XYZ

Studienarbeit

des Studienganges Elektrotechnik Automation

an der Dualen Hochschule Baden-Württemberg Ravensburg

von

Marius Gerstner

16.07.2023

Matrikelnummer, Kurs 7643074, TEA20

Eidesstaatliche Erklärung

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017.

Ich versichere hiermit, dass ich meine Projektarbeit mit dem Thema:

XYZ

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Rastatt, den 25.09.2023

_____

Marius Gerstner

Sperrvermerk

gemäß Ziffer 1.1.13 der Anlage 1 zu §§ 3, 4 und 5 der Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg vom 29.09.2017:

„Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung vom Dualen Partner vorliegt.“

Rastatt, den 16.07.2023

_____

Marius Gerstner

Disclaimer

Das in diesem Bericht gewählte generische Maskulinum bezieht sich zugleich auf den männlichen, weiblichen und anderen Geschlechteridentitäten. Zur besseren Lesbarkeit wird auf die Verwendung männlicher und weiblicher Sprachformen verzichtet. Alle Geschlechteridentitäten werden ausdrücklich mitgemeint, soweit die Aussagen dies erfordern.

Abkürzungsverzeichnis

MQTT *Message Queuing Telemetry Transport*

Abbildungsverzeichnis

Abbildung 1: Portainer Weboberfläche des implementierten IoT-Stacks 5
Abbildung 2: Dataflow zur Extraktion der Motordaten aus der SPS 7
Abbildung 3: Dataflow zur Extraktion der QR-Code und Varianteninformation 7
Abbildung 4: Dataflow des Auf- und Ablagen-Zählers 8
Abbildung 5: Dataflow zur Bestimmung der Ladungsträgeranzahl zwischen den Stationen 9
Abbildung 6: Tabelle der aktuellen Motordaten 10
Abbildung 7: Tabelle mit den aktuellen, relationalen Anlagenzuständen 11
Abbildung 8: Visualisierung der Motorzustandsdaten 12
Abbildung 9: Peakboard Visualisierungsoberfläche 13

Tabellenverzeichnis

Tabelle 1: IP-Adressierung der Soft- & Hardwaregeräte 14

Inhaltsverzeichnis

EIDESSTAATLICHE ERKLÄRUNG II
SPERRVERMERK III
DISCLAIMER IV
ABKÜRZUNGSVERZEICHNIS V
ABBILDUNGSVERZEICHNIS VI
TABELLENVERZEICHNIS VI
INHALTSVERZEICHNIS VII
1. INTEL NUC IOT BACKBONE 1
1.1. Gründe für den Plattformwechsel 1
1.2. Virtualisierungsumgebung Proxmox 1
1.3. Implementierung des Software-Stacks 2
2. ERLÄUTERUNG DER DIENST-FUNKTIONALITÄTEN 6
2.1. MQTT-Broker 6
2.2. NodeRED 6
2.3. Datenbank TimescaleDB 10
2.4. Datenvisualisierung mittels Grafana 11
3. DAS PEAKBOARD ALS SCHNITTSTELLE ZUM MENSCHEN 13
4. INFORMATIONEN ZUM ARBEITEN MIT DEM IOT-STACK 14
LITERATURVERZEICHNIS XV

1. Intel NUC IoT Backbone

1.1. Gründe für den Plattformwechsel

Zu Beginn der zweiten Projektphase dieser Studienarbeit wird festgelegt, einen Plattformwechsel durchzuführen. Hierbei entfernt man sich von dem vorherig verwendeten InHub GM100 IoT Gateway und entschließt sich für die Verwendung eines Mini-PCs in Form des Intel NUC 7PJYHN Celeron J5005.
Gründe für den Wechsel der Plattform sind unter anderem die Erweiterbarkeit des IT-Systems und dessen Hardware. Beim GM100 ist eine Erweiterung des Speicherplatzes ohne weitere Hardware-Änderungen entgegen der Hersteller Vorschriften nicht möglich. Da für eine Langzeitdatenspeicherung der aufzunehmenden Motordaten, jedoch auch kein Cloud-Abonnement-Modell in Frage kommt, wird beschlossen eine lokale Datenspeicherung durchzuführen. Der Intel NUC bietet den Vorteil der dynamischen Speichererweiterung. So wird bei der Anschaffung des Gerätes beschlossen, wie viel Speicherplatz benötigt wird. In diesem Fall legt man sich auf einen Terrabyte in Form eines SSD-SATA-Speicher fest. Dieser bietet dem Anwender neben einer hervorragenden Lese- und Schreibleistung auch ein gutes Preis-Leistungsverhältnis.
Des Weiteren bietet die Intel Plattform mehr Flexibilität, wenn es um die Kompatibilität von Software-Tools geht. Im Vergleich zur ARM-Plattform des GM100, ist auf dem Intel NUC die Möglichkeit geboten, eine Virtualisierungsumgebung einzurichten. Diese wird im Folgenden näher beschrieben.

1.2. Virtualisierungsumgebung Proxmox

Für die Ausführung der benötigten Software-Infrastruktur wird beschlossen, eine Virtualisierungsumgebung zu verwenden. Diese bietet die Vorteile der abgegrenzten Dienstausführung und einem damit erweiterten Schutz der einzelnen Software-Programme. Zum Einsatz kommt deshalb die kostenlose Community-Version des Betriebssystems Proxmox. Dieses wird auf den Intel NUC aufgespielt und kann anschließend über eine TCP-Verbindung und die dazugehörige Weboberfläche gesteuert werden. Nach erfolgter Einrichtung werden alle verfügbaren Updates auf das System aufgespielt. Des Weiteren kann überprüft werden, ob die verbaute Hardware korrekt erkannt und verwendet werden kann. In

unserem Fall wurden 16 GB DDR4 3,2 GHz Arbeitsspeicher, sowie ein 1 TB großer SSD-Speicher verbaut. Alle Komponenten wurden ohne Probleme erkannt.
Der für den IoT-Stack vorgesehene LXC-Container wird erstellt und bekommt eine entsprechende volle Hardwarezuweisung, da er derzeit der einzige Container auf dem System ist. Aufgrund der Kompatibilität der Software-Tools miteinander wird beschlossen, alle Tools in einem Container parallel zu betreiben. Für weitere Gruppen des Gesamtprojektes wird zudem die Möglichkeit geboten, eigene IoT-Skripte und Tools auf dem Intel NUC zu platzieren. Diese würden dann in separaten Containern ausgeführt werden, damit eine Trennung der unterschiedlichen Gruppen-Dienste gegeben ist.

1.3. Implementierung des Software-Stacks

Der IoT-Stack wurde auf einem LXC-Container implementiert, auf welchem das Betriebssystem Linux Debian in der Version 14.02 als Basis installiert wird. Nach der erfolgreichen Einrichtung des Containers wurde der Docker-Dienst installiert. Der Intel NUC erhält die Netzwerk-Adresse 192.168.0.70. Unter dieser Adresse ist auch der Proxmox Webservice zu erreichen. Dafür wird die IP mit dem Port 8006 als HTTP-Webseite aufgerufen. Die Installation erfolgt mittels der folgenden Befehle über den Tab „Console“ des Containers in der Proxmox Weboberfläche:

Allgemeine Docker und Docker Compose Installation:

- *apt update && upgrade -y*
- *apt install curl -y*
- *curl -fsSL https://get.docker.com -o get-docker.sh*
- *sh get-docker.sh*
- *mkdir -p ~/.docker/cli-plugins/*
- *curl -SL https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose*
- *chmod +x ~/.docker/cli-plugins/docker-compose*

Im Anschluss wird der Service „Portainer“ über das Docker Toolkit installiert. Dies erfolgt mit Hilfe der folgenden Befehle:

Installation des Docker Dienstes Portainer:

- *docker volume create portainer_data*
- *docker run -d -p 8000:8000 -p 9000:9000 --name=portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-ce*

Der Dienst Portainer ist über die IP-Adresse des erstellten IoT-Containers (192.168.0.71) und den Port 9000 aufrufbar. In Portainer können die Docker Container, die im Folgenden angelegt, grafisch verwaltet werden. Die Verwaltung kann auch direkt über die Docker CLI erfolgen, jedoch verfügt die CLI über keine Funktionalitäten, wie das Erstellen von Software Stacks, die gemeinsam erstellt und gesteuert werden können.
Die Implementierung des Stacks erfolgt mit Hilfe des folgenden Docker Compose Files:

IoT-Stack Docker Compose File:

```
version: "3.8"
services:
  node-red:
    container_name: nodered
    image: nodered/node-red:latest
    restart: always
    environment:
      - TZ=Europe/Berlin
    volumes:
      - nodered-data:/data
    ports:
      - "1880:1880"
    networks:
      - node-red-net
```

```
  mqtt:
    container_name: mqtt
    image: eclipse-mosquitto:latest
    restart: always
    volumes:
      - /srv/mosquitto/config:/mosquitto/config
      - /srv/mosquitto/data:/mosquitto/data
      - /srv/mosquitto/log:/mosquitto/log
    ports:
      - "1883:1883"
      - "9001:9001"

  timescaledb:
    container_name: timescaledb
    image: timescale/timescaledb-ha:pg14-latest
    restart: always
    volumes:
      - timescaledb-data:/home/postgres/pgdata/data
    ports:
      - "5432:5432"
    environment:
      - POSTGRES_PASSWORD=hubadmin

  grafana:
    container_name: grafana
    image: grafana/grafana-oss:latest
    restart: always
    ports:
      - "3000:3000"
    volumes:
      - grafana-data:/var/lib/grafana

networks:
```

node-red-net:

volumes:

nodered-data:

timescaledb-data:

grafana-data:

Nach dem erstmaligem Hochfahren des Software-Stacks mit seinen Einzelkomponenten sieht die Portainer-Weboberfläche wie in Abbildung 1 zu erkennen ist aus.

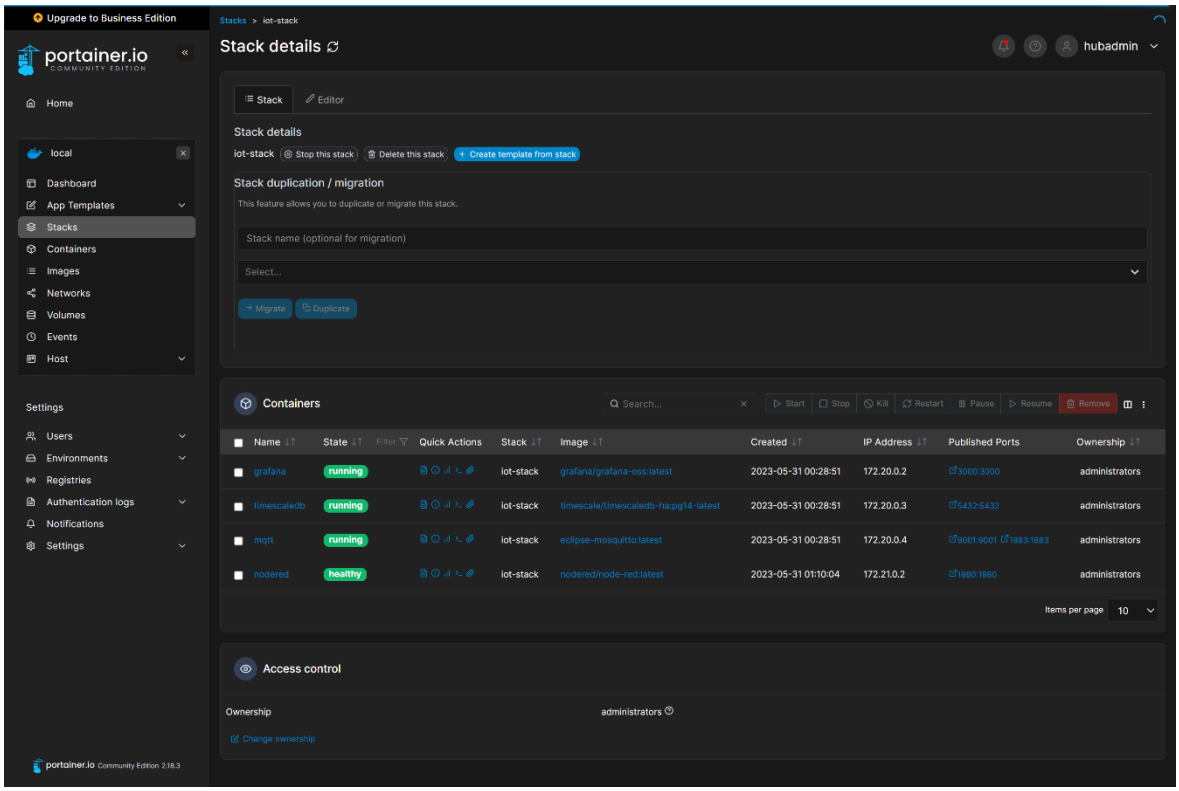


Abbildung 1: Portainer Weboberfläche des implementierten IoT-Stacks

Zu erkennen sind die vier unterschiedlichen Service-Container, auf die im Folgenden weiter eingegangen wird. Portainer bietet die Möglichkeit auf eine simple Art und Weise mit den Docker Containern zu interagieren. Hierfür stehen beispielsweise die Quick Actions zur Verfügung, mit denen die einzelnen Logs der Dienste schnell eingesehen werden können, ohne eine Vielzahl von SSH-Sessions mit den LXC-Containern und Befehlen einzugeben. Des Weiteren lassen sich die Dienste Starten, Stoppen oder Unterbrechen. Dies erleichtert die Steuerung des Stacks im Falle der Einrichtung und späteren Instandhaltungsarbeiten erheblich.

2. Erläuterung der Dienst-Funktionalitäten

Im Folgenden wird auf die verschiedenen Dienste des IoT-Stacks, so wie die Darstellungs-Software Peakboard eingegangen. Die Struktur und Überlegungen zur Umsetzung, sowie die angewandten Methoden werden aufgezeigt und erklärt. Dabei soll dieses Kapitel als Wissens-Ansatz für die Weiterentwicklung der bisher etablierten Mechanismen dienen.

2.1. MQTT-Broker

Der Kommunikationsdienst MQTT stellt den Broker des gleichnamigen Nachrichten-Protokolls „Message Queuing Telemetry Transport“ dar. Dieser Dienst läuft permanent im Hintergrund des Stacks und akzeptiert Nachrichten von allen Teilenehmern, die sich mit dem Broker verbinden. Im Anschluss verteilt der Broker die empfangenen Nachrichten an die Teilnehmer, die die entsprechenden Topics im Vorfeld abonniert haben. Dies ermöglicht beispielsweise die Kommunikation zwischen dem Dienst NodeRED und der Visualisierungs-Software Peakboard.

2.2. NodeRED

Mit Hilfe des Tools NodeRED können Dataflows erstellt werden, mit welchen eingehende Informationen verarbeitet aufbereitet und weitergeleitet werden können. Hierbei ist vor allem die Verwendung des S7-Connectors von Vorteil, da mit Hilfe von diesem eine direkte Anbindung an die S7-1500 SPS von Siemens hergestellt werden kann. Ein nicht optimierter Datenbaustein im Programm der SPS wird für den Service unter einem Vollzugriff und der Gestattung von GET- und POST-Methoden freigegeben. Somit kann ein Zugriff auf den Baustein und die darin enthaltenen Daten erfolgen.

Einer der Dataflows extrahiert mit Hilfe des Connectors die Motorzustandsdaten aus dem SPS-Baustein. Die Informationen werden hier von der SPS abgelegt. Sie erhält diese Daten von der an das PROFINET angebundenen Schwingungsüberwachung ifm VSE150. In Abbildung 2 ist der beschrieben Flow zu erkennen.

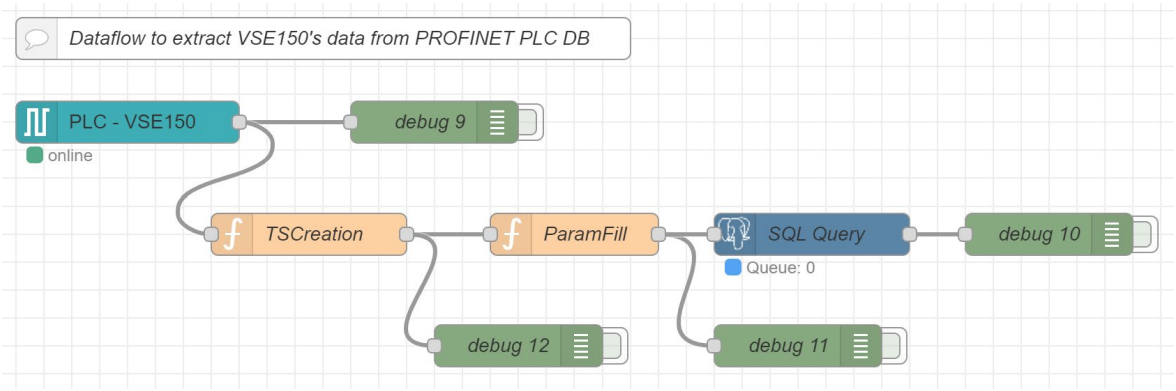


Abbildung 2: Dataflow zur Extraktion der Motordaten aus der SPS

Die Node „PLC – VSE150“ stellt hierbei die Verbindung in die SPS dar. Von dieser erhält der Flow zyklisch, jede Sekunde die Informationen über die aktuelle Schwingung, Temperatur, Drehzahl und Alarme des Motors. Diese Informationen werden in den nachfolgenden JavaScript-Nodes aufbereitet und somit für das Abspeichern in der Timescale Datenbank vorbereitet. In der Node „TSCreation“ wird ein Timestamp erstellt, welcher weitergeben wird. Die Node „ParamFill“ erstellt ein Array von Variablen, welches mit den Motordaten gefüllt und anschließend in der SQL-Query mit samt des Timestamps an die Datenbank verschickt werden.

Das Auslesen des zuletzt eingescannten Bauteil QR-Codes und der derzeit angewählten Produktvariante erfolgt nach dem gleichen Schema und ist in Abbildung 3 zu erkennen.

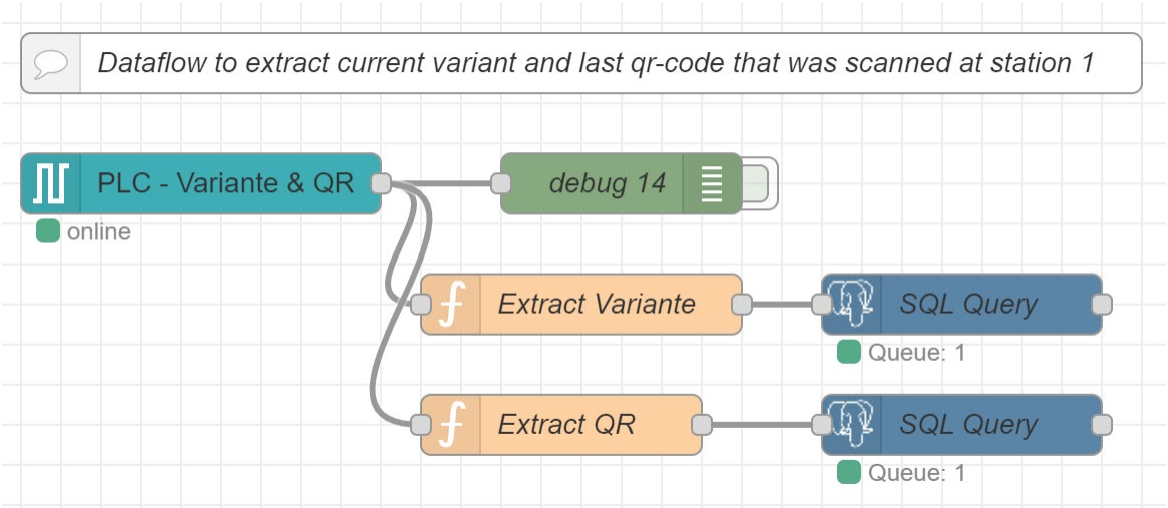


Abbildung 3: Dataflow zur Extraktion der QR-Code und Varianteninformation

Die Informationen werden wiederum zyklisch aus der SPS abgefragt, in den darauffolgenden Nodes aufbereitet und zuletzt in die Datenbank abgespeichert, indem ein statischer Tabellenplatz mit Hilfe der jeweiligen SQL-Query aktualisiert wird.

Weiterhin wird die Auf- und Ablage von Bauteilen verfolgt. Da zum Zeitpunkt der Implementierung noch keine Qualitätsinformation der Gruppe des End-of-Line-Testings zur Verfügung stand, werden nur die tatsächlichen Auf- und Ablagen ermittelt. Die Unterscheidung zwischen IO und NIO Bauteilen erfolgt noch nicht, kann aber ohne großen Mehraufwand zu einem späteren Zeitpunkt in den bestehenden Dataflow hinzugefügt werden. Abbildung 4 zeigt den Ablauf der Datenaufbereitung.

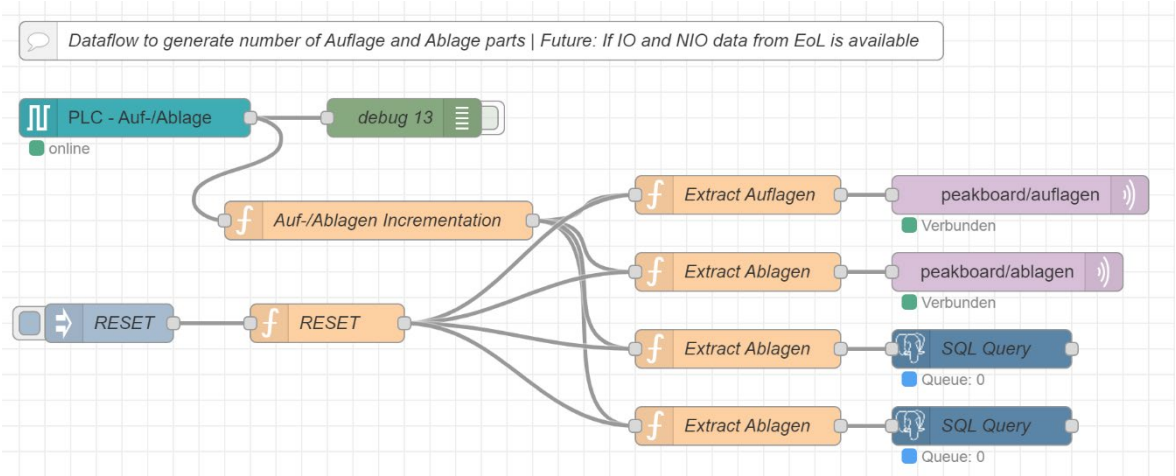


Abbildung 4: Dataflow des Auf- und Ablagen-Zählers

Den Beginn markiert die S7-Connector Node, welche zyklisch die benötigten Sensordaten von der SPS bezieht. Diese werden in der JavaScript-Node „Auf-/Ablagen Incrementation“ interpretiert und anschließend an die Kommunikationsschnittstellen weitergeleitet. Die nachfolgenden „Extraction“-Nodes bereiten die Daten für die jeweilige Schnittstelle auf. Der Versand der Informationen findet in diesem Fall über das MQTT-Protokoll und über die SQL-Querys statt.

Des Weiteren ist auf der linken Seite eine manuelle „RESET“-Injection Node zu erkennen, mit welcher die globalen Kontext-Variablen der Auf- und Ablagen zurückgesetzt werden. An die Informationskanäle wird parallel der Wert 0 versendet.

Um die Anzahl der Ladungsträger zwischen den einzelnen Stationen zu bestimmen, wird der Flow, der in Abbildung 5 dargestellt ist, verwendet.

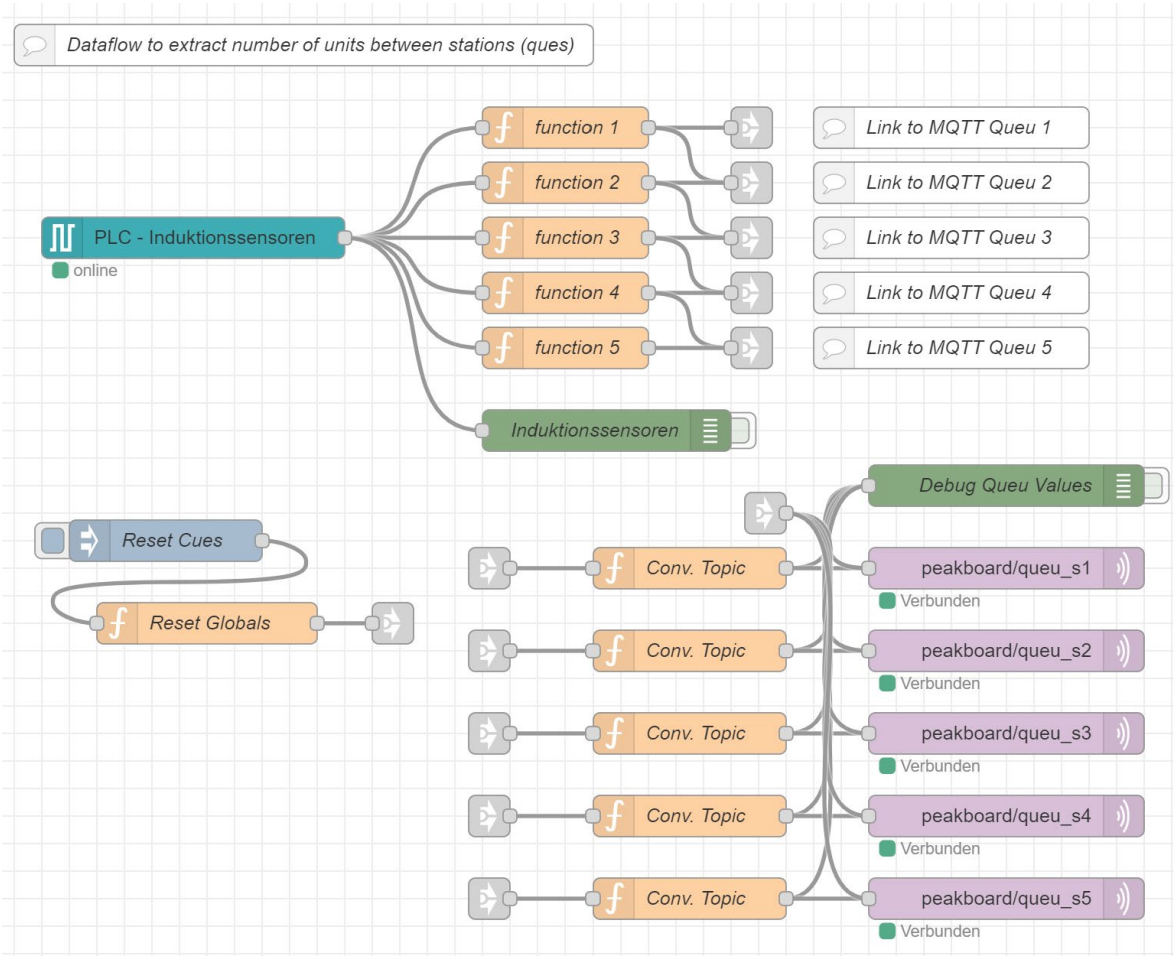


Abbildung 5: Dataflow zur Bestimmung der Ladungsträgeranzahl zwischen den Stationen

Hierbei werden die Induktionssensoren in der Anlage auf deren Status abgefragt. Kommt es zur Veränderung an einem der Sensoren kann eine Bauteilbewegung im Bereich der jeweiligen Station interpretiert werden. Die Logik der oberen fünf JavaScript Nodes übernimmt diese Aufgabe. Der Algorithmus in den Nodes sieht vor, globale Zählervariablen zu in- oder dekrementieren, je nachdem, ob ein Ladungsträger in den Stationsweg ein- oder ausgefahren ist. Die Informationen werden dann an die MQTT Kommunikations-Nodes weitergeleitet. Auf der linken Seite ist wiederum eine Reset-Node zu erkennen, die das System und dessen Ausgangswerte auf den Wert 0 zurücksetzt.

2.3. Datenbank TimescaleDB

Die aktuellen Anlagen- und Motorzustände werden innerhalb des TimescaleDB Dienstes gesichert. Der Vorteil der TimescaleDB die auf der PostgreSQL Plattform basiert, besteht in der Möglichkeit, gleichzeitig relationale, sowie zeitreihenbasierte Daten abzuspeichern. Dies deckt den vorliegenden Anwendungsfall ab, da Statusinformationen, wie bspw. der aktuelle Zustand der Anlage bestehend aus den Möglichkeiten „fahrend“ oder „stehend“, statisch abgespeichert werden und nur der aktuelle Zustand von Bedeutung ist. Es muss nicht zwingend eine Datenhistorie zur Verfügung gestellt werden. Anders ist dies bei den Daten des Motorzustands. Für spätere „predictive maintenance“ Anwendungsfälle muss auf eine Datenbasis zurückgegriffen werden können. Dabei sollten Informationen von einem möglichst langen Zeitraum zur Verfügung stehen, um Verhaltens- und Alterungsmuster in den Datensätzen analysieren zu können.

Nach Anbindung des IoT-Stacks mittels NodeRED an das Anlagennetz, erhält die Datenbank alle benötigten Informationen. Die Motordaten werden in der Tabelle „data_motor_1“ gespeichert, welche in Abbildung 6 zu erkennen ist.

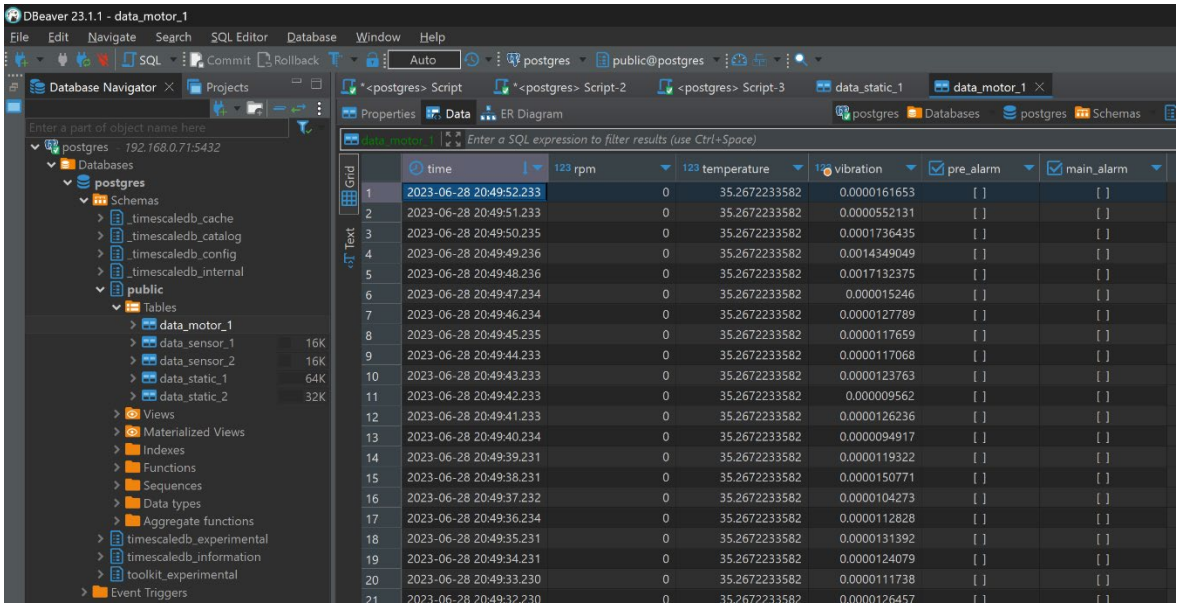


Abbildung 6: Tabelle der aktuellen Motordaten

Zu erkennen sind die Spalten „time“, welche den in NodeRED erstellten, aktuellen Zeitstempel enthält. Die Spalte „rpm“ , welche die zu diesem Zeitpunkt vorherrschende Drehzahl des Motors sichert. Es folgt die Spalte mit der Motortemperatur, sowie des aktuellen Schwingungswertes. Während die beiden booleschen Spalten Vor- und Hauptalarm das Ende bilden.

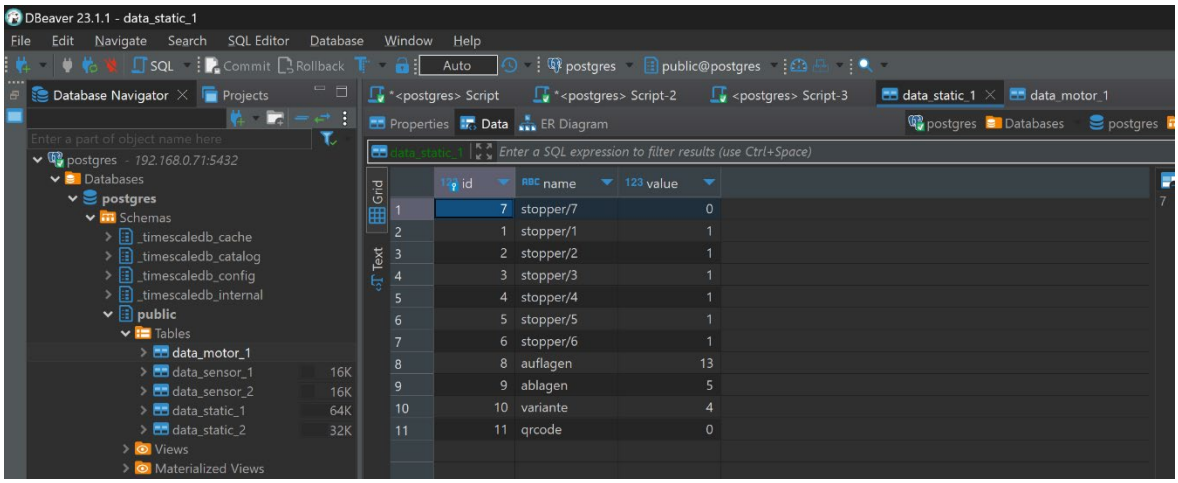


Abbildung 7: Tabelle mit den aktuellen, relationalen Anlagenzuständen

Abbildung 7 zeigt die Status-Daten des Anlagensystems. Diese werden bei einem Neuzustand mit dem aktuellen Zustand überschrieben und bedienen sich daher der relationalen Funktionalität des Datenbanksystems. Zu erkennen sind bspw. die Zustände der Stopperaktoren der Anlage, sowie die Anzahl der Auf- und Ablagen von Bauteilträgern. Zuletzt werden noch die aktuell vorherrschende Bauteilvariante und der zuletzt eingescannte QR-Code in der Tabelle gesichert.

2.4. Datenvisualisierung mittels Grafana

Für eine übersichtliche Darstellung der gesammelten Daten eignet sich das Softwaretool Grafana. Nach der Anbindung an die lokale Datenbank, können mit Hilfe von einfachen Querys die Daten in eigens kreierte Diagramme geladen werden. Diese sind flexibel in ihrer Formatierung und Anpassung, sodass komplexe Datenzusammenhänge simpel und verständlich dargestellt und interpretiert werden können.

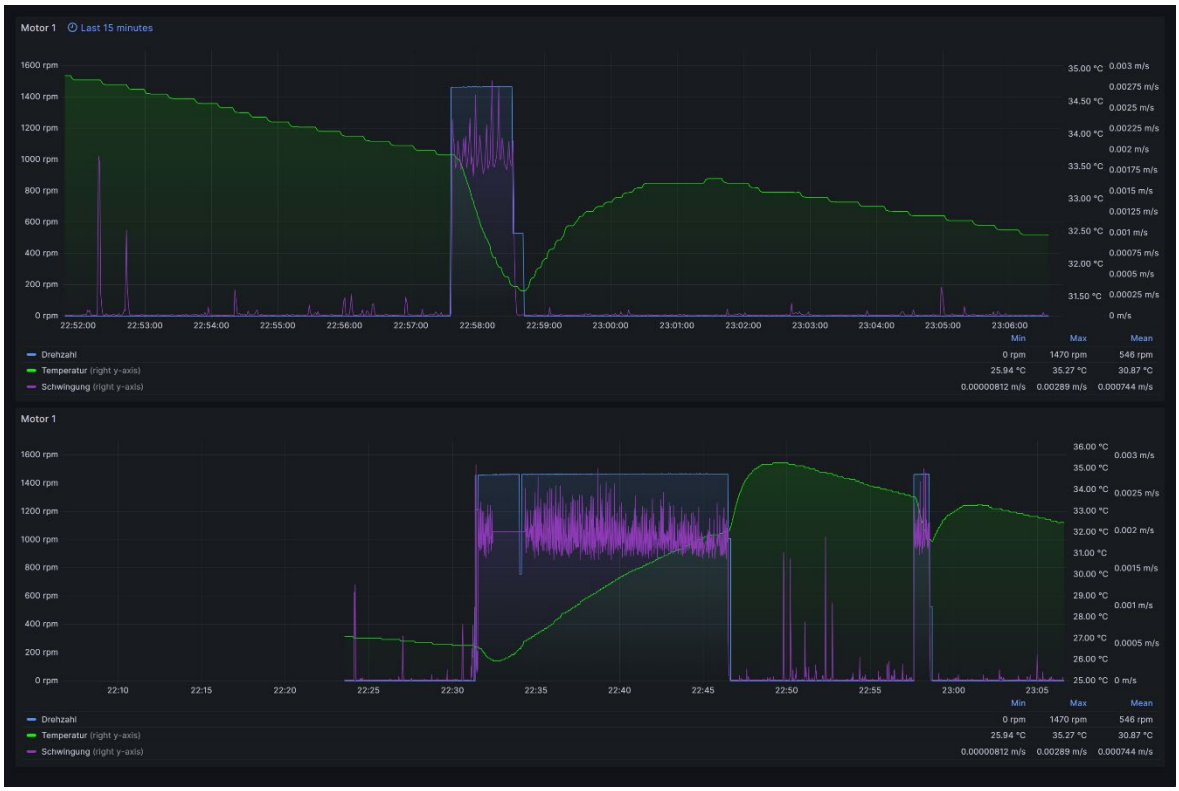


Abbildung 8: Visualisierung der Motorzustandsdaten

In Abbildung 8 ist der Zustandsverlauf des Bandmotors zu erkennen. Beide Diagramme zeigen dieselben Informationen, jedoch wird im oberen Diagramm ein Datenausschnitt der letzten 15 Minuten und im unteren ein variabler Zeitraum dargestellt. In grün wird die Temperatur, in blau die Drehzahl und in lila die Schwingung des Motors visualisiert.

Es ist deutlich zu erkennen, wann der Motor in Betrieb war und wie sich die aufgenommenen Parameter in diesem Zeitraum verändern. So ist zu Beginn des Motorstarts ein Abfallen der internen Temperatur zu verzeichnen, da das Lüfterrad mit seiner Kühlfunktion in Kraft tritt (vgl. oberes Diagramm, Bereich mit stark absinkender grüner Kurve). Jedoch wird der Motor im Verlauf der Zeit wieder wärmer als in seinem Ruhezustand, natürlich abhängig von der vorherrschenden Umgebungstemperatur. Dies zeigt das untere Diagramm im Bereich zwischen 22:30 Uhr und 22:45 Uhr. Anfangs sinkt die grüne Kurve kurzzeitig ab, bevor sie dann über den Verlauf der Zeit wieder kontinuierlich zu steigen beginnt. Auch die Schwingungswerte erhöhen sich deutlich beim Einschalten des Motors wie an der lila farbigen Kurve und deren Ausschlag, während des sich bewegenden Motors zu erkennen ist.

3. Das Peakboard als Schnittstelle zum Menschen

Mit Hilfe des großen Fernsehers im Bereich der Anlage der Zukunftsfabrik, können die gesammelten und aktuellen Anlagenzustände visualisiert werden. Hierzu ist es notwendig eine übersichtliche und einfach zu verstehende Oberfläche zu gestalten. Diese soll den Arbeitern im Anlagenbereich in möglichst kurzer Zeit, wichtigen Informationsgehalt bereitstellen.

Dies erfolgt in unserem Fall mit Hilfe der in Abbildung 9 dargestellten Visualisierung.

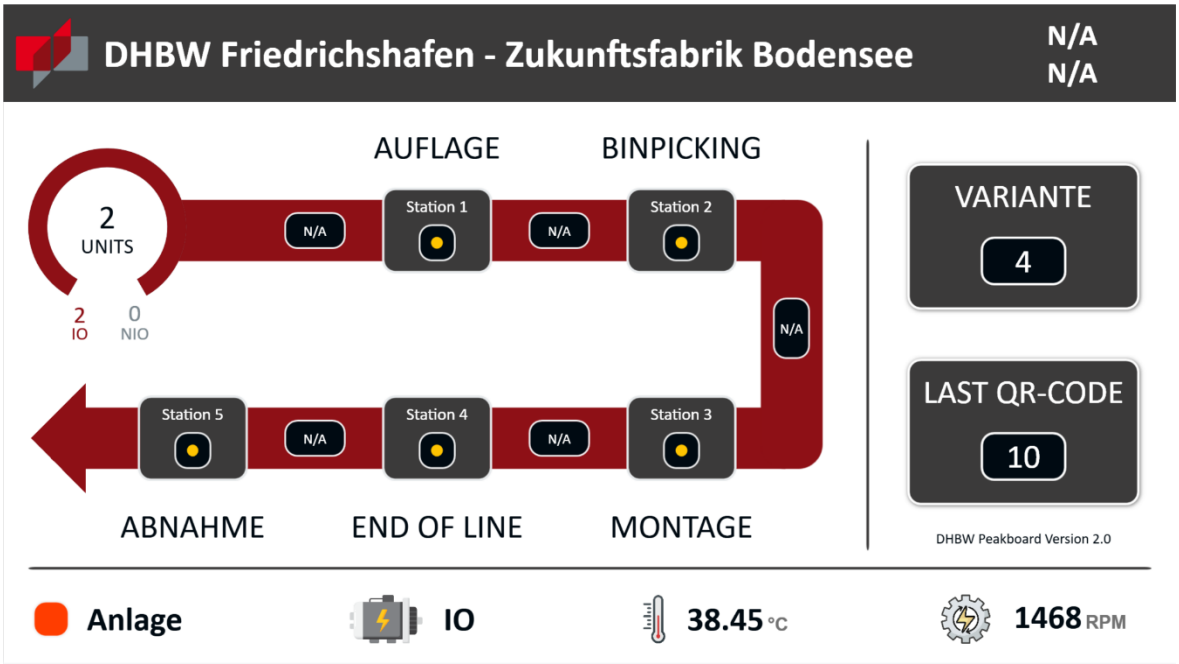


Abbildung 9: Peakboard Visualisierungsoberfläche

Im Kopf der Oberfläche sind die Anlagenbezeichnung „Zukunftsfabrik Bodensee“, sowie das Datum und die aktuelle Uhrzeit platziert.

Auf der linken Seite ist das Anlagenabbild zu erkennen. Dieses startet im linken oberen Eck mit einem Kreisdiagramm, welches die Auflage an Bauteilen und deren IO- und NIO-Gliederungen darstellt. Im weiteren Verlauf der Anlagenvisualisierung sind immer wieder schwarze Rechtecke zu erkennen, welche im laufenden Betrieb die Anzahl der Ladungsträger und Bauteile auf den Bandabschnitten zwischen den Stationen enthalten. Die Stationsblöcke besitzen einen Farbindikator der Aufschluss über den aktuellen Zustand innerhalb der Station geben soll. Diese Indikatoren sind noch nicht vollständig implementiert und zeigen derzeit nur rot oder grün, in Abhängigkeit des Gesamtanlagenstatus, an.

Die rechte Seite hält Informationen über die aktuell eingestellte Bauteilvariante und den zuletzt eingescannten Bauteil-QR-Code.

Die Fußzeile gibt Aufschluss darüber, in welchem Zustand sich die Anlage und der mit Sensorik versehene Bandmotor befinden. Der Indikator „Anlage“ wechselt von rot zu grün, sobald es keinen Nothalt mehr im System geht und das Band normal fährt. Die nächste Variable gibt Aufschluss über den Gesundheitszustand des Motors. Kommt es zum Einsetzen des Voralarms, färbt sich der Text „IO“ gelb ein. Sobald es zum Hauptalarm kommt, springt der Status in ein rotes „NIO“ um. Des Weiteren werden die Temperatur und die aktuelle Motordrehzahl dargestellt.
Für den Erhalt der Daten werden im Peakboard Designer die Schnittstellen S7-Connector, MQTT und PostgreSQL verwendet.
Zudem wird geraten die Peakbox in das bestehende Studenten-Netzwerk einzubinden, damit ein Zeitabgleich mit einem Internet-NTP-Server durchgeführt werden kann. Alternativ wurde versucht einen lokalen NTP-Server auf der S7-1500 SPS zu installieren, jedoch wurde dieser Weg aufgrund seiner Komplexität verworfen, da es zu Problemen mit der Synchronisation zwischen Peakbox und der SPS kam.

4. Informationen zum Arbeiten mit dem IoT-Stack

Die Zugangsdaten für alle Systeme und Geräte wurden beim zuständigen Betreuer hinterlegt und können dort abgerufen werden.
Die folgenden IP-Adressierungen wurden für Soft- und Hardwaregeräte gewählt:

Tabelle 1: IP-Adressierung der Soft- & Hardwaregeräte

Soft- / Hardwaregerät	IP-Adresse
S7-1500 SPS	192.168.0.1
Peakbox	192.168.0.60
Proxmox	192.168.0.70
IoT-Stack (NodeRED, TimescaleDB,…)	192.168.0.71
VSE150	192.168.0.72

Für das Installieren von Updates oder funktionserweiternden Paketen auf dem Intel NUC, ist in der Regel eine Internetverbindung nötig. Diese kann hergestellt werden, indem ein persönlicher Computer, der einen aktiven Internetzugriff besitzt, direkt mit dem NUC per Ethernet verbunden wird. Anschließend muss im Netzwerk- und Freigabecenter die Internetverbindung des entsprechenden Netzwerkadapters für den Ethernet-Adapter freigegeben werden.

Literaturverzeichnis

-