Cyber Physical Systems

Wintersemester 2020/21

Hochschule München University of Applied Sciences

Lektion 04: Node Red installieren

von: Max Holzmann

Node Red wird verwendet, um die unterschiedlichen Technologien miteinander zu verknüpfen und stellt somit den gemeinsamen Nenner dar. Im ersten Teil geht es ausschließlich um die Installation. Im zweiten Teil wird das gesammte Programm in Node Red erklärt. Es wird auch auf Technologien eingegangen, welche erst in den folgenden Lektionen detaillierter beschrieben werden. Passende Verweise zum Nachlesen in den Lektionen werden angegeben.

Ziele

- Einrichten von Node Red auf dem Raspberry
- Erklärungen des Node Red Programms. Besonders interessant, da das Node Red das Herzstück der Rallye-Anwendung ist.

Voraussetzungen und erforderliches Equipment

Die Tutorial Lektion 01: Grundlegende Einstellungen und Installationen muss erledigt sein link

- Hardware
  - 1x Raspberry Pi 3 B

Lösungsschritte

1. Installation von Node Red über das Terminal (link zur Node Red Doku)

   ```
   sudo apt-get install nodered
   ```

   oder

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

2. Starten von Node Red

   ```
   node-red-start
   ```

3. Testen, ob Node Red läuft. Am Einfachsten über {IP_OF_PI}:1880 den Webserver von Node Red öffnen
4. Um Node Red zu stoppen, kann der folgende Befehl ausgeführt werden

```
node-red-stop
```

5. Um Node Red bei jedem Start automatisch zu starten, kann mit dem folgenden Befehl Node Red als Service gestartet werden.

```
sudo systemctl enable nodered.service
```

6. Für die Rallye Anwendung müssen Zusatzbibliotheken installiert werden. Dies erfolgt über den Paletten-Manager oder über die Konsole.

- Eine Bibliothek für MQTT, welche dynamisch aus der msg den Topic verwendet

```
npm install node-red-contrib-mqtt-dynamic
```

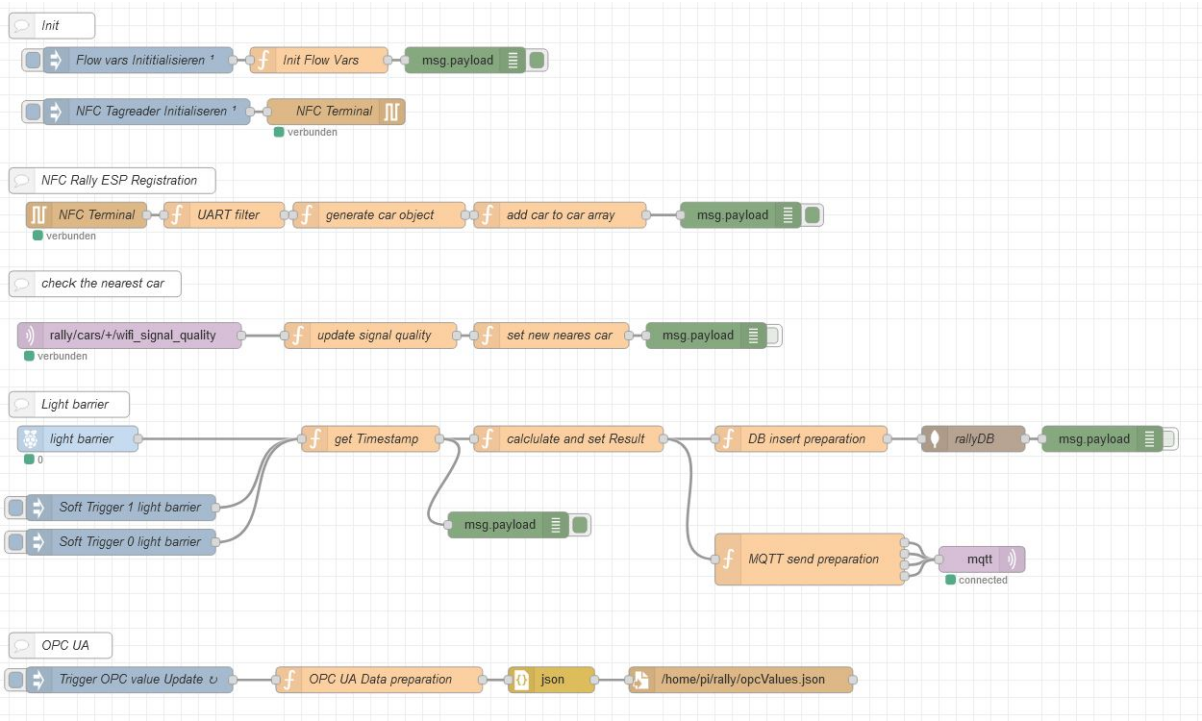
- Eine Bibliothek zur Verbindung zu MongoDBs <= V2.4

```
npm install node-red-contrib-mongodb2
```

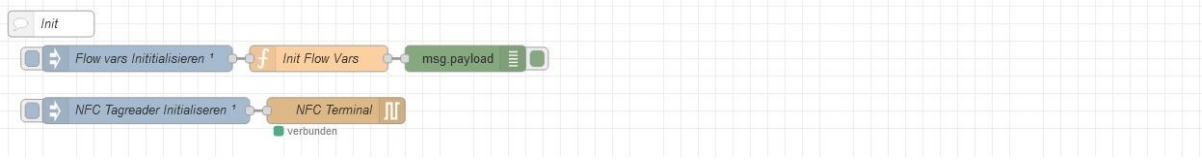
Erklärung Rallye Node Red Programm

Node Red bietet durch die Knoten(Nodes)-Struktur die Möglichkeit das Programm in beliebig viele, kleine Unterknoten aufzuteilen. Dadurch kann man übersichtliche und leicht verständliche Knoten schreiben. Aus diesem Grund verweise ich darauf, dass es zum Verständnis der Anwendung hilfreich ist, sich die einzelnen Nodes in Node red direkt anzusehen.

Übersicht über alle Nodes:

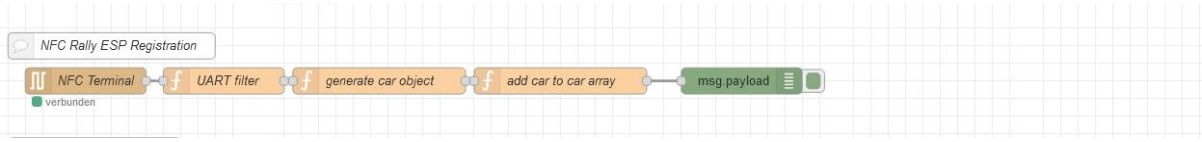


Details Init



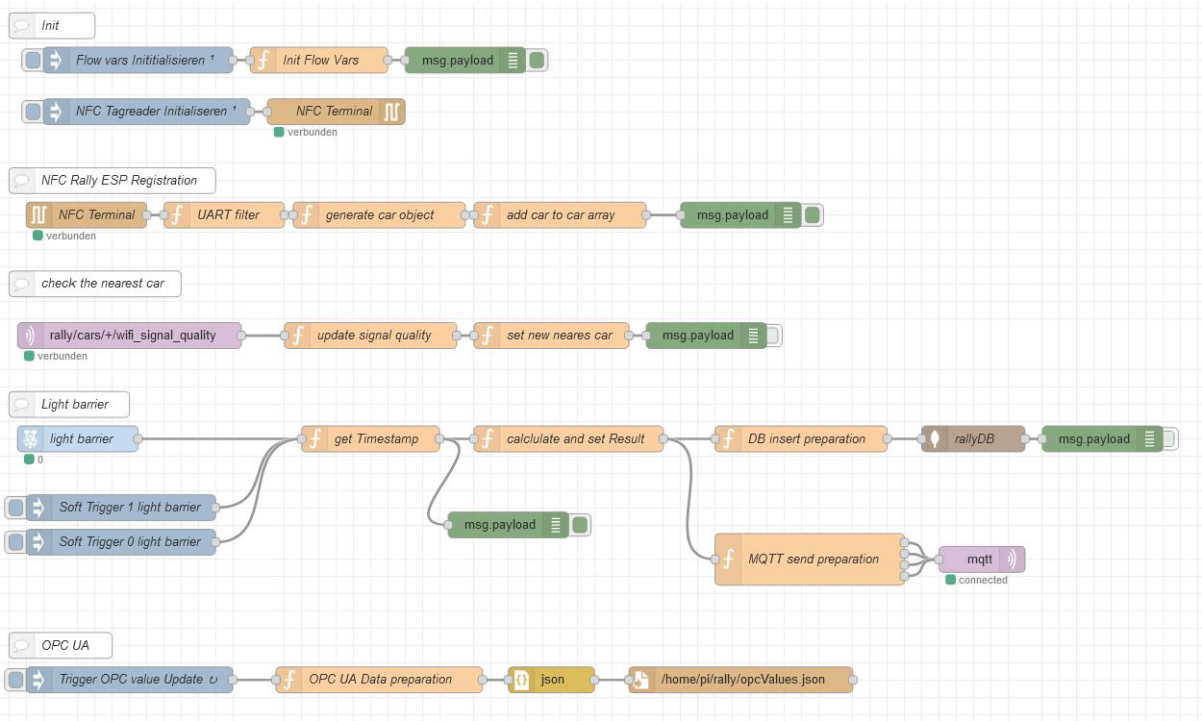
- Die erste Node Reihe wird immer beim Neustart von Node Red einmalig ausgeführt. In der Funktion *Init Flow vars* werden für den Bereich dieses Flows die Variablen *cars* und *current_carID* erzeugt und initialisiert. *cars* ist ein Array, welches für jeden aktuell registrierten RallyESP ein Objekt speichert. Das Objekt beinhaltet die Daten zu diesem RallyESP. Die Variable *current_carID* speichert den Index des aktuell am Nächsten zur Lichtschranke befindlichen RallyESP.
- In der zweiten Node Zeile wird der Startbefehl über die serielle Schnittstelle an das NFC device gesendet. Diese Node Kette wird ebenfalls einmal zu Beginn von Node Red automatisch ausgeführt. Detailles hierzu in der Lektion 05: NFC link zur Lektion

Details NFC Registration



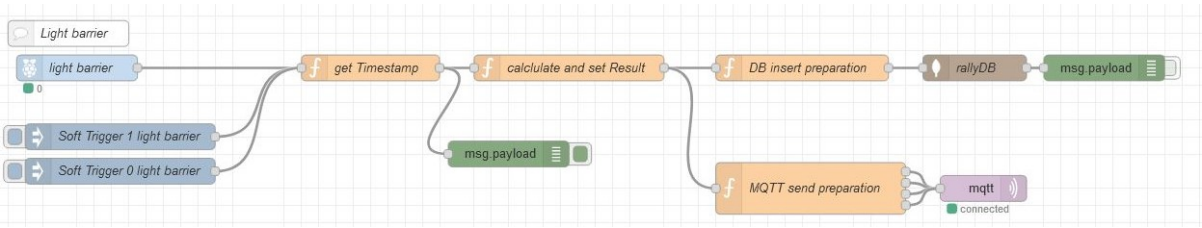
- *NFC Terminal*: Das NFC Device registriert einen RallyESP und teilt ihm einen eindeutigen Topic mit. Über den NFC Terminal Node wird Node Red auch dieser Topic zur Verarbeitung übergeben.
- *UART Filter*: Alle UART Nachrichten, welche keine Topic Nachricht enthalten werden raus gefiltert.
- *generate car Object*: Aus dem übergebenen Topic wird ein car Objekt erzeugt mit den Datenfeldern für den Topic, die Messungsnummer, die Messdifferenz, die Durchfahrtszeit und den Status des Buzzers.
- *add to car array*: Dieser Node fügt das car Objekt in das flowweite *cars* Array ein.

Details Check the nearest Car



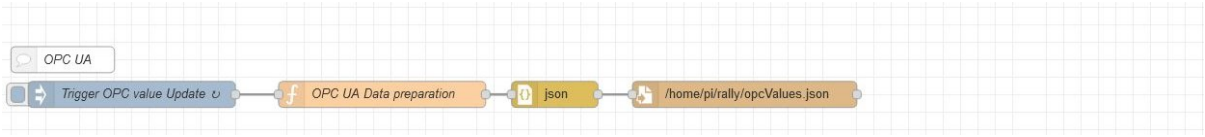
- *MQTT Node signal quality*: Jede Sekunde senden alle verbundenen RallyESPs ihre Verbindungsqualität zum WLAN Accespoint über MQTT. Dieser Node subscribed alle signal quality Topics und gibt die Nachricht weiter.
- *update signal quality*: Dieser Node updatet, anhand der eintreffenden MQTT Nachricht, das signal quality Datenfeld des passenden car objects im *cars Array*.
- *set new nearest car*: car in *cars* mit der besten Signalqualität wird gesucht und die Variable *current_carID* wird neu gesetzt.

Details Light barrier



- *light barrier*: Node überwacht den GPIO Pin an dem die Lichtschranke angeschlossen ist und sendet eine Nachricht bei State wechseln.
- *get Timestamp*: Holt den aktuellen Zeitstempel der Durchfahrt, wenn die Lichtschanke ausgelöst hat.
- *calculate Result*: Dieser Node berechnet das Durchfahrtsergebnis und updatet das car Objekt in *cars*.
- *DB insert Preparation*: Vorbereitung des Messergebnisses, um in die Datenbank geschrieben zu werden.
- *rallyDB*: Ausführen des Insert Befehls für die MongoDB, um Messergebnisse abzuspeichern.
- *MQTT send Preparation*: Vorbereiten der Nachricht mit passendem MQTT topic Attribut für den dynamic MQTT Node.
- *mqtt*: Publishen der Ergebniswerte an den RallyESP, welcher durch die Lichtschranke gefahren ist.

Details OPC UA



- Nodekette, welche die für den OPC UA Server relevanten Werte aus den Flowvariablen in einer json Datei ablegt. Auf diese Datei kann der OPC Server zugreifen.

Quellen

[1] steves-internet-guide.com