DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Condition Monitoring eines Transfersystems

Studienarbeit

im Studiengang Elektrotechnik / Automation

an der DHBW Ravensburg
Campus Friedrichshafen

von
Marius Gerstner, Tobias Anselm, Malte Neumann

16.01.2023

Bearbeitungszeitraum:	5. Theoriephase
Matrikelnummer, Kurs:	7643074, 9367555, 5747438, TEA20
Betreuer:	Florian Bulander

Eidesstattliche 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 Seminararbeit mit dem Thema

Condition Monitoring eines Transfersystems

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.

Ort, Datum	Unterschrift (Marius Gerstner)
Ort, Datum	Unterschrift (Tobias Anslem)
Ort, Datum	Unterschrift (Malte Neumann)

Vorwort und Danksagung

Industrie 4.0 und Condition Monitoring sind eines der wichtigsten Themen in vielen produzierenden Unternehmen. Als Studenten des Studienganges Elektrotechnik Automation ist das Thema von besonderer Wichtigkeit. Im Rahmen der Studienarbeit im Produktionsnetzwerk Fallenbrunnen der DHBW Ravensburg Campus Friedrichshafen wird uns ermöglicht die Themengebiete auf Praxisebene näher kennenzulernen und zu erproben, sowie neue Technologien auszuprobieren. Es wird ein guter Einblick im projektorientierten Arbeiten zusammen mit verschiedenen Teams geschaffen.

Wir möchten uns daher für diese Möglichkeit bei allen beteiligten bedanken. Allen Betreuer und wissenschaftlichen Mitarbeitern des PTZ, sowie allen anderen Studenten die am Großprojekt teilnehmen und ihre Studienarbeit verfassen.

Besonderer Dank gilt hierbei unseren direkten Betreuern Florian Bulander und Thomas Dietmüller.

Des Weiteren geht ein Dank an Wolfgang Müller der uns als Ansprechpartner im Labor bzw. in der Werkstatt zur Seite stand.

Kurzfassung

Während mit fortschreitendem Automatisierungsgrad moderner Produktionsanlagen die Komplexität der Anlage an sich, aber auch die möglichen Funktionen stetig steigen, wird auch die Fehleranfälligkeit höher. Moderne Industrieanlagen sind ein komplexes Netzwerk aus elektrischen, pneumatischen, hydraulischen und mechanischen Sensoren und Aktoren. Vor allem auf der Seite der Sensorik kam es in der Vergangenheit zu einem fortschreitenden Wandel. Durch neue Sensoren und vor allem neue Möglichkeiten der Vernetzung, Speicherung und anwendungsbezogenen Auswertung der Daten, ergeben sich völlig neue Möglichkeiten, die Produktivität möglichst zu steigern, Kosten gleichzeitig zu senken und Fehlerquellen zu erkennen und zu eliminieren.

Eine der wichtigsten Methoden hierzu ist das sogenannte „Condition Monitoring“(CM) (Dt. „Zustandsüberwachung“). Im Rahmen des Condition Monitorings werden verschiedenste Sensoren an und um die Anlage angebracht, die unterschiedlichste Daten aufnehmen, anhand deren Infos über die Produktion und die verschiedenen Teile der Anlage gewonnen werden können. Anhand dieser Daten können Zustände bestimmt und Vorhersagen, beispielsweise zum Lebenszyklus von Bauteilen und deren Restlebensdauer getroffen werden. Daraus ergeben sich viele Vorteile. Man kann große Teile der „Planned Maintenance“, also dem zyklischen Warten und Austauschen von Bauteilen durch die sogenannte „Predictive Maintenance“, dem Tauschen der Teile vor vorhergesagtem Bauteilversagen ersetzen. Teile können zum benötigten Zeitpunkt bestellt und nur in geringen Stückzahlen auf Lager gehalten werden. Bei unvermeidbaren Gewaltbrüchen von Bauteilen können Folgeschäden durch sofortiges Anhalten der Anlage minimiert werden. Darüber hinaus kann indirekt die Produktqualität verbessert und Ausschuss reduziert werden.

Im Rahmen dieser zweiteiligen Studienarbeit soll eine Produktionsanlage im Produktionszentrum (PTZ) am Campus Friedrichshafen entstehen. In diesem Projektbericht wird darauf eingegangen, wie die Implementierung eines Condition Monitorings für ein solches Produktionssystem durchgeführt wird. Als Basis dient in diesem Projekt ein IoT-Gateway der Marke inHub, welches schon zum Projektstart vorgegeben wurde. Neben der benötigten Hardware, zur Datensammlung wird ebenfalls die verarbeitende Software vorgestellt. Es wird genauer auf die Programmierung und die entsprechenden Prozesse eingegangen. Des Weiteren wird die Produktionsanlage auf Bereiche untersucht, in welchen das CM seinen größten Nutzen ausbilden kann. Für diese Bereiche wird dann die Etablierung der Datenüberwachung dargelegt. Zuletzt werden anhand der eingebrachten Sensorik Beispieldaten aufgenommen, weiterverarbeitet und die dadurch gewonnenen Erkenntnisse präsentiert.

Inhaltsverzeichnis

Eidesstattliche Erklärung **II**

Vorwort und Danksagung **III**

Kurzfassung **IV**

Abkürzungsverzeichnis **VII**

Abbildungsverzeichnis **VIII**

Tabellenverzeichnis **IX**

1 Einführung **1**

2 Grundlagen **3**

2.1 Allgemeine Instandhaltungsansätze 3

2.2 Instandhaltungsstrategien 5

2.3 Condition Monitoring 6

2.3.1 Condition Monitoring System 7

2.3.2 Vor- und Nachteile des Condition Monitoring 8

3 Fallbeispiel Förderbandmotor **10**

3.1 Aufbau des Motors 10

3.2 Indikatoren 12

3.3 System und Hardware 13

3.3.1 Motorstromanalyse 13

3.3.2 Vibrationsanalyse 14

3.3.3 Temperaturanalyse 15

3.3.4 Filtern und Grenzwerte 16

4 Einleitung zur Implementierung des Condition Monitoring Systems **19**

4.1 Zielsetzung bei der Implementierung des Condition Monitoring Systems . . 19

4.2 IO-Situation, Protokollfähigkeiten und technische Daten des GM-100 . . . 19

4.3 Erstinbetriebnahme über serielle und Netzwerkschnittstelle 20

4.4 Vorbereitung der Softwaretools 22

5 Softwareseitige Implementierung des Condition Monitoring Systems **24**

5.1 Docker als Basis für Softwarepakete 24

5.1.1 Vorteile von Docker 24

5.1.2 Erstellen und Verwenden von Docker Containern 25

5.2 MQTT-Netzwerk als Kommunikationsnetzwerk 25
5.2.1 MQTT-Broker 26
5.2.2 Interner MQTT-Client zum Empfangen von Daten 27
5.2.3 Floweditor Node-RED 28
5.3 TimescaleDB zur Archivierung von Informationen 29
5.3.1 Relationale und Zeitreihen Datenbanken 29
5.3.2 TimescaleDBs Konzepte der Datenverwaltung 30
5.3.3 Installation und Verwendung der TimescaleDB 31
5.4 Grafana zur grafischen Datenauswertung 32
5.4.1 Einrichtung von Grafana 32
5.4.2 Datenvisualisierung und Erstellung von Dashboards 32
5.4.3 Alarmierung bei Grenzwertüberschreitung 33
5.5 Implementierung eigener Softwarepakete 34
5.5.1 Python Skripte in Form von Docker Containern 34
5.5.2 Umgebungsbedingungen und Abläufe festlegen mittels Dockerfile . . 35
5.5.3 Python Programm zur Exportierung von Daten zu Analysezwecken 35
5.6 Ausblick zur weiteren Datenverarbeitung 36
5.7 MES-Dashboard mit Peakboard Designer 37

6 Fazit **39**

A Softwareversionen **40**

B InCore Programm – MQTT-Broker **41**

C InCore Programm – MQTT-Database-Interface **42**

D Dockerfile **45**

E Python Programm **46**

Literatur **48**

Abkürzungsverzeichnis

CM ..... Condition Monitoring
CMS ..... Condition Monitoring System
GPIO ..... General Purpose Input Output
HMI ..... Human machine interface
IIot ..... Industrial Internet of Things
IoT ..... Internet of Things
MES ..... Manufacturing Execution System
MQTT ..... Message Queuing Telemetry Transport
REST ..... Representational State Transfer
SFTP ..... Secure File Transfer Protocol
SQL ..... Structured Query Language
SSH ..... Secure Shell
UART ..... Universal Asynchronous Receiver Transmitter

Abbildungsverzeichnis

2.1 Bereiche der Instandhaltung - Quelle: forcam GmbH 3
2.2 Entwicklung der Instandhaltung - Quelle: Bauer und Dremel 4
2.3 Strategien der Instandhaltung - Quelle: Bauer und Dremel 6
2.4 Condition Monitoring System - Quelle: Bauer und Dremel 8
3.1 Aufbau eines Motors 10
3.2 Messaufbau einer Motorstromanalyse 14
3.3 Vibration Messung an Motor 15
3.4 Condition Monitoring Algorithms in MATLAB - Quelle: Adam Jablonski . 17
4.1 SIINEOS Management Console 21
4.2 SSH-Login und Erlangung von Superuser-Rechten 22
4.3 WinSCP Benutzeroberfläche zur Dateiverwaltung 23
4.4 DBeaver Benutzeroberfläche zur Verwaltung von Datenbanken 23
5.1 Das Prinzip von Docker und dessen Containern - Quelle: Joshua Cook . . . 25
5.2 Auflistung der aktiven Docker Container 26
5.3 Schematische Funktionsweise eines MQTT-Netzwerkes - Quelle: Ignition Edge IIoT 27
5.4 Weboberfläche des flowbasierten Editor Node-RED 28
5.5 Funktionsweise des kontinuierlichen Datenaggregats mit automatischer Aktualisierungsrichtlinie - Quelle: Timescale 30
5.6 Grafana Reiter Erkunden 32
5.7 Dashboard der CPU-Temperatur mit rohen und aggregierten Daten 33
5.8 Slack Benachrichtigung ausgelöst durch Grafana Alarm 34
5.9 Ausschnitt des Datenexports durch das Python Programm 36
5.10 Matlab Toolbox Diagnostic Feature Designer 37
5.11 MES-Dashboard aus dem Peakboard Designer 38
6.1 Vereinfachte Darstellung der Prozessabläufe des IoT-Gateways 39

Tabellenverzeichnis

2.1 Begriffe für die Grundlagen der Instandhaltung nach DIN 31051:2019-06 . 4
A.1 Softwareversionen zum Zeitpunkt der Erstellung dieser Arbeit 40

1 Einführung

Diese Studienarbeit ist Teil des Produktionsnetzwerk Fallenbrunnen. „Das Produktionsnetzwerk am Campus Fallenbrunnen soll eine reale Wertschöpfungskette mit vernetzten, adaptiven und selbstlernenden Systemen abbilden können. Gemeinsam mit der bestehenden Lernfabrik des IWT entsteht somit ein einzigartiges Produktionsnetzwerk. In dem Netzwerk soll ein „Smart Product“ in hoher Variantenvielfalt gefertigt und im Sinne der Kreislaufwirtschaft wieder zurückgeführt werden. Mit dem Produktionsnetzwerk wird die Grundlage für eine zielgerichtete, praxisorientierte Forschung und Lehre in den Bereichen Digitaler Zwilling, KI in der Produktion, Mensch-Maschine-Interaktion, Produktionsplattformen sowie Kreislaufwirtschaft ermöglicht. Die Forschungsvorhaben der KI- und Produktionsexperten des Campus Fallenbrunnen sollen sich durch das Einbeziehen der regionalen KMU stets an den aktuellen Bedürfnissen der Industrie orientieren.“[1]

„Über ein mit Sensoren ausgestattetes Bandsystem soll das Produkt seine vier Stationen zur richtigen Zeit mit der richtigen Variante anfahren. Zu Beginn des Bandsystems wird mit Hilfe eines QR-Readers die benötigte Variante ins System eingelastet und das Produkt von dem definierten Bereitstellungsort mittels eines kollaborativen Roboters auf den Werkstückträger auf das Bandsystem gegeben. An der Station 2 werden Bolzen für die jeweilige Variante mittels Bin Picking kommissioniert und auf den Werkstückträger gegeben. Die dritte Station ist ein manueller Handarbeitsplatz. Dieser ist ergonomisch ideal ausgerichtet und beinhaltet eine entkoppelte Vormontage wie auch die finale Montage des Vorprodukts vor der Endprüfung im PTZ. Geführt wird der Werker durch ein hochmodernes Werkerassistenzsystem. Zum Ende des Bandes kommt die Endprüfung und die Übergabe auf das Transportsystem. Mit dem Transportsystem wird das Produkt über den Campus zur Lernfabrik geliefert, wo dann nach einer kurzen manuellen Montagetätigkeit die Endmontage mit einer automatisierten Schraubstation stattfindet. Diese Endmontage wird über einen weiteren Endprüfstand auf Richtigkeit geprüft, ehe das Produkt eingelagert und anschließend wieder für die Demontage bereitgestellt wird. Die Demontage erfolgt im selben Kreislauf wie die Montage.“ [2]

Während die meisten anderen Studienarbeiten sich mit einer bestimmten Station als solche befassen, behandelt das dreiköpfige Team hinter dieser Studienarbeit Condition Monitoring der gesamten Produktionsteils im PTZ. Es gilt zu untersuchen, an welchen Stellen Condition Monitoring lohnenswert betrieben werden kann. An den entsprechenden Stellen wird ein Condition Monitoring System implementiert. Hierfür steht ein System der Marke In-Hub zur Verfügung, das eine einfache Anwendung verspricht, die jederzeit erweitert werden kann, um auf eventuelle Änderungen in Zukunft reagieren zu können. Mit Hilfe dieser Condition Monitoring Lösung sollen Fehler in allen Teilen der Produktions-

[1] ZentrumFurDigitalisierung
[2] ZentrumFurDigitalisierung

linie frühzeitig erkannt und behoben werden. Dafür sind eine optimale Einbindung und entsprechende Überwachung aller einzelnen Anlagen notwendig.

2 Grundlagen

„Condition Monitoring“ hat sich als ganzheitliches und breit angewendetes Tool in der Instandhaltung etabliert. Der Fortschritt der vierten Industriellen Revolution beeinflusst auch diesen Bereich und wird stetig weiterentwickelt. Die Anforderungen an Systeme und Produktion in Bezug auf Sicherheit, Wirtschaftlichkeit und Flexibilität wachsen.[3] In diesem Kapitel werden die Grundlagen und Anforderungen, sowie die Möglichkeiten von allgemeinen Instandhaltungsansätzen sowie der des Condition Monitorings erläutert.

2.1 Allgemeine Instandhaltungsansätze

Die Instandhaltung wird nach DIN EN 13306 und DIN 31051 als die „Kombination aller technischen und administrativen Maßnahmen, einschließlich Überwachungsmaßnahmen, mit denen eine Einheit im funktionsfähigen Zustand erhalten oder in ihn zurückversetzt werden soll.“[4] Hiermit ergeben sich die vier Maßnahmen Bereiche Inspektion, Wartung, Instandsetzung und Verbesserung.[5]



Abb. 2.1: Bereiche der Instandhaltung - Quelle: forcam GmbH

Die Inspektion, also die Erfassung des Ist-Zustand, kann mithilfe von Prüfungen durchgeführt werden. Hierzu kommen Hilfsmittel wie zum Beispiel Lehren und oder verschiedene Messmittel aus der Messtechnik zum Einsatz. Der Zustand einer Anlage kann somit bewertet werden. Wartungen bewahren den Soll-Zustand. Maßnahmen aus diesem Bereich können das Putzen, Schmieren, Nachfüllen, Nachstellen und Konservieren sein. Sie gehören zur Prävention um den Abbau des vorhandenen Abnutzungsvorrates zu Verzögern. Sollte der Ist-Zustand nicht mehr dem Soll-Zustand entsprechen, so müssen Wartungen

[3] Bauer und Dremel, vgl. S. 3
[4] DEUTSCHES INSTITUT FÜR NORMUNG E.V.:, vgl. S. 6
[5] Andreas

durchgeführt werden. Sie dienen zur Ausbesserung und Austausch von nicht ordnungsgemäßen Zuständen. Hierbei wird der Ist-Zustand zurück in den Soll-Zustand versetzt. Stetige Verbesserungen können die Produktivität erhöhen und einen Vorteil verschaffen. Die Umfänge der drei anderen Bereiche werden hierdurch verringert bzw. vereinfacht. Durch Auswertungen und Analysen aus verschiedenen Prüfungen können Entscheidungen getroffen werden, um die Ziele zu erreichen.

Begriff	**Definition**
Wartung	Maßnahmen zur Verzögerung des Abbaus des vorhandenen Abnutzungsvorrats
Inspektion	Prüfung auf Konformität der maßgeblichen Merkmale eines Objektes durch Messung, Beobachtung oder Funktionsprüfung
Instandsetzung	Physische Maßnahme, die aufgeführt wird, um die Funktion eines fehlerhaften Objektes wiederherzustellen
Objekt	Teil, Bauelement, Gerät, Teilsystem, Funktionseinheit, Betriebsmittel oder Systems, das / die für sich allein beschrieben und betrachtet werden kann
Schwachstelle	Objekt, bei dem ein Ausfall häufiger als es der geforderten Verfügbarkeit entspricht, eintritt und bei der eine Verbesserung möglich und wirtschaftlich vertretbar ist
Abnutzung	Abbau des Abnutzungsvorrates, hervorgerufen durch chemische und/ oder physikalische Vorgänge
Nutzung	Bestimmungsgemäße Verwendung unter Berücksichtigung der allgemeinen anerkannten Regeln der Technik eines Objektes
Ausfall	Verlust der Fähigkeit eines Objekts, eine geforderte Funktion zu erfüllen

Tab. 2.1: Begriffe für die Grundlagen der Instandhaltung nach DIN 31051:2019-06

Zuvor lag der Fokus der Instandhaltung auf einer reaktiven Strategie. Analgen wurden so lange verwendet bis diese Funktionsunfähig und zu einen Ausfall geführt haben. Die Arbeiten waren ungeplant und nicht voraussehbar. Präventionsarbeiten wurden eingeführt um die Wirtschaftlichkeit zu erhöhen und den Ausfall zu verringern. Die Anlagen wurden gewartet wonach die sogenannte Zustandsorientierte Wartung folgte. Heutzutage werden präventive Maßnahmen in Form der vorausschauenden Instandhaltung getätigt. Fehler können erfasst und ihrem Ursprung zugeordnet werden. Diverse Komponenten einer Anlage können somit frühzeitig repariert oder ausgetauscht werden, bevor es zu einem Ausfall kommt. Hieraus ergeben sich verschiedene Instandhaltungsstrategien.[6]

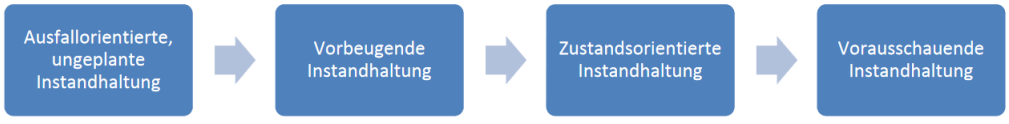


Abb. 2.2: Entwicklung der Instandhaltung - Quelle: Bauer und Dremel

[6]Bauer und Dremel, vgl. S. 6 ff.

2.2 Instandhaltungsstrategien

Wie zuvor beschrieben lassen sich die Methoden zu verschiedenen Instandhaltungsstrategien zuordnen. Die zwei Hauptstrategien sind „reaktiv“ und „proaktiv“.

Die reaktive Strategie wird verflogt, sofern der Ausfall einer Anlage als nicht kritisch eingestuft wird und die Ersatzteilverfügbarkeit hoch ist. Bis zum Ausfall entstehen keine Kosten im Sinne der Instandhaltung und keine Maßnahmen werden ergriffen. Der Zeitpunkt ist nicht voraussehbar und ungeplant. Des Weiteren ist wichtig, dass keine folge Schäden für Umwelt, Personen und Image entstehen.[7]

Die proaktive Strategie, also vorausschauend und geplant, lässt sich weiter in zwei verschiedene Unterstrategien unterteilen. Der präventiven und voraussagenden Instandhaltung. Bei beiden Strategien wird im Gegensatz zur reaktiven versucht ein Defekt vorzubeugen, sodass Ausfälle vermieden werden können.

Bei der präventiven Strategie werden in den meisten Fällen Maßnahmen wie die Inspektion und Wartung durchgeführt. Sie finden auf der Grundlage von Vorwissen statt, wie zum Beispiel die Lebensdauer eines Bauteils, Betriebsstunden, Kilometer und Anzahl der Starts oder produzierten Produkte. Die Wartungsintervalle werden periodisch in festen Intervallen durchgeführt. Hierbei kann es passieren, dass diese zu früh oder zu spät geplant werden, sodass höhere Kosten entstehen. Anderseits können die Wartungen auf Zeitpunkte gelegt werden, bei denen keine Produktion stattfindet. Durchführungen werden somit Kostengünstiger.[8]

Im Zuge der Industrie 4.0 wird immer mehr von festen Wartungsintervallen oder Inspektionen abgewichen. Viel mehr wird mit Hilfe von Sensoren eine ständige Zustandsüberwachung von Anlagen zu ermöglichen. Das sogenannte „Condition Monitoring“ spielt in diesem Zusammenhang ein große Rolle.

Hiermit ist die voraussagende Strategie der Instandhaltung gemeint. Maßnahmen sollen optimal je nach Abnutzungsgrad von Maschine und Komponenten dynamisch geplant werden. Hierzu können Messwerte wie Stromaufnahme, Schwingungen (zum Beispiel Vibrationen) und chemische Größen bzw. Indikatoren verwendet werden. Möglichkeiten zur Erfassung sind in der Messtechnik zu finden.[9]

Abweichungen vom normalen Betriebsverhalten können somit Aufschluss über den Zustand geben und nötige Maßnahmen abgeleitet werden. Im Vergleich zur präventiven Strategie ist der Zustand jederzeit und nicht nur zu bestimmten Zeitpunkten bekannt. Die initialen Kosten sind dementsprechenden sehr hoch und der Einsatz muss wirtschaftlich gerechtfertigt sein.

In der Realität sind die meisten Instandhaltungskonzepte nicht einer Strategie zuzu-

[7] Vollmüller
[8] Bauer und Dremel, vgl. S. 7
[9] Andreas

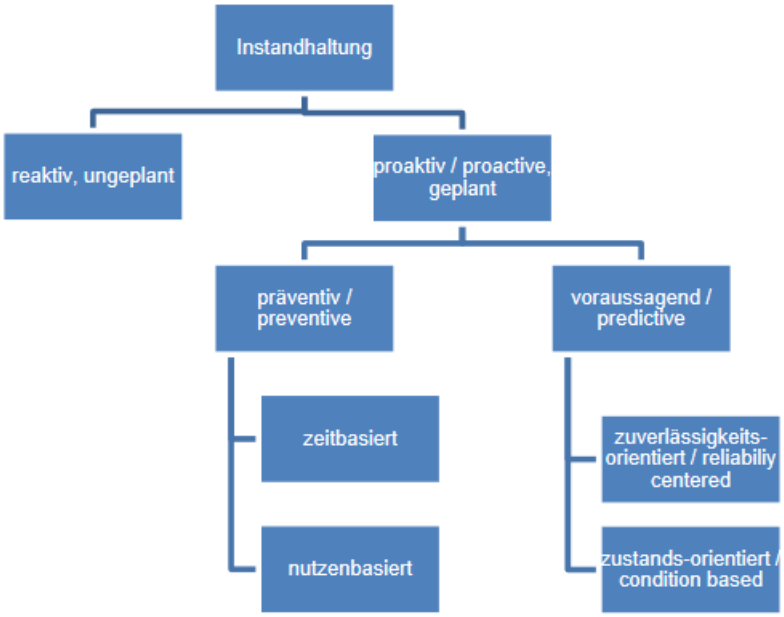


Abb. 2.3: Strategien der Instandhaltung - Quelle: Bauer und Dremel

ordnen. Begriffe sind nicht eindeutig definiert und geben somit viel Interpretationsspielraum. Des Weiteren wird Anhand von Faktoren wie „die Verkettung und Redundanz der Anlagen, Gültigkeit von Qualitäts-, Umwelt- und Sicherheitsnormen, Arbeitszeitvereinbarungen, Instandsetzungszeit, Ersatzteilverfügbarkeit, Materialpuffer zwischen den Anlagen, Belastungsspitzen durch saisonal bedingten Materialbedarf und Rohstoffverfügbarkeit etc.“[10] komplexe Analysen durchgeführt um für jeden Teilaspekt eine eigene Entscheidung bzw. vorgehen auszuwerten. Hierbei stehen der Nutzen und die Kosten für die Umsetzung im Vordergrund.

2.3 Condition Monitoring

Das Condition Monitoring (CM) (deutsch *Zustandsüberwachung*) sind „manuell oder automatisch ausgeführte Tätigkeit zur Messung der Merkmale und Parameter des Ist-Zustands einer Einheit in bestimmten Zeitabständen“[11]. Aus den Tätigkeiten lassen sich im Anschluss die vorherigen Kapitel beschrieben Maßnahmen ableiten. Unter anderem bedeutet dies, dass sich das Condition Monitoring auf die Erfassung und Auswertung der Anlage beschränkt. Die Umsetzung kann verschieden Formen annehmen, die im Folgenden weiter erläutert werden. Grundsätzlich beruht das System darauf, dass sich Fehler und Störungen über einen Zeitraum hinweg entwickeln und nicht schlagartig auftreten.

Im normalen Betriebszustand einer Anlage werden kontinuierlich oder in Zeitintervallen verschiedene Messwerte erhoben. Diese stammen von bereits verbauten Sensoren in der

[10] Bauer und Dremel, S. 8
[11] DEUTSCHES INSTITUT FÜR NORMUNG E.V.:, vgl. S. 26

Anlage, speziell für das Condition Monitoring verwendete Sensoren und Produktionsdaten bzw. Prozessdaten aus übergeordneten Systemen wie zum Beispiel das „Manufacturing Execution System“ (MES). Grundlage sind hierfür erfassbare Werte, die im Verlauf der Zeit mit den Soll-Parametern verglichen werden.

Anlagen mit hoher Komplexität und einer hohen Ausfallwahrscheinlichkeit eignen sich Besonders für das Condition Monitoring. Sie liefern viele Werte, die sich verändern, um mögliche Fehler frühzeitig zu erkennen.

Die Auswertung erfolgt Softwaregestützt, entweder Online oder Offline. Bei der Offline-Auswertung werden alle Informationen auf einem Datenträger gespeichert der im Anschluss manuell in das Auswertungssystem eingespielt wird. Dies hat den Vorteil, dass diese nicht direkt im Produktionsnetzwerk hängt, jedoch gleichzeitig einen Mehraufwand in den Analysetätigkeiten erfordert. Beim Online-System gibt weitere Möglichkeit. Die Datenerfassung und Auswertung kann kontinuierlich direkt in der Anlage passieren oder gespeichert und in bestimmten Zeitintervallen erfolgen. Das System kann auch als sogenannter „Remote Service“ implementiert werden.

Im Sinne von Industrie 4.0 sind mit Hilfe von Künstlicher Intelligenz komplett automatisierte Prozesse denkbar. Anlagen sollen selber Entscheidungen über mögliche Maßnahmen treffen und an Experten übermitteln, die ein Fehlerbild beheben können.[12]

2.3.1 Condition Monitoring System

Um eine komplette Zustandsüberwachung zu realisieren sind verschiedene Komponenten notwendig. Zusammen ergibt sich ein „Condition Monitoring System“ (CMS). Die einzelnen Teilaspekte werden im Folgenden Betrachtet.

Hardware sind alle Objekte die zur Berechnung oder Aufnahme von Werten zur Verfügung stehen bzw. benötigt werden. Beispiele hierfür sind Sensoren oder Server.

Software dient zur Speicherung und Strukturierung aller erhoben Daten. Mit ihr wird die Logik des Condition Monitoring realisiert. Sie ist für die Archivierung, Visualisierung und Auswertung zuständig. Des Weiteren werden Alarme über mögliche Fehler generiert und weitergeleitet. Einzelne Schritte werden voll automatisch oder unterstützend ausgeführt.

Personal wird benötigt um ein CMS zu implementieren und konfigurieren. Jedes System muss speziell von Experten auf die Umstände und Anforderungen der Anlage und des Kunden angepasst und konzipiert werden. Auch im Anschluss ist das Personal ein

[12] Bauer und Dremel, vgl. S. 9

kritischer Faktor. Analysen, Anpassungen und auch Wartungen an dem Instandhaltungssystem selbst müssen durchgeführt werden, um ein einen Reibungslosen Gesamtablauf zu realisieren.

Remote Services und Plattform ermöglichen die Kontrolle und Überwachung aus der Ferne. Eine Verlagerung in Cloud-Dienste und Off-Site-Lösungen sind auch für physikalische Systeme attraktiv. Besonders in Bezug auf Fachkräfte ermöglicht eine Cloud-Lösung hohe Flexibilität und schafft ein gutes Angebot. Dieses kann über das Internet Probleme Analysieren und Anpassungen vornehmen.

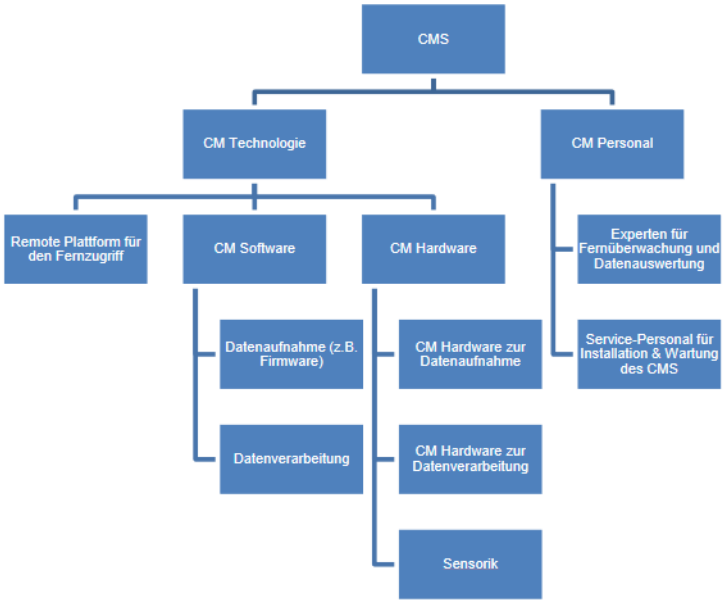


Abb. 2.4: Condition Monitoring System - Quelle: Bauer und Dremel

2.3.2 Vor- und Nachteile des Condition Monitoring

Condition Monitoring ist ein mittlerweile weit verbreitetes Konzept in diversen Industrien. Es verfügt über großes Potenzial im Hinblick auf Kosten, Sicherheit und Betrieb.

Die Installationskosten sind in den meisten Applikationen sehr hoch. Viele und teilweise Spezielle Sensoren sind Notwendig, um aussagekräftige Systeme zu implementieren. Andere Strategien könnten die bessere und kostengünstigere Wahl sein. Die Kosten bleiben auch nach der Installation hoch, da geschultes Personal benötigt wird. Auswertungen, Wartungen sowie Anpassungen sind im Vergleich teuer.

Wird das Condition Monitoring jedoch richtig angewendet amortisieren sich die Kosten mit besonderem Augenmerk auf ungeplante Wartungen und Ausfälle. Produktionsverzögerungen und Imageschäden können durch eine geringe Downtime verhindert werden. Ersatzteile müssen nicht für Notfälle auf Lager sein, sondern können bestellt werden,

wenn ein Fehlerbild anfängt aufzutreten. Der Austausch erfolgt dann in den regulären Wartungsintervallen ohne jeglichen Ausfall. Dies reduziert unter anderem auch die Lagerkosten und das Kapital ist nicht gebunden. Gut funktionierende Systeme sorgen außerdem für Sicherheit. Das Risiko für Mitarbeiter sich an fehlerhaften Komponenten und Anlagen zu verletzten sinkt.[13]

[13] WhatConditionMonitoring

3 Fallbeispiel Förderbandmotor

Das Transfersystem im PTZ wird durch vier Drehstrommotoren angetrieben. Diese bewegen über ihr Getriebe jeweils einen der vier Förderband-Teile. Ziel des Condition Monitorings ist es in diesem Anwendungsfall den Ausfall der Motoren zu verhindern. Aufgrund geringer Pufferkapazitäten innerhalb der Produktionslinie hätte ein Ausfall eines der Förderbänder schnell einen Produktionsstillstand der gesamten Fertigungsstraße im PTZ zur Folge. Außerdem kann der Umbau nicht vollzogen werden, solange sich Ladungsträger auf dem jeweiligen Förderband befinden. Somit müssten die entsprechenden Ladungsträger manuell entnommen und im späteren Produktionsverlauf wieder eingeschleust werden. Sollte bei erhöhter Varianz auf eine Just-In-Sequence Materialbereitstellung umgestellt werden, wird ein solches Ereignis weitere erhebliche Probleme mit sich führen. Daher gilt es unbedingt einen Ausfall zu verhindern. Mit einem funktionierenden Condition Monitoring System kann ein Austausch geplant werden. So wird das entsprechende Förderband gegen Ende der Produktion leer gefahren und der Motor wird nach Produktionsende ausgetauscht. Bei Wiederaufnahme der Produktion am Folgetag oder in der Folgewoche treten lediglich minimale Stillstandszeiten der nachfolgenden Stationen auf.

3.1 Aufbau des Motors

„Bei größerem Leistungsbedarf, z.B. in der Industrie, bei der Produktion oder in elektrischen Maschinen, werden Elektromotoren nicht mit Wechselspannung, sondern mit Dreiphasen Wechselstrom oder auch „Drehstrom“ betrieben. In der Praxis haben sich Kurzschlussläufer bestens bewährt, da sie so gut wie keinerlei Verschleißteile aufweisen und extrem wartungsarm sind.“ Auch die Motoren der Produktionslinie im PTZ benutzen diese Art von Motoren.

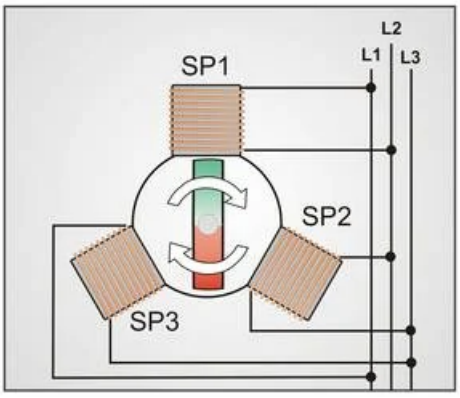


Abb. 3.1: Aufbau eines Motors

„Werden drei Drahtspulen (SP1 bis SP3) dreiecksförmig angeordnet und mit den drei Phasen verbunden, bauen die Spulen ein Magnetfeld auf, das sich im Rhythmus der Netz-

frequenz „dreht“.

Nun muss nur noch ein magnetischer Rotor in das Zentrum der drei Spulen montiert werden, der vom drehenden Magnetfeld „mitgenommen“ wird. Demzufolge kommt ein Drehstrommotor ohne Kommutator aus, was den Aufbau deutlich vereinfacht.

Der Rotor besteht aus einer Welle mit kreisrunden und untereinander isolierten Eisenlamellen. In den Rotor sind mehrere massive Metallstäbe eingebettet, die als Stromleiter dienen. Auf beiden Seiten des Rotors sind die Metallstäbe mit jeweils einer Metallplatte leitend verbunden (kurzgeschlossen). Der dadurch entstandene Leiterkäfig sorgte für die Bezeichnung Käfigläufer oder auch Kurzschlussläufer.

Das Magnetfeld der Statorspulen induziert in den Leitern des Rotorkäfigs einen Strom, der seinerseits ebenfalls ein Magnetfeld erzeugt. Die gegenseitige Beeinflussung der Magnetfelder sorgt dafür, dass der Rotor in eine Drehbewegung versetzt wird.

Die Drehrichtung ist von der Reihenfolge der Phasenverschiebung auf den Anschlussleitungen abhängig und kann durch das Vertauschen von zwei der drei Anschlussleitungen geändert werden.“[14]

Da in diesen Motoren wenige elektronische Bauteile verbaut sind, sind die auftretenden Fehler überwiegend mechanischer Natur. Da man gerade diese Fehler anhand verschiedener Indikatoren vorhersehen kann, werden im Rahmen des Condition Monitoring die Gruppe der mechanischen und die der prozessbedingten Fehler betrachtet. Mechanische Fehler sind meist auf beschädigte Teile des Motors zurückzuführen. Dazu zählen beispielsweise lose Komponenten am Motor oder Spiel in der Befestigung des Motors. So kann der Motor das Band nicht mehr optimal antreiben oder streift an anderen Bauteilen des Förderbands, was zu starken Abnutzungen führen kann. Eine solche ungewollte Fehlausrichtung kann den Verschleiß ungemein erhöhen. Auch eine Blockade im Förderband oder zu starke Verschmutzungen in den Laufschienen kann zu einer erhöhten Motorlast und somit zu erhöhter Leistungsaufnahme und Verschleiß führen. Bekommt der Motor eine mechanische Unwucht, die unentdeckt bleibt, wird sie sich mit der Zeit verschlimmern. Eine weitere Gefahr entsteht durch einen Lagerschaden, der den Motor oftmals für eine gewisse Zeit zum Stillstand zwingt oder je nach Modell einen kostspieligen Austausch des gesamten Motors zur Folge hat.

Auf der Seite der Prozessbedingten Fehlern besteht viel oftmals ungenutztes Potential, dass mit Hilfe von Condition Monitoring genutzt werden kann. Bei falscher Ansteuerung und ungünstigen Einsatzbedingungen arbeitet ein Elektromotor mit einem schlechten Wirkungsgrad, was eine Energieeffiziente Produktion nicht ermöglicht. Auch ein über- oder unterdimensionierter Motor kann nicht besonders effizient eingesetzt werden. Wenn man die Motortemperatur überwacht, findet man Leistungsverluste durch Wärme. Diese schaden bei übermäßiger Beaufschlagung dem Motor und zeugen von einer hohen Verlust-

[14] ElektromotorenAufbauFunktionsweise

leistung, also einem schlechten Wirkungsgrad oder einer schlechten, nicht ausreichenden Kühlung. Sind Motor und Getriebe ungenügend geschmiert oder übermäßig verschmutzt, kann dies den Verschleiß drastisch erhöhen und zu ernsten Schäden führen.

Alle Anzeichen dieser mechanischen und prozessbedingten Probleme und Fehler, gilt es frühzeitig zu erkennen und richtig zu deuten. Nur so können Folgeschäden vermieden und die Produktion möglichst effizient gestaltet werden. Aus diesem Grund müssen die richtigen Indikatoren gewählt werden.

3.2 Indikatoren

Grundsätzlich stehen bei einem Drehstrommotor mehrere geeignete Indikatoren zur Verfügung.

Die meisten Fehler kann man eigentlich durch die Betrachtung der Stromaufnahme erkennen. „Dieser modellbasierte Ansatz funktioniert nach dem Prinzip, dass der von einem Elektromotor aufgenommene Strom nicht nur durch die angelegte Spannung, sondern auch durch das Verhalten sowohl des Motors als auch des angetriebenen Geräts beeinflusst wird. Dadurch werden die Verzerrungen der Stromverläufe identifiziert, die nicht durch Verfälschungen des Spannungsverlaufs verursacht wurden und daher durch das Verhalten des Systems aus Motor und angetriebenen Geräten verursacht worden sein müssen. Die Frequenz dieser Verfälschungen gibt die Art der Ursache an, und das Ausmaß der Verfälschungen gibt den Schweregrad der Ursache an.“[15]

Allerdings fordert diese Lösung ein hohes Level an Input vergangener Daten, um für einige der genannten Probleme aussagekräftige Ergebnisse treffen zu können und verlässliche Prognosen aufzustellen. Da in unserem Projekt keine historischen Daten vorliegen und aufgrund kurzer Testläufe und lediglich Einsatzzeiten zu Demonstrationszwecken auch keine verlässlichen Datensätze in absehbarer Zeit entstehen werden, kann dieser Ansatz nur teilweise genutzt werden. Einige Fehler kann man anhand der Motorstromanalyse und der daraus resultierenden Leistungsaufnahme allerdings erkennen. Dazu zählen beispielsweise Prozessfehler wie eine Über- oder Unterdimensionierung, oder mangelnde Schmierung / erhöhte Verschmutzung. Auch mechanische Probleme wie Bauteildefekte oder Anstreifen führen zu einer stark erhöhten Motorlast und somit zu einer erhöhten Leistungsaufnahme. All diese Fehler können mit Hilfe der Motorstromanalyse erkannt werden.

Vor allem bei mechanischen Problemen oder Beschädigungen ist eine Überwachung der Vibration am Motor ein geeigneter Indikator. Anhand erhöhter Vibrationen können Beschädigungen einfach erkannt werden. Bereits eine geringe Unwucht erzeugt bei hohen Umdrehungen deutlich messbare Schwingungen des Motors. Auch Beschädigungen führen oftmals zu Vibrationen. Anstreifen oder Fehlausrichtung können erkannt werden, da hierdurch oftmals eine Unwucht auftritt, die durch die Vibrationssensoren detektiert

[15] EMCMStationareMotorstromanalyse

wird. Die am eindeutigsten durch die Vibrationsanalyse erkennbaren Fehlerarten sind Lagerschäden, eine lose Aufhängung des Motors, sowie lockere Bauteile. Diese Fehlerarten führen unmittelbar zu einem starken, unüblichen Auftreten von Vibrationen, die gemessen werden können und einen Fehler Indizieren.

Ein dritter Indikator, der bei Elektromotoren zum Einsatz kommt, ist die Temperatur. Anhand der Temperatur können vor allem zu hohe Lasten und zu viel Reibung erkannt werden. Eine Unterdimensionierung resultiert meist in einer zu hohen Last und in einer für den Motor grenzwertig hohe Leistungsaufnahme. Da die thermischen Verluste ebenfalls steigen, kann dieses Problem durch Temperatursensoren detektiert werden. Auch bei einem nicht unmittelbar erkennbaren Schaden, der erhöhte Reibung im Motorinneren hervorruft, wie beispielsweise ein Lagerschaden, steigt der thermische Verlust durch die erhöhte Reibung an und die Temperatur des Motors an der entsprechenden Stelle steigt an. Temperatursensoren sind günstig und leicht einzusetzen, decken aber nicht besonders viele auftretende Probleme ab. Daher ist die Temperatur als Indikator zwar bedingt geeignet und kann ohne große Unkosten als zusätzlicher Indikator eingesetzt werden, ist aber als alleiniges Merkmal nicht das am besten geeignete. Allerdings bringt dieser Indikator bei der genauen Fehlerbestimmung einen guten Anhaltspunkt.

Die einzelnen Indikatoren sind ohne eine große Anzahl an vorliegenden Daten nicht besonders spezifisch, wenn es darum geht, die genaue Fehlerart zu bestimmen. Beim Einsatz mehrerer Indikatoren wird diese Eingrenzung einfacher. Bei einem Lagerschaden beispielsweise können starke Vibrationen gemessen werden, während aufgrund der auftretenden Reibung auch die Leistungsaufnahme, sowie die Temperatur steigen. Im Gegensatz dazu werden sich bei einem losen Bauteil Temperatur und Leistungsaufnahme wenig bis gar nicht verändern, während deutliche Vibrationen messbar sind. Mit Hilfe dieser Vergleiche kann der auftretende Fehlerf all besser analysiert und zumindest eingegrenzt werden. Oftmals kann so sofort eine aussagekräftige Prognose getroffen werden.

3.3 System und Hardware

3.3.1 Motorstromanalyse

Für eine Motorstromanalyse ist nicht besonders viel Hardware nötig. Es genügen drei einfache Spannungswandler und Abgriffe der anliegenden Spannungen, wie in Abb. 3.2 dargestellt.

Zu Beginn werden Referenzwerte auf Basis von historischen Daten oder Produktionstests vorgegeben. Die ständigen Messwerte werden im laufenden Betrieb mit diesen Referenzwerten verglichen. So können Abweichungen und Anomalien schnell erkannt werden. Je nach Schwere und Art der Abweichung kann auf eine bestimmte Art von Fehler geschlossen werden. Aufschlussreich ist auch die Art des Auftretens, also ob ein Wert

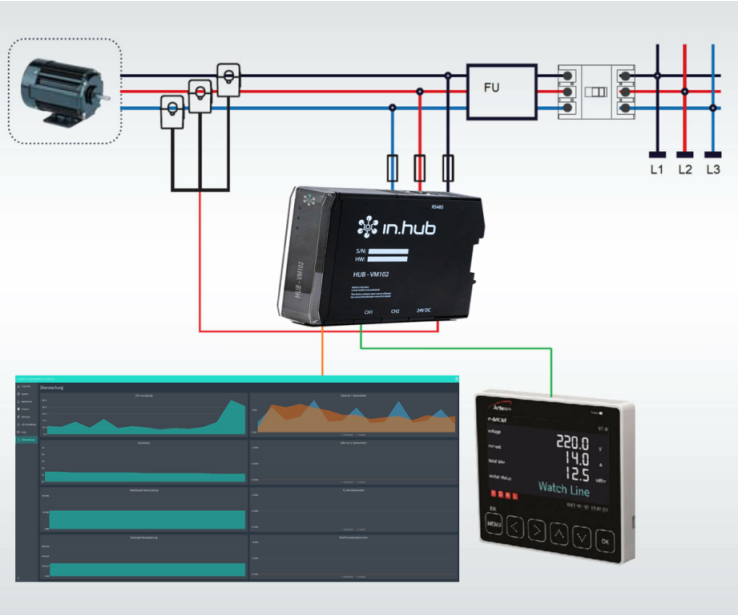


Abb. 3.2: Messaufbau einer Motorstromanalyse

sich langsam, beispielsweise bedingt durch Abnutzung, seinem definierten Grenzwert annähert, oder diesen plötzlich sprunghaft überschreitet. Dies würde eher auf einen Defekt, beispielsweise mechanischer Natur hindeuten. Wenn man die normalen Parameter des Motors kennt, kann man den Strom nicht nur messen, sondern auch berechnen. Treten bei richtiger Berechnung Differenzen zwischen berechneter und gemessener Stromstärke auf, dann kommt das vom Verhalten des Motors oder Förderbands. Die Höhe der Abweichungen lässt auf das Ausmaß des Problems schließen, die Frequenz der Verfälschungen auf die Ursache. Durch Betrachten von Leistung, Phasenwinkel und Gesamtoberwellenverzerrung können noch deutlich präzisere Aussagen getroffen werden, allerdings wird hierfür eine Menge an qualitativ hochwertigen Daten benötigt, um das System angemessen zu initialisieren.

3.3.2 Vibrationsanalyse

Bei der Vibrationsanalyse werden mithilfe von Beschleunigungssensoren Schwingungen des Motors aufgenommen. Um das Gesamtsystem optimal zu erfassen, sollten Messungen in horizontaler, vertikaler und axialer Richtung beider Motorlager vorgenommen werden. Abb. 3.3 zeigt Beispiele für geeignete Positionierung der Beschleunigungssensoren.

Grundlegend funktioniert die Erfassung der Messdaten wie folgt: Mithilfe der Messwandlers (Beschleunigungssensor) wird die Bewegung des schwingenden Körpers in ein elektrisches Signal umgewandelt. Im Allgemeinen ist ein Messwandler ein Gerät, das mechanische Größen wie Verdrängung, Geschwindigkeit, Geschwindigkeitszunahme oder Leistung in elektrische Größen (wie Spannung oder Strom) umwandelt. Da das resul-

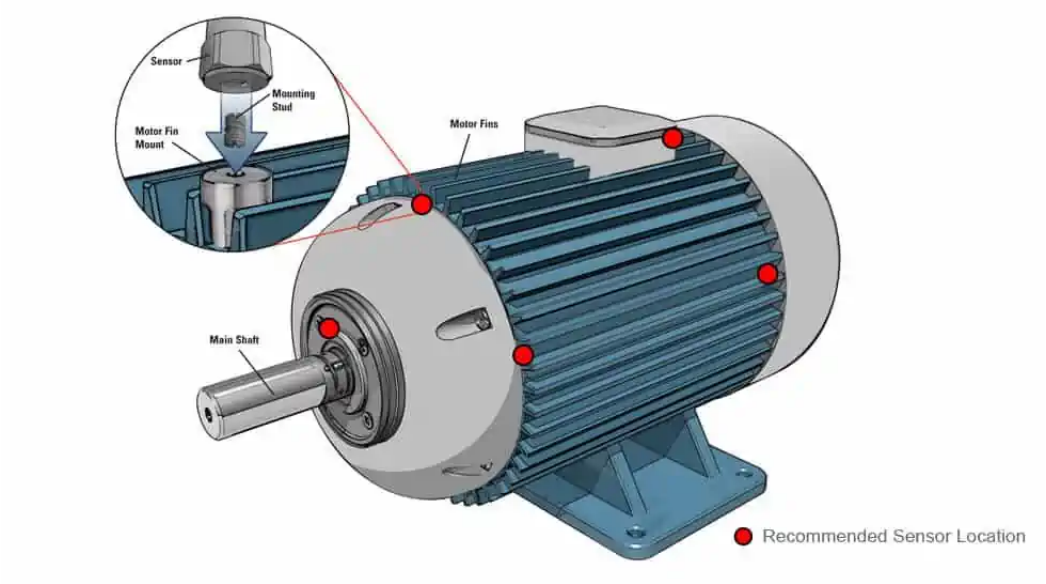


Abb. 3.3: Vibration Messung an Motor

tierende Signal, z. B. Spannung oder Strom eines Messwandlers, zu klein ist, um genau aufgezeichnet und erfasst zu werden, wird ein Messverstärker eingesetzt, um das Signal auf den erforderlichen Wert zu verstärken. Das Ergebnis aus dem Messverstärker kann zur visuellen Überprüfung angezeigt, von einer Aufzeichnungseinheit aufgezeichnet oder in einer Datenbank für einen späteren Zeitpunkt gespeichert werden. Die Informationen können dann ausgewertet werden, um die idealen Schwingungseigenschaften der Maschine oder des Designs zu bestimmen. Abhängig von der geschätzten Größe wird als Schwingungsmessgerät ein Vibrometer, Geschwindigkeitsmesser, Beschleunigungsmesser, ein Stufenmesser oder ein Wiederholungszähler eingesetzt. In manchen Anwendungen möchte man eine Maschine oder Konstruktion in Schwingung versetzen, um Nachhalleigenschaften aufzuspüren. Hierfür können elektrodynamische Vibratoren, elektrohydraulische Vibratoren und Vorzeichengeneratoren (Oszillatoren) verwendet werden.[16]

3.3.3 Temperaturanalyse

Die Temperatur am Motor kann auf verschiedene Arten gemessen werden. Es können beispielsweise Laser- oder Infrarotmessungen vorgenommen werden, um die Temperaturen zu bestimmen. Sehr häufig werden aber Temperaturfühler, wie z.B. ein PT100 verwendet.

„Der PT100 Thermistor ist ein variabler elektrischer Widerstand mit einer fast linearen Widerstandsgeraden (es wird nur der lineare Bereich verwendet), der seinen Wert bei Temperaturänderung verändert. Über die Veränderung des Thermistorwiderstandes kann eine Strom- oder Spannungsänderung ermittelt werden. Anhand dieser Änderung kann über eine Widerstandskurve die Temperatur interpretiert und ermittelt werden. Heute

[16] of Management, vgl. S. 40 f.

geschieht der ganze Prozess in Schaltungen oder Programmen vollkommen automatisch. Unterteilt werden Thermistoren in Heiß- und Kaltleiter, sogenannte NTC (PT100)- und PTC-Thermistoren.“[17]

Im Regelfall wird die äußere Temperatur des Motors bestimmt. Ist ein Getriebe vorhanden und im Falle eines Ölgeschmierten Getriebes das Getriebeöl zugänglich, kann auch hier die Gehäuse- sowie die Öltemperatur gemessen werden. Optimale Punkte für die äußere Temperaturmessung sind die Lager. Gerade bei Beschädigungen oder mangelnder Schmierung erhöht sich an diesen Stellen durch auftretende Reibung schnell die Temperatur. Daher erhitzt sich auch das Gehäuse an den entsprechenden Stellen. So kann schleichender thermischer Verschleiß frühzeitig erkannt werden. Dies ist notwendig, um Gegenmaßnahmen rechtzeitig einzuleiten und einen Motortausch zu verzögern oder zu verhindern.

Möchte man auf die Belastung des Motors schließen, so sollte die Wicklungstemperatur bestimmt werden. Diese gibt Aufschluss darüber, wie die Strommenge den Motor belastet und ob ein guter oder schlechter Wirkungsgrad vorliegt. Die Wicklungstemperatur kann allerdings beim nachgerüsteten Condition Monitoring nicht ohne weiteres gemessen werden. Der Elektromotor kann in den wenigsten Fällen einfach geöffnet und mit Temperatursensoren im Inneren erweitert werden. Daher muss auch hier an geeigneten Stellen die Gehäusetemperatur gemessen werden. Mit Hilfe von Modellberechnungen und Messwerten, werden Berechnungsgrundlagen erstellt. Durch diese Berechnungen lässt sich aus den Messwerten an verschiedenen Stellen des Gehäuses auf die Wicklungstemperatur schließen.

3.3.4 Filtern und Grenzwerte

Beim Implementieren eines Condition Monitoring Systems ist es sehr wichtig, alle Parameter richtig einzustellen. Grundsätzlich sollten zur Verhinderung von Fehlmessungen und damit zusammenhängenden falschen Warnmeldungen und Handlungsempfehlungen Filter für die Messwerte implementiert werden. Hierfür sollte ein Bereich bestimmt werden, in dem alle plausiblen Werte eines Sensors liegen. Zu beachten gilt es, dass die Definition eines plausiblen Wertes sich hier sowohl auf die Werte des Normalbetriebs als auch auf Wertebereiche, die im Fehlerfall auftreten bezieht. Auf keinen Fall dürfen Werte, die auf einen Defekt hinweisen durch einen Filter entfernt werden. Dennoch sollten alle nicht plausiblen Werte ausgeblendet werden. Dies führt zu weniger Störmeldungen und lässt auf einen Sensordefekt schließen, wenn wenige oder keine Werte mehr ankommen.

Abb. 3.4 beschreibt den Gesamtprozess. In unserem Fallbeispiel werden Sensoren für Temperatur, Vibration, Strom und Spannung aus den in den vorherigen Kapiteln genannten Gründen verwendet. Nun beginnen die Referenzwertfestlegung und die Zustandsde-

[17] PT100ErklarungUnd

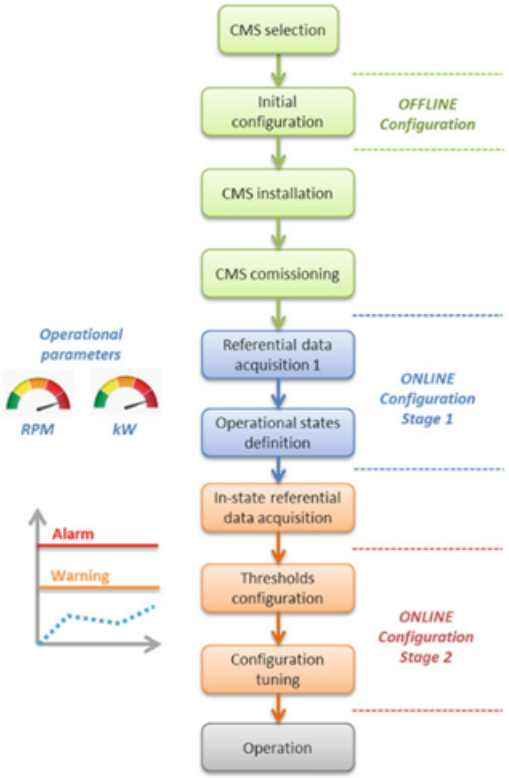


Abb. 3.4: Condition Monitoring Algorithms in MATLAB - Quelle: Adam Jablonski

finition. Hierfür muss entweder auf bestehende Daten zurückgegriffen werden oder Probedaten aus Testläufen hinzugezogen werden. Da sich in unserem Fall, das Fördersystem im Umbau befindet und keinerlei historische Daten verfügbar sind, müssen Testläufe für die Datenerhebung genutzt werden. Mit diesen Daten können nun alle Grenzwerte gesetzt werden. Die Schwellen sollten in eine Warn- und eine Alarmschwelle unterteilt werden. Während die erste Schwelle nur einen Check zum nächstmöglichen Zeitpunkt nach sich zieht, definiert die Alarmschwelle klare Handlungsanweisungen und erfordert ein sofortiges Eingreifen.

Anhand mehrerer verfügbaren Motoren können verschiedene Verschleißzustände gemessen oder mit manuellen Eingriffen auch simuliert werden. Damit kann bereits mit einer geringen Menge an Daten ein guter Trend der relevanten Parameter bei Verschleiß erkannt werden. Die Parameter zur Berechnung nicht direkt gemessener Daten, müssen in Modellen abgebildet und errechnet werden. Diese Modelle dürften sich als sehr aufwändig erweisen, da der Hersteller sie bereits länger nicht mehr vertreibt und auch kein virtuelles Abbild zur Verfügung steht. Das Erstellen eines solchen Abbilds wäre im Rahmen einer solchen Studienarbeit mit zu viel Aufwand verbunden, weshalb auf aufgenommene Messwerte, sowie angenommene Werte zurückgegriffen werden muss.

Sind die Schwellen definiert, muss eine Reihe an prozessnahen oder prozessgetreuen Testläufen durchgeführt werden. Diese geben Aufschluss darüber, ob die Konfigurierung

im Normalbetrieb keine Probleme erkennt oder Fehler meldet. Im zweiten Schritt werden alle Fehlerquellen, die simuliert werden können (meist mechanische Probleme an Motor oder Fördersystem) simuliert. Hierbei wird geprüft, ob Fehler zuverlässig erkannt und richtig gedeutet werden. Im Anschluss wird das Condition Monitoring System freigegeben und kann in der Produktion eingesetzt werden.

4 Einleitung zur Implementierung des Condition Monitoring Systems

Das nachfolgende Kapitel beschreibt den gewünschten Zielerreichungsgrad bezüglich der Implementierung des Zustandsüberwachungssystems. Des Weiteren wird das zentrale Gateway vorgestellt, welches für die Umsetzung der Aufgabe ausgewählt wurde. Abschließend werden die Erstinbetriebnahme und die Einrichtung der wichtigsten Software-Werkzeuge dargelegt.

Das Inhub GM-100 „Internet of Things“ (IoT) Gateway dient im Folgenden als Basis für die Datenerfassung und -verarbeitung. Neben der Datenspeicherung und Weitergabe der Informationen über das IoT-Gateway, können Daten für Analysezwecke archiviert und für externe Projekterweiterungen bereitgestellt werden. Somit stellt das GM-100 den zentralen Mittelpunkt des „Condition Monitoring“ (CM) Systems dar.

4.1 Zielsetzung bei der Implementierung des Condition Monitoring Systems

Dieses Projekt zielt in seiner Gesamtheit darauf ab ein Zustandsüberwachungssystem zu implementieren. Dabei soll das vorgegebene IoT-Gateway verwendet werden.

Im Zuge des „Retrofittings“ von Teilen der Anlage, wie bspw. der Bandmotoren, sollen ausgewählte Sensoren so eingesetzt werden, dass mit Hilfe einer aufgenommenen Datenbasis, eine effektive Aussage über den Zustand der vermessenen Bauteile getroffen werden kann.

Letztlich soll das Gateway als eine Schnittstelle zu weiteren Systemen in Form eines „Enterprise Resource Planning“ (ERP) Systems oder „Manufacturing Execution System“ (MES) dienen.

Aufgrund der noch nicht stattgefundenen Inbetriebnahme des Produktionssystems, rücken in dieser ersten Arbeit, die softwareseitigen Faktoren in den Vordergrund. Dazu zählen neben der Ersteinrichtung des Gateways, die Auswahl, Installation und Konfiguration aller benötigten Software-Werkzeuge. Neben einem MQTT-Netzwerk das errichtet wird, findet eine Einführung in die Datenaufnahme mittels Zeitreihendatenbank statt. Zudem werden Programme zur Analyse der Daten eingerichtet und die Schnittstellen eines Python Skripts implementiert. Ein solches soll zu einem späteren Zeitpunkt des Projekts ebenfalls als auswertender Algorithmus dienen.

4.2 IO-Situation, Protokollfähigkeiten und technische Daten des GM-100

Das zugrunde liegende Gateway verfügt über mehrere Ein- und Ausgangsmöglichkeiten.

Dazu zählen die Eingangspins der Stromversorgung, an welche eine in der „Industrial Internet of Things“ (IIoT) Welt übliche Gleichspannung von 24 Volt angelegt werden kann, um das Gerät entsprechend betreiben zu können. Das versorgende Element wird während des Betriebes mit maximal 120 Watt belastet. Neben zwei USB-Schnittstellen des Typs A und Micro, finden sich am GM-100 zwei RJ45-Buchsen, über welche das Gateway über die Protokolle TCP/IP oder UDP in ein bestehendes Netzwerk eingebunden werden kann. Eine Kommunikation des Geräts über das Modbus Protokoll ist möglich. Die dritte RJ45-Buchse kann für die Kommunikation in Netzwerken, die auf den Protokollen CAN oder RS485 basieren, verwendet werden. Dabei wird eine Abschlussterminierung innerhalb des Geräts direkt standardmäßig eingestellt.

Eine „General Purpose Input Output“ (GPIO) -Schnittstelle stellt zwei konfigurierbare Ein- oder Ausgänge zur Verfügung. An diese können beispielsweise einfache Sensoren angeschlossen werden, deren Ausgangssignale digitale Pegel darstellen. Falls Sensoren mit analogen Ausgangssignalen verwendet werden, können diese über die zwei vorhandenen 12 Bit genauen Analogkanäle eingelesen werden. Eine Ausgabe eines analogen Signals ist ohne Weiteres nicht möglich. Weiterführend ist ein mechanischer Schalter in Form eines Relais als Wechselkontakt für ohmsche Lasten auffindbar. Das Relais verfügt über einen „normally closed“, sowie einen „normally opened“ Kontakt.

An der Frontseite befinden sich drei RGB-LEDs, welche zu Diagnosezwecken verwendet werden können. Eine der drei LEDs ist hierbei in der Anwendersoftware frei programmierbar, um einen eigenen Diagnosestatus optisch ausgeben zu können.

Neben der erwähnten Ethernet-Schnittstellen besitzt das GM-100 ein integriertes WLAN-Modul und somit die Möglichkeit das Gateway kabellos in das Produktionsnetzwerk einzubinden. Aus Zuverlässigkeits- und Leistungsgründen sollte eine kabelgebundene Einbindung jedoch bevorzugt werden.

An der Rückseite des Gateways lässt sich zur einfachen Befestigung in einem Schaltschrank ein Hutschienenadapter montieren.

Die Umgebungstemperatur für den Betrieb des Gerätes sollte 0 °C nicht unter- und 50°C nicht überschreiten. Des Weiteren ist eine trockene und möglichst staubfreie Betriebsumgebung aufgrund der Schutzklasse IP20 nötig.[18]

4.3 Erstinbetriebnahme über serielle und Netzwerkschnittstelle

Für die erste Inbetriebnahme wird das GM-100 über eine serielle Schnittstelle angesprochen. Dazu wird die Micro USB Schnittstelle, sowie ein serielles Kommunikationsprogramm, wie bspw. PuTTY, verwendet. Die Kommunikation basiert auf einer „Universal Asynchronous Receiver Transmitter“ (UART) Verbindung mit einer Geschwindigkeit von 115200 Baud.

[18] BetriebsanleitungIoTGatewayHubGM1002022

Mittels des Benutzernamens „root“ und des Passworts „hubadmin“ gelingt ein Zugriff auf das Gateway. Nach dem Login präsentiert sich eine Shell-Oberfläche eines GNU-Betriebssystems, welche als Kommandointerpreter dient.

Des Weiteren kann das InHub GM-100 über eine Weboberfläche angesprochen werden. Dazu ist es nötig das Gateway in ein bestehendes Netzwerk einzubinden. Dies geschieht durch die Verbindung mittels Ethernet Kabel an Port eins des GM-100. Um den Softwarefunktionen des Gateways einen vollen Internetzugang zu ermöglichen, kann es notwendig sein die Konfigurationsdatei des zugrunde liegenden Betriebssystems entsprechend anzupassen.

Die IP-Adresse des Geräts kann konfiguriert oder per DHCP-Dienst des Netzwerk-Routers zugewiesen werden. Über die IP-Adresse mittels des Zusatzes der Form

http://XXX.XXX.XXX.XXX/smac

gelingt ein Aufruf der Weboberfläche. Für den Login in die SIINEOS Weboberfläche sind der Nutzername „hubadmin“, sowie das gleichnamige Passwort „hubadmin“ notwendig.

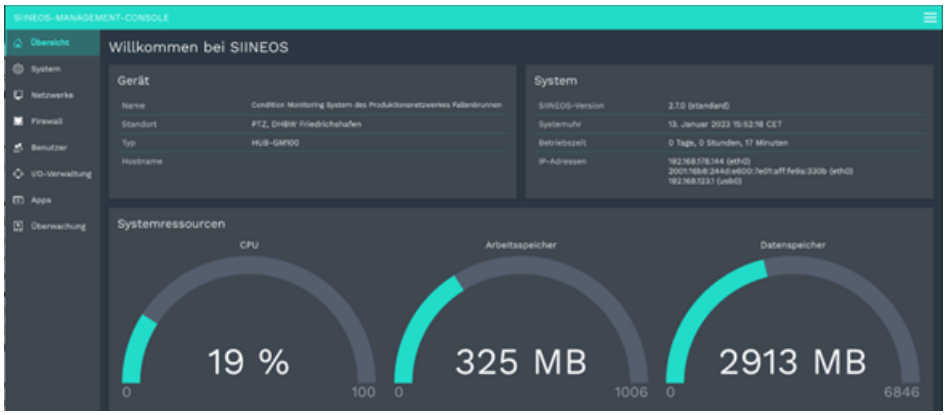


Abb. 4.1: SIINEOS Management Console

Die SIINEOS-Management-Console (vgl. Abb. 4.1) gibt einen Überblick über den aktuellen Gateway Zustand. Des Weiteren lassen sich in der grafischen Benutzeroberfläche einfache, jedoch wichtige Aspekte des Gerätes konfigurieren und überwachen. Dazu zählen neben den Netzwerkeinstellungen, auch Firewall-Anpassungen, sowie eine Benutzerverwaltung. Es lassen sich weiterführend die Ein- und Ausgangsports des Gerätes, die in Kapitel 4.1 (Zielsetzung bei der Implementierung des Condition Monitoring Systems) vorgestellt wurden konfigurieren und deren Zustände auswerten. Im Applikations-Tab werden alle installierten Programme und Dienste aufgelistet. Dies ermöglicht einen schnellen Zugriff auf die unterschiedlichen Softwarepakete.

4.4 Vorbereitung der Softwaretools

Um auf dem IoT-Gateway eigene Programme in Betrieb nehmen zu können, bedarf es mehreren Werkzeugen zur Implementierung und Verwaltung von Programmdateien.

Der generelle Zugriff auf das IoT-Gateway findet nach der Erstinbetriebnahme über das „Secure Shell“ (SSH) Protokoll statt. Eine solche Verbindung kann über das Programm „Windows Power Shell“ aufgebaut werden. Der Befehl zur Verbindung mit dem Gateway lautet dann wie folgt:

ssh hubadmin@192.168.178.144

Die IP-Adresse muss der Hostadresse des Gateways entsprechen. Das für den Benutzernamen „hubadmin“ dazugehörige Passwort lautet standardmäßig „hubadmin“. Abbildung 4.2 zeigt einen erfolgreichen SSH Login Prozess, sowie die Schritte zur Erlangung der Superuser Rolle auf dem System.

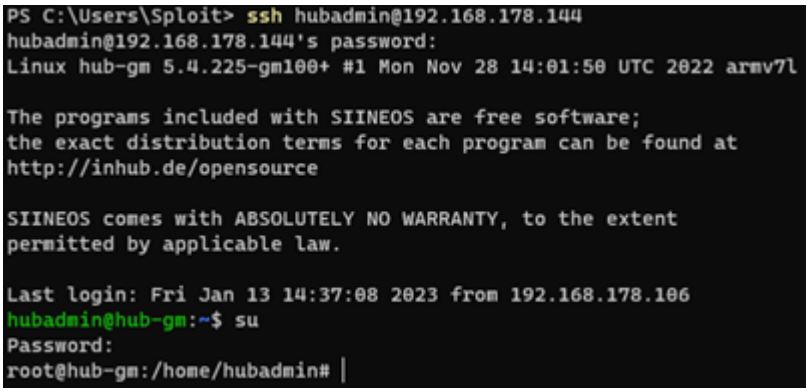


Abb. 4.2: SSH-Login und Erlangung von Superuser-Rechten

Nach einem erfolgreichen Login über SSH, kann das Gateway genauso verwaltet werden, wie es in Kapitel 4.3 (Erstinbetriebnahme über serielle und Netzwerkschnittstelle) nach der Erstinbetriebnahme und der Kommunikation über die serielle Schnittstelle möglich war.

Der Hersteller schlägt des Weiteren vor, das Programm „Qt Creator“ als Entwicklungsumgebung zu verwenden. Durch eine integrierte Netzwerkkomponente, können mittels des Qt Creators eigens kreierte Programme direkt auf das netzwerkgebundene Gateway aufgespielt werden. Das Dokument „InCore Install Guide“ leitet den Anwender hierbei durch den Einrichtungsprozess der Entwicklungsumgebung.[19]

Für die Verwaltung von Benutzerdaten, kann das Programm „WinSCP“ verwendet werden. Über einen Netzwerkzugriff via „Secure File Transfer Protocol“ (SFTP) können somit Dateioperationen im bekannten Stil eines Dateiexplorers auf dem GM-100 durchgeführt werden.

[19] InCoreInstallGuide

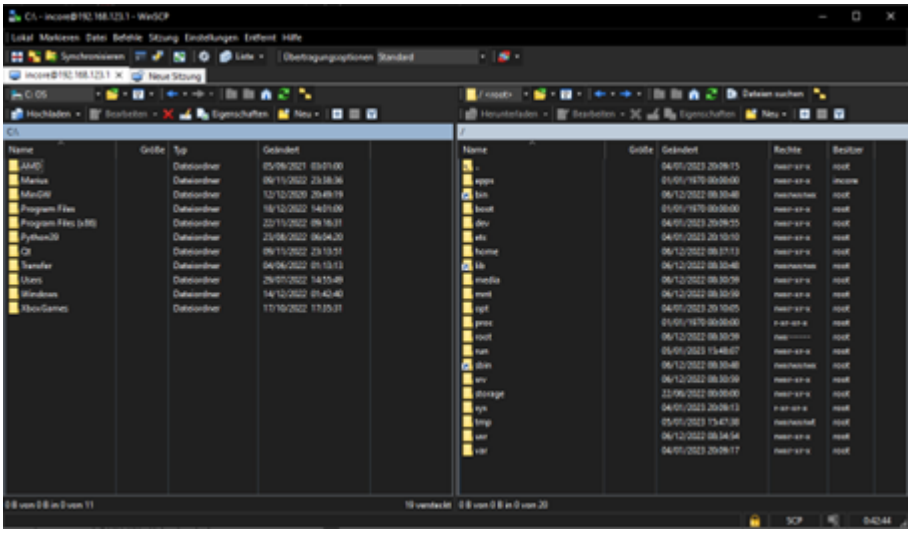


Abb. 4.3: WinSCP Benutzeroberfläche zur Dateiverwaltung

Für die spätere Verwaltung und die grafische Darstellung der zentralen Datenbank eignet sich das Programm „DBeaver“. Das Programm ermöglicht es eine einfache und grafisch gestützte Administration der zeitreihenbasierten Datenbank durchzuführen. Der integrierte „Structured Query Language“ (SQL) Skript Editor stellt hierbei eine große Hilfe dar, wenn es darum geht, sogenannte „Queries“ bzw. Abfragen zu erstellen. Des Weiteren bietet das Programm einen Datenbankexplorer, mit welchem die einzelnen Tabellen inspiziert und verändert werden können. Abbildung 4.4 zeigt die Benutzeroberfläche der Software. Links ist das Datenbank Dateisystem bzw. dessen Struktur zu erkennen. Im oberen Bereich ist die Eingabemaske für SQL-Skripte ersichtlich. Resultierende Ergebnisse werden in den zwei darunterliegenden Fenstern ausgegeben. Im untersten Teil ist das Traffic-Dashboard zu erkennen. Welches aktive Datenbankverbindungen, sowie Transaktionen und Datenbankzugriffe grafisch präsentiert.

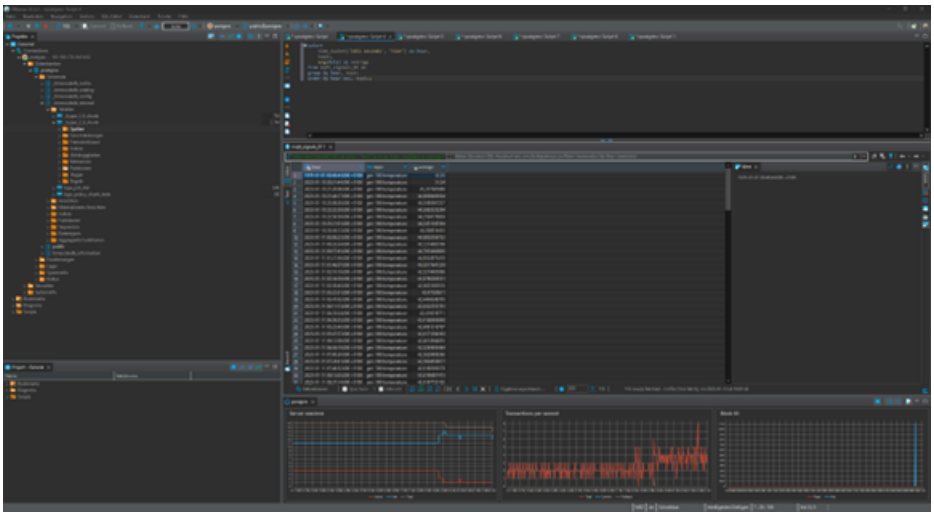


Abb. 4.4: DBeaver Benutzeroberfläche zur Verwaltung von Datenbanken

5 Softwareseitige Implementierung des Condition Monitoring Systems

Dieses Kapitel widmet sich der Auswahl, Konfiguration und Inbetriebnahme diverser Softwaresysteme zur gesamtheitlichen Erstellung eines CM-Systems. Dazu wird zuerst auf die Betriebssystem Basis der Softwarepakete eingegangen. Weiterführend wird das Konzept des MQTT-Netzwerkes aufgegriffen und die Funktion im industriellen Umfeld verdeutlicht. Die Datenspeicherung mittels zeitreihenbasierter Datenbank wird behandelt. Neben den Werkzeugen zur Auswertung der gesammelten Daten, wird betrachtet, wie eigens programmierte Softwarepakete auf dem Gateway ausgeführt werden können. Zuletzt wird die Software „Peakboard“ als mögliches MES vorgestellt, sowie dessen erstellte Schnittstelle zum Produktionssystem dargelegt.

5.1 Docker als Basis für Softwarepakete

Das Grundgerüst der Softwarepakete stellt das Virtualisierungsprogramm Docker dar. Dieses ist dazu in der Lage sogenannte Container für ausgewählte Anwendungen zu erstellen, sodass diese isoliert voneinander, jedoch auf der selbigen darunter liegenden Hardware, arbeiten. Durch die Prozessvirtualisierung gewährt Docker eine Betriebssystem unabhängige Ausführung der Softwarepakete. Das InHub GM-100 besitzt die Docker-Software bereits vorinstalliert. Der Hersteller entschloss sich ebenfalls seine vorinstallierten Programme als Docker-Container auf dem Gateway zu hinterlegen und auszuführen.[20]

5.1.1 Vorteile von Docker

Der Vorteil durch die Ausführung der Softwarepakete in Form von Containern liegt, zum einen in der Übertragbarkeit und Systemunabhängigkeit dieser. Ein Container, welcher auf einem Windows basierten Docker-System erstellt wurde, kann ebenfalls auf einem Linux basierten Docker-System ausgeführt werden.

Die Docker Engine sorgt bei der Erstellung der Container für die Einrichtung einer speziell vorab definierten Betriebssystemumgebung. Des Weiteren können unterschiedliche Architekturen vorab eingestellt werden, sodass beispielsweise die Ausführung von Containern sowohl auf Basis einer Desktop-typischen AMD64- oder AMD86-, aber auch einer mobilen ARMV7-Plattform stattfinden kann. Die Problematik der unterschiedlichen Prozessorbefehlssätze wird durch die Virtualisierung und universelle Schnittstelle von Docker zum eigentlichen Betriebssystems des hostenden Geräts gelöst. Abbildung 5.1 verdeutlicht die Lösung dieser Problematik nochmals.[21]

[20] Cook, vgl. S. 29
[21] Cook, vgl. S. 30

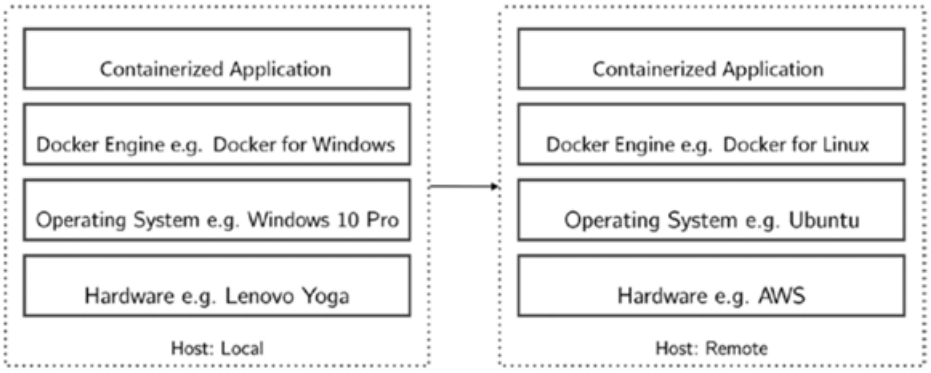


Abb. 5.1: Das Prinzip von Docker und dessen Containern - Quelle: Joshua Cook

5.1.2 Erstellen und Verwenden von Docker Containern

Die Hersteller von bekannten Softwarepaketen bieten heutzutage bereits erstellte Docker Images an. Diese Images muss man sich wie eine Art Bauanweisung für den zu erzeugenden Container vorstellen. Nach dem Herunterladen eines solchen Image ist es möglich unter der Angabe von Optionen für die Containerisierung spezifische Einstellungen zu treffen. Zu diesen zählen neben Portfreigaben, bspw. auch die Vergabe von Passwörtern für den auszuführenden Dienst im Container. Nach erfolgreichem Bau des Containers, kann dieser auf der Plattform ausgeführt werden.

Ein Docker Image lässt sich mittels mehrerer Befehle herunterladen. Neben dem „docker run <image-name/tag>“ Kommando, gibt es auch den Befehl „docker pull <image-name/tag>“. Der erste Befehl wird verwendet, um bspw. Images von Herstellern herunterzuladen und direkt in Form eines Containers auszuführen. Mit dem zweiten Befehl lässt sich ein Image aus einem spezifischen Repository herunterladen. Ein solches Repository wird zu einem späteren Zeitpunkt der Implementierung für die Ausführung eigens kreierter Container benötigt.

Eine Liste der gebauten Container lässt sich mit dem Kommando „docker container ls -a“ ausgeben. Aktive Container lassen sich mittels „docker ps“ auflisten (vgl. Abbildung 5.2). Container lassen sich ebenfalls starten und stoppen. Dazu wird der Befehl docker start <container-name> bzw. „docker stop <container-name>“ verwendet. Dass ein Container nach dem Starten des Docker Dienstes direkt gestartet wird, wird bei der Erstellung des Containers die Option „–restart always“ angegeben. Falls es zu Problemen mit Sicherheitsmodulen bei der Ausführung unter Linux Debian kommt, kann die Option „–security-opt seccomp=unconfined“ Abhilfe schaffen.

5.2 MQTT-Netzwerk als Kommunikationsnetzwerk

Das „Message Queuing Telemetry Transport“ (MQTT) Protokoll ist ein simpler Gerät zu Gerät Kommunikationsweg auf Basis des „Representational State Transfer“ (REST)

```
root@hub-gm:/home/hubadmin# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
a06f44e1ee56   prom/prometheus                      "/bin/prometheus --c…"   4 hours ago    Up 4 hours
279dc1ea71fc   grafana/grafana-oss                  "/run.sh"                4 hours ago    Up 4 hours
90d37804039f   nodered/node-red:2.2.3-16-minimal    "npm --no-update-not…"   4 hours ago    Up 4 hours (healthy)
29dc6043ffc9   spl01t/t3000_01_dockerpython         "python app.py"          13 hours ago   Up 4 hours
2b412d5ef086   timescale/timescaledb:2.0.0-pg12     "docker-entrypoint.s…"   3 days ago     Up 4 hours
```

Abb. 5.2: Auflistung der aktiven Docker Container

Prinzips auf der TCP/IP Ebene. Durch Methoden des HTTP-Protokolls kann durch das gegenseitige Anfragen von Teilnehmern ein Datenaustausch durchgeführt werden. Die Client-Broker-Netzwerkarchitektur basiert auf dem Abonnier-Prinzip von bestimmten Kanälen. So können die Teilnehmer auswählen, welche Informationen von welchen Kanälen für sie interessant sind. Der Broker ist dabei für die Verarbeitung der sogenannten „Subscriptions“ zuständig und leitet den Teilnehmern die entsprechenden Nachrichten weiter.[22]

Abbildung 5.3 zeigt wie das MQTT-Netzwerk in einer beispielhaften IIoT Umgebung funktioniert. Der Broker stellt dabei den zentralen Austauschknoten des Netzwerkes dar. Bei der Implementierung des CM-Systems wird der Broker auf dem IoT-Gateway ausgeführt. Auf dem gleichen laufen ebenfalls der Datenbank-Server und andere Dienste. Dies wird im rechten Teil der Abbildung dargestellt. Die Systeme wie ERP oder MES sind ebenfalls an das Gateway angebunden. In dieser Arbeit kommuniziert das MES ebenfalls über MQTT mit einer Softwareanwendung auf dem IoT-Gateway. In Abbildung 5.3 ist dies nicht der Fall.

Der wichtigste Teil findet sich im linken, oberen Teil wieder. Die SPS, sowie IoT fähige Geräte kommunizieren in unserem Fall ausschließlich über MQTT miteinander. Die in der Abbildung blau dargestellten Kommunikationswege stellen die Verbindungen zwischen den Clients, bspw. in Form von mit Sensorik ausgestatteten Mikrocomputern, und dem Gateway-Broker dar.

Das Netzwerk auswertende Softwaretools können über „human machine interfaces“ (HMIs) bedient und kontrolliert werden.

5.2.1 MQTT-Broker

Das GM-100 IoT Gateway dient als zentraler MQTT-Broker. Durch die Verwendung eines einfachen Javascripts lässt sich der interne MQTT-Broker des GM-100 aktivieren (vgl. Anhang B – InCore Programm – MQTT-Broker). Der Broker ist über die IP des Gateways im gleichen Netzwerk erreichbar. Für den MQTT-Dienst wird der Standard-Port 1883 verwendet. Die Netzwerk-Teilnehmer sind nach Start des Brokers in der Lage miteinander zu kommunizieren. Für die Kommunikation über das geschaffenen Netzwerk ist weder eine Authentifizierung noch eine Verschlüsselung der Botschaften notwendig.

[22] Wimmer und Garrigós, vgl. S. 49

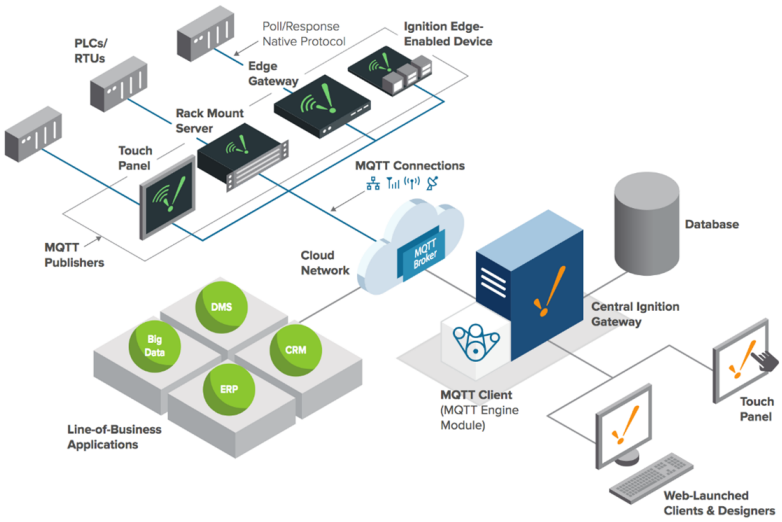


Abb. 5.3: Schematische Funktionsweise eines MQTT-Netzwerkes - Quelle: Ignition Edge IIoT

Dies vereinfacht den Kommunikationsprozess unter der Annahme, dass die im Netzwerk verteilten Informationen für keinen Dritten von Interesse sind.

5.2.2 Interner MQTT-Client zum Empfangen von Daten

Unter der Annahme, dass in Zukunft die Daten von IoT-Nodes, die mit Sensoren ausgestattet sind, über den MQTT-Broker an das Gateway gelangen sollen, wird ein Javascript erstellt, welches in der Lage ist, die MQTT-Nachrichten zu interpretieren und ggf. weiterzuverarbeiten.

In Anhang C wird der Algorithmus dargelegt, mit welchem das Gateway als MQTT-Client Informationen vom Broker abgreift. Dazu wird in Zeile 87 eine MQTT-Client Instanz der InCore Bibliothek erstellt. Unter Angabe der internen Broker Adresse„127.0.0.1“ verbindet sich der Client mit dem zentralen Datenverteiler. In Zeile 94 wird eine Publikation definiert, welche nach dem Erhalt einer aktualisierten Prozessor-Temperatur die Information auf dem Kanal „test/temperature“ veröffentlicht. In den darauffolgenden Zeilen wird der Kanal „gm-100/temperature“ abonniert. Diese Listener-Instanz führt beim Erhalt neuer Daten vom Broker eine Speicherung der Information in einer SQL-Datenbank durch. Hierfür wird in Zeile 101 ein spezifischer Zeitstempel mit Zeitzone für den Zeitpunkt des Informationserhalts erstellt. Dieser wird in Zeile 107 an den Database-Handler übergeben. In den Zeilen 108 und 109 wird jeweils der Name des abonnierten Kanals, sowie die konkrete Information übergeben. Zuletzt wird die blaue Diagnose-LED des GM-100 eingeschaltet, um den Anwender zu signalisieren, dass der Empfang einer MQTT-Nachricht stattfand. Diese LED wird von einem Timer nach 500 Millisekunden wieder deaktiviert. (vgl. Zeile 74f).

Diese Schnittstelle dient somit der universellen Verarbeitung der empfangbaren MQTT-Daten. Die Abspeicherung in eine Datenbank ist nur eine Möglichkeit, die Daten weiterzuleiten und zu verwenden. Durch die Javascript Implementierung lassen sich somit komplexe Logiken erzeugen, was für den späteren Verarbeitungsaufwand, bei einer erhöhten Datenrate im aktiven Produktionsnetzwerk von Vorteil werden kann.

5.2.3 Floweditor Node-RED

Node-RED ist ein flowbasierter Editor, der es dem Anwender ermöglicht logische Datenverarbeitungsabläufe mit Hilfe von Bausteinen durchzuführen. Die Weboberfläche des Tools bietet eine Vielzahl von Funktionsabläufen mit denen Daten aus unterschiedlichen Quellen, v.a. aber aus MQTT-Nachrichtenkanälen empfangen werden können. Anschließend lassen sich die Daten auf unterschiedlichste Weise verarbeiten oder grafisch veranschaulichen. Um die Daten zu empfangen, verbindet sich die Node-RED Instanz als MQTT-Client mit dem Broker.

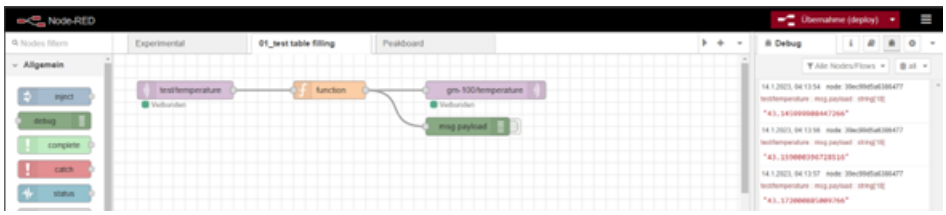


Abb. 5.4: Weboberfläche des flowbasierten Editor Node-RED

Abbildung 5.4 zeigt die Oberfläche der Browser basierten Software. In der linken Spalte finden sich die einzelnen Nodes mit denen in der mittleren Fläche Abläufe angelegt werden können. Der hier dargestellte Ablauf besteht aus einer Eingangs-Node des MQTT-Netzwerks, welche alle Daten aus dem Kanal „test/temperature“ an die Funktions-Node weiterleitet. In dieser können simple Javascript Funktionen kodiert werden, um bspw. eine Rundung der empfangenen Gleitkommazahl auf zwei Nachkommastellen durchzuführen. Das Ergebnis der Funktion wird an eine Ausgangs-Node weitergeleitet, die die Information auf dem Kanal „gm-100/temperature“ veröffentlicht. Die grüne Debug-Node dient der zusätzlichen Ausgabe im sich rechts befindlichen Debug-Fenster. Mittels des roten „Übernahme-Buttons“ lassen sich Änderungen in den Abläufen validieren.

Der Vorteil der Software besteht vor allem in der simplen Verknüpfung von Informationswegen. Falls Anpassungen der Daten in Form von bspw. Typen-Konvertierungen notwendig sind, lassen sich diese mit Hilfe des breiten Pakets an Nodes simpel implementieren. Des Weiteren lassen sich mit der Software schnelle Auswertungen des MQTT-Netzwerk-Verkehrs machen, indem Debug-Nodes die Informationen von ausgewählten Nachrichtenkanälen im Debug-Fenster ausgeben. Ein einfaches Anwendungsbeispiel stellt das Versorgen des MES mit den entsprechenden Anlagen-Daten dar. Es gibt in der Anla-

ge beispielsweise Haltepunkte der Bauteilträger, welche als binäre Zustände an das MES übergeben werden können, um den Pufferzustand des Produktionsbandes darstellen zu können. Da das MES Daten über MQTT-Kanäle empfangen kann, können mit Hilfe der Node-RED Abläufe einfach und schnell die benötigten Informationswege erstellt werden.

5.3 TimescaleDB zur Archivierung von Informationen

Die Datenbank dient als zentrales Archiv aller aufgenommenen Informationen des IoT-Gateways. Alle Daten, die aus dem Produktionsnetzwerk empfangen wurden, werden in der Datenbank hinterlegt, sodass eine Informationsbasis für analysierende Algorithmen geschaffen werden kann.

5.3.1 Relationale und Zeitreihen Datenbanken

Die wichtigsten Anforderungen an das Datenbanksystem stellen die Schnelligkeit und Effizienz der Informationsverarbeitung, die Skalierbarkeit und die möglichst einfache Verwaltung der Daten dar.

Des Weiteren muss die grundlegende Frage der Art der Datenspeicherung geklärt werden. Neben der herkömmlichen Methode der relationalen Datenbanken, welche vor allem darauf ausgelegt sind, Zustände von Objekten festzuhalten, gibt es die Zeitreihen basierten Modelle, welche den Fokus auf die zeitlich sequenzielle Abspeicherung der Informationen legen.

Dabei haben beide Varianten ihre Vor- und Nachteile. Zum einen bieten sich relationale Datenbanken an, um Zustandsdaten abzuspeichern, welche auch nach einem langen Zeitraum wieder verändert werden müssen. Ein klassisches Beispiel wäre hierfür die Anzahl und der Online-Zustand von Netzwerkteilnehmern, die Sensordaten an unser Gateway übermitteln. Diese Datenpunkte müssten egal wie lange das Zeitintervall zwischen Informationsänderungen ist, einfach abänderbar sein, ohne lange Datenoperationen zu verursachen.[23]

Eine Zeitreihendatenbank hingegen bringt den Vorteil der schnelleren Abspeicherung von Daten mit sich. Dies erhöht den maximalen Datendurchsatz des Gesamtsystems. Weiterführend lassen sich effizientere Datenanalysen, in Form von Aggregierungen, mit Hilfe des Zeit-sequenziellen Modells durchführen. Für den Anwendungsfall der Datenaufnahme im IIoT Sektor und der bestehenden Notwendigkeit der kontinuierlichen Datenauswertung, stellen Zeitreihendatenbanken die bessere Alternative dar.

Neben den ebenfalls als „NoSQL“ bezeichneten Datenbanken, gibt es aber auch Mischmodelle. Diese bauen auf relationalen „SQL“ Datenbanken auf, besitzen jedoch ein Datenbankverwaltungssystem, welches die Informationen auf eine effiziente Weise abspeichert.

[23] SudokrewApplicationWeb

Somit wird eine zeitlich aufeinander ab folgende Datenspeicherung auf dem herkömmlichen Datenbankmodell auf Ressourcen schonende Weise möglich. Eine solche Mischform stellt das Datenbankverwaltungssystem Timescale dar. Es basiert auf dem relationalen Modell der PostgreSQL Datenbank und dient lediglich als Erweiterung dieser, um eine effiziente Steuerung von zeitsensitiven Datenströmen zu gewährleisten.[24]

5.3.2 TimescaleDBs Konzepte der Datenverwaltung

Das Grundprinzip der Timescale Datenbank besteht in der Unterteilung von Datentabellen anhand zeitlicher Intervalle. Dies geschieht in sogenannten Hypertables, welche in Form von Chunks in Abhängigkeit der Länge des eingestellten Zeitintervalls unterteilt im Speicher hinterlegt werden.

Der Vorteil dieser Speicherungsmethode tritt auf, wenn es darum geht kürzlich aufgenommene Daten zu analysieren. Das Datenbankverwaltungssystem lädt die zu analysierenden Daten aus ihren Chunks in den Arbeitsspeicher und muss somit nicht auf eine große Tabelle, welche sich auf dem langsameren Speicher befindet, zugreifen.

Des Weiteren bietet TimescaleDB unterschiedliche Funktionen an, um die Daten in Form von bspw. Zeitintervallen zu aggregieren. Somit lassen sich komplexe Datenabfragen in Form von kurzen SQL Queries durchführen. Weiterführend existiert die Option der kontinuierliche Aggregierung von Daten. Mit diesen lassen sich automatisch, auf Basis des Grunddatensatzes, neue Hypertables erstellen, welche bspw. gemittelte Werte von Daten aus spezifizierten Zeitintervallen bestimmen. Dieses Konzept gepaart mit der Kompression von Daten ist wichtig, um Datenmengen möglichst kompakt zu halten, wenn es darum geht eine statistische Datenbasis für ein späteres Machine Learning Modell zu erstellen. Dieses Modell muss zukünftig ebenfalls mit aktuellen Prozessdaten versorgt werden, was ebenfalls über Datenaggregate geschehen wird.

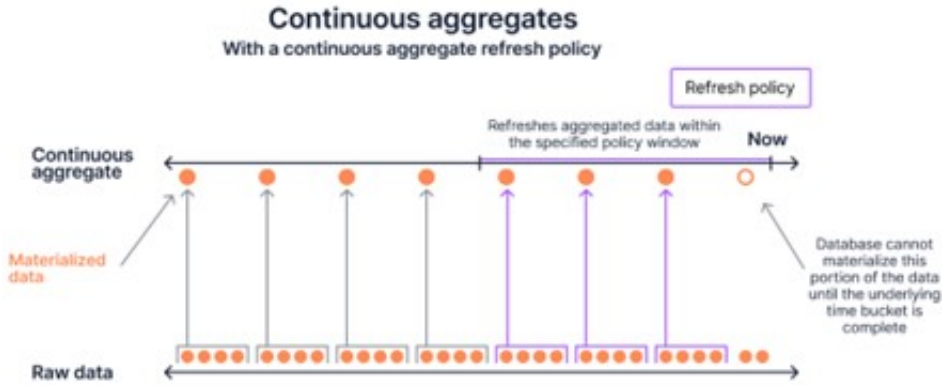


Abb. 5.5: Funktionsweise des kontinuierlichen Datenaggregats mit automatischer Aktualisierungsrichtlinie - Quelle: Timescale

[24] TimescaleDocumentationWhy

Abbildung 5.5 zeigt das Funktionsprinzip der Datenaggregierung in der Timescale Datenbank. Die Rohdaten werden innerhalb festgelegter Zeitintervalle und nach entsprechender Verarbeitung zu einem Datenaggregat hinzugefügt. Dieser neue Hypertable besteht aus materialisierten Daten, welche anhand einer Aktualisierungsrichtlinie aktuell gehalten werden (vgl. lila Zeitintervall). Die Richtlinie gibt an, wie lange das Datenaggregat mit aktualisierten Daten versorgt wird, bevor diese entweder nur noch unverändert im neuen Hypertable bestehen bleiben oder aufgrund einer zeitlich festgelegten Entfernung alter Datensätze im Rohdatensatz, ebenfalls nach der darauffolgenden Aktualisierung des Aggregats entfernt werden.[25]

Durch die Festlegung einer sogenannten „data retention policy“ können ältere unbrauchbare Daten automatisch aus der Datenbank entfernt werden. Legt man eine solche Richtlinie auf einen Hypertable mit dem Intervall von bspw. einer Woche an, werden im zeitlichen Abstand eines festgelegten Auftragsintervalls die Daten, die älter als eine Woche sind automatisch gelöscht. Dies spart Speicherplatz und Systemressourcen und schafft gleichzeitig für Übersichtlichkeit in der Datenbank.[26]

5.3.3 Installation und Verwendung der TimescaleDB

Die Installation des Datenbanksystems erfolgt mit Hilfe eines von Timescale erstellten Docker Images. Dieses kann heruntergeladen und als Container ausgeführt werden. Der Datenbankserver agiert auf dem Port 5432, welcher beim Erstellen des Containers in den Optionen angegeben werden muss.

Nach der Installation des Images und dem Starten des Containers, ist ein Interagieren mit der Datenbank mittels DBeaver oder dem PSQL-Kommandozeileninterpreters möglich.

Es wurde zu Testzwecken ein Hypertable mit der Bezeichnung „mqtt_signals_01“ erstellt, welcher als Hauptdatenspeicher für die in dieser Arbeit umgesetzten Informationen dient. Ein kontinuierliches Aggregat mit dem Namen „cputemp_agr_avg_hourly“ berechnet auf Basis von stündlichen Intervallen einen Mittelwert aus den Rohdaten des zuvor erstellten Hypertables. Im folgenden Kapitel zur grafischen Auswertung der Daten mittels Grafana werden die beiden Datenspeicher vergleichbar dargestellt. Dabei lassen sich die aggregierten Datenpunkte sehr gut erkennen (vgl. Abbildung 11).

Der Zugriff auf die Datenbank erfolgt mittels SQL Queries. Eine solche Query ist bspw. Anhang C zu entnehmen. Genauer wird in Zeile 146 eine PostgreSQL Instanz erstellt, welche auf die Datenbank unter Angabe eines Benutzernamens und Passworts zugreifen kann. Dies geschieht mit Hilfe der SQL Query, welche in Zeile 154 hinterlegt ist. In diesem Fall fügt das Programm unter Anwendung der SQL-Methode „INSERT“

[25] TimescaleDocumentationContinuous
[26] TimescaleDocumentationData

einem Hypertable die Daten aus den Objekten „mqttDB_time“, „mqttDB_topic“ und „mqttDB_data“ hinzu.

5.4 Grafana zur grafischen Datenauswertung

Das Softwarepaket Grafana kommt vorinstalliert mit dem GM-100. Die Hauptaufgabe des Programms ist die Visualisierung von Daten. Dazu greift es auf hinterlegte Datenbanksysteme zurück. Mit Hilfe von SQL Queries können Daten gesammelt und entsprechend dargestellt werden.

5.4.1 Einrichtung von Grafana

Zu Beginn muss die Verknüpfung zur Datenbank geschaffen werden. Dies geschieht über den Reiter Konfiguration und Datenquellen. Hier wird eine neue PostgreSQL Datenquelle angelegt und mit entsprechenden Informationen befüllt. Es bietet sich an nach der Befüllung der Daten einen Verbindungstest mit Hilfe des sich im unteren Teil befindlichen Buttons durchzuführen.

5.4.2 Datenvisualisierung und Erstellung von Dashboards

Anschließend kann im Reiter Erkunden die Erstellung und Validierung von SQL Queries durchgeführt werden (vgl. Abbildung 5.6). Dazu wählt man die zuvor konfigurierte Datenquelle aus und verwendet entweder eine eigens kreierte SQL Query oder lässt sich mit dem Grafana Query Editor behelfen. Im oberen Teil findet sich die Zeitintervalleinstellung, welche als Zeitfilter in den Queries verwendet werden kann, um beim Erkunden der Daten simpel in Zeitbereichen springen zu können.



Abb. 5.6: Grafana Reiter Erkunden

Hat man eine Visualisierung erstellt, welche dauerhaft abrufbar sein soll, lässt sich die Kreation aus dem Erkunden Reiter auch in ein Dashboard konvertieren. Dieses ist dann unter dem Reiter Dashboards auffindbar. Das in Abbildung 5.7 dargestellte Dashboard zeigt im oberen Diagramm die vom GM-100 über MQTT empfangene CPU-Temperatur des Geräts an, welche in einem Intervall von einer Sekunde abgespeichert wird. Das untere Diagramm zeigt das zugehörige Aggregat, welches die Temperaturwerte stündlich zusammenfasst und einen Mittelwert daraus bildet.



Abb. 5.7: Dashboard der CPU-Temperatur mit rohen und aggregierten Daten

5.4.3 Alarmierung bei Grenzwertüberschreitung

Des Weiteren ist es in Grafana möglich Alarme zu definieren. Diese verfolgen den Verlauf ausgewählter Datenströme und benachrichtigen den Anwender auf unterschiedlichen Wegen, sobald Schwellwerte für bestimmte Zeiträume überschritten werden. Die im oberen Diagramm ersichtlichen horizontalen Linien in Abbildung 5.7 stellen Events eines Alarms dar, welcher bei der roten Markierung ausgelöst und bei der darauffolgenden grünen Markierung wieder aufgehoben wurde. Grund der Alarmierung war die für den Moment eingestellte Temperaturgrenze von 43 °C, welche kurzzeitig überschritten wurde.

Die Alarmierung erfolgt in diesem Fall über das Portal Slack. Die in Abbildung 5.8 dargestellten Benachrichtigungen, werden von Grafana über einen Webhook in einer Slack Entwicklungsapp ausgelöst.



Abb. 5.8: Slack Benachrichtigung ausgelöst durch Grafana Alarm

5.5 Implementierung eigener Softwarepakete

In den bisherigen Kapiteln wurden Softwarepakete renommierter Hersteller verwendet. Eine Installation des entsprechenden Docker Images macht das Ausführen der Programme bequem. Jedoch sind die Softwarepakete auf den vom Hersteller vorgesehenen Funktionshorizont beschränkt. Die Implementierung von Javascripts über die InCore Plattform des InHub GM-100 bedient ebenfalls nicht alle Wünsche. Wie ist es dem Anwender möglich ist die eigene Vorstellung von Algorithmen und Funktionsabläufen auf dem IoT Gateway umzusetzen zeigen die folgenden Kapitel.

5.5.1 Python Skripte in Form von Docker Containern

Die Problematik der Funktions-Beschränktheit lässt sich durch die Entwicklung eigener Python Skripte lösen. Durch diese Methode ist es dem Anwender möglich, Algorithmen in einer bekannten Desktop-Umgebung zu entwickeln und zu testen. Nach erfolgreicher Funktionsabnahme bedarf es einer Konvertierung des Programms zu einem Docker Image, welches anschließend auf unserem Gateway, genauso wie die Images der anderen Hersteller, installiert und in einen Container verwandelt werden kann.

Es wurde sich darauf festgelegt, die IDE Visual Studio Code zu verwenden. Sie bietet zahlreiche Erweiterungen, die das Entwickeln und Verwandeln eines Programms in ein Docker Image erleichtern.

Des Weiteren können die Entwicklung und das Überprüfen der Images auf einem Windows oder Linux basierten Betriebssystem unternommen werden. Windows bietet das sogenannte „Windows Subsystem für Linux“ (WSL) an, mit welchem bspw. die Linux Distribution Ubuntu virtualisiert werden kann. Das Verrichten der Arbeit in einer Linux basierten Umgebung hilft dabei, mögliche Problematiken auf dem ebenfalls Linux basierten IoT Gateway, während den Funktionstests aufzudecken. Beispielhafte Problematiken, auf die man während der Entwicklung der Python Images stieß, waren bspw. das Fehlen bestimmter Bibliotheken, die zum Bau des späteren Docker Container benötigt wurden. Das Aufdecken und Ausmärzen solcher Fehler in der virtualisierten Umgebung, steigert die Effizienz und Wahrscheinlichkeit eines erfolgreichen Deployments des finalen Docker

Images.

5.5.2 Umgebungsbedingungen und Abläufe festlegen mittels Dockerfile

Das sogenannte Dockerfile stellt eine Art Anleitung dar, um der Docker Engine beim Bau des Docker Containers gewisse Randbedingungen und Abläufe zu vermitteln. So wird im Dockerfile bspw. die Betriebssystemumgebung und -Architektur des virtualisierten Systems vorgeschrieben. Die Docker Engine installiert daraufhin das entsprechende Betriebssystem und führt alle weiteren im Dockerfile definierten Befehle aus.

In Anhang D ist eine solche Datei zu erkennen. In Zeile 2 wird die Architektur und das Betriebssystem festgelegt. Ab Zeile 12 sind Befehle aufgelistet, die von der Bash des zuvor installierten Systems ausgeführt werden. In diesem Beispiel wird das Verzeichnis des Paketmanagers ßgqqapt in Zeile 12 aktualisiert. Anschließend finden Installationen zweier benötigter Pakete statt, um den restlichen Docker Container später zu bauen. Mit dem Befehl in Zeile 15 wird der Python Interpreter dazu aufgefordert die in der Datei „requirements.txt“ definierten Python-Bibliotheken zu installieren. In Zeile 26 findet der letzte Aufruf statt, welcher die Ausführung des Hauptprogramms „app.py“ durchführt.

5.5.3 Python Programm zur Exportierung von Daten zu Analysezwecken

Für die Analyse der aufgenommenen Daten, müssen diese in einem geeigneten Format exportiert werden.

Es muss zunächst festgehalten werden, welche Daten aus welchen Zeitintervallen exportiert werden sollen. Dazu bedient man sich den in Kapitel 2.3.2 vorgestellten Datenaggregaten des Timescale Datenbanksystems. Mit diesen ist es möglich Rohdaten über bestimmte Zeitintervalle zusammenzufassen und vorzuverarbeiten. Das angelegte Aggregat „cputemp_agr_avg_hourly“ speichert hierbei den Datensatz der CPU-Temperatur in stündlich gemittelten Werten ab. Diese Werte sollen nun stündlich aktualisiert in eine Exceldatei exportiert werden. Die Weiterverwendung dieser Daten erfolgt dann anhand eines Datenimports aus der erstellten Datei.

Der in Anhang E dargelegte Algorithmus verarbeitet diese Daten entsprechend. Des Weiteren wurde testweise ein MQTT-Client implementiert, um die Verbindung aus dem Docker Container auf den Gateway-Broker zu testen. Dieses Vorgehen ähnelt dem in Kapitel 2.2.2 vorgestellten Programmablauf.

Zuerst müssen die entsprechenden Bibliotheken importiert werden. Für den Zugriff auf den MQTT-Broker wird die „Paho MQTT“ Bibliothek, für Datenbank Transaktionen wird die Bibliothek „Psyopg2“ und für das Exportieren der Daten wird die Bibliothek „XLSXWriter“ verwendet. Die Zeitbibliotheken, sowie die Threading Bibliothek kommen vorinstalliert mit dem Python3 Paket und müssen daher nicht in der Dockerfile „requirements.txt“ angegeben werden.

Die Main-Funktion findet sich ab Zeile 64 wieder. Nach einer initialen Ausgabe des Programmnamens, werden Funktionen ausgeführt, die die MQTT-Verbindung konfigurieren. In Zeile 75 wird die Adresse zur Anmeldung an der PostgreSQL Datenbank erstellt. In Zeile 76 wird die Verbindung zur Datenbank aufgebaut und das handelnde Objekt in einer Variable hinterlegt. Anschließend wird ein Thread erstellt, welcher die Funktion „psql_fetchData“ parallel ausführt. Da es sich bei dieser Funktion um eine unendliche Schleife handelt, wird dieser Prozess sich erst bei einem erzwungenen Programmhalt beenden.

Die Funktion „psql_fetchData“ ist ab Zeile 29 definiert. Sie ist dafür verantwortlich alle 3600 Sekunden einen Datenbankzugriff durchzuführen. Dabei wird die in den Zeilen 23 bis 26 definierte SQL Query verwendet, wie in Zeile 32 zu erkennen ist. Die in Form einer Liste erhaltenen Daten werden in der Variable „qry_res“ festgehalten und in Zeile 36 an die Funktion „xlsx_writeData“ übergeben. Sieht man sich diese entsprechende Funktion an, ist zu erkennen, dass die Parameterliste in eine Xlsx Datei exportiert werden. In Zeile 44 bis 46 wird zuerst ein eindeutiger Dateiname generiert, indem ein Stempel der aktuellen Zeit mit einfließt.

Das in den Zeilen 48 und 49 erstellte Datenblatt wird mittels einer For-Each-Schleife in den Zeilen 54 bis 59 befüllt. Zuletzt wird die Datei geschlossen und ist somit bereit zur weiteren Verarbeitung.

Das Ergebnis in Form einer stündlich neu erstellen Excel-Tabelle ist dann aus dem Container abrufbar. Abbildung 13 zeigt den Ausschnitt eines solchen Datenexports.

	A	B	C
1	13/01/2023 18:00:00	gm-100/temperature	42.12365737
2	13/01/2023 19:00:00	gm-100/temperature	43.07579473
3	13/01/2023 20:00:00	gm-100/temperature	42.51185621
4	13/01/2023 21:00:00	gm-100/temperature	41.65588
5	13/01/2023 22:00:00	gm-100/temperature	41.26950572
6	13/01/2023 23:00:00	gm-100/temperature	41.04605907

Abb. 5.9: Ausschnitt des Datenexports durch das Python Programm

5.6 Ausblick zur weiteren Datenverarbeitung

Die Software Matlab bietet mit der integrierten Toolbox „Diagnostic Feature Designer“ eine Möglichkeit Datensätze zu untersuchen. Dabei werden Informationen aus unterschiedlichen sensorischen Quellen aufgenommen. Des Weiteren sollten sich diese Datensätze im Gesundheitszustand der untersuchten Hardware unterscheiden. D.h. dass ein Teil der Daten an bspw. einem Motor aufgenommen wird, der keine Verschleiß- und baldigen Ausfallerscheinungen aufweist. Der andere Teil der Informationen wird an einem Gerät aufgenommen, welches einen bekannten, kritischen Gesundheitszustand aufweist.

Mit den gewonnenen Informationen ist man mit Hilfe der genannten Toolbox in der Lage dazu, ein Modell zu trainieren, bzw. Auffälligkeiten in den Datensätzen zu erarbeiten. Dazu werden verarbeitende Methoden angewandt, wie bspw. das Umwandeln der Informationen in den Frequenzbereich. Anhand dieser Umwandlungen können selektive Features untersucht werden, die Aufschluss darüber geben, wie sich die Daten bei den aufgenommenen Gesundheitszuständen typischerweise verteilen.

Abbildung 14 zeigt die testweise Importierung und Darstellung zweier Sätze von Temperaturdaten des Gateways unter erhöhter und verminderter Last.

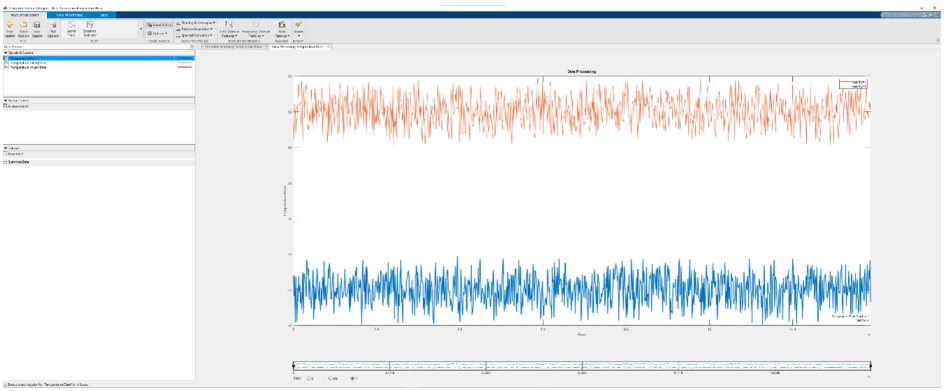


Abb. 5.10: Matlab Toolbox Diagnostic Feature Designer

5.7 MES-Dashboard mit Peakboard Designer

Der Peakboard Designer ermöglicht es Dashboards zu erstellen, die in Echtzeit auf Datenänderungen reagieren. Des Weiteren bietet das Programm an, mehrere Datenquellen zu verwenden. Unter anderem lassen sich über MQTT-Kanäle Daten an das Dashboard weiterleiten. Diese können dann repräsentativ dargestellt werden. Die Logik kann dabei entweder im Backend durch einen eigens entwickelten Algorithmus erfolgen. Peakboard Designer bietet aber auch eine eigene Skriptsprache zur Erstellung einfacher, interner Logiken an.

In diesem Beispiel wurde ein einfaches Template-Dashboard so angepasst, dass 6 Produktionsstationen dargestellt werden (vgl. Abbildung 5.11). Die Ampelanzeige soll dabei den Pufferzustand der Station signalisieren. Testweise wurden die Anzeigen mit MQTT-Kanälen verknüpft und über Node-RED mit Daten versorgt. Auf der rechten Seite ist die aktuelle Uhrzeit, sowie das Ziel an zu produzierenden Einheiten zu erkennen.

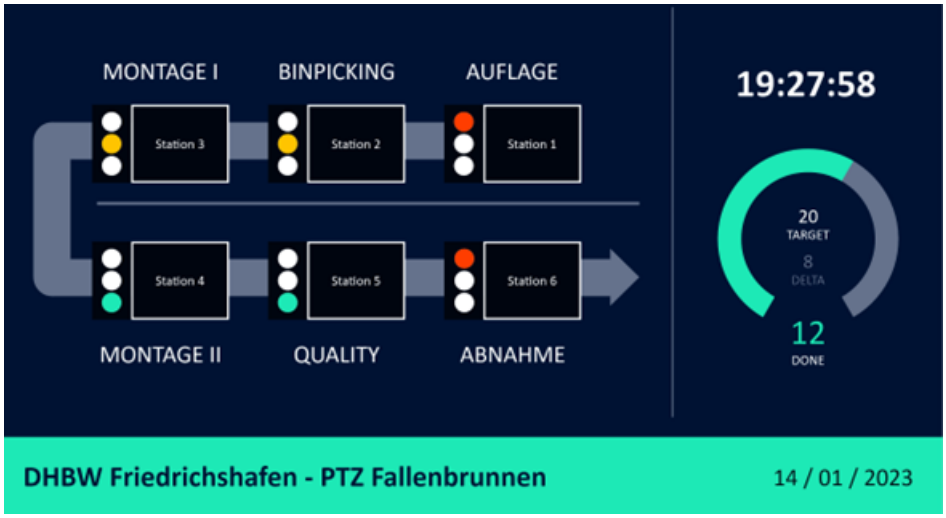


Abb. 5.11: MES-Dashboard aus dem Peakboard Designer

Zum Zeitpunkt der Inbetriebnahme in der Produktionsumgebung können über die MQTT-Kanäle reale Zustandsdaten übertragen werden, die aus dem Produktionsnetz aktiv ermittelt werden.

6 Fazit

Die in dieser Arbeit vorgestellten Methoden zur Implementierung eines Condition Monitoring Systems ermöglichen die Basis der intelligenten Zustandsüberwachung von Maschinen.

Durch die Einrichtung des MQTT-Brokers wird eine universelle und simple Schnittstelle geschaffen, um mit allen benötigten Geräten des Produktionsnetzwerkes zu kommunizieren. Dazu zählen neben der Siemens S7-1500 CPU, IoT-Geräte die Sensordaten zur weiteren Analyse aufnehmen, aber auch Softwaretools, um bspw. das MES-Dashboard informativ aktuell zu halten.

Mit Hilfe der Algorithmen, die in Javascript und Python implementiert wurden, ist das GM-100 dazu in der Lage alle aufgegriffenen Informationen geordnet abzuspeichern. Dies geschieht mit Hilfe der vorgestellten SQL Queries, die Transaktionen auf der Timescale Datenbank ausführen.

Das implementierte Datenbanksystem überzeugt dabei mit den kombinierten Vorteilen von relationalen und Zeitreihen-Datenbanken.

Das vorgestellte Python Skript kann im Hintergrund Datenaggregate verwalten und deren Informationsgehalt für externe Analysen exportieren. Des Weiteren sind schnelle, simple, aber informative, grafische Auswertungen mit Hilfe der Software Grafana möglich. Diese verfügt weiterführend über eine Alarmierungsfunktion, welche zukünftig das Benachrichtigen bei vorzeitig erkannten Problemfällen ermöglicht. Das Konzept der erarbeiteten Prozessabläufe wird in Abbildung 6.1 nochmals vereinfacht dargestellt.

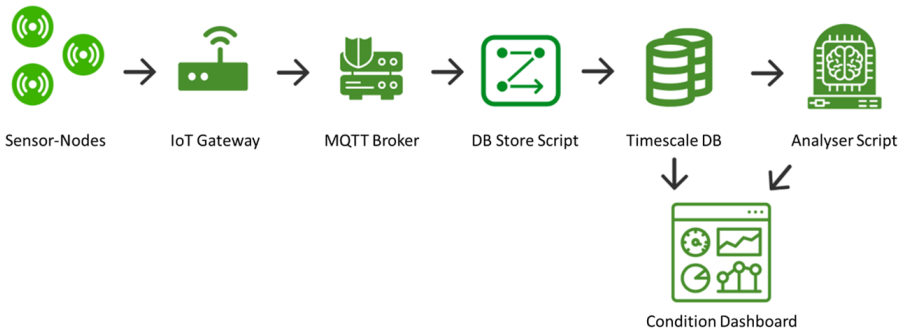


Abb. 6.1: Vereinfachte Darstellung der Prozessabläufe des IoT-Gateways

Zusammenfassend lässt sich daher festhalten, dass durch die in dieser Arbeit getroffenen Schritte, eine optimale Vorbereitung auf das nachfolgende Ermitteln von Datenmodellen durchgeführt wurde. Wie in Kapitel 5.6 (Ausblick zur weiteren Datenverarbeitung). bereits erwähnt wurde, soll anschließend mit Hilfe einer ermittelten Datenbasis, welche Informationen von Sensoren an den Bandmotoren enthält, ein künstliches Intelligenz Modell erstellt werden. Somit sollen zukünftig Aussagen über den Gesundheitszustand der Gerätschaften ermittelbar sein.

A Softwareversionen

Softwarepaket	Version
SIINEOS	2.7.0
Qt Creator	8.0.2
Qt	5.12.2 MinGW 64-Bit
CMake Tools	3.24.2
Timescale DB	2.0.0
PostgresSQL	12.5
NodeRed	2.2.3
Docker	20.10.21
Python Skript Container Image	arm32v7//python:3.8-slim
Grafana	9.0.1
Peakboard Designer	3.0.1.0

Tab. A.1: Softwareversionen zum Zeitpunkt der Erstellung dieser Arbeit

B InCore Programm – MQTT-Broker

```
1  import InCore.Foundation 2.6
2  import InCore.Mqtt 2.6
3
4  Application {
5      name: "mqttBroker"
6
7      MqttBroker {
8          enabled: true
9          internal: false
10     }
11 }
12
```

C InCore Programm – MQTT-Database-Interface

```
import InCore.Foundation 2.6
import InCore.Mqtt 2.6
import InCore.IO 2.6
import InCore.Database 2.6

Application {
    name: "mqttDBInterface"

    function toPSQLTS(timestamp) {
        const date = new Date(timestamp);
        const datevalues = [
            date.getFullYear(),
            date.getMonth()+1,
            date.getDate(),
            date.getHours(),
            date.getMinutes(),
            date.getSeconds(),
            date.getMilliseconds()
        ];

        let ts_month = "";
        if (datevalues[1] < 10)
            ts_month = "0" + datevalues[1];
        else
            ts_month = datevalues[1];

        let ts_day = "";
        if (datevalues[2] < 10)
            ts_day = "0" + datevalues[2];
        else
            ts_day = datevalues[2];

        let ts_hour = "";
        if (datevalues[3] < 10)
            ts_hour = "0" + datevalues[3];
        else
            ts_hour = datevalues[3];

        let ts_minute = "";
        if (datevalues[4] < 10)
            ts_minute = "0" + datevalues[4];
        else
            ts_minute = datevalues[4];

        let ts_second = "";
        if (datevalues[5] < 10)
            ts_second = "0" + datevalues[5];
        else
            ts_second = datevalues[5];

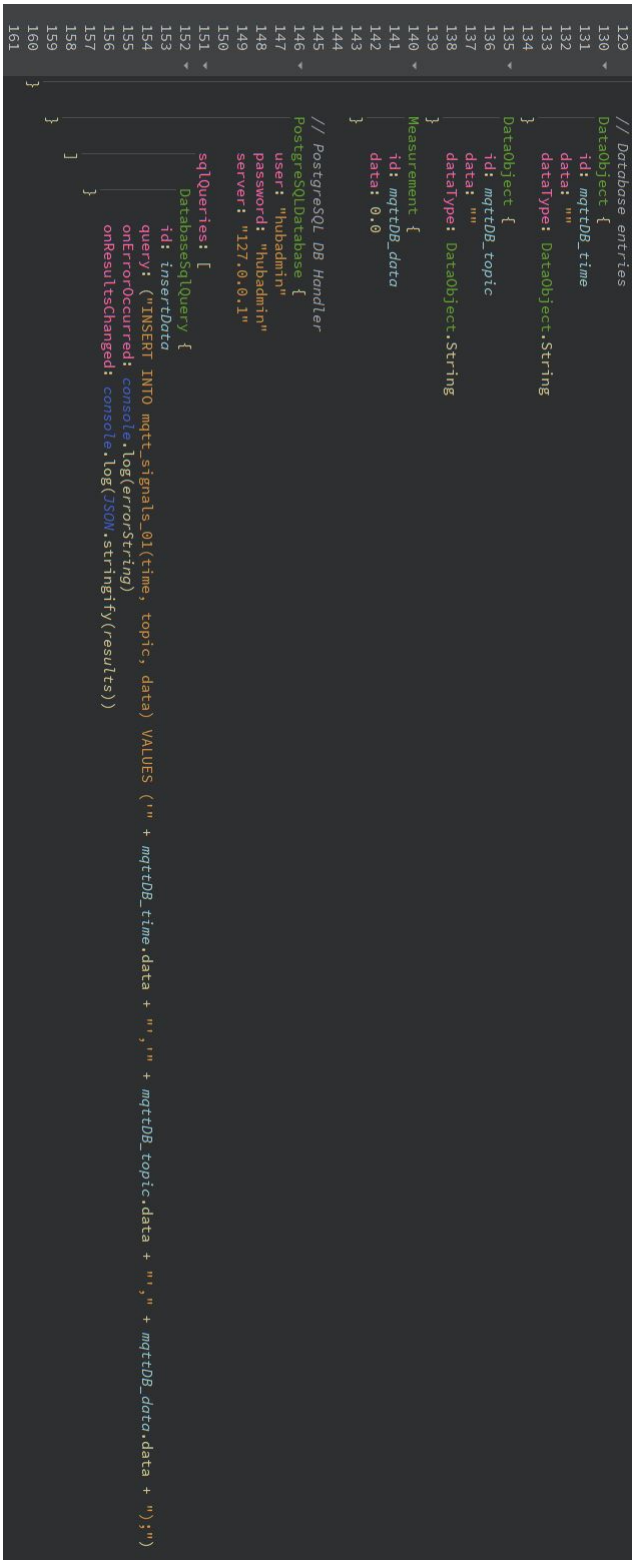
        let ts_ms = "";
        if (datevalues[6] < 10)
            ts_ms = "00" + datevalues[6];
        else if (datevalues[6] < 100)
            ts_ms = "0" + datevalues[6];
        else
            ts_ms = datevalues[6];

        return (datevalues[0] + "-" + ts_month + "-" + ts_day + " " + ts_hour + ":" + ts_minute + ":" + ts_second + "." + ts_ms + " +0000");
    }

    LED {
        id: led_green
        index: LED.StatusGreen
        value: false
    }

```

```
68      LED {
69          id: led_blue
70          index: LED.StatusBlue
71          value: false
72      }
73
74      Timer {
75          id: timer_mqttDataLED
76          interval: 500
77          repeat: false
78
79          onTriggered: led_blue.value = false;
80      }
81
82      System {
83          id: system
84          Polling on deviceTemperature { }
85      }
86
87      MqttClient {
88          hostname: "127.0.0.1"
89
90          autoConnect: true
91          clientId: "gm-100_" + Math.floor(Math.random() * 101)
92
93          MqttPublication {
94              MqttTopic { name: "test/temperature"; data: system.deviceTemperature }
95          }
96
97          MqttSubscription {
98              MqttTopic {
99                  name: "gm-100/temperature"
100                 onDataChanged: {
101                     let timestamptz = toPSQLTS(timestamp);
102
103                     // Log incoming MQTT message
104                     console.log("[MQTT] " + name + " | " + timestamptz + " | " + data);
105
106                     // Save message data to database table elements
107                     mqttDB_time.data = timestamptz;
108                     mqttDB_topic.data = name;
109                     mqttDB_data.data = data;
110                     insertData.execute();
111
112                     // Blink blue status LED to indicate ongoing traffic
113                     timer_mqttDataLED.restart();
114                     led_blue.value = true;
115                     led_green.value = true;
116                 }
117             }
118         }
119
120         onConnected: {
121             led_green.value = true
122         }
123
124         onDisconnected: {
125             led_green.value = false
126         }
127     }
128
```


// Database entries
DataObject {
id: mqttDB_time
data: ""
dataType: DataObject.String
}
DataObject {
id: mqttDB_topic
data: ""
dataType: DataObject.String
}
Measurement {
id: mqttDB_data
data: 0.0
}
// PostgreSQL DB Handler
PostgreSQLDatabase {
user: "hubadmin"
password: "hubadmin"
server: "127.0.0.1"
sqlQueries: [
DatabaseSqlQuery {
id: insertData
query: ("INSERT INTO mqtt_signals_01(time, topic, data) VALUES ('" + mqttDB_time.data + "','" + mqttDB_topic.data + "'," + mqttDB_data.data + ");")
onErrorOccurred: console.log(errorString)
onResultsChanged: console.log(JSON.stringify(results))

D Dockerfile

```
# For more information, please refer to https://aka.ms/vscode-docker-python
FROM arm32v7/python:3.8-slim

# Keeps Python from generating .pyc files in the container
ENV PYTHONDONTWRITEBYTECODE=1

# Turns off buffering for easier container logging
ENV PYTHONUNBUFFERED=1

# Install pip requirements
COPY requirements.txt .
RUN apt-get update
RUN apt-get -y install gcc
RUN apt-get -y install libpq-dev
RUN python -m pip install -r requirements.txt

WORKDIR /app
COPY . /app

# Creates a non-root user with an explicit UID and adds permission to access the /app folder
# For more info, please refer to https://aka.ms/vscode-docker-python-configure-containers
RUN adduser -u 5678 --disabled-password --gecos "" appuser && chown -R appuser /app
USER appuser

# During debugging, this entry point will be overridden. For more information, please refer t
CMD ["python", "app.py"]

```

E Python Programm

```python
import paho.mqtt.client as mqtt
import psycopg2
import xlsxwriter

import time
from datetime import datetime
import threading


# MQTT helper functions
# onconnect subscribes to needed topics
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("gm-100/temperature")

# onmessage prints message data to console
def on_message(client, userdata, msg):
    print("[MQTT] " + msg.topic + " | " + str(msg.payload, 'UTF-8'))


# TimescaleDB helper functions
# SQL query to fetch aggregate data
qry_agr_avg = """
              select * from cputemp_agr_avg_hourly
              order by hour asc, topic;
              """

# function to fetch data from SQL query
def psql_fetchData(psql):
    while True:
        cursor = psql.cursor()
        cursor.execute(qry_agr_avg)
        qry_res = cursor.fetchall()
        cursor.close()

        xlsx_writeData(qry_res)

        time.sleep(60 * 60);

```

```python

# XLSX Writer helper functions
# fnction to export data list to xlsx file
def xlsx_writeData(data):
    date_day = datetime.now().strftime("%d-%m-%y")
    date_time = datetime.now().strftime("%H-%M-%S")
    wb_name = "dataExport_" + str(date_day) + "_" + str(date_time) + ".xlsx"

    wb = xlsxwriter.Workbook(wb_name, {'remove_timezone': True})
    ws = wb.add_worksheet()

    row = 0
    col = 0

    for listElement in data:
        ws.write_datetime(row, col, listElement[0])
        ws.write_string(row, col + 1, listElement[1])
        ws.write_number(row, col + 2, listElement[2])

        row += 1

    wb.close()


def main():
    print("[INFO] Project: 01 - Python | Author: Marius Gerstner | Date: 12/01/2023 | Version: 1.0")

    # MQTT setup functions
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect("192.168.178.144", 1883, 60)
    client.loop_start()

    # TimescaleDB setup functions
    CONNECTION = "postgres://hubadmin:hubadmin@192.168.178.144:5432/postgres"
    psql = psycopg2.connect(CONNECTION)
    thread_psql = threading.Thread(target=psql_fetchData, args=(psql,))
    thread_psql.start()


if __name__ == "__main__":
    main()
```

Literatur

[BetriebsanleitungIoTGatewayHubGM1002022] : *Betriebsanleitung IoT-Gateway Hub-GM100 Version 1.3.* – URL https://www.inhub.de/files/Downloads/2022-09-22-Betriebsanleitung-HUB-GM100-V1-3-de.pdf. – Zugriffsdatum: 2023-01-15

[EMCMStationareMotorstromanalyse] : *E-MCM - Stationäre Motorstromanalyse.* – URL https://www.avibia.de/ueberwachung-cms/e-mcm-motorstromanalyse/. – Zugriffsdatum: 2023-01-15

[ElektromotorenAufbauFunktionsweise] : *Elektromotoren » Aufbau, Funktionsweise und Arten einfach erklärt.* – URL https://www.conrad.de/de/ratgeber/technik-einfach-erklaert/elektromotor.html. – Zugriffsdatum: 2023-01-15

[InCoreInstallGuide] : *InCore Install Guide — InCore 2.7.0 Documentation.* – URL https://incore.readthedocs.io/en/latest/installation/index.html. – Zugriffsdatum: 2023-01-15

[PT100ErklarungUnd] : *PT100 - Erklärung Und Temperaturüberwachung.* – URL https://www.sourcetronic.com/glossar/pt100/. – Zugriffsdatum: 2023-01-15

[SudokrewApplicationWeb] : *Sudokrew Application and Web Development.* – URL https://www.sudokrew.com/insights/relational-vs-time-series-databases. – Zugriffsdatum: 2023-01-15

[TimescaleDocumentationContinuous] : *Timescale Documentation | Continuous Aggregates.* – URL https://docs.timescale.com/timescaledb/latest/how-to-guides/continuous-aggregates/. – Zugriffsdatum: 2023-01-15

[TimescaleDocumentationData] : *Timescale Documentation | Data Retention.* – URL https://docs.timescale.com/timescaledb/latest/how-to-guides/data-retention/. – Zugriffsdatum: 2023-01-15

[TimescaleDocumentationWhy] : *Timescale Documentation | Why Use TimescaleDB over PostgreSQL?.* – URL https://docs.timescale.com/timescaledb/latest/overview/how-does-it-compare/timescaledb-vs-postgres/. – Zugriffsdatum: 2023-01-15

[WhatConditionMonitoring] : *What Is Condition Monitoring? (Everything You Need to Know).* – URL https://www.twi-global.com/technical-knowledge/faqs/what-is-condition-monitoring.aspx. – Zugriffsdatum: 2023-01-15

[ZentrumFurDigitalisierung] : *Zentrum Für Digitalisierung in Produktion Und Produktentwicklung / DHBW Ravensburg.* – URL `https://www.ravensburg.dhbw.de/forschung-transfer/kompetenzzentren/zentrum-fuer-digitalisierung-in-produktion-und-produktentwicklung#produktionsnetzwerk-fallenbrunnen`. – Zugriffsdatum: 2023-01-14

[Andreas] Andreas: *Alles was Sie über Instandhaltung wissen müssen - Definition, Aufgaben & die richtige Strategie finden.* – URL `https://forcam.com/instandhaltungsstrategien-und-instandhaltung-aufgaben/`. – Zugriffsdatum: 2023-01-14

[Bauer und Dremel] Bauer, Jörg ; Dremel, Alexander: Condition Monitoring als Schlüsseltechnologie – Eine Analyse der Anforderungen an neue Geschäftsmodelle für den Remote Service.

[Cook] Cook, Joshua: *Docker for Data Science.* – ISBN 1-4842-3011-6

[DEUTSCHES INSTITUT FÜR NORMUNG E.V.:] DEUTSCHES INSTITUT FÜR NORMUNG E.V.:: *Zustandsüberwachung Und -Diagnostik von Maschinen – Schwingungs-Zustandsüberwachung.* Beuth Verlag GmbH Berlin (1)

[of Management] Management, Panyapiwat I. of: *3.* Bd. 14: *INTERNATIONAL SCIENTIFIC JOURNAL OF ENGINEERING AND TECHNOLOGY (ISJET)*

[Vollmüller] Vollmüller, Bernd: *Welche Instandhaltungsstrategien gibt es?.* – URL `https://www.beratung-vr.de/welche-instandhaltungsstrategien-gibt-es/`. – Zugriffsdatum: 2023-01-15

[Wimmer und Garrigós] Wimmer, Manuel ; Garrigós, Irene: *Current Trends in Web Engineering.* – ISBN 978-3-319-74433-9