DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Zukunftsfabrik Bodensee: Predictive Maintenence für das Transfersystem

Studienarbeit T3_3100

Studiengang Elektrotechnik

Studienrichtung Automation

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Aaron Hahn, Louis Ranke

Abgabedatum:	10. Januar 2025
Bearbeitungszeitraum:	01.10.2024 - 31.12.2024
Matrikelnummer:	9080428, 8908532
Kurs:	TEA22
Professorin:	Herr Prof. Dr. Thomas Dietmüller

Disclaimer

Das in dieser Arbeit gewählte generische Maskulinum bezieht sich zugleich auf die männliche, weibliche und andere Geschlechteridentitäten. Zur besseren Lesbarkeit wird auf die Verwendung männlicher und weiblicher Sprachformen verzichtet. Alle Geschlechteridentitäten werden ausdrücklich mitgemeint, soweit die Aussagen dies erfordern.

Sperrvermerk

gemäß Ziffer 1.1.14 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 in der Fassung vom 24.07.2023:

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

Erklärung

gemäß Ziffer 1.1.14 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 in der Fassung vom 24.07.2023.

Ich versichere hiermit, dass ich meine Studienarbeit T3_3100 mit dem Thema:

Zukunftsfabrik Bodensee: Predictive Maintenence für das Transfersystem

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.

Friedrichshafen, den 10. Januar 2025



Aaron Hahn



Louis Ranke

Inhaltsverzeichnis

1 Einleitung **1**

2 Theoretische Grundlagen **3**

2.1 Predictive Maintenance 3

2.2 Machine Learning 4

2.2.1 Sensorik 6

2.2.1.1 TS2229 6

2.2.1.2 IFC201 7

2.2.1.3 VSA001 8

2.2.2 Intel NUC 9

2.2.3 Peakboard 9

3 Systemerweiterung und -wiederherstellung **11**

3.1 Sensoren 11

3.1.1 Installation Temperatursensor 12

3.1.2 Installation Induktiver Sensor 13

3.1.3 Parametrierung Sensoren 14

3.2 Intel Next Unit of Computing (NUC) 14

3.2.1 Neuaufsetzung des NUC und Installation der IoT-Dienste 15

3.2.2 Anbindung des NUC an das Gesamtsystem 17

4 Konzeptentwicklung für Predictive Maintenance **18**

4.1 Anforderungen und Ziele eines Predictive Maintenance Systems 18

4.2 Anomalieerkennung 19

4.2.1 Isolation Forest 20

4.2.2 Autoencoder 22

4.3 Visualisierung und Entscheidungsunterstützung 23
4.3.1 Rolle der Visualisierung 23
4.3.2 Peak Board als Tools zur Datenvisualisierung 24
4.3.3 Design eines Dashboards 25

5 Fazit **27**

Literaturverzeichnis **29**

Abbildungsverzeichnis **32**

Abkürzungsverzeichnis **33**

A Code Autoencoder **34**

B IoT-Stack Installationscode **37**

Kurzfassung

Abstract

1 Einleitung

In einer zunehmend digitalisierten und automatisierten Produktionslandschaft ist die vorausschauende Instandhaltung von Maschinen und Anlagen zu einem entscheidenden Erfolgsfaktor geworden. Predictive Maintenance (PdM), also die Vorhersage und Vermeidung von Ausfällen durch kontinuierliche Zustandsüberwachung und datenbasierte Prognosen, ermöglicht es, Wartungsintervalle optimal zu planen und ungeplante Stillstände zu vermeiden. Besonders in stark vernetzten Umgebungen, in denen die Verzögerung oder der Ausfall eines einzelnen Maschinenabschnitts Produktionsketten erheblich beeinträchtigen kann, spielt Predictive Maintenance eine zentrale Rolle. Die Implementierung einer effizienten Wartungsstrategie wird umso wichtiger, wenn hohe Verfügbarkeit, Prozesssicherheit und Kosteneffizienz gefordert sind. In den letzten Jahren haben sich durch Fortschritte in den Bereichen Internet of Things (IoT), Big Data und maschinelles Lernen neue Möglichkeiten zur Umsetzung solcher Systeme eröffnet.

Industrieunternehmen sehen sich jedoch immer wieder Herausforderungen gegenüber, wenn es um die Implementierung und Integration solcher prädiktiven Instandhaltungssysteme in ihre Produktionsprozesse geht. Neben der Auswahl geeigneter Sensoren für die Datenaufnahme und einer stabilen Infrastruktur zur Verarbeitung und Speicherung der Messdaten erfordert Predictive Maintenance auch ein tiefes Verständnis der Maschinenphysik, um aussagekräftige Kennzahlen zu definieren. Gleichzeitig ist die hohe Datenmenge, die durch kontinuierliche Sensoraufnahmen erzeugt wird, nur nutzbar, wenn die Daten zuverlässig verarbeitet und analysiert werden können. Die Qualität der erfassten Daten spielt ebenfalls eine zentrale Rolle, da ungenaue oder unvollständige Daten zu falschen Vorhersagen und damit zu unnötigen Wartungen oder sogar ungeplanten Stillständen führen können.

Im Rahmen dieser Arbeit soll ein Predictive Maintenance System für ein förderbandbasiertes Transportsystem entwickelt werden, das mithilfe von Sensorik, einer IoT-Architektur und modernen Datenanalysemethoden frühzeitig Anzeichen für potenzielle Störungen erkennen kann. Die Wahl eines förderbandgestützten Systems als Fallbeispiel bietet sich an, da ein Ausfall der Förderbänder aufgrund ihrer zentralen Rolle im Produktionsprozess oft zu umfassenden Produktionsverzögerungen führt. Ziel dieser Arbeit ist es, ein System zu entwickeln, das verschiedene Betriebsparameter wie Temperatur, Vibration und Drehzahl kontinuierlich überwacht und basierend auf den gewonnenen Daten Muster und Anomalien identifiziert, die auf zukünftige Defekte hinweisen könnten.

Durch die erfolgreiche Entwicklung und Implementierung des Predictive Maintenance Systems sollen mehrere zentrale Forschungsfragen beantwortet werden. Wie kann ein förderbandgestütztes System so ausgestattet werden, dass typische mechanische und betriebsbedingte Defekte frühzeitig erkannt werden? Welche technischen und organisatorischen Voraussetzungen müssen geschaffen werden, um Sensordaten stabil und zuverlässig zu erfassen und für die Analyse verfügbar zu machen? Und welche Analyse- und Visualisierungsmethoden erweisen sich als besonders geeignet, um Produktionsverantwortliche proaktiv auf Störungen hinzuweisen?

Insgesamt soll die Arbeit einen praxisnahen Einblick in die Entwicklung und Umsetzung eines Predictive Maintenance Systems geben und einen Beitrag zum Verständnis der technischen und operativen Anforderungen leisten, die für die Einführung solcher Systeme notwendig sind. Das Projekt zielt darauf ab, nicht nur eine prädiktive Wartungslösung für das Transportsystem bereitzustellen, sondern auch Erkenntnisse über die Anwendbarkeit und die Herausforderungen von Predictive Maintenance in industriellen Umgebungen zu gewinnen. Diese Arbeit stellt somit einen wichtigen Schritt in Richtung einer intelligenten, vorausschauenden Produktionsumgebung dar, die sowohl Effizienz als auch Zuverlässigkeit erhöht und gleichzeitig die Kosten durch optimierte Wartungsprozesse senkt.

2 Theoretische Grundlagen

2.1 Predictive Maintenance

Im Folgenden werden die theoretischen Grundlagen der Predictive Maintenance (zu dt. vorausschauende Instandhaltung) behandelt. Worum handelt es sich jedoch bei PdM? SAP definiert Predictive Maintenance als „Ansatz der Instandhaltung, bei dem der Schwerpunkt darauf liegt, die Signale Ihrer Unternehmensanlage zu deuten.“ [SE24] Dadurch kann ein „Unternehmen Anlagenprobleme antizipieren und Wartungsarbeiten planen, wenn und wo sie unmittelbar anstehen.“ [SE24] Somit verspricht PdM erhebliche Vorteile in Bezug auf Effizienz und Kostenersparnis, im Gegensatz zur reaktiven Wartung, welche historisch eingesetzt wurde. PdM erlaubt die kontinuierliche Überwachung und vorausschauende Analyse des Maschinenzustands. Dies verhindert bzw. verringert unnötige Wartungen und ungeplante Ausfallzeiten. [KHV06]

Technologisch stützt sich die PdM dabei auf Weiterentwicklungen der Industrie 4.0, insbesondere auf das IoT sowie auf datenbasierte Künstliche Intelligenz (KI) und Large Language Models (LLM).

Die Grundvoraussetzung für die PdM ist eine zuverlässige Datenerfassung. Dabei erfassen Sensoren kontinuierlich Betriebsdaten wie Temperatur, Vibrationen oder Feuchtigkeit, die über IoT-Systeme an zentrale Datenbanken übertragen werden. [LBK15] Diese Daten werden analysiert, um Muster zu identifizieren, die auf potenzielle Ausfälle hinweisen können. Hierbei zeigen sich vor allem Machine-Learning-Algorithmen als effektiv, um aus großen Datenmengen relevante Informationen herauszufiltern und so die

verbleibende Lebensdauer von Maschinenkomponenten vorherzusagen. [Mob02]

Eine weitere Komponente der PdM ist die Anomalieerkennung. Diese wird durch Deep-Learning-Algorithmen unterstützt. Diese Algorithmen lernen aus historischen Daten und erkennen Abweichungen von normalen Betriebszuständen der Maschinen, die auf mögliche Störungen hinweisen. [Luo+20] Durch die Nutzung neuronaler Netze, konkret Deep Neural Networks (DNN), können PdM-Modelle auf komplexe industrielle Systeme skaliert werden.

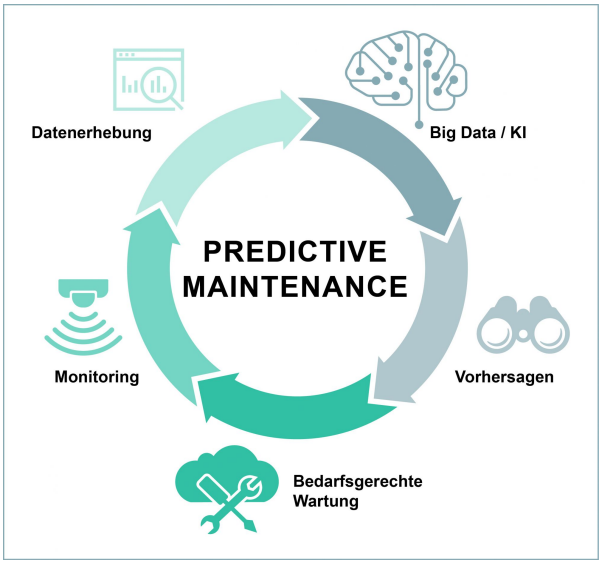


Abbildung 2.1: Grafik zur Predictive Maintenance. Quelle: All-Electronics.de

2.2 Machine Learning

Um das gewünschte Ziel einer Predictive Maintenance umsetzen zu können, benötigt es Modelle, die Maschinendaten analysieren und interpretieren. Dafür empfehlen sich Machine-Learning-Modelle. Folgend wird eine Einführung in das Machine Learning gegeben.

Grundsätzlich ist Machine Learning ein Teilgebiet der KI. Es zielt darauf ab, Systeme zu entwickeln, die aus Daten lernen und Entscheidungen treffen können. Dies geschieht dabei ohne eine explizite Programmierung. Es kommen Algorithmen und statistische

Modelle zum Einsatz, die auf der Verarbeitung und Analyse von Daten basieren. Machine Learning ermöglicht es Computern, „auf Basis von Daten zu lernen, Muster zu erkennen und Vorhersagen zu treffen“. [Bur19] Dabei besteht die Zielsetzung, manuelle Eingriffe zu minimieren und durch automatische Lernprozesse effizientere Lösungen zu entwickeln. Beispiele dafür sind unter anderem die Bilderkennung, Sprachverarbeitung und vorausschauende Analytik.

Machine Learning kann dabei in Hauptkategorien unterteilt werden. Diese sind das überwachte Lernen (Supervised Learning), das unüberwachte Lernen (Unsupervised Learning) und das bestärkende Lernen (Reinforcement Learning). Im überwachten Lernen werden Algorithmen mit einem beschrifteten Datensatz trainiert. Beispielhaft ist die Klassifikation von E-Mails in „Spam“ und „Nicht-Spam“. Jene Algorithmen lernen aus den Eingabedaten (Features) und den entsprechenden Ausgaben (Labels). Diese dienen dazu, neue Daten korrekt zu kategorisieren. [Mül17]

Das Gegenstück dazu ist das unüberwachte Lernen. Hierbei analysiert das System, im Gegensatz zum überwachten Lernen, unbeschriftete Daten, um Muster oder Gruppierungen zu identifizieren. Ein Beispiel dafür kann die Clusterbildung bei Kundendaten sein. Diese Methoden eignen sich besonders für Datenanalysen, bei denen keine vorab bekannten Kategorien existieren. [Gér23]

Letztlich ist noch die Methode des bestärkenden Lernens anzuführen. Es nutzt Belohnungen und Strafen, um sogenannte Agenten zu trainieren, die in dynamischen Umgebungen optimale Entscheidungen treffen, etwa bei der Steuerung von Robotern oder der Strategieentwicklung bei Spielen wie Schach. [Sut20]

Machine-Learning-Algorithmen basieren dabei grundsätzlich auf den mathematischen Konzepten der linearen Algebra, der Statistik, inklusive Wahrscheinlichkeitsrechnung, und der Optimierung. Laut Géron ist die Fähigkeit, „Grundlagen der Mathematik mit modernster Technologie zu kombinieren“, die Grundlage für die Entwicklung effektiver Machine-Learning-Modelle. [Gér23]

Zur Anwendung in der Praxis werden Programmiersprachen wie Python mit Bibliotheken wie „scikit-learn“, „TensorFlow“ und „Keras“ eingesetzt. Diese Tools ermöglichen die

Implementierung und Analyse von Machine-Learning-Algorithmen. [Mül17]

2.2.1 Sensorik

Zur Zustandsüberwachung der Elektromotoren des Transportsystems werden pro Motor bis zu drei verschiedene Sensoren verwendet. Die Sensoren dienen der Überwachung der Temperatur, der Drehzahl und der Vibration. Um dies zu realisieren, wurden diese Sensoren von der Firma ifm bereitgestellt. Die verschiedenen Sensoren werden im Folgenden grundlegend erläutert.

2.2.1.1 TS2229

Für die Überwachung der Temperatur wird der ifm-Temperatursensor TS2229 eingesetzt. Der TS2229 ist ein Kontaktsensor, der zur Messung von Temperaturen im Bereich von -40 °C bis 90 °C konzipiert ist. Das Messelement des Sensors ist ein Pt100-Widerstand, der nach DIN EN 60751, Klasse A, zertifiziert ist. Der Sensor ist als Anlegefühler für feste Körper ausgelegt und verfügt über spezielle vergoldete Kontakte, die seine Eignung für spezifische Industrieanwendungen weiter erhöhen.

Elektrisch zeichnet sich der TS2229 durch eine hohe Isolationsfestigkeit von 2500 V zwischen Sensorgehäuse und Innenaufbau aus, was eine sichere Anwendung auch unter anspruchsvollen Bedingungen ermöglicht. Der Sensor entspricht der Schutzklasse III, was ihn für den Einsatz in Niederspannungssystemen geeignet macht. In Bezug auf seine Messgenauigkeit erreicht der Sensor eine Toleranz von $\pm$ (0,15 K + 0,002 $\times$ |t|), was ihn für präzise Anwendungen qualifiziert. Die Ansprechzeiten betragen T05 = 9 Sekunden und T09 = 15 Sekunden bei der Verwendung auf einer planen Aluminiumfläche mit Wärmeleitpaste; in bewegtem Wasser (bei $<$ 0,1 m/s) liegen die Zeiten bei 10 und 24 Sekunden, was auf eine schnelle Reaktionsfähigkeit hinweist.

Der TS2229 ist auf Umgebungsbedingungen gemäß Schutzart IP67 ausgelegt, die Schutz vor Staub und zeitweiligem Eintauchen in Wasser bieten. Die erwartete Mean Time to

Failure (MTTF) des Sensors liegt bei 11.415,5 Jahren. Mechanisch hat der Sensor eine quaderförmige Bauform mit den Abmessungen 12 × 8,7 × 51 mm und einem Gewicht von 88,5 g. Sein Gehäuse besteht aus Edelstahl (1.4303), ergänzt durch vernickelte Kupferelemente, die für zusätzliche Robustheit sorgen.

Für den elektrischen Anschluss ist der Sensor mit einem 2 Meter langen PUR-Kabel ausgestattet, das einen Durchmesser von 4,9 mm aufweist. Der Anschluss erfolgt über einen M12-Stecker mit A-Codierung und vergoldeten Kontakten, was die Signalübertragung verbessert und die Korrosionsbeständigkeit erhöht. [gmb16b]

2.2.1.2 IFC201

Für die Überwachung der Drehzahl des Elektromotors wird ein induktiver Sensor eingesetzt, genauer gesagt der ifm-Sensor IFC201. Der induktive Sensor IFC201 arbeitet auf Basis einer PNP/NPN-Schaltung und fungiert als Schließer mit einem Schaltabstand von 7 mm. Der reale Schaltabstand beträgt ebenfalls 7 mm, mit einer Toleranz von ±10 %, und der Arbeitsbereich erstreckt sich bis 5,7 mm, was ihn besonders für Anwendungen mit erhöhtem Schaltabstand prädestiniert.

Elektrisch ist der IFC201 für einen Spannungsbereich von 10 bis 30 V DC ausgelegt und entspricht der Schutzklasse II. Der Sensor verfügt über einen Verpolungsschutz, ist überlastfest und gegen Kurzschluss gesichert. Der Schaltausgang hat eine dauerhafte Strombelastbarkeit von bis zu 100 mA, mit einem maximalen Spannungsabfall von 2,5 V und einer Schaltfrequenz von 700 Hz, was eine schnelle Signalverarbeitung gewährleistet.

Die Genauigkeit des Sensors variiert abhängig vom Zielmaterial; der Korrekturfaktor beträgt beispielsweise 0,7 für Edelstahl, was im gewünschten Anwendungsfall relevant ist. Die Hysterese liegt bei 3 bis 15 % des Schaltabstands, und der Schaltpunktdrift bewegt sich im Bereich von -10 bis +10 % des Schaltabstands.

Der Sensor ist in einem kompakten, gewindeförmigen Messinggehäuse (weißbronze be-

schichtet) mit einer Gesamtlänge von 50 mm und einem Durchmesser von M12 × 1 untergebracht. Die aktive Fläche besteht aus Liquid Crystal Polymer, was eine robuste Materialwahl für anspruchsvolle Einsatzbedingungen darstellt. Der Sensor entspricht der Schutzart IP68, was ihn gegen Staub und eindringendes Wasser schützt, und ist für Umgebungstemperaturen von -25 °C bis 70 °C geeignet. Die MTTF beträgt 2487 Jahre, was eine hohe Zuverlässigkeit über lange Einsatzzeiträume hinweg signalisiert. Der elektrische Anschluss erfolgt über einen M12-Stecker mit A-Codierung und vergoldeten Kontakten. [gmb16a]

2.2.1.3 VSA001

Zur Messung der Schwingungen der Elektromotoren wird der ifm-Beschleunigungssensor VSA001 eingesetzt. Der VSA001 besitzt einen Messbereich von -25 g bis 25 g und einen Frequenzbereich von 1 bis 6000 Hz, was ihn optimal zur Überwachung von Vibrationen macht. Der Sensor arbeitet auf Basis eines kapazitiven Messprinzips in Form eines mikroelektromechanischen Systems und ist zur Verbindung mit externer Diagnoseelektronik, wie der VSE-Serie, ausgelegt, um Schwingungen zuverlässig zu analysieren.

Elektrisch erfordert der VSA001 eine Betriebsspannung zwischen 7,2 und 10,8 V DC und weist eine geringe Stromaufnahme von unter 15 mA auf. Der Isolationswiderstand beträgt 100 MΩ bei 500 V DC. Der Sensor ist verpolungsgeschützt und entspricht der Schutzklasse III. Der analoge Stromausgang reicht von 0 bis 10 mA mit einer maximalen Bürde von 300 MΩ, wobei eine Empfindlichkeit von 142 $\frac{\mu A}{g}$ angegeben ist.

Mechanisch besitzt der Sensor eine zylindrische Form mit einem Durchmesser von 17 mm und einer Länge von 53 mm, gefertigt aus Edelstahl. Die Montage erfolgt über ein M8-Gewinde, und das Gerät ist gegen mechanische Überlastungen bis zu 500 g geschützt. Die Schutzarten IP67, IP68 und IP69K machen den Sensor widerstandsfähig gegen Staub, Wasser und Hochdruckreinigung, während der Temperaturbereich von -30 °C bis 125 °C seine Einsatzfähigkeit auch unter extremen Bedingungen gewährleistet.

Der VSA001 hat eine MTTF von 1701 Jahren. Der Sensor ist EMV-zertifiziert gemäß

EN 61000-6 und entspricht den Standards für Schock- und Vibrationsfestigkeit nach DIN EN 60068. Der elektrische Anschluss erfolgt über einen M12-Stecker mit A-Codierung, und das Gerät unterstützt Kabellängen bis zu 250 m. [gmb23]

2.2.2 Intel NUC

Der Intel NUC ist eine kompakte Mini-PC-Serie, die 2013 von Intel eingeführt wurde. Die Geräte zeichnen sich durch ihre kleine Bauform, Energieeffizienz und vielseitige Einsatzmöglichkeiten aus. Im Laufe der Jahre wurden verschiedene Prozessorgenerationen, von Sandy Bridge bis hin zu Raptor Lake, integriert, was die NUCs für Anwendungen von einfachen Office-Aufgaben bis hin zu anspruchsvollen Multimedia- und Gaming-Anwendungen geeignet macht. [asu24]

Im Juli 2023 gab Intel die Einstellung der NUC-Produktion bekannt, übertrug jedoch Entwicklung und Vertrieb an ASUS, die die Serie unter eigener Lizenz fortsetzen. Die neueste Generation, darunter der NUC 14 Pro und ein Gaming-orientierter ROG NUC, wurde Anfang 2024 vorgestellt und verdeutlicht die anhaltende Bedeutung dieser Geräteklasse im Bereich der kompakten Computerlösungen. [asu24]

In der Zukunftsfabrik Bodensee wird der Intel NUC als Server eingesetzt, um Sensordaten innerhalb des Netzwerks auszulesen, zu verarbeiten und zu visualisieren. Dabei fungiert der NUC auch als Schnittstelle zur Visualisierungssoftware Peakboard, die die aufbereiteten Daten schließlich über ein Dashboard auf einem Bildschirm darstellt.

2.2.3 Peakboard

Peakboard ist eine Low-Code-Plattform, die es Unternehmen ermöglicht, individuelle industrielle Anwendungen zur Digitalisierung und Optimierung von Produktions- und Logistikprozessen zu erstellen. Mit der benutzerfreundlichen Software können interaktive Dashboards gestaltet werden, die Daten aus verschiedenen Quellen wie SAP, Maschinen-

steuerungen oder Excel in Echtzeit erfassen, verarbeiten und visualisieren.

Der Peakboard Designer, eine Desktop-Anwendung für Windows, dient als zentrales Tool zur Erstellung dieser Anwendungen. Durch Drag-and-Drop-Funktionalitäten können Nutzer ohne tiefgehende Programmierkenntnisse komplexe Logiken und interaktive Elemente implementieren. Die erstellten Dashboards werden anschließend über die Peakboard Box oder das Bring Your Own Device (BYOD)-Modell auf Anzeigegeräten dargestellt, um Prozesse und Kennzahlen in Echtzeit transparent zu machen.

Peakboard bietet zudem zahlreiche Schnittstellen zu über 100 Datenquellen, was eine flexible Integration in bestehende IT-Landschaften ermöglicht. Dies unterstützt Unternehmen dabei, ihre Produktions- und Logistikprozesse effizient zu digitalisieren und zu steuern. [Gmb24]

In der Zukunftsfabrik Bodensee findet die Softwarelösung PeakBoard Anwendung zur Visualisierung produktionsrelevanter Daten im Kontext der Zustandsüberwachung. Die generierten Dashboards werden auf einem großformatigen Bildschirm dargestellt, um eine übersichtliche und klare Darstellung der relevanten Informationen zu gewährleisten.

3 Systemerweiterung und -wiederherstellung

Die Entwicklung eines Systems zur Predictive Maintenance erfordert zunächst die Auswahl eines geeigneten Zielsystems für die Überwachung. Im vorliegenden Fall wurde das Transportsystem der Zukunftsfabrik Bodensee ausgewählt, insbesondere die Motoren, die das Fördersystem antreiben. Dieses System wird durch vier Motoren betrieben, die jeweils ein Transportband bewegen. Zur Implementierung prädiktiver Wartung ist es notwendig, die Motoren mit Sensoren auszustatten, welche kontinuierlich den Zustand der Komponenten erfassen und Betriebsdaten aufzeichnen.

3.1 Sensoren

Im Rahmen dieser Untersuchung kommen Sensoren zum Einsatz, um in Echtzeit spezifische Parameter wie Temperatur und Vibration zu überwachen und Abweichungen vom Normalbetrieb frühzeitig zu detektieren. Die erfassten Daten ermöglichen eine präzise Analyse der Motorenleistung und unterstützen die Vorhersage potenzieller Ausfälle. Die folgenden Abschnitte beschreiben detailliert die Installation der Sensoren an den Motoren, um eine zuverlässige und kontinuierliche Überwachung des Systemzustands sicherzustellen.

In einer vorausgegangenen Arbeit wurde die Anbringung der Sensoren bereits untersucht, jedoch beschränkte sich diese Implementierung auf einen einzelnen Motor. Für eine um-

fassende Systemüberwachung müssen daher nun die restlichen Motoren ebenfalls mit entsprechenden Sensoren ausgestattet werden.

3.1.1 Installation Temperatursensor

Der Temperatursensor wurde an der wärmsten Stelle des Motorgehäuses unterhalb des Typenschilds angebracht, wo die Kühlrippen nur eingeschränkt hinterlüftet sind und sich somit eine höhere Temperatur bildet. Diese Platzierung wurde gezielt gewählt, da sie eine optimale Erfassung der Temperaturentwicklung ermöglicht, die bei erhöhter Belastung oder Reibung besonders schnell auftritt. Die gewählte Position stellt sicher, dass eine kritische Temperatur bei ungewöhnlichen Betriebsbedingungen, wie etwa internem Verschleiß oder Lagerproblemen, frühzeitig erfasst wird. So dient der Temperatursensor als verlässlicher Indikator zur Erkennung potenzieller Störungen und trägt wesentlich zur vorbeugenden Wartung bei. Der Sensor wurde am Typenschild befestigt, indem ein kleines Loch in eine unbeschriftete Stelle des Typenschilds gebohrt wurde. Anschließend wurde hinter das Typenschild eine Mutter geführt, um den Sensor wie in Abbildung 3.1 festzuschrauben.



Abbildung 3.1: Anbringung des Temperatursensor an das Typenschild; eigene Darstellung

3.1.2 Installation Induktiver Sensor

Der induktive Sensor zur Messung der Motordrehzahl wurde am Gehäuse des Lüfterrads am Ende der Motorwelle montiert. Diese Position ermöglicht eine direkte, kontaktlose Erfassung der Drehzahl, da das Lüfterrad sich synchron zur Motorwelle bewegt. Um die Erfassung sicherzustellen, musste ein Loch in das Gehäuse des Lüfterrads gebohrt sowie eine kleine Metallschraube am Lüfterrad befestigt werden (siehe Abbildung 3.2), die vom Sensor erfasst wird und ein verlässliches Signal für die Drehzahlmessung liefert. Zur Befestigung des Sensors wurde ein passend geformtes Metallblech am Gehäuse des Lüfterrads montiert. Ein zentrales Loch im Blech ermöglicht die Fixierung des Sensors mit Muttern auf beiden Seiten, sodass dieser präzise durch die Öffnung im Lüftergehäuse positioniert ist. Der angebrachte induktive Sensor ist in Abbildung 3.3 zu sehen. Durch diese Anbringung können Drehzahlabweichungen, die auf erhöhten Widerstand oder Reibung im System hindeuten, frühzeitig identifiziert und behoben werden, was zur Stabilität des gesamten Antriebssystems beiträgt.

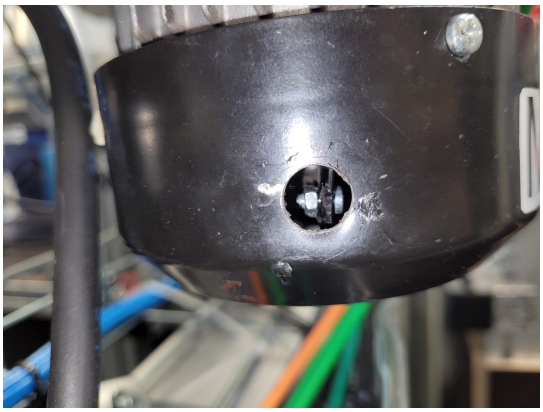


Abbildung 3.2: Loch im Lüftergehäuse sowie Schraube am Lüfter, eigene Darstellung

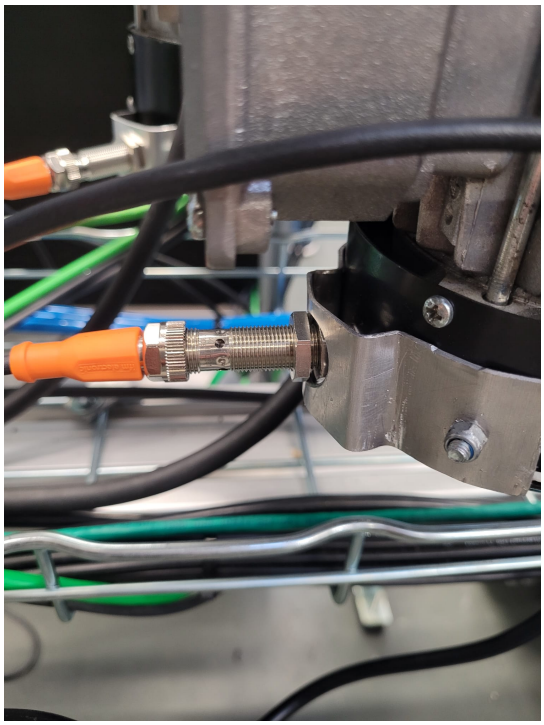


Abbildung 3.3: Induktiver Sensor am Lüftergehäuse; eigene Dartstellung

3.1.3 Parametrierung Sensoren

3.2 Intel NUC

Zur Erfassung und Verarbeitung der Sensordaten wird ein externer Mini-PC verwendet, konkret ein Intel NUC 7PJYHN mit einem Celeron J5005-Prozessor. Da sich dieser jedoch nicht mehr in der unmittelbaren Nähe der speicherprogrammierbare Steuerung (SPS) befand, musste er zunächst erneut angeschlossen und in Betrieb genommen werden.

Während dieses Prozesses wurde festgestellt, dass die installierte Proxmox-Software nicht wie gewohnt startete und wiederholt Fehlermeldungen auftrat. Eine Analyse ergab, dass es sich hierbei nicht um einen einzelnen Fehler handelte, sondern dass verschiedene Fehlermeldungen angezeigt wurden. Diese deuteten darauf hin, dass entweder das Speichermedium selbst oder die Verbindung zwischen dem Intel NUC und dem Speichermedium defekt war.

Da der Intel NUC eine zentrale Komponente des Systems darstellt, war es erforderlich, diesen inklusive Speicherkarte und Arbeitsspeicher neu zu beschaffen, um das System vollständig neu aufzusetzen. Bei der Auswahl der neuen Komponenten wurde darauf geachtet, möglichst eng an das Konzept des Vorgängerprojekts anzuknüpfen, um eine kohärente Verbindung zwischen den Projekten zu gewährleisten. Diese Vorgehensweise soll die zukünftige Arbeit sowie das Verständnis der Systemstruktur erleichtern.

Es wurde ein identisches Modell des Intel NUC bestellt, ergänzt durch eine SSD mit einer Kapazität von einem Terabyte und zwei Arbeitsspeichermodule zu je acht Gigabyte.

3.2.1 Neuaufsetzung des NUC und Installation der IoT-Dienste

Nach dem erfolgreichen Einbau des SSD-Speichers und des Arbeitsspeichers in den Intel NUC wurde die Installation der Virtualisierungsplattform Proxmox begonnen. Hierfür wurde die Proxmox-ISO-Datei über einen USB-Stick gebootet. Während der Installation wurden die grundlegenden Netzwerkeinstellungen sowie der Hostname konfiguriert. Die zugewiesenen Parameter lauteten:

- **Hostname:** proxmox.local
- **IP-Adresse (CIDR):** 192.168.0.90/24
- **Gateway:** 192.168.0.2
- **DNS-Server:** 8.8.8.8

Zusätzlich wurden ein Benutzername und ein Passwort für den Zugriff auf Proxmox definiert. Diese Zugangsdaten lauten:

- **Benutzername:** root
- **Passwort:** password

Nach Abschluss der Installation wurde der NUC per LAN mit einem Rechner verbunden. Das Webinterface von Proxmox konnte anschließend über die URL https://192.168.0.90:8006/ aufgerufen werden. Hierfür war es erforderlich, den verbundenen Rechner in das gleiche Subnetz zu bringen, indem die Netzwerkeinstellungen entsprechend angepasst wurden.

Im Webinterface von Proxmox wurde ein neuer Container erstellt, der für die Anwendung im Bereich Predictive Maintenance vorgesehen ist. Für diesen Schritt war ein temporärer Internetzugang erforderlich, da sowohl ein Betriebssystem-Template als auch die benötigten Dienste heruntergeladen werden mussten.

Um den NUC mit dem Internet zu verbinden, wurde ein Rechner verwendet, der über WLAN Zugang zum Internet hatte. Dieser Rechner teilte die Internetverbindung über LAN mit dem NUC. Durch diese Methode wird dem internetfähigen Rechner eine temporäre IP-Adresse zugewiesen. Um den Internetzugang über den NUC für Proxmox zu

ermöglichen, war es erforderlich, dem NUC über die Proxmox-Shell ebenfalls temporär eine IP-Adresse innerhalb desselben Netzwerks zuzuweisen. Nach Herstellung der Internetverbindung wurde der Container erstellt. Die Zugangsdaten für den Container entsprechen den oben gennanten von Proxmox.

Im Anschluss daran wurden die benötigten Dienste installiert, die bereits in der Vorgängerarbeit definiert wurden. Im ersten Schritt wurde Docker über die Shell des Containers installiert. Dazu wurden folgende Befehle in der angegebenen Reihenfolge ausgeführt:

1. apt update && apt upgrade -y
2. apt install curl -y
3. curl -fsSL https://get.docker.com -o get-docker.sh
4. sh get-docker.sh
5. mkdir -p /.docker/cli-plugins/
6. curl -SL https://github.com/docker/compose/releases/download/v2.18.1/docker-compose-linux-x86_64 -o /.docker/cli-plugins/docker-compose
7. chmod +x /.docker/cli-plugins/docker-compose

Nach der Installation von Docker wurde "Portaineräls Verwaltungsoberfläche installiert. Dies erfolgte über folgende Befehle:

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

Portainer ermöglicht die einfache grafische Verwaltung der im Container laufenden Dienste. Das Interface von Portainer ist unter der IP-Adresse des Containers mit dem Port 9000 erreichbar (z. B. http://192.168.0.91:9000).

Im Portainer-Webinterface wurde ein neuer Stack unter dem Reiter SStacksërstellt. Hier wurde der in der Vorgängerarbeit beschriebene Code (siehe Anhang 5B) eingefügt, um die für Predictive Maintenance erforderlichen Dienste zu installieren. Nach der erfolgrei-

chen Installation wurden die Dienste im Portainer-Webinterface unter dem Namen des erstellten Stacks angezeigt. Diese Dienste konnten anschließend über die während der Installation definierten Ports aufgerufen werden. Durch diese Konfiguration wurde eine funktionsfähige Umgebung für Predictive Maintenance geschaffen, die sowohl die notwendigen Software-Dienste als auch eine effiziente Verwaltung über Portainer bietet.

3.2.2 Anbindung des NUC an das Gesamtsystem

Um den NUC in das Gesamtsystem zu integrieren, ist es erforderlich, ihn in das gleiche Netzwerk zu implementieren. Dies wird idealerweise durch die Platzierung des Geräts im Schaltschrank und die Verbindung mit dem Netzwerk der SPS mittels eines LAN-Kabels realisiert. Das LAN-Kabel wird dabei in einen der verfügbaren LAN-Ports der SPS eingesteckt. Mit der werkseitig voreingestellten IP-Adresse wird der Mini-PC automatisch in das Netzwerk eingebunden und kann anschließend für die benötigte Datenintegration konfiguriert werden. Aus zeitlichen Gründen wird die Umsetzung dieses Schrittes in den zweiten Teil der Arbeit verlegt.

4 Konzeptentwicklung für Predictive Maintenance

4.1 Anforderungen und Ziele eines Predictive Maintenance Systems

Bevor ein System zur Predictive Maintenance entwickelt und implementiert wird, sollten die Zielsetzungen und Anforderungen des Systems klar definiert werden. Laut dem Forschungsprojekt WerkPrima lassen sich vier zentrale Zieldimensionen identifizieren, auf die PdM Einfluss nehmen soll: Wirtschaftlichkeit, Zeit, Qualität und strategische Zielsetzungen im Kontext von Produktions- oder Unternehmensstrategien. Das übergeordnete Ziel der PdM besteht darin, positive Effekte in allen vier Dimensionen zu erzielen. [Sch22]

Im Rahmen unseres Anwendungsfalls, der Zukunftsfabrik Bodensee, können diese Ziele ebenso effektiv umgesetzt werden wie in großflächigen Produktionsstätten, da die Größe der Anlage keinen maßgeblichen Einfluss auf die Anwendbarkeit der Methode hat. Durch die frühzeitige Erkennung potenzieller Maschinendefekte können gezielte Maßnahmen rechtzeitig ergriffen werden. Dies führt zu erheblichen Zeitgewinnen, da beispielsweise zyklische Wartungen reduziert oder entfallen können. Diese Zeitersparnis hat bereits einen positiven Effekt auf die Wirtschaftlichkeit, da Produktionsunterbrechungen minimiert werden und die Anlagen eine höhere Auslastung erreichen. Zudem verbessert sich die Produktqualität, da die Maschinen kontinuierlich in einem optimalen Betriebszustand gehalten werden können.

Die Anforderungen an Predictive Maintenance umfassen im Allgemeinen mehrere Schritte: Zunächst müssen kritische Anlagen identifiziert und priorisiert werden, um die Auswirkungen von Ausfällen auf den Betrieb zu bewerten. Es folgt eine Investition in Technologien zur Datenerfassung und -analyse, etwa durch IoT-Sensoren und prädiktive Software. Anschließend werden Vorhersagemodelle entwickelt, häufig unter Einsatz von KI und maschinellem Lernen, um Ausfälle frühzeitig zu erkennen. Die Integration in bestehende Prozesse erfordert neue Arbeitsabläufe und Schulungen für das Wartungspersonal. Schließlich ist eine kontinuierliche Überwachung und Anpassung der Modelle notwendig, um die Effizienz langfristig zu sichern. [Kel23]

In der Zukunftsfabrik Bodensee wird zunächst das Förderbandsystem durch den Einsatz von Predictive Maintenance überwacht. Das Förderbandsystem stellt eine kritische Komponente dar, da ein Ausfall dieses Systems die gesamte Produktion zum Stillstand bringen würde, was ebenfalls alle anderen Maschinen in der Produktionskette betreffen würde. Die erforderlichen Investitionen umfassen hierbei die Anschaffung geeigneter Sensoren zur Überwachung der Motoren, eines Mini-PCs zur Datenverarbeitung sowie eines Peakboards zur Visualisierung der erfassten Daten. Der nächste Schritt besteht in der Entwicklung von Vorhersagemodellen, welche den Kern der vorliegenden Arbeit darstellen.

4.2 Anomalieerkennung

Um im Normalbetrieb des Bandsystems mögliche Anzeichen für Störungen im Zuge der Predictive Maintenance zu erkennen, benötigt es Daten, die vom Normalzustand abweichen. Konkret müssen also Anomalien von einem Machine-Learning-Algorithmus identifiziert werden. Eine Studie der University of California, Berkeley bezeichnet Anomalien in Datensätzen als „wenige Zustände, die sich in der Menge und im Wert von einem Großteil der Zustände unterscheiden“. [SKS15]

Dies gilt es im Zuge dieser Arbeit anzuwenden, um die Predictive Maintenance umzusetzen. Dafür kämen unter anderem zwei verschiedene Modelle infrage: zum einen das

„Isolation Forest“-Modell und zum anderen das „Autoencoder“-Modell. Jene werden im Folgenden erläutert.

4.2.1 Isolation Forest

Isolation Forest ist ein Algorithmus, der von Datensampling, binären Baumdiagrammen und Partitionen Gebrauch macht. Besonders wichtig ist die Partition, also die Unterteilung von Datenelementen in verschiedene Datensegmente, um Anomalien festzustellen. Diese werden anschließend rekursiv isoliert. [SKS15] Isolation Forest ist ein Bestandteil der scikit-learn-Bibliothek in der Python-Programmierung. Es kann in ein Python-Programm über die folgende Programmzeile eingebunden werden:

```
1       from sklearn.ensemble import IsolationForest
```

Isolation Forest kann anschließend als Funktion verwendet werden. Im Folgenden wird ein kurzes Codebeispiel mit zusätzlichen grafischen Darstellungen der Anomalien gezeigt.

```
1     import numpy as np
2     import pandas as pd
3     from sklearn.ensemble import IsolationForest
4     import matplotlib.pyplot as plt
5
6     # 1. Simulierte Sensordaten erstellen
7     np.random.seed(42)
8     n_samples = 1000
9     # Normale Betriebsdaten simulieren
10    temperature = np.random.normal(70, 5, n_samples)  #
    Durchschnitt: 70°C, Schwankung: ±5
11    rpm = np.random.normal(1500, 50, n_samples)       #
    Durchschnitt: 1500 U/min, Schwankung: ±50
12    vibration = np.random.normal(0.03, 0.01, n_samples)  #
    Durchschnitt: 0.03g, Schwankung: ±0.01
```

```
13
14      # Anomalien hinzufügen
15      n_anomalies = 50
16      temperature_anomalies = np.random.uniform(100, 120,
    n_anomalies)
17      rpm_anomalies = np.random.uniform(1000, 1200, n_anomalies)
18      vibration_anomalies = np.random.uniform(0.1, 0.2, n_anomalies
    )
19
20      # Kombinieren von normalem Betrieb und Anomalien
21      temperature = np.concatenate([temperature,
    temperature_anomalies])
22      rpm = np.concatenate([rpm, rpm_anomalies])
23      vibration = np.concatenate([vibration, vibration_anomalies])
24
25      # Daten in einem DataFrame speichern
26      data = pd.DataFrame({
27        "Temperature": temperature,
28        "RPM": rpm,
29        "Vibration": vibration
30      })
31
32      # 2. Isolation Forest trainieren
33      model = IsolationForest(n_estimators=100, contamination=0.05,
        random_state=42)
34      data['Anomaly_Score'] = model.fit_predict(data[["Temperature"
    , "RPM", "Vibration"]])
35      data['Anomaly'] = data['Anomaly_Score'].apply(lambda x: 1 if
    x == -1 else 0)
36
37      # 3. Ergebnisse visualisieren
38      # Farben für Anomalien (rot) und normale Punkte (blau)
39      colors = data['Anomaly'].map({0: 'blue', 1: 'red'})
40
41      # Paarplot (Temperature vs. RPM)
42      plt.figure(figsize=(10, 6))
```

```
43      plt.scatter(data['Temperature'], data['RPM'], c=colors, alpha
    =0.6)
44      plt.title("Anomalieerkennung: Temperature vs RPM")
45      plt.xlabel("Temperature (°C)")
46      plt.ylabel("RPM")
47      plt.grid(True)
48      plt.show()
49
50      # Anomalienanzahl ausgeben
51      anomalies_detected = data['Anomaly'].sum()
52      anomalies_detected
```

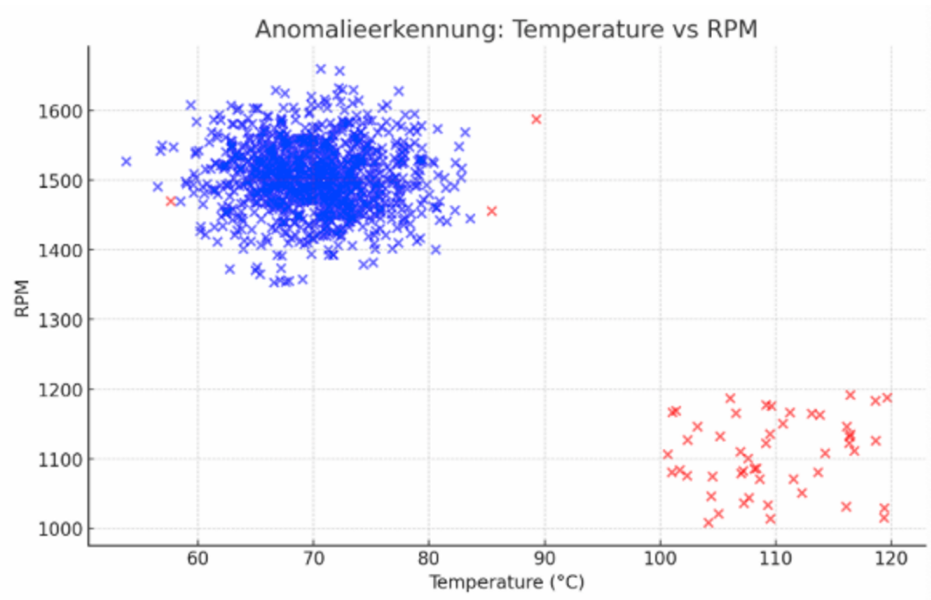


Abbildung 4.1: Anomalienerkennung Isolation Forest (Anomalien = rot); eigene Darstellung

4.2.2 Autoencoder

Ein Autoencoder ist eine weitere Möglichkeit zur Anomalieerkennung in großen Datenmengen. Dabei ist ein Autoencoder „eine Art Architektur für neuronale Netze, die darauf ausgelegt ist, Eingabedaten zu komprimieren [und] sie auf ihre wesentlichen Merkmale zu

reduzieren“. [Dav23] Autoencoder werden mithilfe von unüberwachtem Lernen trainiert, um nicht messbare Zustände in Eingabedaten festzustellen, die eine Datenmenge unmittelbar beeinflussen. Daraus werden anschließend Merkmale extrahiert, mit denen die Originaldaten rekonstruiert werden. Auf Basis dieser Konvertierung kann anschließend eine Anomalieerkennung erfolgen. [Dav23]

In der Python-Programmierung können grundlegende Autoencoder über die Bibliothek „tensorflow“ verwendet werden. Ein Codebeispiel für die grundlegende Anwendung von Autoencodern ist in Anhang 5 zu finden.

4.3 Visualisierung und Entscheidungsunterstützung

Die Visualisierung von Daten und deren Einbindung in Entscheidungsprozesse spielt eine zentrale Rolle in modernen Produktions- und Logistiksystemen. Mithilfe von grafischen Darstellungen, wie Dashboards, können komplexe Daten übersichtlich präsentiert und relevante Informationen benutzerfreundlich aufbereitet werden. Dieses Kapitel beleuchtet die Bedeutung der Visualisierung für die Entscheidungsunterstützung sowie die Möglichkeiten und Vorteile, die durch die Nutzung der Low-Code-Plattform PeakBoard geboten werden.

4.3.1 Rolle der Visualisierung

Die grafische Darstellung der Zustandsüberwachung eines Systems ist zwar nicht zwingend erforderlich, bietet jedoch eine effektive Möglichkeit, ein besseres Verständnis der Funktionsweise und der Entwicklung von Daten zu fördern. Insbesondere die Visualisierung von Produktionsdaten und die Überwachung von Motorzuständen erweisen sich als besonders interessant. Zu diesem Zweck können Diagramme oder Schaubilder in Dashboards integriert werden, um relevante Informationen strukturiert und übersichtlich darzustellen. [MW21]

Dashboards bieten zudem die Möglichkeit zur Interaktion durch die Integration von Schaltflächen (Buttons), die unterschiedliche Funktionen übernehmen können. Einerseits ermöglichen sie die Reaktion auf Benutzereingaben, andererseits können sie als Auslöser (Trigger) für spezifische Aktionen dienen. Darüber hinaus können Benachrichtigungen generiert werden, wenn definierte Schwellenwerte erreicht werden. Diese Benachrichtigungen können ebenfalls grafisch aufbereitet werden, um eine schnelle Erfassung der Informationen zu gewährleisten. [MW21]

Auf Basis der ausgelösten Benachrichtigungen oder Warnmeldungen lassen sich anschließend konkrete Handlungsempfehlungen visualisieren, wodurch ein systematisches und datenbasiertes Entscheidungsmanagement unterstützt wird. [MW21]

4.3.2 Peak Board als Tools zur Datenvisualisierung

Zur Visualisierung von Daten steht eine Vielzahl von Anbietern zur Auswahl. Letztlich fiel die Entscheidung jedoch auf die Low-Code-Plattform Peakboard, was auf mehrere Gründe zurückzuführen ist. Der Hauptgrund liegt darin, dass diese Plattform bereits in vorherigen Arbeiten eingesetzt und somit in das bestehende System integriert wurde. Dies spart den Aufwand, eine alternative Software einzurichten, und ermöglicht eine nahtlose Fortführung der Arbeit.

Darüber hinaus bietet Peakboard eine benutzerfreundliche Lösung für smarte Digitalisierungsprozesse, was die Bedienung erleichtert und ein schnelleres Vorankommen im Projekt sicherstellt. Ein weiterer wesentlicher Vorteil liegt in der hohen Flexibilität der Plattform in Bezug auf verschiedene Anwendungsfälle. So erlaubt Peakboard die Kombination von Daten aus unterschiedlichen Quellsystemen, was die Anpassungsfähigkeit und Vielseitigkeit unterstreicht. [Gmb24]

Besonders hervorzuheben ist die Möglichkeit, Lösungen mit Manufacturing Execution System (MES)- oder Web Map Service (WMS)-Funktionen zu realisieren. Dies ermöglicht die Entwicklung von Produktions- und Logistiksystemen, die später über Human Machine Interfaces steuerbar sind. Dadurch lässt sich die Wertschöpfung des Bandsys-

tems maximieren, indem die Produktivität gesteigert wird, ohne die Kosten erheblich zu erhöhen. [Gmb24]

Ein weiterer entscheidender Aspekt ist die IT-Sicherheit. Durch den Betrieb der Software im eigenen Netzwerk entfällt die Notwendigkeit einer externen Verbindung, sodass alle Daten intern verarbeitet werden können. Dies gewährleistet ein hohes Maß an Datensicherheit und schützt sensible Informationen vor externen Zugriffen. [Gmb24]

4.3.3 Design eines Dashboards

Um die analysierten Daten übersichtlich darzustellen und den Mitarbeitenden an der Fertigungslinie relevante Informationen bereitzustellen, kann ein Dashboard mit der Softwarelösung PeakBoard gestaltet werden. Ein exemplarisches Design eines solchen Dashboards ist in Abbildung 4.2 visualisiert.

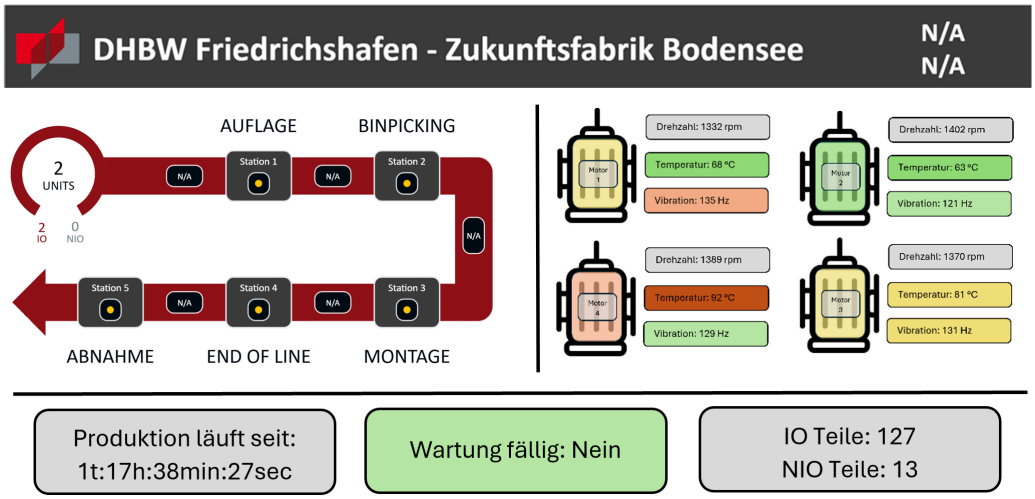


Abbildung 4.2: Beispielhaftes Dashboard zur Visualisierung mithilfe des PeakBoard; eigene Darstellung

Das Dashboard gliedert sich in zwei Hauptbereiche. Im linken Bereich wird das Bandsystem mit den einzelnen Stationen abgebildet. Jede Station zeigt an, ob sie aktuell in Betrieb ist. Eine aktive Station wird durch einen grünen Punkt gekennzeichnet, während ein gelber Punkt auf einen Stillstand hinweist. Der rechte Bereich des Dashboards

dient der Zustandsüberwachung sowie der Implementierung von Predictive-Maintenance-Funktionen für das Transportsystem. Hier werden vier verschiedene Motoren dargestellt, einschließlich der Sensorwerte, die zur Überwachung erhoben werden. Diese umfassen die Drehzahl (erfasst durch induktive Sensoren), die Temperatur (ermittelt durch Temperatursensoren) und die Schwingungsintensität (gemessen durch Beschleunigungssensoren).

Im Kontext der Predictive Maintenance erfolgt eine dynamische Farbzuordnung der Sensorwerte sowie der Motoranzeige, basierend auf deren Zustand. Die Drehzahl wird als Referenzwert verwendet, um Abweichungen in Temperatur und Vibration zu bewerten. Je nach Abweichungsgrad werden die entsprechenden Werte und der Motor von grün (optimal) bis rot (kritisch) eingefärbt. Die aggregierte Bewertung der einzelnen Parameter führt zu einer Gesamtfarbzuweisung für den Motor, die dessen Zustand anschaulich widerspiegelt.

Im unteren Bereich des Dashboards werden allgemeine Systeminformationen präsentiert. Dazu gehören beispielsweise die Gesamtbetriebszeit des Bandsystems sowie die Anzahl der produzierten Teile, differenziert nach einwandfreien und fehlerhaften Produkten. Zusätzlich gibt es ein Feld, das erforderliche Wartungsmaßnahmen anzeigt, welche auf Algorithmen der Predictive Maintenance basieren.

5 Fazit

Die vorliegende Arbeit liefert eine fundierte Grundlage für die umfassende Implementierung eines Predictive-Maintenance-Systems für ein förderbandbasiertes Transportsystem. Im Rahmen der Untersuchung wurden alle vier Motoren des Systems mit Sensoren ausgestattet und verschiedene Methoden zur Umsetzung von Predictive Maintenance experimentell erprobt. Dabei konnte gezeigt werden, dass der Einsatz moderner Datenanalysemethoden wie Isolation Forest und Autoencoder sowie die Visualisierung durch Low-Code-Plattformen wie Peakboard zu einer erheblichen Verbesserung der Systemzuverlässigkeit und Wartungseffizienz beitragen können.

Ein unerwarteter Rückschlag war der Ausfall des ursprünglich vorgesehenen Intel NUC, auf dem die Arbeit aufbauen sollte. Trotz intensiver Fehleranalysen und Reparaturversuche stellte sich heraus, dass ein irreparabler Defekt der Speicherkarte den NUC unbrauchbar machte. Dieser Umstand durchkreuzte den ursprünglichen Plan, spezifische Codelösungen zu implementieren und deren Praxistauglichkeit zu testen. Um dennoch die Zielsetzungen der Arbeit zu erfüllen, wurde ein neuer Mini-PC beschafft und aufgebaut. Dies gestaltete sich aufgrund von Lieferverzögerungen sowie der Einarbeitung in neue Software und Systeme als herausfordernd. Dennoch führte dieser Schritt zu wertvollen persönlichen Lernfortschritten und trug dazu bei, die Projektziele weiterzuverfolgen.

Die durchgeführten Maßnahmen und erzielten Ergebnisse unterstreichen das Potenzial von Predictive Maintenance, Ausfallzeiten zu minimieren und die Produktionssicherheit zu erhöhen. Die kontinuierliche Erfassung und Analyse von Parametern wie Temperatur, Drehzahl und Vibration ermöglicht eine frühzeitige Identifikation von Anomalien, wodurch präventive Maßnahmen rechtzeitig eingeleitet werden können. Neben der Op-

timierung von Wartungsprozessen leistet das entwickelte System einen theoretischen Beitrag zur Effizienzsteigerung und Senkung von Betriebskosten.

Gleichzeitig offenbarte die Arbeit technische und organisatorische Herausforderungen. Insbesondere die Datenqualität und die Integration der Sensorik in bestehende Prozesse erforderten große Sorgfalt. Die technische Komplexität der Systemkonfiguration sowie die Notwendigkeit, den Umgang mit der neuen Technologie zu schulen, waren ebenfalls zentrale Punkte. Dennoch konnte eine stabile Infrastruktur geschaffen werden, die als Basis für weiterführende Entwicklungen dient.

Für zukünftige Arbeiten ergeben sich vielversprechende Perspektiven. Dazu zählen die Weiterentwicklung der Machine-Learning-Modelle sowie die Verbesserung der Visualisierungs- und Analysefunktionen. Langfristig sollte eine Integration in umfassendere Industrie-4.0-Strategien angestrebt werden, um die Potenziale der vorausschauenden Wartung vollständig auszuschöpfen. Insgesamt leistet diese Arbeit einen wesentlichen Beitrag zur Weiterentwicklung intelligenter, effizienter und zuverlässiger Wartungskonzepte in der modernen Produktionsumgebung.

Literaturverzeichnis

[asu24] asus. *NUC - Viel Leistung trotz kompaktem Formfaktor*. Online. Abgerufen am 10. Dezember 2024. 2024. URL: https://www.asus.com/de/content/nuc-overview/.

[Bur19] Andriy Burkov. *The hundred-page machine learning book*. [Polen]: Andriy Burkov, 2019. 141 S. ISBN: 9781999579517.

[Dav23] Cole Stryker Dave Bergmann. *Was ist ein Autoencoder?* IBM. Aufgerufen am 27.11.2024. Nov. 2023. URL: https://www.ibm.com/de-de/topics/autoencoder.

[Gér23] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. Concepts, tools, and techniques to build intelligent systems*. Third edition. Data science / machine learning. Literaturangaben. - Index. Beijing: O'Reilly, 2023. 834 S. ISBN: 9781098125974.

[gmb16a] ifm electronic gmbh. *IFC201 Induktiver Sensor*. ifm electronic gmbh. Essen, Germany, Nov. 2016.

[gmb16b] ifm electronic gmbh. *TS2229 Temperatur-Kabelsensor mit Anlegefühler*. ifm electronic gmbh. Essen, Germany, Okt. 2016.

[gmb23] ifm electronic gmbh. *VSA001 Beschleunigungssensor*. ifm electronic gmbh. Essen, Germany, Apr. 2023.

[Gmb24] Peakboard GmbH. *Peakboard - Visualisierung von Daten in Echtzeit*. Zuletzt aufgerufen am 2024-11-22. 2024. URL: https://peakboard.com/.

[Kel23] Patricia Kelbert. *Predictive Maintenance umsetzen: Wie geht das?* Zugriff am 21. November 2024. Juni 2023. URL: https://www.iese.fraunhofer.de/blog/predictive-maintenance/.

[KHV06] Ranganath Kothamasu, Samuel H. Huang und William H. VerDuin. „System health monitoring and prognostics - a review of current paradigms and practices“. In: *The International Journal of Advanced Manufacturing Technology* 28.9–10 (Juli 2006), S. 1012–1024. ISSN: 1433-3015. DOI: 10.1007/s00170-004-2131-6.

[LBK15] Jay Lee, Behrad Bagheri und Hung-An Kao. „A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems“. In: *Manufacturing Letters* 3 (Jan. 2015), S. 18–23. ISSN: 2213-8463. DOI: 10.1016/j.mfglet.2014.12.001.

[Luo+20] Weichao Luo u. a. „A hybrid predictive maintenance approach for CNC machine tool driven by Digital Twin“. In: *Robotics and Computer-Integrated Manufacturing* 65 (Okt. 2020), S. 101974. ISSN: 0736-5845. DOI: 10.1016/j.rcim.2020.101974.

[Mob02] R. Keith Mobley. *Introduction to Predictive Maintenance.* Hrsg. von R. Keith Mobley. 2nd ed. Plant Engineering Ser. Description based on publisher supplied metadata and other sources. Oxford: Elsevier Science & Technology, 2002. 1451 S. ISBN: 9780080478692.

[Mül17] Andreas Christian Müller. *Introduction to machine learning with Python. A guide for data scientists.* Hrsg. von Sarah Guido. First edition. Sebastopol, CA: O'Reilly Media, 2017. 1378 S.

[MW21] Stefan Meinhardt und Felix Wortmann, Hrsg. *IoT - Best Practices. Internet der Dinge, Geschäftsmodellinnovationen, IoT-Plattformen, IoT in Fertigung und Logistik.* Edition HMD. Literaturangaben. Wiesbaden: Springer Vieweg, 2021. 422 S. ISBN: 3658324384.

[Sch22] Günther Schuh. *Predictive Maintenance.* Hrsg. von Christoph Kelzenberg u. a. Erscheinungsjahr der gedruckten Ausgabe des Digitalisats: 2020. Aachen: WBA Aachener Werkzeugbau Akademie GmbH, 2022. 1 S. ISBN: 9783946612513.

[SE24] SAP SE. *What is Predictive Maintenance?* Abgerufen am 13. November 2024. 2024. URL: https://www.sap.com/germany/products/scm/apm/what-is-predictive-maintenance.html.

[SKS15] Carol Stanton, Gilad Katz und Dawn Song. *Isolation Forest for Anomaly Detection*. Techn. Ber. Aufgerufen am 27.11.2024. Center for Energy Efficient Electronics Science (E3S), University of California, Berkeley, 2015. URL: https://e3s-center.berkeley.edu/wp-content/uploads/2017/08/RET_CStanton-2015.pdf.

[Sut20] Richard S. Sutton. *Reinforcement learning. An introduction*. Hrsg. von Andrew Barto. Second edition. Adaptive computation and machine learning. Hier auch später erschienene, unveränderte Nachdrucke. Cambridge, Massachusetts: The MIT Press, 2020. 526 S. ISBN: 9780262039246.

Abbildungsverzeichnis

2.1 Grafik zur Predictive Maintenance. Quelle: All-Electronics.de 4

3.1 Anbringung des Temperatursensor an das Typenschild; eigene Darstellung 12
3.2 Loch im Lüftergehäuse sowie Schraube am Lüfter, eigene Darstellung . . 13
3.3 Induktiver Sensor am Lüftergehäuse; eigene Dartstellung 13

4.1 Anomalienerkennung Isolation Forest (Anomalien = rot); eigene Darstellung 22
4.2 Beispielhaftes Dashboard zur Visualisierung mithilfe des PeakBoard; eigene Darstellung 25

Abkürzungsverzeichnis

IoT Internet of Things

MQTT Message Queuing Telemetry Transport

MTTF Mean Time to Failure

PdM Predictive Maintenance

KI Künstliche Intelligenz

LLM Large Language Models

DNN Deep Neural Networks

SPS speicherprogrammierbare Steuerung

MES Manufacturing Execution System

WMS Web Map Service

BYOD Bring Your Own Device

NUC Next Unit of Computing

A Code Autoencoder

```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt

# Simulierte Daten erstellen
np.random.seed(42)
n_samples = 1000

# Normale Betriebsdaten simulieren
temperature = np.random.normal(70, 5, n_samples)  # Mittelwert:
  70°C, Schwankung: ±5
rpm = np.random.normal(1500, 50, n_samples)       # Mittelwert:
  1500 U/min, Schwankung: ±50
vibration = np.random.normal(0.03, 0.01, n_samples)  #
 Mittelwert: 0.03g, Schwankung: ±0.01

# Anomalien hinzufügen
n_anomalies = 50
temperature_anomalies = np.random.uniform(100, 120, n_anomalies
 )
rpm_anomalies = np.random.uniform(1000, 1200, n_anomalies)
vibration_anomalies = np.random.uniform(0.1, 0.2, n_anomalies)

```

```
23  # Kombinieren der normalen Betriebsdaten und Anomalien
24  temperature = np.concatenate([temperature,
     temperature_anomalies])
25  rpm = np.concatenate([rpm, rpm_anomalies])
26  vibration = np.concatenate([vibration, vibration_anomalies])
27
28  # Daten in einem DataFrame speichern
29  data = pd.DataFrame({
30    "Temperature": temperature,
31    "RPM": rpm,
32    "Vibration": vibration
33  })
34
35  # Autoencoder-Modell erstellen
36  input_layer = Input(shape=(3,))  # 3 Eingabefunktionen (
     Temperatur, RPM, Vibration)
37  encoded = Dense(64, activation='relu')(input_layer)
38  encoded = Dense(32, activation='relu')(encoded)
39  encoded = Dense(16, activation='relu')(encoded)
40  decoded = Dense(32, activation='relu')(encoded)
41  decoded = Dense(64, activation='relu')(decoded)
42  decoded = Dense(3, activation='linear')(decoded)  #
     Wiederherstellung der Eingabedimension (3)
43  autoencoder = Model(inputs=input_layer, outputs=decoded)
44
45  # Modell kompilieren
46  autoencoder.compile(optimizer=Adam(), loss='mse')
47
48  # Modell mit normalen Daten trainieren (ohne Anomalien)
49  normal_data = data[:n_samples][['Temperature', 'RPM', '
     Vibration']].values
50  autoencoder.fit(normal_data, normal_data, epochs=50, batch_size
     =32, verbose=0)
51
52  # Rekonstruktionsfehler berechnen
```

```
reconstructed = autoencoder.predict(data[['Temperature', 'RPM',
    'Vibration']].values)
reconstruction_error = np.mean(np.square(data[['Temperature', '
    RPM', 'Vibration']].values - reconstructed), axis=1)

# Anomalien erkennen: Hoher Rekonstruktionsfehler bedeutet
    Anomalie
threshold = np.percentile(reconstruction_error, 95)  # Setze
    Schwelle basierend auf den oberen 5% der Fehler
data['Anomaly_Autoencoder'] = (reconstruction_error > threshold
    ).astype(int)

# Visualisierung der erkannten Anomalien
colors = data['Anomaly_Autoencoder'].map({0: 'blue', 1: 'red'})

# Plot der Rekonstruktionsfehler vs. Temperatur
plt.figure(figsize=(10, 6))
plt.scatter(data['Temperature'], reconstruction_error, c=colors
    , alpha=0.6)
plt.title("Autoencoder: Rekonstruktionsfehler vs Temperatur")
plt.xlabel("Temperature (°C)")
plt.ylabel("Rekonstruktionsfehler")
plt.grid(True)
plt.show()

# Anzahl der erkannten Anomalien ausgeben
anomalies_detected_autoencoder = data['Anomaly_Autoencoder'].
    sum()
print(f"Anzahl der erkannten Anomalien: {
    anomalies_detected_autoencoder}")
```

B IoT-Stack Installationscode

IoT-Stack Docker Compose File:

```
    version: "3.8"

services:

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

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

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

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

networks:
```

```
    node-red-net:

volumes:

    nodered-data:
    timescaledb-data:
    grafana-data:
```