Motivation

Der Wecker klingelt und der Kaffee ist bereits gekocht, da die Kaffeemaschine wusste, wann der Handywecker klingeln wird und die Rollläden im ganzen Haus fahren auch bereits automatisch hoch. Früher noch als „Zukunftsmusik“ beschrieben, sind Funktionen wie diese heute bereits in schätzungsweise 10 Millionen [1] deutschen Haushalten verbaut. „Home Automation“, „Smart Home“ und „Internet of Things“ (engl. kurz IoT) sind damit keine Fremdwörter mehr.

Dank dem anhaltenden Verbrauchertrend zu „Smart Home“ im eigenen Haus gelangen mehr und mehr Menschen in Kontakt mit Technik, welche Internetfähig ist, Sensoren und Aktoren besitzt und vollautomatisiert Entscheidungen treffen kann. Wie bei den bereits genannten Beispielen ist das Internet der Dinge oft leicht zu verstehen, jedoch häufig komplex in der Implementierung. Ein von IBM veröffentlichtes Tool, welches beim Einstieg in das komplexe Thema helfen kann, ist Node-Red. Mit grafischem Editor und Bausteinsystem hilft es Sensoren und Aktoren leicht miteinander zu verbinden.

Die Anwendungen für das Tool selbst sind nahezu unbegrenzt. Mit Node-Red kann man schnell, übersichtlich und grafisch dargestellt mehrere Webservices miteinander verbinden, Daten abfragen, auswerten und speichern. Dank eingebautem Dashboard können die gewonnenen Erkenntnisse dazu noch im Webbrowser visualisiert werden.

Auf diese Weise erfüllt das Tool viele Aufgaben der modernen Automatisierungstechnik, die oftmals nur mit viel Programmieraufwand zu realisieren sind. Die große Community hinter Node-Red ermöglicht zusätzlich noch durch Tausende sogenannte "Module" den Funktionsumfang stark zu erweitern. Weiterhin benötigt das Tool sehr wenig Speicherplatz, ist auf nahezu allen Umgebungen installierbar und findet daher auch leicht Platz auf Industriesystemen.

Werkzeuge und Grundlagen

Hauptbestandteil der Arbeit ist das Open-Source-Tool Node-Red [2]. Es basiert vollständig auf einem anfängerfreundlichen Bausteinsystem. In den sogenannten Flows kann man kinderleicht ohne große Programmierkenntnisse komplexe Anwendungen aufbauen. Ein Flow ist ein Programm, das Ereignisse leitet, während diese Ereignisse durch ein Netzwerk von Knoten fließen. Das Tool kann lokal auf jedem PC mit nahezu jedem Betriebssystem, auf dem Android Mobilgerät, auf Raspberry Pis und BeagleBone Boards oder auch direkt in Cloudservices wie: IBM-Cloud, Amazon AWS oder Microsoft Axure installiert werden. Zur Installation sind je nach Plattform zumeist einfache Computerkenntnisse nötig.

In dieser Arbeit wurde Node-Red auf einem Raspberry Pi 4 installiert. Nach erfolgreicher Installation kann Node-Red bequem über den Browser erreicht werden. In Node-Red selbst wird unterschieden zwischen dem Node-Red UI und dem Node-Red Dashboard. Das Node-Red UI, dient als grafische Benutzeroberfläche zur Programmierung der Flows. Um die durch die Flows ermittelten Werte besser zu veranschaulichen, existiert zusätzlich noch das Node-Red Dashboard. Beide Instanzen können durch unterschiedliche Ports auf dem Node-Red System erreicht werden.

Die Software bietet ebenfalls die Möglichkeit, das Funktionsspektrum der bereits installierten „Standard“-Nodes durch Module zu ergänzen. Diese Module enthalten dann zumeist zusätzliche Nodes und Funktionalität oder Schnittstellenerweiterungen. Durch die Größe der Node-Red Community und der

nahezu unbegrenzten Anwendungsmöglichkeiten im Bereich des IoT, kann das Tool durchaus auch komplexen Industrie-Anwendungen gerecht werden. Beispielsweise kann Node-Red, wie bereits von Zare et al. [3] gezeigt, als SCADA System zum Einsatz kommen kann. Ein ähnliches System wie bereits von Zare et al. vorgeschlagen wurde im Rahmen dieses Projektes als Beispiel umgesetzt und wird im nachfolgenden näher erläutert.

Um das Zusammenschalten mehrere Webservices zu simulieren, wurde die Open-Source Javascript Laufzeitumgebung Node.js genutzt. Die Webservices können so alle mithilfe des "Node Package Managers", kurz npm [4], hintereinander gestartet werden. In der Simulation laufen diese dann auf einem Windows-PC. Um die Webservices lokal laufen zu lassen, muss das git-repository [5] gedownloadet werden und "Node.js" sowie der "Node Package Manager" auf dem System installiert sein. Durch den Aufruf von „npm install“ im Stamm-Verzeichnis des Projektes werden die nötigen „node-modules“ wie „mqtt“ installiert. Durch den Aufruf von "npm run start" wird dann die Webservices.js Datei ausgeführt und sofern der MQTT Server richtig konfiguriert wurde beginnt Node-Redmit der Auswertung der Daten.

Anforderungen

Es soll das Tool „Node-Red“ hinsichtlich seiner Anwendbarkeit in der Industrie, Funktionalität und Erweiterbarkeit untersucht werden. Dabei soll mithilfe dieses Tools eine einfache Möglichkeit implementiert werden, um mehrere Web-Services gewinnbringend miteinander zu verschalten. Die Anforderungen der Industrie 4.0 sind dabei basierend auf einem Artikel von Nathalie Kletti.[6]

Anforderungen der Industrie 4.0

1. [IA01]Das Tool soll in industriellen Umgebungen lauffähig sein
2. [IA02]Das Tool soll 24/7, rund-um-die-Uhr lauffähig sein
3. [IA03]Das Tool soll einen hohen Grad an Interoperabilität besitzen
4. [IA04]Das Tool soll Analytics Funktionen effizient anbinden können
5. [IA05]Das Tool soll ergonomisch und anwendungsorientiert sein
6. [IA06]Das Tool soll mit älteren und neuen Maschinen kompatibel sein

Anforderungen an die Erweiterbarkeit

1. [EA01]Das Tool bietet eine Möglichkeit zur Funktionserweiterung
2. [EA02]Das Tool bietet eine Möglichkeit eigene Erweiterungen einzubinden
3. [EA03]Programmcode soll einfach zu importieren und exportieren sein

Anforderungen an das Beispiel-Szenario Node-Red als SCADA System

1. [SA01]Das Beispielsystem soll Änderungen im Ausgangssystem gut sichtbar für Benutzer visualisieren
2. [SA02]Das Beispielsystem soll Alarme und deren Attribute aller Services anzeigen
3. [SA03]Das Beispielsystem soll Alarme geordnet nach Zeit anzeigen
4. [SA04]Das Beispielsystem soll Alarme zu den Webservices zuordnen von denen Sie stammen
5. [SA05]Das Beispielsystem soll ein UI besitzen welches leicht verständlich und übersichtlich ist
6. [SA06]Das Beispielsystem soll empfangene Daten als Historie anzeigen können

Entwurfsvarianten und Entwurf

Node-Red und seine Verwendung in der Industrie 4.0

Die Anforderungen IA01-IA05 sind mit dem Node-Red System an sich bereits vollumfänglich erfüllt. Node-Red kann auf verschiedensten Plattformen installiert werden [7] und läuft selbstständig im Hintergrund. Speziell auf Linux basierten System sind damit IA01 und IA02 als Anforderungen erfüllt. Weiterhin stellt die Node-Red Plattform verschiedene Analytics Funktionen wie zum Beispiel das Debug Fenster nativ zur Verfügung. Die Anforderungen IA03 und IA05 werden durch die einfache Handhabung, Installation und Benutzerfreundlichkeit von Node-Red erfüllt. Der Zugriff auf die Entwicklungsumgebung ist einfach durch einen beliebigen Browser möglich und erfordert keine außergewöhnlichen IT-Kenntnisse oder Zusatzinstallationen auf dem Bediener-System.

Anforderung IA06 fordert EA01 bzw. EA02. Die Kompatibilität von Node-Red kann enorm erhöht werden durch Module, die entweder durch die Online-Community zur Verfügung gestellt werden oder auch komplett selbstständig implemntierbar sind. Viele Tausende Module sind auch bereits in einem Online-Katalog innerhalb der Node-Red Plattform jederzeit einfach integrierbar. Schnittstellen die Node-Red bereits zu bieten hat sind unter anderem, HTTP/REST, MQTT, Websocket, Twitter, OPC-UA und E-Mail. Somit erweist sich Node-Red auch für diese Anforderung als gute Lösung.

Node-Red basiert vollständig auf einer JSON-Objektumgebung. Das bedeutet, dass alle Elemente inklusive Notizen und Gruppierungen, Konfigurationen und Parameter, welche zu einem Node-Red Flow gehören, als JSOn-Objekt betrachtet werden. Damit kann der gesamte Code eines jeden Node-Red Flows jederzeit einfach per export/import Funktion als .json heruntergeladen und weiter verwendet werden. Ein Anbieter für IoT-Software könnte so beispielsweise sehr leicht eine komplexe neue Funktion einfach an seine Kunden weitergeben. Somit ist auch Anforderung EA03 erfüllt.

Um nun die Eignung von Anforderungen IA01-IA06 als erfüllt zu beweisen wurde ein Beispielszenario entworfen, welches Node-Red als SCADA System für einen Industrieofen simuliert. SA01 fordert IA03, wegen SA02-SA04 ergeben sich die Varianten V1 und V2. SA05 fordert IA05, wegen SA06 ergeben sich die Varianten V3 und V4. Dabei ergeben sich folgende Vor- und Nachteile der Varianten:

- **Variante V1: Kommunikation per HTTP**
  - Vorteile:
    - HTTP Funktionalität bereits eingebaut in Node-Red
  - Nachteile:
    - HTTP GET Request ist viel zu groß um nur ein Datenpaket zu holen
    - Überlastung bei 10 GET Requests gleichzeitig
    - kein Quality of Service
- **Variante V2: Kommunikation per MQTT**
  - Vorteile:
    - Publish/Subscribe Modell ist sehr leichtgewichtig und datenarm
    - Übertragung erfolgt als byte array
    - 3 Level Quality of Service
    - Daten können im Falle von unerwarteten Beendigungen bei der nächsten Transaktion nachgereicht werden
    - Insgesamt übertragene Datenmenge ist kleiner als bei einer http-Nachricht
  - Nachteile:

        - MQTT muss erst nachinstalliert und speziell eingerichtet werden
- **Variante V3: Visualisierung der historischen Daten per Tabelle**
    - Vorteile:
        - Einzelne Zahlenwerte können über einen langen Zeitraum eingesehen werden
        - Einzelne Datenpunkte und deren zugehörige Attribute können direkt erkannt werden
    - Nachteile:
        - Ein Gefühl für die Entwicklung der Daten gegenüber der Zeit entsteht nicht
        - Die Ober/Unter-grenzen der jeweiligen Sensoren liegen nicht in direkter Sicht des Betrachters
- **Variante V4: Visualisierung der historischen Daten per Liniengrafik**
    - Vorteile:
        - Die Entwicklung der Daten gegenüber der Zeit kann leicht erkannt werden.
        - Eine sprunghafte Veränderung wird sofort in der Grafik sichtbar
        - Durch sinnvolle Anordnung der Grafiken kann die Übersichtlichkeit gewährleistet werden
    - Nachteile:
        - Die Grafiken können viel Platz auf dem Dashboard des Bedieners einnehmen.

Für die Realisierung des Beispielsystems haben sich die Varianten V2 und V4 erwiesen. Aufgrund der hervorragenden Eignung von MQTT als Kommunikationsprotokoll in der IoT und der einfachen Integration in Node-Red wurde dieses Protokoll dem Standard HTTP Protokoll bevorzugt. Die Variante V4 wurde ebenfalls aufgrund der vielen Vorteile, der Benutzerfreundlichkeit und der frühzeitigen Erkennbarkeit von Anomalien im Verlauf der Daten als gute Wahl eingestuft.

Node-Red als SCADA System

Der Entwurf eines Beispielsystems begann mit dem nachfolgenden Schaubild. Es zeigt alle implementierten Elemente des Prozesses und wo sich in einem realen System entsprechende Sensoren befinden würden. Insgesamt besteht das System aus fünf Durchfluss- und zwei Temperatursensoren sowie einem Rotationsgeschwindigkeits- und einem Füllstandssensor. Der simulierte Prozess beschreibt eine Flüssigkeit, welche in einen Ofen bis zu einer bestimmten Füllhöhe eingelassen, verrührt und erhitzt wird. Jeder Sensor symbolisiert ein eigenständiges netzwerkfähiges IoT-Gerät und sendet per MQTT an Node-Red im Sekundentakt seine aktuellen Prozessdaten. Weiterhin kennt jedes Gerät seine individuellen Grenzwerte und generiert somit zusätzlich Alarme, welche ebenfalls an Node-Red weitergegeben werden.

{{protele:documentation:node-red:rothhaupt_marcus:scada_system_sketch.png?500}}

Kommunikationsdiagramm

Das hier gezeigte Kommunikationsdiagramm zeigt, wie die einzelnen Bestandteile des Beispielsystems miteinander verknüpft sind. Node-Red bekommt per MQTT die Prozessdaten und Alarme von den Webservices, verarbeitet diese und zeigt sie auf dem internen Dashboard, welches über einen separaten Port per http aufgerufen werden kann. Der Anwender kann sich dann über seinen Browser das Dashboard anzeigen und alle Prozessdaten sowie grafischen Verläufe einsehen.

{{protele:documentation:node-red:rothhaupt_marcus:project_diagramm.png?500}}

UML Klassendiagramm von Webservices.js

Die Generierung der Prozessdaten und Alarme erfolgt in einer einzelnen Node.js Datei, welche mehrere Klassen und Funktionen beinhaltet. Diese sind im nachfolgenden UML Klassendiagramm dargestellt.

{{protele:documentation:node-red:rothhaupt_marcus:node-red_class-node_red_uml_class_diagramm.png?400}}

UML Sequenzdiagramm der MQTT Datenübertragung

Die Übertragung der simulierten Sensordaten und Alarme erfolgt ähnlich mittels des Pub/Sub Prinzips von MQTT. Die Sensoren senden ihre Daten alle an die ihnen zugeordnete "MQTT-Topic" und werden so von dem Node-Red Flow weiterverarbeitet. Schließlich werden die einzelnen Alarme an eine gemeinsame "MQTT-Alarmtopic" gesendet und gebündelt. Die Alarme werden alle in der Datei alarms.csv abgelegt.

{{protele:documentation:node-red:rothhaupt_marcus:node-red_mqtt_uml-mqtt_communication_of_node_red_webservice.png?600}}

UML Sequenzdiagramm der Node-Red Alarmauswertung

Die Datei alarms.csv wird vom Node-Red Flow ständig auf Änderungen überwacht. Falls eine Änderung erkannt wird, beginnt die Auswertung, Sortierung und Einordnung der Alarme in die Alarmtabelle des Dashboards.

{{protele:documentation:node-red:rothhaupt_marcus:node-red_alarms-node_red_alarm_handling.png?400}}

Entwurf des Node-Red Dashboard

Das Design des Dashboards wurde so gewählt, dass es einfach, verständlich und übersichtlich ist. Veränderungen der Daten sind leicht sichtbar und durch entsprechende Farbwahl leicht zu interpretieren. Wie im Buch von Zuehlke [8] erwähnt, hat der Mensch das Bedürfnis nach Klarheit und Ordnung in seiner Wahrnehmung. Jedes Element des SCADA-Dashboards musste also gut ausgerichtet und innerhalb der Bildschirmfläche platziert werden, um diesen Designprinzipien zu entsprechen.

{{protele:documentation:node-red:rothhaupt_marcus:image_dashboard.jpg?800}}

Entwurf des Node-Red Flows

Hier abgebildet ist der Node-Red Flow mit allen Elementen und Verbindungen des SCADA-Systems. Beim Entwurf des Flows wurden die Best Practices für Node-Red beachtet.

{{protele:documentation:node-red:rothhaupt_marcus:node-red_flow.jpg?800}}

Implementierung

Generierung und Abruf der Simulationsdaten eines Webservices

Node.js

Die Simulationsdaten werden in einer dauerhaften Schleife im Abstand von einer Sekunde für jeden Sensor generiert. Die Generierung basiert dabei weder auf komplexen mathematischen Operationen noch auf Beispieldaten von echten Sensoren. Zur Generierung werden ausschließlich einfache Addition und Subtraktion sowie kleinere zufällig generierte Werte verwendet. Die Simulationsdaten werden für jeden Sensor per MQTT an die jeweilige MQTT-Topic veröffentlicht("published").

Node-Red

Die Simulationsdaten werden von Node-Red direkt zu den Dashboard-Elementen weitergeleitet. Der entsprechende Teil des Node-Red Flows ist beispielsweise für den Motorrotationsgeschwindigkeitssensor hier dargestellt. Die Funktion "reset chart" wird nur aufgerufen, wenn die Node.js Simulationsumgebung neu gestartet wird.

{{protele:documentation:node-red:rothhaupt_marcus:mqtt_motor_rotation_speed_flowelement.jpg?400}}

Generierung und Abruf von Alarmen

Node.js

Die Alarme werden basierend auf vorher festgelegten Grenzwerten generiert. Die Grenzwerte variieren je nach Sensortyp. Sobald ein Grenzwert erreicht ist, geht wachsen die simulierten Rohdaten wieder in die entgegengesetzte Richtung. In realen Systemen wäre zu diesem Zeitpunkt ein Handeln des Personals erforderlich. Nach der Generierung werden diese im comma-seperated-value(csv)-Format per MQTT an den Node-Red MQTT Server übertragen.

Node-Red

Wie bereits im Sequenzdiagramm in der Sektion "Entwurf" gezeigt, werden die empfangenen Alarme zunächst in eine Datei geschrieben, bevor sie verarbeitet werden. Danach wird direkt eine Alarmnachricht rechts unten im Dashboard angezeigt und zeitgleich die zweite dargestellte Zeile des Node-Red-Flows ausgelöst. Sobald Änderungen an der überwachten Datenbankdatei alarms.csv erkannt werden, beginnt die Ordnung aller Alarme aus der alarms.csv Datei entsprechend ihrer Zeitstempel. Kurz darauf werden sie in die Tabelle des Dashboards eingefügt.

{{protele:documentation:node-red:rothhaupt_marcus:node-red_alarm_flowelements.jpg?800}}

Node-Red Best Practices

Node-Red bietet eine schnelle Möglichkeit zur Erstellung einfacher IoT relevanter Applikationen. Jedoch können Flows auch ebenso schnell mit wachsendem Funktionsumfang in der Komplexität stark zunehmen. Umso wichtiger ist es, sich an gute Vorgaben "Best Practices" zu halten und diese auch konsequent umzusetzen. Mit der Hilfe von [9] wurden folgende Best Practices zusammengefasst:

1. Flows Organisieren:
    i. Flows in kleinere Gruppen teilen und Flow Reiter nutzen, um Funktionen zu separieren
    ii. Wiederverwendbare Komponenten erstellen, indem Link Nodes und Subflows verwendet werden
    iii. Zustand der Flows im Auge behalten und wenn möglich (per MQTT) von außen steuern
    iv. Fehlerbehandlung korrekt durchführen, indem Catch Nodes und die Debug Funktion von Node-Red genutzt werden
2. Nachrichten Organisieren:
    i. Vorhandene Nachrichteneigenschaften wie msg.payload sinnvoll nutzen
    ii. Eigenschaften der Nachrichten sinnvoll benennen
    iii. Neue Nachrichteneigenschaften entwickeln, um die Übersichtlichkeit des Codes zu garantieren
3. Flows dokumentieren:
    i. Guter Code sollte gut dokumentiert sein
    ii. Node-Red als visuelle Programmierschnittstelle bietet verscheiden Möglichkeiten zur Dokumentation von Code

iii. Anordnung von Flow Elementen muss gut strukturiert sein
iv. Die Benennung einzelner Flow Element hilft beim Verständnis der Funktionalität
v. Elemente mit mehreren Ausgängen können Ausgangsbeschriftungen besitzen
vi. Kommentare sollten zwischen den Nodes genutzt werden
vii. Die Gruppierung der Elemente kann die Übersichtlichkeit bei vielen Nodes vereinfachen

Fallstudie / Test

An sich ist das Node-Red SCADA Beispiel bereits ein realistisches Beispielszenario, welches auch von anderen Autoren bereits erprobt worden ist [3]. Es zeigt eindeutig die Fähigkeit von Node-Red, als industrietaugliches Programm eingesetzt zu werden. Dennoch möchte ich trotzdem im nachfolgenden ein Beispielszenario für die Funktion des Node-Red SCADA Systems näher erläutern.

Test 1: Anzeige eines Alarms im Dashboard

Nach Festlegung der Obergrenze von 240 u/min für den Sensor der Rotationsgeschwindigkeit des Motors wurde die Simulation für alle Services gestartet. Nachdem der Wert der Rotationsgeschwindigkeit zunächst stieg erreicht dieser den kritischen Wert von 240 u/min nach genau 74 Sekunden. Dadurch wurde ein Alarm ausgelöst und per MQTT an Node-Red gesendet. Dabei wurde der Zeitstempel, der Typ des Alarms sowie die Sensorzugehörigkeit und der aktuelle Wert des Sensors mitgeliefert.

Die Übertragung der Daten per MQTT an Node-Red dauerte nur einen Bruchteil einer Sekunde. Node-Red verarbeitet den Alarm , ordnet ihn zusammen mit den anderen Alarmen (SA03)(SA04) und stellt die Daten in einer Tabelle, welche übersichtlich im Dashboard angezeigt wird (SA05)(SA02), dem Nutzer zur Verfügung. Siehe Abbildung. Zusätzlich wird eine Meldung angezeigt, die dem Nutzer den Eingang eines neuen Alarms symbolisiert. Der augenscheinlich vorhandene zweite Alarm zum Zeitpunkt 75 wird angezeigt, da der Grenzwert nach einer Sekunde immer noch überschritten ist.

{{protele:documentation:node-red:rothhaupt_marcus:image_alarm_test?300}}

Test 2: Anzeige von Daten im Dashboard

Ähnlich zur Übertragung der Alarme werden Daten von Sensoren sekündlich übertragen und von Node-Red ausgewertet. Hierbei kommt nun noch die Visualisierung der Daten und der Historie hinzu (SA06) (SA01). Wie in der Abbildung zu sehen, sind die Sensordaten des Rotationssensors bis zum Zeitpunkt kurz vor 19:38:50 kontinuierlich, entsprechend der festgelegten Simulation gestiegen. Danach haben die Daten begonnen wieder sinken. Durch die Darstellung als Liniendiagramm kann der Nutzer den Verlauf gut verfolgen.

{{protele:documentation:node-red:rothhaupt_marcus:image_rotationspeed_data?300}}

Diskussion der Ergebnisse / Validierung

Alle Anforderungen an die Fertigungs-IT von morgen werden durch Node-Red abgedeckt und eventuelle Funktionalitätslücken werden durch eine große Online-Community, leicht austauschbaren Quellcode und individuell erstellbare Module wett gemacht. Im speziellen wurden im Entwurf des SCADA Beispielsystems folgende Entscheidungen getroffen.

Basierend auf den Anforderungen IA01 und IA02 wurde das SCADA System auf einem linux-basierten Raspberry Pi implementiert, um bestmöglich eine Industrieumgebung zu simulieren. Durch den Einsatz der Node-Red Weboberfläche und dem Einsatz der Debug und Analytics Funktionen von Node-Red wurden die Anforderungen IA03-IA05 belegt. Anforderung IA06 fordert einen hohen grad an Kompatibilität und Erweiterbarkeit. Auch das wurde durch das Beispielsystem gut gezeigt, denn es wurde mithilfe der Modulerweiterung das MQTT protokoll zur Funktionalität von Node-Red hinzugefügt. Dieses Protokoll ist stark verbreitet in der Automatisierungstechnik und sichert so die Kommunikation sowohl mit neuen als auch mit alten Maschinen.

Die Fallstudie hat weiterhin gezeigt, dass sogar die zusätzlich gestellten Anforderungen an das SCADA System erfüllt sind. Node-Red weißt somit alle wichtigen Eigenschaften auf, um als Werkzeug der industriellen Automatisierungstechnik infrage zu kommen. Weitere Arbeiten zum Thema Node-Red beweisen,dass das Tool auch als Modbus Kommunikationsschnittstelle [10], als Modbus-OPC UA Wrapper [11] und als Prototyp-Werkzeug für IoT Applikationen [12] genutzt werden kann. Außerdem wurde Node-Red bereits von Siemens in eine IOT2000 SIMATIC Steuerung integriert. Somit ist gezeigt, dass das Tool bereits seinen Einsteig in die industrielle Automatisierungstechnik vollzogen hat [13].

Ausblick

Das Tool Node-Red erfüllt alle wichtigen Anforderungen an die Industrie von morgen. Es ist leicht zu benutzt, höchst kompatibel und interoperabel. Somit kann Node-Red in einer Vielzahl von Industrie-Anwendungen genutzt werden. Dabei sollte stets darauf geachtet werden, dass die Node-Red Best Practices eingehalten werden. Wie eine Umfrage aus dem Jahr 2019 [14] gezeigt hat, ist Node-Red weiterhin am Wachsen und besonders im kommerziellen Bereich am Aufsteigen. Auch in Zukunft wird es daher eine große Node-Red Community und somit noch einen langen Support für das Open-Source-Tool geben.

Trotz alledem gibt es noch offene Fragestellungen, welche in einem späteren Projekt noch genauer beleuchtet werden könnten. Eine Frage bezieht sich auf die genaue Entwicklung von eigenen Modulen für Node-Red. Es wäre interessant zu wissen, wie genau die Entwicklung von eigenen Modulen zur Funktionserweiterung von Node-Red funktioniert. In einigen Online-Forum ist zu lesen, dass dies durch die Manipulation des Source-Codes von Node-Red möglich ist.

Weiterhin ist zu sagen, dass an der Implementation des SCADA Beispiels ebenfalls noch einige Dinge verbessert werden könnten. Die verschiedenen Klassen könnten zum Beispiel noch in separate Dateien ausgelagert werden, um so eine bessere Struktur zu schaffen. Außerdem wäre es möglich die Konfiguration der einzelnen Webservices ebenfalls in eine Datei auszulagern und eine Orchestrierung der einzelnen Webservices per UI implementieren.

Schlussendlich bleibt noch zu sagen, dass die Arbeit mit Node-Red große Freude bereitet hat, da die Entwicklung mithilfe von Flows schon an einigen Stellen übersichtlicher und vor allem aber einsteigerfreundlicher ist, als die Nutzung einer herkömmlichen Programmiersprache und Programmierprinzipien.

Durch den großen Funktionsumfang und besonders gute Eignung als Heimautomatisierungstool wurde weiterhin beschlossen, dass das Tool auch ein hohes Maß an persönlichen Nutzen mit sich bringt und daher fortan beim Ersteller dieser Arbeit zur Anwendung kommt.