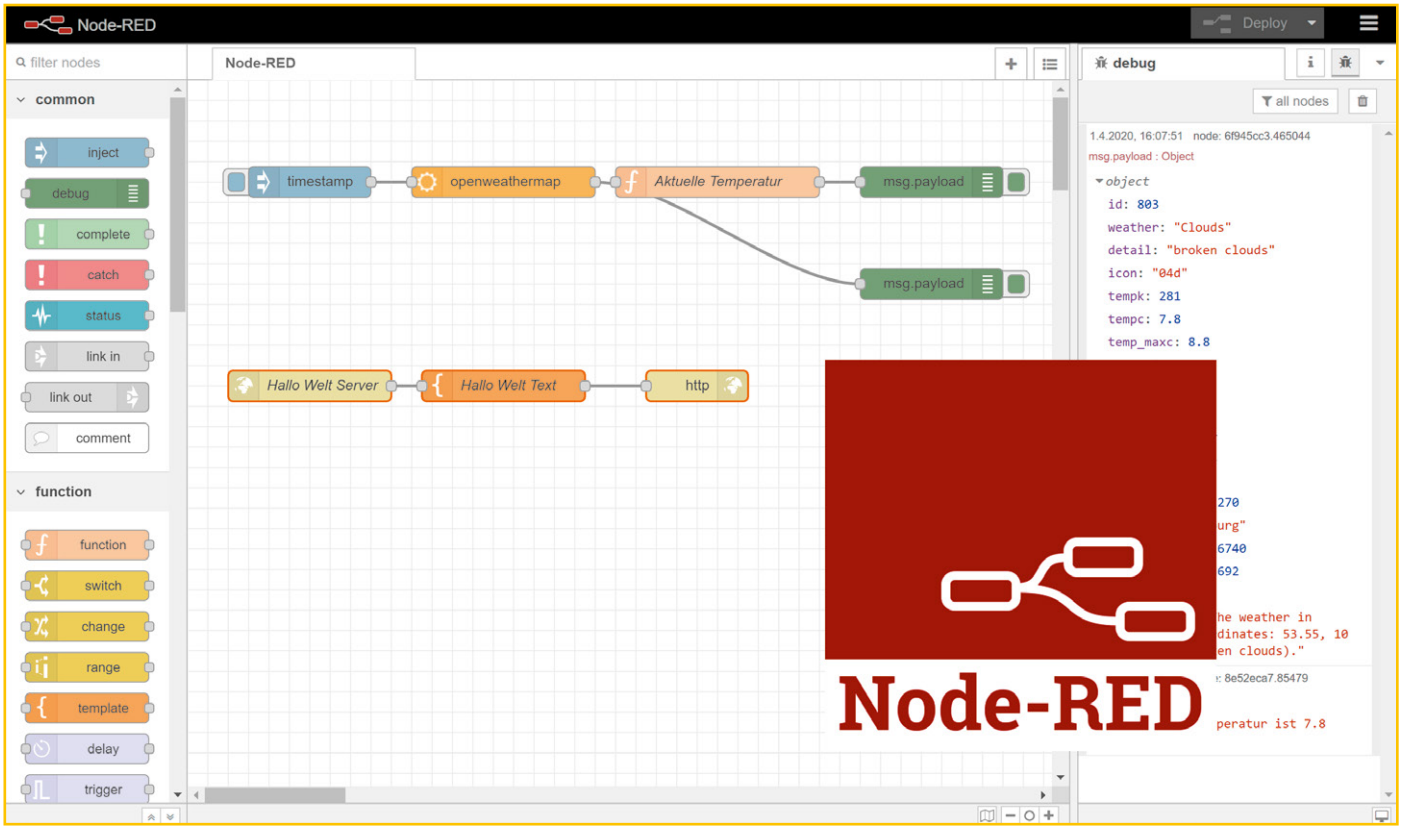
Programmieren (fast) ohne Code

Node-RED als universelles Prototyping-Tool

Teil 1

Mal eben ein neues elektronisches Gerät mit seriellen Daten testen oder am Raspberry Pi angeschlossene Elektronik in das System einbinden und die Daten an andere Empfänger weiterleiten. Oder den Abruf des aktuellen Wetters bis hin zur gesamten Steuerung des Smart Homes – das alles lässt sich mit dem Open-Source-Software-Tool Node-RED realisieren. Und dabei ist die Programmierung nahezu kinderleicht. Denn als visuelle Programmieroberfläche ist bei Node-RED kaum selbst geschriebener Code notwendig. In diesem Beitrag beschäftigen wir uns mit den Grundlagen, der Installation auf einem Raspberry Pi und ersten Anwendungsbeispielen mit diesem universellen Software-Tool.



Allgemeines

Grundlage für Node-RED ist Node.js – eine serverseitige, ereignisgesteuerte Plattform zum Betrieb von Netzwerkanwendungen. Sie ist hauptsächlich dazu gedacht, Webserver aufzusetzen. Node.js wird in der JavaScript-Laufzeitumgebung „V8" ausgeführt. Ursprünglich für Google Chrome entwickelt, bietet sie eine ressourcensparende Architektur, die eine besonders große Anzahl gleichzeitig bestehender Netzwerkverbindungen ermöglicht.

Das auf Performance ausgelegte Node.js zeichnet sich durch nicht blockierende In- und Outputs aus. Sogenannte Threads sollen dabei verhindern, dass vergleichsweise langsame Operationen wie Zugriffe auf das Netzwerk oder Dateisystem andere, schnelle Anweisungen für den Prozessor blockieren.

Ein weiteres Merkmal von Node.js sind Module, die zum einen direkt in das Binärpaket kompiliert, aber auch zusätzlich für weitere Funktionalitäten eingebunden werden können. Verwaltet wird dies vom Paketmanager npm (früher: node package manager), für den es ein riesiges Angebot von Open-Source-Bibliotheken (Paketen) gibt. Er sorgt unter Berücksichtigung der Abhängigkeiten für die Installation, Aktualisierung und das Kompilieren der Module. „Abhängigkeit" bedeutet in diesem Zusammenhang u. a., dass für das zu installierende Paket weitere Pakete notwendig sein können, die für die Ausführung vorhanden sein müssen. Diese werden dann bei der Installation automatisch hinzugefügt.

Node-RED schließlich ist ein von IBM entwickeltes, ereignisgesteuertes Tool mit einer visuellen Drag-and-Drop-Programmieroberfläche. Immer wiederkehrende Abläufe werden in sogenannten Nodes gekapselt, um den Programmieraufwand so gering wie möglich zu halten. Diese Nodes werden untereinander verbunden und ergeben in einem so genannten Flow das Anwendungsprogramm.

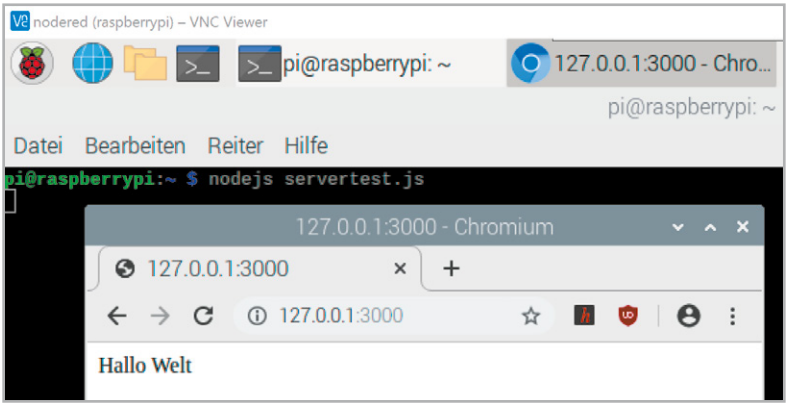


Bild 1: Nach dem Start des Skripts gibt der Browser die zuvor programmierte Rückmeldung „Hallo Welt".

Bild 2: „Hallo Welt" Server in Node-RED

Von den Nodes gibt es zurzeit mehr als 2.500 für die verschiedensten Anwendungszwecke [1]. Zudem gibt es unter [1] viele Beispielprogramme (Flows).

„Hallo Welt" in Node.js

Dass Node.js vor allem für Server-Anwendungen gedacht ist, merkt man schnell, wenn man einen Server aufsetzt, der beispielsweise im Browser (Client) ein einfaches „Hallo Welt" zurückgibt. Der Code an sich ist übersichtlich:

servertest.js

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  res.write(,Hallo Welt');
  res.end();
}).listen(3000);
```

Startet man das Skript beispielsweise auf einem Raspberry Pi mit nodejs servertest.js kann man mit einem Browser den lokal bereitgestellten Server (http.createServer) auf dem Port 3000 (.listen(3000)) aufrufen und erhält die Antwort „Hallo Welt" (Bild 1).

Zum Vergleich: „Hallo Welt" in Node-RED

Die gute Nachricht: In Node-RED kann man dieses Beispiel gänzlich ohne Code programmieren. Wer also mit JavaScript, Kommunikation per http oder Ähnlichem noch nichts zu tun hatte, braucht sich nicht zu sorgen. In Node-RED sind es drei Nodes, eine Zeile Text und eine URL, die in der visuellen Programmieroberfläche konfiguriert werden müssen (Bild 2). Eine ausführliche Erklärung dazu folgt weiter unten bei der Beschreibung des Node-RED-Editors.

Ruft man den Browser auf, erhält man wie im Beispiel mit dem servertest.js-Script wieder die „Hallo Welt"-Ausgabe im Browser (Bild 3).

Installation auf einem Raspberry Pi

Einer der einfachsten Wege, sich schnell ein System mit den benötigten Software-Komponenten Node.js, npm und Node-RED aufzusetzen, ist die Installation auf einem Raspberry Pi. Ein Raspberry Pi 4 Model B mit 2 GB RAM ist hierfür völlig ausreichend [2]. In sieben Schritten kommen wir damit zu einer voll funktionsfähigen Node-RED Installation.

Schritt 1

Zunächst lädt man sich den Raspberry Pi Imager – ein Software-Tool zur einfachen Auswahl und Installation des Betriebssystems für den Raspberry Pi – herunter [3]. Nach der Installation und Start der Software wählen wir bei Operating System → Choose OS den obersten Eintrag Raspbian (ein auf Linux basierendes Betriebssystem) aus und als Nächstes die entsprechende SD-Karte (microSD, mindestens 8 GB) mit SD-Card → Choose SD-Card. Anschließen klicken wir auf Write (Bild 4). Das Betriebssystem-Image wird nun heruntergeladen und auf die SD-Karte geschrieben (Bild 5).

Schritt 2

Jetzt wird die SD-Karte in den Karten-Slot des Raspberry Pi eingesteckt sowie Monitor, Keyboard, Maus und optional ein Netzwerkkabel angeschlossen. Als Letztes folgt die Spannungsversorgung. Der Raspberry Pi 4 benötigt ein Netzteil mit 5 Volt und 3 A – hier sollte man auf keinen Fall sparen, vor allem, falls man später leistungshungrige Geräte an die USB-Ports anschließen möchte.

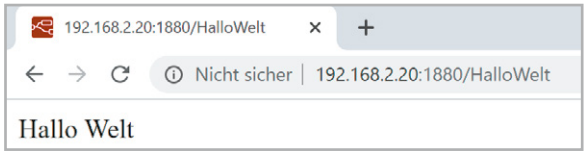


Bild 3: „Hallo Welt" im Browser – erzeugt durch Node-RED

Bild 4: Auswahl des Betriebssystems und Speichermediums

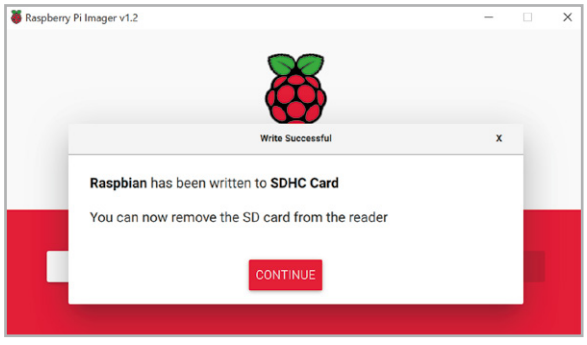


Bild 5: Nach erfolgreichem Schreibvorgang kann die SD-Karte entfernt werden.

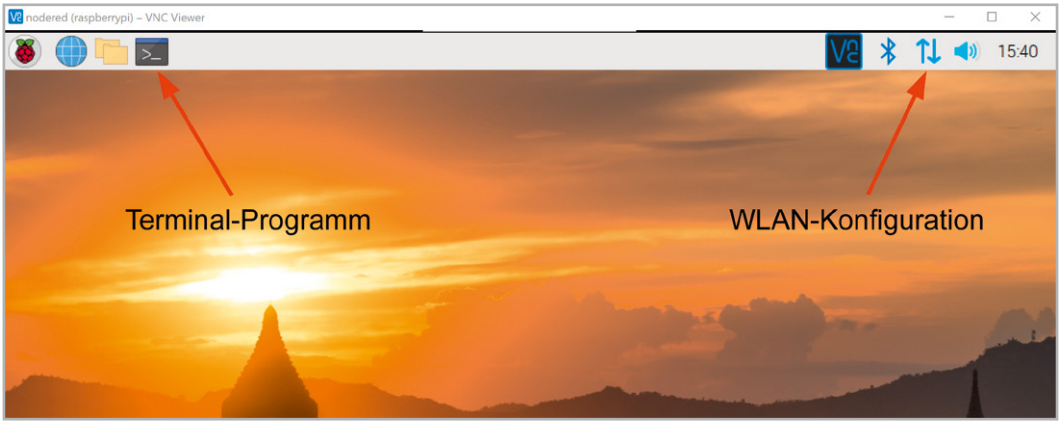


Bild 6: Raspberry Pi Desktop mit Terminal-Icon und WLAN-Konfiguration

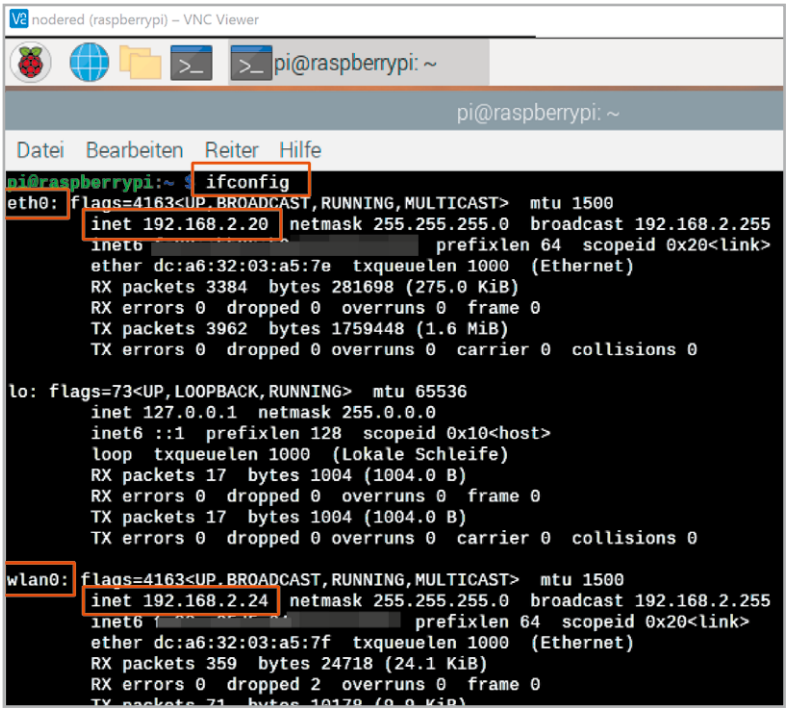


Bild 7: Mit ifconfig in einem Terminal erhält man die IP-Adressen des Raspberry Pi.

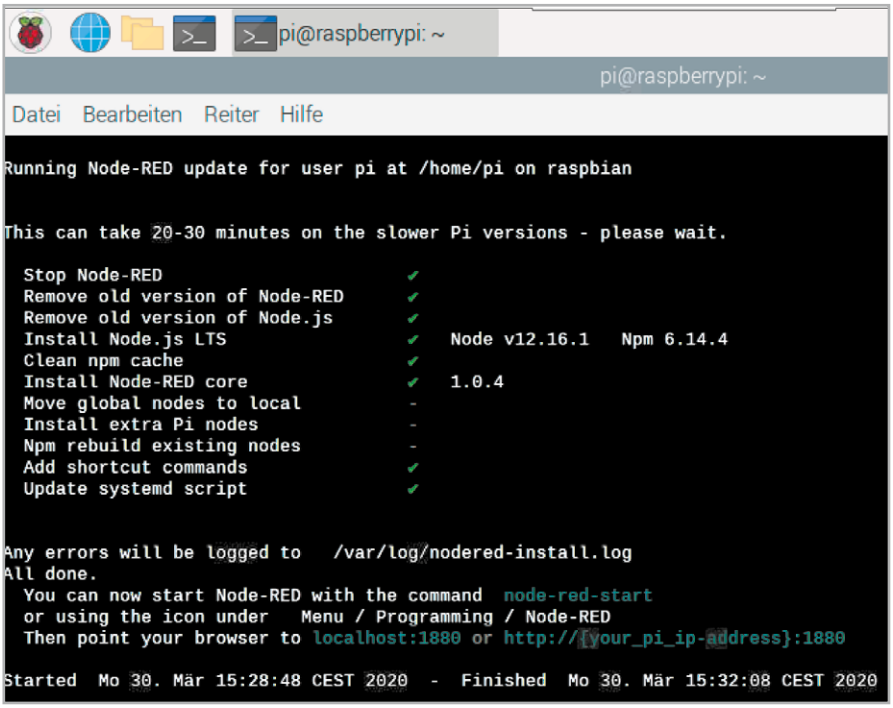


Bild 8: Installation von Node.js, npm und Node-RED

Schritt 3

Nach dem Booten gelangt man auf den Desktop des Raspberry Pi. Beim ersten Start werden einige Informationen zur Konfiguration abgefragt, die man entsprechend eingeben sollte. Hier können je nach Konfiguration (LAN/WLAN) noch die Informationen für das WLAN ergänzt werden, damit der Raspberry Pi später auch per Funk im lokalen Netz erreichbar ist.

Nach einem Neustart sollte der Raspberry Pi jetzt über die aktivierten Netzwerk-Schnittstellen erreichbar sein.

Manuell kann man das WLAN übrigens über das WiFi-Symbol in der rechten oberen Ecke des Bildschirms (Bild 6) konfigurieren. Hier aktiviert man zunächst das WiFi, wählt den Access Point und gibt dann entsprechend die Zugangsdaten ein.

Schritt 4

Da wir später die IP-Adresse des Raspberry Pi für Zugriffe von außen im lokalen Netz benötigen, öffnen wir mit STRG + T bzw. dem kleinen Bildschirmsymbol an der oberen Leiste (Bild 6) ein Terminal und geben auf der Kommandozeile ifconfig ein. Hier erhalten wir unter eth0 sowohl die IP-Adresse für den LAN-Anschluss bzw. die IP-Adresse für den WLAN-Anschluss unter wlan0 (Bild 7). In unserem Beispiel ist der Raspberry Pi über die LAN-Schnittstelle (eth0) unter der IP-Adresse 192.168.2.20 und auf der WLAN-Schnittstelle (wlan0) unter 192.168.2.24 erreichbar.

Optional kann man nun den Fernzugriff des Raspberry Pi per SSH auf die Kommandozeile bzw. mit einem VNC-Server (RealVNC/TightVNC) auch auf den Desktop von einem entfernten Gerät im lokalen Netzwerk per VNC-Viewer einrichten.

Schritt 5

Als Nächstes steht nun die Installation von Node.js, npm und Node-RED an. Glücklicherweise sind die Installation auf einem Raspberry Pi und vor allem die verschiedenen Optionen auf der Node-RED Webseite unter [4] ausführlich erläutert.

Daher beschränken wir uns hier auf die Erklärung der Installation, die per Eingabe von:

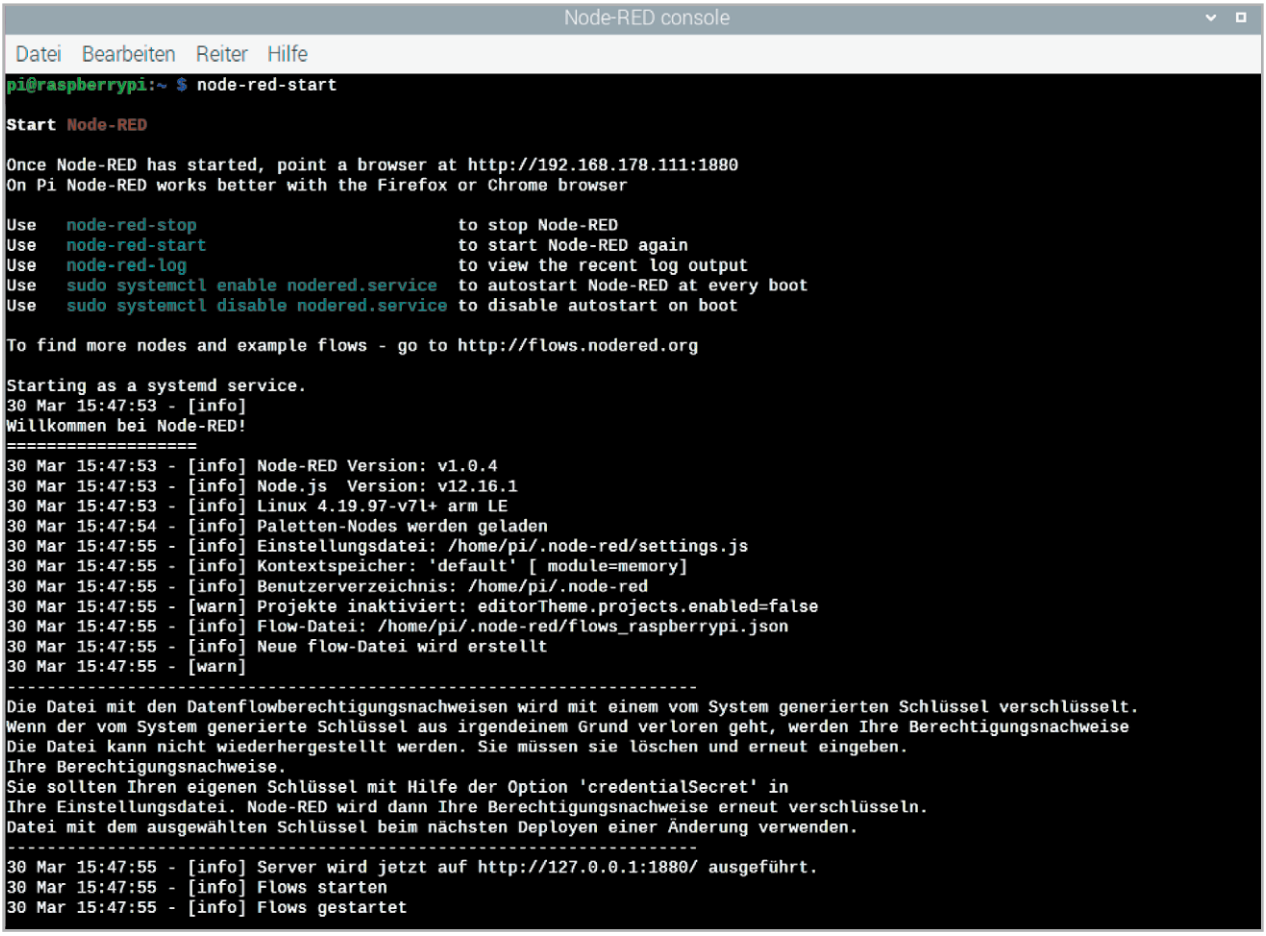


Bild 9: Node-RED mit der Ausgabe der Meldungen der Logdatei

bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)

auf der Kommandozeile in einem Terminal automatisch abläuft. Dabei werden die u. U. bereits vorhandenen Versionen von Node.js, npm und Node-RED entfernt und aktuelle Versionen bereitgestellt. Zudem werden die Nodes von Node-RED lokal und je nach Konfiguration möglicherweise zusätzlich für den Raspberry Pi spezifische Nodes installiert. Auf diese werden wir in Teil 2 (siehe Kasten „Weitere Beiträge zu Node-RED") noch zurückkommen.

Da wir eine frische Installation auf der SD-Karte nutzen, brauchen wir uns in diesem Fall nicht um ein sonst empfehlenswertes Back-up zu kümmern und bestätigen die folgenden Fragen mit y(es).

Das Herunterladen der Software und die Installation kann einige Minuten dauern. Nach Abschluss sollte der Prozess ohne Fehlermeldung beendet sein und wie in Bild 8 aussehen.

Schritt 6

Mit der Eingabe von:

node-red-start

starten wir Node-RED als Service und bekommen als Rückmeldung die Ausgabe der Node-RED Logdatei (Bild 9).

Drei der Einträge wollen wir uns dabei genauer anschauen:

1. Hier werden die IP-Adresse und der Port (1880) angezeigt, unter der wir die Node-RED-Oberfläche im lokalen Netzwerk erreichen.
2. Das Node-RED-Log gibt zum Start immer Informationen über aktuelle Versionen der Software aus. Hier lohnt es sich nachzuschauen, falls es später einmal Probleme geben sollte.
3. Auch lokal ist die Node-RED-Oberfläche zu erreichen. Will man vom Raspberry Pi darauf zugreifen, dann nutzt man die dort angegebene IP-Adresse.

Schritt 7

Der letzte Schritt zum ersten Aufruf der Node-RED-Oberfläche ist nun die Eingabe der IP-Adresse (auf den Port 1880 als zusätzliche Angabe achten), am besten von einem Rechner aus dem lokalen Netzwerk aus. So kann man zum einen die korrekte Installation der Software als auch die Erreichbarkeit testen. Zudem läuft der Zugriff mit einem externen Browser flüssiger ab als auf dem Raspberry Pi selbst.

Node-RED-Grundlagen

Schauen wir uns nun in Bild 10 einmal die Oberfläche genauer an: Die Node-RED-Programmieroberfläche ist in drei Bereiche aufgeteilt. Links sind die einzelnen Nodes, in Kategorien aufgeteilt, aufgelistet. Ganz oben gibt es ein Suchfeld für Nodes, das gerade bei wachsender Anzahl von Nodes sehr hilfreich ist.

Im mittleren Bereich ist der Editor oder Flow-Bereich. Hierher werden die Nodes per Drag-and-Drop gezogen, verbunden und es entsteht das flussgesteuerte Programm.

Der rechte Bereich ist für Informationen zu den einzelnen Nodes, Debug-Nachrichten, Konfigurations-Nodes und Kontextdaten gedacht. Über die kleinen Icons im oberen Bereich kann man die verschiedenen Optionen anwählen.

Über das „Hamburger"-Menü (drei horizontale Balken) kann man Flows im- und exportieren, Einstellungen verändern und die installierten Nodes aktualisieren bzw. neue installieren.

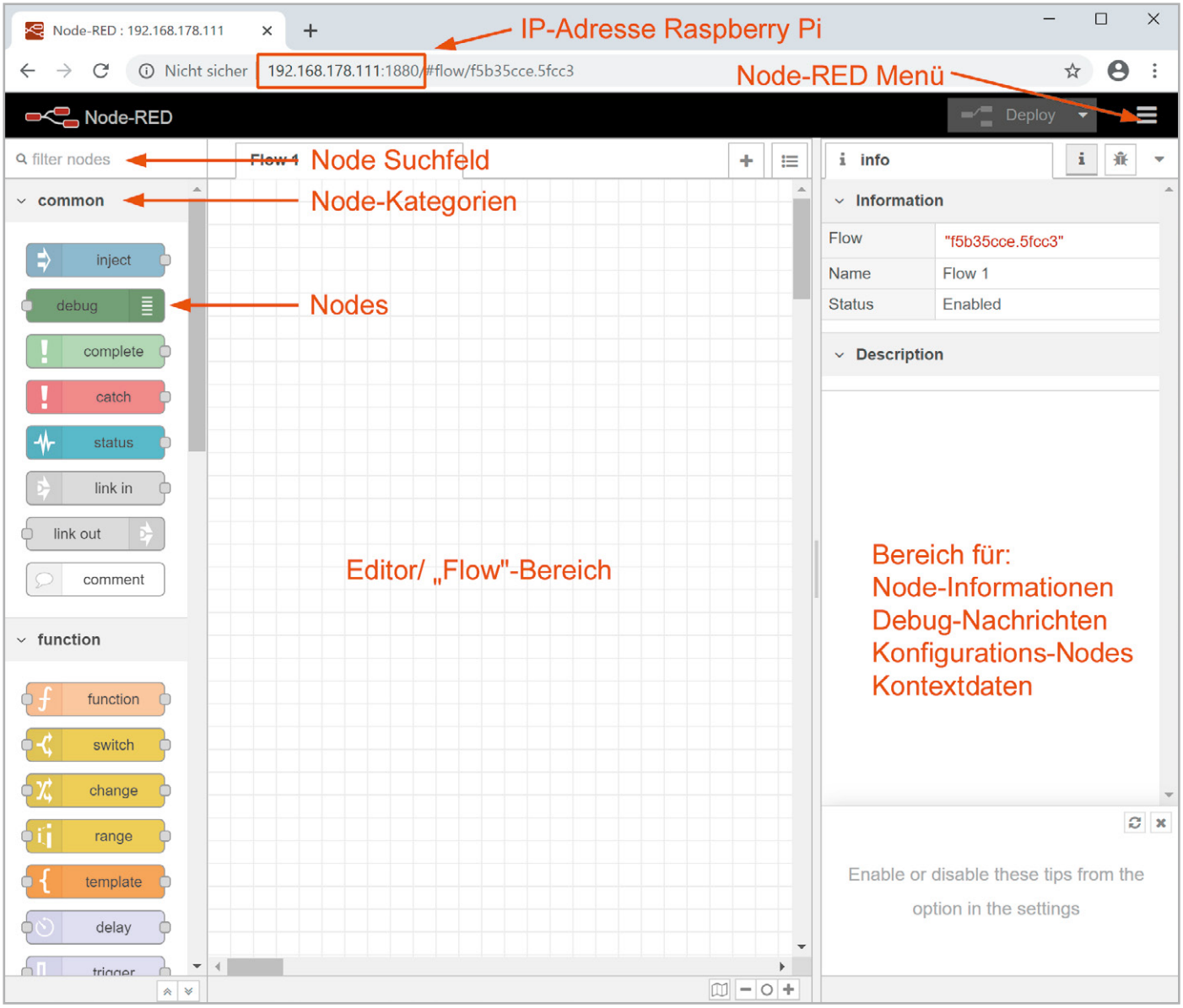


Bild 10: Die Oberfläche von Node-RED

„Hallo Welt"-Server in Node-RED

Kommen wir nun zurück auf den eingangs erwähnten Hallo-Welt-Server, den wir jetzt mithilfe unserer Node-RED-Installation programmieren wollen. Dazu ziehen wir in der Node-RED-Oberfläche aus dem linken Bereich, wo sich die Nodes befinden, aus der Kategorie network den http in Node auf die mittlere Fläche. Hier entsteht unser Flow (= das Programm) durch Verknüpfen der einzelnen Nodes (Bild 11).

Tipp: Bleibt man mit dem Mauszeiger über dem ausgewählten Node, bekommt man zusätzliche Informationen in einem Pop-up-Fenster. Außerdem werden im rechten Fenster nach der Auswahl eines Nodes weitere Informationen angezeigt, sofern dort die entsprechende Funktion info ausgewählt wurde.

Das orange Dreieck am oberen rechten Rand der Node signalisiert, dass in dem Node noch Einstellungen vorgenommen werden müssen. Auch hier hilft ein kurzes Verharren des Mauszeigers über dem Dreieck für weitere Informationen. Ein Doppelklick auf den

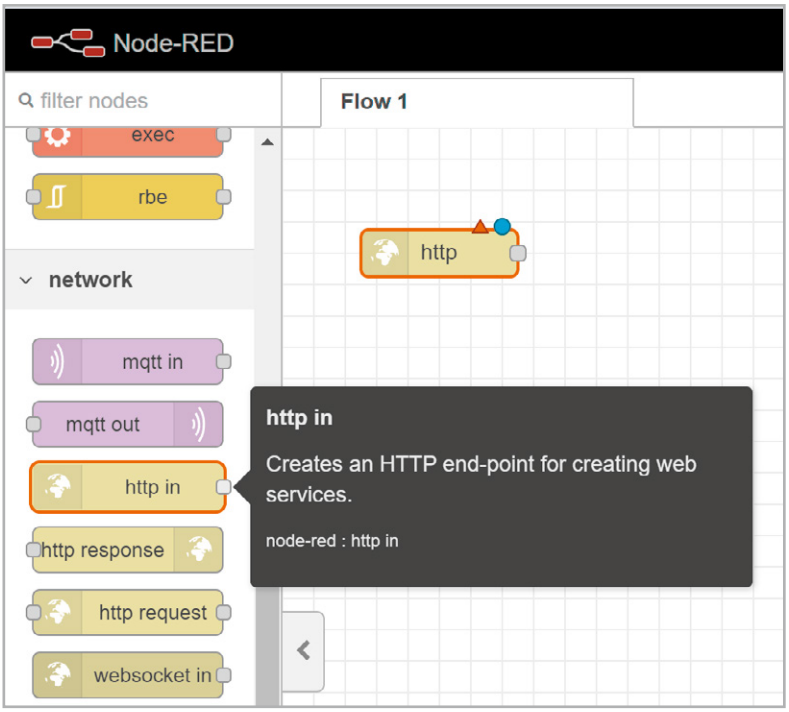


Bild 11: Ein http in Node wird auf die Flow-Oberfläche gezogen.

Node öffnet den dazugehörigen Dialog. Der blaue Kreis signalisiert, dass dieser Node noch nicht deployed, d. h. im Flow aktiviert wurde. Dazu später mehr.

Das Verbindungselement, an das eine oder mehrere virtuelle Leitungen (wire) angeschlossen werden können und so später die einzelnen Nodes verbunden werden, ist jeweils links oder rechts bzw. an beiden Seiten (je nach In-/Output oder beides) als graues Quadrat zu sehen (Bild 12).

Wir definieren nun die Eigenschaften des Servers, indem wir als Methode GET auswählen, die URL mit „/HalloWelt" festlegen und dem Node den Namen „Hallo Welt Server" geben. Am Ende schließen wir die Konfiguration ab, indem wir auf den „Done"-Button klicken (Bild 13).

Um den Text „Hallo Welt" auszugeben, wenn jemand die URL unseres Servers aufruft, benutzen wir einen Template-Node, den wir in der Kategorie function unter template (linke, geschweifte Klammer als Node-Symbol) finden. Wir ziehen diesen rechts neben den http in Node, doppelklicken den Node und legen als Payload „Hallo Welt" fest. Dem Node geben wir den Namen „Hallo Welt Text" und bestätigen wieder mit „Done" (Bild 14).

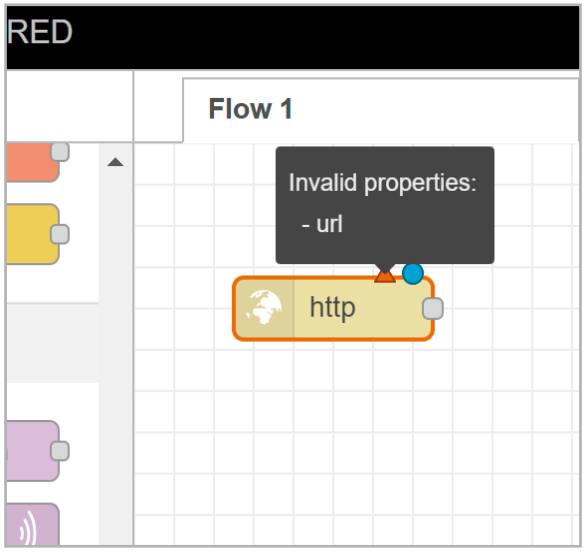


Bild 12: http in Node mit Statusanzeigen

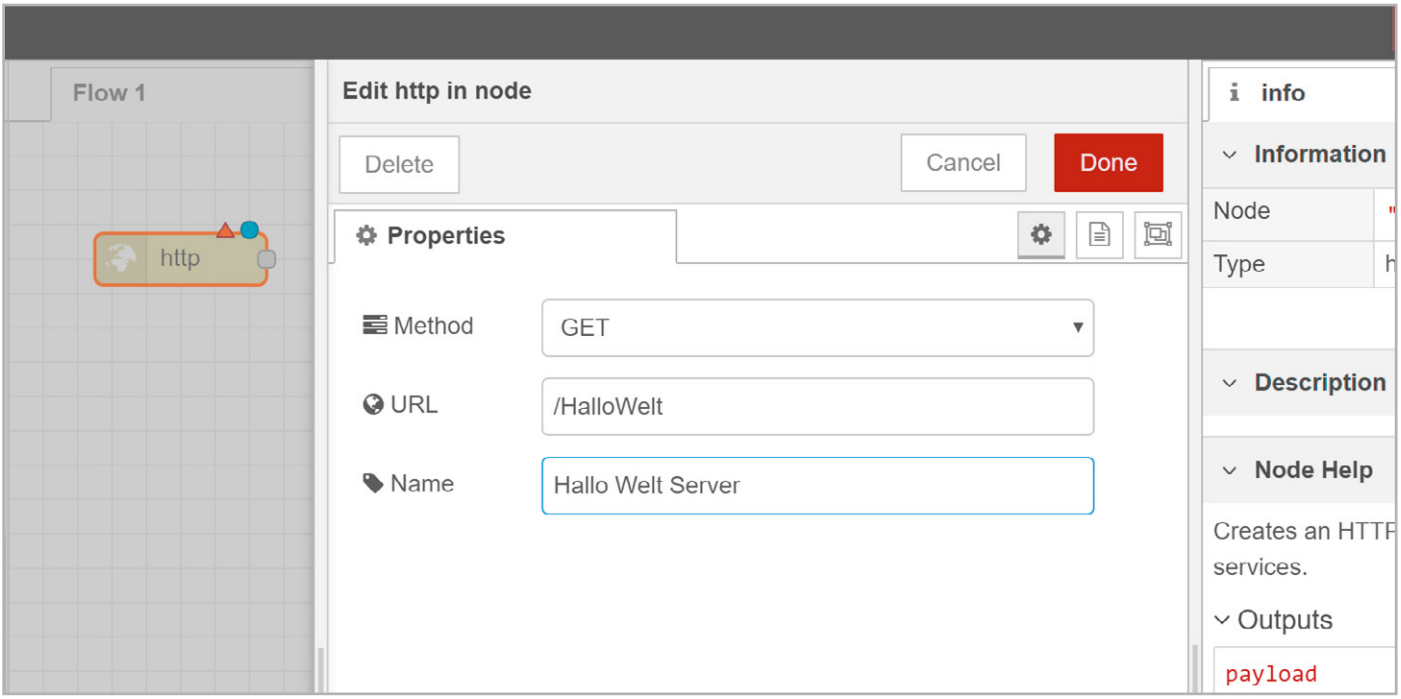


Bild 13: Eigenschaften des „Hallo Welt"-Servers

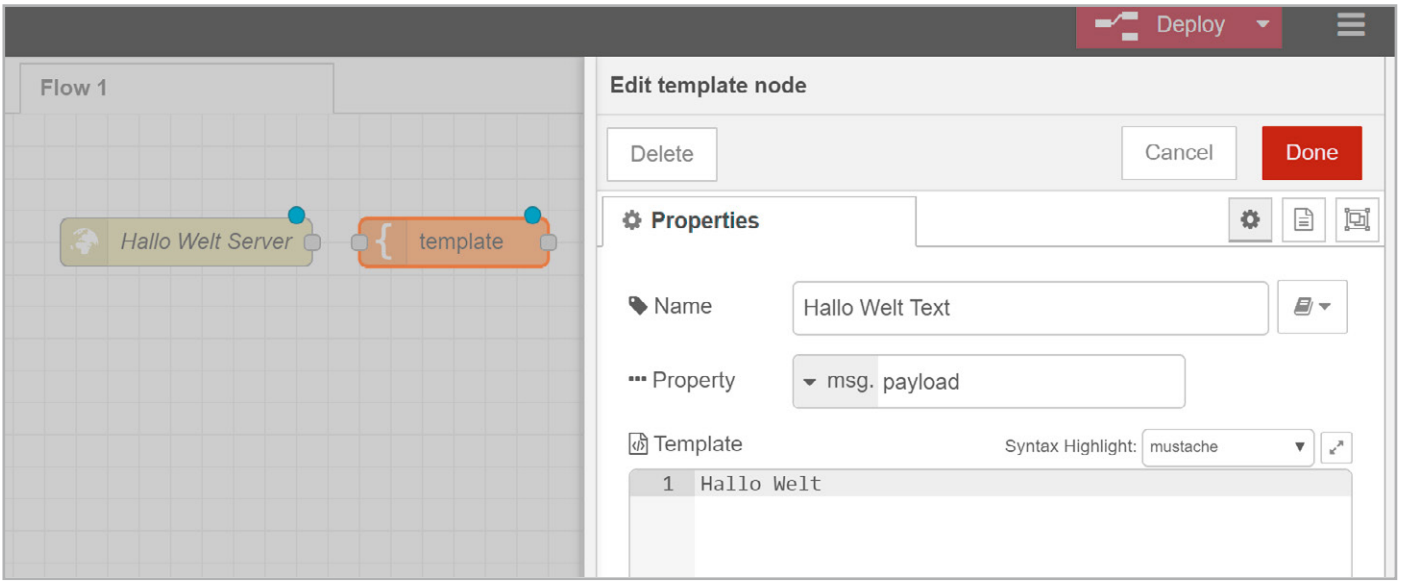


Bild 14: Konfiguration des Template-Node

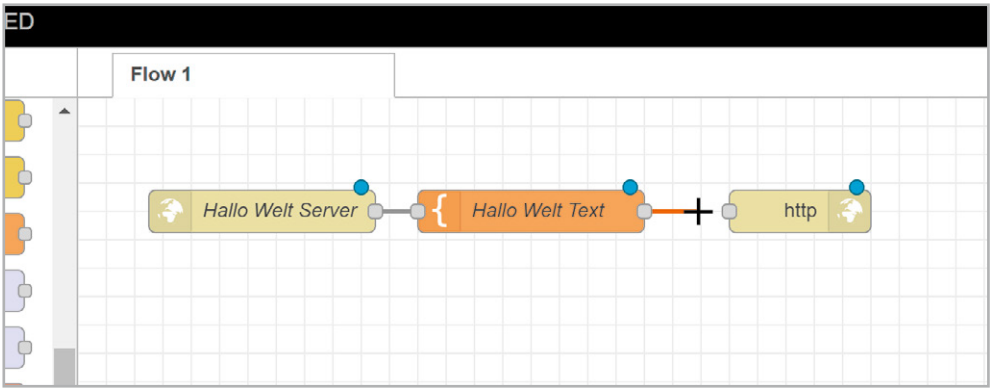


Bild 15: Verbinden der konfigurierten Nodes

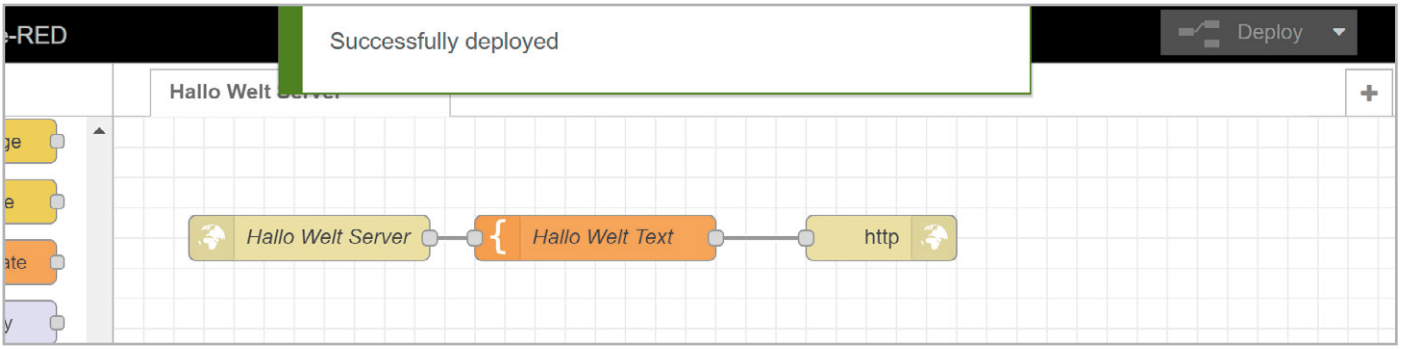


Bild 16: Erfolgreiches Deployment des Flows

Als Letztes ziehen wir einen „http response"-Node (Kategorie network) aus dem linken Fenster in die Mitte rechts neben den Template-Node. Anschließend verbinden wir die drei Nodes, indem wir jeweils auf die grauen Kästchen an den Nodes klicken und Leitungen zwischen den Nodes ziehen (Bild 15).

Um den Flow einen Namen zu geben, doppelklicken wir auf den Reiter mit der Bezeichnung Flow1 und geben „Hallo Welt Server" in das Fenster für die Einstellungen unter Name ein.

Am Ende klicken wir auf den Button Deploy rechts oben im Fenster und unser Server ist ab sofort aktiv. Ein „Successfully deployed" bestätigt, dass alle Einstellungen richtig zu sein scheinen (Bild 16).

Das Deployment bezeichnet den Vorgang, mit dem wir unsere ausgewählten und konfigurierten Nodes in Node-RED aktivieren. Dieser Vorgang muss wiederholt werden, wenn man Änderungen an allen oder einzelnen Nodes vornimmt, beispielsweise den Text in dem Template-Node ändert. Nun rufen wir in unserem Browser in einem neuen Tab die URL:

http://IP Raspberry Pi:1880/HalloWelt

auf (Bild 17).

Dies ist natürlich nur ein einfaches Beispiel für einen Server, aber mithilfe der vorhandenen Nodes kann man sich so schnell einen sehr flexiblen Server bauen, der z. B. Daten holt, verarbeitet und in veränderter Form wieder ausgibt bzw. auf bestimmte Eingaben oder Ereignisse reagiert.

Eine interessante Eigenschaft ist noch das Im- und Exportieren von ganzen Flows, das über das Node-RED-Menü möglich ist.

Wählt man Export aus, wird der Flow im JSON-Format angezeigt (Bild 18) und kann heruntergeladen oder in die Zwischenablage exportiert werden.

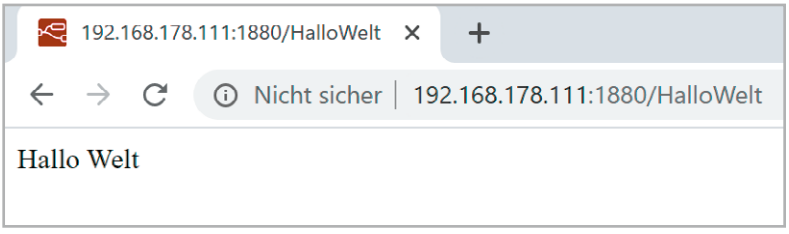


Bild 17: Aufruf des Node-RED-Servers

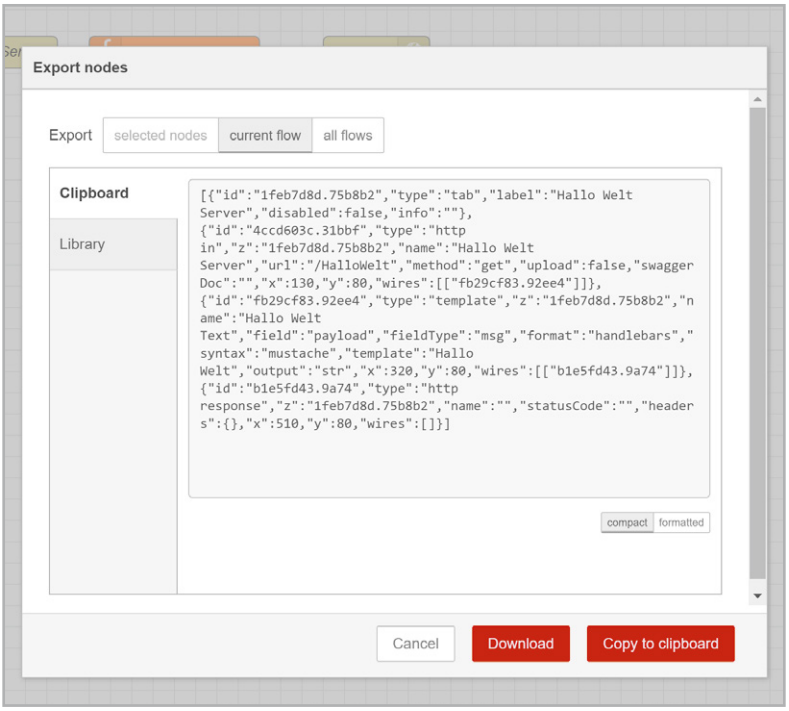


Bild 18: Über das Node-RED-Menü kann man Flows exportieren.

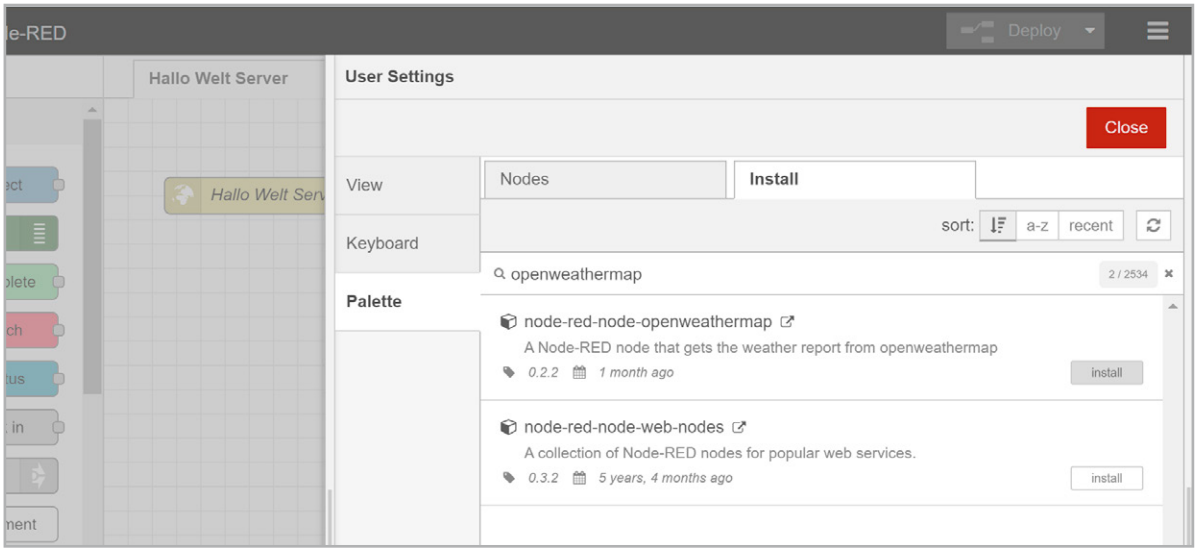


Bild 19: Suche von Nodes über den Palettenmanager

Unser „Hallo Welt Server"-Code sieht beispielsweise so aus (alle Node-RED-Beispiele aus diesem Beitrag zum Download unter [5]):

```
[{"id":"1feb7d8d.75b8b2","type":"tab","label":"H
allo Welt Server","disabled":false,"info":""},{"
id":"4ccd603c.31bbf","type":"http in","z":"1feb7
d8d.75b8b2","name":"Hallo Welt Server","url":"/
HalloWelt","method":"get","upload":false,"swagge
rDoc":"","x":130,"y":80,"wires":[["fb29cf83.92ee
4"]]},{"id":"fb29cf83.92ee4","type":"template",
"z":"1feb7d8d.75b8b2","name":"Hallo Welt Text","
field":"payload","fieldType":"msg","format":"handl
ebars","syntax":"mustache","template":"Hallo We
lt","output":"str","x":320,"y":80,"wires":[["b1
e5fd43.9a74"]]},{"id":"b1e5fd43.9a74","type":"h
ttp response","z":"1feb7d8d.75b8b2","name":"","
statusCode":"","headers":{},"x":510,"y":80,"wir
es":[]}]
```

Findet man im Internet einen interessanten Flow, kann man ihn auf die gleiche Weise importieren. Zahlreiche Beispiel-Flows findet man unter [1].

Weiteres Beispiel

Wir wollen im Folgenden ein weiteres Beispiel mit Node-RED zeigen, das uns mit der Programmierumgebung vertraut macht und Möglichkeiten zur Anwendung vorstellen. Mit dem Beitrag zur Anbindung unseres ELV Feinstaubmessgeräts PM2.5 finden Sie außerdem in diesem ELVjournal ab Seite 34 ein ausführliches Beispiel zur Anbindung über die serielle Schnittstelle und der Verwendung einer Anzeigeoberfläche – bei Node-RED als Dashboard bekannt.

Da nicht jeder diese Hardware besitzt, beschäftigen sich die nachfolgenden Beispiele rein mit Software. In Folge 2 dieser Serie zur Node-RED-Einführung werden wir dann auch Hardware wie Raspberry Pi, Arduino & Co. in die Node-RED-Umgebung einbinden.

Aktuelles Wetter und Vorhersage mit OpenWeatherMap

Das aktuelle Wetter und die Vorhersage für die kommenden Tage stehen mittlerweile über zahlreiche Websites und Apps zur Verfügung. Doch muss man diese häufig erst aufrufen, den Wohnort eingeben oder die gewünschten Daten zusammensuchen. Außerdem stehen diese Daten oft nicht für weitere Anwendungen wie der Steuerung des eigenen Smart Homes zur Verfügung.

OpenWeatherMap [7] ist ein Online-Dienst, der eine frei nutzbare Programmierschnittstelle (API) für Wetterdaten, Wettervorhersagen und historische Wetterdaten bereitstellt. Diese können wir ohne großen Aufwand in Node-RED nutzen, da es für den Dienst spezifische Nodes gibt.

Diese müssen wir zuerst in unser Node-RED-System einbinden. Dies geschieht über das Node-RED-Menü mit „Palette verwalten" (Manage palette). Im folgenden Fenster wählen wir den Reiter Install und geben in das Suchfeld openweathermap ein (Bild 19).

Nach dem Klicken auf Install und dem Beachten des Hinweises zu Informationen der Nodes werden die zu der Bibliothek gehörenden Nodes (openweathermap, openweathermap in) in Node-RED installiert.

Nach dem Schließen des Fensters sollten die neuen Nodes im linken Fenster in der Kategorie weather erscheinen. Ist dies nicht der Fall, hilft unter Umständen ein Neustart von Node-RED in einem Terminalfenster auf der Kommandozeile mit node-red-stop und anschließendem node-red-start.

Den Installationsvorgang kann man übrigens auch in dem Terminal sehen, in dem man Node-RED gestartet hat. Hier sollten sich in der Logdatei die entsprechenden Einträge zur Installation der OpenWeatherMap-Nodes wiederfinden.

Sollte die Installation aus irgendeinem Grund nicht über den Palettenmanager möglich sein, lässt sich dies ebenfalls über die Kommandozeile realisieren. Dazu wechselt man in das Node-RED-Verzeichnis mit

```
cd /home/pi/.node-red
```

und gibt hinter der Eingabeaufforderung

```
npm install node-red-node-openweathermap
```

ein.

Für den Zugriff auf die Anwenderschnittstelle (API) von OpenWeatherMap benötigt man noch einen API-Key, den wir unter

https://openweathermap.org/appid

erhalten. Hier gibt es weitere Erklärungen und Beispiele, wie man mithilfe der API an Wetterdaten kommt, sowie nützliche Links beispielsweise zur API-Dokumentation. Auf jeden Fall sollte man beachten, nicht häufiger als alle zehn Minuten Daten vom OpenWeatherMap-Server abzurufen.

Wir nutzen für unser Beispiel den inject-Node (Kategorie common), den man für zahlreiche Zwecke wie beispielsweise zeitgesteuerte Aktionen oder dem Einsteuern von Texten in einen Flow nutzen kann. In unserem Fall nutzen wir den Node lediglich zum Anstoßen des nächsten Node, dem OpenWeatherMap-Node, den wir neben den inject-Node auf die Flow-Oberfläche ziehen.

Diesen Node müssen wir noch durch Eingabe des API-Keys und des Standorts (Location) ergänzen. Abschließend ziehen wir einen Debug-Node rechts neben den OpenWeatherMap-Node, verbinden alle drei Nodes miteinander und klicken auf Deploy.

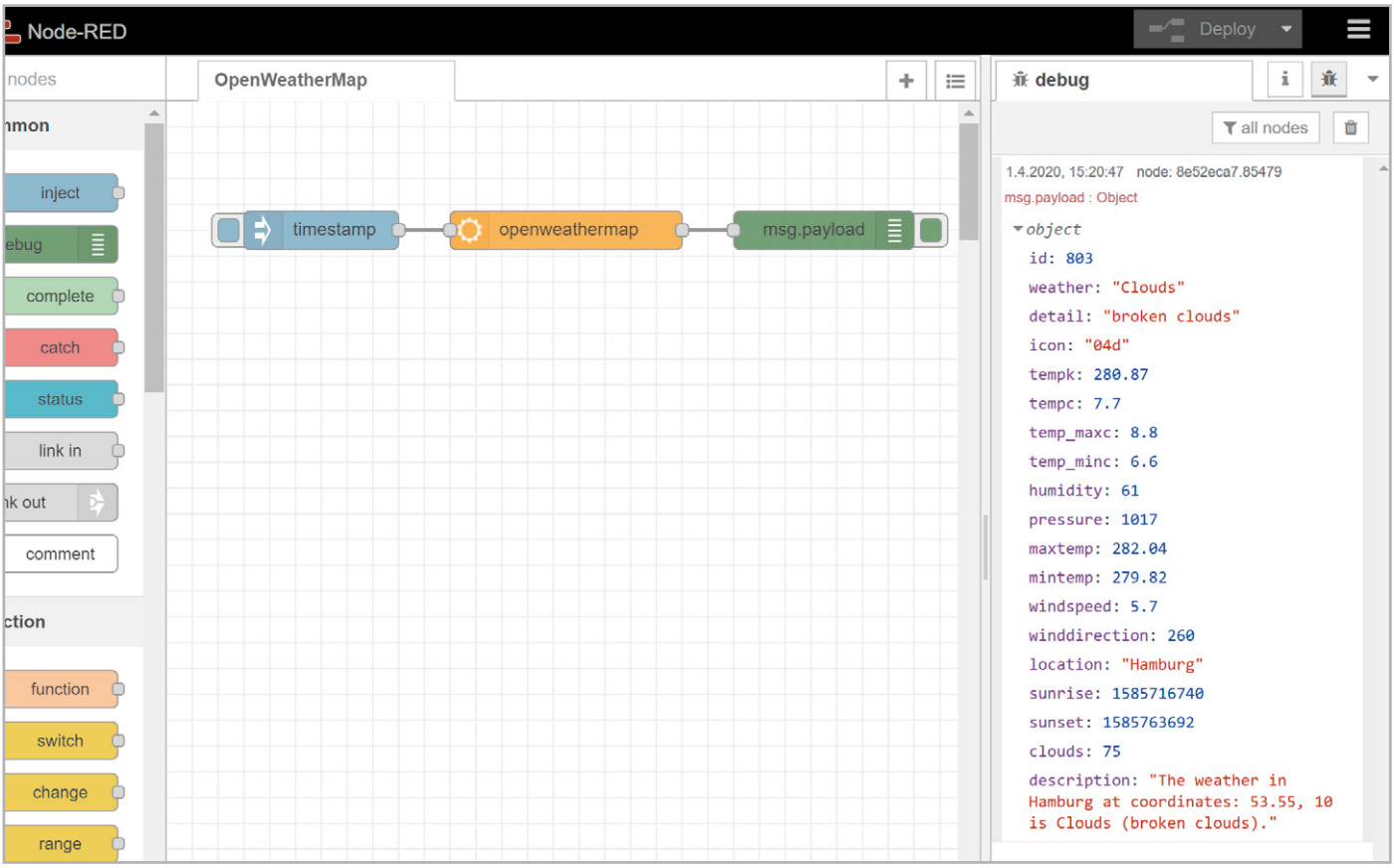


Bild 20: Wetterdaten im JSON-Format

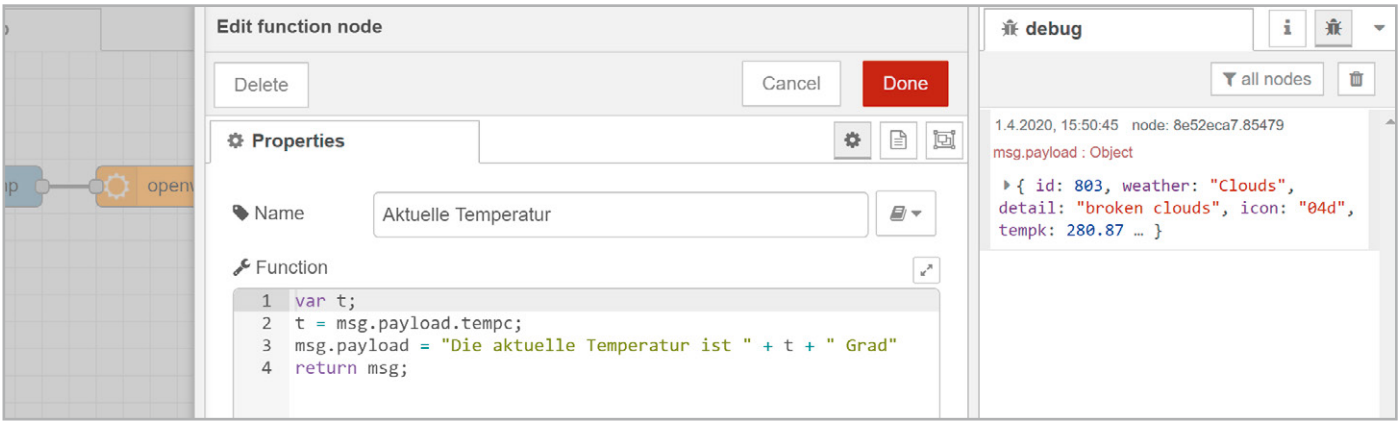


Bild 21: Einzutragender Code für den Function Node

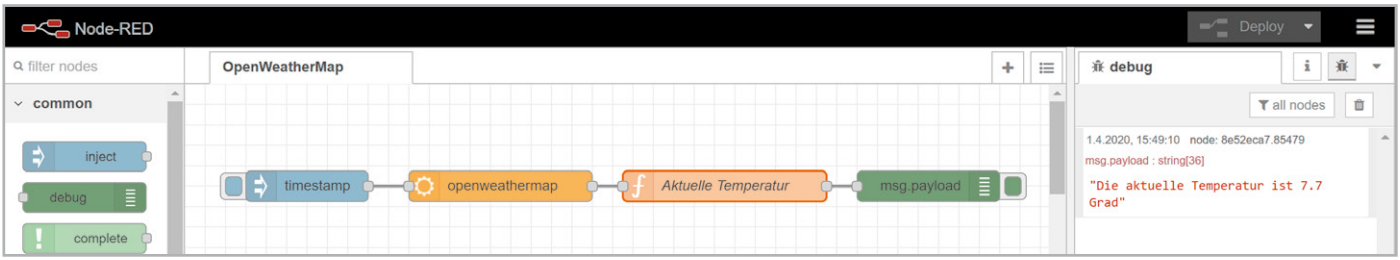


Bild 22: Der Flow zur Ausgabe der Temperatur

Im rechten Fenster klicken wir nun auf das Käfer-Symbol oder aktivieren über das Dropdown-Menü Debug messages. In diesem Fenster sind nun alle Debug-Nachrichten sichtbar.

Um den Abruf aktueller Wetterdaten für unseren gewählten Standort abzurufen, müssen wir auf das linke Fähnchen des Inject-Node klicken. Der Inject-Node schickt als Payload zwar den timestamp im Unixzeit/Epoch Format und als Datentyp number an den nächsten Node – das ist in diesem Fall aber nicht relevant, da der nächste Node mit diesem Signal lediglich angestoßen wird, ohne den Payload auszuwerten.

Ist alles richtig konfiguriert, sollten im Debug-Fenster die Wetterdaten von OpenWeatherMap im JSON-Format ausgegeben werden. Durch einen Klick auf das kleine Dreieck bei den soeben erschienenen Daten kann man das ganze JSON-Objekt anschauen (Bild 20). Mehr zum JSON-Format gibt es in diesem ELVjournal auf Seite 34 in unserem Node-RED-Beispiel für das ELV Feinstaubmessgerät PM2.5.

Wir wollen in den meisten Fällen aber nicht die gesamten Wetterdaten für unsere Anwendung wie beispielsweise die Steuerung von Smart Home Geräten nutzen. Ein interessanter Wert für viele nachgelagerte Steuerungen kann zum Beispiel die Temperatur sein. Um nur diesen Wert zu erhalten, ziehen wir uns einen Function-Node (Kategorie: function) auf die Flow-Oberfläche. Wir klicken auf die Leitung zwischen dem OpenWeatherMap- und dem Debug-Node und löschen mit der Entfernen-Taste diese Verbindung. Zwischen die beiden Nodes ziehen wir den Function-Node und verbinden alle Nodes wieder miteinander.

Nun müssen wir noch den Function-Node konfigurieren, um nur den aktuellen Temperaturwert zu erhalten. Dazu klicken wir auf den Function-Node und geben folgenden Code ein (Bild 21):

```
var t;
t = msg.payload.tempc;
msg.payload = "Die aktuelle Temperatur ist " + t + " Grad"
return msg;
```

In dem Code definieren wir zunächst eine Variable *t*, die wir im Folgenden dem aus dem JSON-Objekt extrahierten, aktuellen Temperaturwert zuordnen. Die flussgesteuerte Arbeitsweise macht sich auch hier wieder deutlich. Waren zuvor in der msg.payload noch alle von OpenWeatherMap empfangenen Wetterdaten enthalten, haben wir diese nun mit der Zuordnung unseres Textes und dem Dazwischenfügen der aktuellen Temperatur verändert. Abschließend wird das msg-Objekt von dem Function-Node an den nächsten Node per return weitergegeben.

Zugegeben – wir mussten hier erstmals etwas Code eingeben. Mit der Verwendung anderer Nodes oder Konfigurieren des Debug-Nodes hätten wir auch das umgehen können. Will man aber beispielsweise mehrere Werte abfragen und an mehrere Ausgänge ausgeben, fängt es dann schnell an unübersichtlich zu werden.

Wir deployen nach Konfiguration und dem Vergeben eines Namens für den Function-Node unseren Flow, aktivieren die Wetterabfrage über unseren Inject-Node und bekommen im Debug Fenster nur noch die Ausgabe der aktuellen Temperatur mit dem Text, den wir in dem Function-Node definiert haben (Bild 22).

Auch dieser Flow ist unter [6] als Download erhältlich. Nur noch der API-Key und der Standort müssen dem OpenWeatherMap-Node konfiguriert werden

Ausblick

Mit unserem „Hallo Welt"-Server und den weiteren Beispielen aus diesem Beitrag haben wir bisher nur an der Oberfläche der Möglichkeiten von Node-RED gekratzt. Trotzdem ist es offensichtlich, wie man mit wenigen, einfachen Schritten zu einem Ergebnis kommt, das in anderen Programmiersprachen nur mit deutlich höherem Aufwand erreicht worden wäre.

Node-RED ist dabei besonders für das Prototyping geeignet – wenn man schnell mit wenig Aufwand zu einem soliden Ergebnis kommen möchte. Die Bandbreite der Anwendungen ist nur durch die vorhandenen Nodes begrenzt. Programmierer können sich zudem eigene Nodes erstellen und in Node-RED nutzen.

Im nächsten Teil beschäftigen wir uns mit dem Anschluss von Raspberry Pi, Arduino, ESP32 und elektronischen Bauteilen, die man in Node-RED einbinden kann. ELV

Weitere Beiträge zu Node-RED:

Teil 2: Node-RED und Elektronik – Einbinden von Raspberry Pi, ESP32, Arduino und Elektronik-Bauteilen in die Node-RED-Oberfläche

Teil 3: Node-RED unter der Haube

Weitere Infos:

[1] Node-RED Library: https://flows.nodered.org/

[2] Raspberry Pi 4 Model B, 2 GB RAM im ELVshop: de.elv.com: Bestell-Nr. 250569

[3] Raspberry Pi Imager bei der Raspberry Pi Foundation: www.raspberrypi.org/downloads/

[4] Installation von Node-RED auf dem Raspberry Pi: nodered.org/docs/getting-started/raspberrypi

[5] Download Node-RED Beispiel-Flows: de.elv.com: Webcode #10316

[6] OpenWeatherMap: https://openweathermap.org

Alle Links finden Sie auch online unter de.elv.com/elvjournal-links