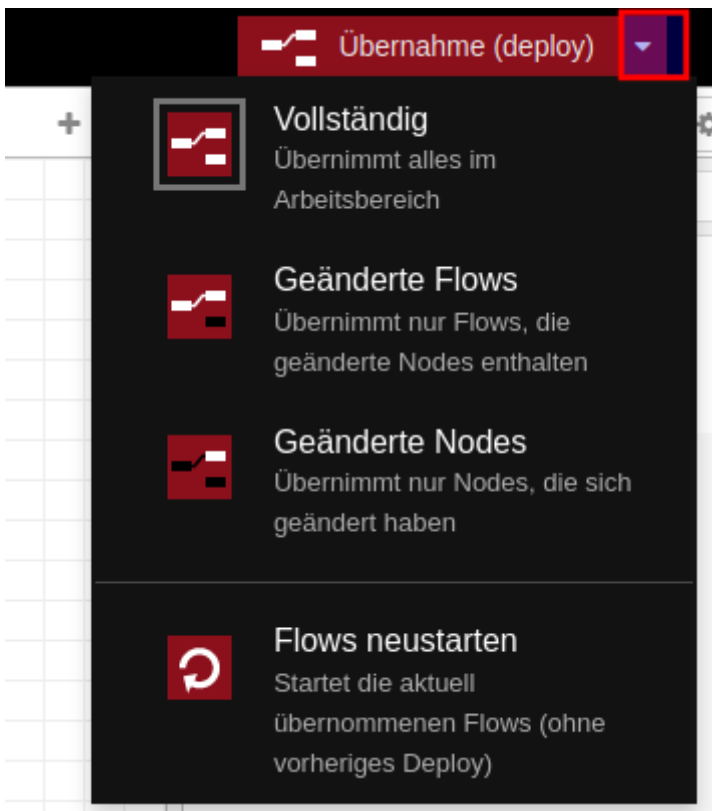
- Previous state doesn't exist
- Previous state is unknown
- Previous state is unavailable

Deployen - scharf schalten

Damit Nodes eigenschften etc. auch Aktiv geschaltet werden, gibt es oben rechts den Butto deploy. Kann man gleich setzen mit speichern oder übernehmen.



Über den Pfeil kann man auch sagen, man will nur bestimmte Bereiche Deployen möchte



Erste Automation

Denn Node RED dient ja dazu Automatisierungen Visuell zu Programmieren mit Blöcken (Nodes)
Was brauchen wir für die Automatisierung:

- Ein Gerät / Entität,
- Diese muss Eigenschaften haben irgendwas zu triggern also ein Sensor quasi (Zum Beispiel ein Temperatur Sensor)

- Dann wird eine Aktion ausgelöst, zum Beispiel eine Benachrichtigung oder ein Gerät wird geschaltet
- Das ganze kann man dann noch mit einer Statusüberwachung prüfen.

Nodes typen

Kommentar Node

Die Kommentar Node hat keine Funktion, sie dient zur Dokumentation von Bausteinen oder sonstigen infos.

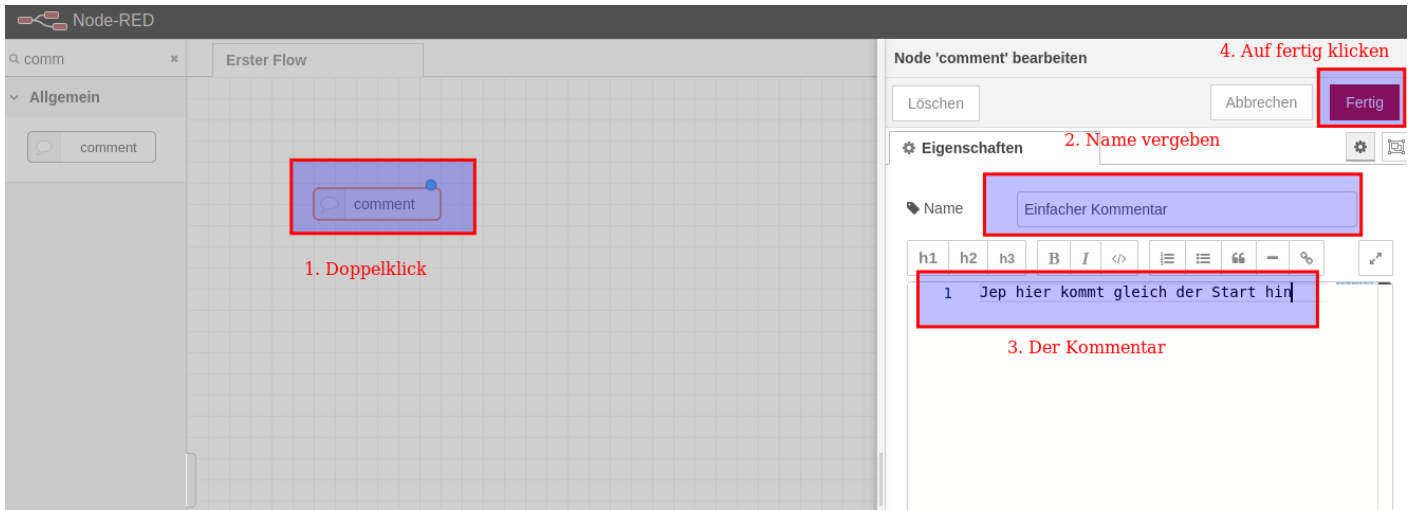
Jetzt suchen wir uns die Comment Node raus. Dazu einfach comm

Oben links ist ein suchfilter. Da kann man nach Nodes filtern.

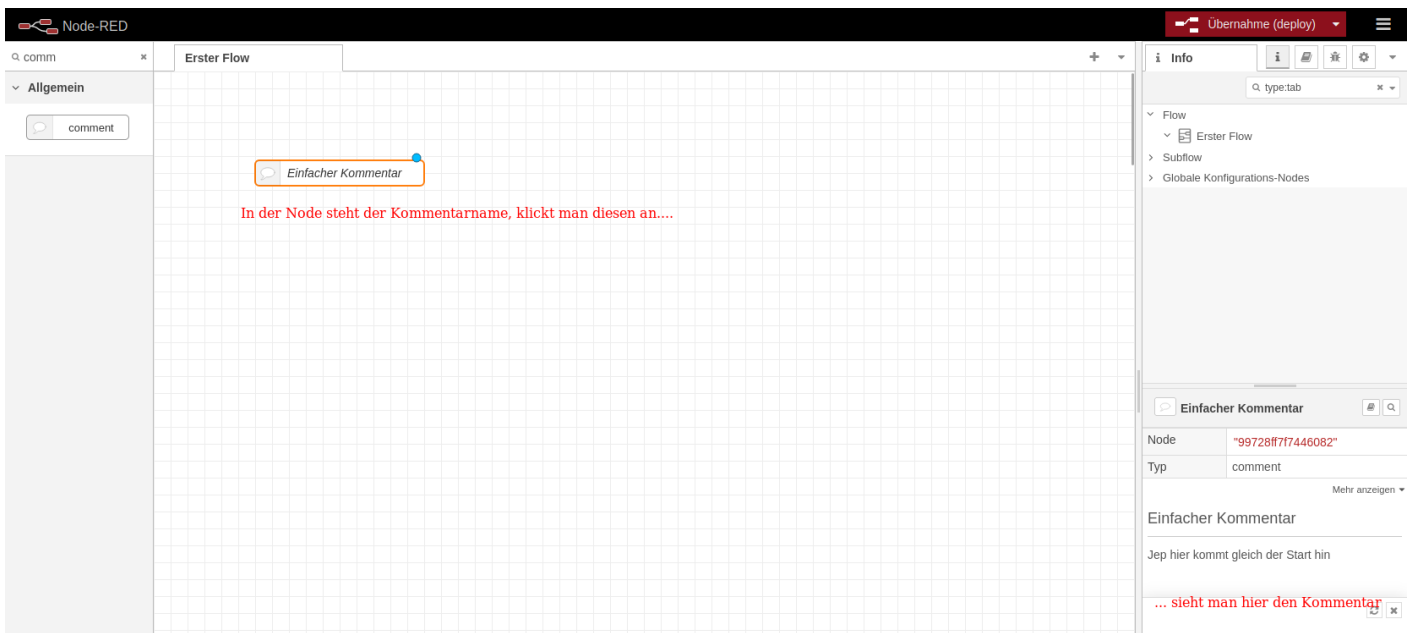
Dann braucht man nicht so viel scrollen. Diese dann wieder mit linker Taste gedrückt halten und auf die freie Fläche ziehen



Nun doppelklick auf die Node, kann man Name und den Kommentar eingeben



In der Node selbst wird der name angezeigt, klickt man die Node an. wird im rechten Fenster der Inhalt also unser Kommentartext angezeigt.



Entity Node (Ersetzt durch eine neue Gruppe von Elementen die "Home Assistant Entities")

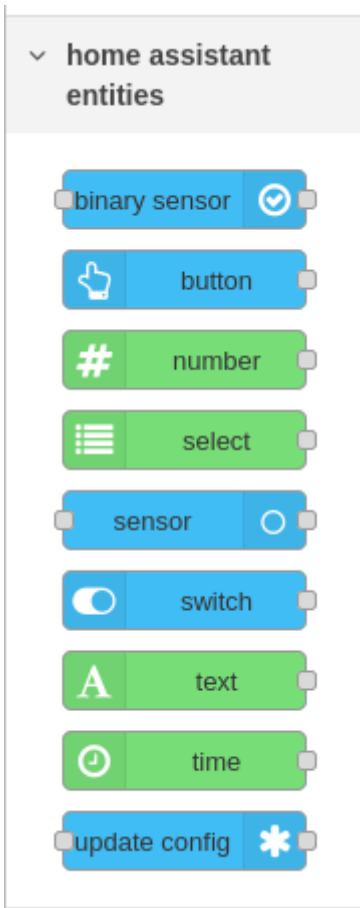
Die Entity Node ist eine Node mit der man eine Entität in Home Assistant anlegen kann. Also ein Virtuelles Gerät.

Zum Beispiel ein Schalter oder ein sensor. Diese Entität ist allerdings deprecated. Denn in diese Entität war zu unflexibel.

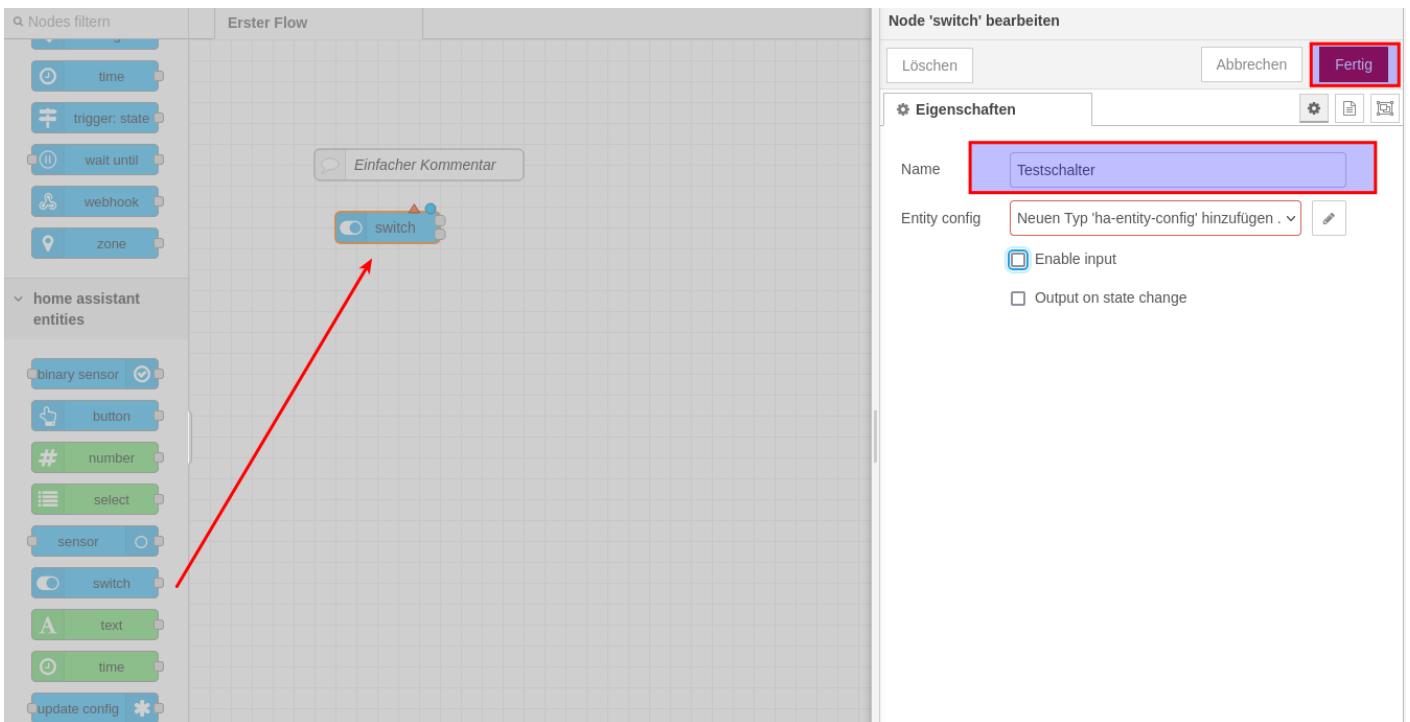
Dafür gibt es jetzt neue Nodes, mit denen Man GUI Elemente oder Sensoren in Home Assistant bauen kann.

Virtuell halt.

Am Namen kann man schon erkennen was was sein soll.



Wir ziehen uns einen Schalter (switch) auf die freie Fläche) doppelklick drauf und geben ihm den Namen. Testschalter.






klicken dann auf den Bleistift.


In Neuere Versionen ist dies ein + Button

Node 'switch' bearbeiten

Löschen Abbrechen Fertig

⚙️ Eigenschaften   

Name

Entity config 

Enable input

Output on state change

Nun Einen Namen angeben, switch auswählen und Device class auch switch.

Mit dem Stift bei Device kann man auch einen Eigenen Geräte Namen und Hersteller angeben.

Hier ist es jetzt in neueren Version auch ein + Button anstatt Stift

Wenn man möchte. Dann auf Hinzufügen

Abbrechen

Hinzufügen

⚙️ **Eigenschaften**



Name

Testschalter

Server

Home Assistant

Device

Neuen Typ 'ha-device-config' hinzufügen ...

Type

switch

Friendly name

soll irgendwas machen

Icon

switch

Category

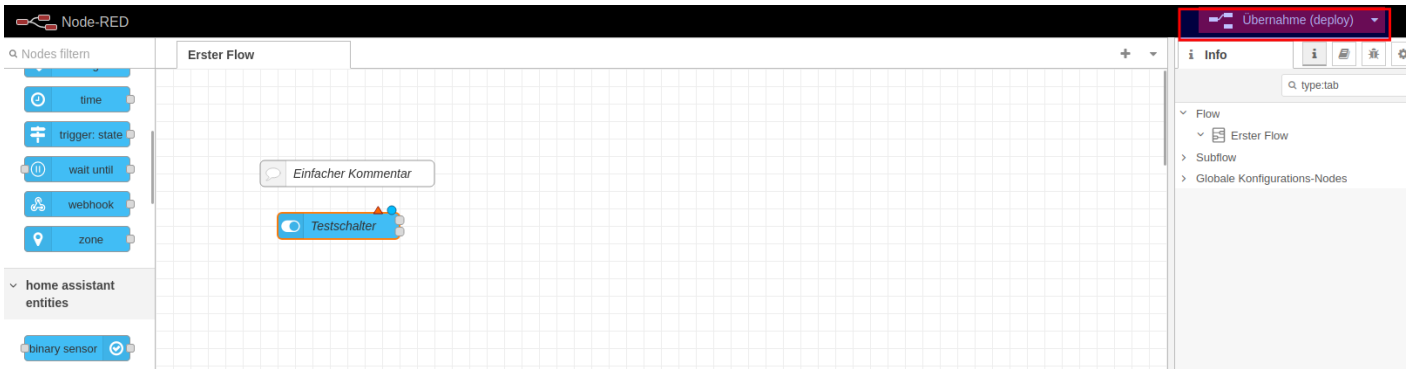
Entity picture

Device class

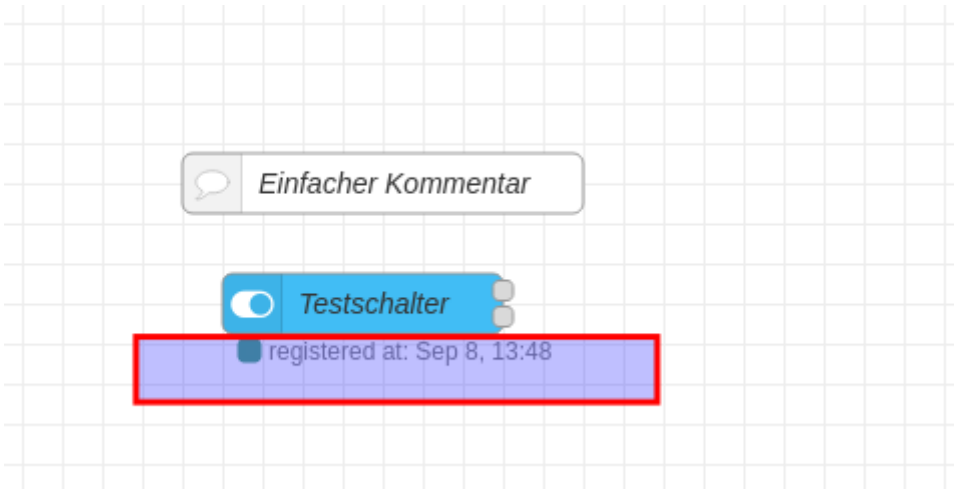
switch

Show debug information

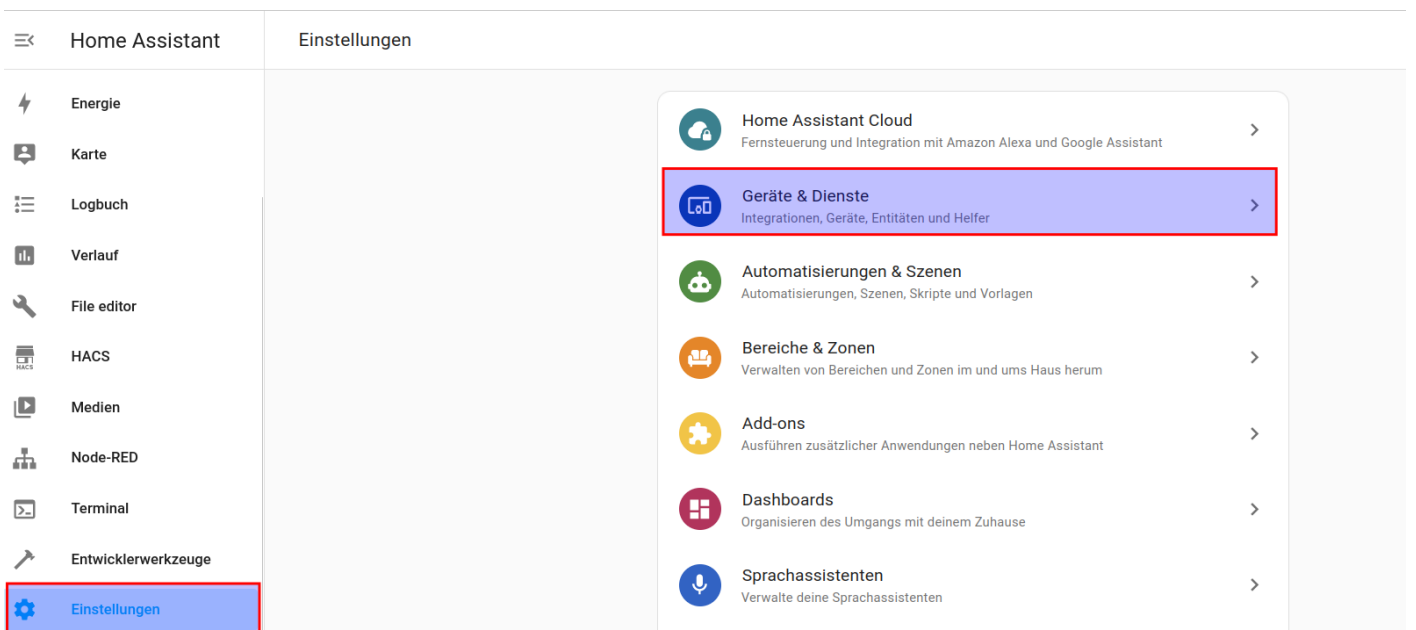
Nun noch Auf Fertig klicken.
Danach auf deploy klicken



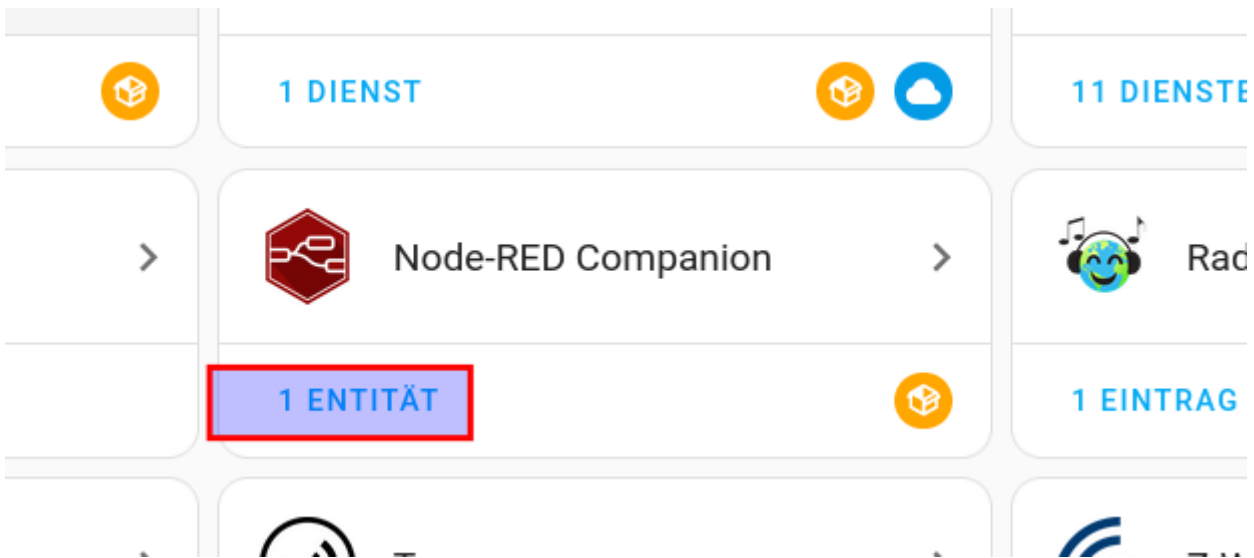
Nun haben wir einen registrierten Schalter



Um zu sehen welche Entitäten von Node-RED Companion es gibt, gehen wir unter Einstellungen -> Geräte und Dienste



Dort sehen wir dann bei Node Red Companion 1 Entität. Weil wir haben zur Zeit einen Schalter. Da klicken wir drauf.

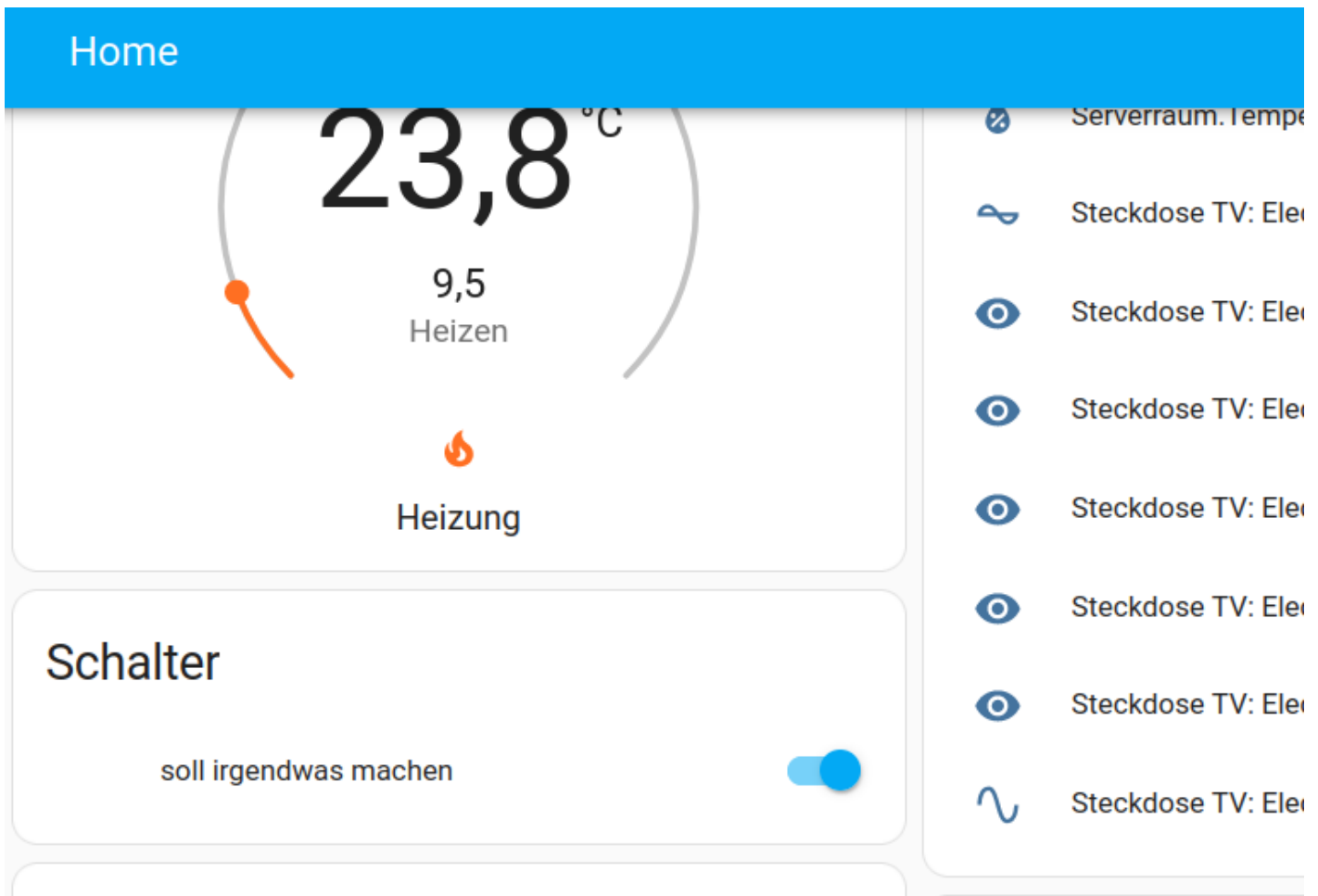


Nun sehen wir unseren Schalter. Name soll irgendwas machen. vom Typ switch.{name} name = soll irgendwas machen.
läuft. Dieses Gerät kann wie jedes anderes Gerät dann auch ein Bereich zugewiesen werden usw.

Entitäten suchen Filtern nach Integration "Node-RED Companion" [LÖSCHEN](#)

<input type="checkbox"/>	↑ Name	Entitäts-ID	Integration	Bereich	Deaktiviert durch	Statu
<input type="checkbox"/>	soll irgendwas mac...	switch.soll_irgendwas_machen	Node-RED Companion	-	-	-

Auf dem Dashboard sieht das dann so aus. Ein Virtueller Schalter.



Event State Node (trigger/sensor) Teil 1

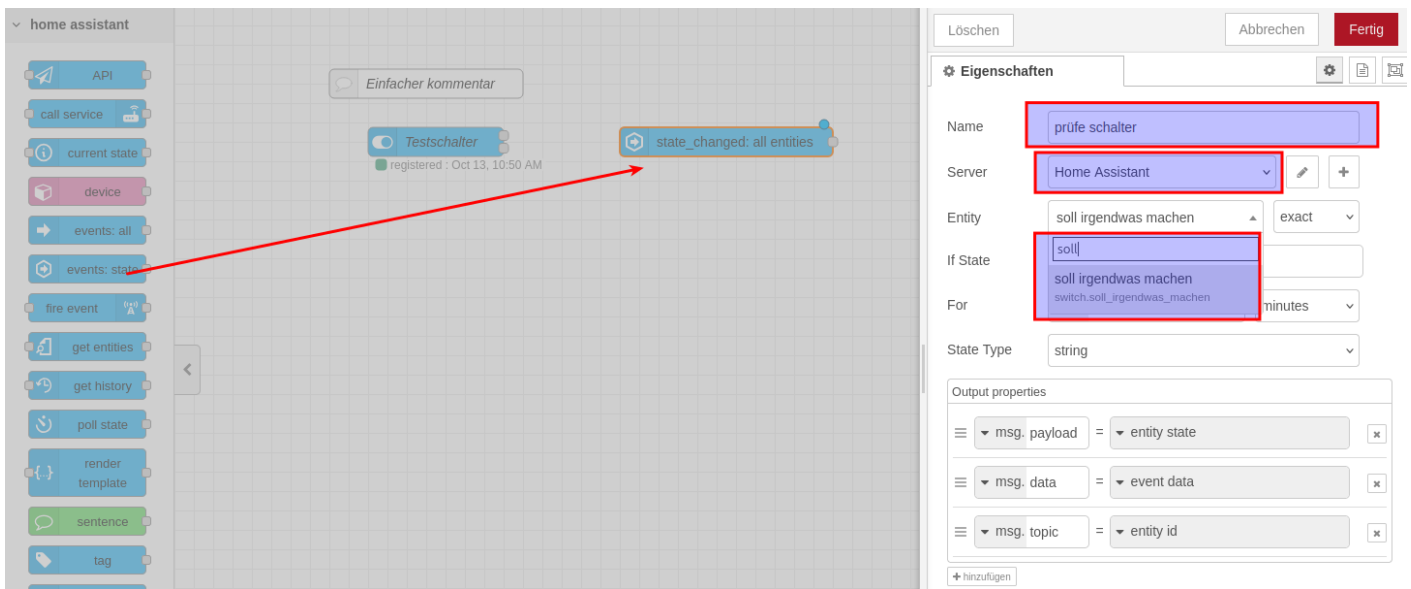
Die Event State Node fragt Geräte eigenschaften / Zustände ab, also ein trigger.

Wieder eine Node hinzufügen, hier die Event State Node einen Namen vergeben, Server auswählen (HomeAssistant)

und das Entity auswählen.

Über die suche hab ich jetzt einfach "soll" eingegeben. Und schon bekommen wir unseren Schalter soll irgendwas machen.

Wir klicken erstmal auf fertig.



Debug Node

Bevor es an die Bedingung geht.

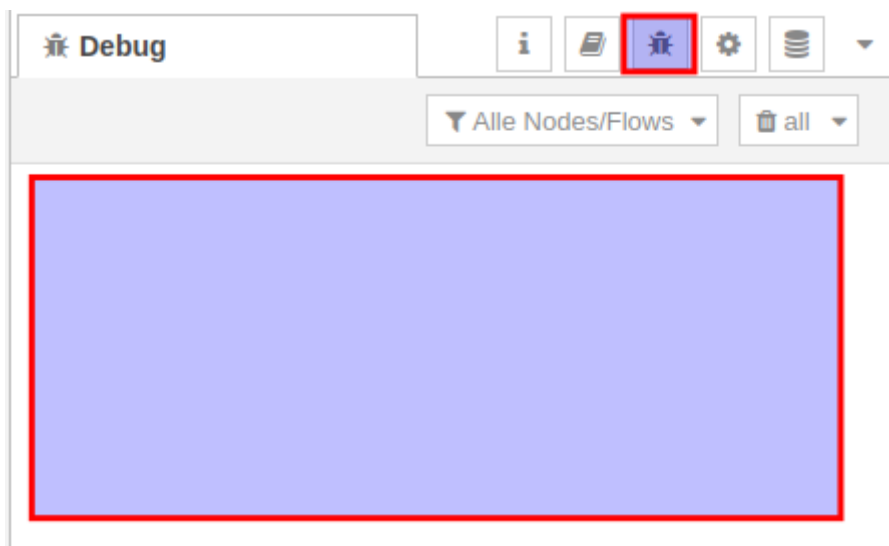
Was für einen Wert gibt unser Schalter denn zurück?

Das bekommen wir mit der Debug Node raus.

Diese Node ist besonders hilfreich um zustände zu bekommen.

In der Seitenleiste Rechts gibt es Symbol mit nem Käfer, da landen alle Infos von Debug Nodes oder Fehler von Nodes.

Über die Mülltonne mit all, lassen sich alle Nachrichten löschen, sprich das Debug Log leeren.



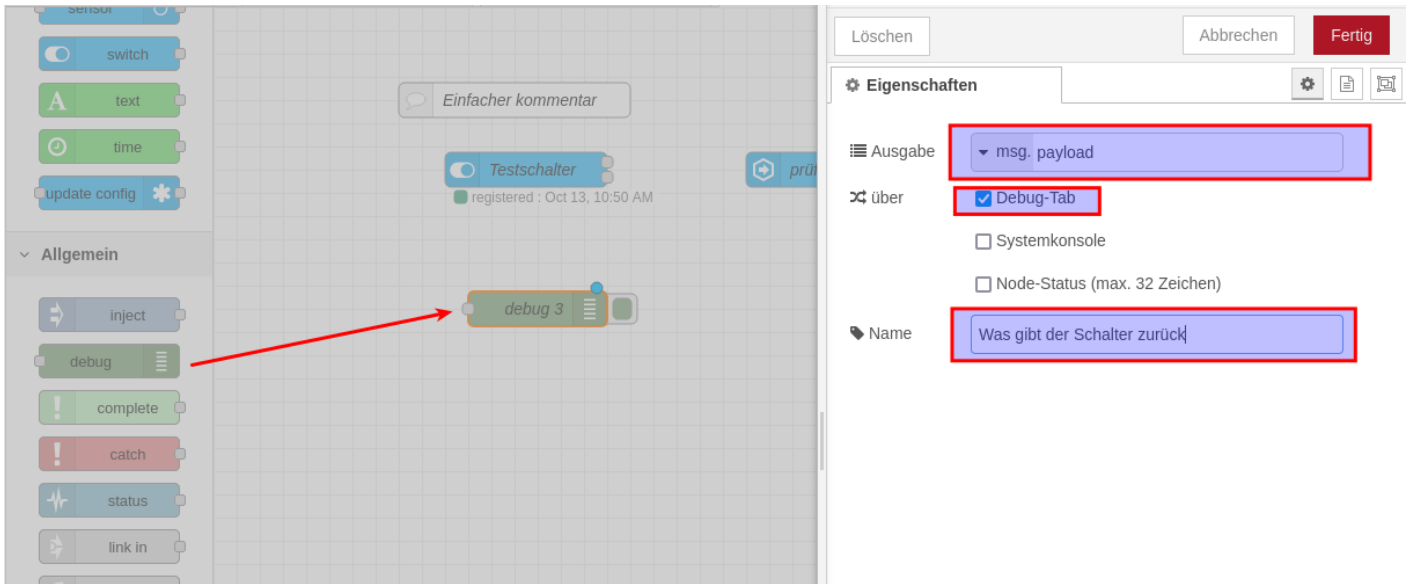
Für Fehler braucht man keine Debug Nodes.

Wir fügen eine Debug Node hinzu und geben Ihr einen Namen.

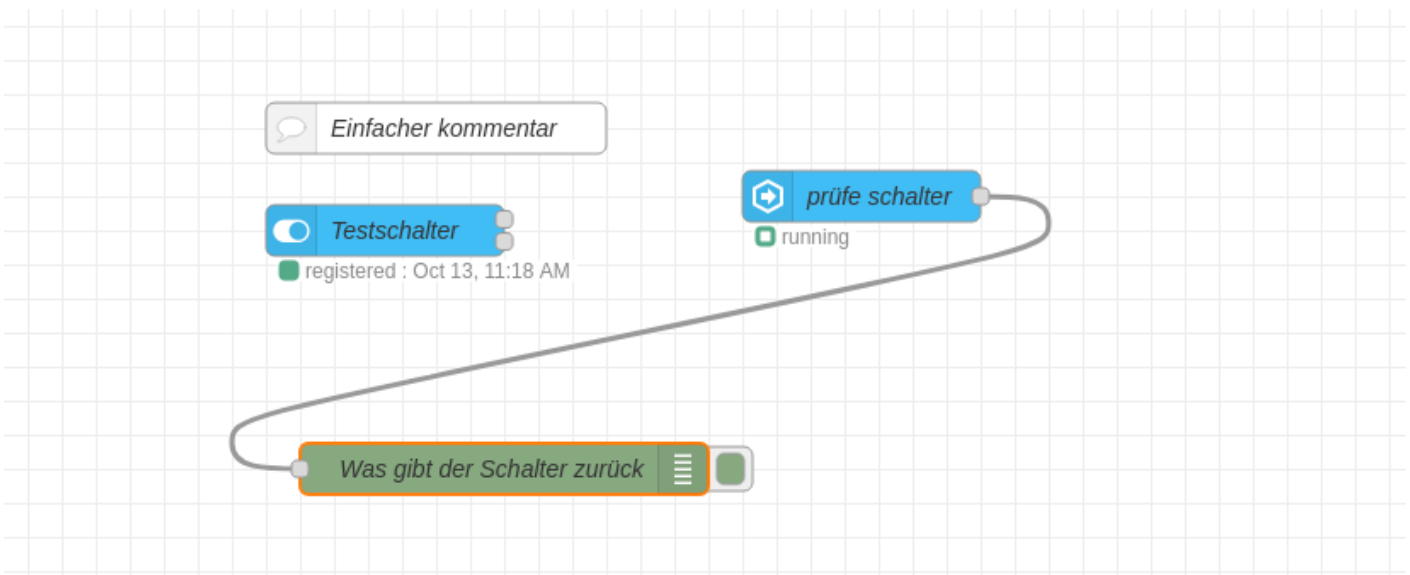
Als Ausgabe wollen wir meistens den payload, also sprich die Daten die das Gerät zurück gibt.

Den Haken bei Debug Tab rein, so das diese Dann auch im Debug Log hinzugefügt werden.

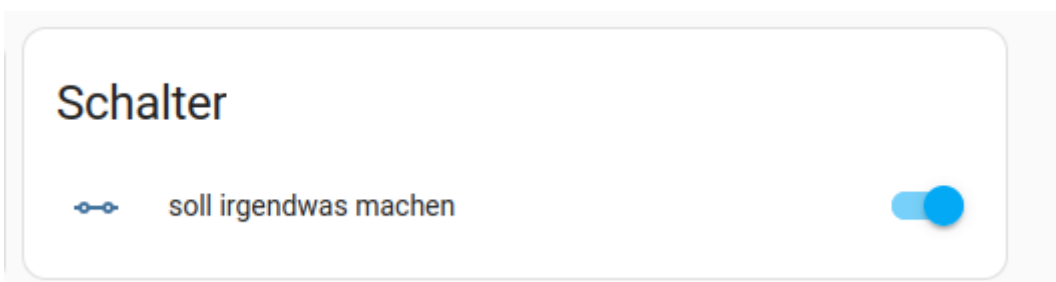
Ist der Haken raus, steht der Zustand nur unten drin



Nun den Ausgang unserer prüfe schalter node mit dem Eingang des Debug Nodes verbinden und auf Deploy klicken




Nun einen neuen Tab öffnen und unseren Schalter aus und wieder einschalten betätigen. Nach Erstellung ist der Schalter Standardmäßig eingeschaltet



Nun machen wir den Schalter aus

Schalter

 soll irgendwas machen



Und wieder ein

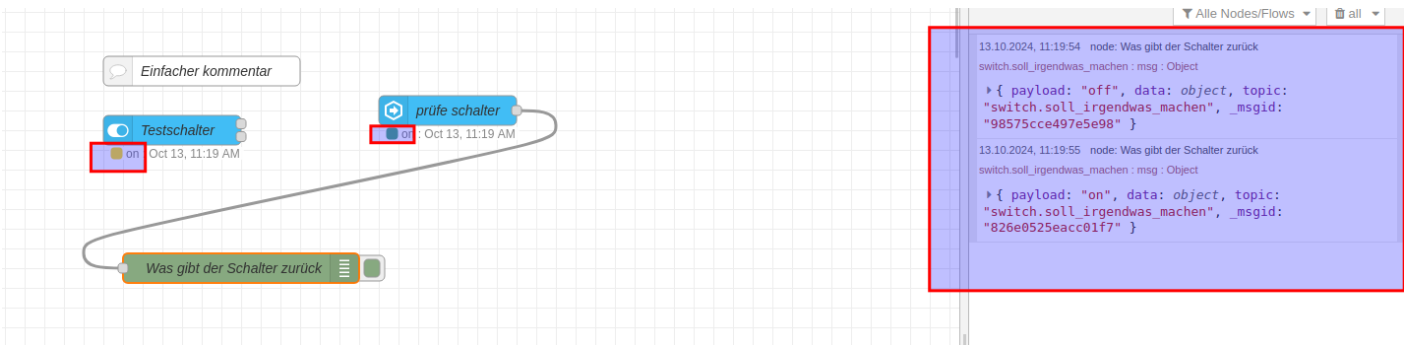
Im Debug ist nun folgendes zu sehen.

Für einfache Zustände braucht man tatsächlich keine Debug Node, aber es geht ums Prinzip, denn wie zu sehen, wir der Wert unterm Testschalter angegeben, allerdings immer nur der letzte.

Übers Debug Node wissen wir nun beide zustände. Wenn der Schalter eingeschaltet ist ist es der Wert on und wenn er ausgeschaltet ist , ist es off.

Es hätte ja genauso gut true and false sein können.

Da der Wert on off ein Text ist, wissen wir das wir einen String vergleichen müssen.



```
13.10.2024, 11:19:54 node: Was gibt der Schalter zurück
switch.soll_irgendwas_machen : msg : Object
  payload: "off", data: object, topic:
"switch.soll_irgendwas_machen", _msgid:
"98575cce497e5e98" }

13.10.2024, 11:19:55 node: Was gibt der Schalter zurück
switch.soll_irgendwas_machen : msg : Object
  payload: "on", data: object, topic:
"switch.soll_irgendwas_machen", _msgid:
"826e0525eacc01f7" }
```

Event State Node (trigger/sensor) Teil 2

Nun können wir wieder unseren Service anklicken und die Bedingung eintragen. Ist Text und on.

Node 'events: state' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name prüfe schalter

Server Home Assistant

Entity soll irgendwas machen exact

If State is **a_z** on

For 0 minutes

State Type string

Output properties

- msg. payload = entity state
- msg. data = event data
- msg. topic = entity id

+ hinzufügen

Unsere Node hat zwei Ausgänge.

Der Erste wenn die Bedingung wahr ist und die zweite Unwahr.

So können wir zwei Zustände erfassen. Ist der Schalter an, mach das, ist der aus mach das.

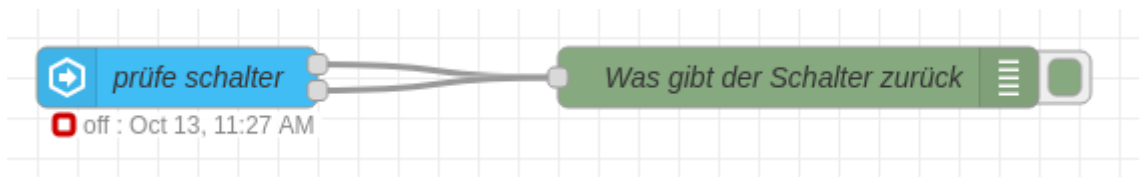
Nachdem wir die Bedingung eingetragen haben, bekommen wir im Debug auch nur noch on.

Denn vorher war die Node ja immer true und der Payload wurde für on wie auch off an dem ersten Ausgang übertragen. Wollen wir im Debug Node auch jetzt noch beide zustände müssen wir den False Ausgang auch noch ans Debug Modul packen.

```
13.10.2024, 11:27:17 node: Was gibt der Schalter zurück
switch.soll_irgendwas_machen : msg : Object
▶ { payload: "on", data: object, topic:
"switch.soll_irgendwas_machen", _msgid:
"ae672065c8630630" }

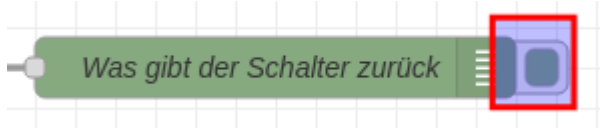
13.10.2024, 11:27:17 node: Was gibt der Schalter zurück
switch.soll_irgendwas_machen : msg : Object
▶ { payload: "on", data: object, topic:
"switch.soll_irgendwas_machen", _msgid:
"0260fe2fa281643f" }
```

Beide Ausgänge aufs Debug Modul.

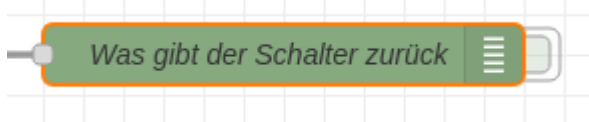


Jetzt wäre ein Kommenta sinnvoll und das Debug modul kann gelöscht werden, wenn man es nicht braucht, oder deaktiviert, damit das Log nicht vollgemüllt wird.

Deaktivieren



Nun ist es deaktiviert. Deployen nicht vergessen.



Ich lösche das Debug Objekt aber und ändere unser Kommentarfeld was jetzt auch einen sinn macht

Node 'comment' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Werte des Schalters: on/off Type: string

h1 h2 h3 B I </> [List Icons]

```
1 Der Schalter gibt als payload = on zurück wenn er eingeschaltet ist
2 Der Schalter gibt als payload = off zurück wenn er ausgeschaltet ist
3
4 Somit muss in der Abfrage ein String abgefragt werden mikt diesen Werten on oder off
5
```

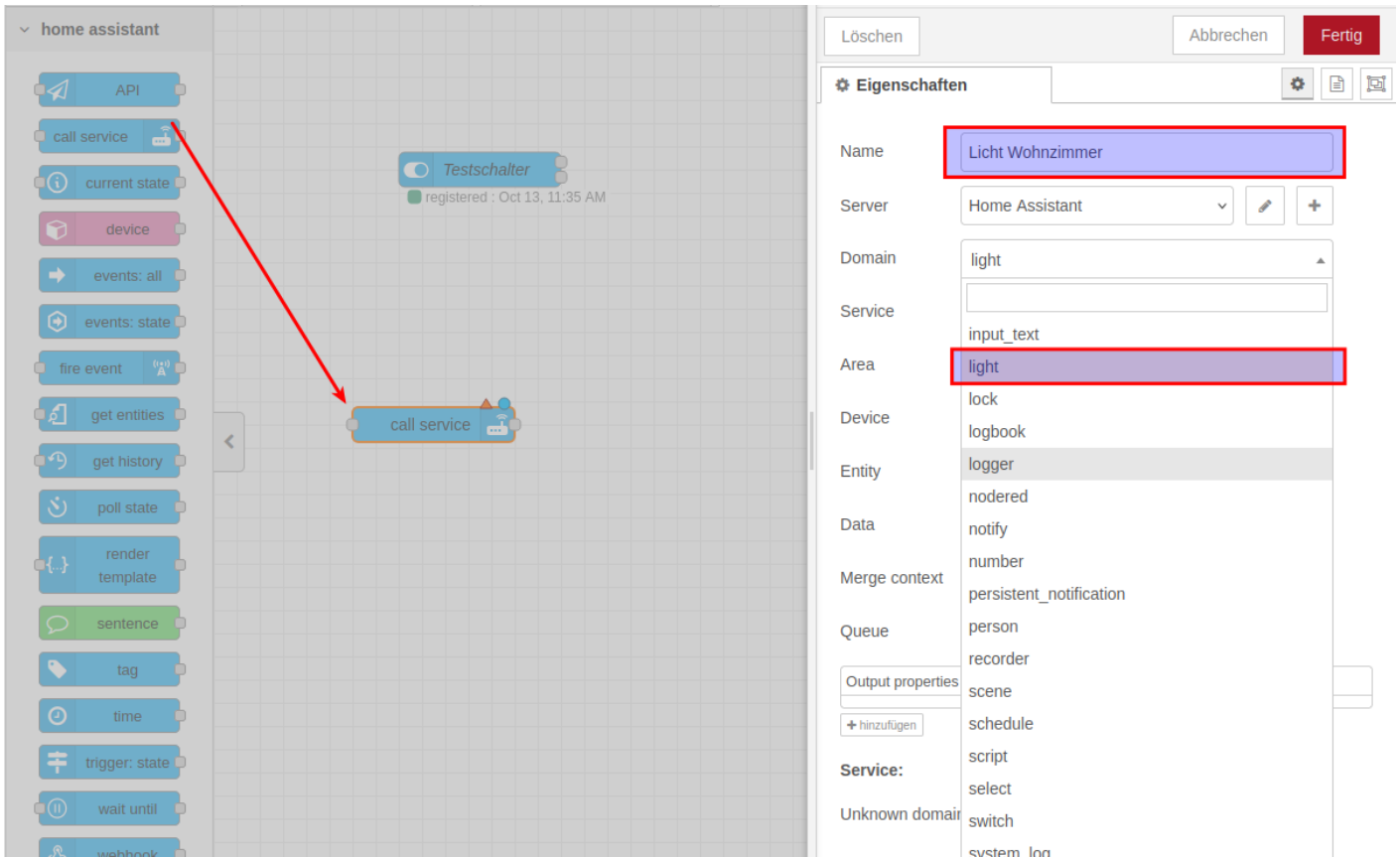
Ich schreibe oben in den Titel schon die Werte rein. Für die Vollbeschreibung dann doppelklick auf das Kommentar Node



Nun können wir endlich unsere Aktion hinzufügen.
Das machen wir mit der call service node

Call Service Node

Wir fügen ein call service node hinzu vergeben einen Namen und die wählen unsere Geräte Klasse,
Hier Licht



Bei Entity wählen wir das eigentliche Gerät.

ich gebe in die suche light an, so werden mir alle Geräte mit light aufgelistet.

ich wähle mein Sofa licht. Auch Gruppen können ausgewählt werden wie meine Gruppe light.wohnzimmerlichter

Node 'call service' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften ⚙️ 📄 🖨️

Name

Server

Domain

Service

Area

Device

Entity ligh

Data

Merge context

Queue

Output properties living room sofa light

Service:

Unknown domain

Nun wählen wir den Service aus. Da wir gesagt bei status on soll das licht an gehen und nicht togglen, wählen wir hier on

Eigenschaften ⚙️ 📄 🖨️

Name

Server ✎ +

Domain

Service

Area

Device

Entity

Nun da wir den Service ausgewählt haben, haben wir weiter unten bei Data eine Reihe von Werten die wir übergeben können angezeigt.

Die Service node hat übrigens auch die Möglichkeit Debug Informationen anzuzeigen. So braucht man keine Extra Debug Node erstellen.

Viele Nodes haben das schon inbegriffen.

Hier aber eine Liste der Werte, die dieser Service unterstützt, einige Werte werden eventuell von dem Entity nicht unterstützt.

Dann passiert halt nichts und wird nur eingeschaltet.

zum Beispiel bei einer weißen Lampe die keine Farben unterstützt, kann ich auch keine Farbe setzen:

Eigenschaften ⚙️ 📄 🖨️

Merge context

Queue

Output properties

Service: light.turn_on

Turn on one or more lights and adjust properties of the light, even when they are turned on already.

Property	Desc	Example
<code>transition</code>	Duration it takes to get to next state.	
<code>rgb_color</code>	The color in RGB format. A list of three integers between 0 and 255 representing the values of red, green, and blue.	[255, 100, 100]
<code>kelvin</code>	Color temperature in Kelvin.	
<code>brightness_pct</code>	Number indicating the percentage of full brightness, where 0 turns the light off, 1 is the minimum brightness, and 100 is the maximum brightness.	
<code>brightness_step_pct</code>	Change brightness by a percentage.	
<code>effect</code>	Light effect.	

Show debug information

Hier sind zufällig alle Parameter optional, das heißt in das Feld Data einfach zwei geschweifte Klammern.

Wichtig: Bei Data Format muss Json ausgewählt sein, nicht json Data

Eigenschaften ⚙️ 📄 🗑️

Name

Server ✎ +

Domain

Service

Area

Device

Entity ✕

Data ⋮

- JSONata Use template tags for the data field
- JSON

Merge context

Queue

Output properties

+ hinzufügen

Service: light.turn_on Load example data

aber möchte man ein Beispiel, einfach auf Load Example Data klicken.
Das find ich total Praktisch, dann braucht man nur noch die Werte ändern.

Data

Merge context

Queue

Output properties

Service: light.turn_on

Turn on one or more lights and adjust properties of the light, even when they are turned on already.

Property	Desc	Example
transition	Duration it takes to get to next state.	
rgb_color	The color in RGB format. A list of three integers between 0 and 255 representing the values of red, green, and blue.	[255, 100, 100]

Nun braucht man nur noch den Farbwert ändern als Beispiel

Device

Entity

Data

Use alternate template tags for the data field

Merge context

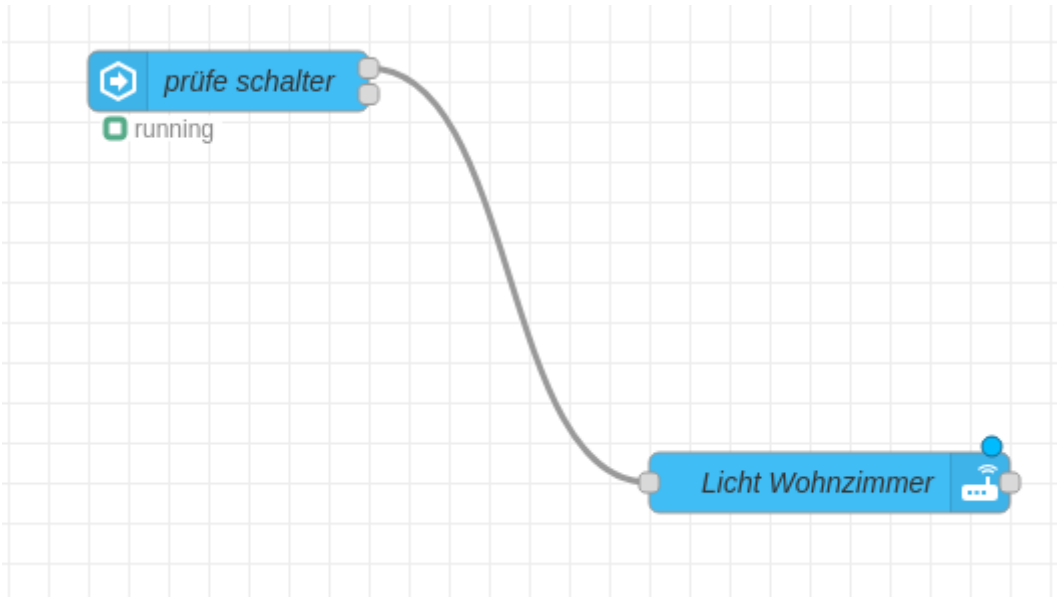
Queue

Output properties

Wir belassen es bei leeren Klammern.

Nun auf Fertig.

Jetzt können wir event state Ausgang wahr mit dem Licht verbinden und deployen das ganze.



Jetzt machen wir zwei Tabs auf. Einmal wieder zu unserem Schalter und dann den Gerätestatus vom Licht.

Wir schalten unser licht als erstes aus, auf der Geräte Seite

Steuerelemente

living room sofa light

ZUM DASHBOARD HINZUFÜGEN

Logbuch

13. Oktober 2024

living room sofa light ausgeschaltet ausgelöst durch Dienst Leuchte: Ausschalten
12:00:02 - Vor 7 Minuten -



living room sofa light eingeschaltet ausgelöst durch Dienst Leuchte: Einschalten
11:59:49 - Vor 7 Minuten -

Nun gehen wir aufs Dashbaord und schalten den schalter aus und wieder ein.

Schalter



soll irgendwas machen

Schalter

 soll irgendwas machen 

Nun wurde auch die Lampe eingeschaltet, siehe Geräte Seite

Steuerelemente

 living room sofa light 

[ZUM DASHBOARD HINZUFÜGEN](#)

Logbuch

13. Oktober 2024

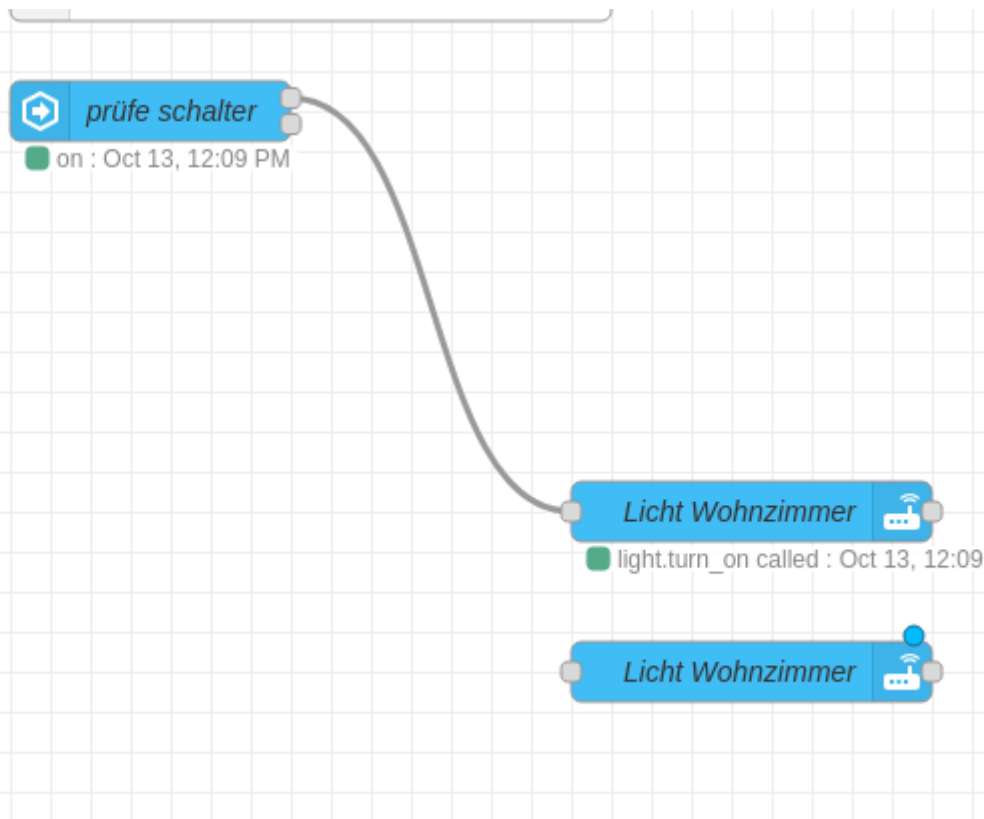
living room sofa light eingeschaltet ausgelöst durch Dienst Leuchte: Einschalten

12:09:32 - Vor 1 Minute - Supervisor

living room sofa light ausgeschaltet ausgelöst durch Dienst Leuchte: Ausschalten

12:00:02 - Vor 10 Minuten

Nun möchten wir natürlich das beim ausschalten des Schalters die Lampe auch ausgeht. Dazu kopieren wir die Node Licht Wohnzimmer. Anklicken strg+c bzw bei mac command+c und strg+v bzw command+v
Jetzt haben wir die Node zweimal



Wir klicken die erste noch verbundene doppelt an und ändern den Namen

Node 'call service' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften ⚙️ 📄 🖨️

Name

Server ✎ +

Domain

Service

Area

Device

Entity ✕

Data ⋮

Use alternate template tags for the data field

Merge context

Queue

Output properties

+ hinzufügen

Service: light.turn_on Load example data

Bei der zweiten Node ändern wir den Namen in aus und wählen den Service turn_off

Node 'call service' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Licht Wohnzimmer aus

Server Home Assistant

Domain light

Service turn_off

Area

Device turn_off

Entity

Data

Use alternate template tags for the data field

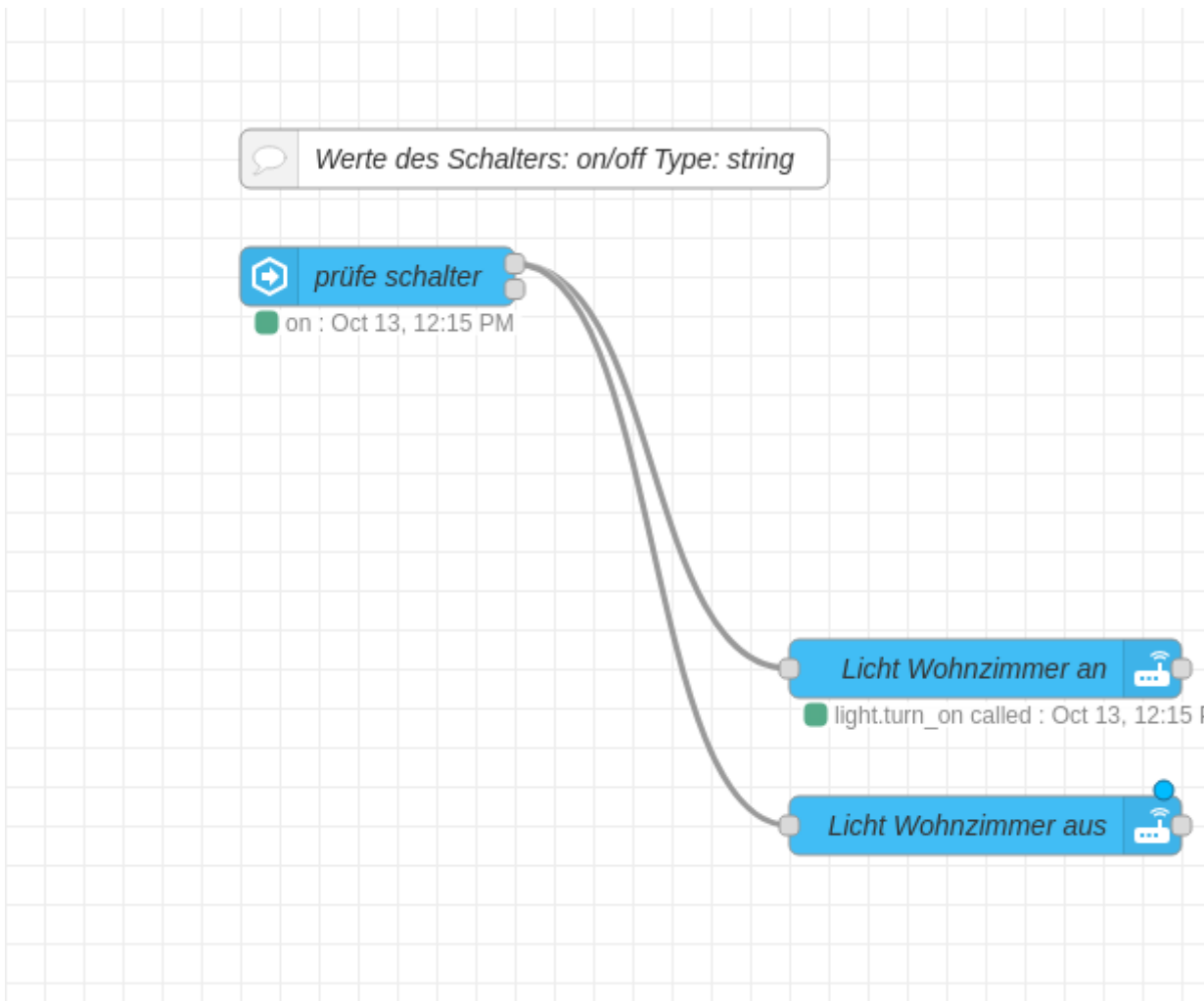
Merge context lightOptions

Queue don't queue messages

Output properties

+ hinzufügen

Nun verbinden wir die zweite node mit dem false Ausgang des event state, dann deployen

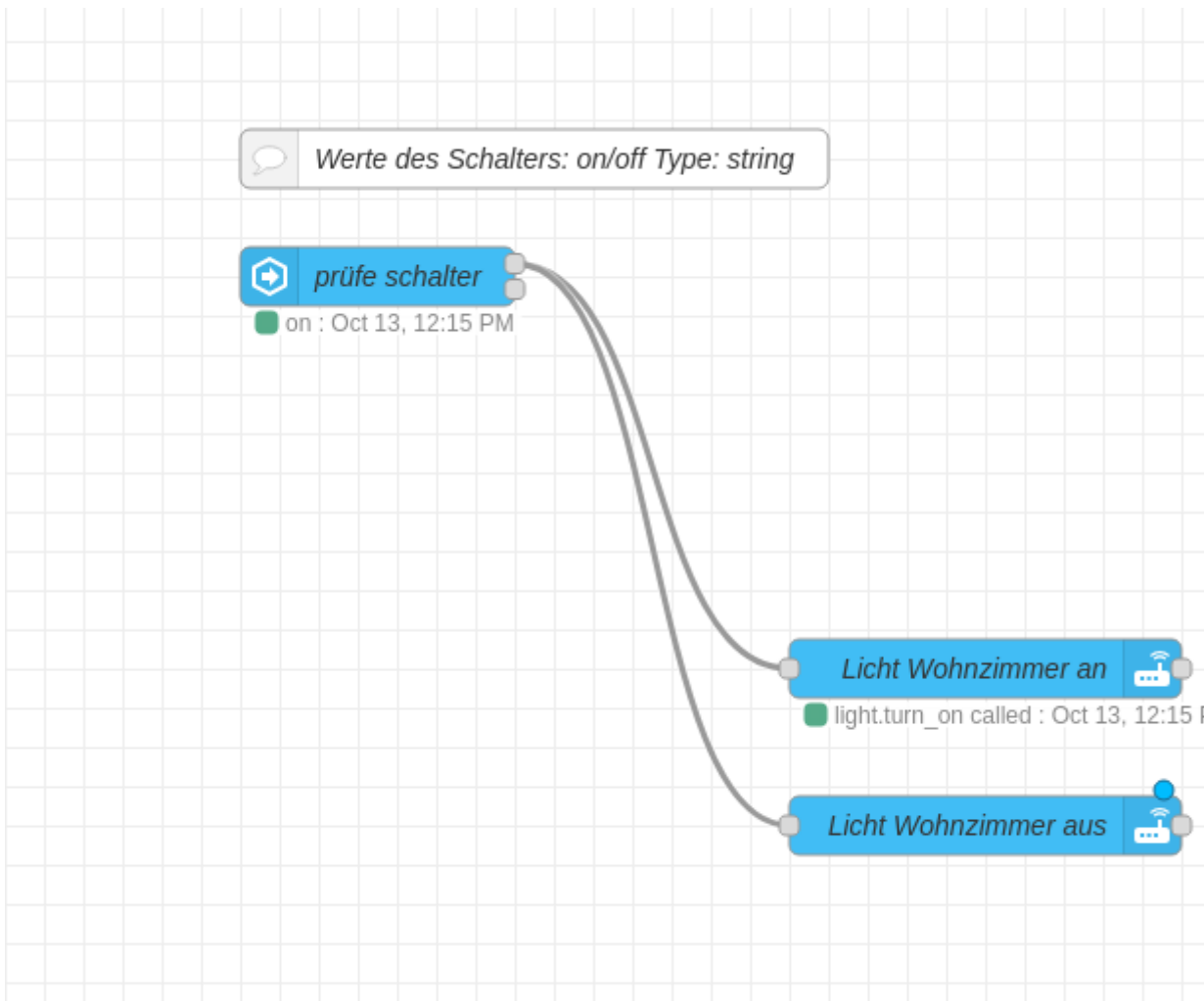


Nun kann man mittels Schalter das Licht ein und ausschalten.

Und jep Ihr habt recht, man hatte auch die Schalter Ausgänge direkt mit den Services verbinden können.

ich wollte hier aber den event state vorstellen, denn anstatt eines Schalters hätte es ja auch ein Bewegungsmelder sein können, und den können wir nur über event state abfragen.

Direkt mit Schalter sähe so aus:



Damit das Aber klappt muss im Schalter noch ein Haken gesetzt werden das der Status als Payload weitergeben werden soll

Node 'switch' bearbeiten

Löschen Abbrechen **Fertig**

Eigenschaften ⚙️ 📄 🖨️

Name

Entity config ✎️ +

Enable input

Output on state change

Dies kann dann bei Standard also belassen werden

Node 'switch' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften ⚙️ 📄 🖨️

Name

Entity config ✎ +

Enable input

Output on state change

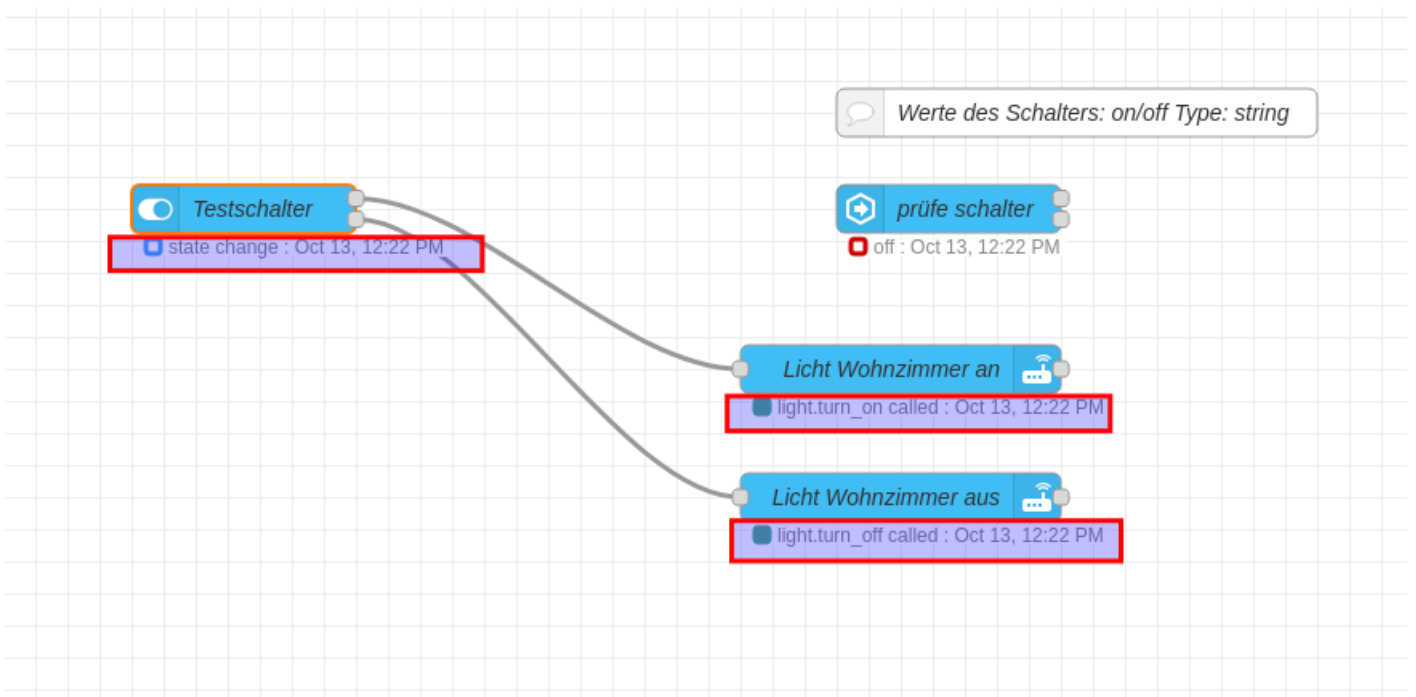
Output properties

☰ msg. outputType = a_z state change ✕

☰ msg. payload = entity state ✕

+ hinzufügen

Fertig, nun kann das Licht direkt ohne event state geschaltet werden, da der Button ja schon ein event (Ereignis auslöst)
 Wie hier auch nochmal zu sehen



Dieser Schalter dient wirklich nur dazu, wenn man gerne einen Knopf in der Oberfläche haben möchte der kein Echter Schalter ist.

Hat man so einen Schalter nur zum debuggen eingebaut, damit ich auch ohne event state meine Aktion testen kann, gibt es einen besseren Weg.

Denn jedes mal zum Bewegungsmelder zu rennen damit er ein Ereignis ausgibt kann ich mir einen Hilfsknopf direkt in Node Red bauen, ohne den Dashboard Knopf.

Über die Entwicklerwerkzeuge hab ich auch die Möglichkeit Werte zu ändern, aber da muss ich ja schon wieder ein Tab im Browser wechseln.

Siehe [Geräte Werte Temporär ändern](#)

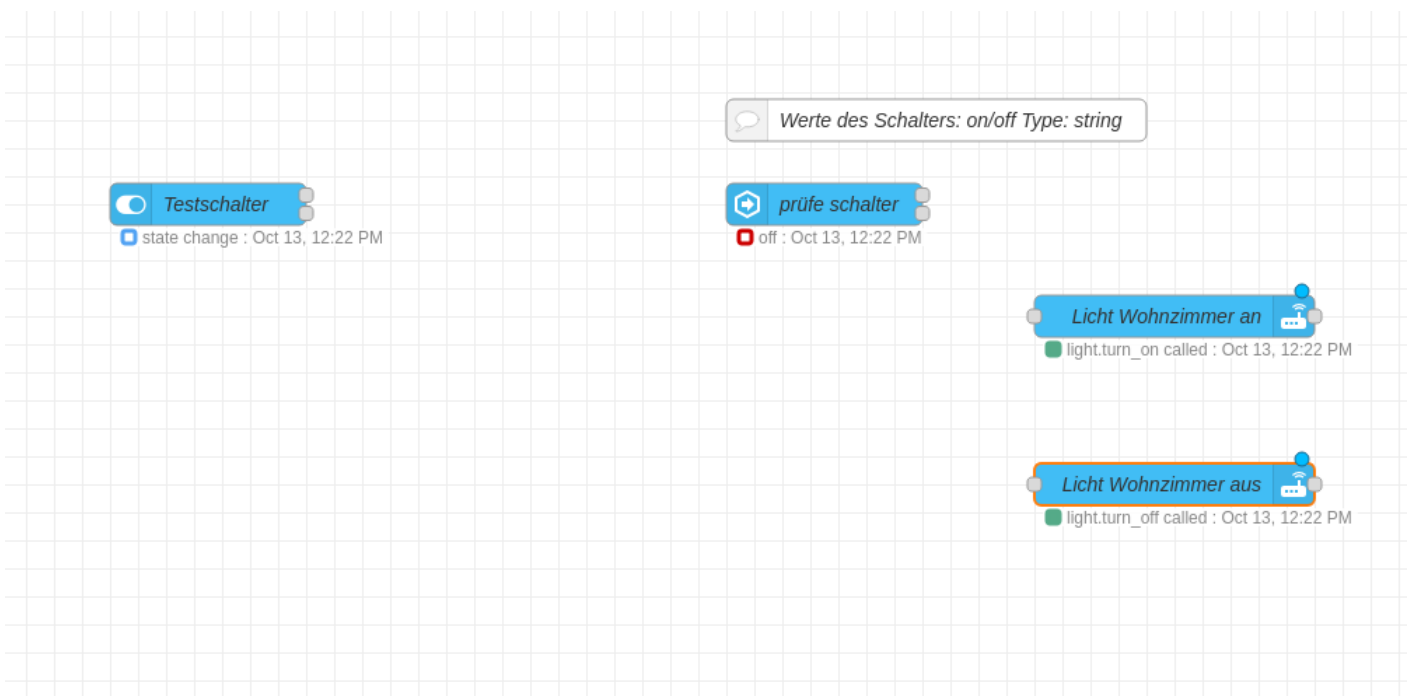
Und nun der Inject Knopf

Inject Node

Der Inejct Knopf ist ein Debug Hilfsmittel mit dem Ich ein Ereignis innerhalb Node Red triggern kann.

Entweder im Intervall oder einmal.

Wir löschen erstmal die Verbindungen zu unserem Vorherigen Button und schieben die Lichter ein wenig nach rechts



Nun fügen wir die Inject Node hinzu. Vergeben einen namen.

ALs Payload wird der Zeitstempel übergeben. Der Payload ist uns eigentlich negal, da er ja sowieso nur ein true rausgibt egal was drin steht. Das teil hat ja nur einen Zustand.

The image shows the Node-RED interface. On the left, a sidebar contains various nodes. The 'inject' node is highlighted with a red arrow. In the main workspace, a 'timestamp' node is placed on a grid. To its right, a 'Testschalter' (Test Switch) node is visible. On the right side, the configuration panel for the selected node is open. It has a title bar with 'Löschen', 'Abbrechen', and 'Fertig' buttons. Below the title bar, the 'Eigenschaften' (Properties) section is visible. The 'Name' property is set to 'Debug Knopf'. Below this, there are two configuration rows: 'msg.payload' is set to 'milliseconds since epoch' and 'msg.topic' is set to 'a-z'. At the bottom of the configuration panel, there is a checkbox for 'Einmal injizieren nach 0.1 Sekunden, danach' (Inject once after 0.1 seconds, then) and a 'Wiederholung' (Repetition) dropdown menu set to 'Keine' (None). There are also buttons for '+ hinzufügen' (Add) and 'inject now'.

Unten wäre die Möglichkeit auch eine Uhrzeit einzusetzen, dann ist es kein Debug Knopf mehr sondern wäre ein Taskplaner.

Wie geil ist das denn ;-)

Node 'inject' bearbeiten

Löschen

Abbrechen

Fertig

Eigenschaften



Name

Debug Knopf

msg. payload = milliseconds since epoch

msg. topic = a-z

+ hinzufügen

inject now

Einmal injizieren nach 0.1 Sekunden, danach

C

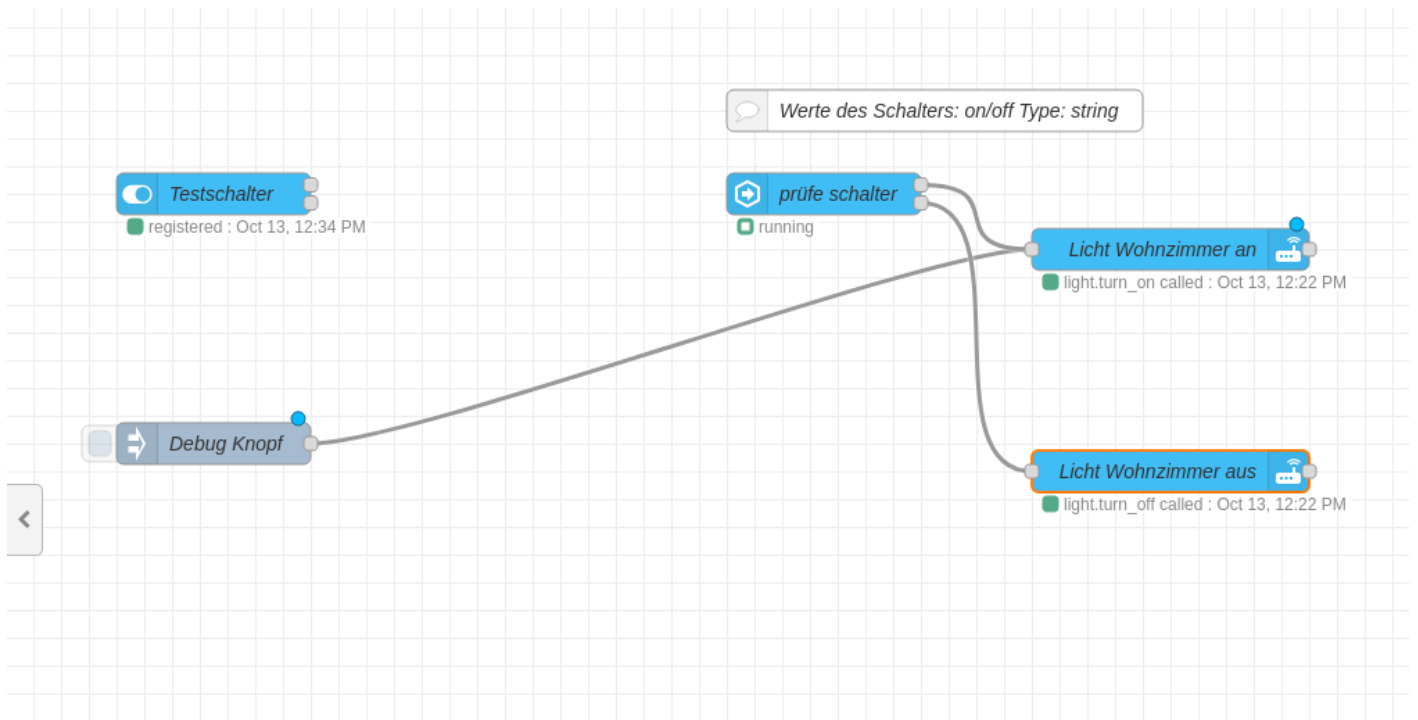
Wiederholung

Täglicher Zeitpunkt

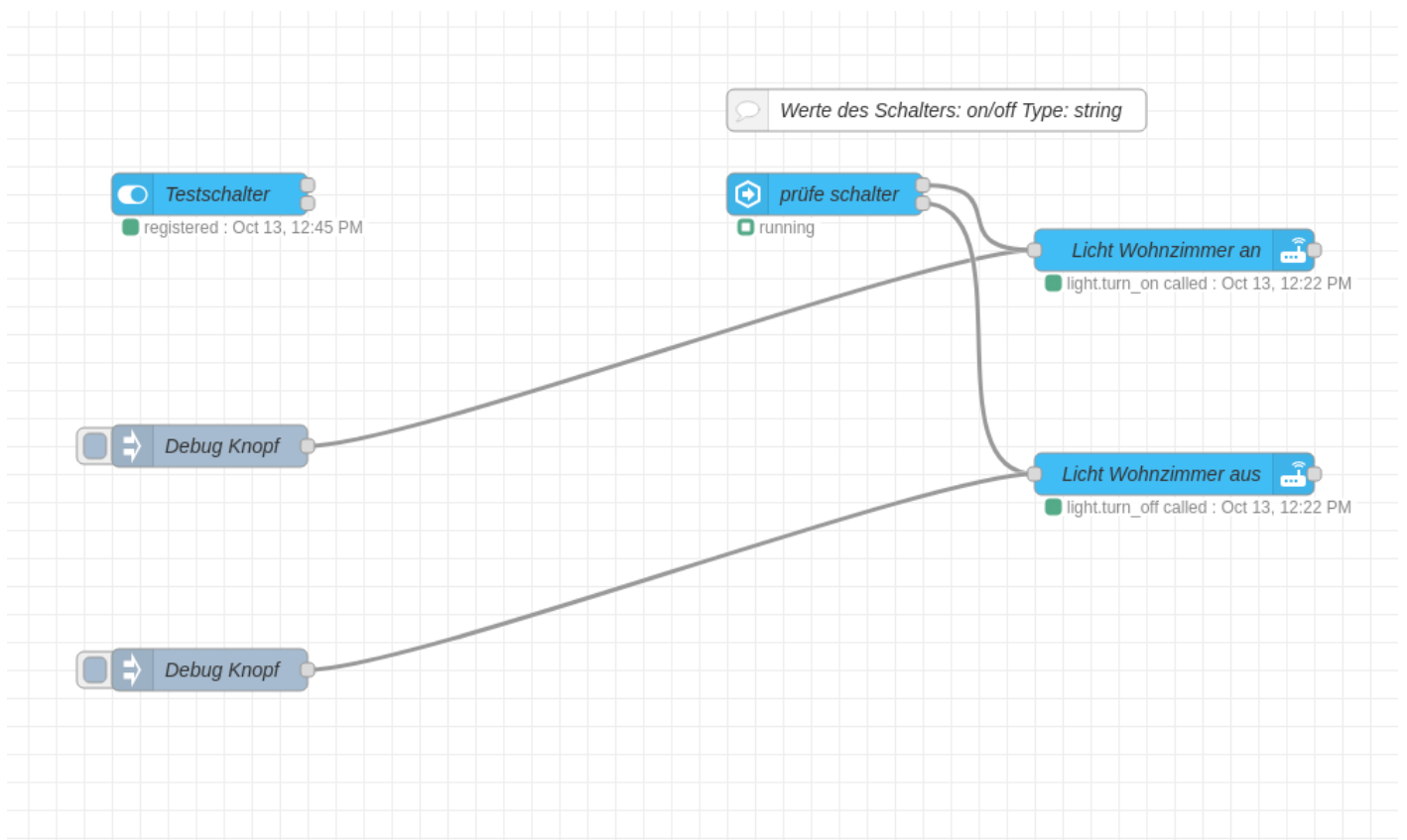
um 12:00

am Montag Dienstag Mittwoch
 Donnerstag Freitag Samstag
 Sonntag

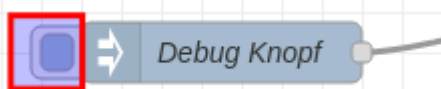
Wir nutzen ihn hier aber nur als Debug Button zum Auslösen beim drauf drücken. Diesen verbinden wir jetzt mit Lmape an. Genauso wie unseren Prüfe schalter Verbinden wir auch wieder mit Lampe an und Lampe aus. Dann sieht das ganze wieder so aus. Wird der Dashboard Schalter gedrückt löst das event state wieder aus jenach nzustand an oder aus. Wird unser Debug Knopf gedrückt wird das licht angeschaltet unabhängig vom event state



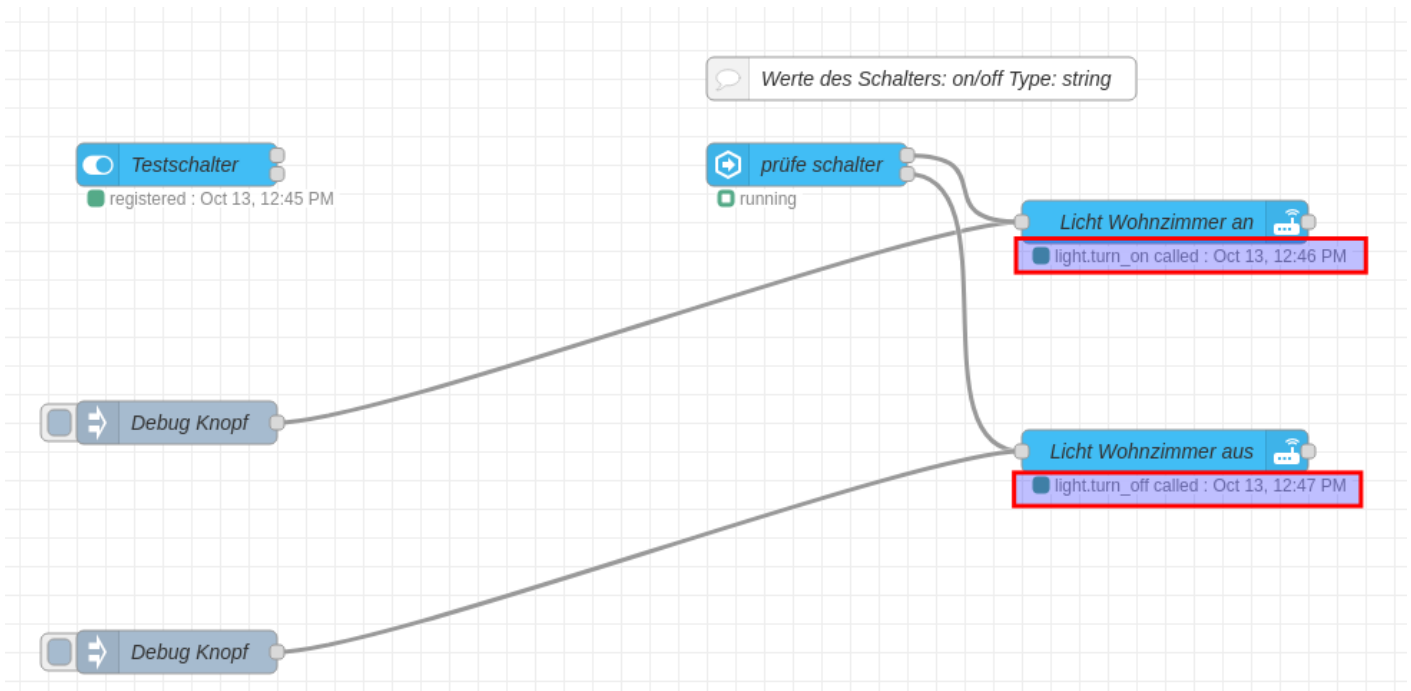
Wir kopieren den Debug Knopf noch einmal und verbinden diesen mit Wohnzimmer aus und deployen das ganze.



man löst den Debug knopf über die Schaltfläche davor aus, wie bei der Debug Node den Knopf



Anhand der Uhrzeit sieht man ich hab erst on dann off gedrückt.



Hier auch nochmal im Log der Lampe

Steuerelemente

living room sofa light

ZUM DASHBOARD HINZUFÜGEN

Logbuch

13. Oktober 2024

living room sofa light ausgeschaltet
ausgelöst durch Dienst Leuchte: Ausschalten
12:47:33 - Jetzt - Supervisor

living room sofa light eingeschaltet ausgelöst
durch Dienst Leuchte: Einschalten
12:46:13 - Vor 1 Minute - Supervisor

Anstatt den Button in der event state node zu überwachen, würde man hier einen Bewegungsmelder nehmen oder was auch immer nehmen, ist schon klar. Es geht hier ums Prinzip ;-)

Variablen: Change und Function Node :

Es gibt zwei Typen von Variablen:

1. Flow Variablen - Gelten nur in dem aktuellen Flow
2. Global Variablen - Gelten über alle Flows

Dazu ziehen wir uns eine Change Node in den Flow, setzen den Namen und wählen dann flow oder Global aus der Liste aus.

The screenshot displays the Node-RED interface. On the left, the 'Allgemein' and 'Funktion' node palettes are visible. A red arrow points from the 'change' node in the 'Funktion' palette to a 'setze msg.payload' node in the flow. The main workspace shows a flow with nodes: 'Testschalter', 'prüfe schalter', 'Debug Knopf', and 'setze msg.payload'. The right-hand panel, titled 'Node 'change' bearbeiten', shows the configuration for the selected 'change' node. The 'Name' field is set to 'Variable Flow' and is highlighted with a red box. The 'Regeln' dropdown is set to 'flow.', and the 'to the value' field is set to 'msg.'. The 'flow.' option in the dropdown is also highlighted with a red box.

Nach flow. den variable Namen und darunter der Wert

Node 'change' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Variable Flow

Regeln

Setze ▼ flow. meinevariable

to the value ▼ a_z Test 1

+ hinzufügen

Aktiviert

Uns stehen folgende Variable Typen zur Verfügung. Flow und Global machen meist keinen Sinn weil wir hier ja eine flow/global Variable definieren. Möchten wir eine bestehende Variable ändern, kommt meist noch mehr dazu als nur Inhalt Variable 1 in Variable 2 kopieren (was natürlich geht in dem to Value dann die Flowvariable ausgewählt wird mit dem Quellvariablennamen).

Für komplexere Sachen nehmen wir lieber dann die später erklärte Function Node.

Das bei Set to value ist das msg Objekt, das msg was am Eingang der Node ankommt zum Beispiel der Wert eines vorherigen verbundenen Sensors oder so.

Node 'change' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Variable Flow

Regeln

Setze ▼ flow. meinevariable

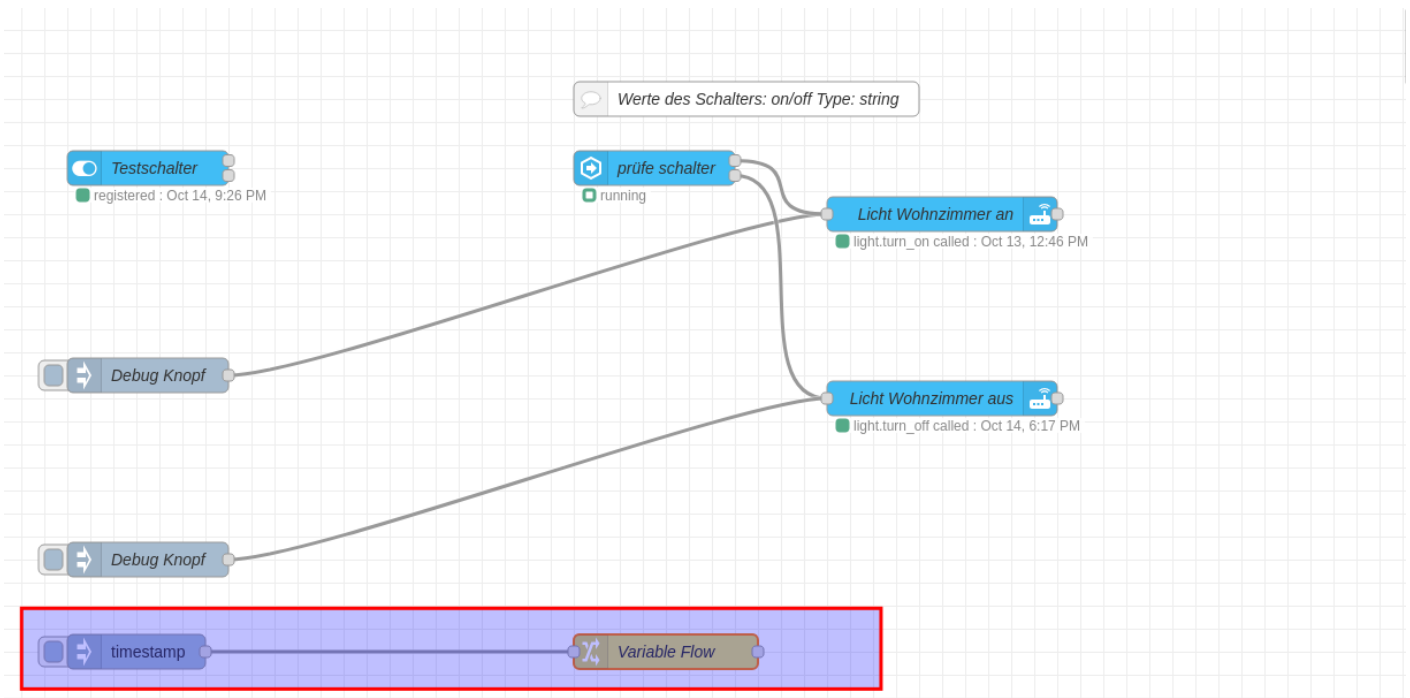
to the value ▼ ^a/_z Test 1

- msg.
- flow.
- global.
- ^a/_z string
- ⁰/₉ number
- boolean
- { } JSON
- ⁰¹/₁₀ buffer
- timestamp
- J: JSONata
- \$ Umgebungsvariable

+ hinzufügen

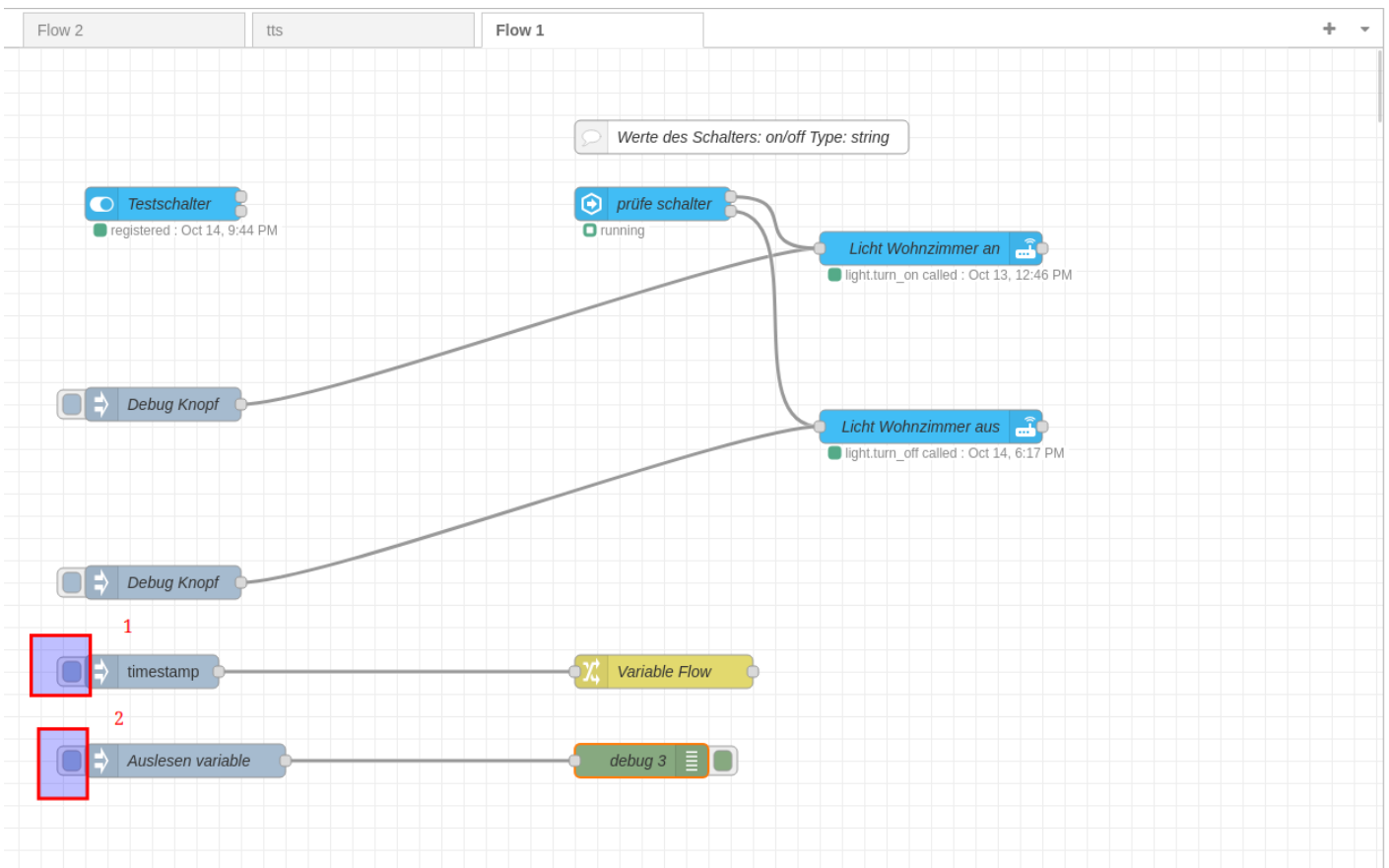
Aktiviert

Nun setzen wir uns ein Inject Knopf und verbinden die Change Node mit um diesen dann triggern.

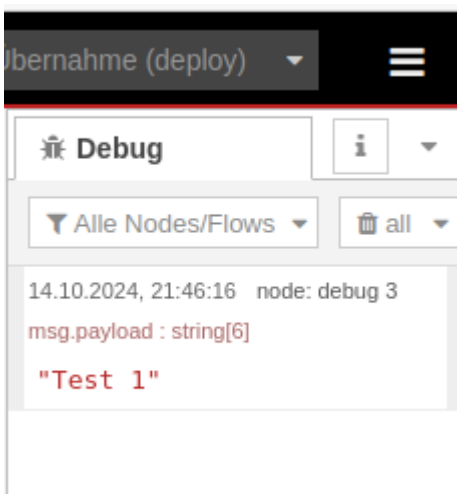


Setzen uns eine weiteres Inject und Verbinden eine Debug Node. Dort einen Namen wählen und als payload flow und unseren Variable namen.

Nun deployen wir das ganze und drücken dann auf den inject button 1 und danach inject button 2

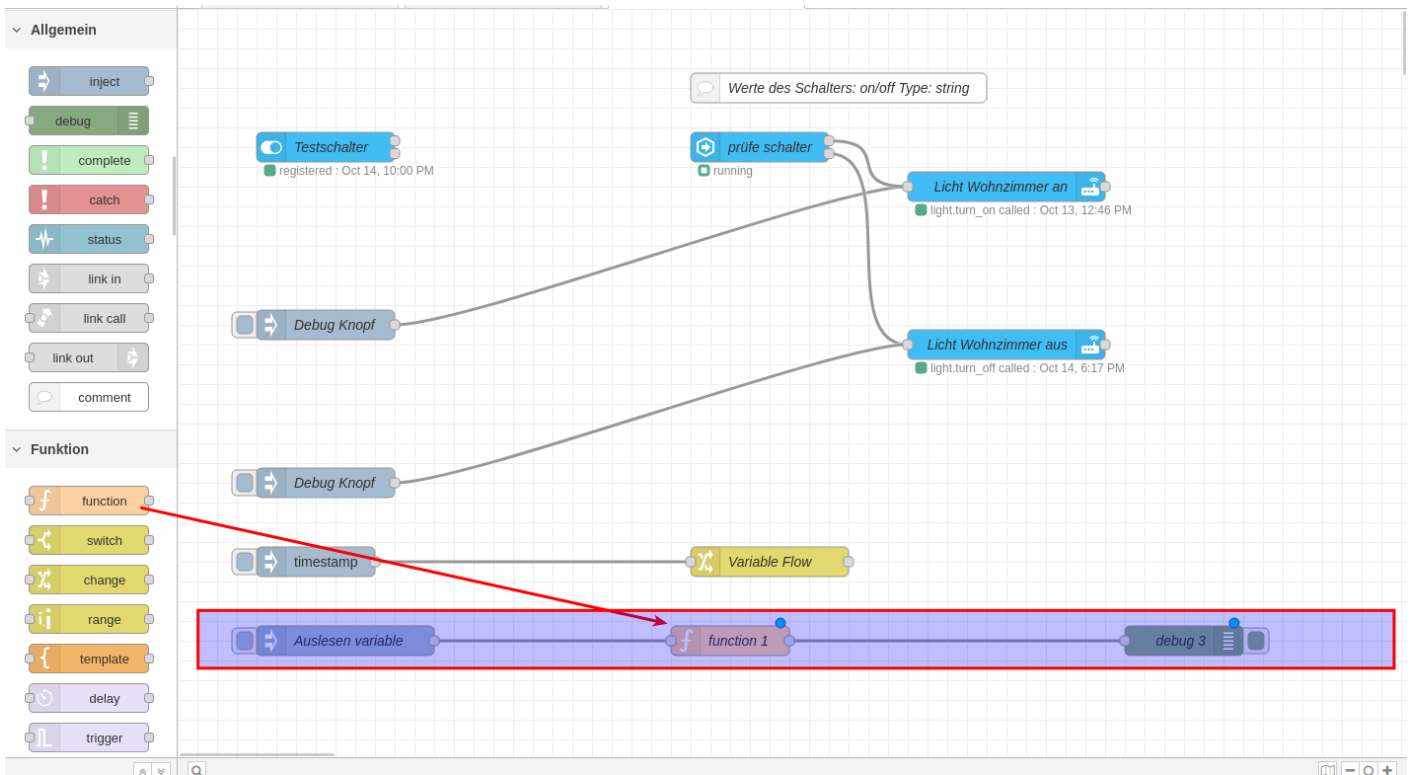


Nun sehen wir in der Debug leiste unseren Wert.



Wir können die Variable aber auch innerhalb eines Payloads nutzen also sprich als Variable in einem text oder so.

Dazu nutzen wir eine Function Node und setzen diese zwischen dem inject und dem debug



Doppelklick auf die Function und die Function anpassen

Node 'function' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name:

Setup Start Funktion Stopp

```

1 var newText = flow.get("meinevariable");
2 newText = "Dieser texte enthält folgende Variable: "+newText+" in der Mitte";
3 flow.set("meineneuevariable",newText);
4 msg.payload = newText;
5 return msg;

```

Aktiviert

Beschreibung:

```

var newText = flow.get("meinevariable"); //wir holen die flow Variable meinevariable und stecken sie in eine
loakle in dieser funktion
newText = "Dieser texte enthält folgende Variable: "+newText+" in der Mitte"; //wir überschreiben newtext mit

```

einem text und in der mitte die ausgelesene Variable die noch in newText steckt.

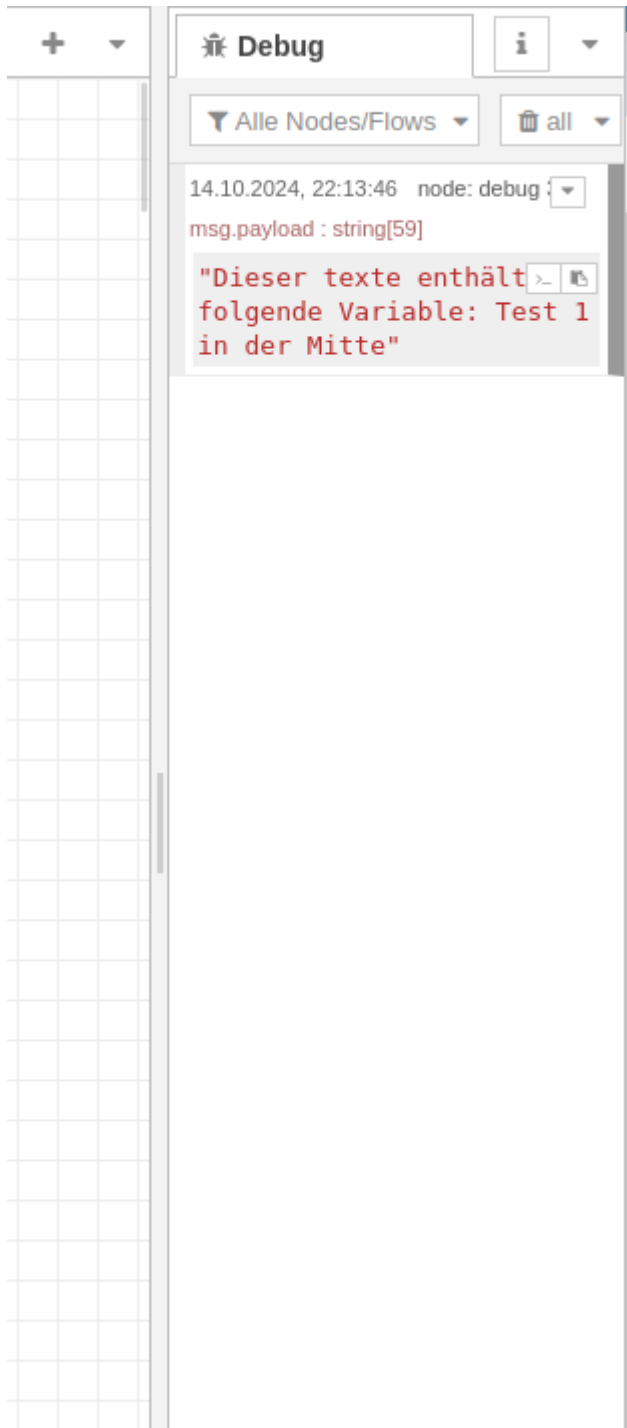
```
//newtext ist jetzt der neue text
```

```
flow.set("meineneuevariable",newText); //wir legen eine neue flow variable an, falls man darauf später zurgeifen möchte und nicht gleich in der nächsten node
```

```
msg.payload = newText; //gleichzeitig geben wir als payload den neuen text aus, falls man gleich schon mit dem payload in der nächsten node weiterarbeiten möchte, man kann aber auch die neue flow variable nehmen
```

```
return msg; //die message zurückgeben
```

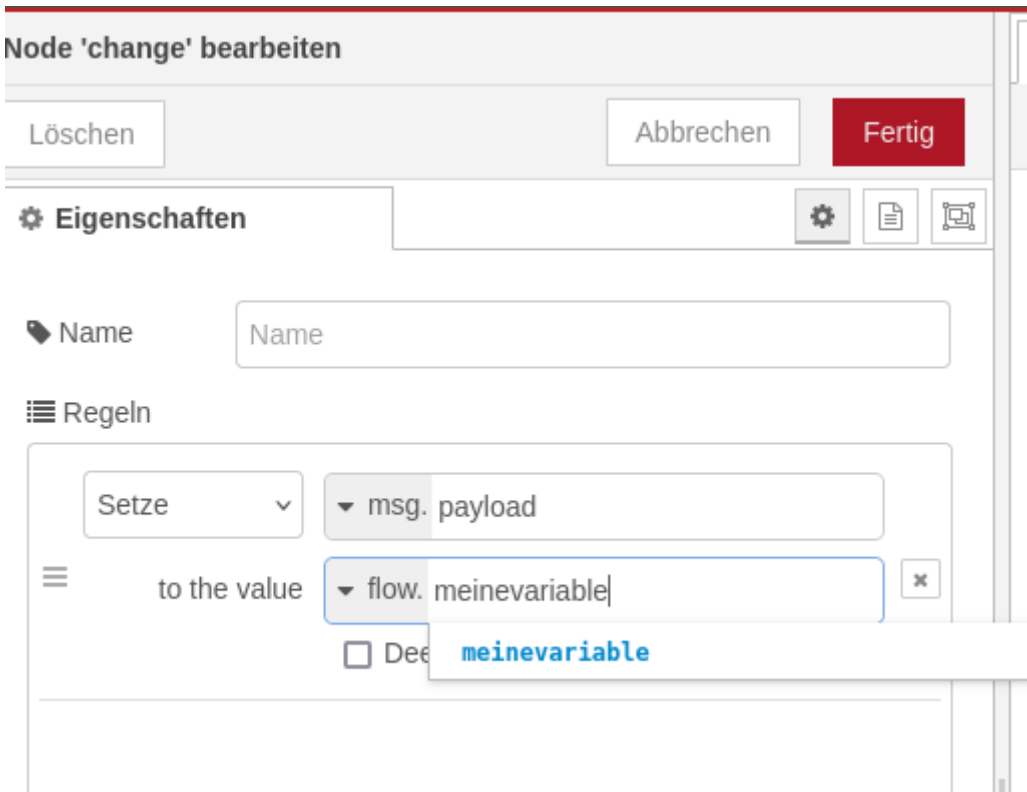
Da wir als Payload auch unseren neuen Text ausgegeben haben, ist der auch sofort im Debug Node sichtbar



The screenshot shows the 'Debug' node interface in a flow editor. The node is titled 'Debug' and has a dropdown menu set to 'Alle Nodes/Flows'. Below the title, there is a timestamp '14.10.2024, 22:13:46' and the text 'node: debug'. The main content area displays the message payload: 'msg.payload : string[59]' followed by a red text box containing the message: '"Dieser texte enthält folgende Variable: Test 1 in der Mitte"'. The interface includes a grid on the left and various control buttons like '+', 'i', and 'all'.

Die flow.meineneue Variable können wir auch in einer weiteren function, weiterverarbeiten oder mit einer change node die msg.payload einfach auf den Wert des flows/global setzten Beispiel Eigenschaften einer change node.

Der Name muss natürlich auch ausgefüllt werden, geht ums Prinzip, die msg payload auf den Inhalt eines flows/global zu setzten:



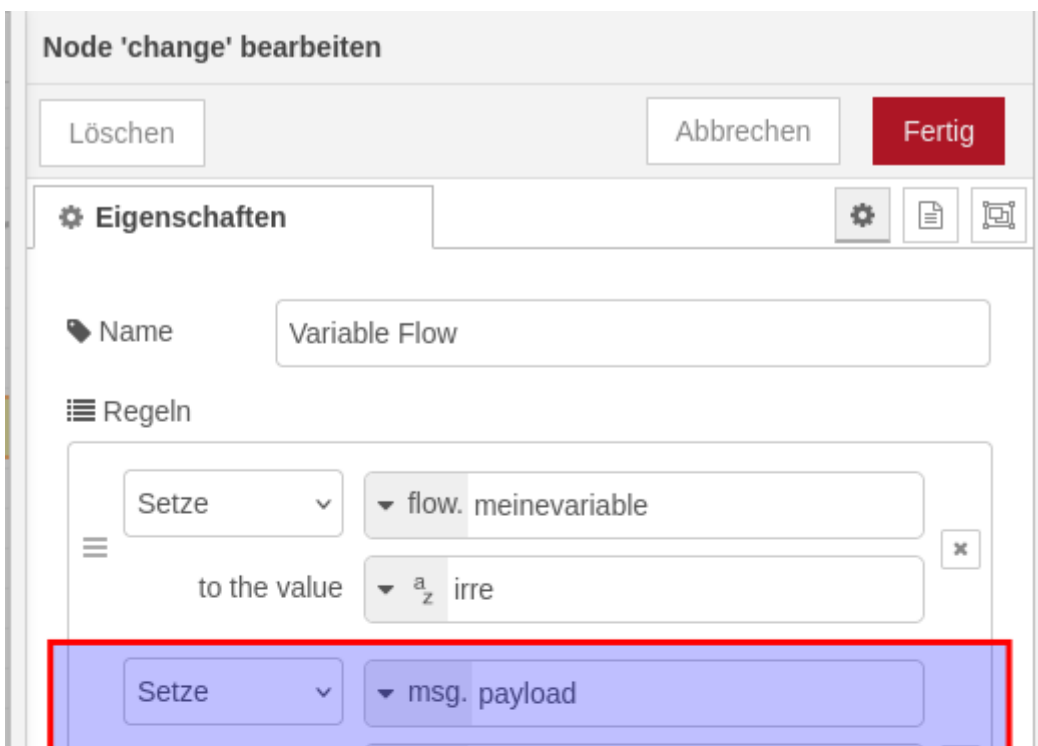
Eine Function macht dann sinn, wenn man einer Variable Werte hinzufügen/ rechnen oder was was ich mach möchte oder gleich mehrere dinge auf einmal.

Wie die Variable komplex ändern und gleichzeitig als Payload ausgeben.

Will man aber tastächlich nur variable definieren und als payload ausgeben geht das auch mit change. denn mit hinzufügen kann man mehre aktionen durchführen.

Diese werden von oben nach unten abgearbeitet.

Beispiel:



Wir können aber auch anstatt eine Change Node zu nehmen auch gleich eine Function Node nehmen um dieses durch zu führen, eine Variable zu definieren und als Payload ausgeben. Allerdings finde ich zum reinen definieren ne Change Node einfacher.

Das gleiche geht natürlich dann auch mit global anstatt flow global schreiben, wenn man global variablen braucht.

JSON in eine Variable laden, so hat man ein Array in einer Variablen:

Wie man Variablen anlegt haben wir ja schon gesehen.

So speichern wir ein JSON Object in einer Variablen und lesen einzelne Elemente wieder aus.

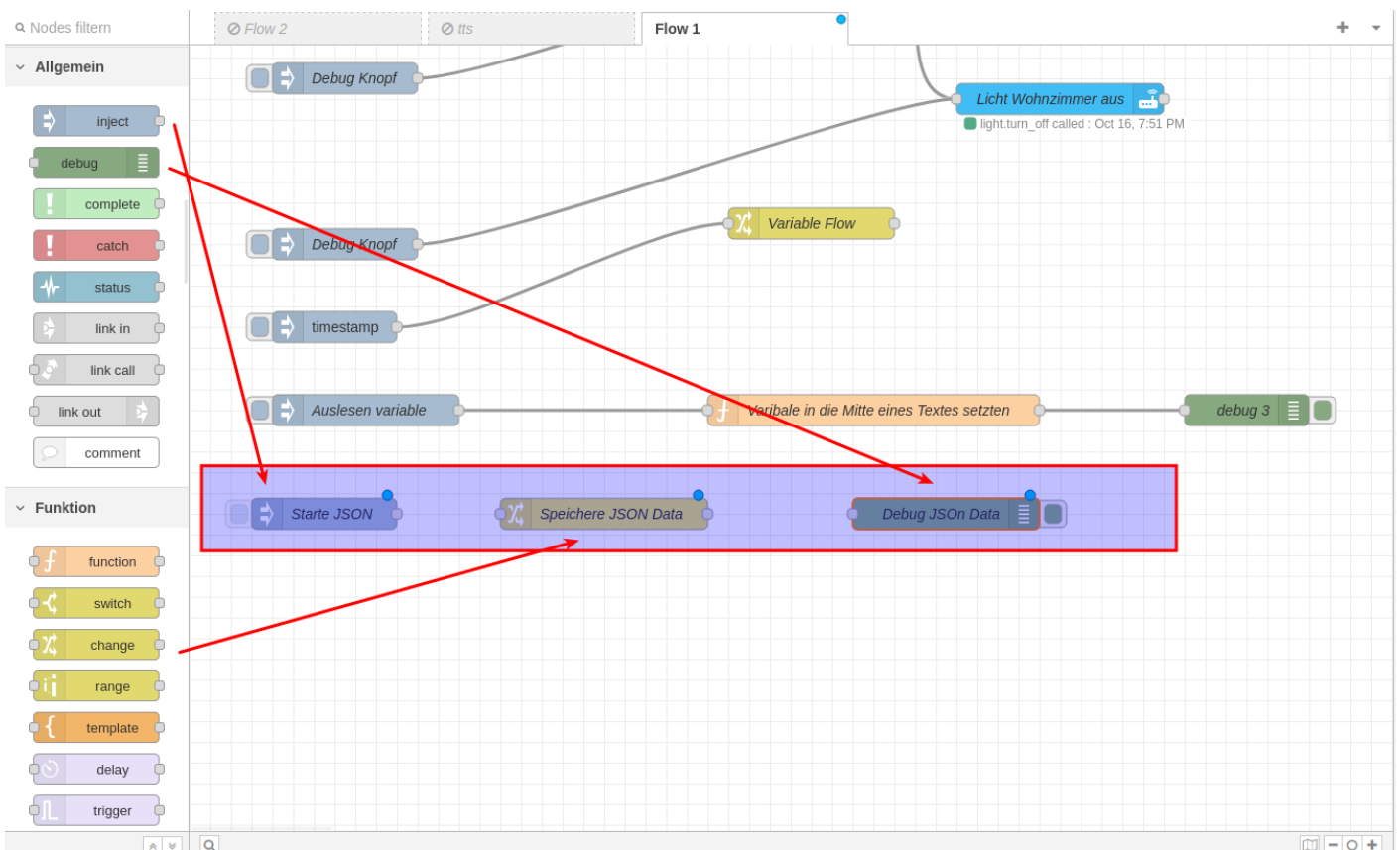
So können wir mehrere Werte in einer Variablen speichern. Und hätten damit ein Array

Hier mal ein Beispiel JSON. Hier haben wir 4 Eigenschaften: name, color, on, brightness

```
{
  "name": "licht Sofa",
  "color": "blau",
  "on": "true",
  "brightness": 100
}
```

Mit einer Change node können wir diese JSON speichern. Kommt sie von einer msg nutzen wir ebenfalls die change node.

Wir erstellen einen inject button eine change node und eine debug node



Doppelklick auf die Change Node

Dort vergeben wir einen Namen, bei setze wieder flow/global und den variablenamen und bei set to wählen wir json aus.

Node 'change' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Speichere JSON Data

Regeln

Setze

flow. meinejsonvariable

to the value {}

- msg.
- flow.
- global.
- string
- number
- boolean
- JSON
- buffer
- timestamp
- JSONata
- Umgebungsvariable

+ hinzufügen

Aktiviert

nun klicken wir auf die 3 Punkte

Node 'change' bearbeiten

Löschen

Abbrechen

Fertig

Eigenschaften



Name

Speichere JSON Data

Regeln

Setze



flow. meinejsonvariable



to the value

{ }



und fügen unser json Data Beispiel von oben ein und dann auf fertig

The screenshot shows a web-based JSON editor interface. At the top, the title bar reads "Node 'change' bearbeiten > JSON-Editor". On the right side of the title bar, there are two buttons: "Abbrechen" (Cancel) and "Fertig" (Done), with the "Fertig" button highlighted by a red box. Below the title bar, there are two tabs: "Bearbeite JSON" (Edit JSON) and "Visueller Editor" (Visual Editor). The "Bearbeite JSON" tab is active, and it contains a text area with a light blue background. The text area contains a JSON object:

```
1 {  
2   "name": "licht Sofa",  
3   "color": "blau",  
4   "on": "true",  
5   "brightness": 100  
6 }
```

 The text area is also highlighted with a red border. In the top right corner of the text area, there is a button labeled "JSON formatieren" (Format JSON). The "Visueller Editor" tab is currently inactive.

Nun sieht das ganze so aus und jetzt unten auf hinzufügen klicken

Node 'change' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Speichere JSON Data

Regeln

Setze ▼ flow. meinejsonvariable




to the value ▼ {} {"name":"licht Sofa","color":"blau"," ...

+ hinzufügen

Nun kann die msg.payload Ausgabe gesetzt werden auf die flow Variable die wir im ersten schritt defniert haben, danach auf fertig. Diese variable kann nun in der nächsten node durch setzten des msg.payload, wie auch später über den Variablenamen wieder verwendet werden.

Node 'change' bearbeiten

Löschen Abbrechen **Fertig**

Eigenschaften   


Name

Regeln

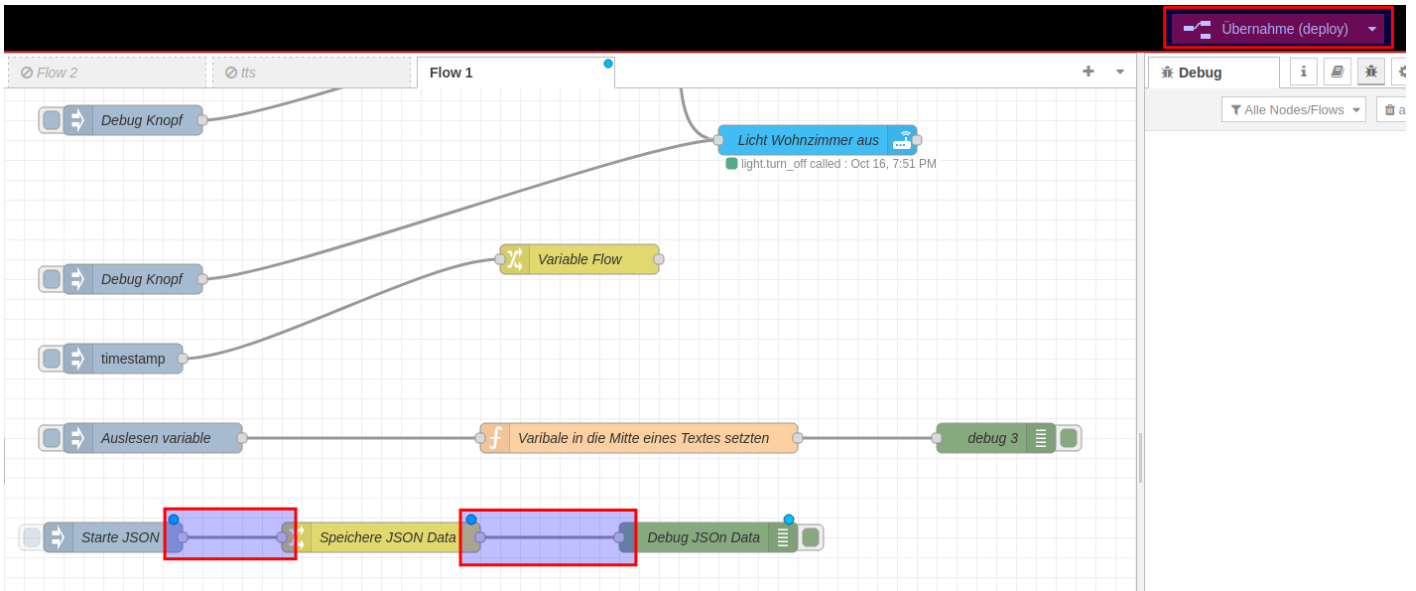
Setze

Setze Deep copy value

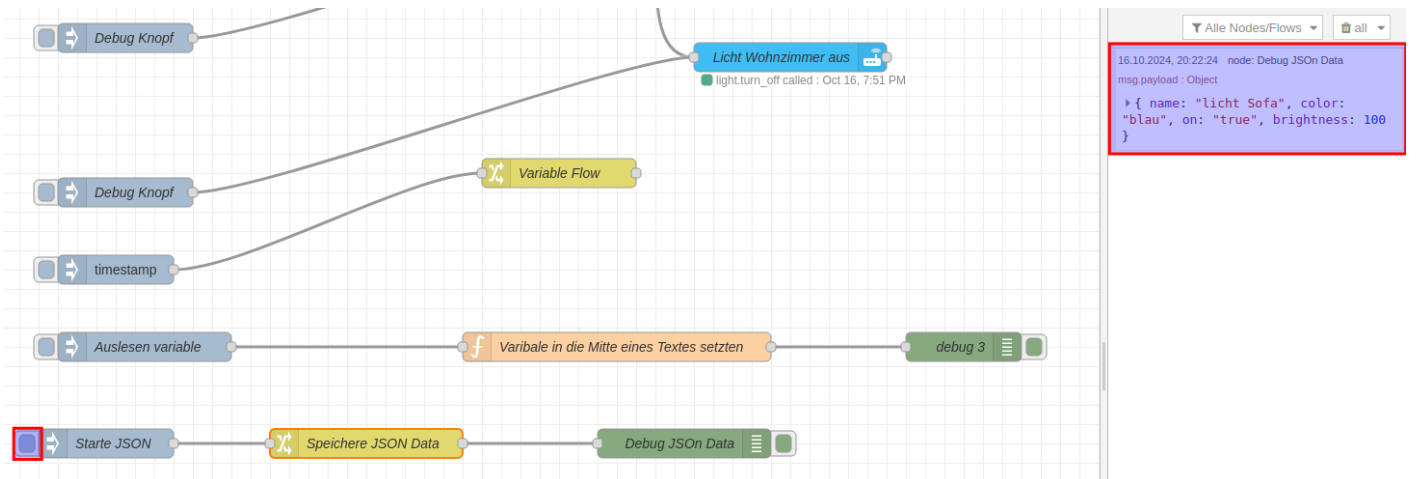
+ hinzufügen

 Aktiviert

Nun unsere Nodes Verbinden und dann deploy anklicken



Wenn wir jetzt auf den Injector klicken sehen wir im Debug Fenster unsere json Objekt



Klicken wir auf den Pfeil gibt ne bessere Übersicht.

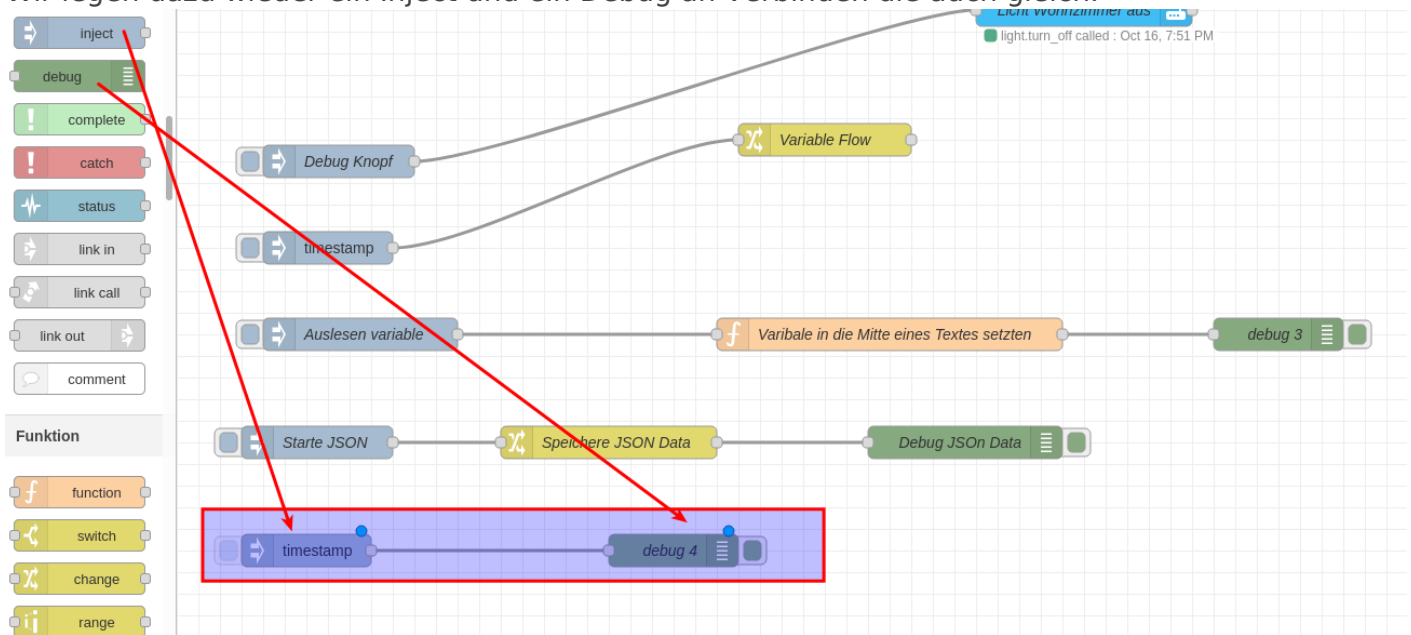
```
16.10.2024, 20:22:24 node: Debug JSON Data
msg.payload: Object
▶ { name: "licht Sofa", color:
"blau", on: "true", brightness: 100
}
```

```
16.10.2024, 20:22:24 node: Debug JSON Data
msg.payload: Object
▼ object
  name: "licht Sofa"
  color: "blau"
  on: "true"
  brightness: 100
```

Daten Zugreifen.

Da das ein JSON Object können wir dem "." auf die Objekte zugreifen..

Wir legen dazu wieder ein Inject und ein Debug an Verbinden die auch gleich.



Nun doppelklick auf die Inject node

Dort einen Namen vergeben und dann bei Flow den Variablenamen eingeben.

Node 'inject' bearbeiten

Löschen Abbrechen Fertig

Eigenschaften

Name Nur ein Element injecten

msg. payload = flow. meine
msg. topic = meinejsonvariable

+ hinzufügen inject now

Einmal injizieren nach 0.1 Sekunden, danach

Wiederholung Keine

Aktiviert

Nun kann hinter den Variablennamen ein punkt angegeben werden und wir bekommen eine auflistung der Elemente

Node 'inject' bearbeiten

Löschen

Abbrechen

Fertig

Eigenschaften



Name

Nur ein Element injecten

msg. payload = flow. meinejsonvariable. |

msg. topic = a-z

meinejsonvariable.brightness

meinejsonvariable.color

meinejsonvariable.name

meinejsonvariable.on

+ hinzufügen

inject now

Einmal injizieren nach 0.1 Sekunden, danach

Wiederholung

Keine

Wir nehmen hier mal den Namen und klicken auf fertig.

Node 'inject' bearbeiten

Löschen Abbrechen **Fertig**

Eigenschaften

Name Nur ein Element injecten

msg. payload = flow. meinejsonvariable.name|

msg. topic = a_z

+ hinzufügen inject now

Einmal injizieren nach 0.1 Sekunden, danach

Wiederholung Keine

Aktiviert

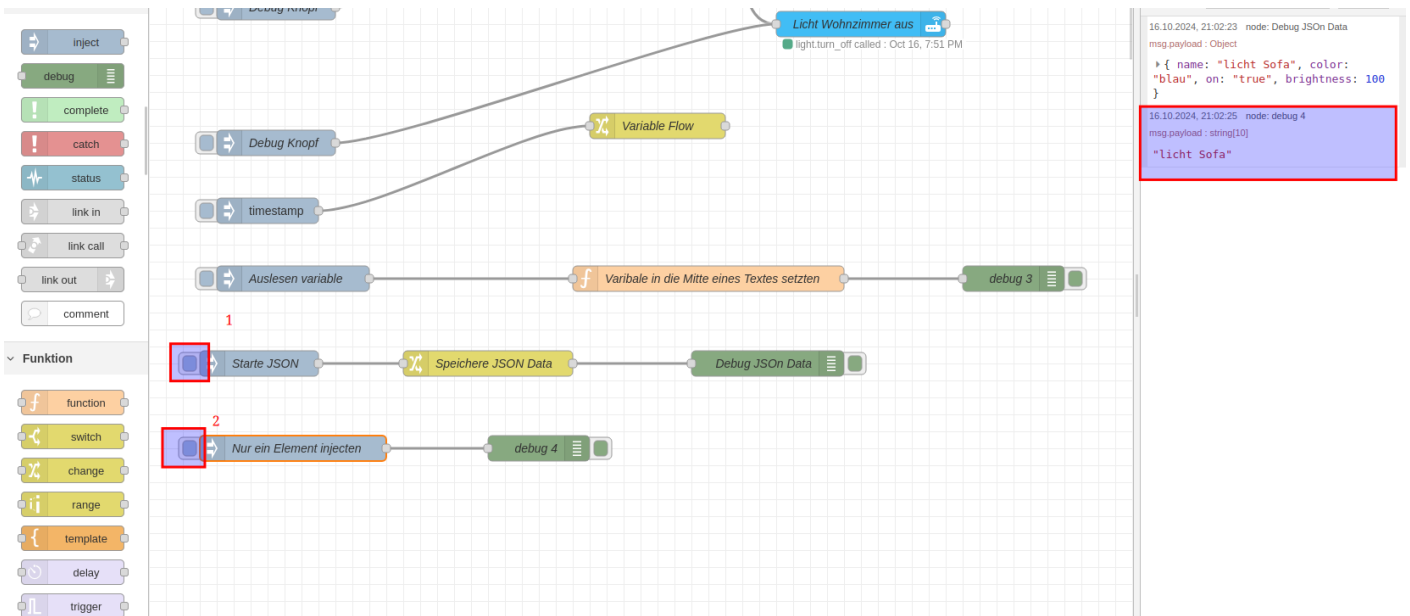
Nun deployen

Dann den ersten inject für die für die json variable erstellen.

Dann den zweiten Inject für das auslesen.

nun sehen wir den Wert der Eigenschaft Name als Payload.

Nämlichen den Namen.



Version #18

Erstellt: 8 September 2023 06:34:42 von Admin

Zuletzt aktualisiert: 16 Oktober 2024 17:03:58 von Admin