FLORIAN SENFT

Kommunikationstechnische Optimierung eines energieautarken funkbasierten Sensorkonzepts

FLORIAN SENFT

Kommunikationstechnische Optimierung eines energieautarken funkbasierten Sensorkonzepts

HTWK Hochschule für Technik, Wirtschaft und Kultur Leipzig

Über den Autor
Florian Senft absolvierte von 2014 bis 2018 sein Bachelor-Studium *Wirtschaftsingenieurwesen Produktions- und Energiewirtschaft* an der HTWK Leipzig, in dessen Rahmen das vorliegende Werk als Abschlussarbeit verfasst wurde. Während des Studiums sammelte er bereits Erfahrungen im Bereich Projektsteuerung bei der BMW Group. Seit 2018 studiert Senft *Wirtschaftsingenieurwesen Maschinenbau* im Masterstudiengang an der HTWK Leipzig.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie. Detaillierte bibliografische Daten sind im Internet unter http://dnb.de abrufbar.

Die Online-Version dieser Publikation ist abrufbar unter
http://doi.org/10.33968/9783966270069-00

Herausgeber
Open-Access-Hochschulverlag
Hochschule für Technik, Wirtschaft und Kultur Leipzig
Karl-Liebknecht-Str. 132
04277 Leipzig, Deutschland

Druck & Bindung in Deutschland und den Niederlanden
Gedruckt auf säurefreiem Papier

ISBN (Hardcover) 978-3-96627-004-5
ISBN (Softcover) 978-3-96627-005-2
ISBN (PDF) 978-3-96627-006-9
ISBN (ePub) 978-3-96627-007-6

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADC	Analog-Digital-Converter (Analog-Digital-Wandler)
BLE	Bluetooth Low Energy
CoAP	Constrained Application Protocol
GPIO	General Purpose Input Output
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things (Internet der Dinge)
IP	Internet Protocol
MEMS	mikroelektromechanische Systeme
MQTT	Message Queuing Telemetry Transport
OPV	Operationsverstärker
OSI	Open System Interconnection
QoS	Quality of Service
RPI	Raspberry Pi 2
SPI	Serial Peripheral Interface
TCP	Transmission Control Program
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network (drahtloses lokales Netzwerk)

1 Einleitung

Ausgangssituation
An technische Anlagen besteht die Anforderung einer hohen Anlagenverfügbarkeit bei gleichzeitig geringen Kosten, welche durch eine zustandsorientierte prädiktive Instandhaltung besser erfüllt werden kann. Voraussetzung dafür ist eine frühzeitige Erkennung von Ermüdungs- und Verschleißerscheinungen vor dem Ausfall funktionskritischer Komponenten. Es wird eine hohe Ausnutzung des Abnutzungsvorrats eines Funktionselements bei gleichzeitig besser planbaren Wartungs- und Instandhaltungsmaßnahmen erreicht.

Für die Umsetzung einer zustandsorientierten prädiktiven Instandhaltungsstrategie ist ein geeignetes Messkonzept notwendig, welches durch die Messung passender zustandsbeschreibender Merkmale den Zustand der Funktionskomponenten erfasst und diese zur Analyse und Darstellung an eine Leitwarte sendet. Ein kabelloses funkbasiertes Messsystem senkt den Installationsaufwand erheblich und ermöglicht Messungen auch an schwer zugänglichen Orten. Ein energieautarkes System reduziert den Wartungsaufwand, da z.B. Batteriewechsel überflüssig werden.

Grundlage dieser Bachelorarbeit ist ein solches in einem früheren Projekt entwickeltes Sensorkonzept.

Problemstellung
Die energetische Lebensdauer eines energieautarken Funksensorsystems hängt von der hard- und softwaretechnischen Umsetzung des Systems ab. Die Sendeleistung des Transceivers beeinflusst den Gesamtenergieverbrauch des Sensorknotens stark. Bei dem bestehenden Sensorkonzept werden deshalb zur Verbesserung der Energieeffizienz vor allem aufgrund der Sendedauer Optimierungspotenziale vermutet.

Zielstellung
Ziel dieser Bachelorarbeit ist die Untersuchung des bestehenden Sensorkonzepts auf kommunikationstechnische Optimierungspotentiale zur Verbesserung der Energieeffizienz. Die größten Potentiale werden softwareseitig bei der Netzwerkkommunikation gesehen. Deshalb soll mit der vorhandenen Hardware ein Alternativkonzept zur Datenübertragung mittels des leichtgewichtigen Netzwerkkommunikationsprotokolls MQTT (Message Queuing Telemetry Transport) umgesetzt werden. Es soll ein Szenario entwickelt werden, welches vergleichbar mit dem bestehenden Sensorkonzept ist, um eine objektive Bewertung der Veränderung der Energieeffizienz durchführen zu können. Dazu ist es notwendig, eine messtechnische Strategie zur Bestimmung der Sendedauer und des Energieverbrauchs zu erstellen.

http://doi.org/10.33968/9783966270069.0001

Lösungsansatz
Es werden zunächst die Anwendungsmöglichkeiten solcher Sensorsysteme in den bestehenden Instandhaltungsstrategien beschrieben, sich dafür grundsätzlich eignende Messprinzipien und zustandsbeschreibende Merkmale erarbeitet, sowie auf bestehende Projekte eingegangen. Anschließend erfolgt die Analyse des bisherigen Sensorkonzepts, auf deren Basis mit der vorhandenen Hardware ein Alternativkonzept zur kommunikationstechnischen Optimierung realisiert wird. Zudem wird ein Messkonzept zur Bestimmung der Sendedauer entwickelt, welches einen Vergleich mit dem bestehenden Sensorkonzept ermöglicht. Es kann somit eingeschätzt werden, inwieweit die Zielstellung erreicht wurde.

2 Stand der Technik

Funkbasierte Sensorsysteme ermöglichen eine Überwachung von Systemen und Anlagen, die mit der herkömmlichen kabelgebundenen Technik vor allem bei schwer zugänglichen Funktionselementen aufgrund des hohen Installationsaufwandes nicht wirtschaftlich realisierbar wäre. Dies schafft neue Möglichkeiten im Bereich der Instandhaltung, da die Kosten für die Überwachung von Prozessen und Anlagen sinken. Zudem können Daten derart erfasst und ausgewertet werden, dass sich der Ausfall einer Funktionseinheit vorherbestimmen lässt. Im Folgenden wird einführend auf Strategien der Instandhaltung und die sich daraus ergebenden Einsatzmöglichkeiten solcher Sensorsysteme eingegangen. Ferner werden die Grundlagen der dafür benötigten zustandsbeschreibenden Merkmale und messtechnischen Prinzipien dargelegt und kurz auf bestehende Projekte an der Professur für Industrielle Messtechnik an der HTWK in diesem Bereich eingegangen.

Instandhaltungsstrategien
Der Einsatz technischer Maschinen und Anlagen führt aufgrund chemischer und physikalischer Vorgänge zu natürlicher Abnutzung. Dazu zählen u.a. mechanischer Verschleiß, Korrosion, Alterung und Ermüdung. Jede Maschine und Anlage, aber auch jedes einzelne Bauelement verfügt über einen individuellen Abnutzungsvorrat, der den Vorrat möglicher Funktionserfüllungen darstellt. Bei Erstinbetriebnahme liegt der Abnutzungsvorrat bei 100%. Dieser sinkt infolge der Abnutzung während des Betriebes. Aufgabe der Instandhaltung ist es sicherzustellen, dass der Abnutzungsvorrat einer Einheit über der individuellen Abnutzungsgrenze liegt. Diese drückt sich z.B. in zunehmenden Ausschuss, technischen Schäden oder den Ausfall der Einheit aus. Die Instandhaltung besitzt damit Einfluss auf die Qualität des Produktionsprozesses und der Produkte. [71]

In modernen Produktionskonzepten erfolgt keine Unterteilung mehr in Haupt- und Nebenprozesse. Die Instandhaltung wird damit zu einer unternehmensübergreifenden Aufgabe, welche zu einer effizienten Produktion beiträgt. In Organisationskonzepten wie z.B. der TPM (Total Produktive Management) ist die Instandhaltung ein Teil der Ansatzpunkte, um Verschwendungen und Verluste zu reduzieren und eine möglichst hohe Anlageneffizienz zu erreichen. [45] [60]

Durch die Instandhaltung erfolgt der Erhalt der Funktions- und Leistungsfähigkeit einer technischen Einheit. Um dies umzusetzen, gibt es einige grundlegende Strategien der Instandhaltung, die den Zeitpunkt und die Art der Maßnahmen regeln. Bei der Auswahl einer geeigneten Strategie spielen gesetzliche, sicherheitsrelevante, technische, produktionsrelevante und wirtschaftliche Aspekte eine Rolle. [60] Da je nach Anwendungsfall unterschiedliche Voraussetzungen, Zusammenhänge und Konsequenzen eines Ausfalls bestehen, gibt es nicht die eine geeignete Instandhaltungsstrategie, sondern der Einsatz muss individuell betrachtet werden.

http://doi.org/10.33968/9783966270069.0002

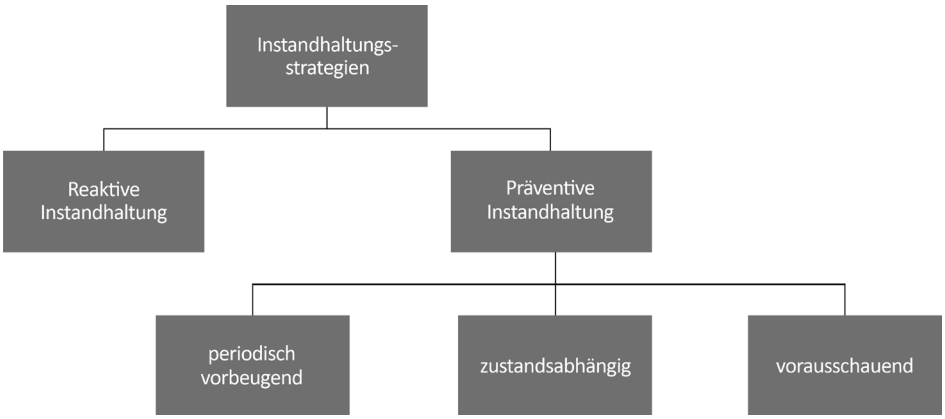


Abb. 1 Instandhaltungsstrategien [60]

In [60] wird die in Abbildung 1 aufgezeigte Unterscheidung in zwei Grundstrategien vorgenommen - der reaktiven und der präventiven Instandhaltung.

Bei der reaktiven Instandhaltung wird der Abnutzungsvorrat vollständig ausgenutzt, da eine Reaktion der Instandhaltung erst nach Ausfall einer Komponente stattfindet. Aufgrund des damit verbundenen Unfallrisikos und Gefährdungspotentials oder erwartbarer Folgeschäden anderer Anlagenteile kann diese Strategie im Einzelfall unzulässig sein. Zumeist tritt der Schadensfall plötzlich ein, sodass eine Planung von Instandhaltungsmaßnahmen nicht möglich ist und die benötigten Instandhaltungsressourcen entweder extra vorgehalten werden müssen oder im Zweifelsfall nicht vorhanden sind. Dies führt im Vergleich mit anderen Strategien zu den höchsten Ausfallzeiten und den höchsten Ausfallfolgekosten. Dadurch ist eine empfehlenswerte Anwendung auf Anlagen mit redundanten Systemen oder einer schnellen Schadensbehebung beschränkt, die keine Sicherheitsanforderungen erfüllen müssen und deren Ausfall keine Unterbrechung der Produktion bewirkt. [60]

Dem gegenüber stehen präventive Instandhaltungsstrategien, bei denen Maßnahmen vor Eintreten des Schadensfalls getroffen werden. Der Zeitpunkt des Einsatzes dieser Maßnahmen wird entweder periodisch vorbeugend, zustandsabhängig oder vorausschauend bestimmt. [60]

Bei der präventiv-periodisch vorbeugenden Instandhaltungsstrategie wird die Komponente unabhängig von ihrem Zustand nach definierten Nutzungsintervallen instandgesetzt oder ausgetauscht. Diese Intervalle werden entweder zeitbezogen zu festgelegten Zeitpunkten bzw. nach einer definierten Anzahl von Betriebsstunden oder aber ereignisbezogen z.B. nach Stückzahlen oder gefahrenen Kilometern bestimmt. Vorteile dieser Strategie sind eine gute Planbarkeit der Instandhaltungsmaßnahmen sowie ein geringes Ausfallrisiko. Allerdings wird der Austausch der Komponenten zumeist vor Ausnutzung ihrer Abnutzungsvorräte erfolgen, sodass der Verbrauch von Materialien erhöht wird. Voraussetzung für diese Strategie ist eine gute Prognose über die zu erwartende Lebensdauer der Funktionselemente.

Sonst entstehen entweder hohe Vorbeugungskosten durch die mangelhafte Ausnutzung der Abnutzungsvorräte oder ein Ausfall der Einheit erfolgt frühzeitiger als vorhergesagt, was zu den Folgen der reaktiven Instandhaltungsstrategie führt. [60]

Die zustandsabhängige Instandhaltungsstrategie setzt bei dem Nachteil des unzureichend ausgenutzten Abnutzungsvorrats an. Statt wie bei der periodisch vorbeugenden Instandhaltung nach starren Nutzungsintervallen vorzugehen, ist die Steuergröße zum Einleiten von Instandhaltungsmaßnahmen der Zustand der zu betrachtenden Einheit. Voraussetzung dafür sind aktuelle Zustandsinformationen der Anlage, die Rückschlüsse auf die Veränderung des Abnutzungsvorrats zulassen und deren Erfassung mit wirtschaftlich vertretbaren Aufwand erfolgen kann. Diese zustandsbestimmenden Parameter können entweder durch Inspektionen, bei denen durch den Menschen zustandsrelevante Parameter erfasst und bewertet werden oder mit technischer Unterstützung durch Systeme zur Zustandsüberwachung gewonnen werden. In Abhängigkeit von den geeigneten zustandsbeschreibenden Merkmalen gibt es verschiedene Messprinzipien und -systeme, worauf im Laufe dieses Kapitels näher eingegangen wird. [60]

Im Vergleich zur vorbeugenden Instandhaltungsstrategie wird der Abnutzungsvorrat der technischen Einheit erheblich besser ausgenutzt, da Instandhaltungsmaßnahmen auf Basis des Zustandes und nicht aufgrund von Erfahrungswerten oder statistischer Auswertungen eingeleitet werden. Auch kann ein frühzeitiger als erwartet erfolgter Ausfall von Komponenten erkannt werden, wodurch die Zuverlässigkeit und Verfügbarkeit der Anlagen steigt. Allerdings fallen Mehrkosten für die notwendigen Überwachungs- und Diagnosesysteme an und die Überwachung der Einheiten muss mit geeigneten Zustandsparametern, Messstellen und Sensoren überhaupt prinzipiell und mit einem vertretbaren Aufwand möglich sein. [71]

Die vorausschauende Instandhaltung setzt gegenüber der zustandsabhängigen Strategie noch deutlich eher an. Ziel ist es, bereits potenzielle Störungen zu erkennen und entweder deren Weiterentwicklung gezielt zu verhindern oder den Ausfall möglichst frühzeitig vorauszusagen und damit eine bessere Einplanung der Instandhaltungsmaßnahmen in den Produktionsplan zu ermöglichen. Voraussetzung dafür sind eine statistisch relevante Datenbasis an Zustandsinformationen und Ausfällen mit den damit verbundenen Zustandsparametern sowie die Kenntnis über die Ausfallzusammenhänge. Langfristig gedacht ermöglichen die daraus gewonnenen Erkenntnisse auch Maßnahmen wie bspw. konstruktive Änderungen, welche die Lebensdauer der betrachteten Einheit verlängern. [68]

Messtechnische Prinzipien und zustandsbeschreibende Merkmale

Die Daten, die eine Zustandsüberwachung für die zustandsabhängige und vorausschauende Instandhaltung ermöglichen, müssen auf eine sinnvolle Weise gewonnen werden. Dafür ist es notwendig, geeignete Merkmale zu finden, die den Zustand einer Einheit beschreiben, Rückschlüsse auf kommende Ausfälle zulassen und sich messtechnisch mit vertretbarem Aufwand erfassen lassen. Solche Merkmale stellen physikalischer Größen wie z.B. Schwingungen und Temperaturen sowie Verschleiß- und Schmierzustände dar. [51]

Weit verbreitet ist die Schwingungsdiagnose, mit der die Messgröße Körperschall überwacht wird. Schwingungssensoren erfassen das durch Betrieb der Anlage entstehende Schwingungssignal. Dieses wird auf seine Zusammensetzung aus Einzelsignalen hin z.B. mittels FFT-Analyse (FFT – Fast Fourier-Transformation) untersucht. So können charakteristische Frequenzen identifiziert werden, welche bei ungeschädigten Anlagen nicht auftreten und Rückschlüsse auf den Anlagenzustand gewonnen werden. [60]

Wenn Zustandsveränderungen der Anlage durch thermische Beanspruchungen hervorgerufen werden, kann eine Temperaturüberwachung sinnvoll sein. Diese erfolgt berührend durch Temperatursensoren an einzelnen Messpunkten oder berührungslos durch Thermografie von ganzen Anlagenabschnitten. [60]

Eine weitere Möglichkeit der Anlagenüberwachung stellt die Analyse von Ölzuständen dar. Abriebsensoren können Metallpartikel in Hydraulik- und Schmiersystemen feststellen und eine Unterscheidung in Eisen- und Nichteisenabrieb vornehmen. So können mit Trendmessungen Rückschlüsse auf den Verschleißzustand der Bauteile in diesen Systemen gewonnen werden. [51]

Erfassung und Verarbeitung der Messdaten

Die klassische Verfahrensweise zur Erfassung und Verarbeitung von Messdaten stellt zum heutigen Zeitpunkt die folgend beschriebene Messkette dar. Ein Sensor setzt die zu messende Größe in eine Spannung um, welche anschließend verstärkt wird. Eine Messkarte dient als Schnittstelle zwischen der Sensorik und einem Auswerterechner. Der Rechner kann über die Messkarte Einstellungen an die Sensorik übermitteln und die Messung auslösen. Sie digitalisiert zudem die Messwerte und ermöglicht damit eine Verarbeitung und Anzeige mittels einer Software auf dem Auswerterechner. Der Bediener des Messsystems nimmt die Analyse und Bewertung der Messwerte vor. [8]

Herkömmlich erfolgt die Übertragung der Daten zu dem Analyserechner kabelgebunden. Es besteht die Möglichkeit, die Messdaten mithilfe eines Transmittermoduls kabellos vom Sensor zur Auswerteeinheit zu übermitteln. Dies vereinfacht die Messung an schwer zugänglichen Orten und erspart eine Verkabelung. Durch den Einsatz von Energie-Harvestern, die Energie aus ihrer Umgebung entnehmen, kann eine solche Einheit auch energieautark betrieben werden, was u.a. Wartungsarbeiten in Form von Batteriewechseln überflüssig macht. So bietet z.B. das Energie- und Automatisierungstechnikunternehmen ABB seit 2017 ein Sensorsystem an, welches direkt und ohne Verkabelung am Gehäuse bestehender Niederspannungsmotoren und Pumpen angebracht wird. Es liefert Zustandsinformationen auf Basis von Schwingungs- und Temperaturmessungen und überträgt die Daten über BLE (Bluetooth Low Energy) an das Smartphone des Anwenders. In einer von ABB bereitgestellten App erfolgen eine Visualisierung der Informationen und Handlungsempfehlungen zur Instandhaltung. [2]

Bisherige Projekte

Zu diesem Themenkomplex der zustandsabhängigen und vorrauschauenden Instandhaltung mithilfe energieautarker drahtloser Sensorsysteme wurde in der

Vergangenheit an der Professur für Industrielle Messtechnik an der HTWK das Projekt TRAINCON realisiert.

Das Ziel dieses Projektes bestand in der Zustandsüberwachung von Bauteilen schienengebundener Fahrzeuge wie z.B. Wälzlagern. Es sollen Ermüdungs- und Verschleißprozesse vor Ausfall der Komponenten erkannt werden, wodurch im Rahmen einer zustandsabhängigen oder vorausschauenden Instandhaltung eingegriffen werden kann. Es wurde ein funkbasiertes, energieautarkes Sensorsystem entwickelt, welches die zustandsbeschreibenden Parameter an eine Basisstation sendet. In dieser erfolgt eine Analyse der Daten mit Methoden der kennlinienorientierten Auswertung sowie der unscharfen Fuzzy-Klassifikation. Zudem wird eine automatisierte Entscheidungsfindung zur zustandsabhängigen Instandhaltung ermöglicht. [78] [36]

Diese Bachelorarbeit basiert auf dem Sensorsystem des Projektes TRAINCON und führt eine Untersuchung hinsichtlich kommunikationstechnischer Optimierungsmöglichkeiten durch. Eine nähere Betrachtung des Sensor- und Sendekonzepts und der dazu verwendeten Hardware erfolgt in Kapitel 3.

Aktuell wird das Projekt SmartTram realisiert, im Rahmen dessen auf der Basis von Schwingungs- und Ultraschallmessungen ein Mess- und Sensorsystem zur Diagnose von Straßenbahn-Komponenten entwickelt wird. Dieses ist nicht an die Straßenbahn gebunden. Unterstützt durch Simulationen wird eine Früherkennung von Abnutzungserscheinungen ermöglicht, welche die Grundlage für eine vorausschauende Instandhaltung bilden. [37]

3 Gegenstand der Untersuchung

In diesem Kapitel wird das bestehende Sensorkonzept analysiert. Es erfolgt eine Erläuterung der Hardware und Aufgaben der verschiedenen Komponenten in 3.1 sowie der Netzwerkkommunikation in 3.2. In 3.3 wird das Zusammenwirken der Komponenten als messtechnischem Basiskonzept vorgestellt. Eine Gesamtenergiebilanz wird in 3.4 aufgestellt, auf deren Basis Optimierungspotentiale abgeleitet werden.

3.1 Verwendete Hardwarekomponenten

3.1.1 Kommunikationszentrum und Zentralorgan des Sensornetzwerks

Der in Abbildung 2 in einer Variante ohne Gehäuse dargestellte Raspberry Pi 2 (RPI) ist mit einem Preis von ca. 35€ ein günstiger Einplatinenrechner und hat etwa die Grundfläche einer Kreditkarte. Er besitzt einen Quad-Core-Prozessor mit einer Taktfrequenz von 900 MHz und 1GB Arbeitsspeicher und weist damit Leistungsdaten leicht unterhalb eines günstigen Smartphones auf. Bspw. besitzt das Motorola Moto C zum Preis von etwa 70€ einen Quad-Core-Prozessor mit 1,1 GHz Taktfrequenz und 1GB Arbeitsspeicher. Der RPI enthält keine fest eingebaute Speichereinheit, sondern einen MicroSD-Karten-Steckplatz, in dem eine Speicherkarte mit dem Betriebssystem eingesteckt wird. Als Betriebssystem wird die an den Raspberry Pi angepasste Linux-Distribution Raspbian verwendet. Als Schnittstellen stehen u.a. vier USB-2.0-Buchsen, ein Videoausgang über HDMI z.B. zum Anschließen an einen Bildschirm, je eine 3,5mm-Klinkenbuchse für Stereoaudio- und Composite-Video-Ausgang sowie eine 40-polige GPIO-Stiftleiste zur Verfügung. [21] GPIOs (General Purpose Input Output) sind programmierbare Ein- und Ausgänge, welche als Schnittstelle zu anderen Systemen bspw. zur Datenübertragung oder zum Ansteuern von LEDs genutzt werden können und teilweise mit speziellen Funktionen versehen sind. [22] Der RPI besitzt kein integriertes WLAN-Modul, sodass für das hier betrachtete Einsatzszenario ein in einer USB-Schnittstelle eingesteckter WLAN-Dongle benötigt wird.

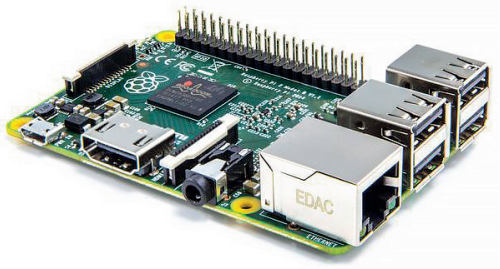


Abb. 2 Raspberry Pi 2 [21]

http://doi.org/10.33968/9783966270069.0003

Eine Aufgabe des Raspberry Pi stellt der Aufbau eines WLANs dar. WLAN (Wireless Local Area Network) ist ein drahtloses lokales Netzwerk, über welches Daten mittels Funk über einer Reichweite von etwa 70 m übertragen werden können. Der RPI nimmt die Funktion eines Routers ein. Ein Router ist ein Gerät, welches Datenpakete weiterleitet. Trifft ein Datenpaket ein, schlägt er in einer sog. Routingtabelle den besten Weg zum Ziel des Pakets nach und leitet es an die ermittelte Schnittstelle weiter. Es wird eine Datenübertragung zwischen den sich in diesem Netzwerk befindlichen Geräten ermöglicht. Zudem werden Filter- und Firewall-Funktionen übernommen, welche vor ungewollten Netzwerkzugriffen schützen. Zusätzlich dienen Router zumeist zur Anbindung des Internets an das lokale Netzwerk. Diese Funktion wird im vorliegenden Sensorkonzept nicht genutzt. Die Kommunikation erfolgt ausschließlich über das lokale Netzwerk. [11]

Der Raspberry Pi übernimmt zudem eine zweite Funktion und dient als Webserver. Ein Webserver übermittelt auf Anfrage Dokumente, z.B. Bestandteile einer Webseite, an den anfragenden Client. Als Client wird eine Anwendung bezeichnet, welche mit einem Server kommuniziert. Er ist beispielsweise ein Webbrowser, der die angefragten Informationen interpretiert und anschließend als Webseite anzeigt. Ein Webserver kann zudem zur Verschlüsselung der Server-Client-Kommunikation und zur Zugriffsbeschränkung eingesetzt werden, bei der sich ein Client für Zugriffe authentifizieren muss. [1]

Die Realisierung des Webservers auf dem RPI erfolgt mittels der kostenfreien Software Apache, welche 2017 bei etwa 46 % aller aktiven Webseiten eingesetzt wurde. [49] Apache unterstützt eine Vielzahl von Modulen wie bspw. Schnittstellen zu Skriptsprachen wie z.B. PHP (Personal Homepage) oder Verschlüsselungs- und Authentifizierungsmodule, welche die Kernfunktion des Webservers erweitern. Wie viele Anfragen gleichzeitig der Webserver bearbeiten kann, ist abhängig vom Arbeitsspeicher des Webservers. Ist die maximal mögliche Anfragenanzahl überschritten und der Arbeitsspeicher vollständig ausgelastet, stürzt der Webserver ab. Um dies zu verhindern, wird mit der Standardkonfiguration von Apache die maximale Anzahl anfragender Clients auf 256 begrenzt. [35] [46]

Als Basissprache zur Erstellung einer Webseite dient HTML (Hypertext Markup Language), womit der strukturelle Aufbau einer Webseite festgelegt wird. Durch Einbindung von CSS-Dokumenten (Cascading Style Sheets) werden Layout, Farben, Schriftgrößen usw. bestimmt. In ein HTML-Dokument lässt sich die Skriptsprache PHP einbinden, deren Programmcode auf dem Server ausgeführt wird. Es wird die Erstellung eines dynamischen Inhalts ermöglicht, der in dem Moment der Anfrage vom Client durch Aufrufen des PHP-Skriptes erzeugt wird. Zudem können durch Nutzereingaben erhaltene Daten mittels PHP verarbeitet und abgespeichert werden. [6] [54]

Im bestehenden Sensorkonzept wurde mittels eines Editors eine lokale Homepage erstellt. Es erfolgte keine Verwendung eines CSS-Dokuments, da das Aussehen der Webseite für den Einsatzzweck nicht relevant ist. Der Webserver gibt im Netzwerk ein PHP-Skript frei, mit dem das Abspeichern der Messdaten in eine CSV-Datei (CSV – comma-separated values) erfolgt. Eine CSV-Datei ist eine Textdatei, in der Daten von Kommas und Zeilenumbrüchen getrennt gespeichert werden.

3.1.2 Leitwarte

Bei der Leitwarte handelt es sich um einen Laptop. Die Verarbeitung und Visualisierung der Messdaten wird auf der Leitwarte mittels einer Matlab-Applikation durchgeführt. Matlab ist eine lizenzpflichtige Software, die vor allem zum Lösen mathematischer Probleme auf Basis numerischer Berechnungen, zur grafischen Darstellung der Lösungen sowie zur Datenerfassung, -analyse und -auswertung verwendet wird. [30]

Die mit der Leitwarte erzeugte Visualisierung zeigt Abbildung 3. Es werden aus dem Beschleunigungssignal die Spitzen- und Effektivwerte berechnet und mit diesen durch Multiplikation der Diagnosekennwert k(t) gebildet. Es erfolgt die Anzeige des Zeitsignals der Messwerte. Über eine Fast-Fourier-Transformation (FFT) wird zudem aus dem Zeitsignal das Frequenzspektrum der Schwingbeschleunigung berechnet. Auf Basis des Frequenzspektrums können Rückschlüsse auf den Bauteilzustand gezogen werden, da Schädigungen zu charakteristischen Veränderungen des Frequenzspektrums führen. Es wird eine Diagnose- und Handlungsempfehlung in Ampelform gegeben. Den Ampelfarben liegen definierte Amplitudenwerte der Schwingbeschleunigung zu Grunde. Bei Überschreitung der hinterlegten Amplitudenwerte färbt sich die Ampel gelb bzw. rot. Gelb steht für einen auffälligen Maschinenzustand, rot für einen kritischen.

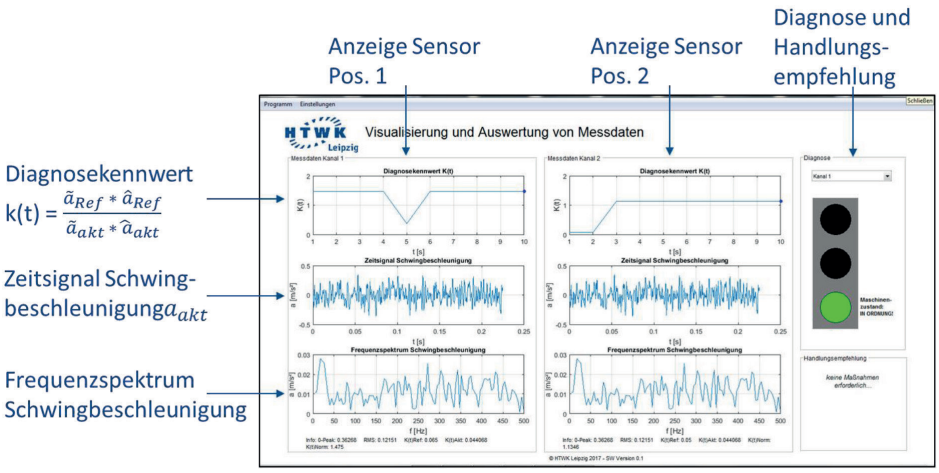


Abb. 3 Visualisierung der kennwertorientierten Zustandsdiagnose auf der Leitwarte [58]

3.1.3 Sensorknoten

Die Aufgabe der in Abbildung 4 dargestellten Einheit Sensorknoten besteht in der Aufnahme von Beschleunigungsmesswerten durch einen Sensor und dem Versenden dieser Werte durch ein Transceivermodul an den Webserver. Die Energieversorgung

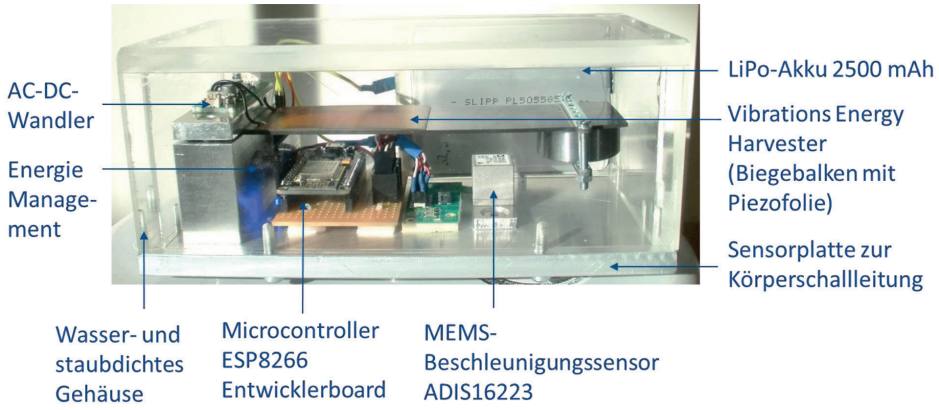


Abb. 4 Bestandteile Sensorknoten [58]

übernimmt ein Vibrations-Harvester, welcher ungenutzte Vibrationsenergie aus der Umgebung des Sensorknotens gewinnt und einen Lithium-Polymer-Akku speist.

Beschleunigungssensor

Zum Messen der Schwingbeschleunigung wird der in Abbildung 5 zu sehende Sensor ADIS 16223 von Analog Devices eingesetzt, welcher ein kapazitiver triaxialer MEMS-Beschleunigungssensor ist. MEMS steht für **m**ikro**e**lektro**m**echanische **S**ysteme. Dies ist ein Überbegriff für miniaturisierte Systeme, welche mechanische und elektronische Komponenten mit Abmessungen im Mikrometerbereich in einem Bauteil vereinen. Der Sensor weist Abmessungen von (1,5 × 1,5 × 1,5) cm auf, was einer platzsparenden Ausführung des Sensorknotens entgegenkommt. [4]

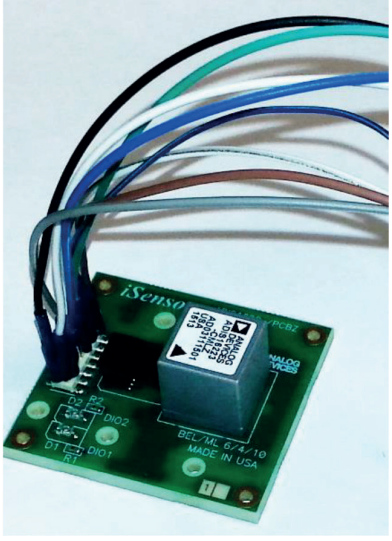


Abb. 5 Beschleunigungssensor ADIS 16223 [eigene Aufnahme]

Den prinzipiellen Aufbau eines kapazitiven MEMS-Beschleunigungssensor zeigt Abbildung 6. Grundlage eines solchen Sensors ist ein seismisches Feder-Masse-System, bei dem ein beweglicher Rahmen eine elastisch aufgehängte Masse bildet und mit einer Platte eines Differentialkondensators verbunden ist. Diese Platte ist in Abbildung 6 mit ② bezeichnet. Die anderen Platten (①) sind fest mit dem Gehäuse verbunden. Sie bilden zusammen mit der beweglichen Platte zwei Plattenkondensatoren mit den Kapazitäten C_1 und C_2, welche zusammen einen Differentialkondensator darstellen. Zwischen den Elektroden befindet sich ein Dielektrikum. Der Plattenabstand d_0 zwischen ① und ② eines Kondensators verringert sich um dem gleichen Betrag Δd, wie sich der Abstand des anderen vergrößert. Erfährt der Sensor eine Beschleunigung, ändern sich die Kapazitäten der beiden Kondensatoren symmetrisch, da folgender Zusammenhang zwischen Kapazität und Plattenabstand besteht [7] [29]:

$$C_{1/2} = \frac{\varepsilon * A}{d_0 \pm d} \quad [67] \tag{3.1}$$

$C_{1/2}$ ... Kapazität Plattenkondensator 1 bzw. 2
ε ... Permittivität
A ... Flächeninhalt der Kondensatorplatten
d_0 ... Plattenabstand im unbeschleunigten System
d ... Änderung des Plattenabstandes durch einwirkende Beschleunigung

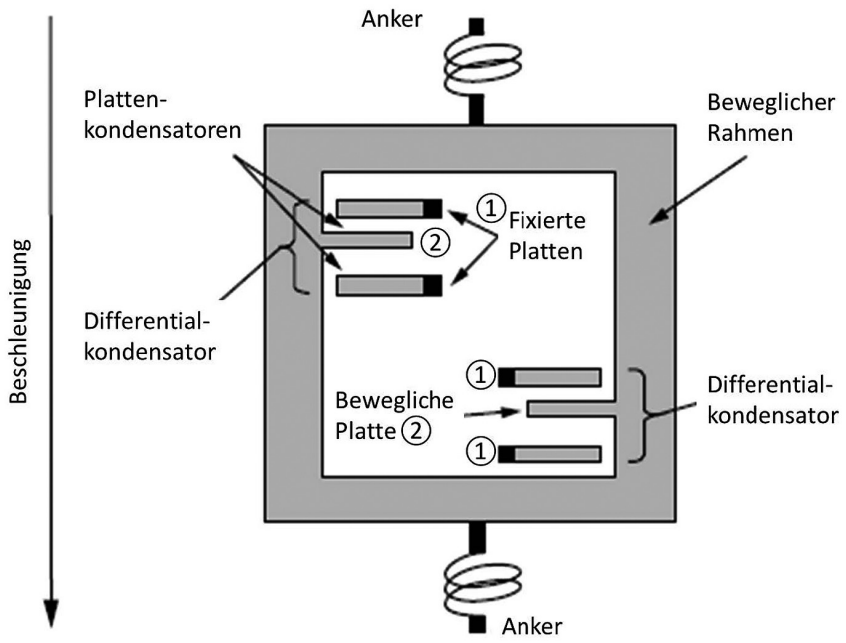


Abb. 6 prinzipieller Aufbau kapazitives MEMS-Beschleunigungssensor-Element nach [4]

Diese gegensinnige Kapazitätsänderung lässt sich mit einer Brückenschaltung auswerten. Es wird eine Spannung erzeugt, die proportional zur Auslenkung Δd und damit zu der die Masse beschleunigenden Kraft ist. Die Kapazitäten dieser Differentialkondensatoren liegen im Bereich von Picofarad (10^{-12}), die Kapazitätsänderungen im Bereich einiger Femtofarad (10^{-15}). [70]

Der eingesetzte Beschleunigungssensor ADIS 16223 besitzt für jede der drei Achsen einen kapazitiven MEMS-Beschleunigungsaufnehmer, welche jeweils eine Messung im Bereich von –70 g bis +70 g mit einer Abtastrate von bis zu 72,9 kSPS (72.900 Abtastvorgänge pro Sekunde) ermöglichen. Der Sensor benötigt eine Betriebsspannung zwischen 3,15V und 3,6V. Er verbraucht im arbeitenden Zustand bei 3,3 V 43 mA und kann nach einer Messung in einen stromsparenden Sleep-Modus versetzt werden, bei welchem 230 µA fließen. [4]

Abbildung 7 zeigt den vereinfachten Aufbau der Signalverarbeitung. Die Kommunikation zwischen Sensor und Transceivermodul erfolgt über ein SPI-Interface. SPI ist ein Bussystem zur synchronen seriellen Datenübertragung mit vier Kanälen, welche in Abbildung 7 zu erkennen sind. Das Chip Select-Signal (CS) aktiviert die Kommunikation, die Serial Clock (SCLK) synchronisiert die Datenübertragung und über die DIN- (Data Input) sowie DOUT-Kanäle (Data Output) erfolgt der direkte Austausch von binären Informationen mittels 16-Bit-Sequenzen. Die SPI-Schnittstelle dient weiterhin zur Übertragung von Filtereinstellungen zum Sensor, zum Aufwecken des Sensors aus dem Sleep-Modus und zur Auslösung einer Messung. [4]

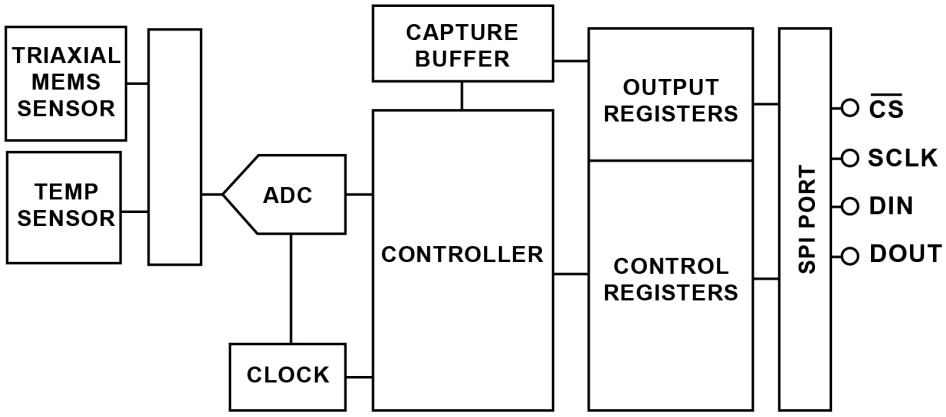


Abb. 7 vereinfachte Übersicht der Signalverarbeitung des ADIS 16223 [4]

Das Auslösen einer Messung kann zudem optional über ein binäres Signal erfolgen, welches vom Transceivermodul ausgekoppelt und am Sensormodul eingelesen wird. Der Sensor verfügt dazu über zwei konfigurierbare GPIO-Anschlüsse. Ein zeit- oder ereignisgesteuertes Auslösen der Messdatenerfassung ist ebenfalls möglich. [4]

Das analoge Spannungssignal des kapazitiven Messaufnehmers wird nach dem Auslösen einer Messung durch einen integrierten 16-Bit-Analog-Digital-Wandler

(engl. Analog-Digital-Converter – ADC) in diskrete digitale Werte umgewandelt. Diese Werte werden vom integrierten Controller verarbeitet und im Erfassungspuffer (engl. Capture-Buffer) zwischengespeichert. Durch Ansprechen des Output-Registers können die digitalen Beschleunigungswerte ausgelesen werden. Für jede Achse existiert ein Capture-Buffer, welcher 1024 Werte zwischenspeichern kann. Optional kann der sog. extended-Modus aktiviert werden, wodurch die Capture-Buffer der anderen Achsen mitgenutzt und so für eine Achse 3072 anstatt 1024 Werte zwischengespeichert und ausgelesen werden. [4]

WLAN–Transceivermodul

Es wird als Transceivermodul der Microcontroller ESP8266 eingesetzt, welcher einen 32-Bit-Prozessor mit einer Taktfrequenz von 80 MHz besitzt. Das integrierte WLAN-Modul erlaubt als WLAN-Station einen drahtlosen Versand von Nachrichten. Es kann zudem als Access Point dienen, welcher als Netzwerkknoten das Netzwerk erweitert und die Reichweite vergrößert. Es ermöglicht den Zugang für andere Endgeräte zum WLAN und damit ein sog. vermaschtes Netzwerk. Der verwendete Microcontroller beinhaltet 4 MByte internen Speicher und etwa 50 kByte Arbeitsspeicher. Der begrenzte Speicherplatz erfordert eine ressourcenschonende Programmierung und limitiert die Anzahl gleichzeitig ausführbarer Aufgaben. Die serielle Kommunikation mit anderen Geräten kann u.a. über ein SPI-Interface erfolgen. Der Microcontroller wird mit 3,3 V betrieben und kann deshalb die gleiche Spannungsquelle wie der Beschleunigungssensor nutzen. [25]

Zum Versenden von Daten über WLAN verbraucht das Transceivermodul bei 3 V Versorgungsspannung laut Datenblatt [25] zwischen 120 und 170 mA, beim Empfangen etwa 50 mA. Werden andere Aufgaben erledigt, bei denen das WLAN-Modul nicht benötigt wird, liegt die Stromaufnahme im Durchschnitt bei 80 mA. Um energiearme Einsatzszenarien zu ermöglichen, kann der Microcontroller zwischen zwei Arbeitszyklen in verschiedene Stand-by-Zustände versetzt werden, welche als sog. Schlafzustände (engl. Sleep-Mode) bezeichnet werden. Im Modem-Sleep-Modus verbraucht das Transceivermodul 15 mA. Das WLAN-Modul wird abgeschaltet. Der Prozessor arbeitet aber weiter, sodass eine kabelgebundene Datenübertragung möglich ist. Im Light-Sleep-Modus kann ein Verbrauch von 0,9 mA erreicht werden, da der Prozessor in einen Stromsparmodus versetzt wird. Um in einen Arbeitszustand zurückzukehren, werden 3 ms benötigt. Eine dritte Möglichkeit stellt der Deep-Sleep-Modus dar, bei dem bis auf die interne Uhr alle Funktionen ausgeschaltet sind. Es wird mit einem Verbrauch von 20 µA ein energiearmer Zustand erreicht. Zwischen 0,3 und 1 s werden benötigt, um nach einer vorher festgelegten Zeitspanne wieder einen Arbeitsmodus einzunehmen. [25] [27]

Zum vereinfachten Einsatz wurde ein Entwicklerboard von NodeMCU verwendet. Dieses ermöglicht es, durch einen integrierten Spannungswandler das Modul über eine USB-Schnittstelle zu betreiben, welche eine Spannung von 5 V liefert und über die die Programmierung des Moduls erfolgen kann. Abbildung 8 zeigt das verwendete NodeMCU-Entwicklerboard, welches Abmessungen von (4,2 x 3 x 1,3) cm aufweist.

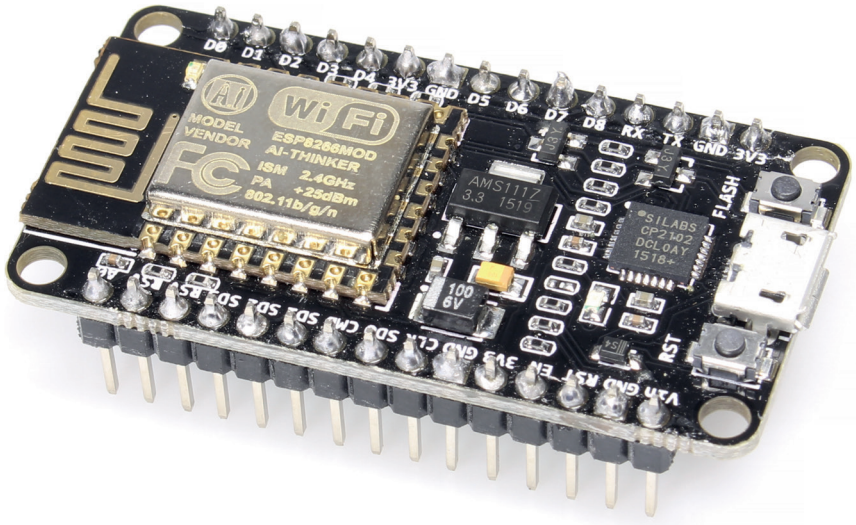


Abb. 8 NodeMCU-Entwicklerboard mit ESP8266 [75]

Für die Entwicklung des Programms, welches auf dem Microcontroller ausgeführt wird, können unterschiedliche Programmiersprachen verwendet werden. Dabei wird grundsätzlich in Compiler- und Interpretersprachen unterschieden. Der Unterschied zwischen den beiden Ansätzen besteht darin, wie und wann die lesbaren Programmierbefehle in für den Prozessor verständliche Maschinenbefehle umgesetzt werden. Bei Compilersprachen wird durch einen Compiler im Voraus der Programmausführung einmalig das eigentliche Programm erstellt, dessen Anweisung direkt vom Prozessor ausgeführt werden. Interpretersprachen führen mit einem Interpreter eine Zwischenschicht zwischen Programmiersprache und Prozessor ein. Der Interpreter stellt eine Software-Bibliothek dar, welche zur Laufzeit des Programmes den ausführbaren Maschinencode generiert. Kompilierte Programme werden schneller ausgeführt. Im Gegenzug ist die Programmentwicklung bei Interpretersprachen einfacher, da die Programme sofort getestet werden können, was die Fehlersuche erleichtert. [18]

Verwendet wurde bei dem vorliegenden Sensorkonzept die Interpretersprache LUA. Für LUA sind Module bspw. für das Aufbauen einer WLAN- und SPI-Verbindung, sowie für einige Übertragungsprotokolle wie HTTP und MQTT vorhanden, welche eine objektorientierte Programmierung ermöglichen. Nach Erstellung eines finalen Messprogramms lässt sich dieses auch als kompiliertes Programm auf den Microcontroller spielen.

An das Transceivermodul werden die folgenden Anforderungen gestellt: Es soll erstens mit dem Beschleunigungssensor über eine SPI-Verbindung kommunizieren und den Ablauf einer Messung steuern. Zweitens sollen die Messwerte über ein WLAN drahtlos versendet werden. Zudem soll das Modul sich in der Zeit zwischen

den Messzyklen in einen energiearmen Zustand versetzen lassen. Der Microcontroller ESP8266 erfüllt mit dem SPI-Interface, dem WLAN-Modul und der Möglichkeit einen energieeffizienten Sleep-Modus einzunehmen diese Anforderungen.

Energieversorgung
Die Energieversorgung des Sensorknotens wird durch einen Lithium-Polymer-Akkumulator mit 2500 mAh Kapazität realisiert. Ein Aufladen mittels Ladegerät oder ein Austausch des Akkus ist im laufenden Fahrbetrieb oder bei schwer zugänglicher Einbaulage des Sensorknotens nur mit Aufwand möglich. Es wird deshalb ein Energiewandler (engl. Energy Harvester) zum Aufladen des Akkumulators eingesetzt, welcher ungenutzte Energie aus der Umgebung „erntet" und diese in nutzbare Energie umwandelt. Je nach Umgebungsbedingungen können verschiedene Energiequellen wie Wärme, Licht oder Vibration genutzt werden.

Die Wandlung von Wärme in elektrische Energie basiert auf dem Seebeck-Effekt, welcher ein elektrisches Potential in einem Leiter aufgrund von Temperaturgradienten beschreibt. Wie groß dieses Potential ist, hängt vom Material des Leiters ab und wird durch den Seebeck-Koeffizienten beschrieben. Werden zwei Werkstoffe mit unterschiedlichen Seebeck-Koeffizienten eingesetzt, entsteht eine Potentialdifferenz zwischen den beiden Materialien. Die erzeugte Spannung beträgt wenige mV. Deshalb werden bei solchen Wandlern viele Thermopaare aus n- und p-dotierten Halbleiterbeinen zusammengeschaltet, womit Spannungen im Bereich einiger hundert mV erzeugt werden können. Für Mikrowandler liegt die Leistung bei einer Temperaturdifferenz von 10 K zwischen 0,1 und 2 mW. [8] [67]

Eine weitere Möglichkeit ist die Wandlung von Licht mittels Solarzellen in elektrische Energie, was z.B. zur Stromerzeugung kommerziell eingesetzt wird. Im Jahr 2017 wurde 6,1 % des in Deutschland erzeugten Stroms durch Photovoltaik gewonnen. [64] Ein miniaturisierter Einsatz erfolgt in Taschenrechnern schon seit den 1980er-Jahren und ist auch zur Versorgung eines Funksensorknotens möglich. Die tatsächlich verfügbare Energie lässt sich allerdings nur grob abschätzen, da diese erheblich von der räumlichen Orientierung der Zelle, des Lichtspektrums und der verfügbaren Lichtintensitäten abhängt. Im Außenbereich beträgt über das Jahr gemittelt die verfügbare Leistung einige mW je cm^2 Fläche der Zelle. Im Innenbereich ist die maximal verfügbare Lichtintensität deutlich geringer, dafür ist die Lichtintensität weitgehend konstant, sodass sich eine verfügbare Leistung von etwa 15 µW je cm^2 ergibt. [67]

Eine Wandlung von kinetischer Energie, die z.B. in der Umgebung in Form von Vibrationen enthalten ist, in elektrische Energie ist ebenfalls möglich. Die kinetische Energie aus der Umgebung wird auf eine Masse übertragen. Durch die Relativbewegung dieser Masse bezüglich eines Trägersystems kann unter Ausnutzung des induktiven, kapazitiven oder piezoelektrischen Effekts eine elektrische Leistung zur Verfügung gestellt werden, welche maximal ist, wenn die Systeme in Eigenfrequenz schwingen. Die meisten Systeme besitzen eine, aufwendigere Systeme mehrere oder einstellbare Eigenfrequenzen, da sich schon bei recht geringen Abweichungen der Anregungsfrequenz von der Eigenfrequenz die Leistungsausbeute erheblich verringert. [67] [14]

Wird ein piezoelektrischer Werkstoff einer Krafteinwirkung wie Zug oder Druck ausgesetzt, verschieben sich die Schwerpunkte der positiven und der negativen Ladungen im Kristallgitter. Die dadurch erfolgte Veränderung der elektrischen Polarisation sorgt für das Entstehen einer elektrischen Spannung. Die dabei erzeugte elektrische Ladung ist proportional zur mechanischen Spannung, also damit proportional zur anliegenden Beanspruchung. Als piezoelektrischer Werkstoff werden zumeist polykristalline ferroelektrische Keramiken wie z.B. Blei-Zirkonat-Titanat (PZT) oder polymeres Polyvinylidenflourid (PVDF) verwendet. Bei solchen Vibrations-Wandlern kommt oft ein mit einem piezoelektrischen Material beschichteter Biegebalken zum Einsatz, welcher auf einer Seite eingespannt ist. Die freie Seite wird in Schwingung versetzt, wodurch Dehnungen und Stauchungen in dem Material auftreten und eine Ladungsverschiebung verursachen. Die Eigenfrequenz des Harvesters muss zu der Erregerfrequenz des Anwendungsfalls passend ausgelegt sein. Je nach Größe des auf dem Biegebalken aufgebrachten piezoelektrischen Materials und der Auslenkung können dann Leistungen im Bereich von 0,5 bis 1 mW erreicht werden. Ein solches System mit einer Schwungmasse am freien Ende zum Erhöhen der einwirkenden Kraft ist in Abbildung 9 dargestellt. [40] [29] [61]

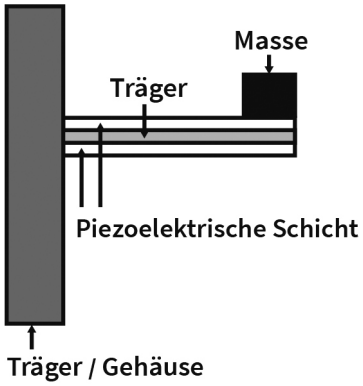


Abb. 9 prinzipieller Aufbau eines Biegebalkens mit piezoelektrischem Material [67]

Bei dem vorliegenden Sensorkonzept wurde ein piezoelektrischer Vibrations-Harvester in Form eines Biegebalkens mit Piezofolie ausgewählt. Die Verwendung von Solarzellen scheidet aufgrund der schlechten Lichtverhältnisse und des Verschmutzungsgrades am Anbringungsort des Sensorknotens unter einer Straßenbahn aus. Bei der Nutzung eines Thermowandlers besteht vor allem im wärmeren Teil des Jahres das Problem, dass die Temperaturdifferenzen für eine ausreichende Energieversorgung nicht groß genug sind.

3.2 Netzwerkkommunikation mit HTTP

Die im Sensorknoten gewonnenen Messdaten sollen über das WLAN versendet werden. Damit eine Datenübertragung zwischen zwei Endgeräten stattfinden kann,

werden Protokolle benötigt, nach denen die Kommunikation erfolgt. Anwendungsprotokolle sind eine Sammlung standardisierter Regeln. Sie steuern die Verbindung sowie den Ablauf des Datenaustauschs zwischen zwei Endgeräten. [20]

Als ein solches Anwendungsprotokoll wurde im bestehenden Sensorkonzept HTTP (**H**yper**t**ext **T**ransfer **P**rotocol) verwendet. Der Austausch von Daten basiert bei diesem Protokoll auf einer Client-Server-Logik, was bedeutet, dass der Aufbau der Kommunikation nur in einer Richtung möglich ist. Der Server nimmt Anfragen entgegen und beantwortet diese. [77]

Abbildung 10 zeigt den Ablauf einer Datenübertragung mittels HTTP. Clients sind Anwendungen, welche mit dem Server kommunizieren. Im vorliegenden Konzept stellen der Sensorknoten und die Leitwarte Clients dar. Zunächst fragt der Client über eine URL (z.B. http://localhost/php/FWRITE_3.php) oder die IP-Adresse (z.B. 10.10.0.1) des Servers eine Verbindung an. Diese wird entweder negiert oder bestätigt. Wenn die Verbindung bestätigt wurde, sendet der Client eine Anfrage (engl. Request) an den Server. Diese besteht aus einem Nachrichtenkopf, einem sog. Header, an welchen als Nachrichtenrumpf Daten angefügt werden.

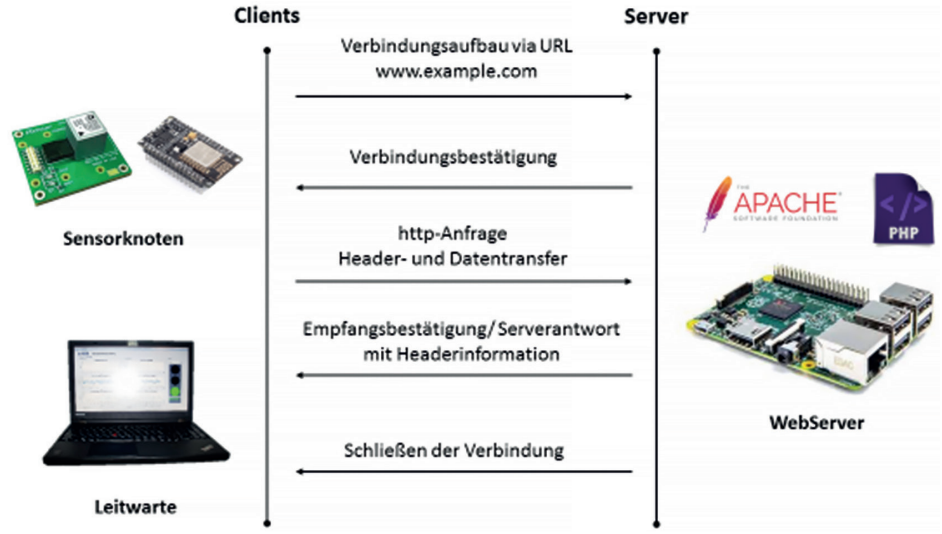


Abb. 10 Verbindungsablauf HTTP

Die konkrete Request-Nachricht des Sensorknotens an den Webserver ist in Abbildung 11 dargestellt, welche mit Ausnahme der letzten Zeile ausschließlich aus dem Header besteht. Auf den Header folgen die zu übertragenden Daten. Die erste Zeile enthält die Anfragemethode und die URL des Servers. Hier erfolgt eine Übermittlung an das auf dem Server freigegebene PHP-Skript zur weiteren Datenbehandlung. Der Header beinhaltet weiterhin Informationen wie die verwendete HTTP-Version, die IP-Adresse des Servers, ob die Verbindung anschließend geöffnet bleibt oder

Abb. 11 HTTP-Anfrage des Sensorknotens an den Webserver

geschlossen werden soll, sowie Angaben über das Format und die Länge der mit der letzten Zeile der Anfrage angebundenen Daten.

Schon ohne Daten müssen aufgrund des Headers je Nachricht etwa 200 Zeichen versendet werden. Ein Zeichen belegt dabei zwischen einem und vier Byte. Wird vereinfacht eine durchschnittliche Zeichengröße von 2,5 Byte angenommen, ergibt sich eine Headergröße von 500 Byte. Die maximal zulässige Länge einer Request-Nachricht liegt mit Verwendung eines Apache-Webservers bei 16 kByte. [26] Durch den Header wird bereits ohne angehangene Messinformationen eine Nachricht von etwa 3 % der maximal zulässigen Größe übertragen. Eine Zeichenkette in LUA besitzt eine maximale Länge von 2 kByte. Eine Nachricht wird als Zeichenkette aus dem Header und den zu übertragenden Daten zusammengesetzt, wobei der Header 25 % der zulässigen Nachrichtengröße verursacht.

Der Server antwortet mit einer Response-Nachricht, die aus Headerinformationen und falls mittels Request angefordert aus einem Dokument besteht. Abbildung 12 zeigt ein Beispiel eines Response-Headers. Dieser enthält z.B. Informationen wie einen Statuscode und den Zeitpunkt der Antwort. Sollen weitere Daten gesendet werden, muss eine erneute Anfrage des Clients nach dem beschriebenen Schema erfolgen.

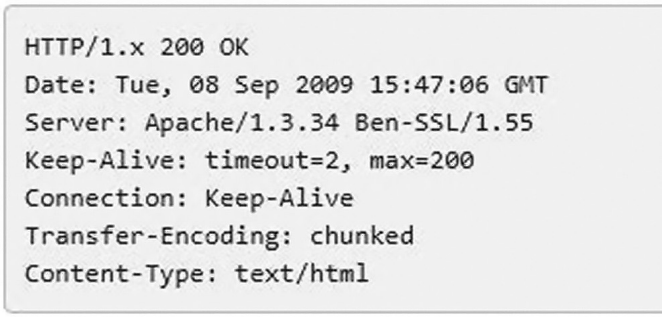


Abb. 12 Beispiel eines Response-Headers [19]

Soll die Verbindung nach der erfolgten Antwort des Servers geöffnet bleiben, um weitere Daten ohne erneuten Verbindungsaufbau zu übertragen, muss dies mit der

Request-Nachricht vom Client angefordert werden. In Abbildung 11 erfolgt das mit dem Ausdruck „Connection: keep-alive“ in der vierten Zeile. Es ist abhängig vom Webserver und dessen Konfiguration, ob und für wie viele Anfragen in welchen Zeitabständen die Verbindung erhalten bleibt. Diese Informationen sind Teil des Response-Headers. Im Beispiel von Abbildung 12 zeigt die vierte Zeile, dass die Verbindung beendet wird, wenn zwischen den Requests mehr als 2 s liegen oder mehr als 200 Anfragen den Webserver erreicht haben. Mit dem darauffolgenden „keep-alive“ wird bestätigt, dass die Verbindung geöffnet bleibt. [59]

Die maximal mögliche Anzahl an Request-Nachrichten, welche innerhalb einer Verbindung gesendet werden können, wird durch die verfügbaren Ressourcen des Webservers bestimmt. Für das Aufrechterhalten der Verbindung wird Arbeitsspeicher beansprucht. Werden nur weitere Verbindungen geöffnet ohne alte zu schließen, wird der Webserver deshalb nach einer gewissen Zeit überlastet sein. Deshalb ist es notwendig, die Anzahl der möglichen Anfragen oder die Zeitabstände zwischen einzelnen Requests zu limitieren und die Verbindung anschließend zu schließen. [44]

Das verwendete HTTP-Protokoll gilt als ein sehr zuverlässiges und sicheres Protokoll. Die Daten werden nur zu einem definierten Ziel übertragen und es erfolgt eine Rückmeldung, ob diese dort auch angekommen oder ob sie korrumpiert worden sind. Aufgrund des großen Datenheaders, des Zeitverzugs durch die vielen Zwischenschritte und des Schließens der Verbindung nach einer gewissen Zeitdauer oder Anzahl an Requests können keine Echtzeitübertragungen oder Datenstreams realisiert werden. Es ist mit HTTP demzufolge nur sinnvoll, größere Datenmengen in Paketen anstatt einzelner Werte zu senden. Eine Kommunikation findet nur zwischen genau einem Client und genau einem Server statt. Es ist nicht möglich, dass die gleiche Nachricht an viele Empfänger gleichzeitig versendet wird. [53]

3.3 Beschreibung des messtechnischen Basiskonzepts

In diesem Kapitel erfolgt eine Beschreibung der Rahmenbedingungen der Messungen. Anschließend wird das Zusammenwirken der Komponenten im Gesamtablauf einer Messung dargestellt.

Mit dem bestehenden Sensorsystem sollen Ermüdungs- und Verschleißprozesse von Straßenbahn-Komponenten wie z.B. Wälzlagern erkannt werden. Schwingungsparameter eignen sich gut als zustandsbeschreibende Merkmale von Komponenten wie Wälzlagern hinsichtlich Verschleiß und Ermüdung. Es erfolgt eine Messung der Schwingbeschleunigung des Körperschalls an den Gehäusen von Getriebe und Radlager einer Antriebseinheit der Straßenbahn. Als weitere Parameter werden Schwinggeschwindigkeit und Schwingweg benötigt, die sich durch Integration der Schwingbeschleunigung berechnen lassen. [78] Diese Schwingungsparameter eignen sich zur Zustandsbeschreibung, da aufgrund von Form- und Lageabweichungen, der Oberflächenbeschaffenheit, infolge von Unwuchten und den Betriebsbedingungen der Elemente eines technischen Systems mechanische Schwingungen entstehen. Diese erzeugen Schallwellen, welche sich u.a. als Körperschall auf die

Gehäuse der Komponenten ausbreiten. Schädigungen bewirken Veränderungen des Schallfeldes einer Komponente und sind somit über eine geeignete Auswertung der zustandsbeschreibenden Parameter erkennbar. Teilweise befinden sich die Schädigungen auch im hörbaren Bereich. So deuten bspw. laute metallische Geräusche auf fehlenden Schmierstoff oder zu hohe Belastung hin. [10] [74]

Verschleiß- und Ermüdungsvorgänge finden bei Wälzlagern sehr langsam und über einen langen Zeitraum statt, sodass für eine Zustandsüberwachung nicht ununterbrochen gemessen werden muss. Vielmehr ist eine Vergleichbarkeit der gemessenen Daten relevant, auf die eine Vielzahl von Einflussgrößen wirken, wie z.B. Umweltbedingungen in Form von Feuchte und Temperaturschwankungen, die Verkehrslage, das Überfahren beschädigter Gleise sowie ein generell instationäres Betriebsverhalten mit vielen Beschleunigungs- und Bremsvorgängen. [78]

Im Vorfeld wurde ein geeigneter Gleisabschnitt identifiziert, welcher die Anforderungen bezüglich einer reduzierten Anzahl an Einflussgrößen erfüllt. Eine Vergleichbarkeit der Messwerte wird hergestellt, indem eine Messung immer am gleichen Abschnitt durchgeführt wird. Zur Positionsbestimmung wird der Raspberry Pi mit einem GPS-Modul ausgestattet. Der RPI fragt laufend die Position ab und spannt das WLAN auf, wenn der zuvor definierte Gleisabschnitt erreicht wird.

Abbildung 13 zeigt, wie eine Messaufnahme ausgelöst wird. Der Sensorknoten wacht nach einer Minute für wenige Sekunden auf und sucht das WLAN. Dieser Vorgang wird im weiteren Verlauf der Arbeit als Grundzustand bezeichnet. Verläuft die Suche erfolglos, wird wieder der Sleep-Modus eingenommen. Ist der Aufbau einer WLAN-Verbindung erfolgt, kommuniziert das Transceivermodul mit dem Sensor

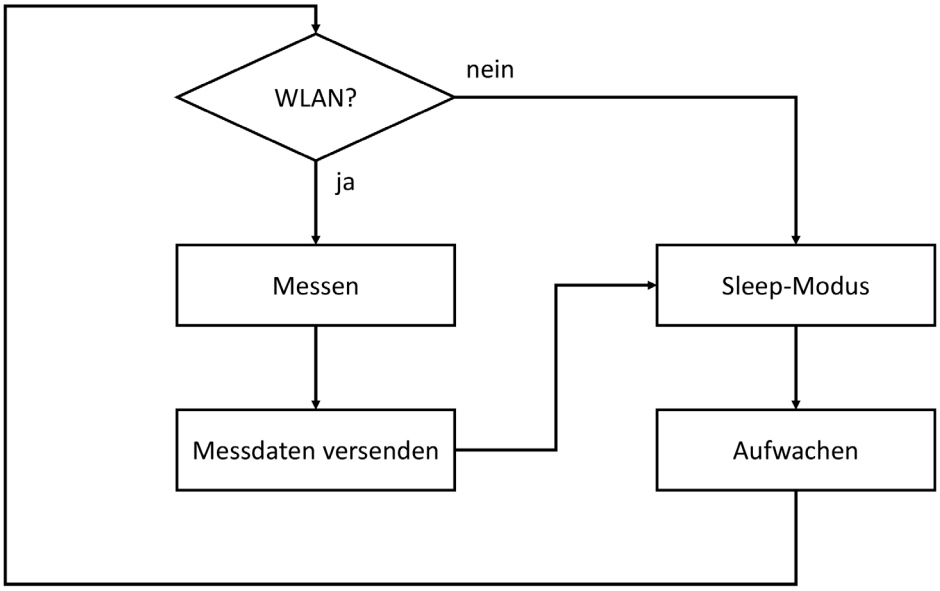


Abb. 13 Ablauf Messwertaufnahme mittels des Sensorknotens

über das SPI-Interface, weckt diesen aus dem Sleep-Modus auf und löst eine Messung aus. Der Beschleunigungssensor nimmt 3072 Werte auf und speichert diese im Capture-Buffer zwischen. Anschließend folgt der Sendevorgang, bei dem die Messwerte in Paketen mit je 128 Werten vom Microcontroller aus dem Capture-Buffer des Sensors ausgelesen und anschließend über das WLAN versendet werden. Sind alle Werte erfolgreich versendet wurden, werden Transceivermodul und Sensor wieder in den Sleep-Modus versetzt. In Abschnitt 3.4 werden die eben genannten Betriebszustände tabellarisch mit den laut Datenblättern zu erwartenden Stromverbräuchen erfasst. Zudem erfolgt eine grafische Darstellung des zu erwartenden Lastprofils.

Über das vom Raspberry Pi aufgespannte WLAN können die in Abbildung 14 dargestellten Komponenten Sensorknoten, Webserver und Leitwarte miteinander kommunizieren. Das Transceivermodul des Sensorknotens sendet die Messdaten an den Webserver, welcher sich auf dem RPI befindet. Auf diesem wird ein PHP-Skript aufgerufen, welches die Daten entgegennimmt, das Datenpaket in Einzelwerte trennt, Metainformationen wie bspw. die aktuelle Zeit hinzufügt und die Messwerte in einer CSV-Datei abspeichert. Der Webserver sendet mit einer Response-Nachricht die Bestätigung der Anfrage an den Sensorknoten zurück. Die Leitwarte schickt ein Request und erhält an die Antwort des Webservers als Nachrichtenrumpf angebunden die Datei mit den Messdaten, welche anschließend durch die Leitwarte verarbeitet, visualisiert und bewertet werden.

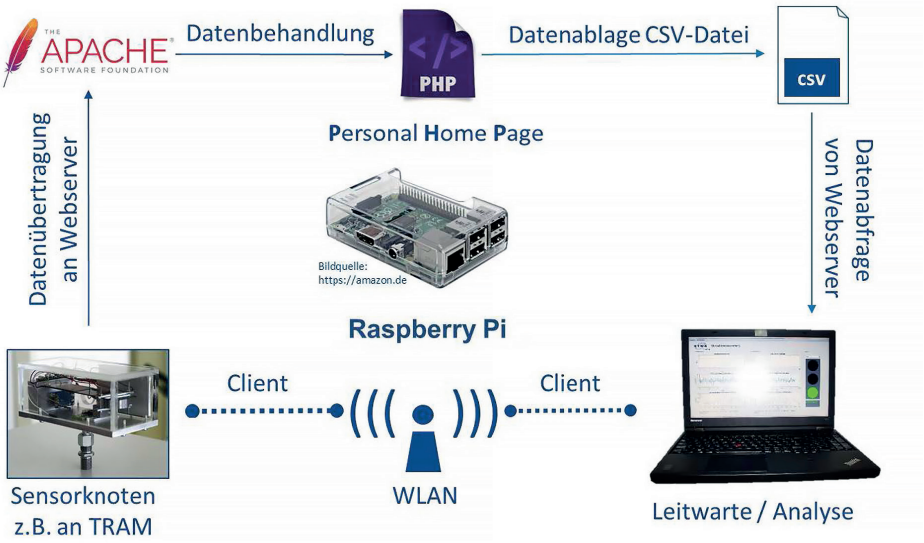


Abb. 14 Datenübertragung im bestehenden Sensorsystem [58]

Raspberry Pi und Leitwarte können so weit vom Sensorknoten entfernt sein, wie das WLAN mit einer Reichweite von etwa 70 m reicht. Im vorliegendem Konzept können deshalb mehrere Sensorknoten an multiplen Stellen unter der Bahn

gleichzeitig im Einsatz sein. Das Messkonzept mit großen Pausen zwischen den Messaufnahmen schafft in Kombination mit den möglichen Sleep-Modi der verwendeten Hardware die Grundlage für energiearme Zustände. Diese ermöglichen eine kabellose Stromversorgung des Sensorknotens sowie eine lange energetische Lebensdauer der eingesetzten Energieversorgung, welche aus einem mit Umgebungsenergie gespeisten Akku besteht. Der RPI und die Leitwarte befinden sich im Fahrgastraum der Straßenbahn, weshalb eine kabelgebundene Stromversorgung problemlos möglich ist.

3.4 Gesamtenergiebilanz des Sensorknotens und Optimierungspotentiale

In Abhängigkeit von den jeweiligen Betriebszuständen des Sensorknotens werden laut den Datenblättern [4] und [25] Verbräuche in den Größenordnungen der in der folgenden Tabelle 1 dargestellten Werte erwartet.

Tab. 1 Stromverbrauch Transceivermodul und Beschleunigungssensor

Zustand	Transceivermodul (bei 3V)	Beschleunigungssensor (bei 3,3V)	Summe
Grundzustand	80 mA		80 mA
Messen	80 mA	43 mA	123 mA
Sendebetrieb	120 – 170 mA	43 mA	163 – 213 mA
Sleep-Modus	20 µA	230 µA	250 µA

Im Rahmen des Traincon-Projektes wurde ein Simulationsmodell des Sensorknotens auf Basis von Datenblattwerten entwickelt. Abbildung 15 stellt das Lastprofil des simulierten Sensorknotens dar. Im unteren Diagramm ist zu erkennen, dass eine stündliche Messung simuliert wurde. Bei dieser findet kurzfristig eine Leistungsaufnahme von 580 mW statt. Das obere Diagramm zeigt den Leistungsbedarf des Sensorknotens bei einem Messzyklus im Detail. Nach dem Aufwachen befindet sich das Transceivermodul im Grundzustand. In diesem baut es innerhalb von etwa 5 s eine Verbindung zum WLAN-Netzwerk auf. Beim anschließenden Messvorgang werden die 3072 Messwerte im Capture-Buffer des Sensors zwischengespeichert. Im vorliegendem Messkonzept wird das Beschleunigungssignal mit 4,56 kHz abgetastet, sodass die 3072 Werte innerhalb von etwa 0,7 s aufgenommen werden. Diese Messwerte werden im Sendebetrieb ausgelesen und an den Webserver versendet. Durch eine Evaluierung der Simulation durch Messungen am realen Sensorknoten wurde eine Sendedauer von 15 s ermittelt. [34]

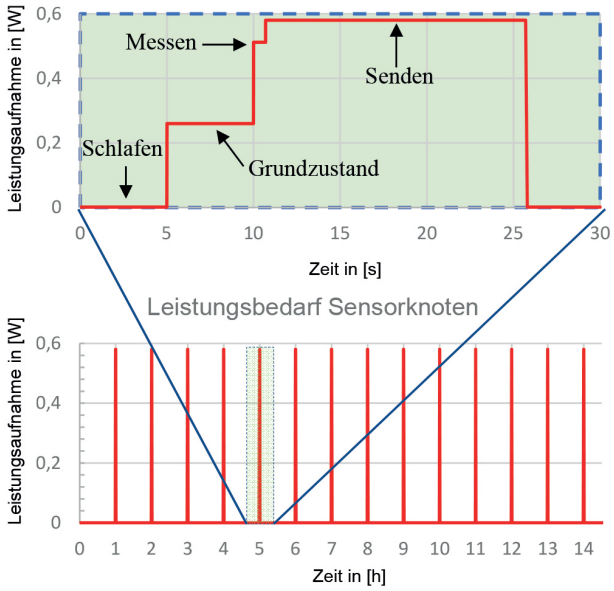


Abb. 15 simuliertes Lastprofil Sensorknoten

Für einen Betrachtungszeitraum von einem Tag mit stündlichen Messungen ergeben sich für den simulierten Sensorknoten die im Sankey-Diagramm in Abbildung 16 dargestellten Energieflüsse. Der Vibrations-Harvester kann den abfallenden Ladungszustand des Akkus nicht vollständig ausgleichen. Der Sensorknoten kann

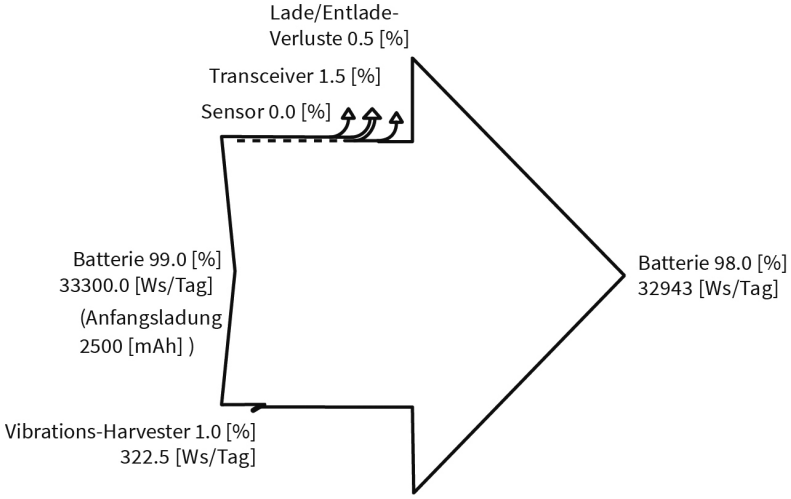


Abb. 16 Sankey-Diagramm mit Energieflüssen des bestehenden Sensorkonzepts [34]

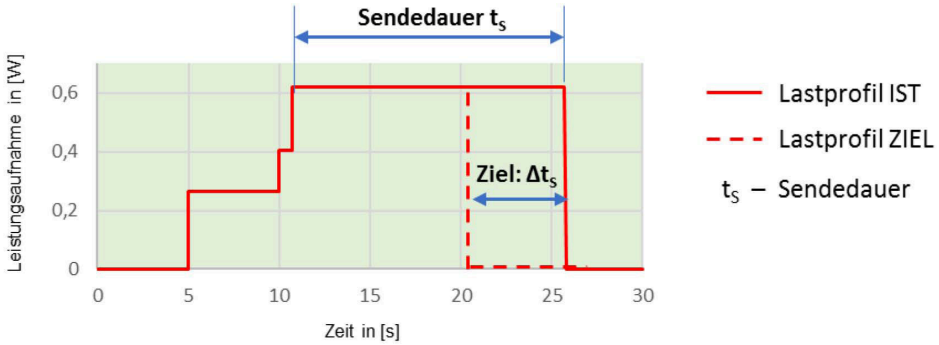


Abb. 17 Lastprofil mit verringerter Sendedauer

so etwa weitere 100 Tage betrieben werden. Um den Sensorknoten mit einer Akkuladung länger versorgen zu können, könnte der Abstand zwischen zwei Messungen z.B. auf 1,5 Stunden vergrößert werden. [34]

Abbildung 16 zeigt, dass das Transceivermodul den größten Anteil am Energieverbrauch besitzt. Wird ein Modul mit einer geringeren Leistungsaufnahme verwendet, kann der Sensorknoten länger betrieben werden.

Ein weiterer Ansatzpunkt besteht in der Verringerung der Dauer des Sendevorgangs. Gelingt es, die Sendedauer wie in Abbildung 17 dargestellt zu reduzieren, befindet sich das Transceivermodul für längere Zeit im energiesparenden Sleep-Modus und die Energieaufnahme sinkt. Eine Möglichkeit wird im Ersetzen des Kommunikationsprotokolls HTTP durch ein leichtgewichtiges Protokoll mit weniger Kommunikationsschritten gesehen.

Optimierungsansätze zu den beiden oben genannten Punkten werden in Kapitel 4 aufgezeigt, die Umsetzung eines Ansatzes durch Verwendung des Protokolls MQTT zur Datenübertragung in Kapitel 5 beschrieben. Dessen Wirksamkeit wird in Kapitel 6 untersucht. Dabei wird ein Messszenario entwickelt, welches eine vergleichende Einschätzung des bisherigen und des optimierten Konzeptes auf Basis realer Werte ermöglicht.

4 Darstellung und Bewertung von IT-Konzepten zur Verbesserung der Energieeffizienz

Schon bei der Entwicklung des bestehenden Sensorkonzepts wurden Möglichkeiten zur Verbesserung der Energieeffizienz durch Änderungen im IT-Konzept der Datenübertragung gesehen. In diesem Kapitel werden einige alternative Möglichkeiten zur Datenübermittlung aufgezeigt. Prinzipiell können die Abläufe einer Datenübertragung und die damit verbundenen softwaretechnischen Anforderungen mit dem OSI-Referenzmodell (OSI – Open System Interconnection) beschrieben werden, welches im Folgenden vorgestellt wird.

4.1 OSI-Schichtenmodell und Einordnung des bestehenden Sensorkonzepts

Der Transport von Daten durch ein Netzwerk wird von Protokollen organisiert, welche verschiedene Funktionen zu erfüllen haben. Diese Abläufe können mittels des OSI-Referenzmodells beschrieben werden, welches einen modularen Aufbau mit unterschiedlichen Schichten beinhaltet. Jede Schicht besitzt eine besondere Fähigkeit, welche der Gesamtkommunikation als Dienstleistung zur Verfügung gestellt wird und wiederum von verschiedenen Protokollen erfüllt werden kann. Einige Protokolle erstrecken sich über mehrere Schichten und erfüllen damit mehrere Aufgaben. Abbildung 18 zeigt den Aufbau des OSI-Schichtenmodells. Die Kenntnis dieses Modells erleichtert das Verständnis und die Einordnung der in diesem Kapitel vorgestellten Alternativkonzepte. [9] [20]

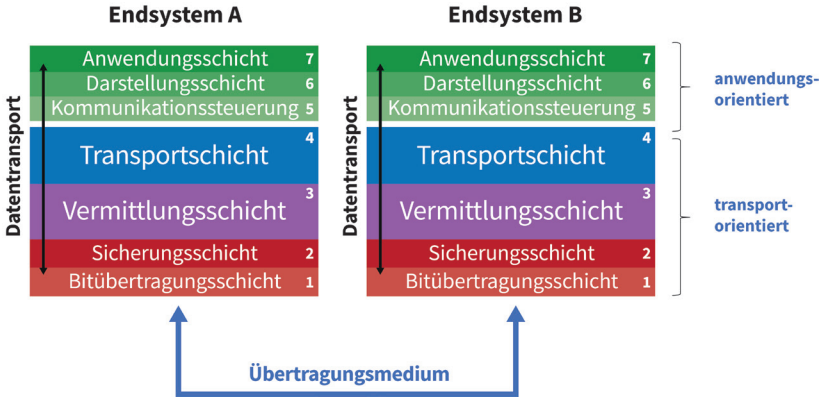


Abb. 18 OSI-Schichtenmodell nach [77] und [9]

http://doi.org/10.33968/9783966270069.0004

Die unterste Schicht ist die Bitübertragungsschicht oder auch Physical Layer genannt. Sie wandelt die Bits in ein zum Übertragungsmedium passendes Signal um und ist damit die Schnittstelle zum Übertragungsmedium, welches selbst nicht Teil des Modells ist. Die Protokolle der ersten Schicht unterscheiden sich jedoch nur nach dem eingesetzten Übertragungsmedium und -verfahren. Die nächste Schicht, die Sicherungsschicht, enthält Funktionen zur Fehlererkennung und Fehlerbehebung und sorgt damit für eine zuverlässige und funktionierende Verbindung. Die dritte Schicht stellt die Vermittlungsschicht dar, in welcher die logische Adressierung der Endgeräte erfolgt, was eine Wegfindung der Daten vom Sender zum Empfänger ermöglicht. In der Transportschicht als vierte Schicht wird der Datenfluss organisiert, indem die Datenpakete einer Anwendung zugeordnet werden. Sie ist damit die Schnittstelle der transportorientierten Schichten 1 bis 4 mit den anwendungsorientierten Schichten 5 bis 7. [20]

Die fünfte Schicht, die Kommunikationssteuerung, beinhaltet die Steuerung der Verbindung und des Datenaustausches. Da die Daten je nach Plattform verschieden dargestellt werden, werden diese in der Darstellungsschicht als sechste Schicht in ein unabhängiges Format gebracht. Zudem erfolgt hier die Verschlüsselung und Kompression von Daten. In der Anwendungsschicht als siebte Schicht findet die Dateneingabe sowie -ausgabe statt. [20] [77]

Im bestehenden Sensorsystem erfolgt die Datenübertragung über ein WLAN. Dieses besitzt in Gebäuden eine maximale Reichweite von etwa 40 bis 70 m und verwendet als Übertragungsmedium Radiowellen im Bereich von 2,4 oder 5 GHz. Die IEEE-Norm (IEEE - Institute of Electrical and Electronics Engineers) 802.11, durch welche die Normung von WLAN erfolgt, spezifiziert im OSI-Schichtenmodell die untersten beiden Schichten. Die Leistungsaufnahme des verwendeten Transceivermoduls im Sendebetrieb beträgt 400 mW. Ein WLAN-Router kann mindestens 65.535 IP-Adressen vergeben, sodass unter Vernachlässigung der realen Ressourcenbeschränkung theoretisch mehr als 65.000 Geräte in einem WLAN miteinander kommunizieren können. Damit Geräte auf der Sicherungsschicht explizit angesprochen werden und den höheren Schichten ihre Dienste bereitstellen können, besitzen sie eine eindeutige individuelle MAC-Adresse. [28] [15] [55]

Das Netzwerk des bestehenden Konzepts arbeitet mit dem IP-Protokoll (IP – Internet Protocol) als Protokoll der Vermittlungsschicht. Mit diesem Protokoll bekommen die Geräte Netzwerkadressen zugeordnet, wodurch Datenpakte adressiert und damit dem richtigen Ziel zugeordnet werden können. In der Transportschicht wird das TCP-Protokoll (TCP – Transmission Control Program) eingesetzt, welches für eine gesicherte Übertragung der Daten in der richtigen Reihenfolge sorgt. Bevor Daten übermittelt werden können, muss zunächst die Verbindung hergestellt sein. Dafür muss eine Seite die Verbindung aktiv anfragen. Die andere Seite muss sich in einem Abhörstatus befinden, welche eine Verbindung erlaubt. Anschließend ist ein Datenstrom zwischen den Endpunkten der Verbindung in beide Richtungen möglich. Das TCP-Protokoll nimmt die Daten von den Anwendungen entgegen, teilt diese in Pakete auf, welche mit einem Header versehen werden und übergibt diese an das IP-Protokoll. Durch den Header entsteht ein

Verwaltungsanteil von mindestens 20 Byte je Datenpaket. Die Datenpakete werden beim Empfänger durch das TCP-Protokoll in die richtige Reihenfolge gebracht, die Daten wieder zu einem Datenstrom zusammengesetzt und der richtigen Anwendung übergeben, welche durch eine Port-Nummer erkannt wird. Das Verwenden des TCP-Protokolls wird von dem eingesetzten anwendungsorientierten Protokoll HTTP, welches in Kapitel 3.2 vorgestellt wird und die Aufgaben der Schichten 5 bis 7 erfüllt, gefordert. [3] [76] [23]

Die folgenden zwei Ansatzpunkte ermöglichen eine kommunikationstechnische Optimierung: Zum einen ist der Einsatz eines anderen Protokolls der anwendungsorientierten Schichten mit kleineren Nachrichtenheadern und weniger Kommunikationsschritten anstatt HTTP möglich (s. Abschnitt 4.3), wodurch teilweise auch ein anderes Protokoll der Transportschicht eingesetzt wird. Je nach konkret verwendetem Protokoll kann hierbei die bestehende Hardware weiterverwendet werden. Eine Alternative stellt der in Kapitel 4.2 vorgestellte Ansatz dar, bei dem bereits in den transportorientierten Schichten angesetzt wird.

4.2 Alternativen in den transportorientierten Schichten

Bei dieser Herangehensweise werden alle Protokolle durch alternative Verfahren wie z.B. ZigBee, Z-Wave oder Bluetooth Low Energy (BLE) ersetzt. Der Nachteil dieses Ansatzes ist es, dass bei einer Umsetzung das gesamte Sensorkonzept geändert werden muss und damit nicht nur neue Hardware beschafft werden, sondern auch eine Einarbeitung in die neuen Hardware-, Software- und Programmierumgebungen erfolgen muss. Diese Protokolle werden im Folgenden vorgestellt, aber aus den eben angeführten Gründen im weiteren Verlauf der Arbeit nicht weiter umgesetzt. Bei einer zukünftigen Neukonzeption eines anderen Messkonzepts kann eine erneute Betrachtung dieser Konzepte in Erwägung gezogen werden.

ZigBee
ZigBee ist ein Funkstandard, welcher für energieeffiziente Anwendungen mit geringer Datenrate, wie der Fernsteuerung von Aktoren und Übertragung von Sensordaten, entwickelt wurde. Er ist ein häufig angewendeter Standard in der Haus-, Gebäude- und Industrieautomation, sowie der Lichttechnik. [72] [28]

Je nach Leistung und verwendeter Sendefrequenz des Funkmoduls und den Bedingungen der Umgebung besitzt ein ZigBee-Netzwerk eine Reichweite von 10 bis 100 m. Die sich im Netzwerk befindenden Endgeräte verwalten dieses autark. ZigBee baut auf dem IEEE 802.15-4-Standard auf, welches die unteren beiden OSI-Schichten spezifiziert und Parameter wie Modulation und Kanalzugriff definiert. Als Übertragungsmedium wird in Europa das 868 MHZ- und das 2,4 GHz-Band verwendet, über welches 20 bzw. 250 kBit je Sekunde übertragen werden können. Ein Transceivermoduls von Texas Instruments (TI) verbraucht im Sendebetrieb etwa 50 mW. [66] [72] [28]

Es werden durch ZigBee aufbauend auf dem IEEE-Standard drei Gerätetypen definiert: Der ZigBee Coordinator, welcher das Netzwerk mit einer

Identifikationsnummer eröffnet und verwaltet, sowie der ZigBee Router, welcher sich einem bestehenden Netzwerk anschließen und dessen Reichweite vergrößern kann. Der Coordinator und die Router bilden, wie in Abbildung 19 dargestellt, sog. Netzwerkknoten. ZigBee End Devices können als dritter Gerätetyp nur mit dem Coordinator oder einem Router, nicht aber direkt untereinander kommunizieren. ZigBee kann im Gegenzug mit minimalen Ressourcen auf diesen Geräten implementiert werden. Die Datenübertragung über die Netzwerkknoten kann über verschiedene Strukturen, sog. Netzwerktopologien, erfolgen, welche in Abbildung 19 dargestellt sind. Vermaschte Netzwerke (engl. mesh network) bieten dabei den Vorteil höherer Übertragungssicherheit, da bei Ausfall eines Netzwerkknotens durch die vorhandene Redundanz die Datenübertragung über einen anderen Weg dennoch möglich ist. Jeder Netzwerkknoten bekommt vom Coordinator eine 16-Bit-Adresse zugewiesen, sodass theoretisch für jedes Netzwerk 65535 Teilnehmer möglich sind, wobei dies durch die eingeschränkte Datenübertragungsrate nicht praktikabel scheint. [28] [43]

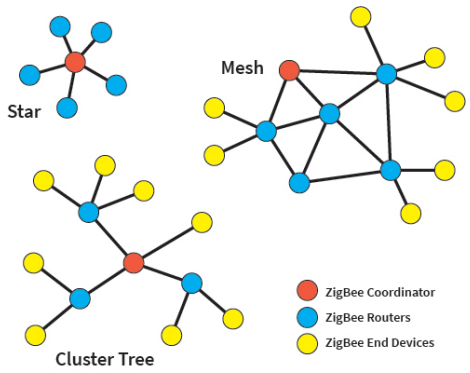


Abb. 19 mögliche Netzwerktopologien [38]

ZigBee ermöglicht das sog. Binding, mit dem mehrere funktional zueinander gehörige Geräte logisch zusammengebunden werden können. So schickt ein End Device nur ein Datenpaket an den Coordinator, welcher die Anfrage wiederum an mehrere sich im Netzwerk befindende und vorher als logisch verknüpft definierte End Devices übermittelt. [28]

In der neuesten Spezifikation ZigBee 3.0 basieren die Anwendungen auf einem standardisierten Profil, welches für definierte Nutzungsfälle wie Energieversorgung, Gebäudeautomatisierung, Lichtsteuerung, Gesundheitswesen und Telekommunikation den Datenaustausch durch Bibliotheken mit hinterlegten Attributen, Parametern und Befehlen regelt. Geräte mit einem entsprechenden Anwendungsprofil greifen auf diese Bibliotheken zurück. Das Erstellen eines eigenen Profils ist im Gegensatz zu früheren Spezifikationen nicht mehr möglich, dafür wird eine Kompatibilität zwischen allen ZigBee 3.0-spezifizierten Geräten sichergestellt. [72]

Z-Wave
Z-Wave wurde als Standard für die Gebäudeautomatisierung entwickelt und ist in diesem Bereich besonders auf dem amerikanischen Markt verbreitet. Wie auch bei ZigBee liegt der Fokus darauf, eine möglichst energieeffiziente Datenübertragung zu ermöglichen, sodass Z-Wave mit einer Datenrate von 40 kBits je Sekunde ebenso ungeeignet zur Übertragung von Audio- und Videodateien ist. Zur Datenübertragung wird das 868 MHz-Frequenzband genutzt. Das Grundprinzip ist das selbe wie bei ZigBee. Z-Wave baut ebenfalls auf dem IEEE 802.15-4-Standard auf, welcher die unteren beiden Schichten im OSI-Referenzmodell definiert und verwendet als Netzwerk-Topologie ein vermaschtes Netz. Da das Protokoll im Vergleich zu ZigBee deutlich simpler aufgebaut ist, können maximal je Netzwerk nur 232 Geräte teilnehmen und eine Übertragung ist nur über 4 Netzwerkknoten hinweg möglich. Demgegenüber steht eine höhere Reichweite von 40 bis zu 200 m. Ein Z-Wave Transceivermodul von Sigma-Designs besitzt eine Leistungsaufnahme von etwa 106 mW. [13] [12] [57] [28]

Bluetooth Low-Energy
Bluetooth Low Energy (BLE) ist ein Bluetooth basierter Standard und seit 2010 Teil der Bluetooth 4.0 Spezifikation mit dem Ziel der besonders energieeffizienten Übertragung von Daten und Sprache. Geräte können entweder Daten über BLE oder über das klassische Bluetooth übertragen oder auch für beide Varianten verwendbar sein. Die meisten Sensoren sind nur BLE-fähig, wohingegen die heutigen, meist als Gegenstelle verwendeten Smartphones und Tablets beide Varianten unterstützen. [41]

BLE nutzt ebenfalls wie ZigBee oder WLAN das 2,4 GHz-Frequenzband, welches in 79 Kanäle mit je 1 MHz Bandbreite unterteilt wird und erreicht eine Reichweite von 10 bis 40 m. Ein Transceivermoduls von Texas Instruments (TI) verbraucht im Sendebetrieb etwa 30 mW. [65] [3] [28]

Die Geräte können verbindungslos Daten übertragen, wobei ein Gerät als Broadcaster Daten versendet und ein weiteres als Observer die Daten empfangen kann, ohne dass eine Verbindung zwischen beiden Geräten aufgebaut wird. Eine zweite Möglichkeit besteht in einer verbindungsorientierten Übertragung. Bei BLE können Geräte die Funktion eines Masters oder eines Slaves einnehmen. Der Master übernimmt die Koordination des Netzes und stellt seine individuelle Stationsadresse, welche vergleichbar mit der MAC-Adresse von Geräten z.B. in WLAN-Netzwerken ist, zur Identifikation des Netzwerkes zur Verfügung. Zum Verbinden senden die Slaves auf bestimmten Kanälen Verbindungsanfragen. Diese Kanäle werden vom Master regelmäßig gescannt und ein Verbindungsaufbau kann erfolgen. Über diese Verbindung können dann anschließend Daten übertragen werden. [28] [3] [41]

Der Datenkanal eines Masters kann von bis zu 7 aktiven Slaves gleichzeitig verwendet werden. Aktive Slaves bekommen eine Adresse zur Kommunikation zugeordnet. Zusätzlich können 256 Slaves im Ruhezustand verwaltet werden, welche aber nicht aktiv kommunizieren können. Sobald die Slaves keine Daten mehr übertragen, wird die Verbindung inaktiv und die Geräte können einen stromsparenden

Sleep-Modus einnehmen. Die Datenübertragung wird mit einer Datenrate von etwa 1 MBit je Sekunde ermöglicht, diese teilen sich aber alle aktiven Slaves. [42] [28] [3]

4.3 Alternativen in den anwendungsorientierten Schichten

Das im bestehenden Sensorkonzept eingesetzte HTTP als Protokoll der anwendungsorientierten Schichten besitzt einige Eigenschaften, weswegen es nicht das ideale Protokoll zur Übertragung von Sensordaten darstellt. Jede mit HTTP versendete Nachricht beinhaltet einen großen Header und nach einiger Zeit oder Anzahl übertragener Nachrichten wird die Verbindung aufgrund eines Timeouts geschlossen. Bei Einsatz eines anderen Sensors als Applikationsklasse, bei dem es möglich und sinnvoll ist, laufend Messwerte auszulesen, wäre zudem die mangelnde Echtzeit- und Streamingfähigkeit ein weiterer Nachteil.

Wenn aus den am Beginn dieses Kapitels aufgeführten Gründen die bestehende Hardware verwendet werden soll, bieten sich als alternative, anwendungsorientierte Protokolle anstatt dem bisher verwendeten HTTP die Protokolle CoAP (Constrained Application Protocol) und MQTT (Message Queue Telemetry Transport) an. Diese werden im Folgenden vorgestellt. Für beide Protokolle sind im Gegensatz zu anderen sich sonst ebenfalls für IoT-Anwendungen anbietenden Protokollen, welche z.B. kurz in [3] vorgestellt werden, Bibliotheken für die LUA-Programmierumgebung vorhanden. [52]

CoAP (Constrained Application Protocol)
Das Ziel bei der Entwicklung von CoAP war es, ein leichtgewichtiges Protokoll zur energieeffizienten Integration von Sensoren und Aktoren in die bestehende Internet-Architektur zu erhalten. So kann CoAP über einen Proxy simpel mit HTTP interagieren und damit in klassische Internet-Anwendungen eingebunden werden. [3] [16]

CoAP baut als Protokoll der anwendungsorientierten Schichten auf UDP (User Datagram Protocol) in der Transportschicht auf. In den im OSI-Referenzmodell darunterliegenden Schichten werden weiter die selben Protokolle wie bei Einsatz von HTTP verwendet. Im Gegensatz zu TCP, welches bisher als Protokoll der Transportschicht eingesetzt wurde, werden mit UDP Daten verbindungslos, also ohne Aufbau einer Verbindung zwischen Sender und Empfänger, übertragen. Deshalb besteht für den Sender keine Möglichkeit über eine Rückmeldung in der Transportschicht festzustellen, ob die Nachrichten überhaupt und in der richtigen Reihenfolge angekommen sind. Es wird eine asynchrone Datenübertragung ermöglicht, bei der Nachrichten priorisiert und unabhängig voneinander anstatt in einer festen Reihenfolge geschickt und bearbeitet werden können, sodass große Nachrichten oder solche mit einer langen Bearbeitungszeit nicht die darauffolgenden Anfragen blockieren. Ein weiterer Unterschied ist, dass das Senden von Nachrichten an einen, aber auch an mehrere Empfänger gleichzeitig, möglich ist. Zudem werden der eigentlichen

Nachricht im Header mit einer Länge von 8 Byte nur sehr wenige Informationen hinzugefügt und es gibt keine Mechanismen zur Fehlererkennung, wodurch eine sehr schnelle Übertragung ermöglicht wird. Die Fehlererkennung und -behandlung wird durch CoAP vorgenommen. [76] [77]

Abbildung 20 zeigt den Ablauf einer Kommunikation mittels CoAP, welches analog zu HTTP eine Client/Server-Logik besitzt. Eine CoAP-Nachricht beinhaltet einen 4 Byte großen Header. Ein Verbindungsaufbau wird nicht benötigt. Ein Client sendet eine Anfrage, welche der Empfänger beantwortet. Der Token ermöglicht eine Zuordnung der Antwort zu der jeweiligen Anfrage. An die Anfrage und die Antwort können die zu übertragenden Daten angehangen werden, welche in diesem Beispiel einen Temperaturwert darstellen.

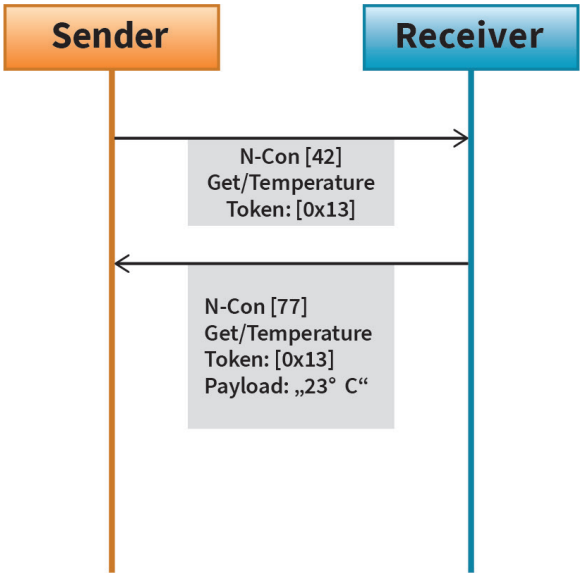


Abb. 20 Beispiel einer CoAP-Kommunikation [63]

Es sind zudem Anfragen möglich, welche eine ereignisgesteuerte Kommunikation auslösen. So kann bspw. ein Sensor nach einem solchen Request dauerhaft bei Änderungen der Messwerte oder nach bestimmten Zeitabständen eine Nachricht an den anfragenden Client senden. [63]

Es existiert eine Bibliothek, welche eine Implementierung von CoAP in das bestehende auf dem Microcontroller ausgeführte LUA-Programm vereinfacht ermöglicht. In der Dokumentation dieses CoAP-Modules in [39] wird allerdings darauf hingewiesen, dass es sich dabei um ein sehr frühes Entwicklungsstadium handelt, welches zudem nicht den vollen Funktionsumfang enthält. Wenn weiterhin LUA als Programmiersprache eingesetzt werden soll, erscheint eine Verwendung nicht sinnvoll, solange keine ausgereiften Module existieren. Ein Einsatz von CoAP

ohne entsprechende Module ist nur mit sehr stark erhöhtem Programmieraufwand möglich. Bei erfolgter Weiterentwicklung des CoAP-Moduls für LUA ist ein zukünftiger Einsatz aber zu überdenken.

MQTT (Message Queue Telemetry Transport)
MQTT wurde von IBM zur Überwachung von Ölpipelines entwickelt und ist seit 2010 frei verfügbar. Es ermöglicht Nachrichten über unzuverlässige Netzwerke und mit Einsatz ressourcenbeschränkter Geräte zu übertragen. MQTT ist ein verbindungsorientiertes Protokoll und baut auf TCP und IP als Protokolle der Transport- und Vermittlungsschicht auf. [53] [17]

MQTT besitzt eine Publish/Subscribe-Architektur, bei der sich Sender und Empfänger mit einem zentralen Server, welcher Broker genannt wird, verbinden. Der Broker speichert keine Daten, sondern verwaltet nur die Kommunikation. Die Clients verbinden sich nie direkt untereinander, sondern immer mit dem Broker. Clients können entweder Nachrichten unter einem bestimmten Topic bereitstellen (= Publisher, Sender) oder Nachrichten eines bestimmten Topics über eine Art Abonnement empfangen (= Subscriber, Empfänger). Ein Client kann auch gleichzeitig Publisher und Subscriber sein. [56]

Ein Topic stellt eine Art Betreff dar, unter welchem die Nachrichten vom Publisher veröffentlicht werden. Der Subscriber gibt an, dass er Nachrichten mit einem bestimmten Topic empfangen möchte. Das Topic kann ein einzelner Begriff sein. Auch eine Zusammensetzung von Begriffen in einer hierarchischen Struktur ist möglich. Unter diesem Kommunikationskanal veröffentlicht dann der Publisher Daten, indem er diese unter Angabe des Topics an den Broker sendet. Dieser überprüft, welche Empfänger dieses Topic abonniert haben und sendet die Nachricht an alle Subscriber weiter. Eine MQTT-Nachricht verursacht einen mindestens 2 Byte großen Header. [56]

Abbildung 21 zeigt anhand eines Beispiels das Grundprinzip von MQTT. Ein Temperatursensor misst als Wert „21°C“ und veröffentlicht diesen als Publisher unter dem Topic „temperature“. Die Subscriber Laptop und Mobile Device haben im Voraus dem Broker mitgeteilt, dass sie Nachrichten mit dem Topic „temperature“ abonnieren möchten. Der Broker empfängt nun die Nachricht „21°C“ unter dem Topic „temperature“, prüft, welche Clients diese abonniert haben, und sendet diese anschließend an die entsprechenden Clients Laptop und Mobile Device.

Der prinzipielle Ablauf einer Nachrichtenübertragung mittels MQTT ist in Abbildung 22 dargestellt. Der Client fragt zu Beginn beim Broker eine Verbindung an. Der Broker bestätigt diese entweder mit einer Nachricht oder negiert diese. Sobald die Verbindung bestätigt wurde, überträgt und/oder empfängt der Client solange Nachrichten, bis die Verbindung aktiv von einer der beiden Seiten geschlossen oder die WLAN-Übertragung unterbrochen wird. Es gibt kein Beenden der Verbindung durch einen Timeout wie bei HTTP.

Es existiert eine Bibliothek, welche eine Implementierung von MQTT in das auf dem Microcontroller ausgeführte LUA-Programm vereinfacht ermöglicht.

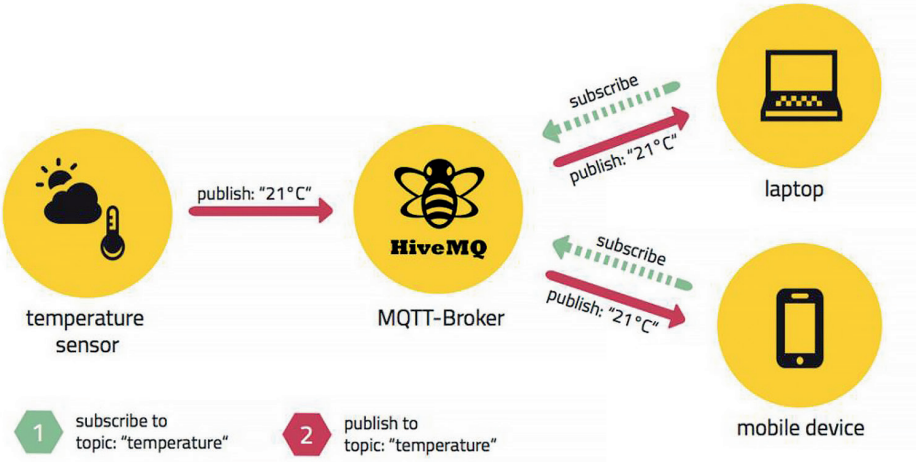


Abb. 21 Grundprinzip MQTT [31]

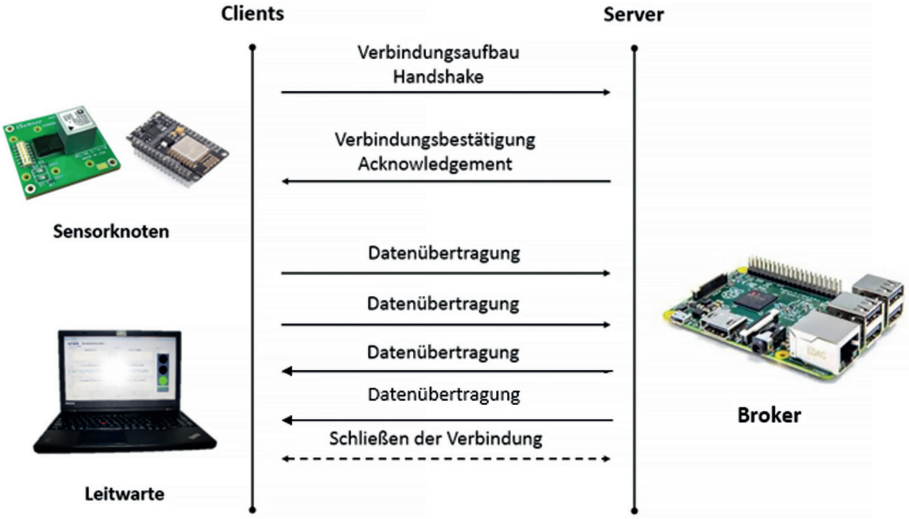


Abb. 22 prinzipieller Verbindungsablauf mit MQTT

4.4 Übersicht und Vergleich der vorgestellten Protokolle

Eine Übersicht über charakteristische Vergleichsgrößen der in den Kapiteln 4.2 und 4.3 vorgestellten Konzepte bietet die folgende Tabelle 2. Die Reichweite der Funkübertragung ist stark von den jeweils konkreten Umgebungsbedingungen

Tab. 2 Übersicht über mögliche Protokolle

Kriterien	WLAN mit HTTP	WLAN mit CoAP	WLAN mit MQTT	ZigBee	Z-Wave	BLE
Reichweite in Gebäuden in [m]	40 - 70			10 -100	40 - 200	10 - 40
max. Datenrate in [MBit/s]	240			0,25	0,04	1
Leistungsaufnahme Transceivermodul im Sendebetrieb in [mW]	400			50	106	30
max. Anzahl aktiver Clients	> 65535			65535	232	7

abhängig, sodass die dort aufgeführten Werte nur für eine grobe Einschätzung dienen. BLE besitzt mit maximal 40 m eine geringere Reichweite als die anderen aufgeführten Standards, die eine maximale Reichweite von 70 bis 200 m aufweisen.

Ein Vergleich zwischen den Protokollen ergibt, dass die Datenrate der auf WLAN aufbauenden Protokolle mit dem Faktor 250 bis 6000 sehr viel größer ist als die Datenrate der anderen Konzepte. Die geringen Datenraten vor allem von ZigBee und Z-Wave erlauben ausschließlich eine Daten- und keine Sprach- oder Bildübertragung.

Im Gegenzug beträgt die Leistungsaufnahme eines WLAN-Transceivermoduls das etwa 10-fache gegenüber einem Zig-Bee- oder BLE- und das Vierfache gegenüber einem Z-Wave-Modul. Die erhöhte Datenrate ermöglicht aber vor allem bei größeren Datenmengen eine deutlich reduzierte Übertragungsdauer, sodass das Transceivermodul dann längere Zeit einen energiesparenden Sleep-Modus einnehmen kann.

Große Unterschiede bestehen auch in der maximal möglichen Anzahl gleichzeitig aktiver Clients. Bei BLE ist diese auf sieben Geräte limitiert. ZigBee und WLAN lassen theoretisch 65535 Geräte je Netzwerk zu. In der Praxis scheint diese Anzahl aufgrund der Ressourcenbeschränkung der Router und der Datenrate unrealistisch.

Im Rahmen dieser Bachelorarbeit soll das bestehende Sensorkonzept unter Verwendung der vorhandenen Hardware kommunikationstechnisch optimiert werden. Für ZigBee, Z-Wave und BLE sind zwar Transceivermodule mit gegenüber WLAN deutlich verringerter Leistungsaufnahme vorhanden. Ein Einsatz kann aber nur mit neuer Hardware stattfinden und scheidet deshalb als Optimierungsmöglichkeit aus. Zudem sind die möglichen Datenübertragungsraten niedrig. Mit BLE lassen sich außerdem durch die begrenzte Anzahl aktiver Clients keine Netzwerke bilden.

Eine Übersicht mit einigen Vergleichskriterien über die in Kapitel 3.2 sowie in 4.3 vorgestellten anwendungsorientierten Protokolle zeigt Tabelle 3. Ein Einsatz dieser Protokolle ist mit der bestehenden Hardware möglich.

Tab. 3 Übersicht der vorgestellten anwendungsorientierten Protokolle

Kriterien		HTTP	CoAP	MQTT
Transport		verbindungs-orientiert	verbindungs-los	verbindungs-orientiert
Mindestgröße Nachricht	**Mindestgröße Header Transportschicht**	20 Byte (TCP)	8 Byte (UDP)	20 Byte (TCP)
	Mindestgröße Header Anwendungsschicht	ca. 420 Byte	4 Byte	2 Byte
Architektur		Request / Response		Publish / Subscribe

Die Unterschiede im Energieverbrauch ergeben sich zwischen den anwendungsorientierten Protokollen vor allem durch die Sendedauer. Diese wird hauptsächlich durch die Größe und Anzahl der zu übermittelnden Nachrichten bestimmt. Bei HTTP und MQTT als verbindungsorientierte Protokolle erfolgt zum Aufbau der Verbindung ein Nachrichtenaustausch. CoAP baut auf dem verbindungslosen UDP-Protokoll in der Transportschicht auf, sodass ein Verbindungsaufbau entfällt. Die Verbindung von HTTP wird nach einer gewissen Anzahl an Nachrichten oder einer bestimmten Zeitdauer unterbrochen. Es muss dann ein erneuter Verbindungsaufbau stattfinden. Bei MQTT bleibt die Verbindung dauerhaft bestehen.

Die Unterschiede in der Nachrichtengröße ergeben sich durch die vom Protokoll angehängten Informationen. HTTP verursacht im Gegensatz zu CoAP und MQTT mit einer Größe von etwa 420 Byte im bestehenden Sensorkonzept einen sehr großen Header. In dieser Angabe ist zur Vergleichbarkeit die URL des Servers nicht enthalten, da bei CoAP und MQTT die Adresse bzw. das Topic nach dem Header angefügt und nicht in die Headergröße miteinbezogen werden. Zählt man die Headergrößen von Anwendungs- und Transportprotokoll zusammen, benötigt CoAP durch Verwenden von UDP in der Transportschicht mit mindestens 12 Byte die geringste Datenmenge. Zudem entfällt bei CoAP der Nachrichtenaustausch zum Verbindungsaufbau. Die von MQTT verursachte Datenmenge liegt mit 22 Byte in einer ähnlichen Größenordnung.

Ein weiterer Unterschied besteht in der Architektur der Protokolle. HTTP und CoAP nutzen eine Request/Response-Logik. Ein Client richtet eine Anfrage an einen Server, welche anschließend beantwortet wird. MQTT arbeitet nach dem Publish/Subscribe-Prinzip. Ein Client veröffentlicht Daten unter einem bestimmten Topic. Ein anderer Client kann dieses Topic abonnieren. Ein sog. Broker verwaltet den Informationsaustausch.

Es sind keine Untersuchungen vorhanden, in denen HTTP, MQTT und CoAP unter gleichen Testbedingungen miteinander verglichen wurden. So erfolgt in [16] ein Vergleich von MQTT mit QoS-Level 0 und CoAP bezüglich Übertragungsdauer und zu übertragener Datenmenge. Es werden unter Laborbedingungen

Informationen mit einer Größe von 2500 Bytes übertragen. Die Sendedauer von MQTT ist gegenüber CoAP 20 % geringer, die Datenmenge um 10 % höher. In diesem Szenario wird von einer einmaligen Informationsübertragung ausgegangen, sodass der Verbindungsaufbau bei MQTT in der Sendedauer und Datenmenge enthalten ist. Wird von einem einmaligen Verbindungsaufbau ausgegangen, nachdem eine Übertragung vieler Nachrichten stattfindet, sinkt die Sendedauer von MQTT auf etwa ein Drittel der Dauer von CoAP. Die zu übertragene Datenmenge ist dann gleich groß. Der Vorteil von CoAP liegt vor allem bei instabilen Verbindungen oder solchen, bei denen der Client nach einer erfolgreich gesendeten Nachricht wieder in den Sleep-Modus versetzt wird.

In dem in [50] vorgestellten Testszenario wird ein Vergleich von HTTP und MQTT mit QoS-Level 1 bezüglich Energieverbrauch und Übertragungsdauer vorgenommen. Als Client dient ein Android-Smartphone und als Server ein Desktop-PC. Die Datenübertragung erfolgt über ein WLAN-Netzwerk. Es wurde die Übertragung von 1024 Nachrichten mit einer Größe von je 1 Byte untersucht, welche so schnell wie möglich übermittelt werden sollen. Zum Verbindungsaufbau benötigt HTTP 20% weniger Energie, da bei MQTT das Abonnieren des Testtopics enthalten ist. Wird kein Topic abonniert, liegt der Verbrauch in der gleichen Größenordnung. Zum Nachrichten-Erhalten benötigt MQTT gegenüber HTTP in diesem Szenario etwa ein Zwanzigstel der Zeit und ein Fünfzigstel der Energie sowie zum Nachrichten-Versenden etwa ein Viertel der Zeit und ein Sechstel der Energie. Es tritt somit nicht nur eine Verringerung der Sendedauer, sondern auch eine Reduzierung der Stromaufnahme ein.

In [47] wird eine Vielzahl an bestehenden Untersuchungen qualitativ zusammengefasst. Es wird bestätigt, dass im Allgemeinen Nachrichtengröße, Energieverbrauch und Datenmenge einer HTTP-Übertragung deutlich über denen von MQTT liegen, welche wiederum leicht höhere Werte als bei Einsatz von CoAP aufweist.

4.5 Auswahl des MQTT-Protokolls

Die Vorteile des leichtgewichtigen MQTT-Protokolls bestehen gegenüber HTTP bei der Übertragung von Sensordaten in der Publish/Subscribe-Logik sowie des geringen Protokolloverheads. Es wird eine Verringerung der Sendedauer erwartet, welche zur Senkung des Energieverbrauchs des Sensorknotens führt. Zudem bietet MQTT den großen Vorteil, dass die Verbindung nicht nach einer gewissen Anzahl übertragener Nachrichten oder einer bestimmten Zeitdauer unterbrochen wird, wodurch eine streaming- und echtzeitfähige Kommunikation ermöglicht wird. Es wird eine ereignisgesteuerte Kommunikation ermöglicht, bei der z.B. Sensoren ohne vorherige Anfrage eines Servers bei Eintreten von bestimmten Ereignissen sofort ohne erneuten Verbindungsaufbau Daten übertragen können. Ein weiterer Vorteil von MQTT besteht in der einfachen Skalierbarkeit mit bis zu mehreren hunderttausend Clients je Broker. Für den Publisher ist es unerheblich, an wie viele Subscriber die Nachricht letztendlich versendet wird. Eine Limitierung erfolgt durch die Ressourcen des Brokers und die Kapazität des Netzwerkes.

In dieser Bachelorarbeit wird deshalb ein alternatives Sensorkonzept erstellt, in dem HTTP durch MQTT als Protokoll der anwendungsorientierten Schichten ersetzt wird.

Konfigurationsmöglichkeiten von MQTT
MQTT baut auf TCP als Protokoll der Transportschicht auf, sodass bei einer stabilen Verbindung eine komplette Übertragung der Nachrichten garantiert ist. Da MQTT für den Einsatz von instabilen Verbindungen entwickelt wurde, ist das Veröffentlichen von Nachrichten in drei verschiedenen Servicequalitäten (QoS – Quality of Service) möglich. [56]

Den Ablauf einer Nachrichtenübertragung der verschiedenen QoS zeigt Abbildung 23. Bei QoS 0 wird die Nachricht genau einmal gesendet. Der Sender erwartet keine Bestätigung und bekommt damit keine Rückmeldung, ob die Nachricht angekommen ist. Diese Vorgehensweise wird auch „fire and forget“ genannt. Mit QoS 1 wird garantiert, dass die Nachricht mindestens einmal ankommt. Ein

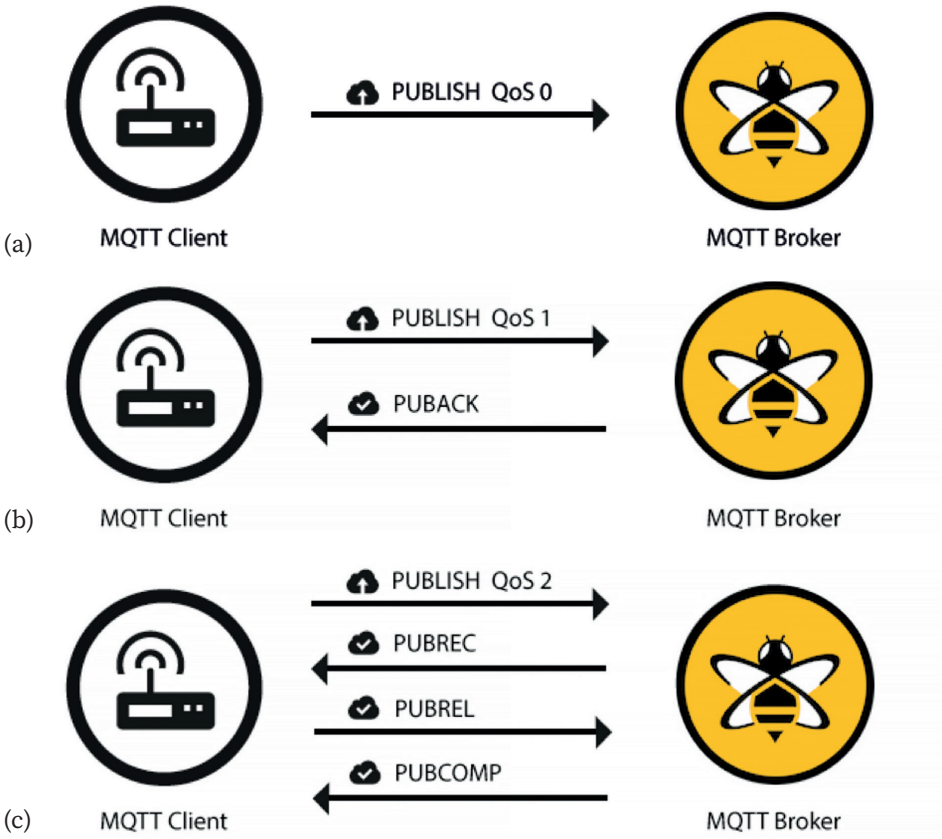


Abb. 23 Ablauf einer Nachrichtenübertragung mit QoS-Level (a) 0, (b) 1 und (c) 2 [33]

mehrmaliges Versenden wird nicht ausgeschlossen. Der Sender wartet auf eine Empfangsbestätigung vom Empfänger und sendet die Nachricht erneut, wenn diese nach einer gewissen Zeit ausbleibt. QoS 2 stellt sicher, dass eine Nachricht genau einmal versendet wird. Dafür wird eine zweistufige Empfangsbestätigung eingesetzt, bei der der Publisher die erste Empfangsbestätigung des Subscribers bestätigt und wiederum eine Bestätigung dieser Bestätigung erwartet. Das QoS-Level bestimmt den Ablauf der Nachrichtenübertragung vom Publisher zum Broker und vom Broker zum Subscriber. Der Publisher erhält unabhängig vom verwendeten QoS-Level keine Rückmeldung, ob und an wie viele Clients der Broker die Nachricht gesendet hat. [32] [56]

Ab QoS 1 wird einer Publish-Nachricht im Header eine Paket-ID hinzugefügt. Die auf eine Publish-Nachricht folgenden Antworten enthalten dann ebenfalls diese ID und können damit der Nachricht zugeordnet werden. Je höher das QoS-Level ist, desto langsamer ist durch die höhere Zahl der Kommunikationsschritte die Datenübertragung. [33]

Das Abonnieren eines Topics wird vom Broker immer mit einer Bestätigung beantwortet. Dabei muss das QoS-Level zwischen Broker und Subscriber nicht mit dem zwischen Broker und Publisher übereinstimmen. Das QoS-Level wird nur zwischen einem Client und dem Broker vereinbart. Topics lassen sich durch das Senden einer „Unsubscribe"-Nachricht wieder abbestellen. [33] [32]

Für Einsatzfälle mit instabilen Netzwerken, bei denen Clients häufig die Verbindung zum Broker verlieren, wurden in MQTT die Funktionalitäten „Last Will and Testament" und „Retained Message" implementiert. Ein Client kann bei dem Verbindungsaufbau mit dem Broker einen sog. letzten Willen bestehend aus einem Topic und einer Nachricht angeben. Sobald der Broker eine Verbindungsunterbrechung feststellt, wird dieser letzte Wille im Namen des Clients veröffentlicht. Eine mögliche Verwendung dieser Funktion ist das Senden der Nachricht „offline" unter dem Topic „Status Sensor x". [56]

„Retained Messages" sind Nachrichten, welche vom Broker unter dem angegebenen Topic gespeichert werden. Abonniert ein Client das entsprechende Topic, bekommt dieser zuerst die gespeicherte Nachricht zugesendet. Sobald ein Client eine „Retained Message" unter demselben Topic versendet, wird die bisher unter diesem Topic gespeicherte Nachricht überschrieben. [56]

Aufbau einer MQTT-Nachricht

Abbildung 24 zeigt den Aufbau einer MQTT-Nachricht, welche Daten versendet. Diese besteht aus einem fixen Header mit einer Länge von 2 Byte, welcher den Nachrichtentyp enthält, also ob es sich z.B. um eine Publish-, Subscribe-, oder Connect-Nachricht handelt. Dem schließt sich die nur für die QoS-Level 1 und 2 relevante Information an, ob die Nachricht ein Duplikat ist. Anschließend wird mitgeteilt, mit welchem QoS-Level die Nachricht versendet werden soll und ob es sich um eine „Retained Message" handelt. Das zweite Byte enthält die Länge des sich anschließenden variablen Headers und der eigentlichen Nachricht. [3]

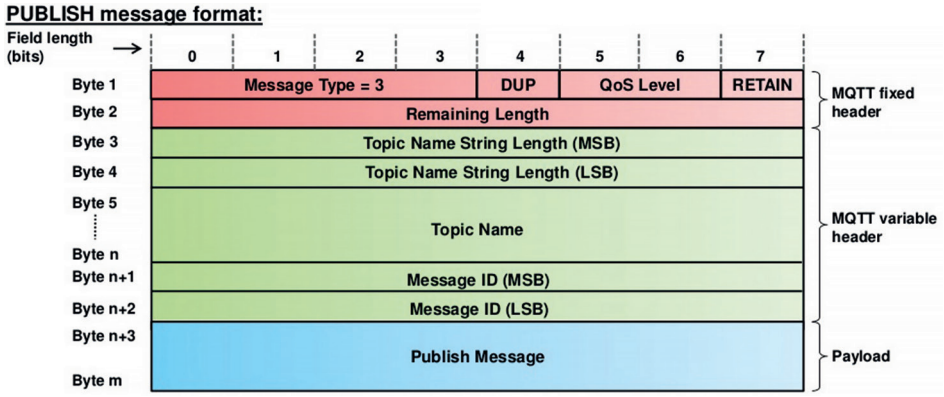


Abb. 24 Aufbau einer Publish-Nachricht in MQTT [17]

Darauf folgt der variable Header, welche die Länge des Topics, das Topic selbst und bei QoS-Level 1 und 2 die Message ID enthält. Danach wird die eigentlich zu übertragende Nachricht angehangen.

Die Länge des variablen Headers wird stark von der Länge des Topics bestimmt. Die geringstmögliche Länge des Headers insgesamt beträgt bei QoS 0 und einem Topic, welches aus nur einem Zeichen besteht, 5 Byte.

5 IT-Konzept zur kommunikationstechnischen Optimierung des Basiskonzepts

Es soll das bestehende Sensorkonzept unter Verwendung der vorhandenen Hardware kommunikationstechnisch optimiert werden. Das bisher eingesetzte Anwendungsprotokoll HTTP wird durch das in Kapitel 4.3 vorgestellte MQTT ersetzt. Es erfolgt ein Einsatz in einer als stabil anzusehenden Netzwerkumgebung, sodass unter dem Ziel der geringen Sendedauer eine Umsetzung mit QoS-Level 0 erfolgt. Die dafür notwendigen und im praktischen Teil dieser Bachelorarbeit durchgeführten softwaretechnischen Anpassungen werden in diesem Kapitel vorgestellt.

5.1 Softwaretechnische Implementierung der Datenübertragung mittels MQTT-Protokoll

Zur Nachrichtenübermittlung über MQTT wird ein Broker benötigt, welcher die Kommunikation verwaltet. Deshalb wurde zunächst auf dem Raspberry Pi unter Verwendung der Open-Source-Software Mosquitto ein MQTT-Broker eingerichtet. Die Aufgaben des RPIs ändern sich damit. Die Funktion als WLAN-Router bleibt weiterhin bestehen, die des Webservers wird durch die des Brokers ersetzt.

Als nächster Schritt erfolgte eine Einarbeitung in die verwendete Programmiersprache LUA und die Funktionsweise der Kommunikation mit dem Beschleunigungssensor über ein SPI-Interface. LUA arbeitet mittels ereignisgesteuerten Aktionen. Treten bei der Programmierung im Voraus festgelegte und von außen kommende Ereignisse ein, wird eine bestimmte Aktion ausgelöst.

Anschließend konnte das bisherige auf dem Microcontroller ablaufende Messprogramm analysiert werden, welches die Messwerte ausliest und mittels HTTP versendet. Darauffolgend wurde die Implementierung von MQTT in das Programm durchgeführt, wodurch sich einige Unterschiede ergeben. Es wird zum Erstellen des Skriptteils mit der Datenübertragung eine andere Bibliothek angesprochen, weshalb andere Befehle verwendet werden. Zudem besitzt MQTT mit der Publish/Subscribe-Architektur ein anderes Konzept zur Datenübertragung.

Bei der Programmierung muss die begrenzte Größe des Speichers des Microcontrollers beachtet werden. Wird während der Skriptausführung mehr Arbeitsspeicher als die zur Verfügung stehenden 50 kByte benötigt, wird der Programmablauf unterbrochen und das Transceivermodul führt einen Neustart durch. Die Länge der erstellten Programme ist ebenfalls begrenzt. Der Programmspeicher besitzt eine Größe von 4 MByte und beinhaltet zudem die Firmware sowie die eingebundenen

http://doi.org/10.33968/9783966270069.0005

LUA-Bibliotheken, welche bspw. zum Aufbau der WLAN-Verbindung, zur Kommunikation mittels SPI-Interface oder zur Datenübertragung durch MQTT benötigt werden.

Programmablauf

In Abbildung 25 ist der prinzipielle Ablauf des Messprogramms mit Einsatz von MQTT abgebildet. Es ist in die zwei Skripte Initialisierungsskript und Messskript unterteilt. Der auskommentierte LUA-Code des Initialisierungsskripts befindet sich in Anhang 1, der des Messkriptes in Anhang 2.

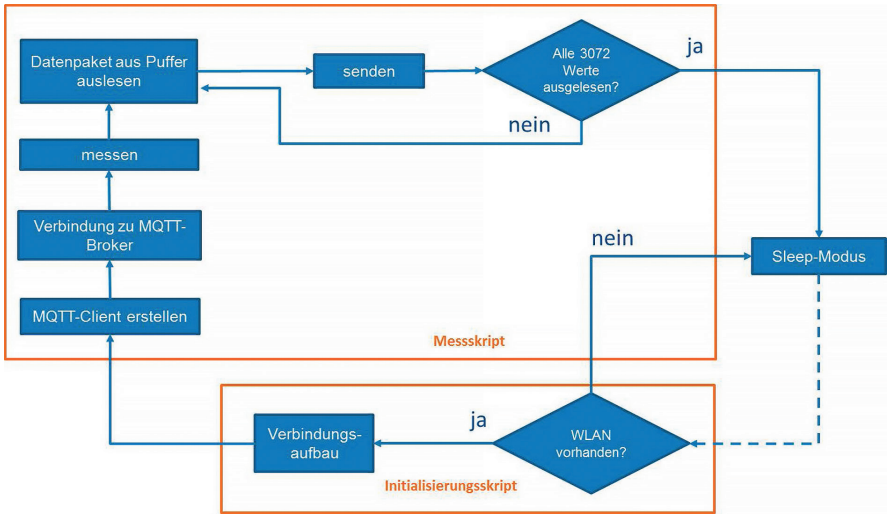


Abb. 25 Ablauf des MQTT-Programms

Das Initialisierungsskript startet, sobald der Microcontroller aus dem Sleep-Modus aufgewacht ist. Es erfolgt zunächst eine Abfrage, ob ein WLAN vorhanden ist. Falls kein WLAN existiert, wird der Microcontroller wieder in den Sleep-Mode versetzt. Wird ein bestehendes Netzwerk gefunden, wird mit diesem eine Verbindung aufgebaut und anschließend das Messskript aufgerufen.

Das Messskript läuft prinzipiell wie in Abbildung 25 dargestellt ab. Zunächst wird ein MQTT-Client eingerichtet, welcher im späteren Verlauf des Programms MQTT-Nachrichten unter dem Topic „ADIS“ veröffentlicht. Anschließend wird eine Verbindung zum Broker hergestellt, welche dauerhaft zur Datenübertragung zur Verfügung steht. Ist der Verbindungsaufbau erfolgreich verlaufen, erfolgt die Kommunikation mit dem Beschleunigungssensor über das SPI-Interface. Es werden Einstellwerte übermittelt und eine Messung des Sensors ausgelöst.

Der Sensor misst und speichert 3072 Werte zwischenzeitlich in einem Puffer. Dieser Zwischenspeicher wird durch das Messskript schrittweise über die SPI-Verbindung ausgelesen. Das Versenden der Messwerte sollte paketweise erfolgen. Zwar ist das Übertragen von Einzelwerten möglich, aber auch mit MQTT wird dafür eine größere Zeitdauer als bei der Übermittlung von Datenpaketen benötigt. Die Ursache dafür besteht darin, dass in dieser Variante 3072 Nachrichten erstellt und

versendet werden müssen, was durch die begrenzten Ressourcen des Microcontrollers eine gewisse Zeit in Anspruch nimmt und allein durch die von TCP und MQTT verursachten Header eine größere Datenmenge verursacht. Der Header einer MQTT-Nachricht besitzt inklusive des TCP-Headers eine Mindestgröße von 22 Byte. Zur Übertragung des Topics „ADIS“ werden 4 Byte benötigt, sodass zur Übermittlung von 3072 Nachrichten insgesamt (3072 * 26 *Byte* =) 79.872 zu übertragende und vor allem durch den Microcontroller zu verarbeitende Byte entstehen.

Um Datenpakete zu bilden, werden die schrittweise ausgelesenen Einzelwerte in einer Tabelle auf dem Microcontroller zwischengespeichert. Wurden 128 Werte ausgelesen, werden diese mit Kommas getrennt zu einer Zeichenkette (engl. String) zusammengefügt. Nachdem diesem String als Metadaten die Achsenposition und die verwendete Filtereinstellung hinzugefügt wurden, wird dieser unter dem Topic „ADIS“ als MQTT-Nachricht veröffentlicht. Das Erstellen eines Nachrichtenheaders mit dem Messskript wie bei HTTP entfällt.

Die maximale durch MQTT übermittelbare Nachrichtengröße beträgt 256 MByte. [47] Ein String besitzt in LUA hingegen eine maximal zulässige Länge von 2048 Byte. Soll die Anzahl an Messwerten je Datenpaket identisch sein, ergibt sich durch diese Begrenzung eine maximale Paketgröße von 128 Werten. Bei der nächstgrößeren möglichen Anzahl von 192 Werten je Paket würde durch die Messwerte, die Kommas sowie die angefügten Metadaten die Stringlänge von 2048 Byte überschritten werden.

Es wurde die Laufzeit des Messskriptes mit verschiedenen Paketgrößen ermittelt. Werden 128 Werte je Datenpaket ausgelesen und versendet, ist die Gesamtlaufzeit am geringsten. Dieses Ergebnis wurde mit dem in Kapitel 6 vorgestellten Messszenario und einer Paketgröße von 64 Werten überprüft. Es wurde festgestellt, dass bei 64 Messwerten je Paket die gesamte Sendezeit 2 s länger andauert als bei Verwenden von 128 Werten je Paket, weshalb eine Paketgröße von 128 Werte eingesetzt wird.

Das erfolgreiche Versenden eines Datenpakets als MQTT-Nachricht dient als Ereignis, um das Auslesen und Versenden des nächsten Pakets auszulösen. Es wird so lange abwechselnd der Zwischenspeicher schrittweise ausgelesen und die Daten in Paketen mit je 128 Werten versendet, bis alle 3072 Messwerte erfasst und insgesamt 24 Datenpakete übermittelt wurden. Wird die Verbindung zum Broker innerhalb des Ablaufs des Messskriptes unterbrochen, wird zum erneuten Verbindungsaufbau das Initialisierungsskript aufgerufen. Sind alle Messwerte ausgelesen, nehmen der Sensor und das Transceivermodul wieder den Sleep-Modus ein.

5.2 Softwaretechnische Anpassungen der Leitwarte

Durch das Verwenden von MQTT als Protokoll zur Netzwerkkommunikation ist eine Datenanalyse und Kennwertbildung mittels Matlab-Applikation wie im Ausgangsstand nicht möglich, da diese nicht auf der Basis von MQTT kommunizieren kann. Zudem verwaltet der Raspberry Pi nun als Broker die Kommunikation und loggt die Daten nicht mehr mittels PHP-Skript in einer CSV-Datei. Diese Aufgabe wird nun von der Leitwarte übernommen.

Bei der Umsetzung des Alternativkonzepts wurde deshalb auf der Clientseite der Leitwarte die Open-Source-Software Node-Red verwendet. Diese wurde von IBM

entwickelt, 2013 veröffentlicht und ist im Gegensatz zu Matlab kostenfrei. Node-Red ist plattformunabhängig und läuft deshalb auch auf Smartphones oder Einplatinenrechnern wie z.B. dem Raspberry Pi.

Die Software besitzt eine webbasierte Benutzerschnittstelle, die mit einem Browser aufrufbar ist. Auf dieser Weboberfläche können auf Grundlage einer datenflussbasierten Programmierung vorgefertigte grafische Funktionsbausteine aus einer Liste ausgewählt, auf der Oberfläche platziert, anschließend konfiguriert und mittels „Drag & Drop"-Aktion miteinander verbunden werden. Es existieren Module mit Eingängen, welche als Senke Daten empfangen, Elemente mit Ausgängen zum Versenden von Daten als Quellen sowie Bausteine, welche beide Funktionen vereinen. Abbildung 26 zeigt eine Auswahl möglicher Funktionselemente in Node-Red. Es sind z.B. Bausteine für eine Kommunikation über HTTP oder MQTT vorhanden. Andere Elemente führen eine FFT durch, bilden einen softwaretechnischen PID-Regler oder versenden Informationen automatisiert als E-Mail. Zudem können in JavaScript geschriebene Funktionen eingefügt werden, womit bei Beherrschung dieser Programmiersprache große Funktionsmöglichkeiten entstehen. [69]

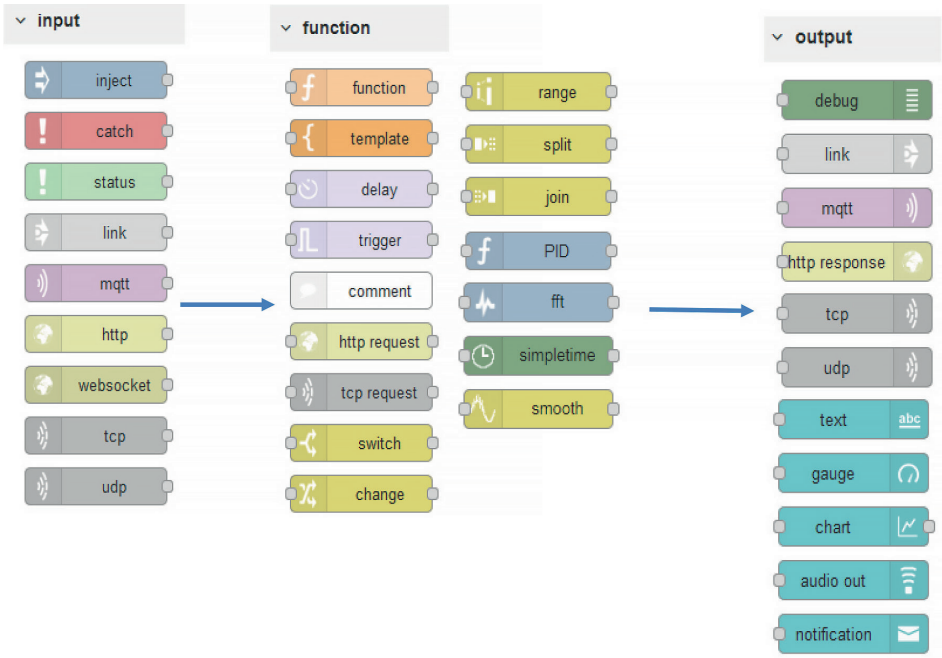


Abb. 26 Auswahl möglicher Funktionselemente in Node-Red

Die Module werden anschließend in einem Fenster in der Weboberfläche konfiguriert. Abbildung 27 zeigt die Konfiguration eines MQTT-Clients, welcher das Topic „ADIS" abonniert. Die Angabe der IP-Adresse des Brokers, des Topic-Namens und des gewünschten QoS-Level ist dafür ausreichend.

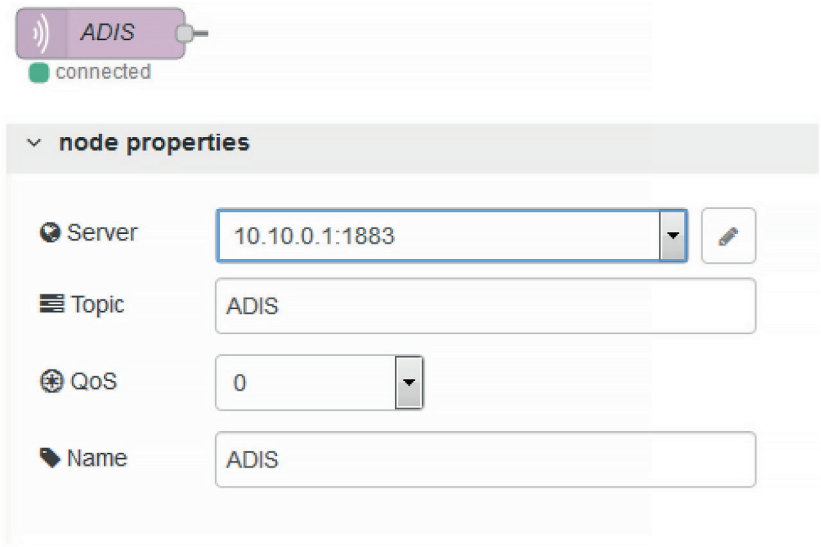


Abb. 27 Konfiguration eines MQTT-Subscriber-Clients

Mit Node-Red kann eine grafische Darstellung und die Eingabe von Werten auf einem sog. Dashboard erfolgen. Dieses fungiert als Webserver und kann über eine URL von jedem beliebigen Gerät im Netzwerk aufgerufen werden.

In der aktuellen Umsetzung des Alternativkonzepts wird das Node-Red auf dem Rechner der Leitwarte ausgeführt und übernimmt das Loggen der Daten in eine CSV-Datei, was im bisherigen Konzept auf dem Raspberry Pi erfolgt ist. Dazu wurde der in Anhang 3 ersichtliche Programmfluss erzeugt. Dieser erstellt zunächst einen MQTT-Client, welcher das entsprechende Topic „ADIS“ abonniert, unter dem das Transceivermodul die Messdaten veröffentlicht. Eine Anforderung an das Mitloggen der Daten besteht im Erstellen eines neuen Ordners je Tag und einer neuen Messdatei je Stunde, damit die Größe einer Messdatei überschaubar bleibt. Die darauffolgenden Funktionselemente überprüfen das Vorhandensein dieser Ordner bzw. Dateien und erstellen diese falls nötig neu. Die Namen enthalten das aktuelle Datum bzw. die Uhrzeit. Eine neue CSV-Datei wird inklusive Spaltenüberschriften angelegt. Zudem wird den Messdaten beim Loggen ein Zeitstempel hinzugefügt.

Die mit dem Node-Red erstellte CSV-Datei kann mit der Matlab-Applikation eingelesen werden und damit die bisherige Darstellung der Leitwarte erfolgen. Alternativ lassen sich auch mit Node-Red kompliziertere Aufgaben wie z.B. eine FFT umsetzen, sodass eine Übertragung der Funktionen der Matlab-Applikation wie die Errechnung und Darstellung des Diagnosekennwerts und die Darstellung des Zeit- sowie Frequenzsignals denkbar ist. Da Node-Red als Webserver fungieren kann, könnten diese Darstellungen zudem von allen Geräten wie z.B. Smartphones und Laptops mit einem Webbrowser angezeigt werden. Die Voraussetzung dafür ist, dass sich diese Geräte im gleichen WLAN wie die Leitwarte befinden.

5.3 Beschreibung des Gesamtkonzepts im Sollzustand

Es ergibt sich mit den in Kapitel 5.1 und 5.2 beschriebenen Veränderungen das in Abbildung 28 dargestellte Gesamtkonzept. Die Hardware besteht unverändert wie auch im bisherigen Konzept aus dem Raspberry Pi, dem Sensorknoten mit u.a. dem Beschleunigungssensor und dem Transceivermodul sowie dem Leitwarten-Rechner. Der Sensorknoten befindet sich räumlich getrennt von der Leitwarte. Es sollen die vom Beschleunigungssensor gemessenen Werte kabellos an die Leitwarte zu übertragen werden.

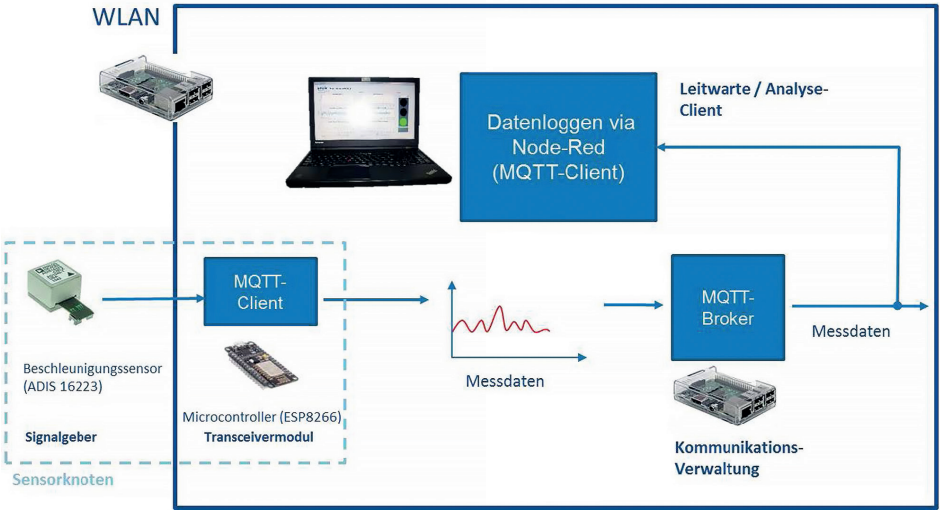


Abb. 28 Gesamtkonzept Nachrichtenübertragung mittels MQTT

Der Raspberry Pi besitzt als Kommunikationszentrum zwei Aufgaben. Zum einen spannt er als Router ein WLAN auf. Zusätzlich verwaltet der RPI als Broker die über dieses Netzwerk ablaufende Kommunikation. Als MQTT-Clients befinden sich in diesem WLAN das Transceivermodul und die Leitwarte. Der Microcontroller veröffentlicht als Publisher-Client die aus dem Sensor ausgelesenen Messdaten unter dem Topic „ADIS“. Die Leitwarte teilt als Subscriber-Client dem Broker einmalig bei Verbindungsaufbau mit, dass alle Nachrichten abonniert werden sollen, welche unter diesem Topic veröffentlicht werden. Werden Daten vom Transceivermodul mit dem Topic „ADIS“ versendet, nimmt der Broker diese entgegen und leitet diese an alle Clients weiter, welche dieses Topic abonniert haben. Die Leitwarte erhält die versendeten Messdaten und loggt diese mit einem Zeitstempel in einer CSV-Datei.

Dieses Sensorkonzept bietet im Vergleich zu dem bestehenden Konzept zusätzliche Möglichkeiten im Einbinden weiterer Geräte, welche die Messdaten zu weiteren Anzeige- oder Analysezwecke erhalten könnten. Es sind weitere Geräte als MQTT-Clients denkbar, die sich in dem WLAN befinden und durch das Abonnieren

des Topics „ADIS“ ebenfalls die Nachricht vom Transceivermodul erhalten. Alternativ kann auf der Leitwarte mit Node-Red ein sog. Dashboard eingerichtet werden, welches bspw. die Messwerte oder die Analysekennwerte anzeigt. Dieses Dashboard fungiert als Webserver, kann von jedem Gerät mit einem Browser wie z.B. einem Smartphone oder einem weiteren Laptop aufgerufen und damit die entsprechenden Darstellungen angezeigt werden.

6 Messtechnische Untersuchung

6.1 Messtechnische Strategie zur Ermittlung der Sendeleistung und der Sendedauer

Durch den Einsatz des leichtgewichtigen MQTT-Protokolls zur Datenübertragung wird eine Verbesserung der Energieeffizienz vor allem durch die Verringerung der zur Datenübertragung erforderlichen Sendedauer vermutet. Zudem ist die messtechnische Aufnahme des realen Lastprofils mit den verschiedenen Betriebszuständen des Sensorknotens von Interesse, für die die Messwerte inklusive eines Zeitsignals ermittelt werden müssen. Diese Informationen werden anhand des im Folgenden beschriebenen Messkonzepts gewonnen.

Um die Leistungsaufnahme des Sensorknotens abschätzen zu können, werden die beiden Größen Spannung (U) und Strom (I) benötigt, da der folgende Zusammenhang gilt:

$$P = U * I \tag{6.1}$$

Die durch die Spannungsquelle bereitgestellte Spannung ist in diesem Messszenario weitgehend konstant, weshalb im weiteren Verlauf die Stromstärke als relevante Messgröße betrachtet wird, welche sich im Zeitverlauf ändert. Wird die Stromstärke zusammen mit einem Zeitsignal aufgenommen, kann ein Lastprofil erstellt werden. Mit diesen Daten lässt sich die Sendedauer sowie die Stromaufnahme des Sensorknotens bestimmen und eine Aussage treffen, inwieweit diese durch den Einsatz von MQTT gegenüber HTTP verringert und damit entsprechend die energetische Lebensdauer des Sensorknotens vergrößert wurden. In der Sendedauer bildet sich zudem die Sendeleistung des Transceivermoduls ab. Die Anzahl zu übertragender Messdaten bleibt konstant. Sinkt nun die Sendedauer, so ist die Sendeleistung gestiegen.

6.1.1 Messschaltung

Im Gegensatz zur Spannung kann der Strom nur über indirekte Methoden ermittelt werden. Eine Möglichkeit bietet das vereinfacht in Abbildung 29 dargestellte Verwenden eines mit dem Verbraucher in Reihe geschalteten Messwiderstands (engl. Shunt) R_{Mess} mit bekannter Größe, dessen Spannungsabfall über ein parallel geschaltetes Voltmeter gemessen wird. Transceiver und Beschleunigungssensor werden zur Darstellung in einem Verbraucher R_{TS} zusammengefasst. In einer Reihenschaltung werden alle Bauteile von einem Strom I gleicher Größe durchflossen.

http://doi.org/10.33968/9783966270069.0006

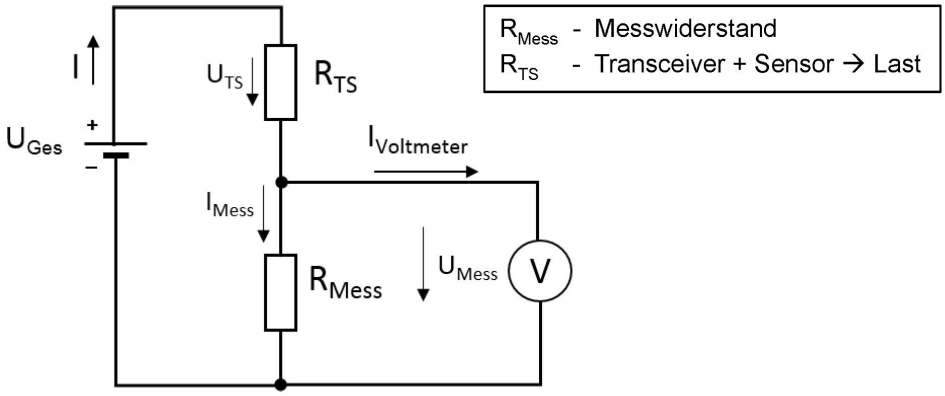


Abb. 29 vereinfachte Low-Side-Messschaltung

Durch Messung der über R_{Mess} abfallenden Spannung U_{Mess} kann wie folgt der Strom berechnet werden, welcher auch den Sensorknoten durchfließt:

$$I = \frac{U_{Mess}}{R_{Mess}} \tag{6.2}$$

Der Stromfluss der Messchaltung teilt sich auf den Messwiderstand und das Voltmeter auf. Es wird ein spannungsrichtiges Messen der über den Messwiderstand abfallenden Spannung U_{Mess} ermöglicht. Durch die Parallelschaltung entspricht der Spannungsabfall über dem Messwiderstand der mittels des Voltmeters gemessenen Spannung.

Es wurde wie in Abbildung 29 dargestellt eine sog. Low-Side-Messung durchgeführt, bei der der Messwiderstand zwischen dem Sensorknoten und der Masse der Spannungsquelle geschaltet wird. Die Nachteile dieses Messansatzes bestehen darin, dass der Massepegel aus Sicht der Last angehoben wird und Kurzschlüsse unerkannt bleiben. [73]

Diese Nachteile lassen sich mit einer wie in Abbildung 30 gezeigten High-Side-Schaltung umgehen, mit der eine Messung an der Versorgungsspannungsleitung zwischen Spannungsquelle und Last stattfindet. In den meisten Fällen befindet sich an der Stelle des Voltmeters in der Abbildung eine Signalaufbereitungsschaltung, welche die Messwerte verstärkt und die bei der High-Side-Messung gegenüber der Low-Side-Messung komplexer und teurer ist. [73]

Es wurde ein Messwiderstand mit einem realen Wert von 1,2 Ω verwendet. Bei Einsatz eines kleineren Messwiderstandes von z.B. 0,1 Ω wäre der zu messende Spannungsabfall sehr klein und damit nur mit größerem Aufwand erfassbar. Um den zusätzlich verursachten Leistungsabfall über die Messschaltung so gering wie möglich zu halten, wird andererseits ein möglichst geringer Messwiderstand angestrebt. Bei Verwendung eines größeren Messwiderstandes von z.B. 10 Ω steigt die

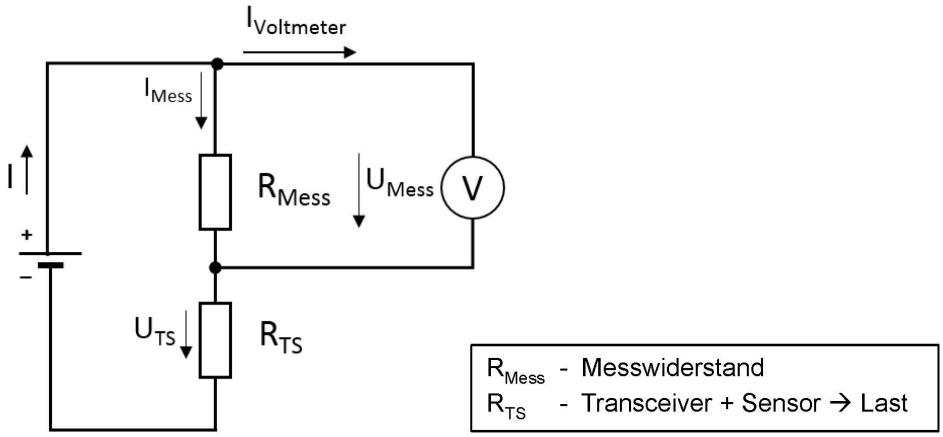


Abb. 30 vereinfachte High-Side-Messung

über diesen Widerstand abfallende Spannung. Laut den Datenblättern wird im Sendebetrieb ein Strom von maximal etwa 220 mA erwartet (s. Kapitel 3.4). Bei einem Messwiderstand von 10 Ω würde sich ein Spannungsabfall von 2,2 V ergeben. Bei 5 V Versorgungsspannung blieben dann noch etwa 2,8 V übrig, mit denen der Microcontroller ESP8266 nicht betrieben werden könnte.

An der Stelle des Voltmeters in Abbildung 29 befindet sich in der realen Messschaltung der Microcontroller Arduino Nano. Dieser ermöglicht ein Loggen der Spannung im Zeitverlauf, was mit den zur Verfügung stehenden Multimetern nicht realisiert werden kann. Im Vergleich zu dem als Transceivermodul eingesetzten ESP8266 besitzt der Arduino Nano mit einer Taktfrequenz von 16MHz, 32kB Flash-Speicher und 2kB Arbeitsspeicher deutlich geringere Leistungsdaten und kein WLAN-Modul, dafür aber 8 analoge Eingänge mit einem integrierten 10-Bit-ADC. [5]

Die über den Messwiderstand abfallende Spannung wird als analoges Signal an einen Eingang des Arduinos angelegt und durch dessen integrierten ADC digitalisiert. Die Stromversorgung des Arduinos erfolgt extern über ein an einen PC angeschlossenes USB-Kabel. Über diesen USB-Anschluss wird auch das Anzeigen der ausgelesenen Werte ermöglicht. Mithilfe der Arduino-Programmierumgebung wurde das im Anhang 4 abgebildete Messskript erstellt, welches auf dem Arduino ausgeführt wird und alle 100 ms die digitalisierten Messwerte aus dem ADC ausliest sowie ein Zeitsignal hinzufügt. Die Daten werden so mit einem Zeitsignal geloggt und auf dem seriellen Monitor in der Arduino-Programmierumgebung auf dem Rechner angezeigt. Von dort werden diese zur weiteren Auswertung in Excel kopiert.

Bei der Digitalisierung wird einem Bereich von Analogwerten ein diskreter Digitalwert zugeordnet. Ein 10-Bit-ADC stellt 2^{10} = 1024 Werte als sog. Quantifizierungsstufen zur Verfügung. Mit einer Eingangsspannung von 5 V ergibt sich die folgende

Auflösung des Wandlers, welche die Breite des Analogwertebereichs darstellt, die eine Quantifizierungsstufe beinhaltet [7]:

$$Auflösung\ U_{ADC} = \frac{U_B}{n} = \frac{5V}{1024} = 4{,}8\,mV \tag{6.3}$$

Auflösung U_{ADC}	...	Auflösung des Spannungswertes durch ADC
U_B	...	Versorgungsspannung ADC/Arduino (5 V)
n	...	Anzahl Quantifizierungsstufen (1024 bei 10-Bit-ADC)

Für den Strom ergibt sich mit dem realen Widerstandswert von 1,2 Ω damit als Auflösung:

$$Auflösung\ I_{ADC} = \frac{Auflösung\ U_{ADC}}{R_{Mess}} = \frac{4{,}8\,mV}{1{,}2\,\Omega} = 4\,mA \tag{6.4}$$

Auflösung I_{ADC}	...	Auflösung des Stromwertes durch ADC

Die Messunsicherheit dieses Messaufbaus liegt somit allein aufgrund des sog. Quantifizierungsfehlers durch die Auflösung des Wandlers und ohne Berücksichtigung anderer Ursachen wie z.B. Verstärkungs- oder Linearitätsfehlern bei 4 mA. [8]

Laut den in Abschnitt 3.4 dargestellten Datenblattwerten ergibt sich ein zu realisierender Messbereich von etwa 0 bis 220 mA. Mit einer Auflösung des Messsystems von 4 mA stünden nur 55 Quantifizierungsstufen zur Abbildung dieses Bereiches zur Verfügung und eine Angabe wäre vor allem bei den kleinen Messwerten in Bezug auf die Größe des Wertes ungenau. Es wurde deshalb die Vorgabe gewählt, auf 1 mA aufzulösen. Laut Goldener Regel der Messtechnik muss zehnmal genauer gemessen werden, als der Wert angegeben werden soll, sodass eine Messung auf 0,1 mA erfolgen muss.

Mit dem 10-Bit-ADC des Arduinos ist dies nicht möglich, weshalb ein Operationsverstärker (OPV) eingesetzt wird. Der OPV verstärkt den analogen Spannungswert, bevor dieser in den Arduino eingelesen wird und ermöglicht eine feinere Auflösung des Messwertes. Alternativ wäre anstatt des OPVs der Einsatz eines 16-Bit-ADCs mit 65536 (= 2^{16}) Quantifizierungsstufen und einer entsprechend kleineren Auflösung denkbar.

Mit Einsatz des OPVs ergibt sich der in Abbildung 31 dargestellte Aufbau der Messschaltung. Der Messwiderstand befindet sich zwischen dem Sensorknoten und dem Masseanschluss der Spannungsquelle. Die Versorgungsspannung des OPVs von 5 V wird durch den Arduino bereitgestellt. Der OPV wird mit den in der Abbildung als R_4 und R_5 bezeichneten Widerständen so beschaltet, dass dieser als nichtinvertierender Gleichspannungsverstärker dient. Für den Verstärkungsfaktor v gilt der folgende Zusammenhang:

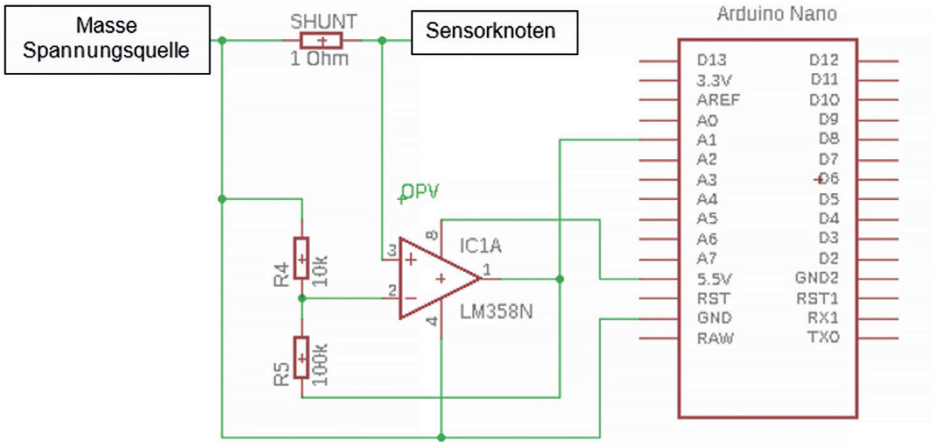


Abb. 31 Aufbau der Messschaltung mit Messwiderstand (engl. Shunt), OPV und Arduino Nano

$$v = 1 + \frac{R_5}{R_4} \quad [48] \tag{6.5}$$

v ... Verstärkungsfaktor

Abbildung 32 zeigt die Verstärkungskennlinie des OPVs für einen Verstärkungsfaktor von 10 sowie von 20. Der Anstieg der Linie entspricht dem Verstärkungsfaktor. Ein idealer OPV kann die Eingangsspannung U_E maximal auf die Höhe seiner Versorgungsspannung U_B verstärken, welche in diesem Messsystem 5 V beträgt. Bei realen OPVs reicht der Bereich der Ausgangsspannung U_A nicht vollständig an U_B heran. Der eingesetzte OPV „LM358“ ermöglicht laut Datenblatt [48] eine Ausgangsspannung mit einer Größe von bis zu U_B – 1,5*V*, sodass sich eine maximale Ausgangsspannung $U_{A_{max}}$ von (5 V – 1,5 V =) 3,5 V ergibt.

Der Verstärkungsfaktor ist deshalb so zu wählen, dass der größte Wert des abzubildenden Messbereiches auf maximal 3,5 V verstärkt wird. Bei 220 mA als höchsten Wert des Messbereiches fällt über dem Messwiderstand die folgende Spannung ab, welche die Eingangsspannung U_E des OPVs ist:

$$U_{E_{max}} = I_{max} * R_{Mess} = 220\ mA * 1{,}2\ \Omega = 264\ mV \tag{6.6}$$

$U_{E_{max}}$ ... Eingangsspannung OPV bei I_{max}
I_{max} ... größter Stromwert des Messbereiches

In Abbildung 32 ist erkennbar, dass mit v = 20 ab 175 mV alle Werte auf 3,5 V verstärkt werden. Alle darüberliegenden Eingangsspannungswerte werden ebenfalls auf 3,5 V verstärkt. Um den Messbereich bis 220 mA vollständig abzubilden, wäre

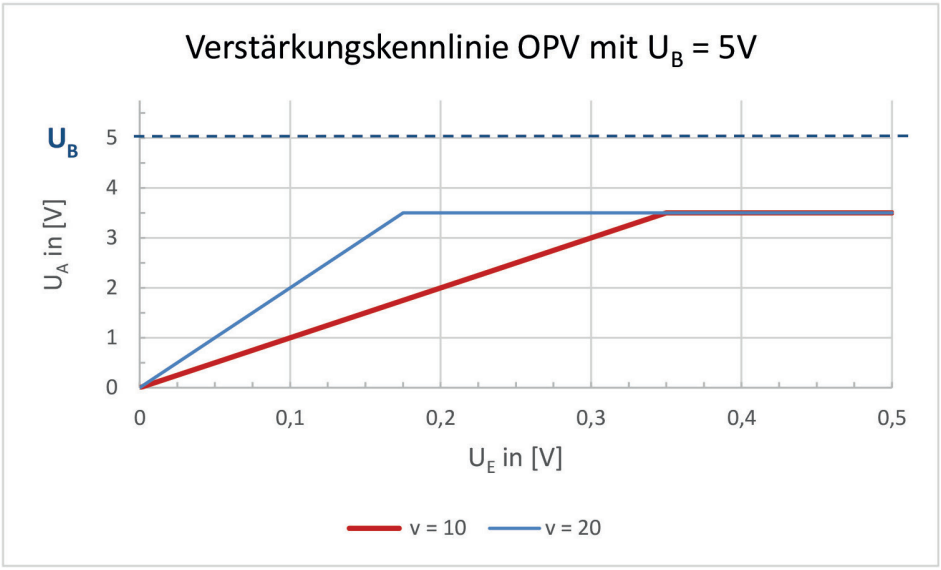


Abb. 32 Verstärkungskennlinie OPV

dieser Verstärkungsfaktor zu hoch gewählt und nicht geeignet. Der Verstärkungsfaktor lässt sich als Verhältnis der Ausgangs- zur Eingangsspannung ausdrücken, sodass sich für den abzubildenden Messbereich der folgende maximale Verstärkungsfaktor ergibt:

$$v_{max} = \frac{U_{A_{max}}}{U_{E_{max}}} = \frac{3{,}5\,V}{0{,}264\,V} = 13{,}3 \tag{6.7}$$

v_{max} ... maximaler Verstärkungsfaktor bei Messbereich I = 0 ... 220 mA
$U_{A_{max}}$ ... maximale Ausgangsspannung OPV bei V_B = 5 V
$U_{E_{max}}$ ... Eingangsspannung OPV bei $I = 220\ mA$

Es wird bei einer größeren Verstärkung durch die Mitverstärkung der Messfehler eine stärkere Verfälschung der Messwerte hervorgerufen. Gleichzeitig wird zur möglichst feinen Auflösung der Messwerte eine hohe Verstärkung angestrebt. Zudem verhält sich ein System im unteren Bereich der Kennlinie zumeist nichtlinear. Mit einer größeren Verstärkung befinden sich die kleineren Messwerte eher im linearen Bereich.

Es wurde deshalb ein Verstärkungsfaktor in der Größenordnung von v_{max} angestrebt. Laut Gleichung 6.5 ist die Verstärkung abhängig von den Werten der Widerstände, mit denen der OPV beschaltet wird. Es wurden Widerstände mit den realen Werten R_4 = 10,01 kΩ und R_5 = 100,6 kΩ gewählt, mit denen sich unter Verwendung der Gleichung 6.5 der folgende Verstärkungsfaktor ergibt:

$$v = 1 + \frac{R_5}{R_4} = 1 + \frac{100{,}6k\,\Omega}{10{,}01k\,\Omega} = 11{,}05 \tag{6.8}$$

Das über dem Messwiderstand abfallende Spannungssignal wird durch den OPV mit einem Faktor von 11,05 verstärkt. Diese Spannung wird an den analogen Eingang des Arduinos angelegt und anschließend digitalisiert. Die bei einem Strom von 220 mA über den Messwiderstand abfallende Spannung $U_{E_{max}}$ von 264 mV wird auf eine Ausgangsspannung von (0,264 V * 11,05 =) 2,9 V verstärkt. Es stehen nun (55 * 11,05 =) 608 Quantifizierungsstufen zur Auflösung des Messbereiches von 0 bis 220 mA zur Verfügung, sodass eine feinere Auflösung des Messwertes erfolgt. Es wird nun folgende Auflösung des Stroms erreicht:

$$Auflösung\ I = \frac{Auflösung\ I_{ADC}}{v} = \frac{4\ mA}{11{,}05} = 0{,}36\ mA \approx 0{,}4\ mA \tag{6.9}$$

Auflösung I_{ADC} ... Auflösung des Stromwertes durch ADC (siehe Gleichung 6.4)

Die angestrebte Messung auf 0,1 mA wird nicht erreicht. Eine Messung auf 0,4 mA wird für den Messzweck des Ermittelns des Lastprofils mit den entsprechenden Zeitdauern der einzelnen Betriebsmodi und der Abschätzung des Stromverbrauchs als ausreichend erachtet.

Mit dem gewählten Verstärkungsfaktor von 11,05 wird bei einem Strom von 220 mA die maximale Ausgangsspannung des OPVs von 3,5 V nicht vollständig ausgenutzt. Der folgende Stromfluss im Messwiderstand verursacht einen Spannungsabfall, welcher mit einem Verstärkungsfaktor von 11,05 auf genau 3,5 V verstärkt wird:

$$I_{max} = \frac{U_{A_{max}}}{R_{Mess} * v} = \frac{3{,}5\ V}{1{,}2\ \Omega * 11{,}05} = 264\ mA \tag{6.10}$$

Das aufgebaute Messsystem besitzt somit einen Messbereich von 0 bis 264 mA und eine Auflösung von 0,4 mA.

Mit dem auf dem Arduino ablaufenden Messskript (s. Anhang 4) erfolgt die folgende Berechnung 6.12, welche ein direktes Anzeigen des indirekt gemessenen Stroms in dem seriellen Monitor auf dem Rechner ermöglicht. x stellt den aus dem ADC ausgelesenen digitalisierten Wert dar.

$$I = Auflösung\ I * x \tag{6.11}$$

x ... aus ADC ausgelesener digitalisierter Wert

mit Gleichung 6.3 ... $Auflösung\ U_{ADC} = \frac{U_B}{n}$,

mit Gleichung 6.4 ... $Auflösung\ I_{ADC} = \frac{Auflösung\ U_{ADC}}{R_{Mess}}$ und

mit Gleichung 6.9 ... $Auflösung\ I = \frac{Auflösung\ I_{ADC}}{v}$ ergibt sich:

$$I = \frac{\frac{U_B}{n} * x}{v * R_{Mess}} \tag{6.12}$$

6.1.2 Übersicht über gesamten Messaufbau

Das Ziel des in Abschnitt 6.1.1 beschriebenen Messkonzepts besteht in der Ermittlung der Stromaufnahme des Sensorknotens im Zeitverlauf. Abbildung 33 zeigt die konkrete Verschaltung aller Komponenten des Messaufbaus. Es sind die drei

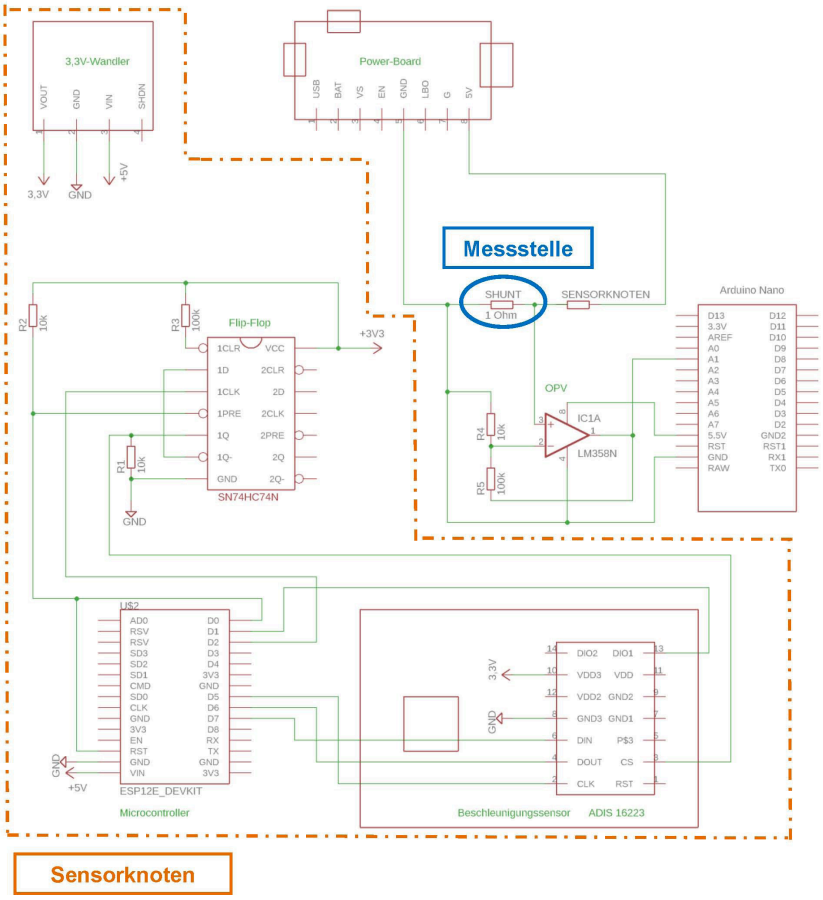


Abb. 33 Überblick gesamter Aufbau mit Sensorknoten, Spannungsversorgung und Messschaltung

Tab. 4 im Messszenario verwendete Bauteile

Sensorknoten	Entwicklerboard NodeMCU mit Microcontroller ESP8266 und integriertem Spannungswandler 5 V auf 3,3 V Beschleunigungssensor ADIS 16223 Flip-Flop SN74HC74N Pullup-/Pulldown-Widerstände 10 kΩ bzw. 100 kΩ
Spannungsversorgung	Spannungswandler 5 V auf 3,3 V Power-Board
Messschaltung	Arduino Nano Operationsverstärker LM358N 10 kΩ- und 100 kΩ-Widerstand Messwiderstand 1 Ω

Baugruppen Sensorknoten, Spannungsversorgung und Messchaltung mit den in Tabelle 4 aufgeführten Bauteilen zu erkennen.

Die Spannungsversorgung der Bauteile des Sensorknotens erfolgt über einen 5 V-USB-Anschluss und ein Power-Board, von welchem aus die Spannung über Jumper-Kabel abgegriffen werden kann. Zwei Spannungswandler wandeln die Spannung auf die benötigten 3,3 V. Ein Spannungswandler ist auf dem Entwicklerboard des Transceivermoduls integriert. Er verursacht im Sleep-Modus einen Strom von etwa 10 mA. Dieser Wandler kann nicht zur Versorgung der anderen Komponenten des Sensorknotens verwendet werden, weshalb ein zweiter Spannungswandler eingesetzt wird.

Der Messwiderstand befindet sich an der in Abbildung 33 eingezeichneten Messstelle zwischen dem Masseanschluss des Powerboards und dem eingezeichneten Widerstand „Sensorknoten“, welcher alle Komponenten des Sensorknotens zusammenfasst und zudem den externen Spannungswandler enthält. Es wird durch diese Messstelle die Stromaufnahme aller Komponenten des Sensorknotens sowie des externen Spannungswandlers erfasst. Nicht berücksichtigt wird die Stromaufnahme des Power-Boards.

Die folgende Tabelle 4 fasst die verwendeten Bauteile des in Abbildung 33 dargestellten gesamten Messaufbaus zusammen.

In diesem Messkonzept wird die Versorgungsspannung einzelner Bauelemente von insgesamt vier verschiedenen Komponenten zur Verfügung gestellt. Die folgende Tabelle 5 gibt einen Überblick, welches Bauteil von welcher Spannungsquelle versorgt wird.

Der reale Aufbau wurde auf Steckplatinen und mit Jumper-Kabeln durchgeführt. Gegenüber einem festen Aufbau ist er mit Stecken der entsprechenden Komponenten und Kabel simpel zu realisieren, besitzt aber den Nachteil häufig auftretender Fehler durch Spannungsverluste aufgrund von Wackelkontakten und durch Bildung ungewollter Kapazitäten.

Tab. 5 Übersicht Spannungsversorgung

	5V Power-Board	3,3 V-Wandler	PC	Arduino Nano
3,3 V-Wandler	X			
Entwicklerboard mit ESP8266	X			
Arduino Nano			X	
Sensor		X		
Flip-Flop		X		
OPV				X

6.2 Ergebnisse der Untersuchung und Gegenüberstellung der Resultate

Der Arduino wird an einen Rechner angeschlossen. Es werden in einem sog. seriellen Monitor der Arduino-Programmierumgebung die gemessenen Stromwerte und die Zeitdauer angezeigt, welche seit Öffnen des seriellen Monitors vergangen ist. Das Auslesen des ADCs und damit die Erfassung des Stroms erfolgt alle 100 ms. Diese Werte wurden von dort aus in ein Excel-Dokument kopiert, in dem eine Auswertung vorgenommen wurde.

Es wurden etwa 500 s mit 15 Messzyklen des MQTT-Programms und 16 Zyklen mit HTTP ausgewertet. Für das Messszenario wurde im Messskript eine Dauer des Sleep-Modus von 10 s hinterlegt. Es erfolgte eine Einteilung der Messwerte in die Betriebszustände Grundzustand, Sendebetrieb und Sleep-Modus analog zu Kapitel 3.4. Der Betriebszustand Messen wurde nicht mehr separat betrachtet, da sich die Stromaufnahme nicht von der des Sendebetriebs unterscheidet. Der Zustand Sendebetrieb enthält nun zusätzlich die etwa 0,7 s des Messvorgangs.

Abbildung 34 zeigt die graphische Darstellung der Stromaufnahme des Sensorknotens bei Ablaufen des MQTT-Messprogramms mit 15 Messzyklen im Verlauf der Zeit. Eine entsprechende Darstellung für HTTP befindet sich in Anhang 5. Es sind im Zeitverlauf Stromaufnahmen verschiedener Größenordnung zu erkennen, welche die unterschiedlichen Betriebszustände kennzeichnen. Innerhalb der einzelnen Betriebsmodi schwanken die Werte. Deshalb wurde für jeden der Messzyklen die Länge der einzelnen Betriebsmodi bestimmt sowie der Mittelwert der Stromaufnahme innerhalb des einzelnen Zustands ermittelt, um das Signal für eine Bewertung zu glätten.

Stellt man die erhaltenen Mittelwerte in Abhängigkeit von der Zeit grafisch dar, erhält man das in Abbildung 36 dargestellte Diagramm für MQTT und in Abbildung 35 eine Darstellung für HTTP. Es ist zu erkennen, dass die Mittelwerte zwischen den Messzyklen leicht schwanken. Im Vergleich der beiden Diagramme von MQTT und HTTP wird ersichtlich, dass in der gleichen Zeitspanne von 500 s mit HTTP ein Messzyklus mehr durchgeführt werden kann. Weiterhin fällt die

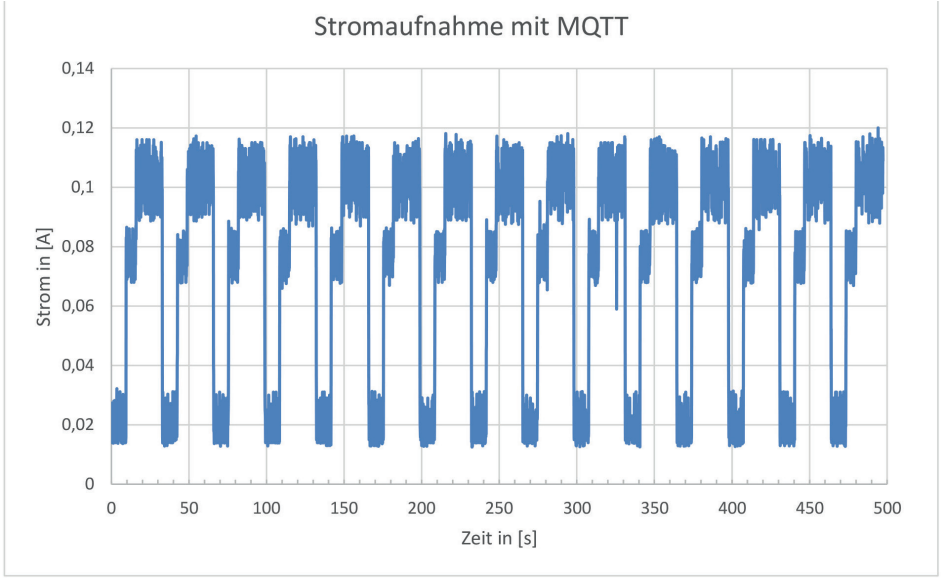


Abb. 34 Stromaufnahme im Zeitverlauf des Sensorknotens mit MQTT-Messprogramms

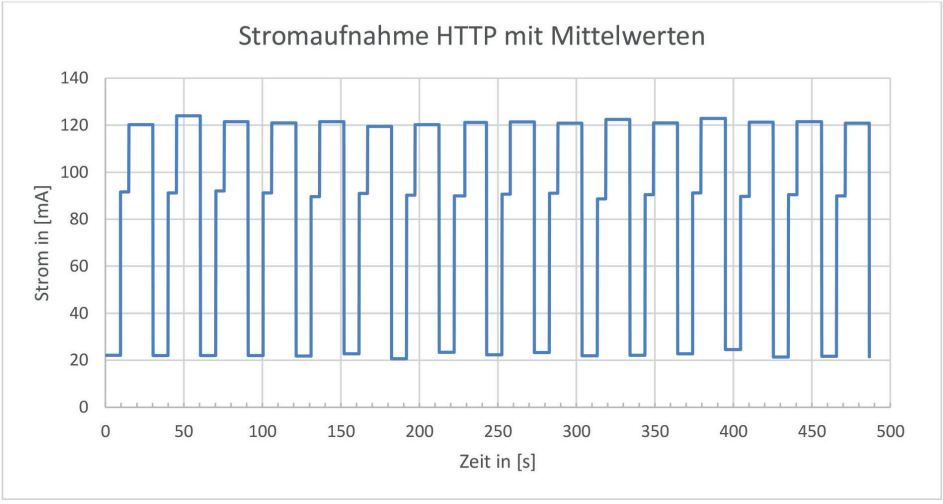


Abb. 35 Diagramm mit Mittelwerten der Stromaufnahme über einzelne Betriebszustände des HTTP-Programms

unterschiedliche Stromaufnahme auf, die im Sendebetrieb bei der MQTT-Messung etwa 20 mA unter der von HTTP liegt.

Die Tabelle in Anhang 6 zeigt die Mittelwerte der Stromaufnahme und die Dauer der Betriebsmodi eines jeden Messzyklus für MQTT. Anhang 7 stellt diese Werte

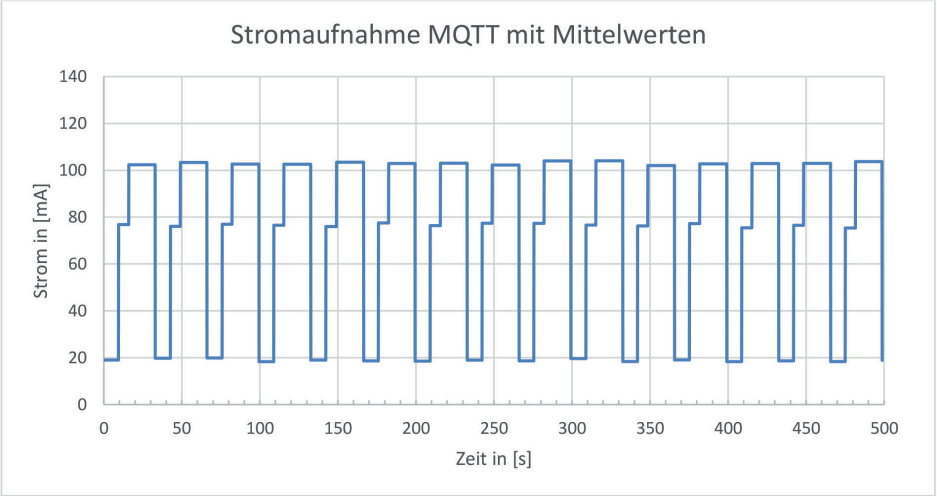


Abb. 36 Diagramm mit Mittelwerten der Stromaufnahme über einzelne Betriebszustände des MQTT-Programms

Tab. 6 Dauer Betriebszustände und durchschnittliche Stromaufnahme des Sensorknotens für Messprogramm mit HTTP und MQTT

	HTTP		MQTT	
Betriebszustand	**Dauer in [s]**	**Stromaufnahme in [mA]**	**Dauer in [s]**	**Stromaufnahme in [mA]**
Grundzustand	5.32	90.57	6.41	76.57
Sendebetrieb	15.27	121.33	17.08	103
Sleep-Modus	9.55	22.32	9.46	18.89

für HTTP dar. Aus den Werten der einzelnen Messzyklen wurde anschließend ein Mittelwert für jeden Betriebszustand gebildet. Es ergeben sich die in der folgenden Tabelle 6 dargestellten Werte.

Die Gegenüberstellung in Tabelle 6 zeigt, dass anders als erwartet keine Verkürzung der Sendedauer durch den Einsatz von MQTT erreicht werden konnte. Die Dauer des Sendebetriebs liegt bei MQTT mit durchschnittlich 17,08 s über der von HTTP mit 15,27 s. Allerdings ist die Stromaufnahme des Sensorknotens gesunken. Berechnet man mit

$$Q = I * t \quad [24] \tag{6.13}$$

Q ... elektrische Ladung (Elektrizitätsmenge)

Tab. 7 Ermittlung der elektrischen Ladung je durchschnittlichen Messzyklus

	HTTP			MQTT		
Betriebszustand	**Dauer in [s]**	**Stromaufnahme in [mA]**	**elektr. Ladung in [mAs]**	**Dauer in [s]**	**Stromaufnahme in [mA]**	**elektr. Ladung in [mAs]**
Grundzustand	5.32	90.57	481.83	6.41	76.57	490.81
Sendebetrieb	15.27	121.33	1852.71	17.08	103	1759.24
Sleep-Modus	9.55	22.32	213.16	9.46	18.89	178.70
Summe			2547.70			2428.75

Tab. 8 Berechnung Anzahl Messzyklen je Akkuladung

		HTTP	MQTT
je Messzyklus	elektr. Ladung in [mAs]	2547.70	2428.75
	elektr. Ladung in [mAh]	0.7077	0.6747
Anzahl Messzyklen je Akkuladung (2500 mAh)		3533	3706

die elektrische Ladung Q je durchschnittlichen Betriebszustand, erhält man die in Tabelle 7 dargestellten Werte. Je Messzyklus konnte lt. den Messwerten durch das MQTT-Messprogramm eine Verringerung der Elektrizitätsmenge um 5% erreicht werden.

Eine Akkuladung der in Abschnitt 3.1.3 vorgestellten Energieversorgung des Sensorknotens enthält eine Elektrizitätsmenge von 2500 mAh. Es ergibt sich unter der idealisierten Annahme der vollständigen Ausnutzung der gesamten Kapazität des Akkus die in Tabelle 8 abgebildete Anzahl möglicher Messzyklen je Akkuladung. Mit dem Einsatz des MQTT-Messprogramms sind 173 Messzyklen mehr möglich. Dies entspricht einer Steigerung um 5 %.

Stellt man die in Tabelle 6 abgebildeten Werte in Form eines Lastprofils für einen durchschnittlichen Messzyklus des Sensorknotens dar, erhält man die Abbildung 37. Es sind die Betriebszustände Sleep-Modus, Grundzustand und Sendebetrieb erkennbar. Im Grundzustand wird das in Kapitel 5.1 vorgestellte Initialisierungsskript ausgeführt, im Sendebetrieb das Messskript. Im Vergleich der beiden Lastprofile des Sensorknotens mit HTTP- sowie MQTT-Messprogramm wird deutlich, dass die angestrebte Verkürzung der Sendedauer nicht realisiert werden konnte, die Stromaufnahme im Sendebetrieb aber um etwa 20 mA gesunken ist.

Die in Tabelle 1 dargestellte Stromaufnahme von 250 µA bei 3 V im Sleep-Modus kann mit diesem Messaufbau nicht erreicht werden. Die beiden Spannungswandler und auftretende Verluste durch den gesteckten Schaltungsaufbau verursachen einen Strom von etwa 20 mA im Sleep-Modus und wurden in Kapitel 3.4 nicht

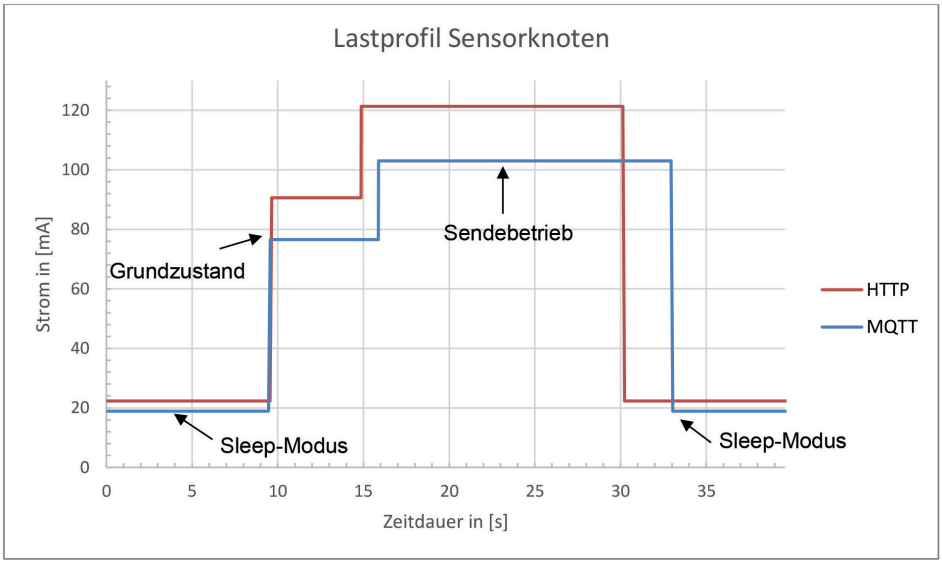


Abb. 37 Lastprofil Sensorknoten

berücksichtigt. Allein der auf dem Entwicklerboard integrierte Spannungswandler verbraucht laut Datenblatt [62] 6 bis 10 mA. Für einen Aufbau mit möglichst langer energetischer Lebensdauer müsste ein fester Verbau der Komponenten und ein Verwenden von im Ruhezustand ebenfalls abschaltbaren Spannungswandlern erfolgen. Nimmt man an, dass dadurch im Sleep-Modus eine Stromaufnahme des Sensorknotens von 250 µA ermöglicht wird, ergibt sich die im Folgenden abgeschätzte energetische Lebensdauer für HTTP und MQTT. Es wurde analog zu Kapitel 3.4 von einer stündlichen Messung ausgegangen.

Analog zu Tabelle 7 wurde in Tabelle 9 die Elektrizitätsmenge eines durchschnittlichen Messzyklus bei einer stündlichen Messung ermittelt. Die Dauer des

Tab. 9 Ermittlung der elektrischen Ladung je durchschnittlichem Messzyklus bei stündlicher Messung

	HTTP			MQTT		
Betriebszustand	**Dauer in [s]**	**Stromaufnahme in [mA]**	**elektr. Ladung in [mAs]**	**Dauer in [s]**	**Stromaufnahme in [mA]**	**elektr. Ladung in [mAs]**
Grundzustand	5.32	90.57	481.83	6.41	76.57	490.81
Sendebetrieb	15.27	121.33	1852.71	17.08	103	1759.24
Sleep-Modus	3579.41	0.25	894.85	3576.51	0.25	894.13
Summe			3229.39			3144.18

Sleep-Modus beträgt nicht wie in Tabelle 7 etwa 10 s. Sie ergibt sich nun aus der Differenz zwischen einer Stunde und der Zeit, die für Grundzustand und Sendebetrieb benötigt werden. Anschließend wurde in Tabelle 10 diese Elektrizitätsmenge auf eine Woche hochgerechnet, in der 168 stündliche Messungen durchgeführt werden.

Aus Abbildung 16 kann entnommen werden, dass der Vibrations-Harvester je Woche eine Energiemenge von $\left(7\frac{Tage}{Woche} * 322{,}5\frac{Ws}{Tag}\right) = 2257{,}5\frac{Ws}{Woche}$ bereitstellt. Zwischen elektrischer Ladung und Energie besteht der folgende Zusammenhang.

$$W = Q * U \tag{6.14}$$

W ... Elektrische Energie/Arbeit

Der zur Energieversorgung des Sensorknotens eingesetzte Lithium-Polymer-Akku besitzt eine Kapazität von 2500 mAh (= 9000 As). Bei einer Nennspannung von 3,7 V ergibt sich eine Energiemenge des Akkus von (9000 As * 3,7 V =) 33.300 Ws. Es wird weiterhin von der idealisierten Annahme der vollständigen Ausnutzung der gesamten Kapazität des Akkus ausgegangen. Die Stromaufnahme des Sensorknotens wurde bei einer Versorgungsspannung von 5 V ermittelt. Es ergibt sich die folgende Tabelle 10. Diese zeigt, dass der Sensorknoten sowohl mit HTTP als auch mit MQTT mehr Energie aufnimmt, als der Harvester aus der Umgebung „erntet". Dividiert man die durch den Akku bereitgestellte Energiemenge von 33.300 Ws durch diese entstehende Differenz, ergibt sich eine energetische Lebensdauer von etwa 87 Wochen bei MQTT und von 73 Wochen bei HTTP. Der Sensorknoten lässt sich durch den Einsatz von MQTT etwa 14 Wochen länger betreiben.

Tab. 10 Ermittlung der energetischen Lebensdauer des Sensorknotens

		HTTP	MQTT
elektr. Ladung Q je Woche (stündliche Messung) in [As]		542.5	528.2
Energiespeichervermögen Akku in [Ws]		33300	
Energiemenge W je Woche in [Ws]	Verbraucher Sensorknoten	2712.7	2641.1
	Harvester	2257.5	
	Differenz Verbraucher - Harvester	455,2	383.6
energetische Lebensdauer in [Wochen]		73.2	86.8

Die durchgeführten Betrachtungen zeigen, dass durch den Einsatz von MQTT eine Steigerung der Energieeffizienz des Sensorknotens eingetreten ist. Diese ergibt sich im Gegensatz zu den Erwartungen nicht aus der Reduzierung der Sendedauer, sondern aus einer Verringerung der Stromaufnahme.

6.3 Nachbetrachtungen

MQTT bietet als Protokoll zur Übertragung von Sensordaten Vorteile. Die größte Verbesserung von MQTT ist die dauerhaft bestehenbleibende Verbindung, welche nicht wie bei HTTP nach einer gewissen Zeit oder Menge an übertragener Nachrichten, wie in Kapitel 3.2 beschrieben, unterbrochen werden muss. Der Neuaufbau einer Verbindung entspricht dem Betriebsmodus Grundzustand und nimmt lt. Tabelle 6 bei HTTP durchschnittlich 5,32 s ein. Weitere Vorteile bestehen in der Publish/Subscribe-Logik und in der geringen vom Protokoll verursachten Datenmenge je Nachricht. Es wird eine streaming- und echtzeitfähige Kommunikation ermöglicht.

Es konnte durch den Einsatz von MQTT anstatt von HTTP keine Reduzierung der Sendedauer, aber durch Verringerung der Stromaufnahme eine Senkung der je Messzyklus benötigten Elektrizitätsmenge von 5 % und damit eine Verlängerung der energetischen Lebensdauer des Sensorknotens erreicht werden.

Es wurde zur Messdurchführung der identische Aufbau mit denselben Hardwarekomponenten verwendet. Die Entfernung zum RPI als Router blieb ebenfalls konstant. Zunächst erfolgte die Messung des MQTT-Programms. Anschließend wurde auf dasselbe Transceivermodul das HTTP-Messprogramm aufgespielt. Eine hardwarebedingte Änderung der Stromaufnahme könnte durch einen Temperaturdrift infolge der andauernden Verwendung des Moduls eingetreten sein. Eine Veränderung in der eingetretenen Größenordnung von etwa 20 mA scheint allerdings aus diesem Grund nicht plausibel. Eine Verringerung der Stromaufnahme durch den Einsatz von MQTT anstatt HTTP ist zudem ebenfalls in der in [47] dargestellten Untersuchung eingetreten.

Im Betriebsmodus Grundzustand wird das Initialisierungsskript ausgeführt. Die Verlängerung des Grundzustandes durch MQTT kann darauf zurückgeführt werden, dass im Initialisierungsskript der Aufruf des Messskriptes erfolgt. Es werden das Messskript und die zur Ausführung benötigten Bibliotheken geladen. Sind diese bei MQTT umfangreicher als bei HTTP, wird mehr Speicherplatz in Anspruch genommen und eine größere Aufrufedauer verursacht. Aufgrund der durchgeführten Betrachtungen erscheint das ermittelte Lastprofil mit der Verringerung der Stromaufnahme und der Verlängerung des Grundzustandes durch MQTT plausibel.

Mit den folgenden Überlegungen wird untersucht, wie sich die Zeitdauer des Zustandes Sendebetrieb zusammensetzt. Ein Teil der Zeit entsteht durch den Messvorgang, welcher etwa 0,7 s in Anspruch nimmt.

Nachdem der Sensor die Messung durchgeführt und die Messwerte in den Zwischenspeicher geschrieben hat, erfolgt die schrittweise Übertragung dieser Werte über die SPI-Verbindung zwischen Sensor und Microcontroller. Abbildung 38 zeigt die Funktionen, die bei einem einzelnen Auslesevorgang ablaufen.

Zu Beginn wird die in der Abbildung 38 links dargestellte Funktion „read_reg“ aufgerufen. Diese wiederum ruft mit der ersten Zeile die rechts abgebildete Funktion „SET_CS“ auf, welche mit dem „tmr.delay“ zweimal den weiteren Programmablaufs für 15µs anhält. Anschließend wird die erste 16-Bit-Sequenz an den Sensor übertragen. Mit dieser erfolgt das Anschreiben der Adresse des Zwischenspeichers des Sensors, in dem sich die Messwerte befinden. Im Anschluss wird eine sog. 16-Bit

```
function read_reg(PIN, address)
    SET_CS(PIN)
    spi.send(1, address);
    tmr.delay(15);
    wrote, hi_byte = spi.send(1, 0x0000)
    SET_CS(PIN)
    data = hi_byte
    tmr.delay(15);
    return data;
end
```

```
function SET_CS(PIN)
gpio.write(PIN, gpio.LOW)
tmr.delay(15)
gpio.write(PIN, gpio.HIGH)
tmr.delay(15)
end
```

Abb. 38 Funktionen zum Auslesen des Sensors

Dummy-Sequenz gesendet, welche ausschließlich aus Nullen besteht. Nachdem eine 16-Bit-Sequenz übertragen wurde, wird ein „delay“ von 15µs programmiert. Diese Unterbrechung ist notwendig und wird vom Hersteller empfohlen, um Fehlkommunikationen zu vermeiden. Wird eine 16-Bit-Sequenz an den Sensor übermittelt, erhält das Transceivermodul zur gleichen Zeit eine Sequenz auf dem anderen Datenkanal vom Sensor zurück.

Zählt man alle aufgerufenen „delays“ der in Abbildung 38 dargestellten Funktionen zusammen, werden dadurch je Auslesevorgang 6 * 15 µs = 90 µs in Anspruch genommen. Um die Dauer eines gesamten Vorgangs zu ermitteln, muss dazu die Zeit addiert werden, welche die Datenübertragung über die SPI-Verbindung verursacht. Es erfolgt je SPI-Datenkanal eine Übermittlung von zwei 16-Bit-Sequenzen mit insgesamt 32 Bit. Der Sensor besitzt eine Taktrate von 2,25 MHz, der Microcontroller von 80 MHz. Die Taktfrequenz der SPI-Verbindung darf maximal so hoch sein, wie die des schwächeren Teilnehmers. Zur Konfiguration der SPI-Verbindung im Messskript wird ein sog. Clock-Devider angegeben, welcher die Taktrate des Transceivermodul durch eine 2er-Potenz teilt. Die größtmögliche Taktfrequenz der SPI-Verbindung beträgt mit diesen Randbedingungen $\frac{80\,MHz}{64} = 1{,}25\,MHz$. Das bedeutet, dass jede Sekunde 1.250.000 Bit je Datenkanal übertragen werden können. Für einen Auslesevorgang müssen je Kanal 32 Bit übertragen werden, sodass $\frac{32Bit}{1.250.000\frac{Bit}{s}} = 25{,}6\mu s$ benötigt werden. Zusammen mit den „delays“ des Programms ergibt sich für einen einzelnen Auslesevorgang damit die folgende Zeitdauer: 90 µs + 25,6 µs = 115,6 µs.

Für 3072 Auslesevorgänge des eigentlichen Messprogramms beträgt die Zeitdauer 3072 * 115,6 µs = 355,1 ms. Die Zeitdauer der Datenübertragung mittels SPI-Verbindung ließe sich durch Einsatz eines Sensors mit der gleichen Taktfrequenz von 80 MHz wie die des Microcontrollers auf $3072 * \left(90\mu s + \frac{32Bit}{80.000.000\frac{Bit}{s}}\right) = 3072 * (90\,\mu s +$ 0,4 µs) = 277,7 ms reduzieren. Diese Betrachtung zeigt, dass die Datenübertragung

mittels SPI-Verbindung mit etwa 0,35 s nur einen kleinen Teil der Dauer des Zustandes Sendebetrieb einnimmt.

Es wird vermutet, dass der größte Anteil der restlichen Zeitdauer des Sendebetriebs nicht auf den reinen Versand der Nachrichten entfällt. Es wurde deshalb das folgende Testszenario entwickelt, in dem ebenfalls eine Messung von 3072 Werten durch den Sensor und ein Versand von 24 Nachrichten mit je 128 Werten vom Transceivermodul erfolgt. Der Unterschied zu dem in Kapitel 6.2 verwendeten Messprogramm liegt in der Anzahl an Auslesevorgängen je erstelltem Datenpaket. Es wird insgesamt je Paket nur ein Wert ausgelesen und auf dem Microcontroller zwischengespeichert. Dieser einzelne Messwert wird 128 Mal hintereinander zu einem String zusammengefügt, sodass die Datenmenge einer übertragenen Nachricht identisch mit der Messung in Kapitel 6.2 ist.

Es wurde zur Ermittlung der Sendedauer analog zu Kapitel 6 vorgegangen und ebenfalls ein Zeitraum von etwa 500 s betrachtet. Es ergeben sich für einen Ablauf des veränderten Messprogramms die in Tabelle 11 abgebildeten Mittelwerte.

Tab. 11 Betriebszustände bei einmal Auslesen je Datenpaket für HTTP und MQTT

	HTTP		MQTT	
Betriebszustand	**Dauer in [s]**	**Stromaufnahme in [mA]**	**Dauer in [s]**	**Stromaufnahme in [mA]**
Grundzustand	5.34	117.78	6.39	117.96
Sendebetrieb	4.15	160.95	3.3	160.92
Sleep-Modus	9.39	29.51	9.43	29.94

Im Vergleich mit den Werten der Messprogramme in Tabelle 6 fällt auf, dass die Dauer des Zustandes Sendebetrieb bei HTTP wie auch bei MQTT erheblich reduziert wurde. Anstatt 17,08 s beträgt die Sendedauer mit MQTT nur noch 3,3 s, eine Differenz von 13,78 s. Im Gegenzug stieg die Stromaufnahme um etwa 40 mA bei HTTP und 60 mA bei MQTT auf 161 mA an. Die Sendedauer des MQTT-Messprogramms ist mit 3,3 s 0,85 s geringer als die des HTTP-Programms mit 4,15 s.

Das Szenario zeigt, dass nicht der reine Sendevorgang die größte Zeitdauer in Anspruch nimmt, sondern das schrittweise Auslesen und Zwischenspeichern der Messwerte in einer Tabelle sowie das Erstellen der Zeichenkette mit 128 unterschiedlichen Werten. Eine Reduzierung der für das Versenden der Nachrichten benötigten Zeitdauer besitzt damit kaum Einfluss auf die Dauer des Zustandes Sendebetrieb. Prinzipiell konnte durch den Einsatz von MQTT eine Verringerung der Sendedauer erreicht werden kann.

7 Zusammenfassung und Ausblick

In dieser Bachelorarbeit wurde eine kommunikationstechnische Optimierung eines bestehenden funkbasierten Sensorkonzepts mit dem Ziel der Steigerung der Energieeffizienz durchgeführt. Es erfolgte zunächst eine Analyse des vorhandenen Sensorkonzepts, auf deren Grundlage Optimierungspotentiale erarbeitet wurden. Durch das Ersetzen des bisher verwendeten Anwendungsprotokolls HTTP durch das leichtgewichtige MQTT entstand ein alternatives Sensorkonzept. Es sollte die Dauer des Sendevorgangs verringert werden, welche maßgeblich die Stromaufnahme des Sensorknotens bestimmt.

Es wurde ein Messszenario entwickelt, welches einen quantitativen Vergleich des bisherigen mit dem überarbeiteten Sensorkonzept ermöglicht. Die Stromaufnahme des Sensorknotens wurde inklusive eines Zeitsignals mit Hilfe eines Messwiderstands und einer sich anschließenden Signalerfassungsschaltung erfasst. Aus diesen Daten konnte das Lastprofil des Sensorknotens erstellt und verglichen werden. Dieses Messkonzept kann zudem zur Bestimmung eines Lastprofils zukünftiger Messaufbauten verwendet werden.

Eine Reduzierung der Sendedauer konnte durch den Einsatz von MQTT anstatt von HTTP nicht erreicht werden. Es wurde aber eine Verringerung der Stromaufnahme erzielt, wodurch die Energieeffizienz des Sensorknotens gesteigert wurde. MQTT bietet als Protokoll zur Übertragung von Sensordaten weitere Vorteile. Diese bestehen u.a. in der geringen vom Protokoll verursachten Nachrichtengröße sowie in der dauerhaft bestehenbleibenden Verbindung ohne Timeouts, wodurch eine streaming- und echtzeitfähige Kommunikation ermöglicht wird.

Von weiteren Interesse ist eine programmiertechnische Optimierung des LUA-Messkripts, in welches MQTT implementiert wurde, da theoretisch zumindest eine geringe Reduzierung der Sendedauer hätte eintreten müssen. Es ist zu hinterfragen, ob alle dort ausgeführten Programmzeilen tatsächlich benötigt werden.

Eine weitere Optimierungsmöglichkeit besteht im Einsatz eines Microcontrollers der aktuellen Generation. Das Nachfolgemodell „ESP32“ des in dieser Bachelorarbeit verwendeten Transceivermoduls „ESP8266“ besitzt einen schnelleren Prozessor mit einer Taktfrequenz von 240 MHz sowie mit 512 kByte einen etwa zehnmal größeren Arbeitsspeicher. Es ist eine schnellere Programmausführung zu erwarten. Zudem ermöglicht es der größere Arbeitsspeicher bspw. 1072 Messwerte in einem Feld mit mehreren Spalten auf dem Microcontroller zwischenzuspeichern. Anschließend könnte ein spaltenweises Zusammenfügen der Werte zu einem String und ein darauffolgendes Versenden des Datenpakets erfolgen, wovon eine Reduzierung der Ausführungsdauer erwartet wird.

Die durchgeführte Nachbetrachtung zeigt, dass durch den Einsatz von MQTT anstatt von HTTP unter anderen Bedingungen als denen des Sensorkonzepts eine Reduzierung der Sendedauer erreicht werden kann. Eine Verwendung von MQTT

http://doi.org/10.33968/9783966270069.0007

in zukünftigen Konzepten mit funkbasierter Datenübertragung bietet sich deshalb und vor allem zur Übertragung kleiner Datenpakete oder Einzelwerte aufgrund der oben genannten Vorteile an.

Ein weiterer Ansatz besteht bei der Umsetzung zukünftiger Konzepte in der grundsätzlichen Änderung des Sensorkonzepts. Bspw. könnte auf dem Microcontroller eine Berechnung von Diagnosekennwerten durchgeführt werden. Anstatt aller Messwerte würde die versendete Nachricht dann nur z.B. einen Begriff zur Zustandseinschätzung enthalten. Alternativ ist auch denkbar, dass nur beim Auftreten kritischer Werte eine Nachricht erstellt und versendet wird. Der Vorteil dieser Variante besteht darin, dass weniger und kürzere Nachrichten übermittelt werden müssten.

In dieser Bachelorarbeit wurde auf der Leitwarte zum Loggen der Messwerte in eine CSV-Datei die Software Node-Red eingesetzt, welche eine datenflussbasierte Programmierung ermöglicht. Es ist darüber hinaus eine Umsetzung der Funktionalitäten der auf der Leitwarte ausgeführten Matlab-Applikation mit Node-Red denkbar. Die auf einem sog. Dashboard angezeigten Informationen können über eine URL-Adresse von jedem beliebigen Gerät mit einem Webbrowser wie bspw. einem Smartphone oder Laptop aufgerufen werden, welches sich im gleichen Netzwerk befindet. Es wird eine Datenanzeige ohne Einrichten eines Programms auf diesen externen Geräten ermöglicht.

Der Einsatz von Node-Red könnte sich in zukünftigen Aufbauten ebenfalls anbieten. Mit dem Dashboard könnte zudem nicht nur eine Anzeige, sondern auch eine Eingabe von Informationen erfolgen.

Literaturverzeichnis

[1] 1 & 1 IONUS SE. *Was ist ein Webserver?* https://www.ionos.de/digitalguide/server/knowhow/webserver-definition-hintergruende-software-tipps/ (2016). Zuletzt geprüft am 10.03.2019.

[2] ABB Automation Products GmbH. *ABB Ability™ Smart Sensor. Zustandsüberwachung für Niederspannungsmotoren*. http://search-ext.abb.com/library/Download.aspx?DocumentID=9AKK106713A3853&LanguageCode=de&DocumentPartId=&Action=Launch (2017). Zuletzt geprüft am 26.05.2018.

[3] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., und Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutorials* (2015) 17, 4, 2347–2376.

[4] Analog Devices, Inc. *Data Sheet ADIS16223 (Rev. A). Digital Tri-Axial Vibration Sensor*. http://www.analog.com/media/en/technical-documentation/data-sheets/ADIS16223.pdf (2017). Zuletzt geprüft am 1.04.2018.

[5] Arduino. *ARDUINO NANO. Tech Specs*. https://store.arduino.cc/arduino-nano. Zuletzt geprüft am 19.04.2018.

[6] Bachem, T. *Lexikon. PHP*. https://www.gruenderszene.de/lexikon/begriffe/php. Zuletzt geprüft am 25.05.2018.

[7] Bechtloff, J. *Messtechnik*. Vogel-Studienmodule. 1. Aufl. Vogel, Würzburg (2011).

[8] Bernstein, H. *Messelektronik und Sensoren. Grundlagen der Messtechnik, Sensoren, analoge und digitale Signalverarbeitung*. Lehrbuch. Springer Vieweg, Wiesbaden (2014).

[9] Brandebusemeyer, J., Hübscher, H., and Szapanski, R. *Elektrotechnik. Fachbildung Kommunikationselektronik*. 1. Aufl. Westermann, Braunschweig (1994).

[10] Brendel, H. *Wissensspeicher Tribotechnik. Schmierstoffe, Gleit-, Roll- u. Wälzpaarungen, Schmiereinrichtungen ; mit 61 Tabellen*. 2., neubearb. Aufl. Fachbuchverl., Leipzig (1988).

[11] Bundesamt für Sicherheit in der Informationstechnik. *IT-Grundschutz. M 2.276 Funktionsweise eines Routers*. https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKataloge/Inhalt/_content/m/m02/m02276.html (2013). Zuletzt geprüft am 15.06.2018.

[12] Charara, S. *Zigbee vs Z-Wave: Two big smart home standards explored*. https://www.the-ambient.com/guides/zigbee-vs-z-wave-298 (2018). Zuletzt geprüft am 29.04.2018.

[13] Digi-Key Electronics. *Z-Wave-HF-Module*. https://www.digikey.de/de/product-highlight/s/sigma-designs/z-wave-rf-modules. Zuletzt geprüft am 31.05.2018.

[14] Diskus, C. Energy Harvesting – ein Überblick. *Elektrotech. Inftech*. (2010) 127, 3, 33–38.

[15] Donner, A. *Was ist eine MAC-Adresse? Definition*. https://www.ip-insider.de/was-ist-eine-mac-adresse-a-665074/ (2017). Zuletzt geprüft am 6.05.2018.

[16] Durkop, L., Czybik, B., und Jasperneite, J. (2015) - (2015). Performance evaluation of M2M protocols over cellular networks in a lab environment. *In 2015 18th International Conference on Intelligence in Next Generation Networks*. IEEE, 70–75. DOI=10.1109/ICIN.2015.7073809.

[17] Egli, P. *MQTT - MQ Telemetry Transport for Message Queueing*. https://www.slideshare.net/PeterREgli/mq-telemetry-transport (2013). Zuletzt geprüft am 23.05.2018.

[18] Elektronik Kompendium. *Compiler und Interpreter*. https://www.elektronik-kompendium.de/sites/com/1705231.htm. Zuletzt geprüft am 27.05.2018.

[19] Elektronik Kompendium. *HTTP - Hypertext Transfer Protocol*. http://www.elektronik-kompendium.de/sites/net/0902231.htm. Zuletzt geprüft am 14.04.2018.

[20] Elektronik Kompendium. *ISO/OSI-7-Schichtenmodell*. http://www.elektronik-kompendium.de/sites/kom/0301201.htm. Zuletzt geprüft am 15.04.2018.

[21] Elektronik Kompendium. *Raspberry Pi 2 Modell B*. http://www.elektronik-kompendium.de/sites/raspberry-pi/2002071.htm. Zuletzt geprüft am 29.03.2018.

[22] Elektronik Kompendium. *Raspberry Pi: GPIO - General Purpose Input Output*. https://www.elektronik-kompendium.de/sites/raspberry-pi/2002191.htm. Zuletzt geprüft am 29.03.2018.

[23] Elektronik Kompendium. *TCP/IP*. https://www.elektronik-kompendium.de/sites/net/0606251.htm. Zuletzt geprüft am 16.04.2018.

[24] Erbrecht, R. *Das grosse Tafelwerk interaktiv. Ein Tabellen- und Formelwerk für den mathematisch-naturwissenschaftlichen Unterricht in den Sekundarstufen I und II ; [Mathematik, Informatik, Astronomie, Physik, Chemie, Biologie ; für das Abitur empfohlen]*. 1. Aufl. Cornelsen; Volk und Wissen Verlag GmbH & Co. OHG, Berlin (2003).

[25] Espressif Inc. *ESP8266EX Datasheet. Version 5.8*. https://www.espressif.com/en/products/hardware/esp8266ex/resources (2018). Zuletzt geprüft am 1.04.2018.

[26] Fabasoft. *Bad Request - Size of a request header field exceeds server limit*. https://www.fabasoft.com/en/support/knowledgebase/bad-request-size-request-header-field-exceeds-server-limit (2017). Zuletzt geprüft am 30.05.2018.

[27] Foxworth, T. *MAKING THE ESP8266 LOW-POWERED WITH DEEP SLEEP*. https://www.losant.com/blog/making-the-esp8266-low-powered-with-deep-sleep (2017). Zuletzt geprüft am 27.05.2018.

[28] Gessler, R. and Krause, T. *Wireless-Netzwerke für den Nahbereich. Eingebettete Funksysteme: Vergleich von standardisierten und proprietären Verfahren*. 2., aktualisierte und erw. Aufl. Springer Vieweg, Wiesbaden (2015).

[29] Glück, M. *MEMS in der Mikrosystemtechnik. Aufbau, Wirkprinzipien, Herstellung und Praxiseinsatz mikroelektromechanischer Schaltungen und Sensorsysteme*. Vieweg+Teubner Verlag, Wiesbaden (2005).

[30] Gramlich, G. *Eine Einführung in MATLAB. aus Sicht eines Mathematikers*. http://www.hs-ulm.de//users/gramlich/EinfMATLAB.pdf (2012). Zuletzt geprüft am 12.03.2019.

[31] HiveMQ. *MQTT Essentials Part 2: Publish & Subscribe*. https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe. Zuletzt geprüft am 13.05.2018.

[32] HiveMQ. *MQTT Essentials Part 4: MQTT Publish, Subscribe & Unsubscribe*. https://www.hivemq.com/blog/mqtt-essentials-part-4-mqtt-publish-subscribe-unsubscribe. Zuletzt geprüft am 22.05.2018.

[33] HiveMQ. *MQTT Essentials Part 6: Quality of Service 0, 1 & 2*. https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels. Zuletzt geprüft am 22.05.2018.

[34] Hofbauer, J., Wolf, M., und Rudolph, M. (2017). Drahtlose, energieautarke Sensorik zur Zustandsüberwachung von Maschinen und Anlagen - Simulationsuntersuchung zur Abschätzung der energetischen Lebensdauer für den praktischen Einsatz. In *Arbeitswelten 4.0. Chancen, Herausforderungen, Lösungen*, 73–82.

[35] Hosterz. *Webhosting-Lexikon. Apache-Webserver*. https://www.hosterz.de/lexikon/apache-webserver/. Zuletzt geprüft am 16.06.2018.

[36] HTWK Leipzig. Fakultät Maschinenbau und Energietechnik. *Forschung - abgeschlossene Projekte. TRAINCON*. https://fme.htwk-leipzig.de/de/fakultaet/personen/professuren/prof-rudolph/forschung/. Zuletzt geprüft am 24.03.2018.

[37] HTWK Leipzig. Fakultät Maschinenbau und Energietechnik. *Forschung - aktuell laufende Projekte. SmartTram*. https://fme.htwk-leipzig.de/de/fakultaet/personen/professuren/prof-rudolph/forschung/. Zuletzt geprüft am 24.03.2018.

[38] ICP DAS. *What is ZigBee?* http://www.icpdas.com/root/product/solutions/industrial_wireless_communication/wireless_solutions/zigbee_introduction.html. Zuletzt geprüft am 29.04.2018.

[39] Jaffey, T. *CoAP Module*. https://nodemcu.readthedocs.io/en/master/en/modules/coap/. Zuletzt geprüft am 13.05.2018.

[40] Kanoun, O. und Wallaschek, J. *Energy Harvesting. Grundlagen und Praxis energieautarker Systeme ; mit 17 Tabellen ; [Fachtagung Energy Harvesting]*. Haus der Technik Fachbuch 92. expert-Verl., Renningen (2008).

[41] Latuske, R. *Bluetooth Low Energy in Smartphones - wie funktioniert das? Seite 1*. http://www.elektroniknet.de/elektronik/kommunikation/bluetooth-low-energy-in-smartphones-wie-funktioniert-das-103061.html (2013). Zuletzt geprüft am 1.05.2018.

[42] Latuske, R. *Bluetooth Low Energy in Smartphones - wie funktioniert das? Seite 2*. http://www.elektroniknet.de/elektronik/kommunikation/bluetooth-low-energy-in-smartphones-wie-funktioniert-das-103061-Seite-2.html (2013). Zuletzt geprüft am 1.05.2018.

[43] Lieder, A. *ZigBee - drahtlose Kommunikation. Projektdokumentation Rechnernetze*. http://www.inw.hs-merseburg.de/~uheuert/pdf/Anwendung%20Rechnernetze/Vortraege/SS2007/Annekathrin%20Lieder%20-%20Zigbee/Dokumentation.pdf (2007). Zuletzt geprüft am 29.04.2018.

[44] Maanas Roy. Technology, Trends and Talk. *Apache Performance Tuning: KeepAlive to remove latency*. https://maanasroyy.wordpress.com/2012/05/05/apache-performance-tuning-keepalive-to-remove-latency/ (2012). Zuletzt geprüft am 30.05.2018.

[45] Matyas, K. *Taschenbuch Instandhaltungslogistik. Qualität und Produktivität steigern*. Praxisreihe Qualitätswissen. 4., überarb. Aufl. Hanser, München, Wien (2010).

[46] Mitasch, C. *Apache Performance Tuning*. https://www.thomas-krenn.com/de/wiki/Apache_Performance_Tuning. Zuletzt geprüft am 16.06.2018.

[47] Naik, N. (2017). Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP. In 2017 *IEEE International Symposium on Systems Engineering. ISSE 2017 : Vienna, Austria, October 11-13, 2017 : 2017 symposium proceedings*. IEEE, Piscataway, NJ.

[48] National Semiconductor Corporation. *LM158/LM258/LM358/LM2904. Low Power Dual Operational Amplifiers*. https://www.ece.ucsb.edu/Faculty/rodwell/Classes/ece2c/labs/lm158.pdf (2000). Zuletzt geprüft am 14.05.2018.

[49] Netcraft Ltd. *February 2017 Web Server Survey*. https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html (2017). Zuletzt geprüft am 10.03.2019.

[50] Nicholas, S. *Power Profiling: HTTPS Long Polling vs. MQTT with SSL, on Android*. http://stephendnicholas.com/posts/power-profiling-mqtt-vs-https (2012). Zuletzt geprüft am 23.05.2018.

[51] Nilgen, S. und Hannifin, P. Alles im Blick. Vom Condition Monitoring zum Total System Health Management. *Fluid* (2015), 11-12, 58–61.

[52] *NodeMCU Documentation*. https://nodemcu.readthedocs.io/en/master/. Zuletzt geprüft am 6.05.2018.

[53] Obermaier, D. *IoT-Protokolldschungel – Ein Wegweiser*. https://www.informatik-aktuell.de/betrieb/netzwerke/iot-protokolldschungel-ein-wegweiser.html (2015). Zuletzt geprüft am 15.04.2018.

[54] Pomaska, G. *Webseiten-Programmierung. Sprachen, Werkzeuge, Entwicklung*. Springer Vieweg, Wiesbaden (2012).

[55] Prus, F. *IEEE 802.11a/b/g/n/ac – alle Infos zu den WLAN-Standards* (2018). Zuletzt geprüft am 12.03.2019.

[56] Raschbichler, F. *MQTT - Leitfaden zum Protokoll für das Internet der Dinge*. https://www.informatik-aktuell.de/betrieb/netzwerke/mqtt-leitfaden-zum-protokoll-fuer-das-internet-der-dinge.html (2017). Zuletzt geprüft am 13.05.2018.

[57] Ray, B. *Z-Wave Vs. Zigbee*. https://www.link-labs.com/blog/z-wave-vs-zigbee (2017). Zuletzt geprüft am 29.04.2018.

[58] Rudolph, M. *Drahtlose energieautarke Messtechnik zur Maschinendiagnose. intec - Internationale Fachmesse für Werkzeugmaschinen, Fertigungs- und Automatisierungstechnik vom 7. bis 10. März 2017*. Präsentation.

[59] Ryte Wiki. *Keep-Alive*. https://de.ryte.com/wiki/Keep-Alive (2016). Zuletzt geprüft am 30.05.2018.

[60] Schenk, M. *Instandhaltung technischer Systeme. Methoden und Werkzeuge zur Gewährleistung eines sicheren und wirtschaftlichen Anlagenbetriebs*. SpringerLink : Bücher. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).

[61] Schulze, B. *WHITEPAPER - ENERGY HARVESTING. Energy Harvesting nutzt den Piezoeffekt*. https://www.piceramic.de/fileadmin/user_upload/pi_ceramic/files/success_story/WP_pi1068_EnergyHarvesting.pdf. Zuletzt geprüft am 8.04.2018.

[62] Semiconductor Components Industries. *NCP1117, NCV1117. 1.0 A Low-Dropout Positive Low-Dropout Positive Fixed and Adjustable Voltage Regulators*. https://www.onsemi.com/pub/Collateral/NCP1117-D.PDF (2017). Zuletzt geprüft am 21.06.2018.

[63] Stal, M. *RESTful mit CoAP*. https://www.heise.de/developer/artikel/RESTful-mit-CoAP-3251225.html (2016). Zuletzt geprüft am 11.05.2018.

[64] statista. *Anteil der Photovoltaik an der Bruttostromerzeugung in Deutschland in den Jahren 2002 bis 2017*. https://de.statista.com/statistik/daten/studie/250915/umfrage/anteil-der-photovoltaik-an-der-stromerzeugung-in-deutschland/. Zuletzt geprüft am 14.06.2018.

[65] Texas Instruments. *Wireless Connectivity. SimpleLink Bluetooth low energy wireless MCUs*. http://www.ti.com/wireless-connectivity/simplelink-solutions/bluetooth-low-energy/overview/overview.html. Zuletzt geprüft am 31.05.2018.

[66] Texas Instruments. *Wireless Connectivity. SimpleLink Zigbee wireless MCUs. Online datasheet*. http://www.ti.com/wireless-connectivity/simplelink-solutions/zigbee/overview.html. Zuletzt geprüft am 31.05.2018.

[67] Tränkler, H.-R. und Reindl, L. M. *Sensortechnik. Handbuch für Praxis und Wissenschaft*. VDI-Buch. 2., völlig neu bearb. Aufl. Springer Vieweg, Berlin (2014).

[68] Wabner, M., Riedel, M., und Ihlenfeldt, S. Vernetzung von Industriemaschinen. Qualität bei der vorausschauenden Instandhaltung. *wiso - IM+io Das Magazin für Innovation, Organisation & Management* (2015), 1, 53–58.

[69] Walter, K.-D. IoT-Schnittstellen einfach verdrahten. Die Open-Source-Software Node-RED bietet einen geeigneten Baukasten, um per Datenflussprogrammierung die notwendigen Verbindungen zwischen IoT-Systemen zu schaffen. *Elektronikpraxis* 2017, 4, 36–42.

[70] Watzka, B., Scheler, S., and Wilhelm, T. *Beschleunigunssensoren*. http://www.thomas-wilhelm.net/veroeffentlichung/Beschleunigungssensoren_PdN.pdf. Zuletzt geprüft am 8.04.2018.

[71] Weißenbach, A. *Professionelles Instandhaltungsmanagement. Strategie – Organisation – Kooperation; mit Online-Analysetool Quick-Maintenance-Check*. ESV. Erich Schmidt Verlag, Berlin (2017).

[72] Wendel, M. *Was ist ZigBee und welche Geräte sind ZigBee kompatibel? Funktionen und Einsatzmöglichkeiten von ZigBee*. https://www.homeandsmart.de/zigbee-funkprotokoll-hausautomation. Zuletzt geprüft am 17.04.2018.

[73] Wenn, D. *Genau überwachen mit Shunts. Stromerfassung per Strommesswiderstand*. https://www.elektroniknet.de/design-elektronik/messen-testen/genau-ueberwachen-mit-shunts-84373.html (2011). Zuletzt geprüft am 4.06.2018.

[74] Wiegert, B. *Symptome für einen Lagerschaden frühzeitig erkennen. Wälzlager*. https://www.konstruktionspraxis.vogel.de/symptome-fuer-einen-lagerschaden-fruehzeitig-erkennen-a-114207/ (2007). Zuletzt geprüft am 29.05.2018.

[75] Wikipedia. *ESP8266 auf einem Development Board*. https://de.wikipedia.org/wiki/ESP8266. Zuletzt geprüft am 13.04.2018.

[76] Wojcieszak, M. *Internet-Protokolle, Teil 1: TCP/IP, der Grundstein für Anwendungsprotokolle*. https://www.heise.de/developer/artikel/Internet-Protokolle-Teil-1-TCP-IP-der-Grundstein-fuer-Anwendungsprotokolle-2548919.html?seite=all (2015). Zuletzt geprüft am 15.04.2018.

[77] Wojcieszak, M. *Internet-Protokolle, Teil 2: Anwendungsprotokolle im Vergleich*. https://www.heise.de/developer/artikel/Internet-Protokolle-Teil-2-Anwendungsprotokolle-im-Vergleich-2632571.html?seite=all (2015). Zuletzt geprüft am 14.04.2018.

[78] Wolf, M., Rudolph, M., und Köllner, J. Schwingungsdiagnostische Untersuchung von Straßenbahn- Antriebskomponenten. Mithilfe von Sensornetzwerken. *ZWF Instandhaltung* 2017, 1 - 2, 62–67.

Anhang

http://doi.org/10.33968/9783966270069.0008

Anhang 1 LUA-Code Initialisierungsskript

```
trial = 0
station_cfg={}
station_cfg.ssid="MQTT-Pi"          -- Name WLAN
station_cfg.pwd="redraspberry"      -- Passwort WLAN
wifi.sta.config(station_cfg)
wifi.setmode(wifi.STATION)
wifi.sta.connect()

    -- jede Sekunde (= 1000 ms) erneuter Versuch
    tmr.alarm(1, 1000, 1, function()

        -- wenn keine IP-Adresse vorhanden und Versuch <= 10
        if wifi.sta.getip() == nil and trial <= 10 then
        trial = trial + 1
        print ("IP unavailable, Waiting..."..trial)

            -- wenn 11. Versuch (trial = 11) --> Sleep-Modus für 60s
            if trial > 10 then
            node.dsleep(60 * 1000000)
            end

        -- wenn IP-Adresse vorhanden (also Verbindung zum WLAN aufgebaut)
        else
        tmr.stop(1)

        print("\n==========================")
        print("ESP8266 mode is: " .. wifi.getmode())
        print("MAC adress is: " .. wifi.ap.getmac())
        print("IP is " .. wifi.sta.getip())
        print("==========================")

        -- Messskript aufrufen
        dofile("180502.lua")
        end
    end)
```

Anhang 2 LUA-Code Messskript

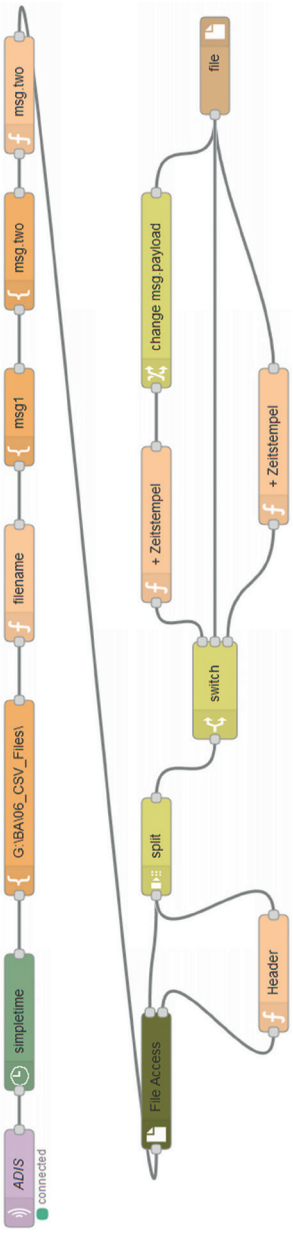
```lua
pin = 2 -- Pin für Chip-Select
rdy = 1 -- Pin für wait-function
ANZ = 128 -- Anzahl Werte je Datenpaket
min = 10 -- Dauer Sleep-Modus in Sekunden


-- ### SPI-Setup ###
spi.setup(1, spi.MASTER, spi.CPOL_HIGH, spi.CPHA_HIGH, 16, 64, spi.FULLDUPLEX);
gpio.mode(pin, gpio.OUTPUT)
gpio.mode(rdy, gpio.INPUT)
gpio.write(pin, gpio.LOW)



-- Funktion zum paketweisen Auslesen der Messwerte. 128 Werte werden ausgelesen und in eine Tabelle
-- geschrieben.
-- Anschließend werden diese Werte von Kommas getrennt zu einem String zusammengefügt.

function read_send_val(PIN, values, address, band)

local array = {}

    -- Auslesen Fehlerausgabe (wenn "Data ready, capture complete" zurückgegeben wird dann weiter,
    -- sonst wird Fehlercode versendet --> s. else)
    if bit.band(read_reg(PIN, 0x3C00), 0x0080) == 0x0080 and read_reg(PIN, 0x3C00) ~= 0xFFFF then

    a = ""

    for i = 1, values do                        -- values --> ANZ --> Anzahl Werte je Datenpaket
        local byte = read_reg(PIN, address)     -- ruft Funktion read_reg zum Auslesen des Messwertes
                                                -- auf --> byte = Messwert
        array[i] = string.format("%x", byte)    -- schreibt ausgelesenen Wert in Tabelle und wandelt
    end                                         -- Float in Hex-Zahl.

        local a = table.concat(array, ";")      -- fügt Werte zu String zusammen, Komma ist Trennzeichen
        send_string(a, values, filter[1])       -- ruft Funktion zum Versenden auf

        a = ""

    else
    --Ausgabe des Fehlercodes als Bitmuster
    send_string(bits(read_reg(PIN, 0x3C00)), values, filter[1])
end

-- veröffentlicht den zusammengesetzten String unter dem Topic "ADIS"
function send_string(value, index, band)

    -- fügt dem String Sensorposotion und Filtereinstellung hinzu
   local data = value.."&Sensor_Achse_POS_3".."&"..band

   -- veröffentlichen einer MQTT-Nachricht mit einem Datenpaket unter Topic "ADIS"
   m:publish("ADIS", data, 0, 0,

                -- wenn Nachricht erfolgreich veröffentlicht --> Aufrufen der Funktion "wert_auslesen"
                 function(client)
                 wert_auslesen() end,

                 function(client, reason) print("reason: "..reason) end)

    data = nil

end
```

```
64
65  -- damit Fehlerausgabe in read_send_val als Bitmuster
66  function bits(num)
67      local t={}
68      while num>0 do
69          rest=num%2
70          table.insert(t,1,rest)
71          num=(num-rest)/2
72      end return table.concat(t)
73  end
74
75  -- setzt CS-Kanal auf definiertes High-Signal
76  -- --> Beginn einer SPI-Datenübertragung mit Setzen auf LOW, dafür muss Signal vorher HIGH sein
77  function SET_CS(PIN)
78  gpio.write(PIN, gpio.LOW)
79  tmr.delay(15)
80  gpio.write(PIN, gpio.HIGH)
81  tmr.delay(15)
82  end
83
84  -- liest Messwert aus Zwischenspeicher des Sensors aus
85  function read_reg(PIN, address)
86      SET_CS(PIN)
87      spi.send(1, address);
88      tmr.delay(15);                      --15 us Zeit zwischen 2 Bytes
89      wrote, hi_byte = spi.send(1, 0x0000) --Dummy-Bytes
90      SET_CS(PIN)
91      data = hi_byte                      -- zeitgleich kommen Daten vom Sensor zurück, high_Byte mit Messwert
92      tmr.delay(15);
93      return data;
94  end
95
96  -- schreibt Adresse im Sensor an und kann Einstellungen setzen
97  function write_reg(PIN, address)
98      SET_CS(PIN)
99      spi.send(1, address)
100     SET_CS(PIN)
101     tmr.delay(15)
102 end
103
104 -- wird nach Auslösen der Messung aufgerufen, unterbricht ablauf des Skripts, solange rdy-Pin high ist
105 function wait()
106   while gpio.read(rdy) == 1 do -- solange rdy-Pin high ist
107     gpio.read(rdy)
108     tmr.delay(10)
109     end
110 end
111
112 -- definiert Variablen und übergibt Einstellungen an den Sensor, löst Messung und Auslesen der Messwerte aus
113 function initialize()
114       filter = {0xB804}                   -- filter 4,56Khz
115       axis =   {0x1400, 0x1600, 0x1800}   -- x, y, z -Achse
116
117       j = 1 -- Messung/ Stringnummer
118       l = 1 -- Achse
119
120       write_reg(pin, filter[1])
121       write_reg(pin, 0x9C0C)                    -- extended-Mode
122       write_reg(pin, 0x9D02)                    -- z-Achse
123       write_reg(pin, 0xBF08)                    -- löst Messung aus
124       wait()                                    -- unterbricht Skript, solange Sensor misst
125       read_send_val(pin, ANZ, axis[l], filter[1]) -- ruft Funktion "read_send_val" zum Auslesen der Messwerte auf
126 end
127
```

```
-- wird aufgerufen, nachdem eine Nachricht erfolgreich veröffentlich wurde
-- sorgt dafür, dass nächstes Datenpaket ausgelesen wird
function wert_auslesen()

        -- Index für Messungsnummer (je Achse/Capture-Buffer)
        j = j + 1

        if j <= 1024 / ANZ then

            -- ruft Funktion "read_send_val" zum Auslesen der Messwerte auf
            read_send_val(pin, ANZ, axis[l], filter[1])

        else
            -- Index für Achse (bzw. Capture-Buffer der x/y/z-Achse)
            l = l + 1

            -- Abbruchbedingung der Schleife, wenn l =3 sind alle 3072 Werte ausgelesen
            if l > 3 then

                l = 1

                -- versetzt für min * 1s Modul in Deep-Sleep-Modus
                write_reg(pin, 0x9C02)
                write_reg(pin, 0xBF08)
                node.dsleep(min * 1000000)
            else

            j = 1
            -- ruft Funktion "read_send_val" zum Auslesen der Messwerte auf
            read_send_val(pin, ANZ, axis[l], filter[1])

            end

      end

end

```

```
--  #### MQTT ####

-- erstellt MQTT-Client
m = mqtt.Client("10.10.0.37")

-- wenn Verbindung zum Broker abreisst, dann Initialisierungsskript ausfrufen
m:on("offline", function(client)
        dofile("init.lua")
    end)

-- wenn Verbindung zum Broker hergestellt, dann Funktion "initialize" aufrufen
 m:on("connect", function(client)

    initialize()

 end)

-- erstellt Verbindung mit Broker
m:connect("10.10.0.1", 1883, 0)
```

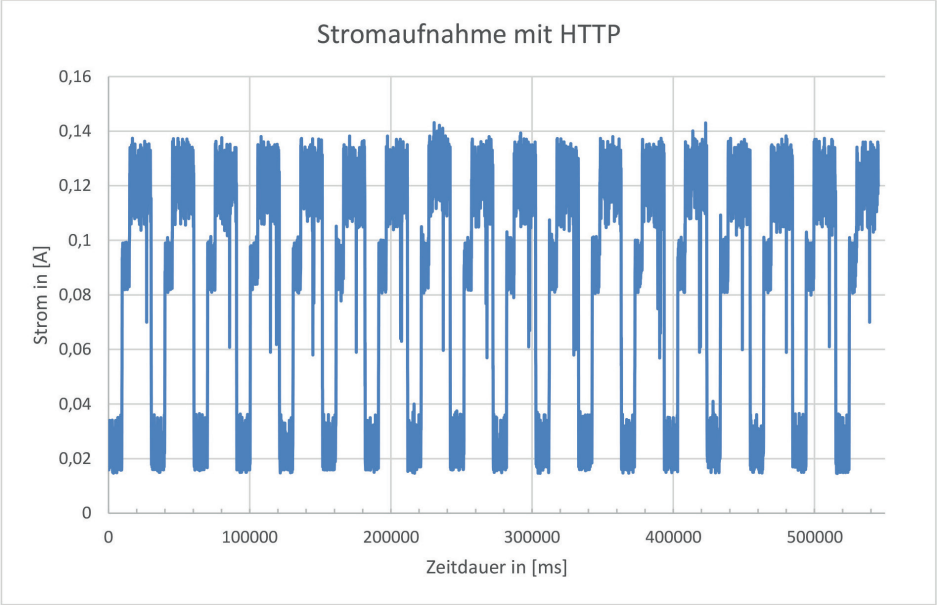
Anhang 3 Node-Red Programmfluss zum Loggen der Messdaten in eine CSV-Datei



Anhang 4 Messskript Arduino

```
int ADC1=1;
int value;
float strom=0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Zeit,Strom");
}

void loop() {
  // put your main code here, to run repeatedly:
  value = analogRead(ADC1);
  strom = (5.0 / 1024.0) * value / 11.0499 / 1.2;
  Serial.print(millis());
  Serial.print(",");
  Serial.println(strom,3);
  delay(100);
}
```

Anhang 5 grafische Darstellung der HTTP-Messwerte



Anhang 6 Mittelwerte der Betriebsmodi der einzelnen Messzyklen mit MQTT

Messung	Sleep		Grundzustand		Senden	
	Zeit in [s]	Strom in [mA]	Zeit in [s]	Strom in [mA]	Zeit in [s]	Strom in [mA]
1	9.37	19.02	6.45	76.83	16.83	102.37
2	9.68	19.78	6.25	76.08	16.93	103.35
3	9.68	19.90	6.25	76.97	17.14	102.65
4	9.38	18.32	6.35	76.56	17.24	102.58
5	9.58	19.01	6.75	76.00	17.24	103.45
6	9.38	18.65	6.55	77.53	16.94	102.91
7	9.48	18.55	6.45	76.40	17.04	103.04
8	9.48	18.98	6.35	77.42	17.14	102.24
9	9.38	18.67	6.45	77.35	17.04	104.01
10	9.48	19.63	6.25	76.61	17.14	104.05
11	9.38	18.38	6.45	76.25	17.04	102.04
12	9.48	19.07	6.45	77.25	17.24	102.76
13	9.48	18.34	6.45	75.45	17.04	102.89
14	9.48	18.67	6.25	76.52	17.24	102.96
15	9.28	18.37	6.45	75.37	16.94	103.72
Mittelwert	9.46	18.89	6.41	76.57	17.08	103.00

Anhang 7 Mittelwerte der Betriebsmodi der einzelnen Messzyklen mit HTTP

Messung	Sleep		Grundzustand		Senden	
	Zeit in [s]	Strom in [mA]	Zeit in [s]	Strom in [mA]	Zeit in [s]	Strom in [mA]
1	9.57	22.17	5.24	91.66	15.22	120.23
2	9.57	22.05	5.24	91.21	15.12	124.01
3	9.68	22.02	5.34	92.09	15.02	121.45
4	9.48	22.05	5.34	91.19	15.43	121.03
5	9.48	21.79	5.34	89.61	15.43	121.55
6	9.58	22.76	5.24	91.02	15.22	119.45
7	9.48	20.67	5.24	90.24	15.22	120.19
8	9.58	22.50	6.65	98.92	13.71	126.02
9	9.68	23.38	5.24	89.94	15.23	121.15
10	9.48	22.35	5.34	90.72	15.23	121.44
11	9.68	23.34	5.24	91.08	15.33	120.88
12	9.48	21.92	5.34	88.63	15.23	122.41
13	9.48	22.11	5.24	90.49	15.43	121.01
14	9.58	22.81	5.45	91.15	15.13	122.90
15	9.58	24.54	5.34	89.69	15.63	121.29
16	9.48	21.39	5.45	90.45	15.12	121.47
17	9.48	21.77	5.45	89.95	15.43	120.89
18	9.48	21.48	5.24	92.44	13.41	120.91
Mittelwert	9.55	22.32	5.32	90.57	15.27	121.33

Anmerkung: Die grau markierten Messungen 8 und 18 wurden nicht in die Mittelwertbildung miteinbezogen, da die Zeitdauern des Betriebszustands Senden unplausibel sind. Vermutlich enthielten Teile der übermittelten Daten Fehlercodes anstatt Messwerte, welche sich vom Transceivermodul schneller auslesen lassen.