

# **Bachelor-Thesis an der Hochschule Luzern - Technik & Architektur**

<b>Titel</b>	<b>Node-RED für Toradex Embedded-Linux-Module</b>
<b>Diplomandin/Diplomand</b>	<b>Williner, Sandro</b>
<b>Bachelor-Studiengang</b>	<b>Bachelor Elektrotechnik und Informationstechnologie</b>
<b>Semester</b>	<b>FS22</b>
<b>Dozentin/Dozent</b>	<b>Kasten, Oliver</b>
<b>Expertin/Experte</b>	<b>Wegmüller, Marc</b>

## **Abstract Deutsch**

Die vorliegende Arbeit zeigt die Entwicklung von Node Red spezifischen Bausteine für System on Module der Firma Toradex. Die Bausteine ermöglichen eine Ansteuerung der GPIO und dem Auslesen von CPU Informationen. Node Red ermöglicht Benutzern einen visuelle Zugang zu Programmierabläufe. Die zur Erstellung benützten Softwarecodes sind aufgezeigt und können reproduziert werden. Die fertig erstellten Bausteine dienen als Grundlage, um weitere zu erstellen. Die ganze Anwendung wurde in einem Container entwickelt und ein entsprechendes Image ist auf Dockerhub verfügbar.

## **Abstract Englisch**

The present work shows the development of node-red nodes for a specific system on module from the company Toradex. The node gives a possibility to control the GPIO and read CPU information. Node-red provides the user with visual access to programmable logic. The source code for the development of the nodes is shown, explained, and can be reproduced. The finish nodes can be used to develop more specific nodes. The application was built in a container and the image is online on Dockerhub available

Ort, Datum Schenkon, 10.06.2022  
© **Sandro Williner, Hochschule Luzern – Technik & Architektur**

# Inhalt

<b>1</b>	<b>Einleitung</b>	<b>1</b>
<b>2</b>	<b>Grundlagen</b>	<b>2</b>
2.1	Node Red . . . . .	2
2.2	Node Red Bausteine . . . . .	3
2.2.1	Java Script . . . . .	3
2.2.2	HTML . . . . .	4
2.2.3	JSON . . . . .	4
2.3	Toradex . . . . .	4
2.3.1	Container . . . . .	4
2.4	Aktuelle Situation . . . . .	6
<b>3</b>	<b>Methodik</b>	<b>7</b>
3.1	Double Diamond . . . . .	7
3.2	Projektziel Toradex . . . . .	8
3.3	Abgrenzung und Ziele BAT . . . . .	8
3.3.1	Meilenstein 1 Inbetriebnahme Carrier Board . . . . .	8
3.3.2	Meilenstein 2 Funktionsbausteine CPU und GPIO . . . . .	9
3.3.3	Meilenstein 3 Funktionsbausteine CPU und PWM . . . . .	10
<b>4</b>	<b>Softwarebeschreibung</b>	<b>11</b>
4.1	CPU Temperature . . . . .	11
4.1.1	HTML . . . . .	12
4.1.2	JavaScript . . . . .	14
4.2	CPU Usage . . . . .	15
4.2.1	HTML . . . . .	15
4.2.2	Java Script . . . . .	17
4.3	GPIO setzen . . . . .	18
4.3.1	HTML . . . . .	19
4.3.2	Java Script . . . . .	21
4.4	GPIO lesen . . . . .	22
4.4.1	HTML . . . . .	23
4.4.2	Java Script . . . . .	24
4.5	Upload . . . . .	24
4.5.1	Vorschriften Node Red . . . . .	24

4.5.2	JSON . . . . .	25
4.5.3	Upload Vorgang . . . . .	26
4.6	Container . . . . .	26
<b>5</b>	<b>Anwendung</b>	<b>29</b>
5.1	Container Aufsetzen . . . . .	29
5.1.1	Container Aufsetzen mit Portainer . . . . .	29
5.1.2	Container aufsetzen Kommandozeile . . . . .	32
5.2	Hinzufügen der Nodes . . . . .	33
5.2.1	Node Hinzufügen Weboberfläche . . . . .	33
5.2.2	Node Hinzufügen Kommandozeile . . . . .	35
5.3	Demo-Anwendung . . . . .	36
<b>6</b>	<b>Lösungsevaluation</b>	<b>39</b>
6.1	Benutzertests . . . . .	39
6.1.1	GPIO Baustein Ansteuerung . . . . .	39
6.1.2	Komfort der GPIO Bausteine . . . . .	40
6.2	Zielerreichung anhand BAT Ziele . . . . .	42
<b>7</b>	<b>Schlussdiskussion</b>	<b>43</b>
7.1	Erfahrungen . . . . .	43
7.2	Offene Punkte . . . . .	43
<b>A</b>	<b>Anhang</b>	<b>I</b>
A.1	Aufgabenstellung . . . . .	I
A.2	Meilensteine . . . . .	III
A.3	Benutzertest mit Notizen . . . . .	VI

# 1 Einleitung

Die Informatikbranche boomt seit längerem und ein weiteres starkes Wachstum ist in Ausblick. Deshalb fehlen in immer mehr Unternehmen Arbeitnehmer mit fundierten Programmierkenntnissen. Aus diesem Grund gibt es stetig mehr Anwendungen die versuchen, den Menschen einen vereinfachten Zugang zur Hardware zu ermöglichen.

Die vorliegende Arbeit ist im Rahmen vom Modul BAT der Hochschule Luzern und befasst sich mit dem Ziel Benutzern einen vereinfachten Hardwarezugang zu gewähren. Dieser soll per Node Red realisiert werden. Im Mittelpunkt der Arbeit steht daher die Entwicklung hardwarespezifischer Bausteine für ein System on Module von der Firma Toradex mit Sitz in Horw. Es ist ein Unternehmen im Embeddey-System-Bereich und bietet diverse Hardware an.

Ausgangspunkt sind die Aufgabenstellung und die vereinbarten Zielen, welche die Arbeit abgrenzen. Zur Erreichung dieser Ziele werden Bausteine entwickelt, mit Nutzern getestet und angepasst.

Zunächst wird im Kapitel zwei auf die notwendigen Grundlagen eingegangen. Dies ermöglicht dem Leser einen ersten Überblick und schliesst grosse Wissenslücken. Danach werden die angewandten Methoden aufgezeigt. Daraus resultiert ein Anforderungskatalog, welcher das Erreichen der Ziele evaluierbar machen. Im Kapitel vier wird die Software beschrieben, welche die Bausteine definiert. Zusätzlich finden sich darin Informationen zum Upload Vorgang und einen erstellten Container. Darauf folgend wird die konkrete Anwendung der Bausteine gezeigt und das Aufsetzen der notwendigen Umgebung. Im Kapitel sechs werden die Bausteine evaluiert und auf ihre Korrektheit kontrolliert. Zum Abschluss sind persönliche Erfahrungen und offene Punkte beschrieben.

## 2 Grundlagen

Im nachfolgenden Kapitel werden Grundlagen erläutert, um das verstehen der Arbeit zu gewährleisten. Insbesondere wird auf die einzelnen Programmiersprachen eingegangen und erklärt, was Node Red ist. Weiter wird der Industriepartner und aktuelle Situation vorgestellt.

### 2.1 Node Red

Node Red ist eine visuelle Oberfläche welche einem ermöglicht, gewisse Programme (Flows) zu erstellen. Diese Flows werden dann jeweils ausgeführt. Somit erhalten Anwender einen visuellen Zugang zur Programmierung, ohne fundierte Programmierkenntnisse. Die web-basierte Oberfläche von Node Red ist in Abbildung 1 zu sehen.

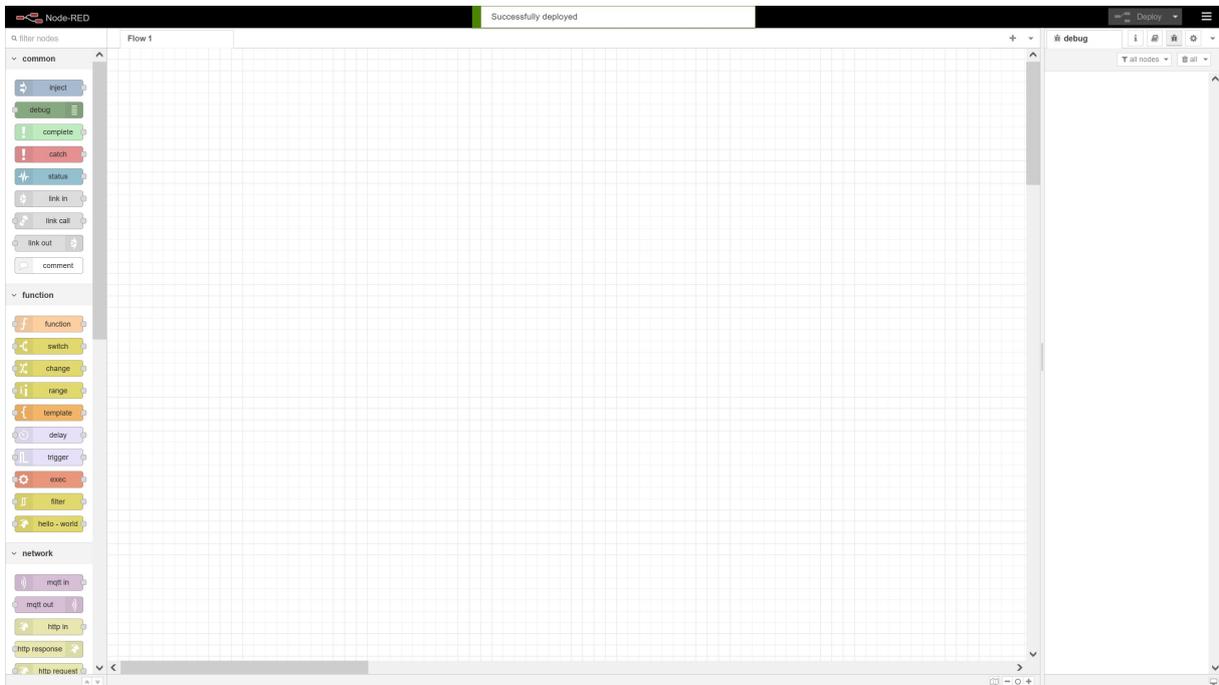


Abbildung 1: Node Red Oberfläche

Auf der linken Seite sind die verfügbaren Bausteine, um einen Flow zu generieren. Diese werden im Kapitel 2.2 beschrieben und erklärt. In der Mitte des Bildes ist die Flow Oberfläche. Hier wird mittels Drag and Drop ein Baustein platziert und die Flows erstellt. Auf der rechten Seite werden Hilfestellungen über Bausteine angezeigt und es lassen sich Ausgaben von Flows überprüfen, Bausteine editieren und vieles mehr.

Ein einfacher “Hello World Flow“ kann aussehen, wie in Abbildung 2. Nach dem zusammenstellen und verbinden der Bausteine muss per deploy Button, oben rechts im Bild, der Flow erstellt werden. Durch betätigen des inject Buttons wird “Hello World“ auszugeben. Auf der rechten Seite ist die entsprechende Ausgabe zu sehen.

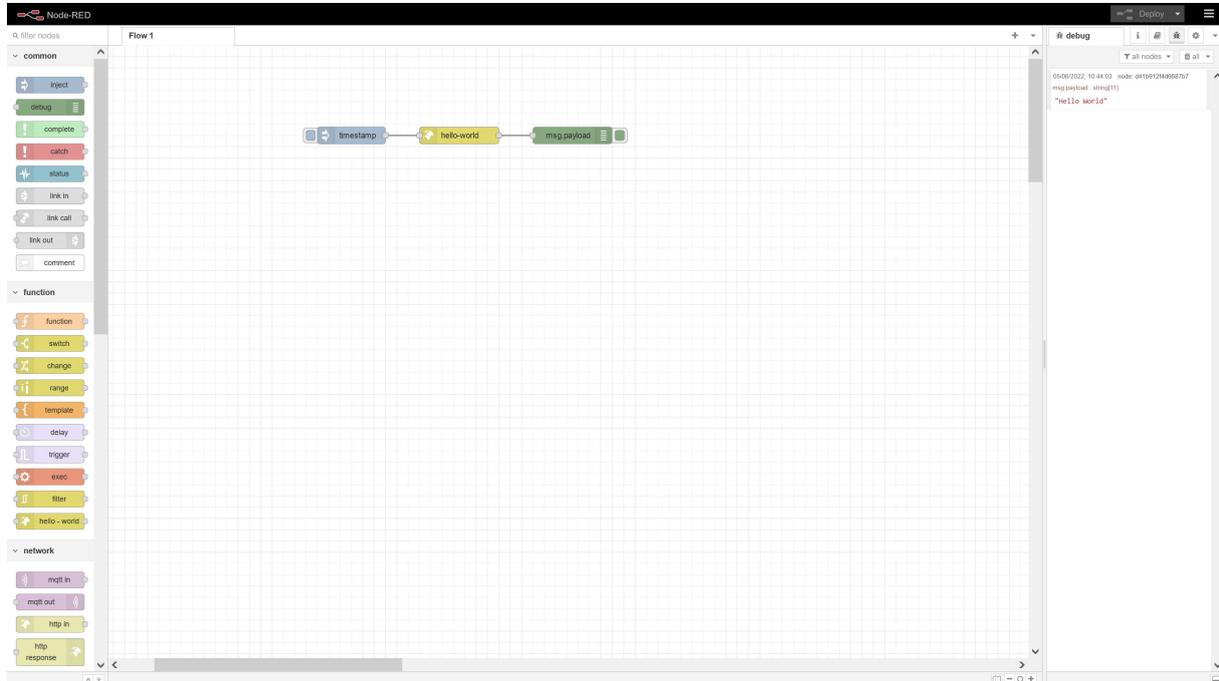


Abbildung 2: Node Red Flow

## 2.2 Node Red Bausteine

Zwei der drei verwendeten Bausteine sind in Node Red von Beginn an vorhanden. Es sind dies die Node Red eigenen “inject“ und “debug“ Bausteine. Der “Hello- World“ Baustein wurde zusätzlich hinzugefügt. Für spezifische Toradex Anwendungen gibt es jedoch keine Bausteine und diese müssen selbst erstellt werden. Für das erfolgreiche programmieren eines solchen benötigt man drei Programmiersprachen. Die Bedeutung dieser Sprachen werden in den folgenden Kapiteln aufgezeigt.

### 2.2.1 Java Script

Wie es der Name bereits vermuten lässt ist Java Script (JS) eine Skriptsprache und wurde im Jahr 1996 veröffentlicht. Sie dient dazu mit HTML zu kommunizieren und Eingaben zu verarbeiten. In Node Red wird darin ebenfalls der Baustein erstellt und es kann entsprechende Logik beschrieben werden. Im Kapitel 4 sind alle erstellten Bausteine mit Code erklärt.

### 2.2.2 HTML

HTML steht für Hyper Text Markup Language und dient der Darstellung im Webbrowser. In diesem sind Hilfestellungen, Beispiele oder zusätzliche Parameter für den Benutzer codiert. Dies erlaubt einem Nutzer von Node Red Bausteine zu parametrisieren, welche das JS weiterverarbeitet.

### 2.2.3 JSON

JSON (Java Script Option Notation) wird für den Upload der Bausteine gebraucht. Es ist ursprünglich als eine Untermenge der Programmiersprache Java Script erstellt worden. Heute verwenden sehr viele Sprachen die Einfachheit der Java Script Objon Notation. JSON ermöglicht der Maschine ein einfaches Auswerten und dem Menschen eine verständliche Lesbarkeit. In diesem File steht der Name des Pakets, die Version, der Autor und vieles mehr. Ein solches File wird im Kapitel 4.5.2 anhand eines Beispiels erläutert .

## 2.3 Toradex

Die Firma Toradex wurde im Jahr 2003 gegründet und beschäftigt aktuell 135 Mitarbeiter und liefert Produkte an über 3000 Kunden. Sie hat einen Sitz in Horw und entwickelt System-on-Modules (SOM) für folgende Bereiche:

- Industrieautomatisierung
- Transport
- Tests und Messungen
- Smart City

Toradex ist bestrebt die erste Wahl im Bereich Embedded-Computing für kleine bis mittlere Projektvolumen zu sein. Deshalb sind Module leicht verständlich und Langzeitverfügbar. Dadurch bleiben die Kosten tief.

### 2.3.1 Container

Toradex arbeitet fast ausschliesslich mit Container. Aus diesem Grund wird Nachfolgend das Grundverständnis für Container-Anwendungen aufgezeigt. Container sind eine Abstraktion im System, welche Eigenständig durch Docker-Engine betrieben werden. Das System kennt dabei die Inhalte im Container und der Container Inhalte auf dem System

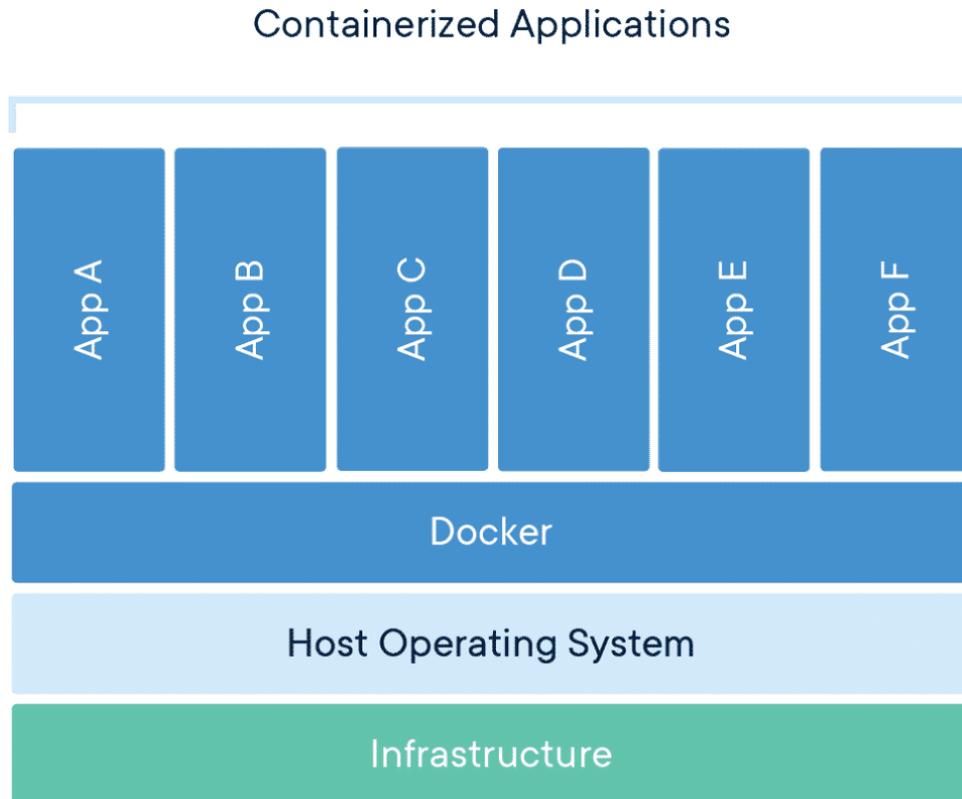


Abbildung 3: Software Container Übersicht

nicht. Der grosse Vorteil liegt darin, dass OS (Operating System) nicht zu verändern. Abbildung 3 <sup>1</sup> zeigt eine entsprechende grafische Übersicht. Die verschiedenen Applikationen entsprechen den Containern. Muss einer auf das OS zugreifen, so kann er dies nur mit zusätzlichen Parametern. Die Parameter werden beim Erstellen des Containers angegeben. Durch die Eigenschaft autark auf dem System zu laufen, erlaubt eine grosse Modularität. So können Container Images auf andere Systeme übernommen werden. Weiter ist die Versionsverwaltung geregelt und es können ohne Probleme ältere Versionen der Container verwendet werden. Sollte in einem Update etwas schief gehen ermöglicht der Container ein Rollback. Dies bedeutet es kann ohne weiteres auf eine Vorversion zurückgegangen werden und die aktuell störende überarbeitet. Docker kann aber Nachteile haben. Bei sehr vielen Containern, welche Daten austauschen, führt dies zu einer zusätzlichen Komplexität und evtl. sind diese Container nicht mehr Eigenständig betreibbar.

<sup>1</sup><https://www.docker.com/resources/what-container/> 06.06.2022

## 2.4 Aktuelle Situation

Toradex liefert zur Zeit erfolgreich SOM an seine Kunden aus. Die Vision des Unternehmens sind einfach verständliche Module zu entwickeln. Damit Nutzer ohne Programmierkenntnisse die Hardware verwenden können, will es einen Zugang dazu ermöglichen. Genauere Informationen sind im Kapitel 3.2 erläutert.

### 3 Methodik

Für einen erfolgreichen Projektabschluss ist eine entsprechende Planung und Methodik sehr wichtig. In dieser Arbeit wurde dabei nach dem Prinzip des Double Diamond vorgegangen.

#### 3.1 Double Diamond

Das Double Diamond Prinzip, welches in der Abbildung 4<sup>2</sup> ermöglicht in einem ersten Schritt eine Einarbeitung in die Problemstellung, bei dem man sich sehr breit aufstellt, bis hin zur Abgrenzung, welche Ziele zentral verfolgt werden. Das allgemeine Problem ist im Kapitel 3.2 und die daraus resultierenden Ziele und deren Abgrenzung im Kapitel 3.3 aufgezeigt. In einem 2. Schritt, dies entspricht dem Orangen Diamond, wird die Entwicklung zur Bewältigung der Problems vorangetrieben.

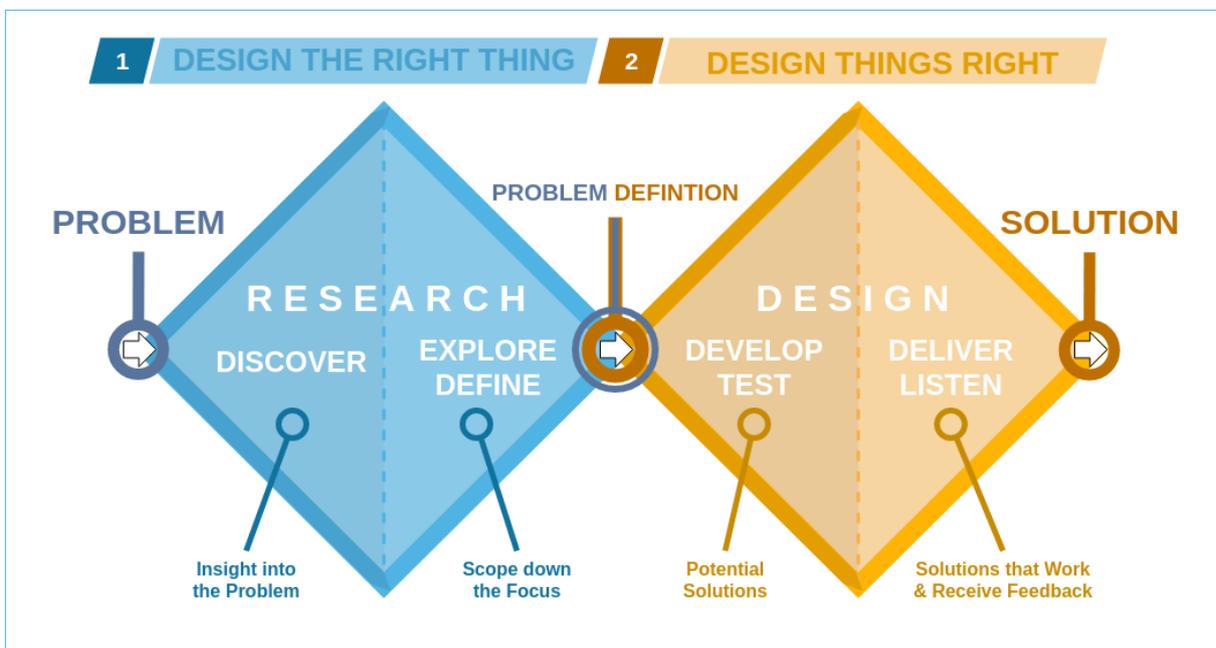


Abbildung 4: Double Diamond

<sup>2</sup><https://bit.ly/3NAZsgt/> 09.06.2022

## 3.2 Projektziel Toradex

Kunden soll ein vereinfachter Zugang zur Hardware ermöglicht werden. Dies mit Hilfe einer webbasierten Oberfläche. In dieser werden Bausteine zur Ansteuerung der Hardware zur Verfügung gestellt. Dadurch soll es dem User möglich sein, selber kleine Programme zu realisieren. Die Bausteine sollen Informationen über das System abfragen, wie zum Beispiel die Temperatur der CPU. Weiter ist es dem Benutzer möglich eventuelle extern angeschlossene Hardware per GPIO zu bedienen. Auch ist eine Implementation von einem I<sup>2</sup>C vorstellbar. Diese Oberfläche soll mittels Node-RED realisiert werden. Das Unternehmen hat selbst keine Kenntnisse über Node Red und deren Entwicklung. Durch diese Arbeit will es deren Tauglichkeit herausfinden. Der User soll diese Bausteine anwenden können ohne fundierte Kenntnisse in der Programmierung zu besitzen.

## 3.3 Abgrenzung und Ziele BAT

Das Projektziel der Firma Toradex ist ambitioniert und nicht in einer einzigen BAT umsetzbar. Aus diesem Grund wird die BAT in Meilensteine unterteilt, **wobei das Erreichen und Abschliessen des Meilensteins 3 das Endziel ist**. Nachstehend sind diese Meilensteine in Prosa erläutert. Damit das Erreichen eines Meilensteins spezifiziert werden kann, dient der anschliessende Anforderungskatalog.

### 3.3.1 Meilenstein 1 Inbetriebnahme Carrier Board

Mit dem Abschluss dieses Meilensteins wird das Carrier Board erfolgreich in Betrieb genommen. Es funktionieren alle Anwendungen und es konnte eine erste eigene Node erstellt werden. Das Abschliessen des Meilensteins eins lässt sich mit der Tabelle 1 überprüfen.

Anforderung	Art Fest: F Wunsch: W	Beschreibung / Definition
F1.1	F	Es wird die entsprechende Hardware beschafft:
F1.2	F	Quickstart Guide von Toradex erfolgreich abgearbeitet
F1.3	F	Erstes Beispiel Node-Red von Toradex (soweit möglich zusätzliche HW wird nicht beschafft)
F1.4	F	Hello Wolrd in Node-Red
F1.5	F	Erstes Beispiel Creat your first Node

Tabelle 1: Anforderungen Meilenstein 1

### 3.3.2 Meilenstein 2 Funktionsbausteine CPU und GPIO

Um den Meilensteins drei erfolgreich zu meistern, werden im Meilenstein zwei erste Funktionsbausteine erstellt. Diese sind dokumentiert um weitere, gemäss diesen Beispielen, zu realisieren. Ein Funktionsbaustein soll dabei Informationen vom Internen CPU ablesen (BSP: CPU Temperatur). Ein zweiter Baustein soll das Auslesen und Setzen eines GPIO Pins ermöglichen. In der Tabelle 2 sind die messbaren Anforderungen aufgezeigt.

Anforderung	Art Fest: F Wunsch: W	Beschreibung / Definition
F2.1	F	Die benötigten Fachkenntnisse in JS, JSON und HTML sind vorhanden
F2.2	F	Erfolgreiche abfragen der CPU Temp
F2.3	F	GPIO low, high kann gelesen werden
F2.4	F	GPIO low, high kann gesetzt werden
F2.5	F	Test der Bausteine mit 3 Benutzern
W2.6	W	GPIO Bausteine skalierbar =>Es kann in einem Baustein GPIO PIN gewählt werden
W2.7	W	Es wird in einem Video dokumentiert wie Funktionsbausteine zu verwenden sind

Tabelle 2: Anforderungen Meilenstein 2

### 3.3.3 Meilenstein 3 Funktionsbausteine CPU und PWM

Mit dem abschliessen der ersten beiden Meilensteine sind die notwendigen Kenntnisse vorhanden. Es ist möglich auf die HW zuzugreifen und weitere Funktionsbausteine können erstellt werden. Durch das Abschliessen des Meilensteins drei ist es dem Benutzer möglich, erste kleine Anwendungen zu realisieren. Das Erreichen der Ziele kann mit der Tabelle 3 verifiziert werden.

Anforderung	Art Fest: F Wunsch: W	Beschreibung / Definition
F3.1	F	Funktionsbaustein für Prozessorauslast
F3.2	F	Funktionsbaustein PWM
F3.3	F	Funktionsbausteine sind mit Hilfestellungen dokumentiert
F3.4	F	Kleine Demoanwendung

Tabelle 3: Anforderungen Meilenstein 3

#### Demo Beschrieb

Ziel der Demo ist es, die Funktionsbausteine zu erklären und deren Richtigkeit aufzuzeigen. Die Demo beinhaltet folgende Funktionen:

- Anhand der Prozessorauslastung wird ein Leuchtmittel angesteuert, deren Helligkeit die Beschäftigung symbolisiert. Ist diese über 80% blinkt das Leuchtmittel
- Mittels LED wird eine visuelle Statusrückmeldung betreffend Prozessor Temperatur angezeigt.

Die hier beschriebenen Meilensteine sind zusammengefasst aus dem Dokument im Anhang. Dieses besitzt weitere und dienen als Ausblick für weiterführende Arbeiten.

## 4 Softwarebeschreibung

Im Folgenden sind die Bausteine und ihr Code beschrieben. Mit Hilfe von Bildern und dem Gegenüberstellen der Bedeutung in Node Red wird ein Verständnis für die Code Abschnitte vermittelt. Weiter wird der Upload Vorgang aufgezeigt und das Aufsetzen der Container. In Abbildung 5 ist das Zusammenspiel der diversen Komponenten ersichtlich. Der User logt sich auf Portainer ein Darüber erstellt, respektive startet er die Node Red

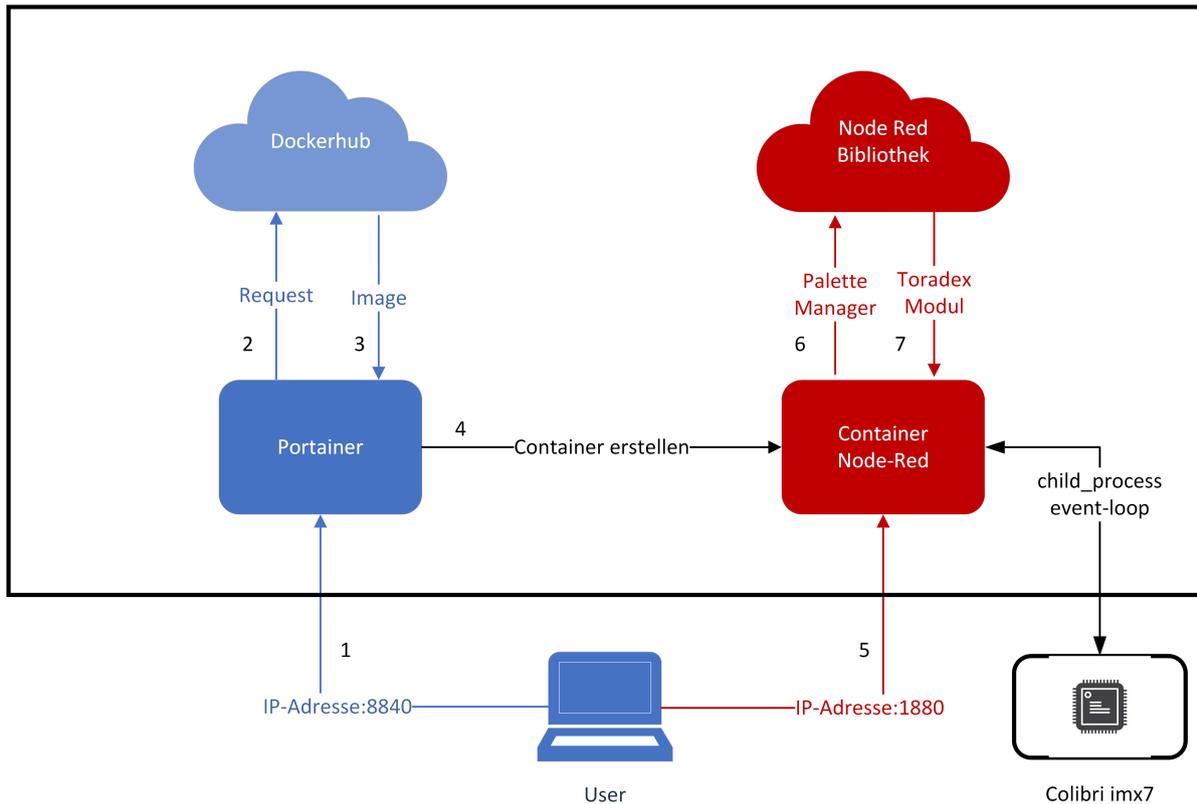


Abbildung 5: Systemübersicht

Container. Für dies wird das zur Verfügung gestellte Image, welches im Kapitel 4.6 beschrieben ist, geladen. Nun kann auf die Weboberfläche von Node Red zugegriffen werden. Darüber lassen sich die erstellten Bausteine mittels Palettenmanager installieren. Mittels “event-loop“ erfolgt der Hardwarezugriff

### 4.1 CPU Temperature

Der “CPU Temperature“ Baustein erlaubt das Auslesen der aktuellen CPU Temperatur. Dabei kann der User selbst entscheiden, ob er das Resultat gerne in Grad oder Kelvin erhalten möchte. In Abbildung 6 ist der erstellte Baustein zu sehen.

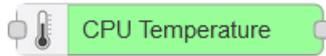


Abbildung 6: Baustein CPU Temperatur

#### 4.1.1 HTML

In der Abbildung 7 ist ein Codeabschnitt zu sehen, welcher einen Teil der Weboberfläche in Node Red beschreibt. Die nummerierten Pfeile zeigen den Zusammenhang aus dem Code und der Weboberfläche. Dies ist Abbildung 8 zu entnehmen. Die Pfeile eins und

```

<!-- CPU Temperature-->

<script type="text/html" data-template-name="CPU Temperature">
  <div class="form-row">
    <label for="node-input-name"><i class="fa fa-tag"></i> Name</label>
    <input type="text" id="node-input-name" placeholder="Degree / Kelvin">
  </div>

  <div class="form-row">
    <label for="node-input-unit"><i class="fa fa-tag"></i> Unit</label>
    <input type="text" id="node-input-unit">
  </div>
</script>

<script type="text/html" data-help-name="CPU Temperature">
  <p> This node send the CPU Temperature in degree or kelvin back, which do you more prefer.<br/>
  It is just a Number, the reason for that is to proof with a switch Statement a Number and do something.<br/>
</p>
</script>
    
```

Abbildung 7: HTML CPU Temperatur

zwei sind dabei innerhalb des “script“ Befehls. Dieser Abschnitt beschreibt, dass es sich um einen HTML Code handelt. Weiter ist mit dem Template Name “CPU Temperature“ angegeben, zu welchem Baustein der folgende Code gehört. Der “div“ Befehl beschreibt eine Sektion im HTML File. Dies bedeutet, dass es einen Platz über den gesamten verfügbaren Bereich einnimmt. Danach wird diese mit Hilfe des Labels beschrieben. Die Klasse “fa-fa-tag“ bewirkt das Zeichen neben dem Name und Unit in Abbildung 8. Der “Placeholder“ ist ein Vorschlag, welcher bei Benutzereingabe verschwindet. Das Resultat der

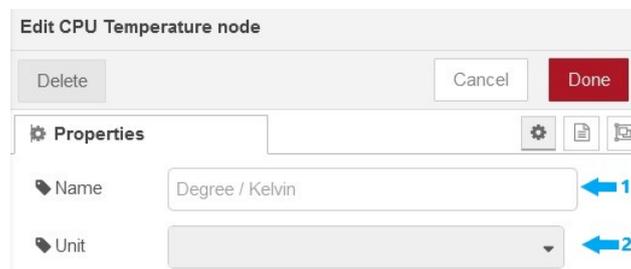


Abbildung 8: Node Red CPU Temperatur

zweiten “div“ Sektion (Pfeil zwei) ist ebenfalls in der Abbildung 8 zu sehen. Es handelt sich um eine Einstellmöglichkeit, wobei das Dropdown-Menü in der Abbildung 9 beschrieben wird. Der Pfeil drei bewirkt einen Eintrag im Hilfebuch des Bausteins. Dabei wird am Ende der Zeile mit dem Befehl “<br/>“ ein Zeilenumbruch realisiert. Das Property

```

<script type="text/javascript">
  RED.nodes.registerType('CPU Temperature',{
    category: 'Toradex',
    color : '#98FF98',
    defaults :{
      name: {value:""},
      unit:{value:0}
    },
    inputs:1,
    outputs:1,
    icon: "thermometer.png",
    label: function(){
      return this.name||"CPU Temperature";
    },
    oneditprepare() {
      const node = this
      const {number} = node

      $('#node-input-unit').typedInput({
        type: 'number',
        types: [
          {
            value: 'number',
            // Should set the default value but only works if you
            // are working with strings will not accept any type.
            default: Number(number),
            options: [
              { value: 0,label: 'degree'},
              { value: 273.15,label: 'kelvin'}
            ]
          }
        ]
      })
      // Sets the current value will only work if same type
    }).typedInput('value', Number(number))
  });
</script>

```

Abbildung 9: HTML CPU Temperatur JS Abschnitt

“category“ bewirkt, dass der Baustein unter einem neu erstellten Abschnitt Toradex zu finden ist. Diese Gruppierungen sind in der Abbildung 1 links zu sehen, wie zum Beispiel “Common“. Bei “color“ ist die Farbe des Bausteins definiert. In diesem Falle entspricht dieser Wert einem grün wie in Abbildung 6. “Default“ ermöglicht vordefinierte Werte, falls der Benutzer eine Eingabe nicht macht. So können inkonsistente Zustände verhindert werden. In diesem Fall wird bei “unit“ den Wert Null angegeben. Dadurch sendet der Baustein die Temperatur standardmässig in Grad zurück. Wie dies genau funktioniert ist im Kapitel 4.1.2 erklärt. Die “inputs“ und „outputs“ beschreiben, wie viele Eingangs- und Ausgangsports der Baustein besitzt. Diese werden mit einem Quadrat symbolisiert und ermöglichen das verbinden mit anderen Bausteinen. Hinter dem “icon“ wird das Bild auf dem Baustein links eingefügt. Mit “function“ erstellt man ein Label mit dem Namen “CPU Temperature“. In der “oneditprepare()“ Funktion wird das Dropdown- Menü beschrieben. Dabei wird angegeben, dass es sich um einen Input handelt und dieser Input eine Nummer ist. Die “options“ definieren die Auswahlmöglichkeiten des Benutzers. In diesem Falle kann der Benutzer die Rückgabe der CPU Temperatur in Kelvin oder Grad auswählen. Hinter diesen ist ein Wert zugeordnet und kann im Java Script verwendet werden

### 4.1.2 JavaScript

In diesem File sind Abläufe programmierbar, kann auf Benutzereingaben interagiert und Befehle ausgeführt werden. In der Abbildung 10 ist das entsprechende Java Script zu sehen. Die Zuweisung “exec“ wird benötigt, um später im Code einen Befehl auszuführen.

```
const {exec} = require('child_process');
const { stdout } = require('process');

function sleep(ms){
  return new Promise(resolve => setTimeout(resolve, ms));
}

module.exports = function(RED){

  function cpuTemp(config){
    RED.nodes.createNode(this, config);
    this.unit = config.unit;
    var node = this;
    this.on('input', function(msg){
      var newMsg;
      exec(`cat /sys/devices/virtual/thermal/thermal_zone0/temp`, (error, stdout, stderr) => {
        if(error){
          node.error = error.toString();
          return
        }
        newMsg = stdout;
      });
      sleep(1000)
      .then(() => { newMsg = (parseFloat(newMsg)/1000) + parseFloat(node.unit)})
      .then(() => { msg.payload = newMsg.toString();})
      .then(() => { node.send(msg)})
    });
  }
  RED.nodes.registerType("CPU Temperatur",cpuTemp);
}
```

Abbildung 10: JavaScript CPU Temperatur

Durch das ausführen des “exec“ wird das Kommando auf den Event Loop gelegt und von JavaScript verarbeitet. In dem Befehl „function“ wird eine Funktion mit dem Namen “cpuTemp“ erstellt. In dieser wird Node Red aufgefordert, den Baustein zu erstellen und mit “this.unit = config.unit“ wird dem Node das Property Unit aus dem HTML File mitgegeben. Anschliessend wird der Befehl zum Auslesen der CPU Temperatur ausgeführt. “Cat“ liest dabei ein file, welches den Wert in Grad Faktor 1000 grösser returniert. Dieser Wert gelangt über den “Callback“ der Funktion “exec“ in stdout und wird “newMsg“ zugewiesen. Mittels “sleep“ wird die eigens erstellte Funktion aufgerufen. Diese bewirkt ein Warten, um sicherzustellen das die Werte bereits im Callback sind. Dies ist notwendig, da JavaScript eine Asynchrone Sprache ist. Mit einem einfachen “setTimeout()“ würden die nachfolgenden Codezeilen ausgeführt. Entsprechend würde ein falscher Wert weiter geschickt. Ein Warten mit “execSync“ wäre fatal, da dann auch der Eventloop stoppt und wartet. Dies bedeutet alle anderen Events warten auch. Danach wird der Rückgabewert verarbeitet und vorbereitet für die Ausgabe. das “parseFloat“ bewirkt ein umwandeln des strings in eine floating point Zahl. Dadurch wird das ausführen von Rechenoperationen ermöglicht. Dabei wird der Wert mit 1000 dividiert um den Faktor zu streichen und der entsprechende Offset dazu addiert. Dieser kommt aus dem Property Unit, bei welchem Kelvin oder Grad ausgewählt wurde. Hinter diesen ist der Wert 0 für Grad und der Wert 273.15 für den Wert Kelvin definiert. Abschliessend folgt die Zuweisung an das Objekt “msg.payload“. Dies erwartet einen String, weshalb der berechnete Wert mittels „to-

String()“ konvertiert wird. Mit „node.send(msg)“ wird das Objekt gesendet und verlässt den Baustein.

## 4.2 CPU Usage

Der „CPU Usage“ Baustein ermöglicht das Messen der CPU Auslastung über eine gewisse Zeitdauer. Der Rückgabewert ist ein gemittelter Wert über eine gewisse Zeit. Damit der Baustein erfolgreich funktioniert muss auf dem  $\mu$ Controller „vmstat“ installiert sein. Im vorgefertigten Container, wie im Kapitel 4.6 beschrieben, ist die bereits vorinstalliert. In Abbildung 11 ist der Baustein zu sehen.

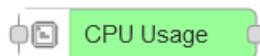


Abbildung 11: Baustein CPU Auslastung

### 4.2.1 HTML

Das HTML File für den “CPU Usage“ Baustein ähnelt dem des „CPU Temperature“, welcher im Kapitel 4.1.1 beschrieben ist. Die Abbildung 12 zeigt den entsprechenden Source Code. Der Pfeil eins zeigt auf den Namen mit Platzhalter. Dieser kann in der Node Red

```
<!-- CPU Usage-->
<script type="text/html" data-template-name="CPU Usage">
  <div class="form-row">
    <label for="node-input-name"><i class="fa fa-tag"></i> Name</label> ← 1
    <input type="text" id="node-input-name" placeholder="2 Seconds">
  </div>

  <div class="form-row">
    <label for="node-input-timeMeasure"><i class="fa fa-tag"></i> Time to measure</label> ← 2
    <input type="text" id="node-input-timeMeasure">
  </div>

  <div class="form-row">
    <label for="node-input-measureUnit"><i class="fa fa-tag"></i> Measure unit</label> ← 3
    <input type="text" id="MeasureUnit">
  </div>
</script>

<script type="text/html" data-help-name="CPU Usage">
  <p> This node send the CPU usage back in %, measured over a specific time.<br/> ← 4
</p>
</script>
```

Abbildung 12: HTML CPU Auslastung

Oberfläche angepasst werden. Dadurch ist der Flow besser zu lesen, da die Bausteine neue Beschriftung annehmen. Im Beispiel wurde der Name als CPU Auslastung 2s beschriftet. Der daraus resultierende Effekt ist in der Abbildung 13 zu sehen. Die Pfeile zwei und drei sind für Benutzereingaben. Dies erlaubt dem User eine Zeit von einigen Sekunden bis einige Stunden auszuwählen. Der Baustein wird mit “Red.nodes.registerTyp“ registriert. Dafür verwendet er das Label „CPU Usage“. Die Anschliessenden Propertyts “category“

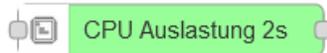


Abbildung 13: Baustein CPU Auslastung beschriftet

und “color“ entsprechen den selben Einstellungen wie beim CPU Temperatur Baustein. Das Property „defaults“, in Abbildung 14 zeigt neu drei Werte. Der erste Wert ist für

```
<script type="text/javascript">
  RED.nodes.registerType('CPU Usage',{
    category: 'Toradex',
    color : '#98FF98',
    defaults :{
      name: {value:""},
      timeMeassure: {value:0},
      measureUnit: {value:1}
    },
    inputs:1,
    outputs:1,
    icon: "cpu_usage.png",
    label: function(){
      return this.name||"CPU Usage";
    },
  },
);
```

Abbildung 14: HTML CPU Auslastung JavaScript 1/3

die Namensgebung. Der zweite für die Benutzereingabe des Dezimalwertes und der dritte für die Einheit. Die Realisierung des Property „timeMeassure“ ist in Abbildung 15 zu betrachten. Dem Benutzer wird dabei im Dropdown-Menü Dezimalwerte von eins bis zehn

```
$('#node-input-timeMeassure').typedInput({
  type: 'number',
  types: [
    {
      value: 'number',
      // Should set the default value but only works if you
      // are working with strings will not accept any type.
      default: Number(number),
      options: [
        { value: 1,label: '1 '},
        { value: 2,label: '2 '},
        { value: 3,label: '3 '},
        { value: 4,label: '4 '},
        { value: 5,label: '5 '},
        { value: 6,label: '6 '},
        { value: 7,label: '7 '},
        { value: 8,label: '8 '},
        { value: 9,label: '9 '},
        { value: 10,label: '10'}
      ]
    }
  ]
})
// Sets the current value will only work if same type
}).typedInput('value', Number(number))
```

Abbildung 15: HTML CPU Auslastung JavaScript 2/3

vorgeschlagen. Durch das explizite Vorschlagen der Werte kann der Nutzer keine falschen übermitteln, die in einem Fehler enden würden. Hinter den Labels verstecken sich wiederum Dezimalwerte. Die Bedeutung dieser wird im Kapitel 4.2.2 erklärt. Die Auswahl der Zeitbasis erfolgt nach dem gleichen Prinzip wie beim auswählen des Zeitwertes. Die Labels dienen als Benutzerschnittstelle, dadurch muss er sich um nichts kümmern. Dies ist in der Abbildung 16 zu sehen.

```

$('#node-input-meassureUnit').typedInput({
  type: 'number',
  types: [
    {
      value: 'number',
      // Should set the default value but only works if you
      // are working with strings will not accept any type.
      default: Number(number),
      options: [
        { value: 1,label: 'second'},
        { value: 60,label: 'minute'},
        { value: 3600,label: 'hour'}
      ]
    }
  ]
})
// Sets the current value will only work if same type
}).typedInput('value', Number(number))
});

```

Abbildung 16: HTML CPU Auslastung JavaScript 3/3

Die Werte hinter den Labels der Messeinheit entsprechen dem Faktor, um auf Sekunden zu kommen.

#### 4.2.2 Java Script

In der Abbildung 17 ist der Code im Java Script ersichtlich. Des Prinzip ähnelt dem des CPU Temperatur Bausteins. Es wird eine Berechnung / Aufgabe ausgeführt und gibt

```

function cpuUsage(config){
  RED.nodes.createNode(this,config);
  this.timeMeassure = config.timeMeassure;
  this.meassureUnit = config.meassureUnit;
  var node = this;
  this.on('input', function(msg){
    var newMsg;
    var command = parseInt(node.timeMeassure) * parseInt(node.meassureUnit) +1;
    exec(`vmstat 1 ${command}|tail -1|awk '{print $15}'`, (error, stdout, stderr) => {
      if(error){
        node.error = (`exec error: ${error}`);
        return;
      }
      newMsg = stdout;
    });
    sleep((parseInt(command) * 1000)+1000 )
    .then(() => { 100 - parseInt(newMsg)})
    .then(() => { msg.payload = newMsg.toString();})
    .then(() => { node.send(msg)})
  });
}
RED.nodes.registerType("CPU Usage",cpuUsage);
};

```

Abbildung 17: Java Script CPU Auslastung

den Wert zurück. Das Zurücksenden erfolgt mit “node.send“. Zuerst werden alle Property dem Node übermittelt. In der Funktion wird die Benutzereingabe verwaltet. Dazu müssen die erhaltenen Werte auf einen “int“ geparkt werden. Dies weil die Property die Werte als String interpretieren. Würde die Berechnung ohne das Parsen stattfinden, so würde ein String zusammengesetzt werden. Dies liegt an der Eigenschaft von Java Script, es ermöglicht Strings mit + zu erweitern. Die Berechnung multipliziert die Eingabe des Benutzers, dadurch wird der richtige Wert im Sekunden berechnet. Das Addieren der Eins am Schluss wird benötigt, da vmstat die Differenz von Eins und der mitgegebenen

Zahl als Messzeit berechnet. Es würde also eine Sekunde zu wenig berücksichtigt. Nach dem erfolgreichen parsen wird das Kommando mit “exec“ auf den Event Loop gelegt. Anschliessend wird mit der erstellten “sleep“ Funktion auf den Callback gewartet. Dieser entspricht noch nicht dem Endergebnis, sondern muss von 100 subtrahiert werden, da vmstat den %-Wert zurück gibt der noch frei ist.

### 4.3 GPIO setzen

Im Kapitel 3.3 wird aufgezeigt, dass der Nutzer auf die Hardware zugreifen soll. Ein Baustein dafür ist der “GPIO Set“, wie in Abbildung 18 zu sehen. Dieser erlaubt die

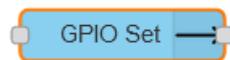


Abbildung 18: GPIO setzen

Ansteuerung aller möglichen 40 Pins auf dem Pin Header. Weiter wird dem Baustein mitgeteilt, ob dieser eingeschaltet oder ausgeschaltet wird. Durch dies ist er skalierbar und kann für weitere GPIO Pin konfiguriert werden. Zum verwenden dieses Bausteins muss „gpiod“ zwingend installiert sein. Im vorgefertigten Container ist dies bereits der Fall, in einem eigenen müsste dies über die Kommandozeile nachinstalliert werden. Mehr dazu ist im Kapitel 4.6 zu finden.

### 4.3.1 HTML

Der HTML Code beinhaltet einen Script-Abschnitt für die Benutzereingaben und einen für die Hilfe. Diese beinhaltet ein Beispiel, welche Eingaben der User machen kann und einen möglichen Rückgabewert. Die Pfeile der Abbildung 19 korrespondieren mit der Abbildung

```

<!-- GPIO Set-->

<script type="text/html" data-template-name="GPIO Set">
  <div class="form-row">
    <label for="node-input-name"><i class="fa fa-tag"></i> Name</label> ← 1
    <input type="text" id="node-input-name" placeholder="Name">
  </div>

  <div class="form-row">
    <label for="node-input-pinNumber"><i class="fa fa-tag"></i> Pin Number</label> ← 2
    <input type="text" id="node-input-pinNumber">
  </div>

  <div class="form-row">
    <label for="node-input-numberOnOff"><i class="fa fa-tag"></i> On / off</label> ← 3
    <input type="text" id="node-input-numberOnOff">
  </div>
</script>

<script type="text/html" data-help-name="GPIO Set">
  <p> You can chose all possible pins from 40 pin header. It send after setting the GPIO pin what was done. <br/> ← 4
  Example: you choose to turn pin 40 on <br/>
  Message: Pin 40 on
  </p>
</script>

```

Abbildung 19: HTML GPIO set

20. Die Benutzereingabe erfolgt wiederum über ein Dropdown-Menü. Der Name dient zur Lesbarkeit im Flow selbst, dadurch kann eine Verwechslung ausgeschlossen werden. Das

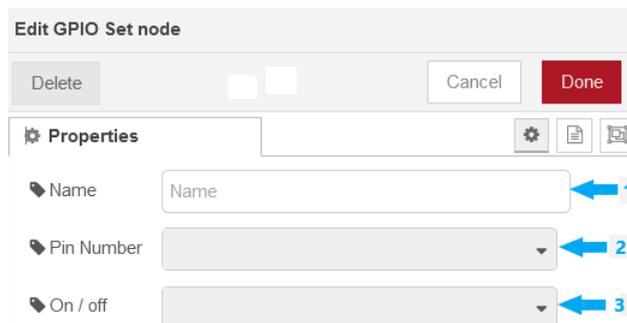


Abbildung 20: GPIO set Node Edit

Property “Name“ wird ansonsten nicht gebraucht. Das Bild ist bei diesem Baustein auf der rechten Seite, da es die Eigenschaft besser wiedergibt.

Dies wird mit dem Property „align: right“, welches in der Abbildung 21 zu sehen ist, ermöglicht. Die Farbe wurde hier, gegenüber dem CPU Bausteinen, auf blau geändert.

```
<script type="text/javascript">
RED.nodes.registerType('GPIO Set',{
  category: 'Toradex',
  color: '#89CFF0',
  defaults :{
    name: {value:""},
    pinNumber: {value:0},
    numberOnOff:{vlaue: 0}
  },
  inputs:1,
  outputs:1,
  icon: "input.png",
  align: "right",
  label: function(){
    return this.name||"GPIO Set";
  },
},
```

Abbildung 21: HTML GPIO set JavaScript 1/2

Dies bietet dem Nutzer einen höheren Komfort zur Identifikation. Die GPIO Nodes sind im gleichen Abschnitt unter Toradex zu finden. Dadurch sind GPIO Bausteine im gleichen Untermenü, wie die CPU Bausteine. Die Eingabe der Pins wurde bereits vorkonfiguriert. Dadurch können keine Pins auf dem Header angesteuert werden die nicht möglich sind, wie zum Beispiel ein Ground Pin. Dies ist in der Abbildung 22 zu sehen.

```
$('#node-input-pinNumber').typedInput({
  type: 'number',
  types: [
    {
      value: 'number',
      // Should set the default value but only works if you
      // are working with strings will not accept any type.
      default: Number(number),
      options: [
        { value: 2,label: 'Pin 3'},
        { value: 4,label: 'Pin 5'},
        { value: 6,label: 'Pin 7'},
        { value: 7,label: 'Pin 8'},
        { value: 9,label: 'Pin 10'},
        { value: 10,label: 'Pin 11'},
        { value: 11,label: 'Pin 12'},
        { value: 12,label: 'Pin 13'},
        { value: 14,label: 'Pin 15'},
        { value: 15,label: 'Pin 16'},
        { value: 17,label: 'Pin 18'},
        { value: 18,label: 'Pin 19'},
        { value: 20,label: 'Pin 21'},
        { value: 21,label: 'Pin 22'},
        { value: 22,label: 'Pin 23'},
        { value: 23,label: 'Pin 24'},
        { value: 25,label: 'Pin 26'},
        { value: 26,label: 'Pin 27'},
        { value: 27,label: 'Pin 28'},
        { value: 28,label: 'Pin 29'},
        { value: 30,label: 'Pin 31'},
        { value: 31,label: 'Pin 32'},
        { value: 22,label: 'Pin 33'},
        { value: 34,label: 'Pin 35'},
        { value: 35,label: 'Pin 36'},
        { value: 36,label: 'Pin 37'},
        { value: 37,label: 'Pin 38'},
        { value: 39,label: 'Pin 40'}
      ]
    }
  ]
})
// Sets the current value will only work if same type
}).typedInput('value', Number(number))
```

Abbildung 22: HTML GPIO set JavaScript 2/2

Das setzen des Pin auf „high“ oder „low“ erfolgt über die Auswahl on oder off. Diese Implementation ist nach dem Gleichen Prinzip wie bei der Auswahl des Pins.

### 4.3.2 Java Script

Der „GPIO Set“ Baustein wie auch der „GPIO Get“ können die gleichen Pins bedienen. Aus diesem Grund ist im Java Script ein zweidimensionales Array definiert. Dies ist in Abbildung 23 zu sehen. Das Array wurde bewusst mit allen Pins, sprich mit Power und

```
const {execSync} = require('child_process');
module.exports = function(RED){
  var GPIO = [
    ['PWR', 'out'],
    ['PWR', 'in'],
    [6, 9],
    ['PWR', '0'],
    [6, 8],
    ['GND', 'GND'],
    [3, 14],
    [3, 2],
    ['GND', 'GND'],
    [3, 3],
    [5, 22],
    [0, 2],
    [3, 19],
    ['GND', 'GND'],
    [4, 1],
    [4, 11],
    ['PWR', 'out'],
    [3, 17],
    [3, 9],
    ['GND', 'GND'],
    [3, 8],
    [3, 16],
    [3, 10],
    [3, 23],
    ['GND', 'GND'],
    [3, 22],
    [0, 7],
    [0, 6],
    [3, 13],
    ['GND', 'GND'],
    [3, 12],
    [0, 9],
    [0, 10],
    ['GND', 'GND'],
    [0, 11],
    [5, 21],
    [3, 18],
    [3, 15],
    ['GND', 'GND'],
    [4, 0]
  ];
};
```

Abbildung 23: Java Script GPIO set 1/2

GND, initialisiert. Dies vereinfacht den Hardwarezugriff. So sind bei den Input Property die Werte des gewünschten Pins minus eins hinterlegt. Dieser Wert entspricht gerade dem Platz im Array. Die 1. Dimension beschreibt dabei, auf welchem Chip sich der Pin befindet. Die 2. Dimension entspricht dem Pin auf dem Chip. Beim Colibri imx7 gibt es total sieben gpiochips, angefangen von null bis sechs.

Die Ansteuerung dieser Pins ist in der Abbildung 24 zu sehen. Zur Verständlichkeit ein Beispiel dazu:

Angenommen der User will Pin 40 mit “on“ ansteuern. In diesem Fall wird in die Variable “command“ den String „gpiocset gpiochip4 0=1“ zugewiesen und anschliessend wird der Code mit „exec“ dem Event Loop übergeben. Der Baustein überprüft beim “if/else“ ob der Pin ein- oder ausgeschaltet wurde. Dann wird „node.pinNumber“ auf einen int geparst um die Rechenoperation auszuführen. Diese muss ausgeführt werden, da hinter den Ausgewählten Pins jeweils der Pinnummer mit minus eins realisiert ist. Als Abschluss wird “msg.payload“ den String zugewiesen und mit “node.send“ im Flow weitergereicht. Es wird nicht auf den Callback gewartet, da man diesen Wert nicht benötigt.

```
function gpioSet(config){
  RED.nodes.createNode(this,config);
  this.pinNumber = config.pinNumber;
  this.numberOnOff = config.numberOnOff
  var node = this;
  this.on('input', function(msg){
    var newMsg;
    var onOff;
    var command = `gpiocset gpiochip${GPIO[node.pinNumber][0]} ${GPIO[node.pinNumber][1]}=${node.numberOnOff}`;
    exec(`${command}`, (error, stdout, stderr) => {
      if(error){
        node.error = error.toString();
        return
      }
      newMsg = stdout;
    });
    if(node.numberOnOff == 0){
      onOff = "Off";
    }else{
      onOff = "On";
    }
    var pin = 1 + parseInt(node.pinNumber);
    msg.payload = `Pin ${pin} ${onOff}`;
    node.send(msg)
  });
}
RED.nodes.registerType("GPIO Set",gpioSet);
```

Abbildung 24: Java Script GPIO set 2/2

## 4.4 GPIO lesen

Der GPIO Baustein zum Lesen der Hardware ähnelt dem “GPIO Set“. Dieser ist in Abbildung 25 zu sehen. Visuell unterscheiden sie sich nur vom Text und dem Symbol auf der linken anstatt auf der rechten Seite. Der Node ermöglicht durch ein Dropdown-Menü



Abbildung 25: GPIO lesen

das Auslesen des 40 Pin Headers. Beim verwenden von diesem kommt ein Return mit eins oder null. Dies ermöglicht eine einfache Weiterverarbeitung im Flow.

### 4.4.1 HTML

Das HTML File unterscheidet sich nur geringfügig vom "GPIO set" Baustein. Deshalb wird nachfolgend nur auf die Änderungen eingegangen. Die Benutzeroberfläche des Codes in Abbildung 26 ist in der Abbildung 27 zu sehen. Die Zusammenhänge sind mit entsprechenden Pfeilen und Nummern gekennzeichnet. Ebenfalls auf dem Bild ist das

```

<!-- GPIO Get-->

<script type="text/html" data-template-name="GPIO Get">
  <div class="form-row">
    <label for="node-input-name"><i class="fa fa-tag"></i> Name</label> ← 1
    <input type = "text" id="node-input-name" placeholder="Name">
  </div>

  <div class="form-row">
    <label for="node-input-pinNumber"><i class="fa fa-tag"></i> Pin Number</label> ← 2
    <input type="text" id="node-input-pinNumber">
  </div>
</script>

<script type="text/html" data-help-name="GPIO Get">
  <p> Get the GPIO value from a pin from the 40 pin header <br/> ← 3
  Return: 1 when it's high <br/>
  Return: 0 when it's low
</p>
</script>

```

Abbildung 26: HTML GPIO get

aufgeklappte Dropdown-Menü. Dadurch kann sichergestellt werden, dass konsistente Zustände und Informationen an den Node übergeben werden. Der Unterschied vom „GPIO get“ zum „GPIO set“ liegt in der geringeren Anzahl an Property.

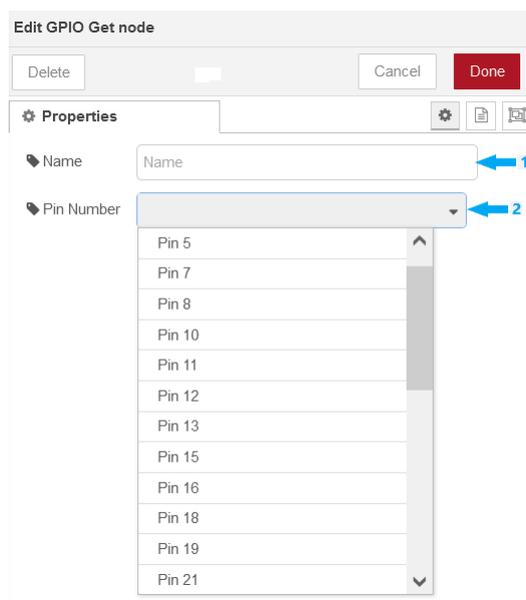


Abbildung 27: GPIO get node edit

## 4.4.2 Java Script

Das Java Script des “GPIO Get“ Bausteins benützt, wie im Kapitel 4.3.2 bereits beschrieben, ebenfalls das zweidimensionale Array. In der Zeile mit “var command“ wird der entsprechende string zusammengesetzt und der Variable zugewiesen. Der Callback

```
function gpioGet(config){
  RED.nodes.createNode(this,config);
  this.pinNumber = config.pinNumber;
  var node = this;
  var newMsg;
  this.on('input', function(msg){
    var command = `gpioget gpiochip${GPIO[node.pinNumber][0]} ${GPIO[node.pinNumber][1]}`;
    exec(`${command}`, (error, stdout, stderr) => {
      if(error){
        node.error = error.toString();
        return
      }
      newMsg = stdout;
    });
    setTimeout(() => {msg.payload= newMsg; node.send(msg)}, 50);
  });
}
RED.nodes.registerType("GPIO Get", gpioGet);
};
```

Abbildung 28: Java Script GPIO get

der Funktion “exec“ wird direkt an “msg.payload“ übergeben. Hier wird das Warten bereits in der Codezeile mit “setTimeout“ ausgeführt.

## 4.5 Upload

Die Bausteine müssen durch das uploaden den Usern zur Verfügung gestellt werden. Dabei hat Node Red einige Vorschriften, welche eingehalten werden müssen, da ansonsten einem das Hochladen zur Node Red library verweigert wird.

### 4.5.1 Vorschriften Node Red

Grundsätzlich werden die Nodes in ein öffentliches npm (node packet manager) Repository gestellt. Sobald sie dort hinzugefügt sind, ermöglicht die Node Red Website das Hinzufügen des Packets zur Node Red library. Was und wie hochgeladen wird, ist im JSON Filfe definiert, welches anhand eines Beispiels im Kapitel 4.5.2 erläutert wird. Eine Vorschrift betrifft die Namensgebung des Packets. Entweder muss der Name einzigartig sein, oder unter einem gewissen Scope stehen. In der Ordnerstruktur, welche in Abbildung 29 zu sehen ist, muss ein README.md File existieren.

 CPU		File folder
 GPIO		File folder
 LICENSE		File
 package.json		JSON File
 README.md		MD File

Abbildung 29: Ordnerstruktur

In diesem sind Informationen über die Anwendungsmöglichkeiten sowie Voraussetzungen zur Verwendung des Pakets beschrieben. Den daraus resultierenden Webauftritt ist in der Abbildung 30 zu sehen.

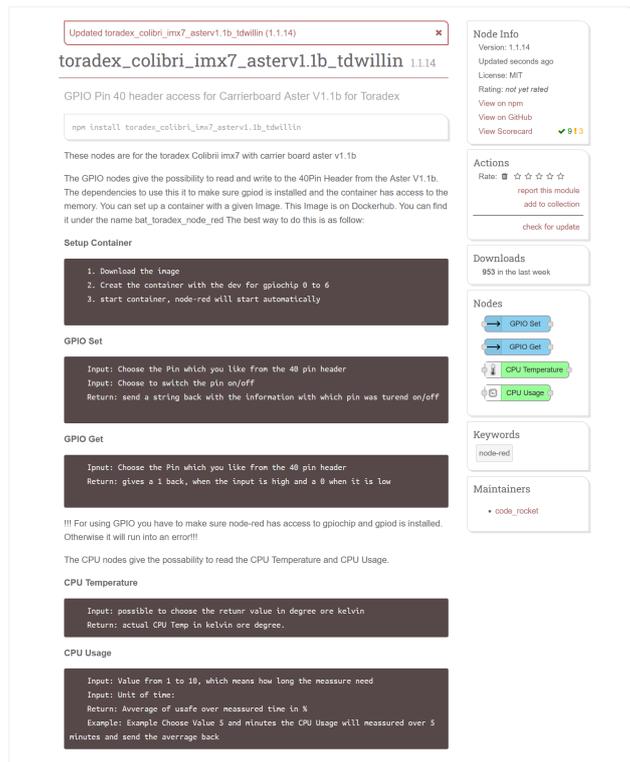


Abbildung 30: Webauftritt

#### 4.5.2 JSON

Das JSON File dient zum Upload und ist in Abbildung 31 zu sehen. Der Name ist dabei einzigartig gewählt und nicht einem Scope untergeordnet. In der dritten Zeile ist die Version des Packets. Bei einem Update muss die Version erweitert werden und kann nicht auf derselben bleiben. In der "description" ist eine Beschreibung des Packets. Dies ist ebenfalls zwingend erforderlich. Damit die Node Red Bibliothek erfolgreich das Paket hinzufügt, muss das Keyword „node-red“ vorhanden sein. Im Repository wird der Pfad zu den Quellcodes angegeben. In diesem Falle verwendete man Github. Anschliessend sind die Pfade zu den entsprechenden Java Script Files in der Ordnerstruktur angegeben. In diesem Paket existieren die entsprechende Files jeweils im Unterordner GPIO und CPU. Zum Schluss wird die Lizenz angegeben, unter welcher die Bausteine stehen.

```
{
  "name": "toradex_colibri_imx7_asterv1.1b_tdwillin",
  "version": "1.1.7",
  "description": "GPIO Pin 40 header access for Carrierboard Aster V1.1b for Toradex",
  "author": "Sandro Williner <sandro.williner@stud.hslu.ch>",
  "keywords": ["node-red"],

  "repository": {
    "type": "git",
    "url": "git+https://github.com/Taswilline/Node-red-test.git"
  },

  "node-red": {
    "nodes": {
      "GPIO": "GPIO/GPIO.js",
      "CPU": "CPU/CPU.js"
    }
  },
  "license": "MIT"
}
```

Abbildung 31: JSON File

### 4.5.3 Upload Vorgang

Das Hochladen auf das npm Repository erfolgt mit der Powershell. Dazu muss folgender Befehl ausgeführt werden und die Login Daten angeben.

```
PC> npm login
```

```
PC> Username: code_rocket
```

```
PC> Password: *****
```

```
PC> Email: sandro.williner@stud.hslu.ch
```

```
PC> enter one-time password: 123456
```

Nach erfolgreichem einloggen wird man in das Verzeichnis mit dem File package.json navigiert. In diesem wendet man folgenden Befehl an:

```
PC> npm publish
```

Durch publish wurden alle Files, welche im JSON definiert sind, ins Repository hochgeladen. Zum Schluss muss auf der Website von Node Red der Name des Moduls eingetragen werden. Ist alles korrekt eingestellt und verletzt keine Vorschriften, so werden die Module erfolgreich in die Bibliothek übernommen. Bis diese in der webbasierten Oberfläche ersichtlich werden, dauert es ca. 30 Minuten.

## 4.6 Container

Die Firma Toradex lagert fast alles in Container aus. Deshalb erweiterte man ein bestehendes Dockerfile, um ein fertiges Image mit Node Red bereitzustellen. Die entsprechende Erweiterungen sind in der Abbildung 32 zu sehen.

```
# Update and Upgrade
RUN apt update && apt upgrade

#install curl to get node-red
RUN apt-get install -y curl

#install build tools
RUN apt-get install -y build-essential

# install node-red
RUN yes | /bin/bash -c 'bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)'

#RUN apt-get install -y nodered
#RUN apt-get install nodered

#install procs to us vmstate for CPU Usage.
RUN apt-get install -y procs

#install stress to stress the CPU for testing CPU Usage
RUN apt-get install -y stress

# Use CMD to pass the GPIO bank line arguments
# Apalis iMX8 MXM3 pin 5 (Apalis GPIO3) is LSI0.GPIO0.IO12
# Apalis iMX6 MXM3 pin 5 (Apalis GPIO3) is GPIO2_IO6
#CMD [ "/bin/bash" ]

#run node-red automaticly after starting container
CMD ["node-red"]
```

Abbildung 32: Dockerfile

Folgender Befehl updatet das System und die Module

```
RUN apt update && apt upgrade
```

updatet das System und die Module. Curl wird im weitem verlauf des Dockerfiles benötigt und wird mit folgenden Befehl installiert:

```
RUN apt-get install -y curl
```

Anschliessend wird folgendens ausgeführt:

```
RUN apt-get install -y build-essentials
```

Dies fügt notwendige Module hinzu, um erfolgreich zu kompilieren. Nun wird Node Red wie folgt installiert:

```
RUN yes | /bin/bash -c 'bash <(curl -sL https://...
```

Dabei wird ein online Verzeichnis aufgerufen, welches ein bash File ausführt. Option Yes wird benötigt um die Fragen des Bash Files mit Ja zu beantworten. Der drittletzte Befehl wird angewendet um vmstat verfügbar zu machen. Die CPU kann im bereitgestellten Image manuell auf eine Auslastung mit dem Kommando „stress“ gesetzt werden. Dies wird im zweitletzten Befehl installiert.

```
CMD ["node-red"]
```

Dies ermöglicht Node Red automatisch zu starten, nachdem der Container gestartet wurde. Dadurch benötigt der User keine Kommandozeile. Das Dockerfile muss nun kompiliert werden. Dies geschieht mit dem Ausführen folgender Zeile in der Shell:

```
PC> docker build -t fussballch/bat_toradex_node_red:finish
```

Dies erstellt das Image mit dem Repository “bat\_toradex\_node\_red“ beim Benutzer fussballch. Finish ist dabei der Tag. Dieser erlaubt es, verschieden Images in einem Repository zu haben. Mit dem folgenden Befehl wird das Image auf Dockerhub hochgeladen:

```
PC> docker push fussballch/bat_toradex_node_red:finish
```

## 5 Anwendung

Im folgenden Kapitel sind Beispiele aufgezeigt, wie die Bausteine angewendet werden können sowie das Aufsetzen der Container mit Hilfe des zur Verfügung gestellten Image. Weiter zeigt das Beispiel die verschiedenen Möglichkeiten zum Hinzufügen der benutzerdefinierten Nodes. Anhand eines Beispiels werden alle Bausteine verwendet und in der Anwendung beschrieben.

### 5.1 Container Aufsetzen

Das Aufsetzen des Containers kann auf verschiedenen Arten erfolgen. Nachfolgend werden zwei Möglichkeiten aufgezeigt. Die erste Möglichkeit beinhaltet das Aufsetzen ohne Verwendung einer Kommandozeile und eine zweite Möglichkeit mit Verwendung dieser.

#### 5.1.1 Container Aufsetzen mit Portainer

Portainer ermöglicht eine visuelle Verwaltung der Container. Dabei handelt sich um eine webbasierte Oberfläche. Um diese aufzurufen muss im Webbrowser die entsprechende IP-Adresse mit dem Port 8840 angegeben werden. Der Befehl sieht wie folgt aus:

```
<IP-Adresse>:8840
```

Nun folgt ein Anmeldebildschirm und nach erfolgreicher Anmeldung ist die webbasierte Oberfläche, wie in Abbildung 33, zu sehen.

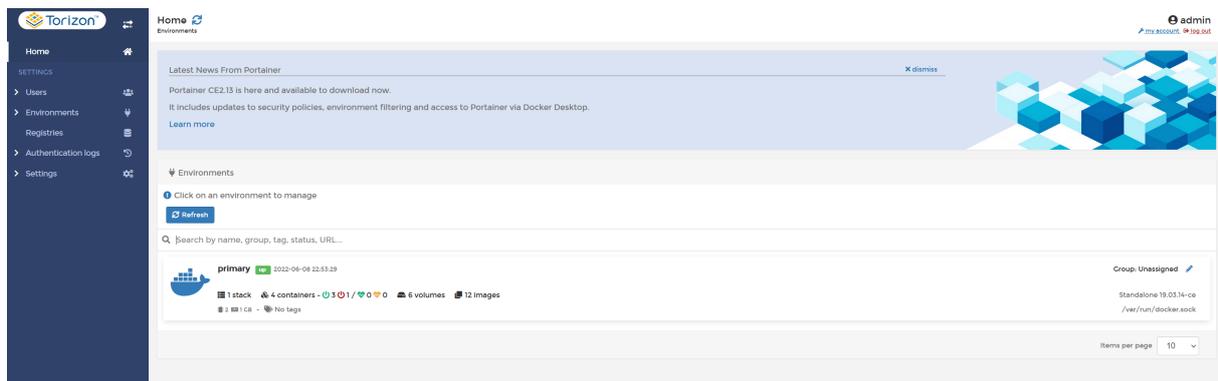


Abbildung 33: Portainer Übersicht

Durch klicken auf Primary landet man auf dem Dashboard. Das Untermenü Image erlaubt das Laden eines neuen Image, welches auf dem Web zu finden ist. Wie in Abbildung 34 zu entnehmen ist, kann das zur Verfügung gestellte Image mit:

fussballch/bat\_toradex\_node\_red:finish.

heruntergeladen werden. In diesem Beispiel hat man das Image mit dem Tag finish verwendet. Das geladen Image wird nun bei den verfügbaren Images gelistet. Dies ist Bild

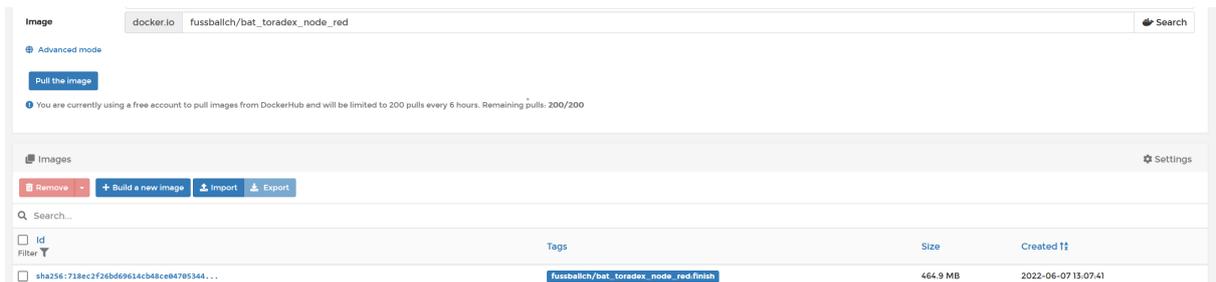


Abbildung 34: Portainer Image

ganz unten zu sehen. Im Untermenü Containers kann nun ein neuer Container mit diesem Image erstellt werden. Für das erfolgreiche Erstellen sind zwingend die Einstellungen anzupassen. Bei den Network Ports muss man manuell den Port 1880 einstellen. Dies ist in der Abbildung 35 beim Pfeil 1 zu sehen. Zusätzlich muss die Console als Interactive & tty bestätigt werden, dies entspricht dem Pfeil zwei. Die Port Einstellung erlaubt uns auf Node Red zuzugreifen, ähnlich dem Zugriff auf Portainer. Es benötigt wieder die IP-Adresse und der entsprechende Port 1880.

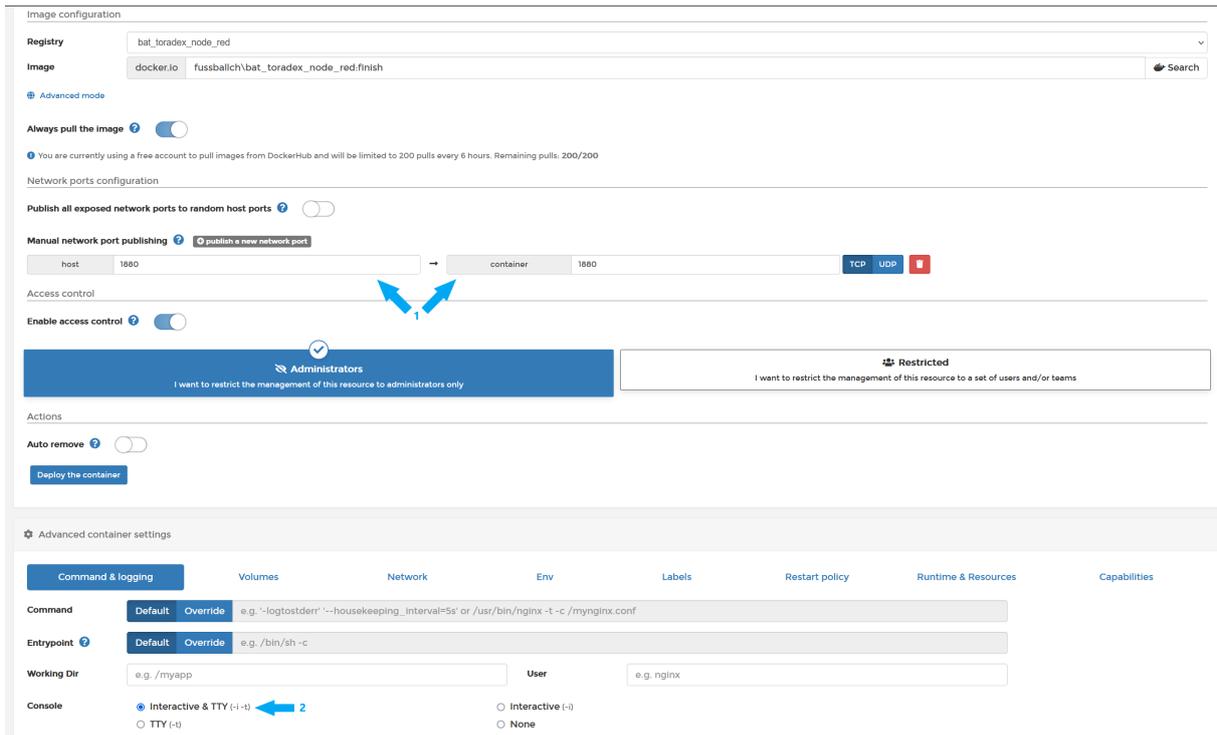


Abbildung 35: Portainer Container erstellen 1/2

Zusätzlich müssen devices geladen werden. Der GPIO Zugriff benötigt dies, ansonsten endet er in einem Error. Dazu wird in der Unterrubrik Runtime & Resources die devices hinzugefügt. Die Eingabe dafür ist in der Abbildung 36 beim Pfeil 2 zu sehen. Der Pfeil 1 dient zur Initialisierung. Sind alle Einstellungen entsprechend diesem Beispiel vorgenommen, kann der Container erstellt werden. Nun ist er bei den anderen Containern mit dem Status running gelistet und es kann auf Node Red zugegriffen werden. Der Zugriff erfolgt mit der Selben IP-Adresse wie beim Portainer, jedoch mit den Ports 1880. Ist der Container neu erstellt, oder wurde er neu gestartet, so kann erst nach ca. 30 Sekunden auf Node Red zugegriffen werden. Dies liegt am Startvorgang von Node Red.

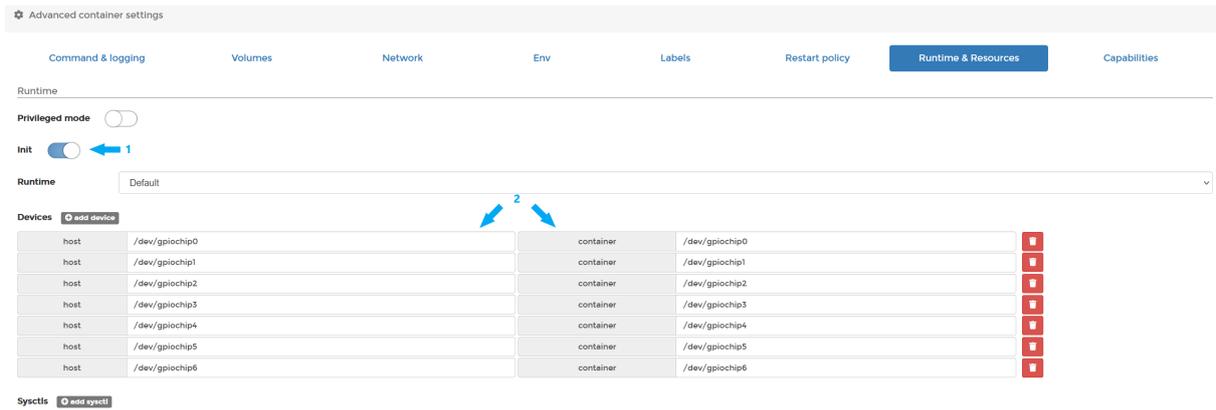


Abbildung 36: Portainer Container erstellen 2/2

### 5.1.2 Container aufsetzen Kommandozeile

Der gleiche Container kann auch mit der Kommandozeile aufgesetzt werden. Dazu benötigt man Putty, um eine SSH Verbindung mit dem SOM zu erstellen. In dieser Shell ist es nun möglich das Image mit folgendem Befehl herunterzuladen:

```
colibri-imx7-emmc:~$ docker pull fussballch/bat_toradex_node_red:finish
```

Das nun geladene Image kann mit dem docker run Befehl erstellt werden.

```
colibri-imx7-emmc:~$ docker run -it -p 1880:1880 --device /dev/gpiochip0
--device /dev/gpiochip1 --device /dev/gpiochip2 --device /dev/gpiochip3
--device /dev/gpiochip4 --device /dev/gpiochip5 --device /dev/gpiochip6
--name BAT_Sandro fussballch/bat_toradex_node_red:finish
```

Dadurch ist der Container nun per Kommandozeile fertig aufgesetzt und läuft. Der Zugriff auf Node Red erfolgt analog zuvor. Es muss im Webbrowser die IP-Adresse und der Port 1880 angegeben werden. Beim ersten Zugriff wird man auf der Node Red Oberfläche zudem durch einige Anleitungen begleitet.

## 5.2 Hinzufügen der Nodes

Um die Hardware auf dem SOM zu verwenden benötigt man die erstellten Funktionsbausteine. Diese sind nicht standardmässig in Node Red vorhanden, können aber über die Weboberfläche oder die Kommandozeile dazu installiert werden. Die hinzugefügten Bausteine sind unter separater Rubrik Toradex gelistet, entsprechend der Abbildung 37.

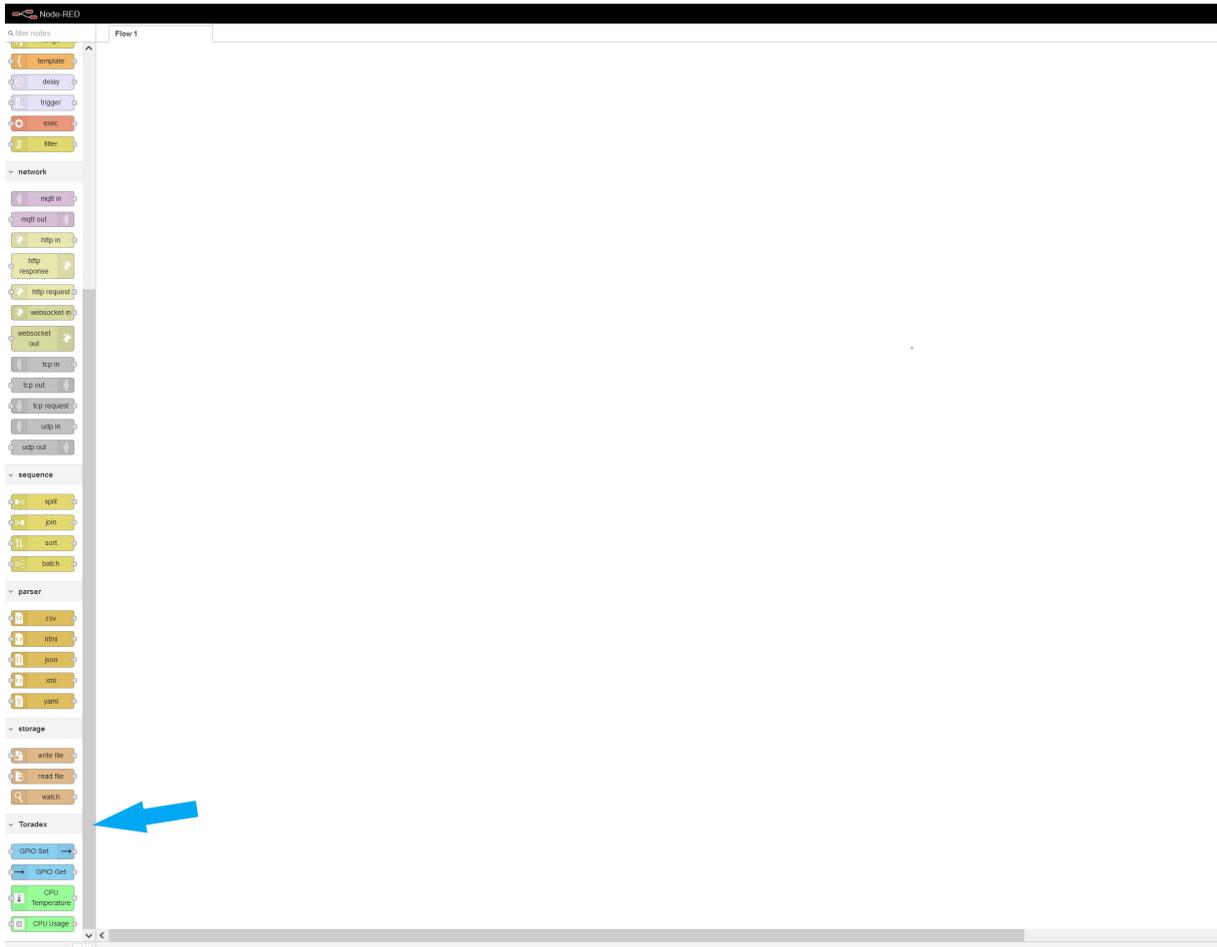


Abbildung 37: Toradex Bausteine in Node Red

### 5.2.1 Node Hinzufügen Weboberfläche

In Node Red gibt es die Möglichkeit zusätzliche Bausteine mithilfe Palettenmanager zu installieren. Dieser findet sich im Dropdown-Menü in der rechten oberen Bildecke hinter dem Hamburger-Symbol. Alternativ kann dieser auch mit dem Shortcut alt-shift-p geöffnet werden. In der Abbildung 38 ist das entsprechende Fenster zu sehen. Im Untermenü Install (Pfeil eins) kann nach einem Modul von Toradex gesucht werden. Dabei sollte das Paket „toradex\_colibri\_imx7\_asterv1.1b\_tdwillin“ gefunden werden. Die aktuelle Version kann sich dabei von diesem Bild unterscheiden. Durch betätigen des install Buttons (Pfeil zwei) wird das Modul installiert. Bei einem allfälligen Update kann dies ebenfalls

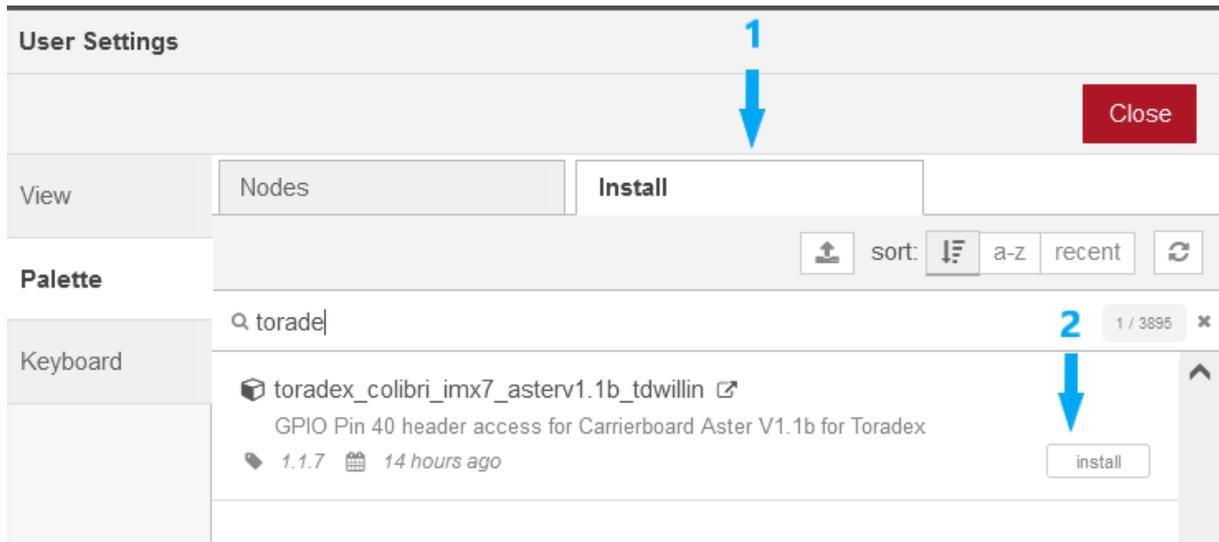


Abbildung 38: Palette Manager

mit dem Palette-Manager durchgeführt werden. Nach der erfolgreichen Installation ist das Paket im Untermenü nodes gelistet. In diesem sind alle installierten Pakte von Node Red aufgeführt. Die Abbildung 39 verdeutlicht dies.

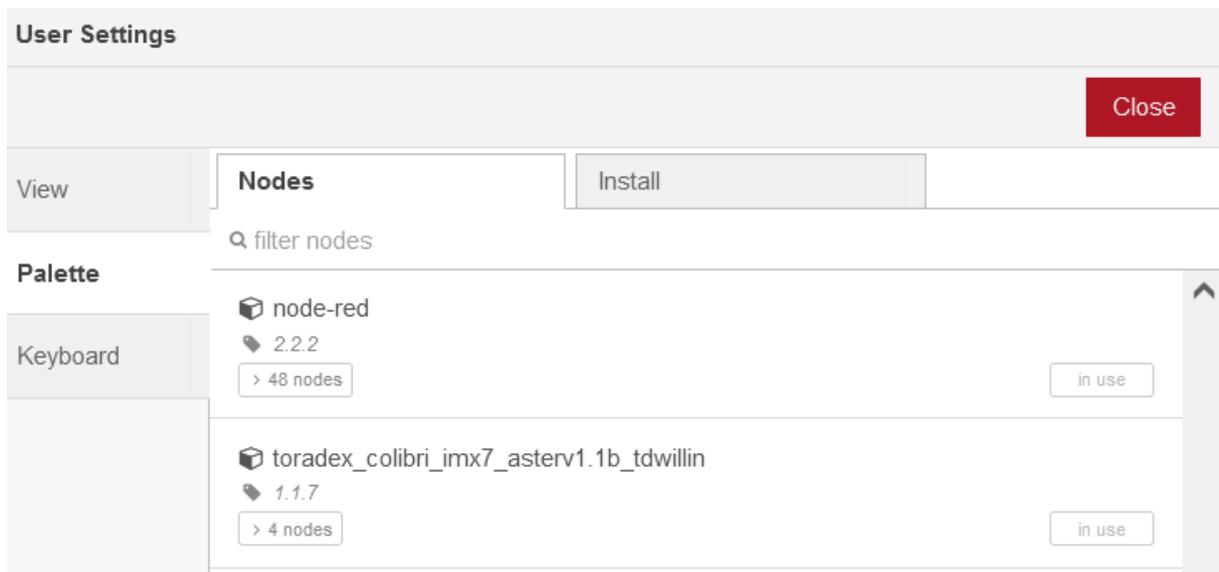


Abbildung 39: Palette Manager installierte Nodes

### 5.2.2 Node Hinzufügen Kommandozeile

Zusätzliche Node-Pakete lassen sich ebenfalls per Kommandozeile hinzufügen. Dazu muss zuerst eine Shell im Container aufgerufen werden. Dies ist auf der Portainer Website beim entsprechenden Container möglich. Eine andere Möglichkeit ist, von Putty auf das SOM zuzugreifen und von da in den Container. Dazu muss folgender Befehl ausgeführt werden.

```
colibri-imx7-emmc:$ docker exec -it <Name> /bin/bash
```

Der Name entspricht dem Namen beim erstellen des Containers. Weiss man diesen nicht mehr, so können die Container wie folgt gelistet werden:

```
colibri-imx7-emmc:$ docker ps
```

Anhand des Image ist nun der Name zu erkennen. Wurde beim Erstellen des Containers kein Name bestimmt, so wird ein zufälliger gewählt. Im Container wird mit dem unten stehenden Befehl das zusätzliche Pakete installiert.

```
Container:$ npm install toradex_colibri_imx7_asterv1.1b_tdwillin
```

### 5.3 Demo-Anwendung

Die Demo-Anwendung dient zur Illustration der Bausteine. Zum einen wird eine Benutzereingabe abgefragt und entsprechend einen GPIO-Pin angesteuert an dem eine LED ist. Zum andern wird die CPU-Auslastung kontinuierlich gemessen und im Falle einer Auslastung über 50% einen GPIO-Pin angesteuert der eine rote LED zum Leuchten bringt. Ebenfalls findet eine kontinuierliche Abfragung der CPU-Temperatur statt, welcher die Werte in Kelvin und Grad zurücksendet. Die Ausgabe ist in der entsprechenden Debugkonsole zu sehen. Abbildung 40 zeigt den Flow.

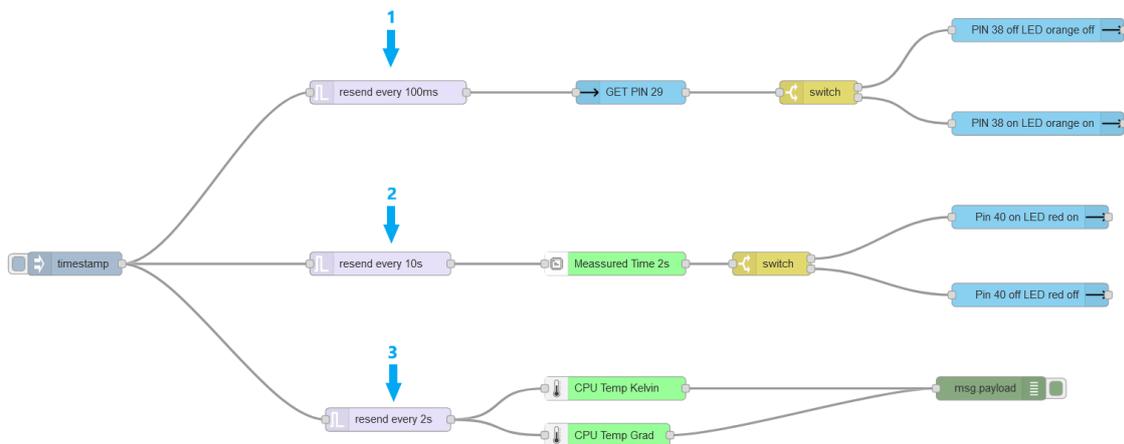


Abbildung 40: Demo Anwendug

Pfeil eins ist für die Abfrage des Joysticks zuständig und das entsprechende Ansteuern der orangen LED. Der erste Baustein in diesem Flow bewirkt, dass alle 100ms die Benutzereingabe abgefragt wird. Die dazu notwendigen Einstellungen sind in der Abbildung 41 zu finden.

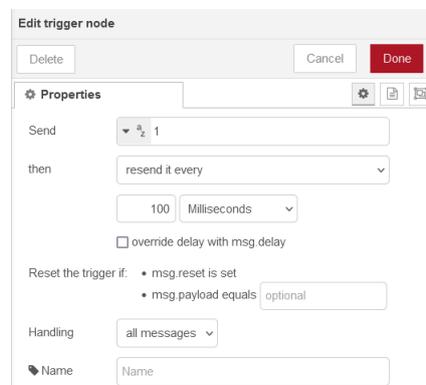


Abbildung 41: Baustein Trigger Einstellungen

Der GPIO-get-Baustein returniert den entsprechenden Wert und im Switch wird auf high und low geprüft, wobei high eins und low null entspricht. Die Einstellung für den GPIO-Get-Baustein sind in der Abbildung 42 zu entnehmen. Das Verwendete HAT zieht den PIN



Abbildung 42: Baustein GPIO get Einstellungen Demo

beim betätigen des Joysticks gegen GND, deshalb ist die Überprüfung gerade invertiert. Anschliessend wird der entsprechende Pin für die orange LED gesetzt. Diese leuchtet solange, bis der User die Taste nicht mehr betätigt. Die Einstellung des GPIO-set-Bausteins ist in der Abbildung 43 zu sehen

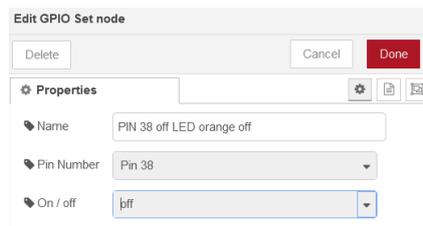


Abbildung 43: Baustein GPIO set Einstellungen Demo

Der Pfeil zwei zeigt den Flow für das Auslesen der CPU-Auslastung. Hier wird nach jeweils 10 Sekunden überprüft, wie gross die CPU-Auslastung über zwei Sekunden gemessen ist. Die entsprechenden Einstellungen am Baustein sind in der Abbildung 44 zu sehen.

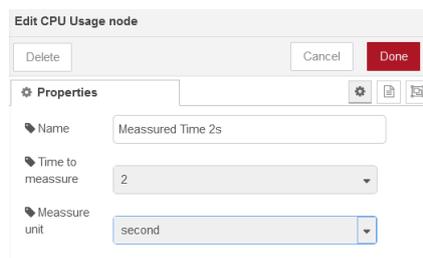


Abbildung 44: Baustein CPU Auslastung Einstellung Demo

Im Switch wird dieser Wert überprüft und falls er  $\geq 50$  ist, wird die rote LED auf dem HAT eingeschaltet. Ansonsten wird die rote LED ausgeschaltet. Die Einstellungen vom Switch-Baustein sind der Abbildung 45 zu entnehmen. Wichtig dabei ist das Prüfen auf eine Nummer und nicht auf einen String. Beim Einfügen hat dieser Baustein nur

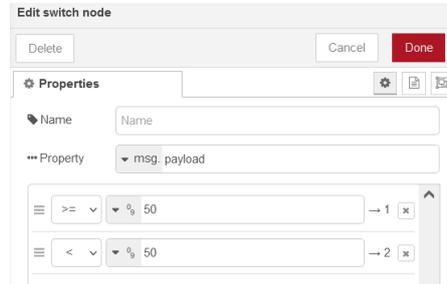


Abbildung 45: Baustein Switch Einstellung Demo

eine Überprüfungsmöglichkeit. Es können aber weitere Ports hinzugefügt werden. Aus der Sicht eines Programmierer ähnelt er sehr stark einem “switch-case-statement“. Der Pfeil drei in der Abbildung 40 korrespondiert mit dem dritten Flow. Dieser ist für das Auslesen der CPU Temperatur zuständig. Es werden zwei Baustein verwendet, da einer die Temperatur in Grad zurücksendet und einer in Kelvin. Die entsprechende Ausgabe ist im Debugfenster zu betrachten. Dies wird alle zwei Sekunden wiederholt.

## 6 Lösungsevaluation

Die im Kapitel 4 beschriebenen Lösungen wurden evaluiert und getestet. Diese Resultate sind in den nachfolgenden Kapiteln aufgezeigt. Der Evaluation ging ein Benutzertest voraus, welcher bereits einige Anpassungen an den Nodes bestimmte. Auf diese wird ebenfalls eingegangen sowie auf offene Punkte oder Probleme.

### 6.1 Benutzertests

In der Mitte des Projekts wurden Usertests mit drei Benutzern durchgeführt. Dabei haben die Benutzer ein Dokument abgearbeitet und versucht, eine kleine Anwendung zu realisieren. Die entsprechenden Dokumente und die dazugehörigen Notizen sind im Anhang [Verlinkung](#) zu finden. Ziel dieser Tests war es herauszufinden, ob die Kunden die Bausteine anwenden können und wie Benutzerfreundlich diese sind.

#### 6.1.1 GPIO Baustein Ansteuerung

In der Abbildung 46 ist die Auswertung der drei Benutzertests bezüglich einer erfolgreichen Anwendung des GPIO-Bausteins sichtbar. Daraus ist gut ersichtlich, dass der Baustein

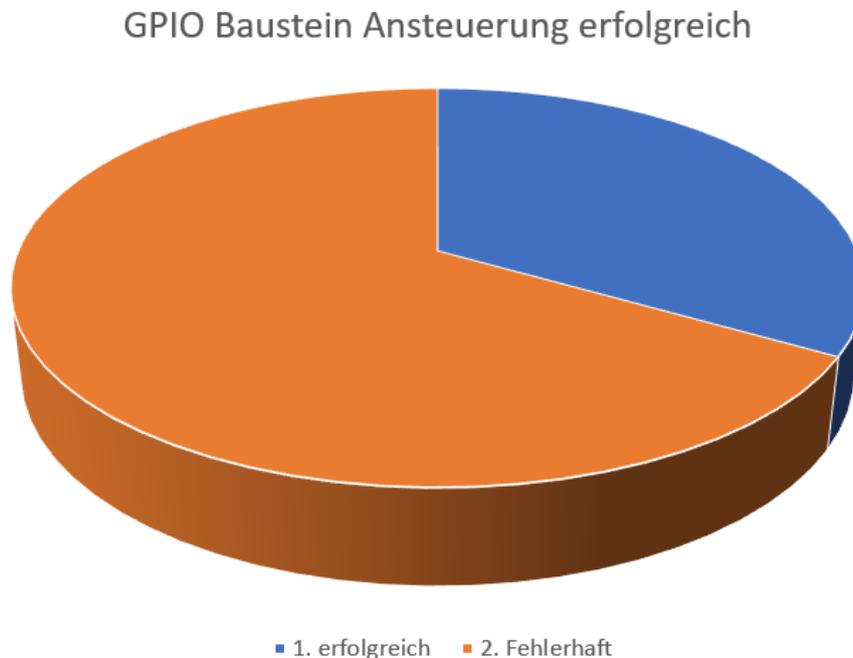


Abbildung 46: Benutzertest Rückmeldung GPIO Ansteuerung

den Anforderungen nicht genügt. So wurde zu diesem Zeitpunkt die Eingabe nicht über ein Dropdown-Menü gelöst, sondern mit einer normalen Texteingabe. Dies führte zu vielen kleinen Tippfehlern und die anschließenden Fehlermeldungen wurden nicht verstanden.

Aus diesem Grund wurden alle Bausteine mit einem Dropdown-Menü gelöst. So wird der User gut begleitet und es können bereits vorab potentielle Fehler vermieden werden. Als zusätzliche Massnahme wurde bei jedem Baustein eine Hilfestellung eingefügt. Dadurch kann der Nutzer bereits Informationen beim konfigurieren des Flows benützen. Zuvor wurden alle Hilfestellungen zentral im Readme.md File gespeichert. Um dieses zu Lesen musste aber die Node Red Bibliothek in einem separaten Browser Tab geöffnet werden.

### 6.1.2 Komfort der GPIO Bausteine

Die jetzige Lösung des GPIO-Bausteins führt das Kommando selber aus. Beim Test musste dies noch mit dem zusätzlichen Baustein “exec“ realisiert werden. Die Auswertung der Tests in Abbildung 47 zeigte, dass dies nicht zufriedenstellend war. Die aktuellen Bau-

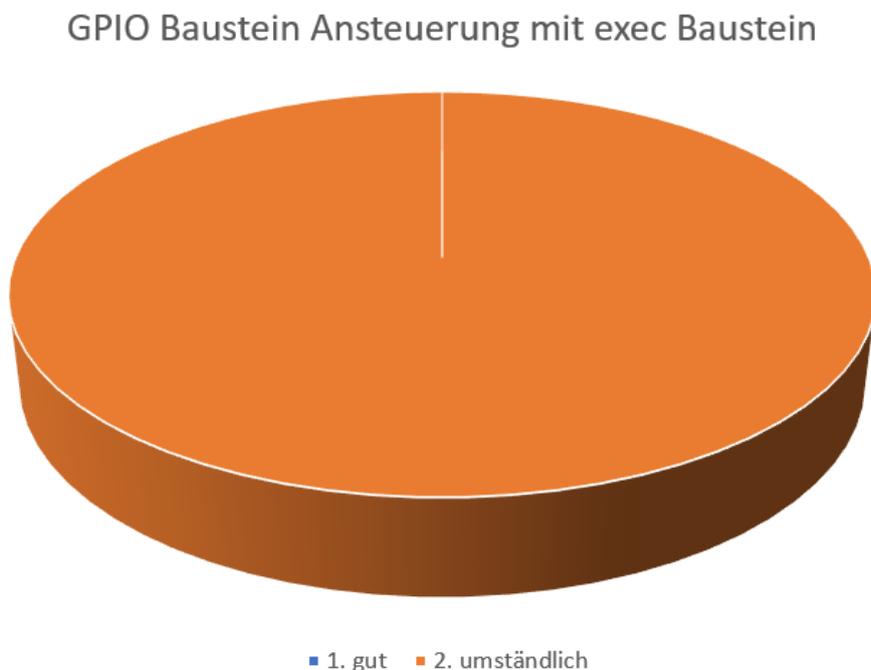


Abbildung 47: Benutzertest Rückmeldung GPIO mit exec

steine wurden alle mit dem “child\_process“ gelöst. Dabei wird eine Shell eröffnet. Die Ansteuerung per Shell müsste in einer weiterführenden Arbeit aber nochmals überdacht werden. Die Zuverlässigkeit bei grösseren Programmen und vielen Eingaben könnte diesbezüglich fraglich sein. Für kleine Anwendungen wurden jedoch keine Probleme festgestellt. Weiter zeigte die Auswertung, dass die Benutzer Mühe hatten die Bausteine zu lokalisieren, wie in Abbildung 48 zu sehen ist. Der Grund dafür war, dass die Bausteine zum bestehenden Menü Punkt “Common“ hinzugefügt wurden. Nun sind diese unter einem separate Menüpunkt “Toradex“ abgespeichert. Weiter sind die GPIO-Bausteine von den CPU-Bausteinen durch ihre Farbe unterscheidbar. Die Benutzer haben zusätzlich weitere persönliche Anregungen angebracht. Unter anderem folgende:

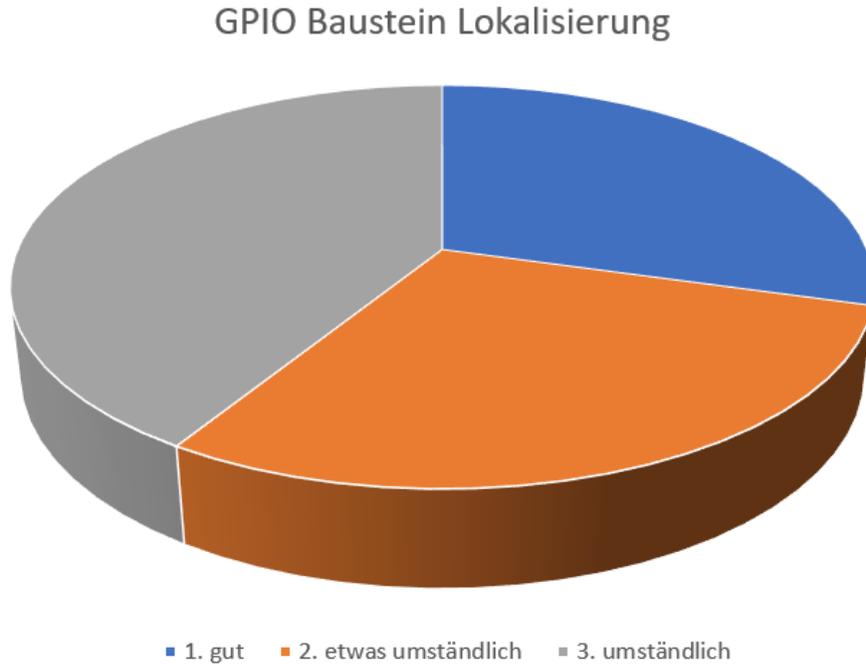


Abbildung 48: Benutzertest Rückmeldung GPIO Lokalisierung

- Bausteine mit Bilder
- wünschen keine Benützung der Kommandozeile
- Beispiele online stellen
- statische IP-Adresse

Einige dieser Punkte konnten erfüllt werden, jedoch nicht alle. Die Bausteinen besitzen ein Bild, welches eine Identifizierung anhand dieses Bildes ermöglicht. Das Aufsetzen der Container ist möglich, ohne eine Kommandozeile zu verwenden. Dazu wurde ein zusätzliches Dockerfile generiert. Beispiele wurden keine online gestellt, da die Hilfestellung erweitert wurde. Dadurch sollte die Anwendung klar sein. Dem SOM eine statische IP-Adresse zu vergeben ist schwierig. Mann kennt den Einsatzort nicht und evtl. ist diese bereits vergeben.

## 6.2 Zielerreichung anhand BAT Ziele

Die Ziele der Arbeit wurden bereits im Kapitel 3.3 beschrieben. Hier wird aufgezeigt, welche Ziele erreicht wurden und welche nicht. In der Tabelle 1 sind alle Ziele erfüllt. Der erste Meileinstein gab einen guten Einblick in das Thema und zeigte mögliche Schwierigkeiten auf. Die in der Tabelle 2 beschriebenen Kenntnisse wurden erarbeitet und konnten somit erfüllt werden. Die Anforderung F2.2 mit dem Abfragen der CPU-Temperatur wurde erfolgreich gelöst mit der zusätzlichen Funktion Kelvin oder Grad anzugeben. Die Ansteuerung der Hardware mittels GPIO konnte ebenfalls erreicht und getestet werden. Somit sind die Ziele F2.3 bis F2.5 erfolgreich abgeschlossen. Den Wunsch der Skalierbarkeit konnte durch das Dropdown-Menü im GPIO-Baustein erfüllt werden. Ein entsprechendes Video, wie die Bausteine angewendet werden, wurde nicht realisiert. In der Tabelle 3 konnte die Anforderung F3.1 der CPU-Auslastung erfolgreich abgeschlossen werden. Die PWM-Bausteine F3.2 wurden nicht realisiert. Im Verlauf des Projekts zeigte sich einen erheblichen Mehraufwand durch Container. So mussten zusätzliche Kenntnisse im Bereich Dockerfile erarbeitet werden. Anstatt des PWM-Bausteins wurde deshalb ein vorgefertigtes Image zur Verfügung gestellt. Dies ermöglicht das Benutzen der Bausteine für jeden Nutzer. Ein nicht zur Verfügungsstellung eines Images hätte zur Folge gehabt, dass einfache Nutzer keinen guten Zugang zum Produkt hätten. Die Bausteine wurden, auch wegen den Resultaten der Benutzertests, mit Hilfestellungen erweitert. Somit konnte F3.3 erfolgreich abgeschlossen werden. Eine Demoanwendung wurde realisiert, jedoch unterscheidet sie sich zur ursprünglich angedachten Demo Anwendung. Dies ist auf das Fehlen der PWM-Bausteine zurückzuführen. Die entsprechende Demo wurde angepasst und im Kapitel 5.3 beschrieben.

## 7 Schlusdiskussion

In diesem Kapitel werden persönliche Erfahrungen wiedergegeben und offene Punkte des Projekts zusammengefasst.

### 7.1 Erfahrungen

Mit dem abschliessen dieser Arbeit ist die BAT in ihrem Umfang abgeschlossen. Es steht noch eine Präsentation an, aber es werden keine neuen Bausteine geschaffen. Nachfolgend sind einige persönliche Erfahrungen und lesson-learned aufgezeigt.

Mit dem Erstellen des Überblickschemas konnte die Komplexität und das Zusammenspiel gut abgefangen werden. Dadurch wurde verhindert, das grosse Ganze während dem Projekt aus den Augen zu verlieren. Dank den frühen Benutzertests konnten einige Schwachstellen der Bausteine korrigiert werden. Dies ermöglichte bereits zu Beginn, grössere Fehler zu korrigieren und eine gute Grundlage für die weiteren Bausteine zu schaffen.

In vielen Bereichen konnten persönliche Kenntnisse vertieft und oder neu erworben werden. Zum einen erweiterte ich mein Programmierhorizont mit der für mich neuen Sprache Java Script. Die minimalen HTML Kenntnisse wurden durch diese Arbeit vertieft und gesichert. Durch das Hochladen der Bausteine in eine Bibliothek bin ich mit JSON vertieft in Kontakt gekommen. Dies vermittelte mir ein neues Verständnis von Upload-Vorgängen. Ebenfalls neue Kenntnisse konnte ich in der Anwendung mit Container erwerben. So erstellte ich zum ersten Mal ein eigenes Image eines Containers und machte es online verfügbar. Dadurch lernte ich Dockerhub und Dockerfile kennen. Die Aufgabenbreite der BAT war durch die vielen verschiedenen Komponenten erheblich. Dies wurde zu Beginn etwas unterschätzt, weshalb nicht alle Ziele erreicht werden konnten.

### 7.2 Offene Punkte

Die vorhanden Bausteine funktionieren und sind verständlich. Der Zugriff per event-loop hat sich bewährt. Er ermöglicht weiterhin einen asynchronen Ablauf und bremst Node-Red nicht aus. Die momentanen Bausteine dienen als sehr gute Grundlage zur Entwicklung von weiteren. Die Realisation eines PWM sowie I<sup>2</sup>C Bausteins sollte als nächstes angegangen werden. Dadurch würden diverse Sensoren abgedeckt und der Einsatzbereich erweitert. Dies könnte im Falle einer Markteinführung die Marktchancen erheblich erhöhen.

## Abbildungen

1	Node Red Oberfläche . . . . .	2
2	Node Red Flow . . . . .	3
3	Software Container Übersicht . . . . .	5
4	Double Diamond . . . . .	7
5	Systemübersicht . . . . .	11
6	Baustein CPU Temperatur . . . . .	12
7	HTML CPU Temperatur . . . . .	12
8	Node Red CPU Temperatur . . . . .	12
9	HTML CPU Temperatur JS Abschnitt . . . . .	13
10	JavaScrip CPU Temperatur . . . . .	14
11	Baustein CPU Auslastung . . . . .	15
12	HTML CPU Auslastung . . . . .	15
13	Baustein CPU Auslastung beschriftet . . . . .	16
14	HTML CPU Auslastung JavaScript 1/3 . . . . .	16
15	HTML CPU Auslastung JavaScript 2/3 . . . . .	16
16	HTML CPU Auslastung JavaScript 3/3 . . . . .	17
17	Java Script CPU Auslastung . . . . .	17
18	GPIO setzen . . . . .	18
19	HTML GPIO set . . . . .	19
20	GPIO set Node Edit . . . . .	19
21	HTML GPIO set JavaScript 1/2 . . . . .	20
22	HTML GPIO set JavaScript 2/2 . . . . .	20
23	Java Script GPIO set 1/2 . . . . .	21
24	Java Script GPIO set 2/2 . . . . .	22
25	GPIO lesen . . . . .	22
26	HTML GPIO get . . . . .	23
27	GPIO get node edit . . . . .	23
28	Java Script GPIO get . . . . .	24
29	Ordnerstruktur . . . . .	25

30	Webauftritt . . . . .	25
31	JSON File . . . . .	26
32	Dockerfile . . . . .	27
33	Portainer Übersicht . . . . .	29
34	Portainer Image . . . . .	30
35	Portainer Container erstellen 1/2 . . . . .	31
36	Portainer Container erstellen 2/2 . . . . .	32
37	Toradex Bausteine in Node Red . . . . .	33
38	Palette Manager . . . . .	34
39	Palette Manager installierte Nodes . . . . .	34
40	Demo Anwendug . . . . .	36
41	Baustein Trigger Einstellungen . . . . .	36
42	Baustein GPIO get Einstellungen Demo . . . . .	37
43	Baustein GPIO set Einstellungen Demo . . . . .	37
44	Baustein CPU Auslastung Einstellung Demo . . . . .	37
45	Baustein Switch Einstellung Demo . . . . .	38
46	Benutzertest Rückmeldung GPIO Ansteuerung . . . . .	39
47	Benutzertest Rückmeldung GPIO mit exec . . . . .	40
48	Benutzertest Rückmeldung GPIO Lokalisierung . . . . .	41

## Tabellenverzeichnis

1	Anforderungen Meilenstein 1 . . . . .	8
2	Anforderungen Meilenstein 2 . . . . .	9
3	Anforderungen Meilenstein 3 . . . . .	10

# A Anhang

## A.1 Aufgabenstellung

Lucerne University of  
Applied Sciences and Arts

**HOCHSCHULE  
LUZERN**

Technik & Architektur

Horw, 21. Februar 2022  
Seite 1/2

### **Bachelor Thesis im Studiengang Elektrotechnik und Informationstechnologie**

#### **Aufgabe für Herrn Williner Sandro**

#### **Node-RED für Toradex Embedded-Linux-Module**

##### **Fachliche Schwerpunkte**

Technische Informatik, Embedded Linux Programmierung, Analyse

##### **Einleitung**

Node-RED ist eine visuelle Programmierumgebung, mit dem sich einfach IoT-Prototypen erstellen lassen. Dazu werden Funktionsbausteine (sogenannte Nodes) graphisch zu einem Flow verbunden. Nodes gewähren Zugang zu Hardwarekomponenten, online (Web) Services und selbst erstelltem JavaScript Code. Diese Eigenschaften machen Node-RED interessant für Anwender, die über wenig Erfahrung im Programmieren verfügen.

##### **Aufgabenstellung**

Toradex entwickelt kleinste Linux-basierte Computerplattformen (System-on-Modules, SoM), welche von ihren Kunden u.a. für Steuerungs- und Anzeigeaufgaben eingesetzt werden. Die Aufgabe dieser Arbeit liegt in der Entwicklung von systemspezifischen Funktionsbausteinen für Node-RED. Diese Funktionsbausteine sollen Anwendern einfachen Zugriff auf die Modul-spezifischen Hardware-Interfaces (GPIOs, I2C, Analog-in, etc.) und Systemdaten (Systemtemperatur, Prozessorlast, etc.) erlauben.

##### **Termine**

Start der Arbeit:	Montag, 21.2.2022
Zwischenpräsentation:	Zu vereinbaren im Zeitraum 11.4. – 6.5.2022
Abgabe Schlussbericht:	Freitag, 10. Juni 2022, vor 16:00 im D311
Abgabe Digitale Doku:	Gemäss separater Anweisung der Studiengangleitung
Abschlusspräsentation:	Zu vereinbaren im Zeitraum 13.6. – 1.7.2022
Diplomausstellung:	Freitag, 8. Juli 2022 (Teilnahme obligatorisch!)

FH Zentralschweiz

Horw, 21.2.2022  
Seite 2/2  
Diplomarbeit im Fachbereich  
Elektrotechnik und Informationstechnologie

## Dokumentation

Der gebundene Schlussbericht enthält keine Selbständigkeitserklärung und ist in einfacher Ausführung zu erstellen. Er enthält zudem zwingend

- einen sehr kurzen, englischen Abstract.
- Ein Titelblatt, gleich hinter dem Deckblatt, gemäss Weisungen der Studiengangleitung
- Eine SD-Hülle, innen, auf der Rückseite des Berichtes für den Betreuer

Alle Exemplare des Schlussberichtes müssen komplett und termingerecht gemäss Angaben der Studiengangleitung abgegeben werden. Zusätzlich muss eine SD-Speicherkarte mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgegeben werden.

Die gesamte Dokumentation ist zudem gemäss Anweisungen der Studiengangleitung elektronisch auf einen Server zu laden. Sämtliche abzugebende Teile der Dokumentation sind Bestandteile der Beurteilung.

## Fachliteratur/Web-Links/Hilfsmittel

### Geheimhaltungsstufe:

Öffentlich

### Verantwortlicher Dozent/Betreuungsteam, Industriepartner

**Dozent** Dr. Oliver Kasten oliver.kasten@hslu.ch

**Industriepartner** Toradex AG  
Ebenastrasse 10  
6048 Horw

Hr. Daniel Lang  
daniel.lang@toradex.ch Tel. +41 77 407 59 99

### Experte

Dr. Marc Wegmüller  
Altran AG,  
Hardturmstrasse 253, 8005 Zurich, Switzerland  
marc.wegmueller@altran.com Tel. +41 58 122 12 70

Hochschule Luzern  
Technik & Architektur

Dr. Oliver Kasten

## A.2 Meilensteine

BAT\_FS\_2022

Firma Toradex

Sandro Williner

### Projektziel Toradex und Abgrenzung BAT

Version	Bearbeitet	Beschreibung	Datum
V1.0	Sandro Williner	Initial Entwurf	08.03.2022

#### Projektziel Toradex

Die Firma Toradex entwickelt System on Modules (SOM) für folgende Bereiche:

- Industriearomatisierung
- Transport
- Test und Messungen
- Smart City

Um diesen Kunden einen vereinfachten Zugang zur Hardware zu ermöglichen, soll eine Webbasierte Oberfläche mit Bausteinen entstehen. Mit Hilfe von diesen Bausteinen ist der Kunde möglich selbst kleine Anwendungen zu realisieren, ohne dabei fundiere Kenntnisse in der Programmierung zu haben. Diese Oberfläche soll mittels Node-RED realisiert werden.

#### Abgrenzung BAT

Das Projektziel der Firma Toradex ist ambitioniert und nicht in einer einzigen BAT umsetzbar. Aus diesem Grund wird die BAT in Phasen unterteilt **wobei das Erreichen und Abschliessen der Phase 3 das Ziel ist**. Nachstehend sind diese Phasen in Prosa kurz erläutert. Damit das Erreichen einer Phase spezifiziert werden kann, dient der anschliessende Anforderungskatalog.

##### Phase 1: Inbetriebnahme Carrier Board

Mit Abschluss dieser Phase ist das Carrier Board erfolgreich in Betrieb. Es funktionieren alle Anwendungen und es konnte eine erste eigene Node erstellt werden. Dies bildet die Grundlage für die weiterführende Phasen.

Anforderung	Art: Fest: F Wunsch: W	Beschreibung / Definition	Status Erfüllt: E Teilweise: TE Nicht Erfüllt: NE
F1.1	F	Es ist die entsprechende Hardware beschafft: <ul style="list-style-type: none"> <li>• Bildschirm VGA</li> <li>• Ethernet</li> <li>• Carrier Board</li> <li>• Aster Board</li> </ul>	E
F1.2	F	Quickstart Guide von Toradex erfolgreich abgearbeitet <a href="https://bit.ly/34q3as6">https://bit.ly/34q3as6</a>	E
F1.3	F	Erstes Beispiel Node-RED von Toradex soweit möglich angewendet (zusätzliche HW wird nicht beschafft) <a href="https://bit.ly/3pQhotU">https://bit.ly/3pQhotU</a>	E
F1.4	F	Hello World in Node-RED	
F1.5	F	Erstes Beispiel Creat your first Node gemeistert <a href="https://bit.ly/3Ct057b">https://bit.ly/3Ct057b</a>	
W1.6	W	Es werden entsprechende Schutzmassnahmen für ESD getroffen: <ul style="list-style-type: none"> <li>• ESD Schutzmatte</li> <li>• Guide Line erstellen im Umgang mit ESD</li> </ul>	TE

09.03.2022

1

## Phase 2: Zwei Funktionsbausteine in Node-RED erstellt

Um die Phase 3 erfolgreich zu meistern werden erste Funktionsbaustein erstellt. Diese sind dokumentiert, um weitere in Phase 3 gemäss diesen Beispielen zu erstellen. Ein Funktionsbaustein soll dabei Informationen vom Internen CPU ablesen (BSP: CPU Temperatur). Ein zweiter Baustein soll das Auslesen und Setzen eines GPIO Pins ermöglichen.

F2.1	<b>F</b>	Die benötigten Fachkenntnisse in JS, JSON und HTML sind vorhanden	
F2.2	<b>F</b>	Es kann erfolgreiche die CPU abgefragt werden, wobei die erhaltenen Informationen noch nicht gegliedert und ausgewertet sind	
F2.3	<b>F</b>	Aus den CPU Informationen wird nur die Temperatur ausgewertet und eine Weiterverarbeitung ermöglicht	
F2.4	<b>F</b>	GPIO Wert low, high kann gelesen werden	
F2.5	<b>F</b>	GPIO Wert low, high kann gesetzt werden	
F2.6	<b>F</b>	Die vorhandenen Bausteine werden mit Benutzer (3 Stk.) auf die Einfachheit und Verständlichkeit getestet. Dabei müssen die Nutzer folgende Kenntnisse mitbringen: <ul style="list-style-type: none"> <li>• Grundkenntnisse in Englisch (Hilfestellungen werden in Englisch angezeigt)</li> <li>• Grundverständnis für technische Anwendungen <ul style="list-style-type: none"> <li>⇒ Ist im entsprechenden Absatzmarkt oder vergleichbaren Absatzmarkt tätig</li> </ul> </li> </ul>	
W2.7	<b><u>W</u></b>	Die Funktionsbausteine wie GPIO sind skalierbar: <ul style="list-style-type: none"> <li>⇒ Es kann in einem Baustein den entsprechenden GPIO Pin gewählt werden</li> </ul>	
W2.8	<b>W</b>	Es wird in einem Video dokumentiert wie die entsprechenden Funktionsbausteine zu verwenden sind	

## Phase 3: Die Grundlegendsten Funkitonsbausteine erstellen

Mit Abschluss der Phase 1 + 2 bestehen die notwendigen Kenntnisse, um auf die HW zuzugreifen und weitere Funktionsbausteine zu erstellen. Mit Abschluss dieser Phase ist es dem Benutzer möglich erste kleine Anwendungen mittels Node-RED zu realisieren. Dies wird mit einer einfachen und kleine Demoanwendung simuliert.

F3.1	<b>F</b>	Erstellung Funktionsbaustein Abfrage der Prozessorlast	
F3.2	<b>F</b>	Erstellung Funktionsbaustein PWM: <ul style="list-style-type: none"> <li>⇒ PWM setzen</li> <li>⇒ PWM abfragen</li> </ul>	
F3.3	<b>F</b>	Aus den CPU Informationen wird nur die Temperatur ausgewertet und eine Weiterverarbeitung ermöglicht	
F3.4	<b>F</b>	Kleine Demo Anwendung: gemäss separatem Beschrieb unterhalb der Tabelle	
W3.5	<b>W</b>	Funktionsbaustein für einen I <sup>2</sup> C: <ul style="list-style-type: none"> <li>⇒ Senden und empfangen von Daten</li> </ul>	
W2.7	<b><u>W</u></b>	Die Funktionsbausteine wie GPIO sind skalierbar: <ul style="list-style-type: none"> <li>⇒ Es kann in einem Baustein den entsprechenden GPIO Pin gewählt werden</li> </ul>	

### Demo Beschrieb:

Ziel der Demo ist es die Funktionsbausteine zu erklären und deren Richtigkeit aufzuzeigen. Eine Erste Idee der Demo ist folgende:

- Anhand der Prozessorauslast wird ein Leuchtmittel angesteuert, deren Helligkeit die Auslastung symbolisiert. Ist diese über 80% blinkt das Leuchtmittel
  - ⇒ Die dazu notwendige HW kann aus der PAIND Arbeit verwendet werden
- Mittels RGB wird eine visuelle Statusrückmeldung betreffend Prozessor Temperatur angezeigt.
  - ⇒ HW dazu muss noch angeschafft werden.

Mit Abschluss der Phase 3 ist die BAT in ihrem Umfang abgeschlossen. Die weiteren Phasen dienen vorausschauend und dienen der Idee wie es weiter gehen kann. Deshalb wurde auf genaue Definitionen mit der Tabelle verzichtete. Sollte es zu einer 4. Phase kommen, so wird diese zusätzlich definiert.

In der Phase 3 wird aus Zeitgründen nicht nochmals eine Benutzeranalyse gemacht. Da aber die Funktionsbausteine nach dem selbe Prinzip wie in der Phase 2 erstellt wurden, wird angenommen diese neuen sind ebenfalls verständlich.

#### Phase 4: Umfassende Benutzeranalyse mit allen Funktionsbausteinen

Damit die Annahmen aus der Phase 3 überprüft werden könne, wird eine weitere nun umfassendere Benutzeranalyse erstellt. Die Kenntnisse aus dieser beschreibt den weitere Verlauf der Phase 4, da die Feedbacks integriert werden sollen

#### Phase 5: Erstellung einer GUI Oberfläche zum Anzeigen von Daten

Mit der Erstellung einer GUI Oberfläche werden viele weiter Möglichkeiten betreffend Anzeigemedium ermöglicht. Einige Ideen dazu wären folgende:

- Diagramm über vergangene Auslastungen/ Temperaturen
- Ermöglicht Bezug zu Anwendungen welche Auswirkungen Sie haben.
- Daten werden in einem File abgespeichert, um lange Zeiträume abzudecken.

#### Phase 6: Steuerung mittel GUI

Es werden weitere Funktionsbausteine erstellt, um GUI Inputs zu verarbeiten. Schwierigkeit dabei wird es sein, dies gemäss Projektziel des Unternehmens für einfache Kunden zu ermöglichen. Evtl. ist eine Machbarkeit zu erstellen ob und falls ja, wie dies umgesetzt werden könnte.

**Phase 5 und folgende werden definitiv nicht in dieser BAT erreicht.**

## A.3 Benutzertest mit Notizen

*frady*

### Benutzertest BAT Node-Red

#### Starten des Node-Red

Node-Red wird als ein Container gestartet. Sie können sich einen Container so vorstellen, dass es eine für sich stehende Box in einem Betriebssystem ist. Dadurch ist das System gut geschützt und bei einem allfälligen Fehler kann einfach der Container neu erstellt werden.

Um Node-Red zu starten bearbeiten Sie bitte die nachfolgenden Schritte:

1. Toradex Modul am Ethernet anschliessen
2. Toradex Modul mit Spannung versorgen

Sie haben nun das Modul erfolgreich gestartet, um den Container zu starten gehen Sie wie folgt vor:

1. Auf Portainer zugreifen: Browser öffnere (BSP. Firefox) und die IP Adresse und Port eingeben  
=> 192.168.0.254:8840
  - a. Fall Sie die IP-Adresse nicht wissen, dann können Sie dies in Ihrem Router Interface nachsehend.
    - i. Bei UPC die UPC Connect App verwenden und die angeschlossenen Geräte nach colibri-imx7 suchen
    - ii. Sie können nun die IP-Adresse ablesen
    - iii. Bei Swisscom können Sie dies ebenfalls tun indem Sie im Browser die IP-Adresse 192.168.1.1 eingeben.
    - iv. Der Benutzername lautet standardmässig admin und das Passwort können Sie in Ihrem Kundencenter herausfinden
  - b. Der Port ist bei allen der Gleiche
  - c. Nun die Ihnen ausgehändigten Login Daten verwenden
2. Klicken Sie nun auf primary

*Wurde  
erfolgreich  
geschafft*



3. Starten Sie nun den Node-Red Container indem Sie auf batsandro klicken und anschliessend oben auf den grünen start Button:



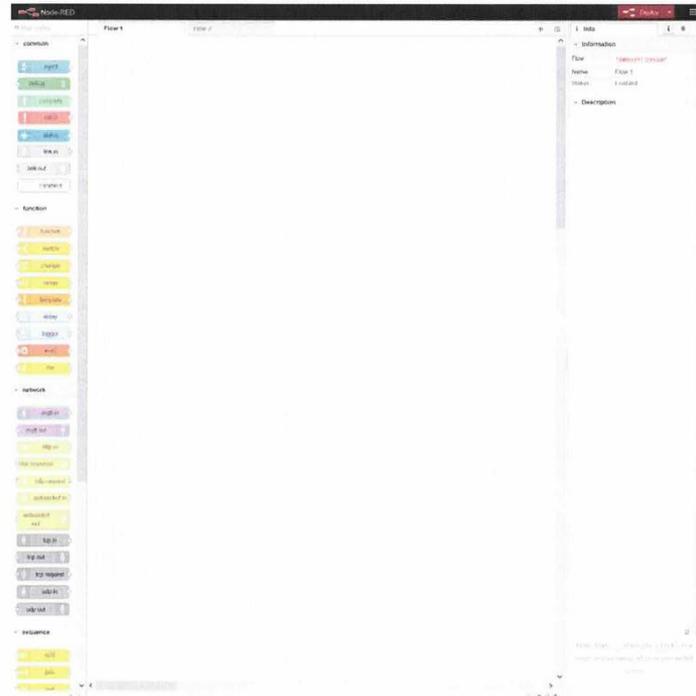
4. Durch Klicken in der Menüleiste auf Container sehen Sie ob der Container erfolgreich gestartet wurde. Durch die Taste F5 können Sie das Fenster wieder aktualisieren. Nach einer Zeit sollten sie folgendes sehen:

Gratulieren Sie haben den Container erfolgreich gestartet.

5. Sie können nun durch die ausgelesene IP-Adresse in diesem BSP 192.168.0.254 und dem Port 1880 auf Node-Red zugreifen: Geben Sie also folgendes im Browser ein wobei die <> weggelassen werden:

a. <Ihre IP-Adresse>:1880

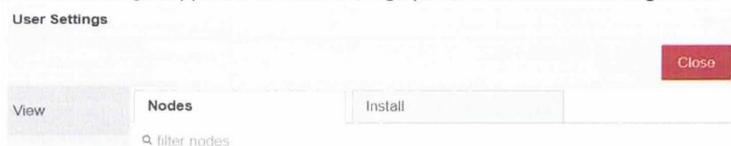
Sie sehen nun die Oberfläche von Node-Red:



### Einfügen der spezifischen Bausteine für das Toradex Modul

Damit wir die Hardware verwenden können müssen wir GPIO (General Purpose Input Output) Bausteine hinzufügen. Um dies zu erreichen befolgen Sie die nächsten Schritte:

6. Klicken Sie oben Rechts auf das Burger Symbol (neben dem Roten Deploy Button)
7. Im dem nun aufgeklappten Menu auf **Manage palette**. Sie sehen nun folgendes:

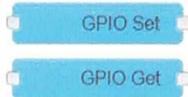


8. Gehen Sie auf Install und suchen nach **toradex**

9. Installieren Sie nun dies durch klicken auf den install Button



10. Sie finden nun die neu eingeführten Bausteine links im Menu:



Glückwunsch Sie haben die Bausteine zum lesen und setzen der für den Hardwareausgang erfolgreich hinzugefügt:

### Erste kleine Anwendung:

Nachfolgend schalten wir eine LED ein. Dafür muss etwas gesetzt werden. Wir verwenden also den GPIO Set Baustein.

1. Ziehen Sie per Drag and Drop den Baustein in den Flow:

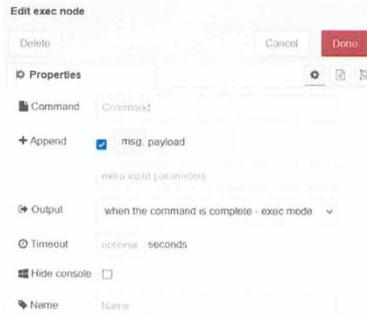


2. Durch Doppelklicken auf diesen Baustein können Sie nun Einstellungen vornehmen:
  - a. Tippen sie bei Pin Number den entsprechende Pin ein: **(in unserem Fall möchten wir die Rote LED ansteuern welche auf dem Pin 40 liegt)**. Gebe Sie also Pin 40 ein.
  - b. Bei On / off müssen wir noch beschreiben ob wie die Led einschalten oder ausschalten möchten. Tippen Sie dafür **on** ein um die LED einzuschalten. **Wir haben nun den spezifischen Hardware stein korrekt eingestellt!**
3. Um in Node-Red einen Flow zustarten benötigt man einen Inject Button. Diesen finden Sie ebenfalls bei den Bausteinen.
4. Damit die Einstellungen beim GPIO Set Baustein auch ausgeführt werden müssen Sie noch einen exec (execute) einfügen.
  - a. In diesem müssen wir eine kleine Anpassung vornehmen damit er den Code korrekt ausführt klicken Sie dafür doppelt auf exec.

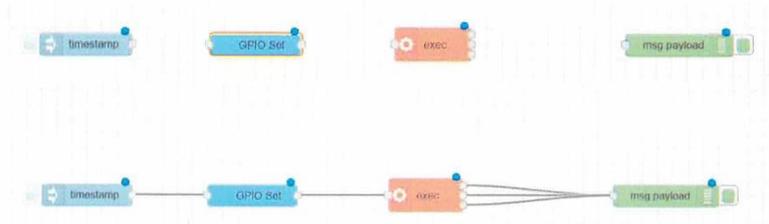
*Probleme bei ansteuern  
Pin 40 sein  
mit Error überfordert*

*Frage kam warum Baustein bereits eingefügt?*

- i. Setzen sie nun bei Append den Hacken wie unten zu sehen:



5. Zum Schluss möchten wir noch eine Ausgabe von evtl. Fehlern sehen. Ziehen sie dafür den debug Baustein in den Flow. Sie sollten nun folgendes Bild vorsich haben:

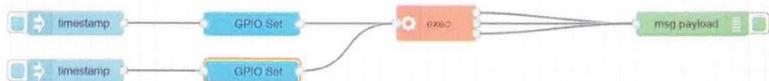


6. Nun müssen diese Bausteine noch miteinander kommunizieren. Dafür werden nun Linien vom jeweiligen Ausgangspunkt auf den nächsten Eingangspunkt angewendet. Das Resultat sehen Sie unten:  
 7. Nun haben Sie alles erfolgreich konfiguriert. Damit nun aber das Programm/Flow funktioniert müssen wir ihn zuerst noch deployen. Klicken Sie dafür auf den roten Deploy Button oben rechts.

**Hinweis: Wenn der Button Rot leuchtete wurden Änderungen vorgenommen und es muss neu deploy werden um die Änderungen auch Anwenden zu können.**

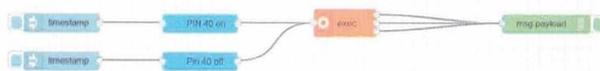
Sie können nun durch klicken auf das Quadrat links vom timestamp Button den Flow ausführen. Dabei wird der Flow einmal ausgeführt. Sie sehen nun auf dem Board eine rote LED die leuchtet.

8. Erstellen sie nun einen 2 Flow um die Gleiche LED auszuschalten. Das Resultat sehen Sie unten:



*Wieder Probleme bei Ansteuern ~~von~~ Off geschrieben*

9. Um den Überblick im Programm zu behalten lohnt es sich sie Bausteine jeweils passend zu beschriften. Sprich wir wissen durch die momentane Ansicht nicht welchen Pin wir verwenden. Deshalb beschriften wir die Bausteine.
  - a. Dies können sie würde im Baustein selbst machen in dem Sie bei Namen etwas für Sie passendes eingeben. BSP: PIN 40 on und PIN 40 Off
    - i. Stellen sie beim Anpassen der Namen sicher, dass die zuvor eingestellten Parameter noch stimmen.



### Erstellen eines eigenen kleinen Programms.

Versuchen Sie nun ein folgendes Programm zu erstellen mit den folgenden Infos. Sie müssen dabei das eben gelernte versuchen anzuwenden. Das Ziel des Programm ist durch das lesen eines Pins eine entsprechende LED einzuschalten. Dafür benötigen wir 2 LEDs und den Steuerknopf.

Die Pin Belegung ist wie folgt:

1. LED Rot => Pin 40
2. LED Gelb => Pin 38
3. Steuerbutton links => Pin 31
4. Steuerbutton rechts => Pin 29



Das Programm soll dabei jeweils den entsprechende Pin lesen und solange der Pin sprich der Steuerbutton gedrückt ist soll die entsprechende Led leuchten. Die Zuordnung soll wie folgt sein:

- Steuerbutton link => LED Rot
- Steuerbutton rechts => LED Gelb

### Tipps

Um eine Button kontinuierlich auszulesen gibt es einen Baustein bei der Untergruppe Funktion mit dem namen **trigger**. Diesen können sie so einstellen dass er nach einer gewissen Zeit immer wieder etwas tut. Als Zeit für eine Benutzer eingabe ist 100ms geeignet.

Um zu entscheide ob der Schalter gedrückt ist eignet sich die switch Function. Dort kann ein Input auf bestimmte Werte überprüft werden und ein entsprechender Output angesteuert werden.

Alles Bausteine haben zudem Inforamtionen die einem sehr hilfreich sind, konsolidiern Sie diese in Node-red falls sie etwas nicht verstehen.

# Benutzertest BAT Node-Red

## Starten des Node-Red

Node-Red wird als ein Container gestartet. Sie können sich einen Container so vorstellen, dass es eine für sich stehende Box in einem Betriebssystem ist. Dadurch ist das System gut geschützt und bei einem allfälligen Fehler kann einfach der Container neu erstellt werden.

Um Node-Red zu starten bearbeiten Sie bitte die nachfolgenden Schritte:

1. Toradex Modul am Ethernet anschliessen
2. Toradex Modul mit Spannung versorgen

Sie haben nun das Modul erfolgreich gestartet, um den Container zu starten gehen Sie wie folgt vor:

1. Auf Portainer zugreifen: Browser öffnere (BSP. Firefox) und die **IP Adresse** und **Port** eingeben  
=> **192.168.0.254:8840**
  - a. Fall Sie die IP-Adresse nicht wissen, dann können Sie dies in Ihrem Router Interface nachsehend.
    - i. Bei UPC die UPC Connect App verwenden und die angeschlossenen Geräte nach colibri-imx7 suchen
    - ii. Sie können nun die IP-Adresse ablesen
    - iii. Bei Swisscom können Sie dies ebenfalls tun indem Sie im Browser die IP-Adresse 192.168.1.1 eingeben.
    - iv. Der Benutzername lautet standardmässig admin und das Passwort können Sie in Ihrem Kundencenter herausfinden
  - b. **Der Port ist bei allen der Gleiche**
  - c. Nun die Ihnen ausgehändigten Login Daten verwenden
2. Klicken Sie nun auf primary



3. Starten Sie nun den Node-Red Container indem Sie auf batsandro klicken und anschliessend oben auf den grünen start Button:



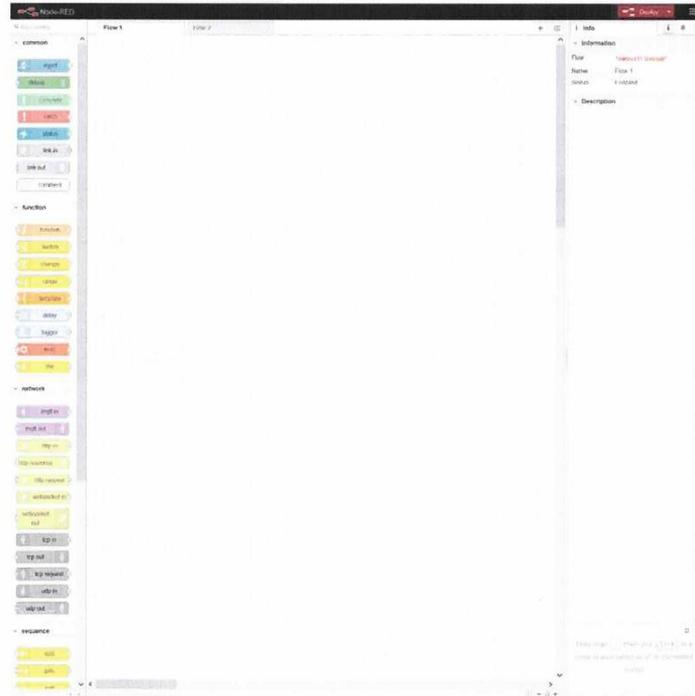
4. Durch Klicken in der Menüleiste auf Container sehen Sie ob der Container erfolgreich gestartet wurde. Durch die Taste F5 können Sie das Fenster wieder aktualisieren. Nach einer Zeit sollten sie folgendes sehen:

Gratulieren Sie haben den Container erfolgreich gestartet.

5. Sie können nun durch die ausgelesene IP-Adresse in diesem BSP 192.168.0.254 und dem Port 1880 auf Node-Red zugreifen: Geben Sie also folgendes im Browser ein wobei die <> weggelassen werden:

a. <Ihre IP-Adresse>:1880

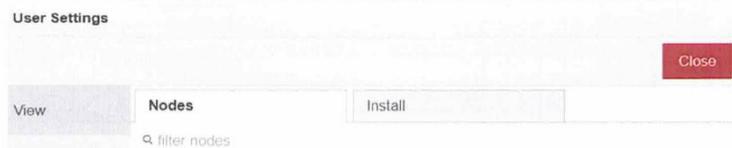
Sie sehen nun die Oberfläche von Node-Red:



## Einfügen der spezifischen Bausteine für das Toradex Modul

Damit wir die Hardware verwenden können müssen wir GPIO (General Purpose Input Output) Bausteine hinzufügen. Um dies zu erreichen befolgen Sie die nächsten Schritte:

6. Klicken Sie oben rechts auf das Burger Symbol (neben dem Roten Deploy Button)
7. Im dem nun aufgeklappten Menu auf **Manage palette**. Sie sehen nun folgendes:

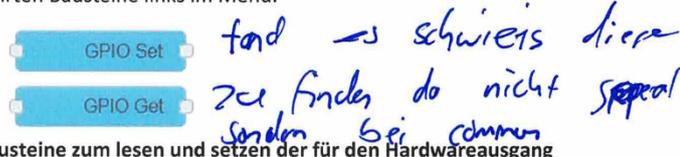


8. Gehen Sie auf Install und suchen nach **toradex**

9. Installieren Sie nun dies durch klicken auf den install Button



10. Sie finden nun die neu eingeführten Bausteine links im Menu:



Glückwunsch Sie haben die Bausteine zum lesen und setzen der für den Hardwareausgang erfolgreich hinzugefügt:

### Erste kleine Anwendung:

Nachfolgend schalten wir eine LED ein. Dafür muss etwas gesetzt werden. Wir verwenden also den GPIO Set Baustein.

1. Ziehen Sie per Drag and Drop den Baustein in den Flow: *GPIO set schwierig zu verstehen erst mit Bild*

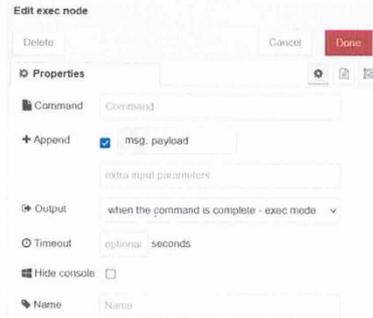


2. Durch Doppelklicken auf diesen Baustein können Sie nun Einstellungen vornehmen:
  - a. Tippen sie bei Pin Number den entsprechende Pin ein: **(in unserem Fall möchten wir die Rote LED ansteuern welche auf dem Pin 40 liegt)**. Gebe Sie also Pin 40 ein. *etwas umständlich*
  - b. Bei on / off müssen wir noch beschreiben ob wie die Led einschalten oder ausschalten möchten. Tippen Sie dafür **on** ein um die LED einzuschalten. *aber ok*

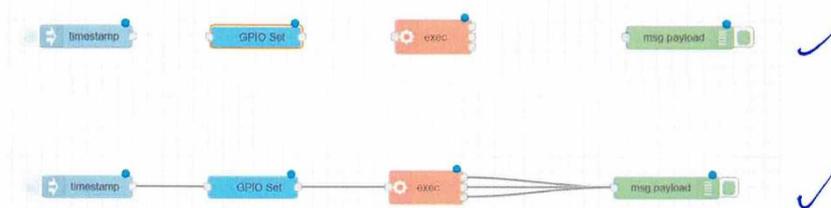
**Wir haben nun den spezifischen Hardwarestein korrekt eingestellt!**
3. Um in Node-Red einen Flow zustarten benötigt man einen Inject Button. Diesen finden Sie ebenfalls bei den Bausteinen.
4. Damit die Einstellungen beim GPIO Set Baustein auch ausgeführt werden, müssen Sie noch einen exec (execute) einfügen.
  - a. In diesem müssen wir eine kleine Anpassung vornehmen damit er den Code korrekt ausführt. Klicken Sie dafür doppelt auf exec.

*↳ Anregung vom User: sollte in einem sein falls möglich*

- i. Setzen sie nun bei Append den Hacken wie unten zu sehen:



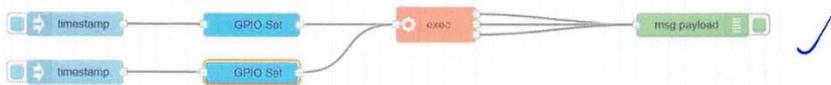
5. Zum Schluss möchten wir noch eine Ausgabe von evtl. Fehlern sehen. Ziehen sie dafür den debug Baustein in den Flow. Sie sollten nun folgendes Bild vorsich haben:



6. Nun müssen diese Bausteine noch miteinander kommunizieren. Dafür werden nun Linien vom jeweiligen Ausgangspunkt auf den nächsten Eingangspunkt angewendet. Das Resultat sehen Sie oben:
7. Nun haben Sie alles erfolgreich konfiguriert. Damit nun aber das Programm/Flow funktioniert, müssen wir ihn zuerst noch deployen. Klicken Sie dafür auf den roten Deploy Button oben rechts.
- Hinweis: Wenn der Button rot leuchtete wurden Änderungen vorgenommen und es muss neu deployt werden um die Änderungen auch Anwenden zu können.**

Sie können nun durch klicken auf das Quadrat links vom timestamp Button den Flow ausführen. Dabei wird der Flow einmal ausgeführt. Sie sehen nun auf dem Board eine rote LED die leuchtet.

8. Erstellen sie nun einen 2 Flow um die Gleiche LED auszuschalten. Das Resultat sehen Sie unten:



9. Um den Überblick im Programm zu behalten lohnt es sich die Bausteine jeweils passend zu beschriften. Sprich wir wissen durch die momentane Ansicht nicht, welchen Pin wir verwenden.
  - a. Dies können sie wieder im Baustein selbst machen in dem Sie bei Namen etwas für Sie passendes eingeben. BSP: PIN 40 on und PIN 40 Off
    - i. Stellen sie beim Anpassen der Namen sicher, dass die zuvor eingestellten Parameter noch stimmen.



### Erstellen eines eigenen kleinen Programms.

Versuchen Sie nun ein folgendes Programm zu erstellen mit den folgenden Infos. Sie müssen dabei das eben gelernte versuchen anzuwenden. Das Ziel des Programm ist durch das lesen eines Pins eine entsprechende LED einzuschalten. Dafür benötigen wir 2 LEDs und den Steuerknopf.

Die Pin Belegung ist wie folgt:

1. LED Rot => Pin 40
2. LED Gelb => Pin 38
3. Steuerbutton links => Pin 31
4. Steuerbutton rechts => Pin 29

Das Programm soll dabei jeweils den entsprechende Pin lesen und solange der Pin spricht der Steuerbutton gedrückt ist soll die entsprechende Led leuchten. Die Zuordnung soll wie folgt sein:

- Steuerbutton links => LED Rot
- Steuerbutton rechts => LED Gelb

#### Tipps

Um eine Button kontinuierlich auszulesen gibt es einen Baustein bei der Untergruppe Funktion mit dem namen **trigger**. Diesen können sie so einstellen dass er nach einer gewissen Zeit immer wieder etwas tut. Als Zeit für eine Benutzer eingabe ist 100ms geeignet.

Um zu entscheiden ob der Schalter gedrückt ist eignet sich die switch Function. Dort kann ein Input auf bestimmte Werte überprüft werden und ein entsprechender Output angesteuert werden.

Alle Bausteine haben zudem Inforamtionen die einem sehr hilfreich sind, konsolidieren Sie diese in Node-red falls sie etwas nicht verstehen.

*Aufgabe war etwas schwieriger zu verstehen*

*Durch die Hilfe beim Einstellen des Trigger Bausteins*

# Benutzertest BAT Node-Red

## Starten des Node-Red

Node-Red wird als ein Container gestartet. Sie können sich einen Container so vorstellen, dass es eine für sich stehende Box in einem Betriebssystem ist. Dadurch ist das System gut geschützt und bei einem zufälligen Fehler kann einfach der Container neu erstellt werden.

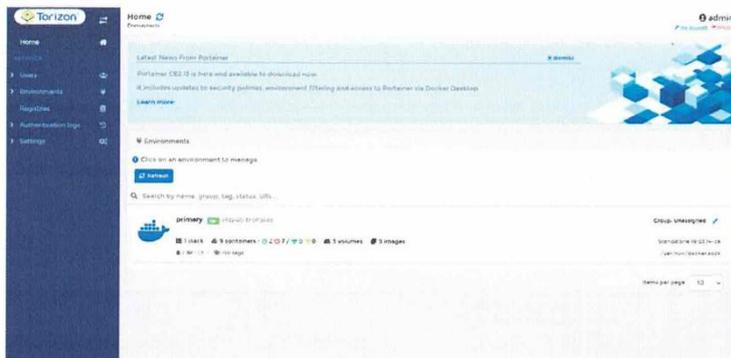
Um Node-Red zu starten bearbeiten Sie bitte die nachfolgenden Schritte:

1. Toradex Modul am Ethernet anschliessen
2. Toradex Modul mit Spannung versorgen

Sie haben nun das Modul erfolgreich gestartet, um den Container zu starten gehen Sie wie folgt vor:

1. Auf Portainer zugreifen: Browser öffnere (BSP. Firefox) und die **IP Adresse** und **Port** eingeben  
=> **192.168.0.254:8840**
  - a. Fall Sie die IP-Adresse nicht wissen, dann können Sie dies in Ihrem Router Interface nachsehend.
    - i. Bei UPC die UPC Connect App verwenden und die angeschlossenen Geräte nach colibri-imx7 suchen
    - ii. Sie können nun die IP-Adresse ablesen
    - iii. Bei Swisscom können Sie dies ebenfalls tun indem Sie im Browser die IP-Adresse 192.168.1.1 eingeben.
    - iv. Der Benutzername lautet standardmässig admin und das Passwort können Sie in Ihrem Kundencenter herausfinden
  - b. **Der Port ist bei allen der Gleiche**
  - c. Nun die Ihnen ausgehändigten Login Daten verwenden
2. Klicken Sie nun auf primary

*Probleme beim IP herausfinden*



3. Starten Sie nun den Node-Red Container indem Sie auf batsandro klicken und anschliessend oben auf den grünen start Button:



4. Durch Klicken in der Menüleiste auf Container sehen Sie ob der Container erfolgreich gestartet wurde. Durch die Taste F5 können Sie das Fenster wieder aktualisieren. Nach einer Zeit sollten sie folgendes sehen:

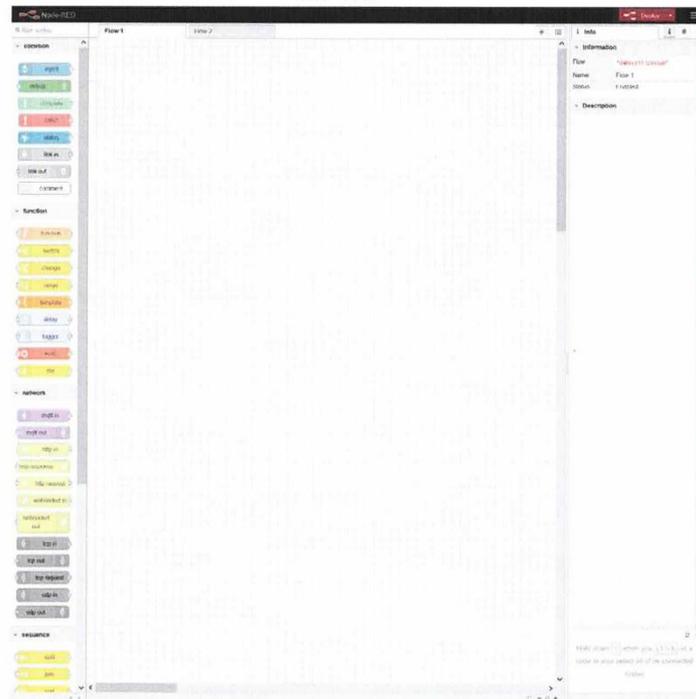


Gratulieren Sie haben den Container erfolgreich gestartet.

5. Sie können nun durch die ausgelesene IP-Adresse in diesem BSP 192.168.0.254 und dem Port 1880 auf Node-Red zugreifen: Geben Sie also folgendes im Browser ein wobei die <> weggelassen werden:

a. <Ihre IP-Adresse>:1880

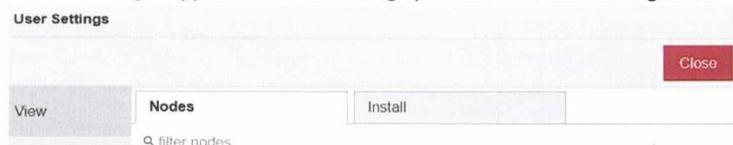
Sie sehen nun die Oberfläche von Node-Red:



## Einfügen der spezifischen Bausteine für das Toradex Modul

Damit wir die Hardware verwenden können müssen wir GPIO (General Purpose Input Output) Bausteine hinzufügen. Um dies zu erreichen befolgen Sie die nächsten Schritte:

6. Klicken Sie oben rechts auf das Burger Symbol (neben dem Roten Deploy Button)
7. Im dem nun aufgeklappten Menu auf **Manage palette**. Sie sehen nun folgendes:

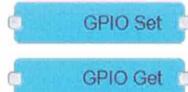


8. Gehen Sie auf Install und suchen nach **toradex**

9. Installieren Sie nun dies durch klicken auf den install Button



10. Sie finden nun die neu eingeführten Bausteine links im Menu:

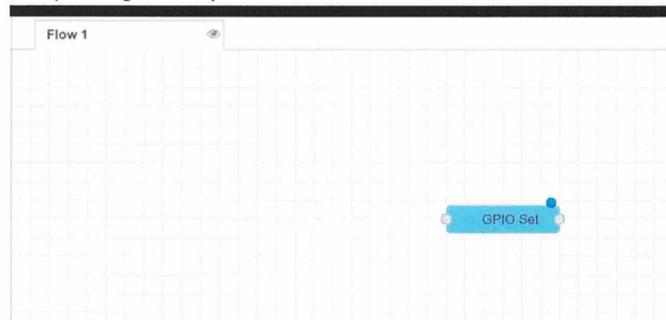


Glückwunsch Sie haben die Bausteine zum lesen und setzen der für den Hardwareausgang erfolgreich hinzugefügt:

*↳ nach dem Einfügen musste node red neu gestartet werden  
⇒ war schwierig da user versuchte nur browserfenster zu schließen  
Erste kleine Anwendung:*

Nachfolgend schalten wir eine LED ein. Dafür muss etwas gesetzt werden. Wir verwenden also den GPIO Set Baustein.

1. Ziehen Sie per Drag and Drop den Baustein in den Flow:



*↳ Tippfehler \*  
Fehler das es  
genau!*

2. Durch Doppelklicken auf diesen Baustein können Sie nun Einstellungen vornehmen:

- Tippen sie bei Pin Number den entsprechende Pin ein: (in unserem Fall möchten wir die Rote LED ansteuern welche auf dem Pin 40 liegt). Gebe Sie also Pin 40 ein.
- Bei on / off müssen wir noch beschreiben ob wie die Led einschalten oder ausschalten möchten. Tippen Sie dafür **on** ein um die LED einzuschalten.  
**Wir haben nun den spezifischen Hardwarestein korrekt eingestellt!**

3. Um in Node-Red einen Flow zustarten benötigt man einen Inject Button. Diesen finden Sie ebenfalls bei den Bausteinen.

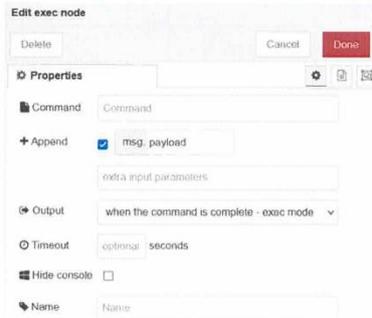
4. Damit die Einstellungen beim GPIO Set Baustein auch ausgeführt werden, müssen Sie noch einen exec (execute) einfügen.

- In diesem müssen wir eine kleine Anpassung vornehmen damit er den Code korrekt ausführt. Klicken Sie dafür doppelt auf exec.

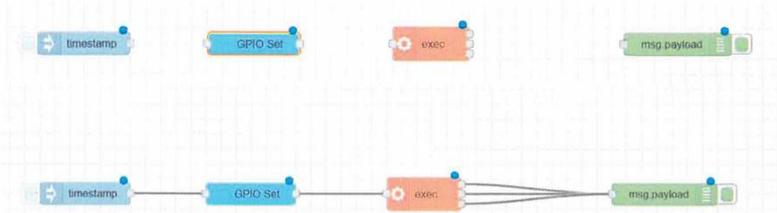
*\* konnte fehler nicht selbst beheben.*

Wurde beim 1 gemacht beim  
 2 vergessen  $\Rightarrow$  anschließende Verbindung konnte nicht  
 $\downarrow$  interpretiert werden

i. Setzen sie nun bei Append den Hacken wie unten zu sehen:



5. Zum Schluss möchten wir noch eine Ausgabe von evtl. Fehlern sehen. Ziehen sie dafür den debug Baustein in den Flow. Sie sollten nun folgendes Bild vorsich haben:



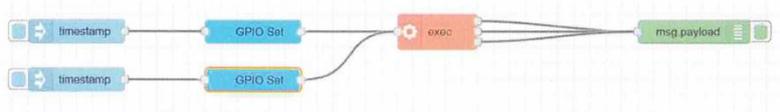
6. Nun müssen diese Bausteine noch miteinander kommunizieren. Dafür werden nun Linien vom jeweiligen Ausgangspunkt auf den nächsten Eingangspunkt angewendet. Das Resultat sehen Sie oben: ✓

7. Nun haben Sie alles erfolgreich konfiguriert. Damit nun aber das Programm/Flow funktioniert, müssen wir ihn zuerst noch deployen. Klicken Sie dafür auf den roten Deploy Button oben rechts. ✓

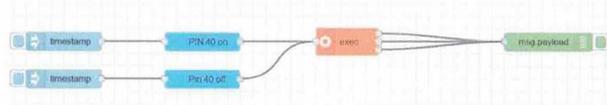
**Hinweis: Wenn der Button rot leuchtete wurden Änderungen vorgenommen und es muss neu deployd werden um die Änderungen auch Anwenden zu können.** ✓

Sie können nun durch klicken auf das Quadrat links vom timestamp Button den Flow ausführen. Dabei wird der Flow einmal ausgeführt. Sie sehen nun auf dem Board eine rote LED die leuchtet.

8. Erstellen sie nun einen 2 Flow um die Gleiche LED auszuschalten. Das Resultat sehen Sie unten: ✓



9. Um den Überblick im Programm zu behalten lohnt es sich die Bausteine jeweils passend zu beschriften. Sprich wir wissen durch die momentane Ansicht nicht, welchen Pin wir verwenden.
  - a. Dies können sie wieder im Baustein selbst machen in dem Sie bei Namen etwas für Sie passendes eingeben. BSP: PIN 40 on und PIN 40 Off
    - i. Stellen sie beim Anpassen der Namen sicher, dass die zuvor eingestellten Parameter noch stimmen.



### Erstellen eines eigenen kleinen Programms.

Versuchen Sie nun ein folgendes Programm zu erstellen mit den folgenden Infos. Sie müssen dabei das eben gelernte versuchen anzuwenden. Das Ziel des Programm ist durch das lesen eines Pins eine entsprechende LED einzuschalten. Dafür benötigen wir 2 LEDs und den Steuerknopf.

Die Pin Belegung ist wie folgt:

1. LED Rot => Pin 40
2. LED Gelb => Pin 38
3. Steuerbutton links => Pin 31
4. Steuerbutton rechts => Pin 29

*→ wurde versehentlich Pin 35 geschrieben \**

Das Programm soll dabei jeweils den entsprechende Pin lesen und solange der Pin spricht der Steuerbutton gedrückt ist soll die entsprechende Led leuchten. Die Zuordnung soll wie folgt sein:

- Steuerbutton links => LED Rot
- Steuerbutton rechts => LED Gelb

### Tipps

Um eine Button kontinuierlich auszulesen gibt es einen Baustein bei der Untergruppe Funktion mit dem namen **trigger**. Diesen können sie so einstellen dass er nach einer gewissen Zeit immer wieder etwas tut. Als Zeit für eine Benutzer eingabe ist 100ms geeignet.

Um zu entscheiden ob der Schalter gedrückt ist eignet sich die switch Function. Dort kann ein Input auf bestimmte Werte überprüft werden und ein entsprechender Output angesteuert werden.

Alle Bausteine haben zudem Inforamtionen die einem sehr hilfreich sind, konsolidieren Sie diese in Node-red falls sie etwas nicht verstehen.

*\* => gibt es nicht in der Ansteuerung ≙ GMP => Fehler wurde nicht selber gefunden*

*o als Alternative benötigt für trigger, switch*