Programmierung mit Node-RED

Inhalt

1 Ziele

In diesem Modul lernen Sie praktische Erfahrungen mit visueller Programmierung durch Node-RED kennen. Damit können Anwendungsfälle im Bereich des Internets der Dinge (kurz IoT) in kurzer Zeit umgesetzt werden. Mit Hilfe eines grafischen Entwicklungswerkzeugs werden IoT-Anwendungen mit vordefinierten Bausteinen realisiert.

2 Installation

Für die Installation öffnen Sie den Link: Running on Windows : Node-RED (node-red.org). Die Installation ist wie folgt unterteilt:

- Install *Node.js*,
- Install *Node-RED* und
- Run *Node-RED*.

3 Einleitung: Node-RED

Node-RED ist ein Werkzeug für visuelle Programmierung. Die Programm-Umsetzung erfolgt mit einem Browser-basierten Editor (kurz *Flow-Editor*), der in vier Bedienfelder (engl. Panel) unterteilt ist:

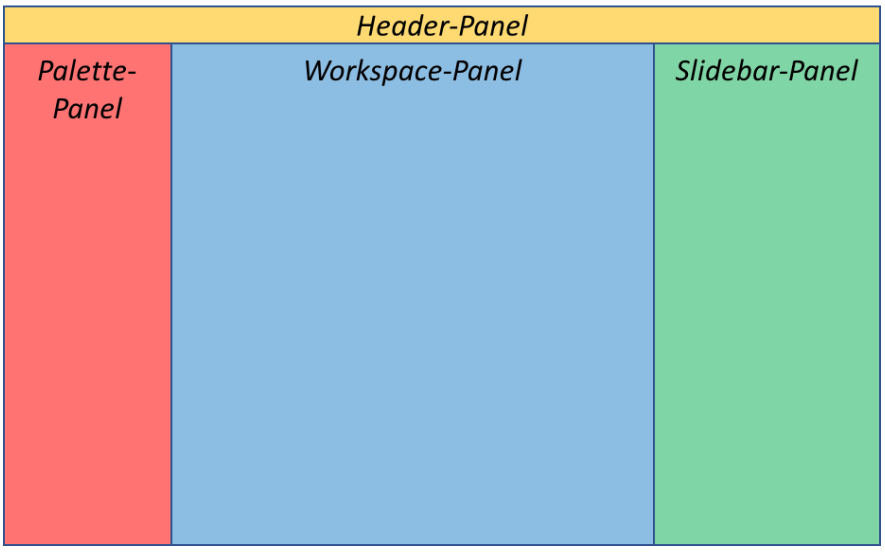


Abbildung 1: Bedienfelder des *Flow-Editors*

Palette-Panel

Im *Palette-Panel* stehen nach dem Baukastenprinzip eine Reihe von vordefinierten *Funktionsbausteine* (engl. *Nodes*) zur Verfügung.

Die relativ grosse Auswahl an mitgelieferten Bausteinen deckt viele der gängigsten Anwendungsfälle ab. Zudem gibt es auch Funktionsbausteine, in denen man **JavaScript (JS)** nutzen kann und somit viele Möglichkeiten offen hat. Falls dies nicht ausreichen sollte, bietet sich die Möglichkeit, eigene Bausteine ebenfalls in JavaScript zu schreiben, um eine gewünschte Funktionalität abzubilden.

Workspace-Panel

Das *Workspace-Panel* ist der Arbeitsbereich des *Flow-Editors*: Mittels Drag-and-drop lassen sich vom *Palette-Panel* einfach *Nodes* auswählen und im Arbeitsbereich anordnen. Durch *Flows* (siehe Abbildung 3) werden Verbindungen zwischen Bausteinen erstellt und so die gewünschten Informationsflüsse in ein Programm umgesetzt.

Header-Panel

Im *Header-Panel* befinden sich das ☰ Haupt-Menü (siehe Abbildung 2) und ein deploy *Deploy-Button* für die Programm-Ausführung.

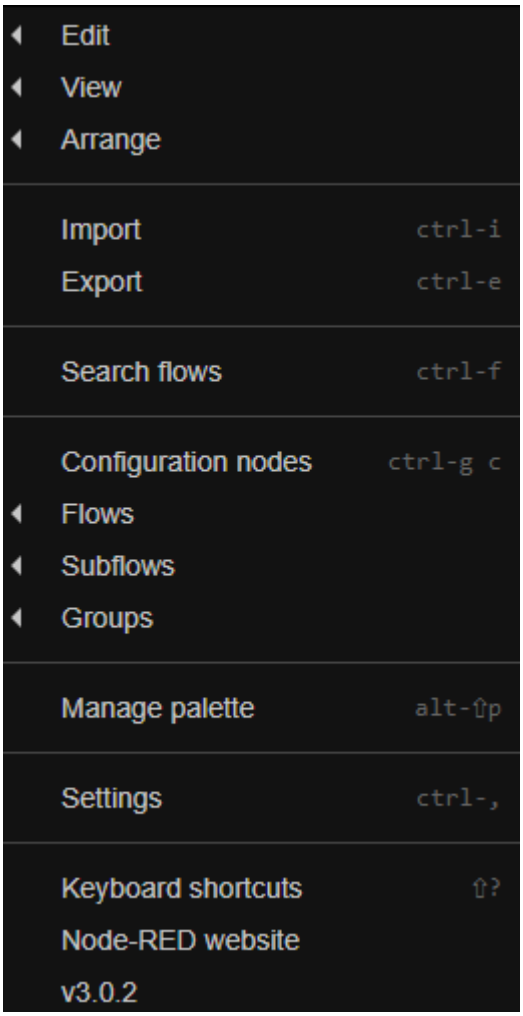


Abbildung 2: Haupt-Menü

Sidebar-Panel

Im *Sidebar-Panel* stehen in Registern Werkzeuge zur Umsetzung eines *Flows* bereit:

- Das Register *Node-Information* stellt die Eigenschaften und Informationen über den gewählten *Flow* respektive *Node* zur Verfügung.
- Das Register *Help* gibt Hilfe über den ausgewählten *Node*.
- Das Register *Debugnachrichten* dient zur Ausgabe der Debug-Nachrichten.
- Das Register *Konfigurations-Node* zeigt die möglichen Konfigurationseinstellungen eines Nodes an.

3.1 Flows, Nodes und Nachrichten

Flows werden im *Workspace-Panel* in *Registern* erstellt und bestehen aus *Nodes*, die über *Ausgangs-* und *Eingangs-Ports* (siehe Abbildung 3) interaktiv miteinander verbunden

werden können. Ein *Node* besteht inhaltlich einerseits aus **JavaScript-** und andererseits aus **HTML-Code**. Der **HTML-Code** bestimmt, wie der *Node* im *Workspace-Panel* visualisiert und konfiguriert werden kann und welche *Info* dem *Node* beigefügt wird.

In Node-RED wird ein Nachrichten-Objekt als *msg* (engl. message) bezeichnet und weist standardmässig drei Eigenschaften auf: *msg.payload*, *msg.topic* und *msg.msgid*.

Die Eigenschaft *msg.payload* enthält die Nutzdaten einer Nachricht: Das kann beispielsweise der Messwert eines Sensors sein. *msg.topic* ist ein benutzerdefinierter String, der es ermöglicht, die Nachricht zu kategorisieren. *msg.msgid* beinhaltet die Kennung der Nachricht. *Nodes* können dem Nachrichten-Objekt weitere Eigenschaften hinzufügen. Die folgende Anweisung stellt die Zuweisung eines Nachrichten-Objekts in **JavaScript** dar:

```
var msg = {payload: "Nutzdaten",
           topic: "IoT-Nugget",
           msgid: "5eb31483.66a65c"};
```

Nodes arbeiten mit einer Eingangsnachricht und generieren eine oder mehrere Ausgangsnachrichten, wobei diese über Ports weitergeleitet werden (siehe Abbildung 3). Das folgende Beispiel zeigt wie ein *Flow* zu den Eingangs-Nutzdaten **„Hello“** die Zeichenkette **„ World“** hinzufügt und die Ausgangs-Nutzdaten für die Ausgabe bereitstellt:

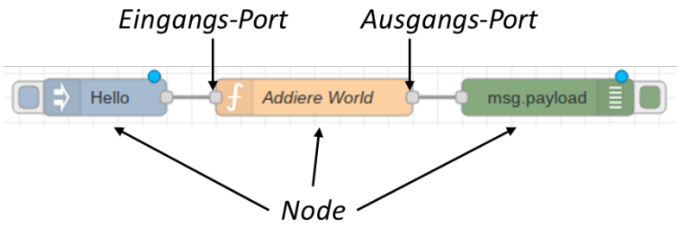


Abbildung 3: *Flow* mit „Hello World” als Ausgabe

3.2 Input, Output und Funktion

Input

Input Nodes beliefern einen *Flow* mit Daten. Diese Eingaben können einer HTTP-Verbindung, einem Twitter-Account, etc. resultieren. *Input Nodes* weisen mindestens einen Ausgangs-Port auf. Die folgende Abbildung zeigt exemplarisch den *Inject Node*, der in regelmässigen Abständen automatisch oder durch Auslösen eines Events Nachrichten versendet.

Abbildung 4: *Inject Node*

Output

Output Nodes senden Daten von einem *Flow* an andere Dienste, wie Tweets, eine TCP-Verbindung, etc. *Output Nodes* haben mindestens einen Eingangs-Port. Die Abbildung 5 stellt exemplarisch einen *Debug Node* dar, der Nachrichten an das Fenster *Debugging* (siehe Register *Debug)* sendet und ausgibt:

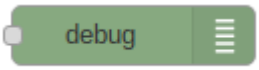


Abbildung 5: *Debug Node*

Funktion

Function Nodes verarbeiten eingehende Daten. Diese haben einen Eingang und mindestens einen Ausgang. *Function Nodes* erlauben eigene Codeteile zu schreiben, Zahlenbereiche zu verändern, Datentypen umzuwandeln, etc.

Abbildung 6: *Function Node*

3.3 Beschreibung der wichtigsten Nodes

Nodes sind im *Palette-Panel* in Kategorien eingeteilt. Einige wie *common*, *function*, *network* oder *parser* sind vorinstalliert. Weitere, wie beispielsweise *dashboard* lassen sich über das ☰ *Haupt-Menü/Manage Palette* installieren.

Kategorie	Funktion
common	Hier stehen sowohl Input- als auch Output-Nodes zum **Auslösen**, **Empfangen** oder zur weiteren **Verarbeitung** von **Nachrichten** zur Verfügung. Beispiele sind *Inject*- oder *Debug-Nodes*.
function	Es stehen Funktionale Nodes (eng. Function Nodes), die eine **Logik** ausführen zur Verfügung. Beispiele sind Nodes vom Type *function*, *switch* oder *template*.

network	Hier stehen Input- als auch Output-Nodes wie *http*, *websocket* oder *mqtt* für die **Netzwerkkommunikation** zur Auswahl.
parser	Hier stehen Nodes zur Konvertierung von Informationen in verschiedene Formate zur Verfügung. Vertreter sind *csv*, *html* oder *xml*. So ist beispielsweise mit Node *csv* eine Konvertierung zwischen einer CSV-formatierten Zeichenfolge und ihrer JavaScript-Objektdarstellung möglich.
dashboard	Es werden eine Reihe von Nodes zur Erstellung eines Dashboards zur Verfügung gestellt. Vertreter sind beispielsweise *button*, *text*, *gauge* oder *chart*. Hinweis: Wenn Sie die Kategorie **dashboard** verwenden, so ist diese zusätzlich zu installieren: Öffnen Sie im ☰ Haupt-Menü *Manage palette* und wählen Sie das Register *Install* aus. Geben Sie unter 🔍 *search modules* **node-red-dashboard** ein und klicken Sie auf install Button.

Tabelle 1: Übersicht Node-RED Kategorien

3.4 Node-RED starten

Machen Sie sich nun mit dem *Flow-Editor* vertraut:

1. Öffnen Sie ▶ Eingabeaufforderung: Geben Sie den Befehl *node-red* ein, um den *Node-Red-Server* zu starten

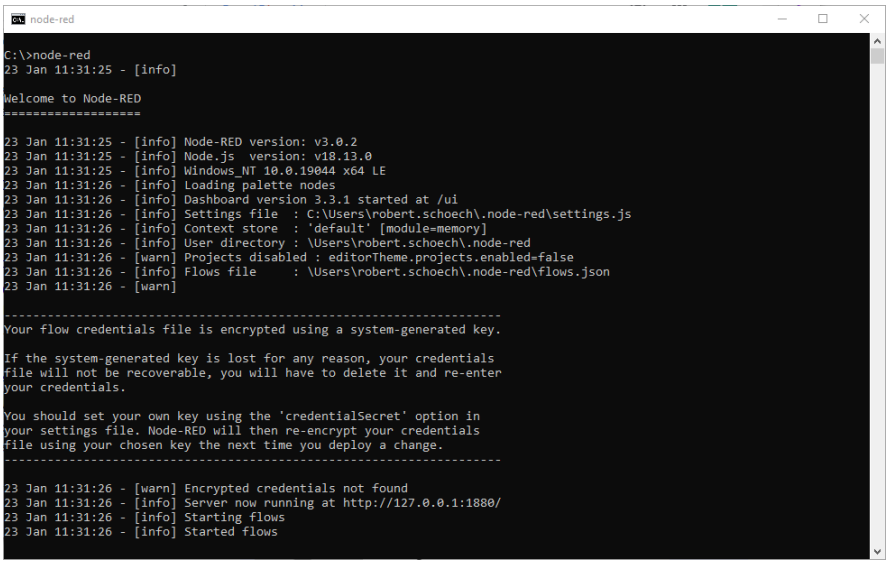


Abbildung 7: Start des *Node-RED-Servers*

Hinweis: Die *Node-RED console* lässt sich ebenfalls im Eingabeaufforderung mit dem Befehl *node-red-start* öffnen.

2. Öffnen Sie den *Flow-Editor* im Webbrowser mit der Adresse `http://127.0.0.1:1880` oder `localhost:1880`.

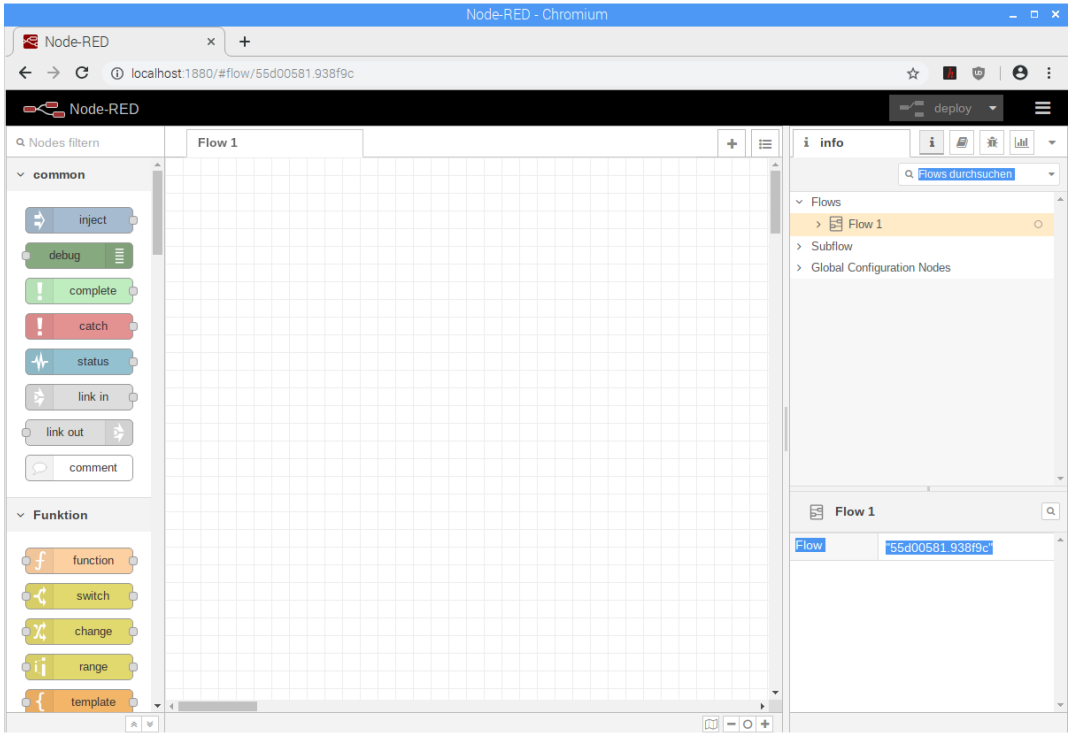


Abbildung 8: *Flow-Editor*

3. Bestimmen Sie die einzelnen Bedienfelder (siehe Abbildung 1) und erkunden Sie deren Funktionalität.

4 Aufgabe 1: Hello World

In dieser Aufgabe lernen Sie das Hallo-Welt-Programm mit einem *Input-*, *Output-* und *Funktions-Node* umzusetzen.

4.1 Implementierung

1. Starten Sie die Entwicklungsumgebung für Node-RED und öffnen Sie anschliessend den *Flow-Editor* im Webbrowser mit der Adresse `localhost:1880`.

2. Benennen Sie im *Workspace-Panel* das Register *Flow 1* auf *Hello World* um.

   Hinweis: Sie können das Fenster «Edit flow: Flow 1» einerseits im *Haupt-Menü/Flows/Rename* anderseits durch Doppelklick direkt im Register *Flow 1* öffnen.

3. Wählen Sie im *Palette-Panel* den Node *inject* aus und legen Sie mittels Drag-and-drop eine Instanz im Flow *Hello World* an. Öffnen Sie in der Instanz mit Doppelklick das Eigenschaftsfenster (engl. Properties). Wählen Sie im Zuweisungsteil unter der Eigenschaft *msg.payload* den Zuweisungstyp *String* aus. Schreiben Sie anschliessend in das Zuweisungs-Textfeld *Hello* hinein. Schliessen Sie das Fenster wieder mit Button Done **Done**.

   Hinweis: Benützen Sie zur Suche der einzelnen Nodes die Filter-Funktion im *Palette-Panel*.

4. Legen Sie weitere *Nodes* namens *function* und *debug* per Drag-and-drop in *Hello World* an, und anschliessend verbinden Sie die Ausgangs- und Eingangs-Ports miteinander:

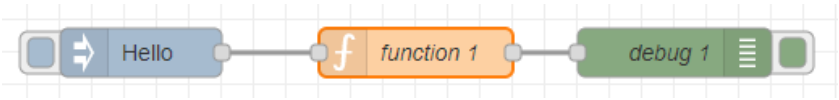
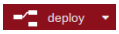


Abbildung 9: Nodes Verbindung

5. Öffnen Sie nun in Node *function* mit Doppelklick das Eigenschaftsfenster und weisen Sie der Eigenschaft Name *World* hinzu. Fügen Sie im Register **On Message** der Eingangsnachricht `msg.payload` den String *World* durch folgende Anweisungen (engl. Statements) hinzu:

   ```
   msg.payload += " World";
   return msg;
   ```

   Schliessen Sie danach das Fenster wieder mit Done **Done**.

6. Mit Auswahl 

Deploy stellen Sie *Hello World* für die Programmausführung bereit. Nach erfolgreicher Bereitstellung erscheint folgende Bestätigung:

Abbildung 10: Nachricht nach erfolgreichem Deployment

Hinweis: Wenn Sie einen Flow für die Programmausführung bereitstellen, entspricht dies dem Starten eines Node.js-Servers. Dieser wartet auf die Abarbeitung von Ereignissen (siehe unter Punkt 8: Nachrichten-Event auslösen).

7. Um die Eingangsnachricht vom *Node msg.payload* anzuzeigen, ist im *Slidebar-Panel* das Register *Debug messages* auszuwählen.

8. Lösen Sie mit Klick im „Injektionsbereich“ von *Node Hello* (siehe Abbildung 11, gelb markierter Bereich) ein Nachrichten-Event aus.

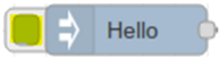


Abbildung 11: Bereich für das Auslösen einer Nachricht

Was wird nun im Fenster **debug** angezeigt?

4.2 Lessons Learned

In dieser Aufgabe lernten Sie den Node-RED Editor anhand des Hallo-Welt-Programms kennen und zu bedienen.

5 Aufgabe 2: Web-Server mit Licht-Steuerung

In dieser Aufgabe schalten Sie über einen HTTP-Request mit Query-String das Licht ein oder aus. Im Beispiel wird das Leuchtmittel durch einen Node repräsentiert.

5.1 Implementierung

1. Starten Sie die Entwicklungsumgebung für Node-RED und öffnen Sie anschliessend den *Flow-Editor* im Webbrowser mit der Adresse `localhost:1880`.

2. Öffnen Sie im *Haupt-Menü/Import* das Fenster «Import nodes» und wählen Sie mit *select a file to import* aus dem Projektordner *nodeRED* die Projektvorlage *Light WebServer.json* aus.

   Anschliessend beenden Sie den Vorgang mit **Import**.

3. Stellen Sie *Light WebServer* mit Auswahl **Deploy** für die Programmausführung bereit: Der Node *WebServer* vom Typ *http in* kann nun HTTP-Requests entgegennehmen. Im Node *Generate Page* vom Typ template wird die Antwort für den Request erzeugt und im Node *Response* vom Typ *http response* wird die Beantwortung der Anfrage abgeschlossen.

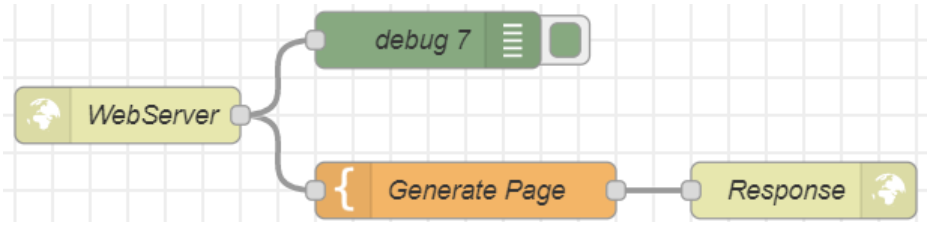


Abbildung 12: Instanz eines Web-Servers

4. Wählen Sie im *Slidebar-Panel* das Register *Debug messages* aus.

5. Testen Sie mit dem Webbrowser den *Light WebServer*, indem Sie folgende Requests ausführen:

   `http://localhost:1880/light?state=0` und
   `http://localhost:1880/light?state=1`.

6. Welche Informationen werden im Fenster **debug** angezeigt sowie welcher Inhalt wird als Response im Webbrowser angezeigt?

7. Vervollständigen Sie nun die Simulation der **Licht-Anbindung** mit dem *WebServer*: Um den Wert *0* oder *1* von *state* zu extrahieren, welcher wiederum Teil vom Eingangsobjekt *msg.payload* vom *WebServer* ist, bedarf es dem sogenannten Node *change*. Wählen Sie deshalb im *Palette-Panel* den Node *change* und legen

Sie eine Instanz im Flow *Flow WebServer* an. Öffnen Sie in der Instanz mit Doppelklick das Eigenschaftsfenster und definieren Sie für Lichtsteuerung folgende Extrahierungsregel:

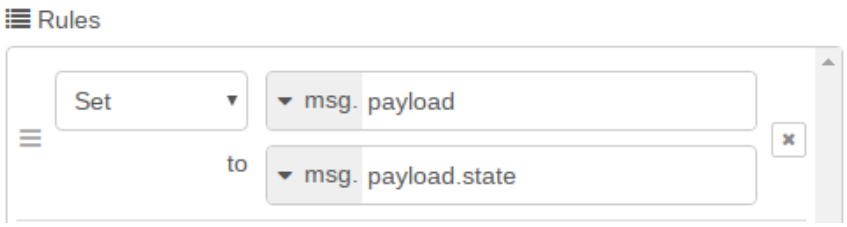


Abbildung 13: Extrahierungsregel im Node *set msg.payload* des Typs *change*

Schliessen Sie danach das Fenster wieder mit Done **Done**.

8. Verbinden Sie nun die Ausgangs- und Eingangs-Ports miteinander:

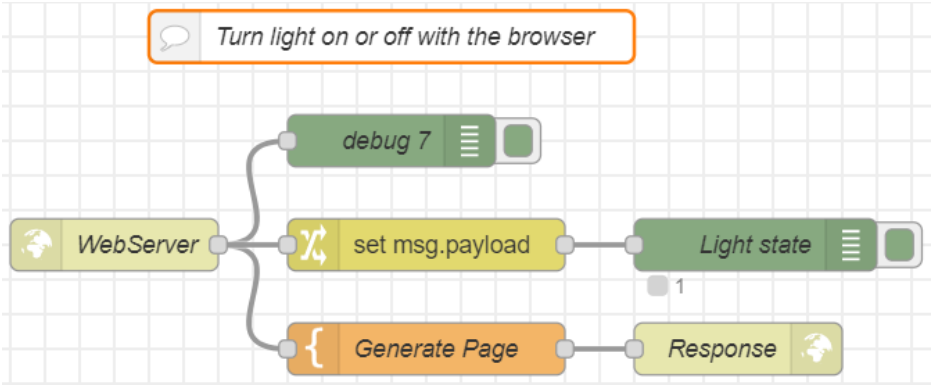


Abbildung 14: Light WebServer

9. Stellen Sie den *LED WebServer* mit Auswahl deploy **Deploy** für die Programmausführung bereit. Testen Sie erneut die Requests mit dem Webbrowser.
10. Welcher Zustand wird im Node *Light state* simuliert, welche Informationen werden im Fenster **debug** und welcher Inhalt wird als Response im Webbrowser angezeigt?
11. **Optional**: Befinden sich Leuchten wie beispielsweise Philips HUE in Ihrem Arbeitsumfeld respektive Netzwerk, so können Sie nach dem gleichen Muster das Licht von Philips HUE lights mit dem Webbrowser ein- und ausschalten.

5.2 Lessons Learned

In dieser Aufgabe lernten Sie HTTP-Requests mit Query-Strings mit einem Netzwerk Node zu verarbeiten. Nach dem Deployment des Flows *Light WebServer* lässt sich das Licht mit einem Webbrowser über das Netzwerk ein- und ausschalten: Die simulierte

Licht-Anbindung lässt mit smarten Leuchtsystemen wie beispielsweise Philips HUE lights für den Echtbetrieb realisieren.

6 Aufgabe 3a: MeteoSchweiz - Wetterstation

In dieser Aufgabe erstellen Sie eine Wetterstation basierend auf den Messdaten von MeteoSchweiz, Station St. Gallen: Neben dem Zeitstempel sollen die Temperatur, Windgeschwindigkeit und -richtung, Luftdruck und -feuchtigkeit abgebildet werden.

6.1 Installation Node MeteoSchweiz

1. Öffnen Sie im Haupt-Menü *Manage pallette* und wählen Sie das Register *Install* aus. Geben Sie unter *search modules* **node-red-contrib-swissforecast** ein und klicken Sie auf install Button.
2. Schliessen Sie anschliessend das Fenster mit Close **Close**. Nun können Sie im *Palette-Panel* den installierten Node von MeteoSchweiz benützen.

Abbildung 15: Node von MeteoSchweiz

6.2 Implementierung

1. Starten Sie die Entwicklungsumgebung für Node-RED und öffnen Sie anschliessend den *Flow-Editor* im Webbrowser mit der Adresse `localhost:1880`.
2. Öffnen Sie im *Haupt-Menü/Import* das Fenster «Import nodes» und wählen Sie mit *select a file to import* aus dem Projektordner *nodeRED* die Projektvorlage *Weather Station.json* aus.

   Anschliessend beenden Sie den Vorgang mit Import **Import**.
3. Stellen Sie *Weather Station* mit Auswahl deploy **Deploy** für die Programmausführung bereit: Der Node *STG* vom Typ *swissforecast* ist sendebereit:

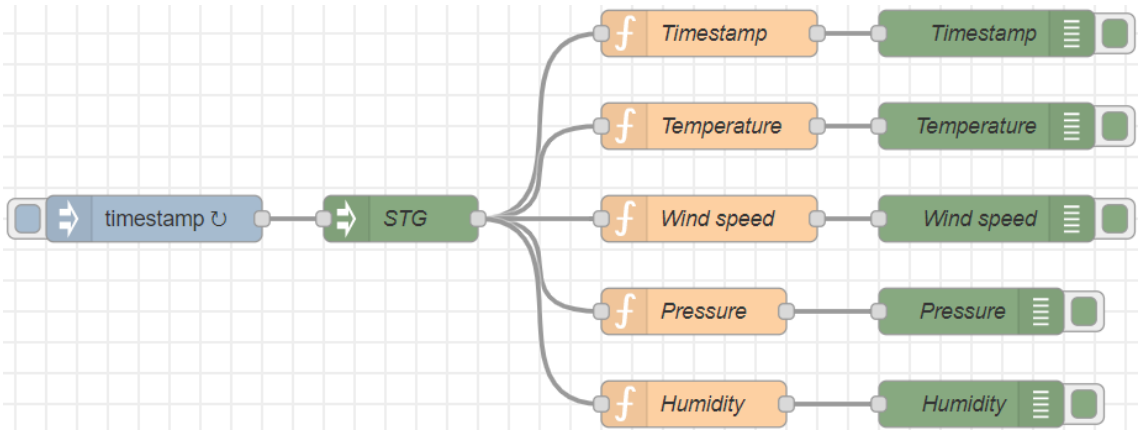


Abbildung 16: Wetterstation St. Gallen

4. Für eine Messdatenabfrage klicken Sie auf *timestamp*. Anschliessend studieren Sie die Ergebnisse im Fenster **debug**: Die Messdaten werden im JSON-Format gesendet. Welche Objekte finden Sie vor?

5. Erweitern Sie die Wetterstation und visualisieren Sie die Windrichtung (*payload.windDirection*) inklusive dem Einheitssymbol °.

Abbildung 17: Nodes für Windrichtung

6. Öffnen Sie nun in Node *function* mit Doppelklick das Eigenschaftsfenster und weisen Sie der Eigenschaft Name *Wind direction* hinzu. Schreiben Sie im Register **On Message** folgende Anweisungen (engl. Statements):

```
msg.payload = msg.payload.windDirection + " °";
return msg;
```

   Schliessen Sie danach das Fenster wieder mit Done **Done**.

7. Stellen Sie die Erweiterung mit Auswahl deploy **Deploy** für die Programmausführung bereit und führen Sie erneut eine Messdatenabfrage durch.

8. Es sollen nun die Messdaten regelmässig aktualisiert werden: Öffnen Sie dazu im Node *timestamp* das Eigenschaftsfenster. Wählen Sie unter Eigenschaft *Repeat* den Wert *Interval*. Definieren Sie eine Updaterate von 10 Minuten. Schliessen Sie das Fenster wieder mit Done **Done**.

9. Stellen Sie die Änderung mit Auswahl deploy **Deploy** für die Programmausführung bereit und lösen Sie erneut eine Messdatenabfrage aus.

10. Öffnen Sie die Liste der Wetterstationen von MeteoSchweiz und wählen Sie eine für Sie passende Station aus (Hinweis: Zur besseren Orientierung öffnen Sie die Karte mit dem Layer der Wetterstationen).

11. Anschliessend öffnen Sie im Projekt *Weather Station* den Node *STG*. Tragen Sie unter Eigenschaft *Location* die Abkürzung (siehe Wetterstationstabelle, Spalte **ABK.**) von der ausgewählten Station ein.

12. Stellen Sie die geänderte Wetterstation mit Auswahl deploy **Deploy** für die Programmausführung bereit lösen Sie zur Überprüfung erneut eine Messdatenabfrage aus.

6.3 Lessons Learned

In dieser Aufgabe lernten Sie empfangene (Wetter)daten im JSON-Format auszuwerten und für die Anzeige einzelner Objektdaten mit Informationen wie Einheitsbezeichnung zu ergänzen. Hier liesse sich die Wetterstation durch ihre eigene Wetterstation wie beispielsweise von Homematic ersetzen.

7 Aufgabe 3b: Hausassistent

In dieser Aufgabe nutzen Sie Messdaten von der Wetterstation St. Gallen für den Hausassistenten: So soll bei bereits bei geringer Windgeschwindigkeit (z.B. Schwellwert von 11 km/h) das Fenster geschlossen und das manuelle Öffnen verriegelt werden. Bei normaler Windgeschwindigkeit wird die Verriegelung zurückgesetzt und das manuelle Öffnen wieder ermöglicht.

7.1 Implementierung

1. Starten Sie die Entwicklungsumgebung für Node-RED und öffnen Sie anschliessend den *Flow-Editor* im Webbrowser mit der Adresse `localhost:1880`.

2. Öffnen Sie im *Haupt-Menü/Import* das Fenster «Import nodes» und wählen Sie mit *select a file to import* aus dem Projektordner *nodeRED* die Projektvorlage *Home Assistent.json* aus.

   Anschliessend beenden Sie den Vorgang mit import **Import**.

3. Stellen Sie *Home Assistent* mit Auswahl deploy **Deploy** für die Programmausführung bereit: Der Node *STG* vom Typ *swissforecast* ist sendebereit. Für eine Abfrage der Windgeschwindigkeit klicken Sie auf *timestamp*.

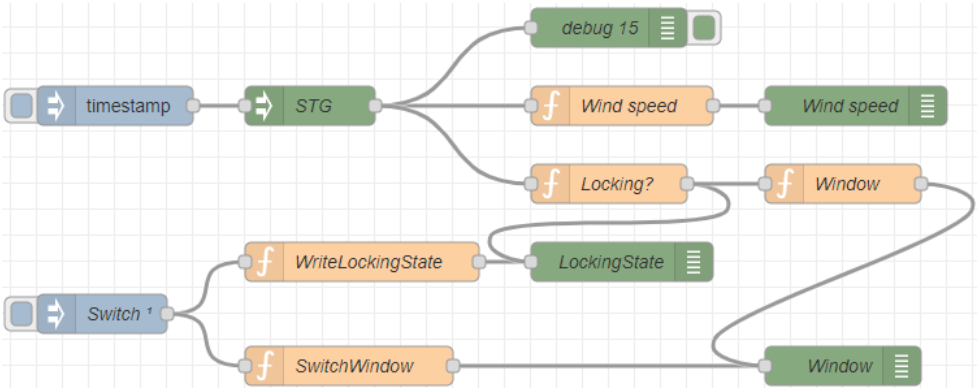


Abbildung 18: Hausassistent

4. Implementieren Sie die Logik für das **manuelle Öffnen und Schliessen des Fensters**. Öffnen Sie dazu in Node *SwitchWindow* mit Doppelklick das Eigenschaftsfenster und fügen Sie im Register **On Message** folgende Anweisungen hinzu:

```
// get state of window and locking
var windowState = flow.get('window');
var unlocked = !flow.get('locked');
// if unlocked and closed window
if (unlocked && windowState == "closed") {
   windowState = "open"
} else {
   windowState = "closed"
}
flow.set('window', windowState);
msg.payload = windowState;
return msg;
```

   Für die Speicherung des Verriegelungs- sowie Fensterzustandes verwenden Sie Variablen *locked* and *window*. Diese Variablen sind in allen Nodes von *Home Assistent* bekannt und sind im Node *SwitchWindow* unter Register **On Start** initialisiert (weitere Details siehe Storing data). Interpretieren Sie die Logik der Anweisung *if-else*.

   Schliessen Sie danach das Fenster wieder mit Done **Done**.

5. Implementieren Sie die **Verriegelung** für das manuelle Öffnen und Schliessen des Fensters ab einer Windgeschwindigkeit von 11 km/h. Öffnen Sie dazu in Node *Locking?* mit Doppelklick das Eigenschaftsfenster und fügen Sie im Register **On Message** folgende Anweisungen hinzu:

```
if (msg.payload.windSpeed > 11) {
   // lock switch window
   flow.set('locked', 1);
} else {
   // unlock switch window
   flow.set('locked', 0);
}
```



```
msg.payload = flow.get('locked');
return msg;
```

Schliessen Sie danach das Fenster wieder mit Done **Done**.

6. Als nächstens soll das Fenster im verriegelten Zustand **automatisch geschlossen** werden. Öffnen Sie dazu in Node *Window* mit Doppelklick das Eigenschaftsfenster und fügen Sie im Register **On Message** folgende Anweisungen hinzu:

```
// get state of window
var windowState = flow.get('window');
// if locked and window is open
if (msg.payload == 1 && windowState == "open") {
   windowState = "closed";
   flow.set('window', windowState);
}
msg.payload = windowState;
return msg;
```

Schliessen Sie danach das Fenster wieder mit Done **Done**.

7. Überlegen Sie sich alle möglichen Testfälle und führen diese anschliessend aus. Hinweis: Verändern Sie für die Tests die Abfrage der Windgeschwindigkeit so ab, dass die möglichen Verriegelungszustände eintreffen.

7.2 Lessons Learned

In dieser Aufgabe nutzen Sie Messungen der Windgeschwindigkeiten von der Wetterstation St. Gallen für den Hausassistenten, mit der ein automatisches Schliessen der Fenster und gleichzeitiges Verriegeln für das manuelle Öffnen bei geringer Windgeschwindigkeit auslöst Verriegelung der Fenster. Neu ist dabei das Speichern von Datenzuständen wie beispielsweise Verriegelungszustand (Beschreibung siehe Storing data) über den *Flow* hinweg.

8 Aufgabe 4: Freie Parkplätze in der Stadt St. Gallen (PLS)

Lernen Sie die Datensätze von öffentlichen Datenportalen (engl. Open Data Portal) zu nutzen. So stellt beispielsweise die Stadt St. Gallen einen Datensatz über die freien Parkplätze der Stadt St. Gallen (kurz PLS) zur Verfügung. Neben dem eigenen Standort sollen die Parkplatzdaten auf einer Karte visualisiert werden.

8.1 Installation Node Worldmap

1. Öffnen Sie im ☰ Haupt-Menü *Manage pallette* und wählen Sie das Register *Install* aus. Geben Sie unter 🔍 *search modules* **node-red-contrib-web-worldmap** ein und klicken Sie auf install Button.

2. Schliessen Sie anschliessend das Fenster mit Close **Close**. Nun können Sie im *Palette-Panel* den installierten Node von Worldmap für die Kartenvisualisierung benützen.

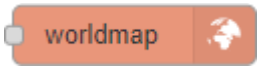


Abbildung 19: Node von Worldmap

8.2 Implementierung

1. Starten Sie die Entwicklungsumgebung für Node-RED und öffnen Sie anschliessend den *Flow-Editor* im Webbrowser mit der Adresse `localhost:1880`.

2. Öffnen Sie im ☰ *Haupt-Menü/Import* das Fenster «Import nodes» und wählen Sie mit *select a file to import* aus dem Projektordner *nodeRED* die Projektvorlage *PLS.json* aus.

   Anschliessend beenden Sie den Vorgang mit Import **Import**.

3. Stellen Sie *PLS* mit Auswahl deploy **Deploy** für die Programmausführung bereit: Für eine Abfrage der OST-Standorte klicken Sie auf *timestamp*. Anschliessend öffnen Sie im Webbrowser die Karte mit der Adresse `localhost:1880/worldmap`.

4. Abbildung 20: Darstellung von Worldmap

   Wechseln Sie auf der Karte zwischen den verschiedenen Layer hin und her und suchen Sie OST-Standorte. Welche Informationen werden von jedem OST-Standort angezeigt?

5. Bestimmen Sie ihren eigenen Standort: Für die Suche der Koordinaten können Sie das Werkzeug Latitude and Longitude Finder on Map Get Coordinates (lat-long.net) verwenden. Öffnen Sie anschliessend im Projekt den Node *My location* und tragen Sie den Breiten- und Längengrad sowie ihren eigenen Namen ein:

```
msg.payload = [
   {
      lat:47.34,
      lon:9.7,
      label: "My location",
      icon: "fa-male",
      name: "Max Mustermann",
   }
];
```

6. Schliessen Sie das Fenster «Edit function node»mit Done **Done**. Anschliessend verbinden Sie den Ausgangs- und Eingangs-Port von *My location* und *worldmap* miteinander.

7. Stellen Sie *PLS* mit Auswahl deploy **Deploy** für die Programmausführung bereit, klicken Sie auf *timestamp* und öffnen Sie im Webbrowser die Karte *worldmap*. Suchen Sie ihren Standort auf der Karte.

8. Erkunden Sie Datensatzbeschreibung über die freien Parkplätze der Stadt St. Gallen. Suchen Sie in der Beschreibung die URL für die Generierung der Daten und vergleichen Sie die URL mit jener in Node *http request*. Anschliessend suchen Sie in der Beschreibung nach *records* (Datensätze). Welche Attribute sind in einem *record* definiert? Anschliessend öffnen Sie Node PLS und studieren Sie die einzelnen Code-Anweisungen.

   Was bedeutet die Anweisung `records.forEach`?

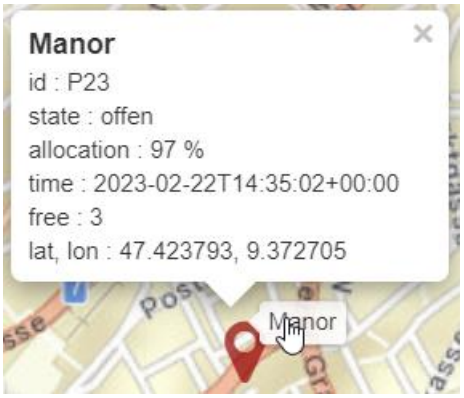
```
// read records and assign each record value
// to list for visualizing in worldmap
var records = Object.values(msg.payload.records);
var list = [];
records.forEach((val) => {
     var obj = {
     id: val.fields.phid,
     state: val.fields.phstate,
     label: val.fields.phname,
     name: val.fields.phname,
     allocation: val.fields.belegung_prozent + "%",
     time: val.fields.zeitpunkt,
     free: val.fields.shortfree,
     lat: val.fields.standort[0],
     lon: val.fields.standort[1],
     icon: "fa-map-marker fa-3x"
```



```
        } // assign to object properties
        list.push(obj);
    })
    msg.payload = list;
```

9. Schliessen Sie das Fenster «Edit function node»mit Done **Done**. Anschliessend verbinden Sie den Ausgangs- und Eingangs-Port von *PLS* und *worldmap* miteinander.

10. Stellen Sie *PLS* mit Auswahl deploy **Deploy** für die Programmausführung bereit, klicken Sie auf *timestamp* und öffnen Sie im Webbrowser die Karte *worldmap*. Suchen Sie die verschiedenen Parkplätze und ermitteln Sie die aktuelle Auslastung.



11. Abbildung 21: Darstellung von Worldmap

8.3 Lessons Learned

In dieser Aufgabe lernten Sie öffentliche Datensätze mit *http request* zu übertragen. Unter opendata.swiss können weitere öffentliche Datensätze finden und diese ebenfalls mit *http request* abfragen.

9 Zusammenfassung

In Node-RED lassen sich IoT-Anwendungen in kurzer Zeit umsetzen, sprich Anwendungen können mit Webdiensten durch visuelle Programmierung mittels Nodes über Port-Verbindungen verknüpft werden. Da Node-RED auf der JavaScript Plattform Node.js basiert, lassen sich Nodes an spezielle Aufgabenstellungen durch vordefinierte Blöcke mit hinterlegten Funktionen mittels JavaScript anpassen: So stehen etwa im Node *function* *On Start*, *On Message* und *On Stop* für Anpassungen mittels JavaScript zur Verfügung.

