Node-RED

Node-RED ist ein Programmierwerkzeug, mit dem sich Hardwaregeräte, APIs und Online-Dienste auf neue und interessante Weise miteinander verbinden lassen.

Es bietet einen browserbasierten Editor, der es einfach macht, Abläufe unter Verwendung der breiten Palette von Knoten zu verdrahten, die mit einem einzigen Mausklick zur Laufzeit bereitgestellt werden können. (Quelle: https://nodered.org/, übersetzt mit DeepL.)

Und die Arbeit mit diesem grafischen Entwicklungswerkzeug macht wirklich Spaß!

Was erwartet Sie in diesem Beitrag?

Sie werden...

- ... die wichtigsten Eigenschaften von Node-RED kennen lernen
- ... Node-RED-Steuerungen erstellen können
- ... die notwendigen Softwarekomponenten kennen lernen und sie installieren können
- ... Bausteine aus der Node-RED-Modulsammlung (sie ist Teil von npm) verwenden können
- ... das Zusammenwirken Node-RED, MQTT, Mosquitto ☝ und ioBroker verstehen

Google liefert zum Suchbegriff *Node-RED* etwa 492 Millionen Einträge: Überblicksartikel, detaillierte Anleitungen, Tipps und Tricks, Source-Code, Grafiken und vieles andere mehr. Was will dann *dieser* Beitrag? *Neugierig machen, zum Probieren einladen und ein paar Tipps geben, die sonst nur schwer zu finden sind.*

Aus Platzgründen sind einige Abbildungen hier verkleinert dargestellt. Sowohl dieser Text, die Kurzfassung, wie auch die wesentlich ausführlichere Langfassung können über den Link am Ende des Beitrags abgerufen werden, auch mit den Bildern in Originalgröße. Ein ☝ im Text darauf hin, dass in der Langversion an dieser Stelle mehr Text, ein Programmcode oder eine detaillierte Anleitung zu finden ist.

Wofür wird Node-RED verwendet?

Übliche Programmiersprachen arbeiten den Programmcode Zeile für Zeile (*linear*) ab, enthalten dabei Verzweigungen, Schleifen und Funktionen (Unterprogramme). Manche Sprachen können Interrupts behandeln. Andere wieder erlauben asynchrone Abläufe: dabei wartet ein Programm vorerst nicht auf das Ergebnis einer langsameren Abfrage (etwa auf eine Datenbank- oder Internetabfrage) und macht weiter, bis das Ergebnis benötigt wird.

In einem automatisierten Haus, einem Smart Home, treten (meist zu nicht vorhersehbaren Zeiten) *Ereignisse* (events) auf, auf die das Steuerungsprogramm mit angemessenen Aktionen reagieren soll.

> **Note**
>
> Beispiel: Jemand betritt einen Raum: der Türkontakt löst ein Ereignis aus. Das Node-RED-Programm sendet ein Signal aus, mit dem das Licht eingeschaltet wird. Zusatzaufgabe: aber nicht tagsüber! Na gut, Bewegungsmelder sind schon erfunden, aber wir fangen eben mit einfachen Aufgaben an.

☝

Node-RED installieren

Was brauchen wir dafür? *Alle folgenden Programme können kostenlos aus dem Internet geladen werden.*

Node-RED ist in JavaScript geschrieben. Damit ein JavaScript-Programm auch außerhalb eines Browsers läuft, ist eine Laufzeitumgebung notwendig. Dafür wurde node entwickelt. ☝ Eine dazu passende, gut gewartete Programmbibliothek ist npm, ursprünglich eine Abkürzung für *Node Package Manager*. ☝ Und natürlich brauchen wir das Node-RED-Programm selbst.

Node-RED

Die aktuelle Version ist 4.0.2 (August 2024). Varianten werden für verschiedene Betriebssysteme angeboten samt einer Detailanleitung für Windows. Der *Quick Start* beschreibt alle Schritte. Das Installieren von Node.js haben wir schon erledigt. Mit der Eingabe von `cmd` im Windowssuchfenster unten wird die Eingabeaufforderung geöffnet. Dort

```
npm install -g --unsafe-perm node-red
```

eingeben und anschließend Node-RED mit der Eingabe

```
node-red
```

starten. Node-RED meldet die aktuelle Version (hier: v4.0.2) und weitere Details. ☝

Node-RED kennenlernen

Node-RED läuft nun auf dem eigenen Rechner und wird über http://127.0.0.1:1880 aufgerufen, ☝ Links am Bildschirm sind bereits vorinstallierte Programm-Module, sogenannte *Paletten*, zu finden. Die Node-RED Gemeinschaft hat eine große Anzahl weiterer Paletten für die unterschiedlichsten Anwendungsfälle vorbereitet. Wer will, kann auch eigene Paletten zusammen stellen.

Die Palette **Allgemein** beginnt mit den Nodes `inject` und `debug` . Diese wollen wir gleich einmal verwenden. ☝

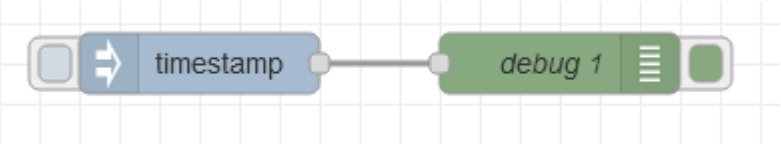
Der erste Flow

Flow bedeutet *Ablauf*, *Fluss*, *Bewegung*...

Vorläufig bleiben wir bei den Standard-Paletten. In jeder Palette gibt es wie erwähnt *Nodes* zur Auswahl. Wir bringen den inject–Node mit Anklicken und Ablegen auf die Arbeitsfläche in der Mitte des Schirms (drag-and-drop). Ebenso einen debug–Node. Der inject–Node injiziert Steuersignale in die Schaltung, der debug–Node gibt Signale (in Textform) aus. Beide sind für das Testen von Steuerungen wichtige Werkzeuge.

☝ Der inject–Node und der debug-Node sind zu verbinden: den kreisförmigen (Ausgangs-)Anschlusspunkt am rechten Rand des inject-Nodes anklicken und mit gedrückter Maustaste eine Linie zum linken (Eingangs-)Anschlusspunkt des debug–Nodes ziehen. Maustaste auslassen — fertig! So wird in Node-RED *verdrahtet*.

Solche einfachen Schritte lassen die Vorteile der grafischen Programmierung erkennen.



Dieses Bild vermittelt auch den Fluss der MQTT-Daten: durch die Verbindung hat der debug–Node die Daten des inject–Nodes abonniert.

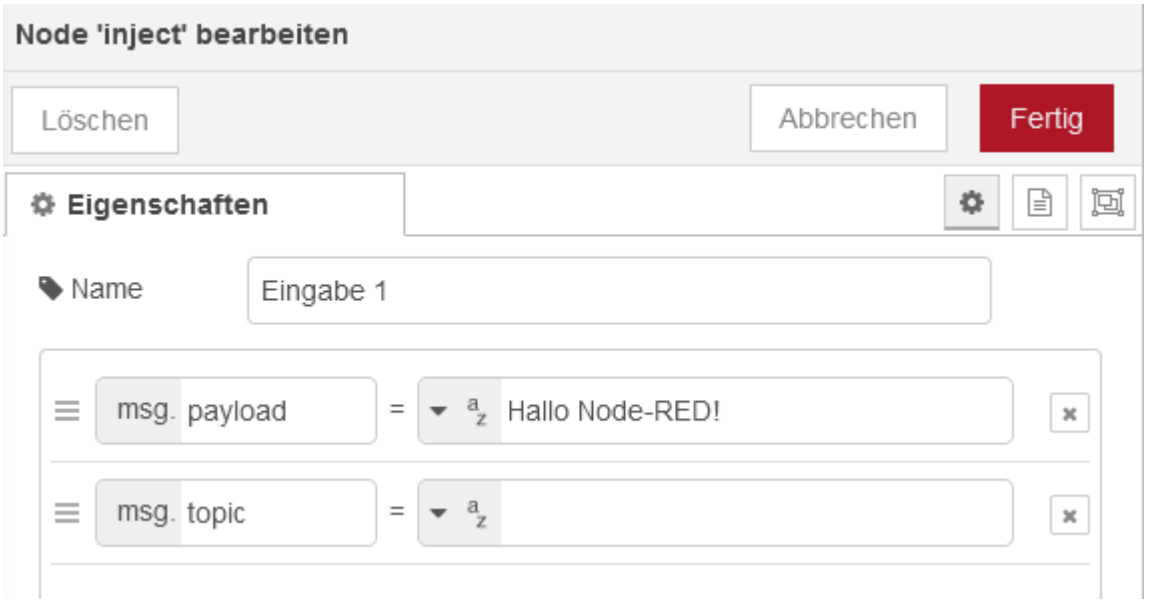
☝️

Ein paar Verbesserungen

Doppelklick auf den inject–Node: wir geben ihm den Namen `Eingabe 1`. Außerdem soll nicht die Uhrzeit, sondern die Zeichenkette "Hallo Node-RED!" gesendet werden.

payload ist der Standardname innerhalb eines JSON-Objekts für die zur Steuerung oder zur Übertragung von Zuständen verwendeten Daten. Um eine Zeichenkette zu übertragen, wird der Datentyp auf Zeichenkette (String) umgestellt; dazu das Symbol `a/z` anklicken und den gewünschten Text dahinter (ohne Anführungszeichen) eingeben.

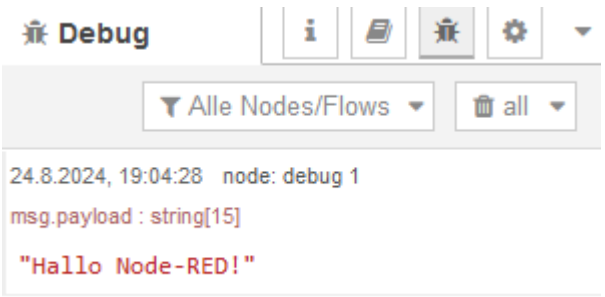
Der Eintrag topic bleibt unverändert, spielt aber bei bestimmten Nodes eine wichtige Rolle. Mehr dazu im zweiten Teil.



Auf `Fertig` klicken

Nach jeder Änderung, die ja vorerst nur im Browserfenster sichtbar ist, muss die eigentliche Steuerung auf den aktuellen Stand gebracht werden. Solange das nicht der Fall ist, erinnert *ein blauer Kreis* auf dem jeweiligen Symbol und neben dem Flow-Namen daran. ☝️

`Inject` anklicken, der Text wird im Debug-Fenster angezeigt.



> **Note**
>
> Über das Testen:
>
> - Der Inject-Node speist beliebige Signale in das System ein.
> - Der Debug-Node liest Signale aus und zeigt sie im Debugfenster.

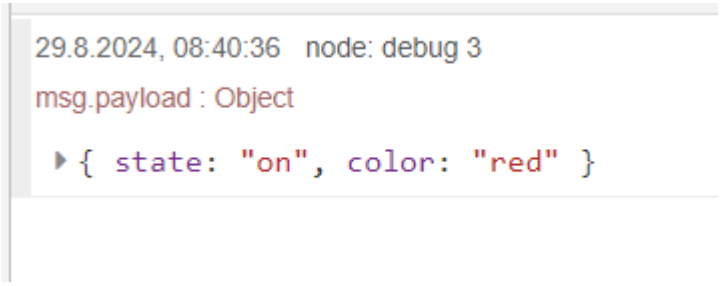
JSON

Daten im JSON-Format spielen bei Node-RED eine große Rolle. Wenn beispielsweise eine RGB-Lampe eingeschaltet und gleichzeitig auf rotes Licht umgestellt werden soll, könnte ein Steuerbefehl als JSON-Objekt (vereinfacht) so aussehen:

```
1 { "state":"on", "color":"red"}
```

Wieder muss der Typ bei `payload` umgestellt werden: die gewungenen Klammern `{ }` erinnern an die JSON-Darstellung. ☝️

Im Debug-Fenster sieht das Ergebnis (als JSON-Objekt) so aus:



Auf den kleinen Pfeil klicken liefert eine übersichtlichere Darstellung. Diese ist bei größeren Objekten sehr nützlich.

☝️

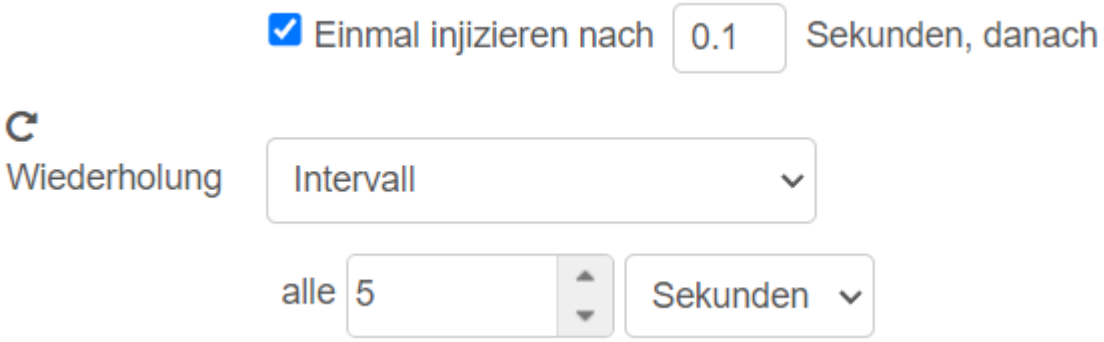
Weitere Varianten

inject–Node

Der Node soll ...

1. automatisch 0,1 Sekunden nach dem Start des Programms aktiv werden, ohne dass die Schaltfläche betätigt werden muss und
2. den (eingestellten) Befehl alle 5 Sekunden wiederholen.

So wird's gemacht:



☝️ Bei größeren Projekten werden viele Debug-Nodes verwendet, die dann auch verständliche Namen erhalten sollen. Der so gewählte Name (hier: *Ausgabe 1*) erscheint danach auch in der Debug-Liste. ☝️

Eine Lampensteuerung

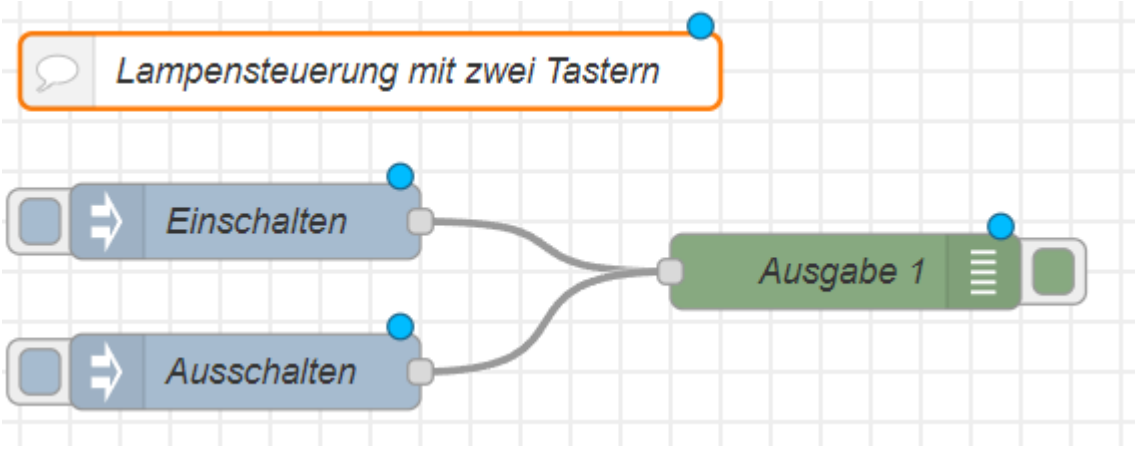
Ein- und Ausschalten

Vorerst wird der Zustand der Lampe (`eingeschaltet` oder `ausgeschaltet`) durch einen Text im Debug–Fenster simuliert. Der spätere Umbau auf eine echte Steuerung ist dann einfach.

Der vorhandene inject–Node wird geändert:

☝️

Wir brauchen noch einen zweiten Node für das Ausschalten. ☝ Der neue Node wird mit der *Ausgabe 1* verbunden und der Kommentar aktualisiert. Alles so platzieren, dass es nett aussieht:



Was fehlt noch? Richtig, die `Übernahme`.

Wenn wir nun den ersten und dann den zweiten inject–Node anklicken, sehen wir das Ergebnis im Debug–Fenster.

```
29.8.2024, 09:06:30   node: Ausgabe 1
msg.payload : Object
▸ { state: "on" }
29.8.2024, 09:06:32   node: Ausgabe 1
msg.payload : Object
▸ { state: "off" }
```

Im debug–Node kann die Ausgabe noch verschönert werden. ☝

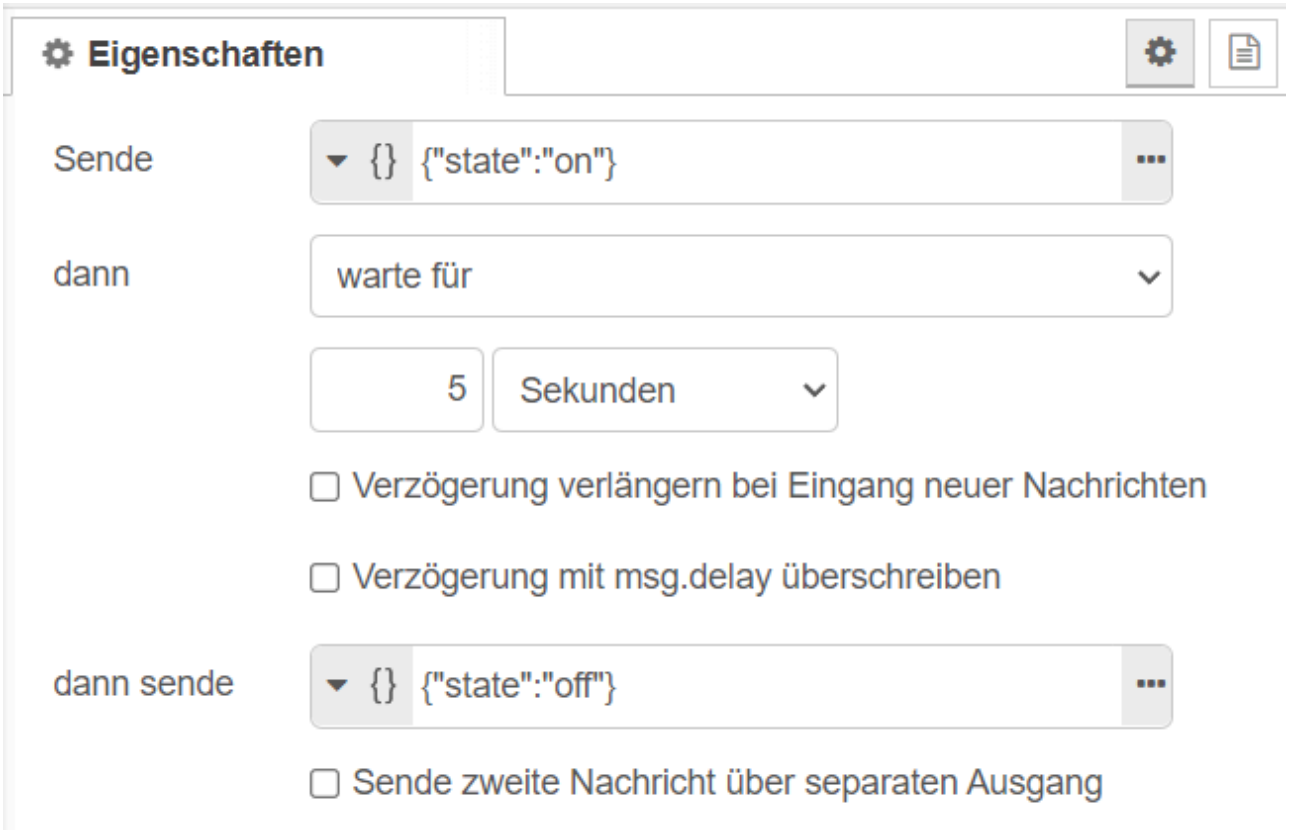
Die neue Ausgabe:

```
29.8.2024, 09:11:10   node: Ausgabe 1
msg : string[27]
"Die Lampe ist eingeschaltet"
29.8.2024, 09:11:12   node: Ausgabe 1
msg : string[27]
"Die Lampe ist ausgeschaltet"
```

Ein 3-Minuten-Licht

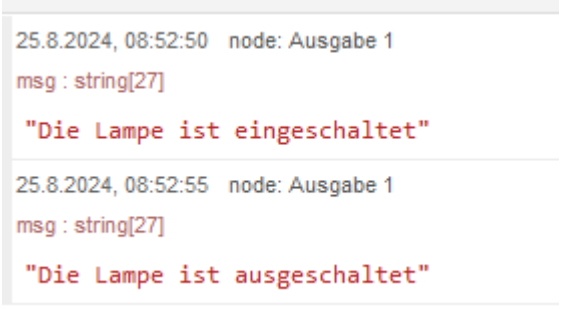
Nach einem Knopfdruck soll das Licht im Stiegenhaus 3 Minuten lang brennen, jedoch setzen wir zum Testen die Zeit auf 5 Sekunden.

Wir wählen aus der Palette `Funktion` den Node *trigger*. ☝ In dem neuen Node *trigger* ist einiges zu ändern:



☝️

Dann übernehmen, Debugger-Fenster löschen, ausprobieren:



Die Lampe wird eingeschaltet und nach 5 Sekunden selbsttätig ausgeschaltet.

Der Node *Ausschalten* ist noch da - damit kann das Licht vorzeitig ausgeschaltet werden. ☝️

Eine Schaltuhr

Neue Aufgaben: wie können wir

- die Lampe von 20 Uhr bis 24 Uhr oder
- von Sonnenuntergang bis 23 Uhr eingeschaltet lassen?

Dazu holen wir einen passenden Node aus der umfangreichen Programmbibliothek *npm*.

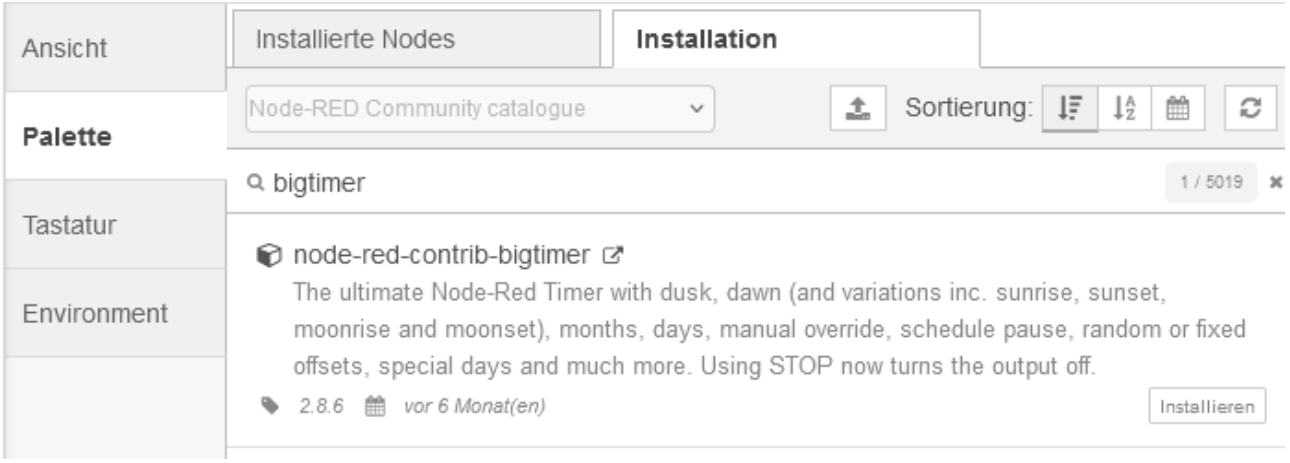
Einen Node aus *npm* verwenden

So geht das:

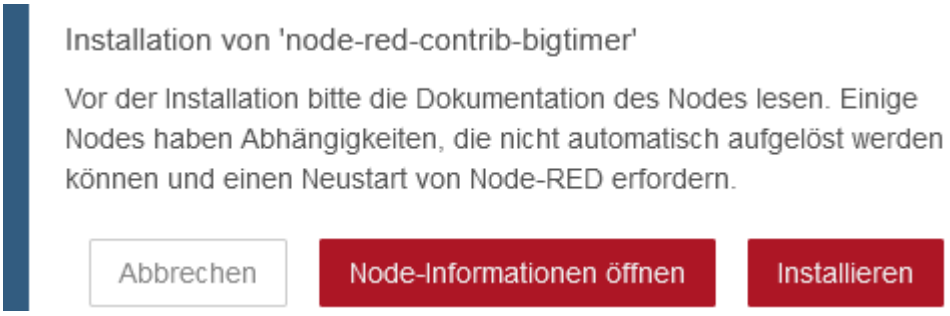
- Auf das Hamburger-Menü ☰ klicken
- `Palette verwalten` und
- `Installation` wählen

- Im Feld *Module durchsuchen* `bigtimer` eingeben

> **Note**
>
> Node-RED zeigt an, dass mehr als 5000 Module verfügbar sind. 😊 Manche sind sehr einfach, manche reichlich komplex und wieder einige fehlerhaft. 👿



Auf `Installieren` klicken und nochmals mit `Installieren` bestätigen



`Schließen` ☝ Wir ziehen einen Big Timer–Node auf die Arbeitsfläche.

Der Big Timer ist gutes Beispiel dafür, wie vielfältig und leistungsfähig die Open Source Module sind, die für Node-RED angeboten werden. Hier gibt es eine Menge einzustellen, ein konkretes Beispiel ist in der Langfassung zu finden. ☝

Damit der Big Timer die astronomischen Zeiten errechnen kann, muss der Ort angegeben werden.

- Beispiel Wien: Latitude = geografische Breite = 48.21, Longitude = geografische Länge = 16.37.

☝ Die Einstellungen werden auch auf der Webseite des Big Timers erklärt.

Der Big Timer fragt alle Einstellungen einmal pro Minute ab und ändert dementsprechend die drei Ausgänge:

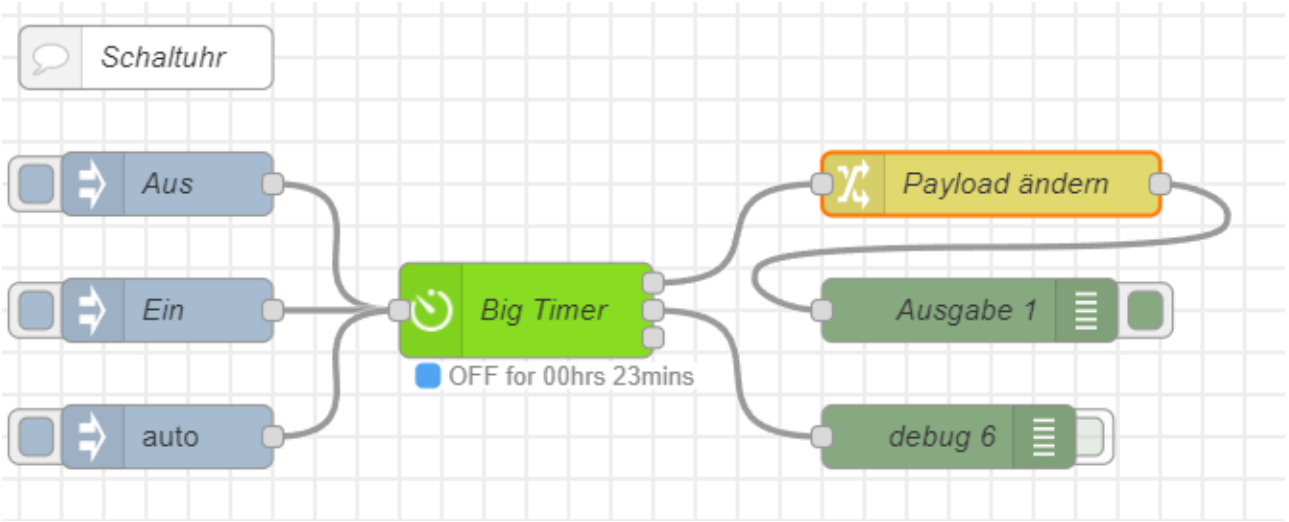
- Ausgang 1 sendet jede Minute ein `msg`-Objekt mit den Zeichenketten, die unter `ON Msg` bzw. `OFF Msg` eingetragen worden sind, als `payload`. Das `msg`-Objekt enthält noch eine Reihe weiterer Informationen
- Ausgang 2 sendet jede Minute ein `msg`-Objekt mit den Zahlen `0` bzw. `1` als `payload`. Das msg-Objekt enthält noch eine Reihe weiterer Informationen.
- Ausgang 3 liefert *nur bei einer Änderung* ein `msg`-Objekt mit den Zeichenketten von `ON Text` bzw. `OFF Text` als `payload`.

Big Timer ist ein universell einsetzbarer Node. Nur die Dokumentation könnte ausführlicher sein — manche Funktionen muss man einfach ausprobieren.

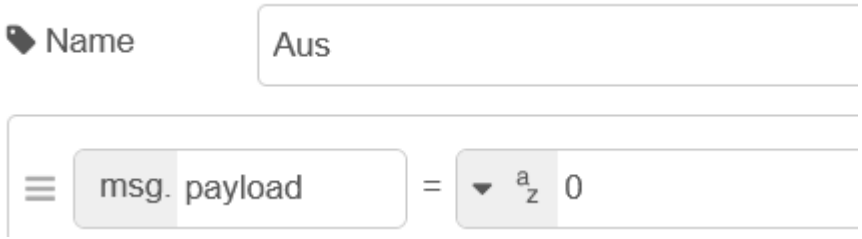
Ansteuern der Lampe

Der Big Timer gibt als `msg.payload` einen String, zum Beispiel `on` oder `off` aus. Wir brauchen aber zum Steuern unserer (im Debugger simulierten) Lampe die Objekte `{"state":"on"}` und `{"state":"off"}`. Wie das im Details gemacht wird, steht in der Langfassung.

☝ Wir fügen außerdem noch drei inject-Nodes und einen debug-Node hinzu:



Das Testen des Big Timers ist etwas mühsam, wenn wir darauf warten, dass eine eingestellte Zeit den Ausgang umschaltet. Die beiden Nodes links in der Arbeitsfläche (*Aus* und *Ein*) senden `0` bzw. `1` und *übersteuern* den Big Timer.

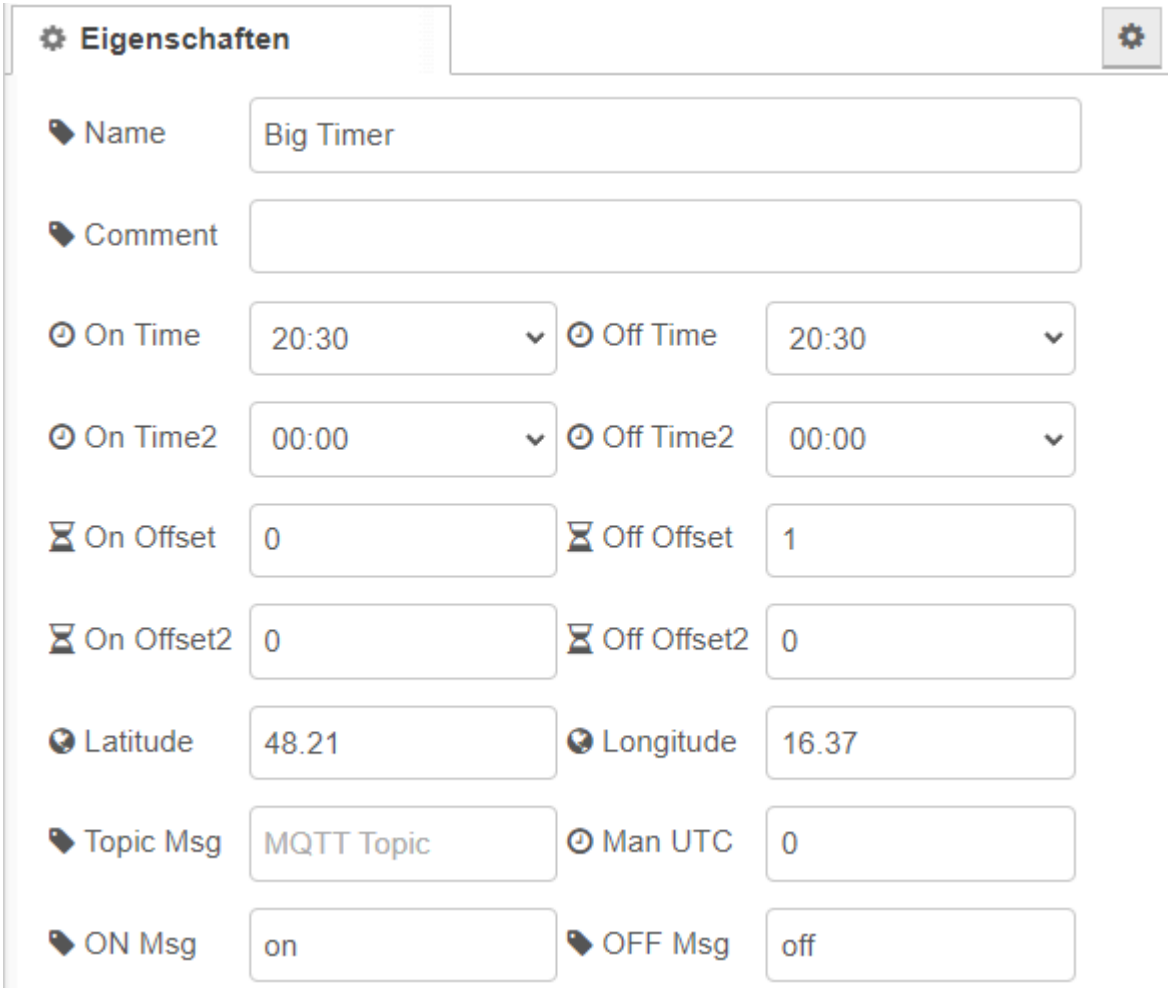


Analog dazu der Node *Ein*.

Der dritte Node *auto* schaltet mit Payload `auto` den automatischen (zeitgesteuerten) Ablauf wieder ein.

☝ Wird auf *Ein* oder *Aus* geklickt, zeigt der Debugger den Zustand der Lampe:

☝ Wir lassen den Timer die Lampe mit der *On Time* um 20:30 einschalten und mit der *Off Time* um 20:31 ausschalten. Die geografischen Koordinaten sind eingetragen Wir könnten daher auch beispielsweise mit dem Sonnenuntergang schalten.



Wie erwartet wird die Lampe um 20:30 eingeschaltet und um 20:31 ausgeschaltet:

Jede Minute wird diese Meldung ausgegeben. ☝

Stromstoßrelais

Manche Lichtsteuerungen arbeiten mit Stromstoßrelais, vor allem, wenn das Licht von mehreren Stellen ein- und ausgeschaltet werden soll. Erster Tastendruck - Licht geht an. Zweiter Tastendruck: Licht geht aus.

Natürlich erlaubt Node-RED auch diese Steuerung. Dazu laden wir wie schon vorhin beschrieben die Palette `node-red-contrib-toggle`. *toggle* heißt *umschalten*. ☝

Zusammenfassung

Die drei Beispiele zeigen, wie herkömmliche Lichtsteuerungen durch Smart Home-Komponenten ersetzt werden können. Jede herkömmliche Lösung hat ihre Vor- und Nachteile. Node-RED kann jedoch noch mehr, nämlich die Vorteile aller drei Varianten in *einer* Steuerung kombinieren. Das ist Thema für einen weiteren PCNEWS-Beitrag.

Arbeiten mit realer Hardware

Es ist zwar recht nett, eine Steuerung zu simulieren. Richtig interessant wird es mit echter Hardware. Dazu müssen die Steuerbefehle über das LAN den Rechner (mit dem Node-RED-Programm) verlassen können und als Funksignale die Hardwarekomponenten erreichen. Dazu dienen MQTT und ZigBee.

MQTT

Die Funktion von MQTT wurde bereits im PCNEWS Heft 181 beschrieben. Für die Experimente ist ein MQTT-Broker notwendig. Häufig wird dafür Mosquitto verwendet. Es gibt davon eine Windows-Version.

Natürlich brauchen wir zusätzlich *echte* Hardware, nicht nur *simulierte*. Node-RED liefert bereits MQTT Signale, die aber meistens noch weiter in die Steuerbefehle der verwendeten Hardware übersetzt werden müssen. Zwei Varianten werden hier vorgestellt: Tasmota und ZigBee. Beide wurden in den PCNEWS bereits erklärt. Geplant ist, später auch Matter dazu zu nehmen.

Tasmota

Tasmota-Komponenten verwenden das WLAN und können ohne zusätzliche Hardware mit dem MQTT-Broker reden. Daher werden sie hier zuerst besprochen. Im PCNEWS Heft 181 sind Details zu Tasmota-Komponenten zu finden.

Aktoren und Sensoren

Ein paar Vorschläge:

- Mit dem ESP32, wie er im PCNEWS Heft 180 beschrieben worden ist, können wir viele Experimente machen.
- Wer gleich *richtige Verbraucher* steuern will, kauft einen Zwischenstecker mit Tasmota-Firmware, zum Beispiel den NOUS A1T, erhältlich unter anderem bei Reichelt oder im Doppelpack bei Amazon.

Damit haben wir Ausgabeelemente (Aktoren). Leider gibt es nur wenig Tasmota-Eingabegeräte, zum Teil auch relativ teuer.

- An den ESP32 lassen sich Taster oder Temperaturfühler anschließen. Er kann sogar Magnetfelder messen. Somit ist der ESP32 auch ein Sensor.

Paletten für Tasmota

In *npm* werden mehrere Paletten für Tasmota angeboten. zum Beispiel *node-red-contrib-tasmota* und *node-red-contrib-tasmotas*. Nach der Installation des MQTT-Brokers und der Tasmota Palette wird der `Ausgabe 1–Node` durch einen passenden Tasmota-Node ersetzt. Eventuell ist noch ein change–Node zur Anpassung des Steuerbefehle dazwischen zu schalten, Fertig! Alle Beispiele funktionieren nun mit realer Hardware

Wie das im Details aussieht, wird im Folgebeitrag beschrieben.

ZigBee

ZigBee baut ein eigenes Funk-Netzwerk auf und braucht daher noch einen ZigBee-Adapter, meist in Form eines USB-Sticks.

Bei ZigBee ist die Auswahl an Sensoren und Aktoren schon wesentlich größer. ☝

Sensoren

Ein paar Sensoren zur Auswahl:

- Ein Türkontakt eignet sich für Experimente auch gut.
- Ein Bewegungsmelder ist auch ein netter Sensor, nicht allzu teuer und in der Praxis gut zu brauchen. ☝
- Wer einen Schalter oder Taster verwenden möchte, wie er auch im Haushalt vorkommt, hat eine große Auswahl.

Paletten für ZigBee

Das Programm Zigbee2MQTT kann über die Kommandozeile angesprochen werden und wandelt ZigBee-Kommandos in MQTT um. Für Node-RED gibt es auch mehrere Programmbibliotheken, zum Beispiel *node-red-contrib-zigbee2mqtt*. Alles, was unter Paletten für Tasmota erklärt wurde, gilt auch hier. ☝

ioBroker und Node-RED

ioBroker ist ein Programmpaket, das verspricht, besonders viele Komponenten unterschiedlicher Hersteller und mit unterschiedlichen Protokollen zu einer kompletten Internet-of-Things-Lösung (IoT) zu verbinden. Besonders bemerkenswert: die schönen Grafiken und Schnittstellen.

☝

Zusammenfassung

Der Beitrag zeigt (vor allem in der Langversion) Schritt für Schritt die Installation und die Anwendung von Node-RED für eine einfache Lampensteuerung. Die Lampe wird durch einen Ausgabetext simuliert.

In der Fortsetzung dieses Beitrags

- wird echte Hardware angesteuert
- werden weitere Nodes von Node-RED erklärt
- werden Steuerungen mit JavaScript programmiert
- wird ein Blick auf die Anwendung von Zustandsdiagrammen geworfen
- wird das Smart Home über einen Bot mit Telegram verknüpft

To be continued ... Fortsetzung folgt ...

Zur Langversion

Die Entwicklung aller Beispiele wird in der Langversion sehr detailliert und mit vielen Zusatzinformationen und Bildern erklärt. Ferner können alle vorgestellten Flows auch unter dieser Adresse geladen werden. Lehrreicher ist es trotzdem, die Beispiele von Hand einzugeben.

Link und QR-Code: http://pcnews.at/markdown/n182/index.html oder https://t.ly/ROrK9

Erstellt mit QR Code Generator - kostenfrei, ohne Login, ohne Konto : https://goqr.me/

Short URL erstellt mit https://t.ly/

Copyright