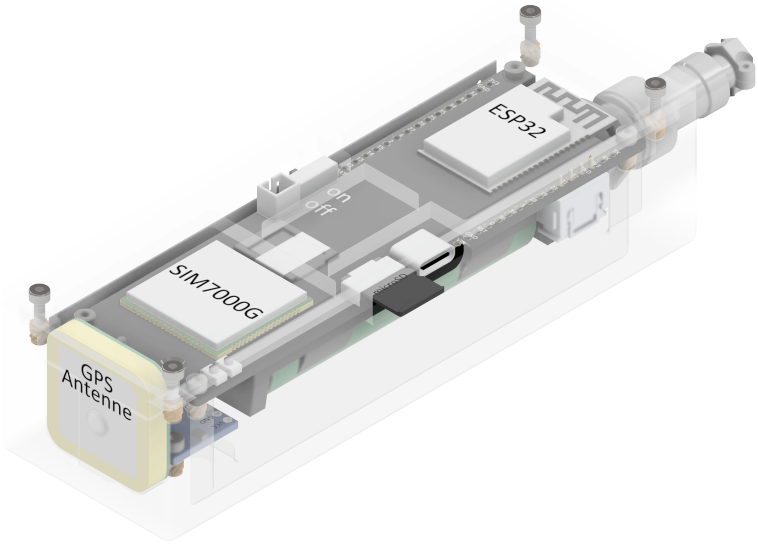
Masterarbeit

Konzeption einer mobilen Plattform zur Erfassung von Sensordaten in der Biomedizintechnik



B.Sc. Lukas Tilch

Matrikel-Nr.: 2922150

Wissenschaftliche Betreuung:

Dr.-Ing. Tim Rittinghaus

Dr.-Ing. Holger Zernetsch

Institut für Mehrphasenprozesse, Gottfried Willhelm Leibniz Universität Hannover

Hannover, 21.06.2022

Hiermit erkläre ich, Lukas Tilch, dass diese Arbeit selbstständig und nur mit den angegebenen Hilfsmitteln und Quellen angefertigt wurde. Ich habe diese Arbeit weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ort, Datum *Unterschrift*

Inhaltsverzeichnis

Abbildungsverzeichnis **IV**

Tabellenverzeichnis **VI**

Einheiten **VII**

Abkürzungen **VIII**

1. Einleitung **1**

2. Grundlagen **3**

2.1. Datenbanken 3

2.2. Netzwerk 5

2.3. Weitere Informationstechnik 8

2.4. Mikrocontroller und Sensoren 14

3. Stand der Technik **15**

3.1. Funktionsweisen der Messsysteme 15

3.2. Messgröße der Messsysteme 17

3.3. Kommerzielle Lösungen 18

3.4. Fazit 18

4. Anforderungen **20**

4.1. Anwendungsfälle am Institut 20

4.2. Use Case Diagramme 21

4.3. Anforderungsliste 24

5. Material und Methoden **25**

5.1. Konzept 25

5.1.1. Auswahl der Technologien 25

5.1.2. Hardware 29

5.1.3. Konstruktion der Gehäuse 33

5.2. Validierung 35

5.2.1. Energieverbrauchsmessung 35

5.2.2. Beschleunigungsmessung 37

5.2.3. Validierung der Transportüberwachung 40

6. Ergebnisse **42**

6.1. Energieverbrauchsmessung 42

6.2. Beschleunigungsmessung 44

6.3. Transportüberwachung 46

7. Diskussion **49**

7.1. Beschleunigungsmessung 49

7.2. Transportüberwachung 51

8. Fazit **55**

9. Ausblick **57**

9.1. Mögliche Anwendungsfälle 57

9.2. Weiterentwicklung 59

Literatur **61**

A. Anhang **66**

Abbildungsverzeichnis

2.1. Dämpfung von Funksignalen 7
2.2. Git Versionsverwaltung 9
2.3. Virtualisierung: Hypervisor - Container 13

3.1. Analoger Schockindikator 16

4.1. Use Case Diagramm Messsystem 22
4.2. Use Case Diagramm Server 23

5.1. Konzeptdarstellung 26
5.2. Pinbelegungsdiagramm 30
5.3. Zusätzliche Schnistellendefinition: Location 31
5.4. Schnittstellendefinition des Messdatenpaketes 32
5.5. Zusätzliche Schnittstellendefinition: Vector 34
5.6. Gehäuse des Messgerätes 34
5.7. Gerätekonfiguration für die Energieverbrauchsmessung 36
5.8. Versuchsaufbau der Beschleunigungsmessung 38
5.9. Gerätekonfiguration für den Beschleunigungstest 39
5.10. Gerätekonfiguration für die Validierung der Transportüberwachung . . . 41

6.1. Stromverbrauchsmessung 43
6.2. Graphen der Beschleunigungsmessung 45
6.3. Kartendarstellung aus der Validierung der Transportüberwachung 47
6.4. Graphendarstellung Vollversuchsmesswerte 48

7.1. *Kibana* Dashboard mit Daten des Messsystems 52

9.1. Ausblick: Use Case Diagramm Containerüberwachung 58
9.2. Ausblick: Use Case Diagramm Stoßerkennung 60

A.1. Programmablaufplan Setup 69
A.2. Programmablaufplan Sensoren initialisieren 70
A.3. Programmablaufplan Senden? 71
A.4. Programmablaufplan Loop 72
A.5. Programmablaufplan GPS-Signal abfragen 73
A.6. Programmablaufplan Modem Verbinden 73
A.7. Programmablaufplan Programmablaufplan Daten loggen 74
A.8. Programmablaufplan Daten Übertragen 75

Tabellenverzeichnis

3.1. Auswahl kommerzieller Messsysteme mit Sensorik und Messbereichen . . 19

4.1. Anforderungsliste 24

Einheiten

Symbol	physikalische Größe	Einheit
I	Strom	A
b	Bit	b
B	Byte	B
f	Frequenz	Hz
g	Erdbeschleunigung	g
L_P	Leistungspegel	db
L_P	Leistungspegel (Bezug 1 mW)	dbm
P	Elektrische Leistung	W
p	Luftdruck	Pa
Q	Elektrische Ladung	Ah
φ	Relative Luftfeuchte	%
ϑ	Celsius-Temperatur	°C
U	Spannung	V

Abkürzungen

Bezeichnung	Beschreibung
ABS	Acrylnitril-Butadien-Styrol-Copolymer
API	Application Programming Interface
EC-GSM-IoT	Extended Coverage GSM IoT
EDGE	Enhanced Data Rates for GSM Evolution
FDM	Fused Deposition Modeling
GNSS	Globales Navigationssatellitensystem
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IoT	Internet of Things
JSON	Javascript Object Notation
LAN	Local Area Network
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LTE	Long Term Evolution
LTE-M	Long Term Evolution for Machines
NB-IoT	NarrowBand IoT
PoE	Power over Ethernet
REST	Representational State Transfer
SBC	Single Board Computer
SoC	System on a Chip
TLS	Transport Layer Security

Bezeichnung	**Beschreibung**
USB-C	Universal Serial Bus Type C
WLAN	Wireless Local Area Network

1. Einleitung

”Hapag-Lloyd stattet als erste Reederei alle Container mit Echtzeit-Tracking aus” [1]. Dieses Zitat einer Pressemeldung aus dem April 2022 zeigt einen Trend an: Die Überwachung von Transportgütern gewinnt zunehmend an Relevanz für die Optimierung von Transport und Logistik. Neuerungen in den Bereichen Internet-Of-Things, Mobilfunknetzwerke, Mikrocontroller und Sensorik haben dazu geführt, dass die Implementierung einer Echtzeitüberwachung von Transportgütern und deren Positionsbestimmung kostengünstiger und leistungsfähiger sind denn je. Quelloffnene, kompetitive Systeme sind jedoch von keiner Relevanz. Am Markt befindliche Systeme setzen auf proprietäre Hardware und auf cloudbasierte, daher undurchsichtige Datenhaltung.

Beim Transport von biomedizintechnischen Gütern, wie Gewebe, ist die Einhaltung von Temperatur- und Luftfeuchtegrenzwerten essentiell, da selbst kurzeitige Über- oder Unterschreitungen das Transportgut schädigen können [2]. Einwegindikatoren sind ein verbreitetes Mittel, um die Einhaltung der Grenzwerte zu belegen [3]. Diese liefern jedoch wenig Informationen, wann, wo und warum die Grenzwerte überschritten wurden und können erst am Ende des Transportes ausgewertet werden. Ein Echtzeit-Messsystem mit Standortermittlung und detaillierten Verläufen für Temperatur, Luftfeuchte und Luftdruck könnte dazu beitragen, Transportwege zu optimieren und ermöglicht, frühzeitig auf Grenzwertüberschreitungen zu reagieren.

Ziel dieser Masterarbeit ist die Konzeptionierung, Entwicklung und Validierung einer quelloffenen und skalierbaren Messplatform zur Überwachung verschiedener Messwerte bei dem Transport von Gütern. Die Konzeptionierung umfasst sowohl die Entwicklung der Messgeräte, als auch die Auswahl und Implementierung der für den Transport der Messdaten benötigten Infrastruktur. Dies beinhaltet die Auswahl eines geeigneten draht-

losen Kommunikationsnetzwerkes für die Übertragung der Daten, sowie das Aufsetzten einer geeigneten Systeminfrastruktur zum Validieren, Konsolidieren und Speichern der Daten. Dabei werden Schnittstellen definiert, welche gewisse Datenstrukturen vorgeben, um die von einander unabhängige Entwicklung der verschiedenen Komponenten zu ermöglichen. Die Fertigung von Prototypen und Validierung ihrer Funktionalität stellt einen weiteren Bestandteil dieser Arbeit dar. Die Aufbereitung und Präsentation dieser Messdaten stellen, neben der intuitiven Benutzbarkeit des Systems, maßgebliche Faktoren für die Akzeptanz der potentiellen Nutzerinnen und Nutzer dar. Eine automatisierte und dynamische Analyse sowie die verständliche Darstellung sollten neben dem möglichen Export bestimmter Datenmengen gegeben sein.

Im folgenden Kapitel werden die Grundlagen der einzelnen Bestandteile eines modernen Messsystems erläutert. Im Kapitel Stand der Technik werden bereits am Markt verfügbare Systeme analysiert und verglichen. Im nächsten Kapitel werden Anwendungsfälle definiert und Anforderungen an das Messsystem abgeleitet. Das Kapitel Material und Methoden befasst sich mit der Konzipierung des Systems und Fertigung der Geräte, sowie der Validierung des Messsystems anhand von Tests welche aus den Anwendungsfällen abgeleitet wurden. Im anschließenden Kapitel werden die Ergebnisse des in dieser Masterarbeit entwickelten Messsystems analysiert und mit den bereits am Markt befindlichen Systemen verglichen. Schlussendlich folgt das Fazit sowie ein Ausblick auf weitere Möglichkeiten des Messsystems.

2. Grundlagen

Moderne Messkonzepte bestehen, neben der eigentlichen Sensorik zur Messwerterfassung, aus einem komplexen informationstechnischen System, welches die Messdaten transportiert, aufbereitet, speichert und bereitstellt. Zu den einzelnen Schritten dieses Systems ist es wichtig, gewisse Grundkonzepte zu kennen und deren Vor- und Nachteile abschätzen zu können. In dem Kapitel Grundlagen werden diese Konzepte angerissen und verschiedene Lösungsansätze vorgestellt.

2.1. Datenbanken

Die Funktionen von Datenbanksystemen beinhalten unter anderem, Daten zu speichern und so zu organisieren, dass sie bedarfsgerecht an die Benutzerin oder den Benutzer oder ein Programm zurückgegeben werden. Es gibt verschiedene Arten von Datenbanken, welche auf gewisse Aspekte der Datenhaltung oder bestimmte Strukturen der Daten spezialisiert sind [4]. Die Auswahl der Datenbank wird maßgeblich durch die Art der Daten beeinflusst, jedoch ist gerade bei kleinen Projekten für die Einhaltung des zeitlichen Rahmens ebenfalls entscheidend, wie gut und mit welchem Aufwand sich die Datenbank mit der restlichen Software verbinden lässt.

Schlüssel-Werte-Datenbank Schlüssel-Werte-Datenbanken, auch *key-value store*, sind eine simple Form von Datenbank, welche nur Schlüssel-Wert-Paare speichern können. Bei Schlüssel-Wert-Paaren wird einem Attributsnamen, dem Schlüssel, ein Attributswert zugewiesen, z.B. Name: "Max". In diesem Fall stellt das Wort "Name" den Schlüssel dar und "Max" den Wert. Mehrere Schlüssel-Wert-Paare müssen nicht unbedingt identische

Strukturen aufweisen und ermöglichen somit auch inkonsistente Datensätze zu speichern, ohne Leer-Felder zu ergänzen. Der daraus resultierende geringe Speicherbedarf ermöglicht es, die komplette Datenbank im volatilen Speicher eines Systems zu halten. Datenbanken wie *Redis* oder *Memcached* werden, trotz ihres limitierten Funktionsumfangs, aufgrund ihrer hohen Geschwindigkeit häufig als Beschleuniger für andere Datenbanken genutzt [5].

Relationale Datenbanken Relationale Datenbanken organisieren Daten in Tabellen mit Zeilen und Spalten. Die Verknüpfung wird mit eindeutigen Schlüsseln realisiert [6]. Dabei müssen die Daten strikt nach einem definierten Schema strukturiert und konsistent sein und somit auch NULL-Werte enthalten [7]. Relationale Datenbanken bieten eine einfache Möglichkeit, Daten miteinander zu verknüpfen, können jedoch bei komplexen Verknüpfungen über mehrere Tabellen beim Durchsuchen langsam werden.

Dokumentendatenbanken Dokumentendatenbanken speichern die Daten im Gegensatz zu Relationalen Datenbanken nicht in Tabellen, sondern in Dokumenten, welche Schlüssel-Wert-Paare enthalten [8]. Dokumente sind semistrukturiert und schemafrei, was das Speichern von komplexen und inkonsistenten Daten vereinfacht. Semistrukturierte Daten beinhalten jene Informationen, welche normalerweise in einem Schema definiert wären, in den Daten selbst. Sie sind "selbst-beschreibend".

Zeitreihendatenbanken Zeitreihendatenbanken sind auf das Speichern und Zurückgeben von Zeit-Wert-Paaren optimiert [9]. Ein Datensatz besteht zumeist aus einem Zeitstempel und einem oder mehreren Messwerten. Datenbanken dieses Typs zeichnen sich zumeist durch geringen Ressourcenbedarf, aber auch geringen Funktionsumfang aus.

Graphdatenbanken Graphdatenbanken eignen sich gut, um stark vernetzte Informationen abzuspeichern [10]. Dabei werden die Informationen in Knoten unterteilt und Relationen zwischen ihnen mittels Kanten dargestellt. Graphen ermöglichen durch Traversierung komplexe Anfragen. Diese können im Gegensatz zu relationalen Datenbanken ohne Rekursion durchgeführt werden.

2.2. Netzwerk

Um die Kommunikation zwischen mehreren Geräten und Sensoren zu ermöglichen, existieren verschiedene Ansätze, diese zu vernetzen.

Ethernet Ethernet ist der Oberbegriff für eine aus Hardware und Software bestehende Spezifikation zur Vernetzung sogenannter Local Area Networks (LAN) [11]. Die gängigsten Formen von Kabeln sind Lichtwellenleiter und Twisted-Pair-Kabel. Diese bestehen aus acht Adern, welche paarweise miteinander verdrillt sind um störende Wechselwirkungen zweier Leitungen aufeinander zu minimieren. Die gängigsten Geschwindigkeiten für Kupferkabel sind 10Base-T (10 Mbit), 100Base-T (100 Mbit) und 1000Base-T (1000 Mbit). Twisted-Pair-Kabel bieten zudem die Möglichkeit, Strom für Endgereäte direkt über das Netzwerkkabel zu leiten. Bei dem sogenannten *Power over Ethernet* (PoE) werden je nach Standard entweder die sonst unbelegten Pins 3, 4, 7 und 8 oder die datenführenden Pins zur Übertragung des Stromes genutzt (Phantomspeisung).

Wireless Local Area Network Bei Wireless Local Area Network (WLAN) handelt es sich um eine Technologie zur kabellosen Datenübertragung basierend auf den IEE 802.11 Normen. Typische Frequenzen für WLAN Signale sind 2,4 GHz (2400–2483,5 MHz) und 5 GHz (5150–5350 MHz und 5470–5725 MHz) [12]. Endgeräte verbinden sich zumeist mit einem Gateway, welches die Pakete weiterleitet. Gateways haben in der Regel die Möglichkeit, mit mehreren Geräten Verbindungen aufzubauen; leistungsfähige Gateways auch parallel.

Long Range Wide Area Network Das *Long Range Wide Area Network* (*LoRaWAN*) ist eine auf Energieeffizienz und Reichweite optimierte, kabellose Systemarchitektur und ein Kommunikationsprotokoll. *LoRa* stellt die Übertragungsschicht des *LoRaWAN* dar. [13] Die Geräte im *LoRaWAN* arbeiten asynchron und kommunizieren nur, wenn Daten gesendet werden sollen. Die für das *LoRaWAN* genutzten Frequenzen liegen generell unter einem Gigaherz, die genauen Frequenzen sind länderabhängig. In Europa liegen die Frequenzen im *Industrial, Scientific and Medical Band* (ISM-Band) zwischen 443,5 bis

434,79 MHz und im SED-Band zwischen 863 bis 870 MHz. Die maximalen Datenraten zwischen *LoRaWAN* Geräten und dem Gateway betragen 50 kbit/s. Die Energieeffizienz von LoRa wird maßgeblich dadurch ermöglicht, dass Geräte meist nur als Sender fungieren. Somit muss die Hardware nur in Intervallen Daten versenden und nicht, wie beim WLAN, dauerhaft mit Strom versorgt werden. Untersuchungen zeigen, dass diese Frequenzen im Verhältnis zu den Frequenzen im WLAN, deutlich schwächer von Freiraumdämpfung und Pfadverlusten durch Medien betroffen sind (siehe Abb. 2.1) und sich somit eher für Kommunikation in Gebäuden und Außerhalb eignen [14–16]. Die eingeschränkten Datenraten grenzen die sinnvollen Anwendungen jedoch ein.

Mobilfunk Der Mobilfunk ermöglicht mobilen Geräten über eine Verbindung zu einem Sendemasten, Daten zu übertragen [17]. Im Mobilfunknetz werden unterschiedliche Standards betrieben. Einer der am weitest verbreiteten Standards ist das *Global System for Mobile Communications* (GSM), welches 1990 eingeführt wurde und heutzutage meist als *Enhanced Data Rates for GSM Evolution* (EDGE) mit gesteigerten Datenraten betrieben wird. Da GSM "volldigital" ist, wird es als die zweite Generation bezeichnet (2G). Ebenfalls weit verbreitet ist *Long Term Evolution* (LTE), welches die vierte Generation der Mobilfunkstandards darstellt. Der LTE Standard umfasst weitere spezialisierte Standards, welche auf erhöhte Geschwindigkeit abzielen oder aber größere Reichweiten und mehr parallele Teilnehmer ermöglichen. Für mobile Sensoren eignen sich besonders die LTE Standards LTE-M und NB-IoT sowie der GSM Standard EC-GSM-IoT, da diese Standards mehr Uploadgeschwindigkeit gegenüber der Downloadgeschwindigkeit priorisieren, und Positionierung ermöglichen.

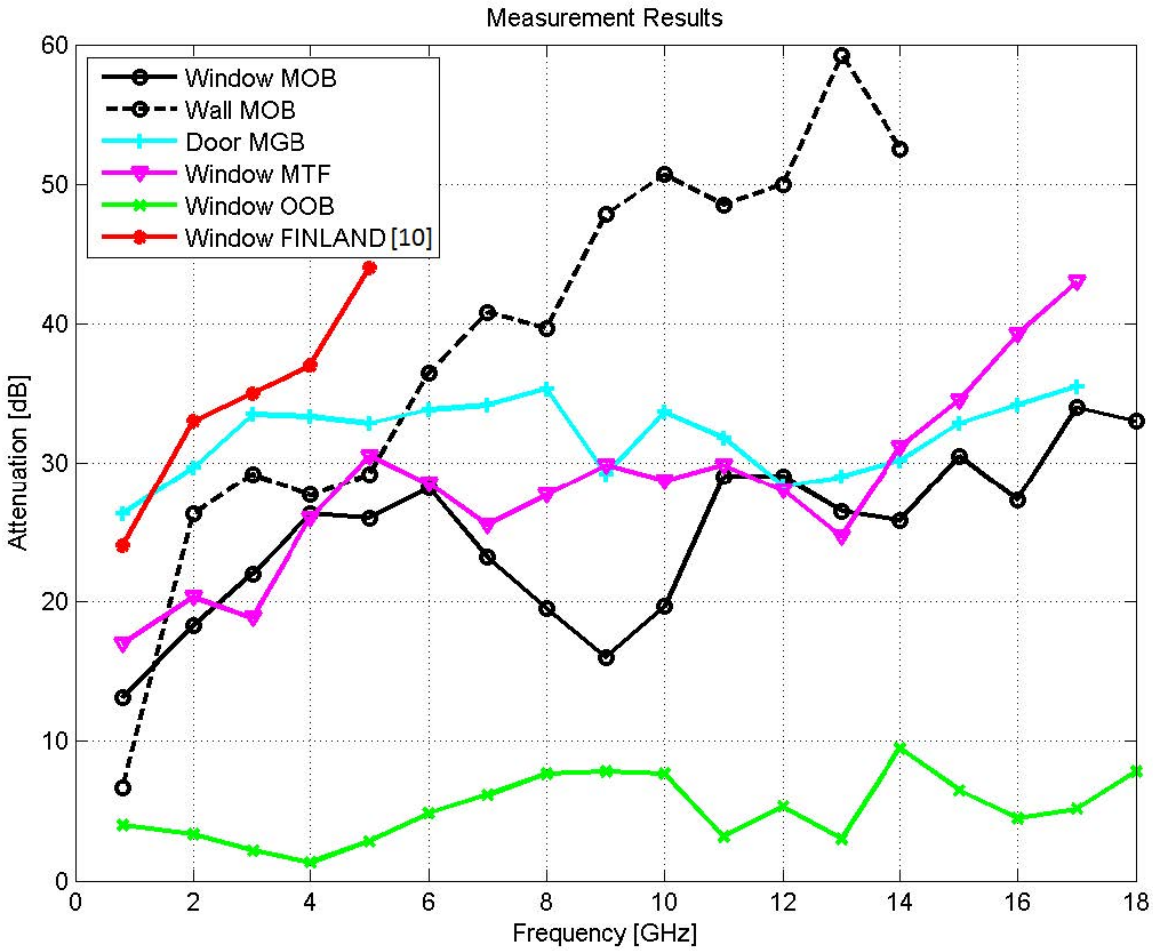


Abbildung 2.1.: Dämpfung von Funksignalen durch verschiedene Material- und Frequenzkombinationen [16]. Auf der Y-Achse ist die Dämpfung in dB aufgetragen, auf der X-Achse die Frequenz der Signale in GHz. Die verschiedenen Graphen zeigen den Durchgang durch Wände, Türen und verschieden klassifizierten Fenstern und Wänden (Modern Office Building, Modern Glass Building, Modern Test Facility, Old Office Building, Vierfachverglasung in Finnland). Die Verläufe der Graphen verdeutlichen, dass die Dämpfung durch Wände, Türen und Fenster in den meisten Fällen mit steigender Frequenz des Signals zunimmt. In der Visualisierung wird ebenfalls abgebildet, dass energieeffizientere Dämmungen negative Auswirkungen auf die Durchlässigkeit von elektromagnetischen Wellen haben.

2.3. Weitere Informationstechnik

Software-Versionsverwaltung Eine Versionsverwaltung in der Softwareentwicklung hilft, Änderungen über mehrere Versionsstände zu implementieren [18]. Dabei überwacht die Versionsverwaltung Änderungen an Dateien und ermöglicht so auch das rückgängig machen von diesen. Die am weitesten verbreitete Versionsverwaltung ist *Git* [19]. "Git ist ein freies und quelloffenes, verteiltes Versionskontrollsystem, das entwickelt wurde, um von kleinen bis zu sehr großen Projekten alles schnell und effizient zu bearbeiten" [20]. Git ermöglicht die Entwicklung in verschiedenen Entwicklungszweigen (*Branches*) (siehe Abb. 2.2). Der Hauptentwicklungszweig (*Master*) stellt den Ausgangspunkt dar, von welchem eine neue Verzweigung erstellt werden kann. Verzweigungen können ebenfalls von anderen *Branches* erfolgen. Änderungen, welche auf einer *Branch* getätigt wurden, können auf andere *Branches* übertragen werden - dies wird als *Merge* bezeichnet. Mit dieser Herangehensweise können Entwicklerinnen und Entwickler eigene *Branches* anlegen und zu einem bestimmten Zeitpunkt in den *Master* mergen. Somit kann gewährleistet werden, dass sich in der *Master branch* nur funktionsfähiger Code befindet, ohne Entwicklerinnen und Entwickler einzuschränken. Verschiedene Versionsstände auf den *Branches* werden als *Commits* bezeichnet. *Commits* können jederzeit manuell erstellt und rückgängig gemacht werden. Dabei können auch *Commits* rückgängig gemacht werden, welche mehrere *Commits* her sind. Ein *Commit* enthält dabei nur die Änderungen an einer Datei. Bei Textdateien, welche einen Großteil bei der Softwareentwicklung ausmachen, werden die hinzugefügten oder entfernten Zeilen in den Dateien gespeichert. Bei binären Dateien wird die ganze Datei entfernt und neu hinzugefügt. Projekte werden in *Git* als *Repositories* bezeichnet. Diese können lokal oder auf einem Server verwaltet werden. Der größte Anbieter für online *Git Repositories* ist *GitHub* [21].

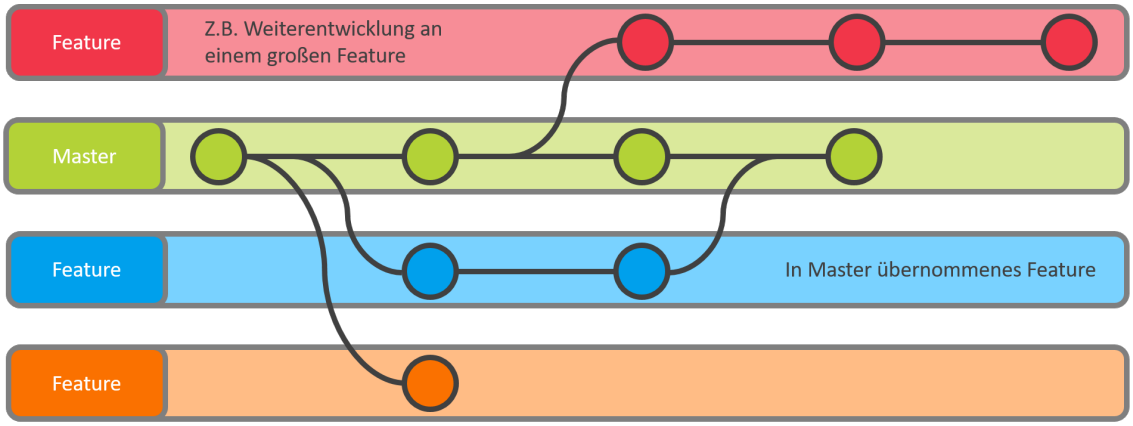


Abbildung 2.2.: Git Versionsverwaltung [22]. Das Bild stellt einen beispielhaften Entwicklungsverlauf für ein Projekt in Git dar. Die *Master Branch* ist der Ausgangspunkt. Einzelne *Commits* werden als Kreise dargestellt. In diesem Fall werden von der *Master Branch* zwei weitere *Branches* abgezweigt, Feature (blau) und Feature (orange) und einen *Commit* später Feature (rot). Die *Commits* in unterschiedlichen *Branches* sind unabhängig voneinander und müssen nicht die gleichen Änderungen enthalten. In dem Beispiel wird die *Branch* Feature (blau) nach zwei *Commits* in die *Master Branch* gemerged und geschlossen.

Globales Navigationssatellitensystem Das *Globale Navigationssatellitensystem* (GNSS) ermöglicht einem Empfangsgerät, Positions- und Uhrzeit-Signale von Satelliten oder Sendeeinheiten, welche vorgeben Satelliten zu sein, sich jedoch auf der Erde befinden - sogenannte Pseudolliten - zu empfangen. Über die vom Empfänger errechnete Laufzeit lässt sich die Entfernung zum Satelliten hinreichend errechnen. Werden die Signale von vier verschiedenen Satelliten empfangen, kann über Trilateration die Position inklusive der Höhe bestimmt werden. Störeinflüsse, welche die Laufzeit der Signale beeinflussen, führen zu Fehlern in der Positionsbestimmung.

Verschlüsselung Sicherheit ist in der Netzwerktechnik von hoher Bedeutung und vor der Umsetzung einer genauen Evaluation zu unterziehen. Generell sollte Dritten kein physischer Zugang zum Netzwerk ermöglicht werden. Kabellose Verbindungen zwischen einzelnen Geräten stellen potentielle Risikofaktoren dar. Durch eine Ende-zu-Ende Verschlüsselung können Daten vor Angreifenden geschützt werden. Verschlüsselung kann in symmetrische und asymmetrische Verfahren unterteilt werden. Bei der symmetrischen Verschlüsselung wird für die Ver- und Entschlüsselung der Daten der gleiche Schlüssel verwendet. Bei der asymmetrischen Verschlüsselung gibt es einen private- und einen public-key. Der public-key kann dabei ausschließlich zur Verschlüsselung verwendet werden, der private-key stellt somit die einzige Möglichkeit zur Entschlüsselung dar. So kann der public-key öffentlich verteilt werden ohne die Sicherheit zu kompromittieren.

Transport Layer Security *Transport Layer Security* (TLS) ist ein Kryptographie-Protokoll, welches zur Absicherung der Datenübertragung im Internet genutzt wird. Es stellt den Nachfolger des veralteten *Secure Sockets Layer* (SSL) dar. *TLS* besteht aus den zwei Komponenten *TLS Handshake* und *TLS Record* [23]. Beim *TLS Handshake* wird zwischen den Kommunikationspartnern ein Sicherheitsschlüssel ausgetauscht und das Gerät authentisiert. Dieser Schlüssel wird von *TLS Record* verwendet, um Daten zu verschlüsseln und so eine sichere Datenübertragung ermöglichen.

Reverse Proxy Ein Reverse Proxy ist eine spezielle Form eines Proxy Servers, welcher externe Anfragen an interne Server verteilt [24]. Dabei ist nur die Adresse des Reverse Proxies von außerhalb erreichbar, um die dahinterliegenden Server, wie z.B. Datenbanken, vor unbefugten Zugriffen zu schützen. Zudem bieten Reverse Proxies die Möglichkeit, die Kommunikation zum Endgerät zu komprimieren und zu verschlüsseln. Zusätzlich ermöglichen sie eine Lastverteilung auf unterschiedliche Server und fungieren durch das Puffern (*Caching*) von statischen und dynamischen Inhalten als sogenannte *web accelerators*.

Application Programming Interface Ein *Application Programming Interface* (API) stellt im Allgemeinen die Schnittstelle einer Software dar, welche anderen Anwendungen oder Nutzern zugänglich gemacht wird. In dieser Schnittstelle sind Routen oder Endpunkte definiert, über welche bestimmte Daten abgefragt oder abgelegt werden können. Eine beliebte Variante für Web APIs basieren auf dem *Representational State Transfer* (REST) Paradigma, welches Prinzipien definiert, um die Kommunikation im Web zu vereinheitlichen [25]. Zu den Prinzipien zählen: Client-Server-Architektur, Zustandslosigkeit, Caching, Einheitlichkeit, Mehrschichtigkeit.

- Client-Server-Architektur: Daten werden auf einem zentralen Server gespeichert und bei Bedarf an die Clients gesendet.
- Zustandslosigkeit: Jede Nachricht zwischen Client und Server enthält alle nötigen Informationen, sodass diese nicht zwischen zwei Nachrichten abgespeichert werden müssen.
- Caching: Soweit möglich soll Caching genutzt werden, um die Skalierbarkeit zu verbessern.
- Mehrschichtigkeit: Ein Client kann nicht feststellen, ob er direkt mit dem Server verbunden ist oder über einen Proxy geleitet wird.
- Einheitlichkeit: Einheitliche Methoden zur Kommunikation aller Komponenten.

Virtualisierung Bei der Virtualisierung werden je nach Ansatz die vorhandenen Hardware-Ressourcen in feste oder dynamische Bereiche unterteilt [26]. Dies ermöglicht das Kapseln einzelner Anwendungen, welche gegebenenfalls andere Versionsanforderungen an die selben Programmbibliotheken haben. Des Weiteren lässt sich durch Virtualisierung die Hardware eines Servers besser ausnutzen, wodurch Vorteile in der Energieeffizienz und beim Platzbedarf entstehen können. Bei der Entwicklung von Software bietet die Virtualisierung eine komfortable Möglichkeit, diese auf verschiedenen Betriebssystemen zu testen.

Hypervisor Ein *Hypervisor* ist eine Software zum Erstellen, Verwalten und Ausführen von virtuellen Maschinen [27]. Diese virtuellen Maschinen bestehen aus der durch den *Hypervisor* verwalteten virtuellen Hardware und einem Betriebssystem. Die Ressourcen, welche dem virtualisierten Betriebssystem zur Verfügung stehen, sind dabei genau definiert. Die Anzahl an verfügbaren Prozessorkernen, Arbeitsspeicher, Netzwerkadaptern und die Größe der virtuellen Festplatte sowie die Möglichkeit des Direktzugriffes auf Hardware-Ressourcen wie z.B. Grafik- oder Netzwerkkarten und Festplattencontroller werden beim Erstellen der virtuellen Maschine zugewiesen und gegebenenfalls für anderen Zugriff blockiert. Dabei lassen sich *Hypervisoren* in Typ 1 und Typ 2 unterteilen. Bei einem Typ 1 *Hypervisor*, auch *Bare-Metal Hypervisor*, handelt es sich um eine Variante, welche ohne Betriebssystem ausgeführt wird. Dadurch kann ausschließlich Hardware verwendet werden, deren Treiber im *Hypervisor* unterstützt wird. Typ 2 *Hypervisoren* werden als Programm auf einem bestehenden Betriebssystem installiert und nutzen dessen Gerätetreiber.

Containervirtualisierung Im Gegensatz zur Hypervisor-Virtualisierung wird bei der Containervirtualisierung keine feste virtuelle Maschine mit einem vollen Betriebssystem erstellt, sondern lediglich ein Container [28]. Dieser stellt in der Regel nur die minimale Grundlage zum Ausführen der zu virtualisierenden Software bereit. Ähnlich wie bei einem Typ 2 *Hypervisor* wird die zur Virtualisierung verwendete Software auf einem bestehenden Betriebssystem installiert (Siehe Abb. 2.3). Es gibt jedoch keine genaue

Zuteilung von Ressourcen zu den Containern; Hardewarelevel-Zugriff ist ebenfalls nicht vorgesehen. Die Anweisungen zum Erstellen des Containers werden in einer Textdatei gespeichert und umfassen eine Basis, auf welcher der Container basiert, z.B. Arch Linux. Weitere Anweisungen können Zusatzpakete wie *Java* oder *Python* installieren. Es können zusätzlich Dateien vom Host-Betriebssystem in den Container kopiert werden. Mit dem Textdokument lässt sich der Container auf jedem kompatiblen Server replizieren. Dabei umfasst die Datei nur wenige Kilobyte, was das Abspeichern und Teilen erleichtert.

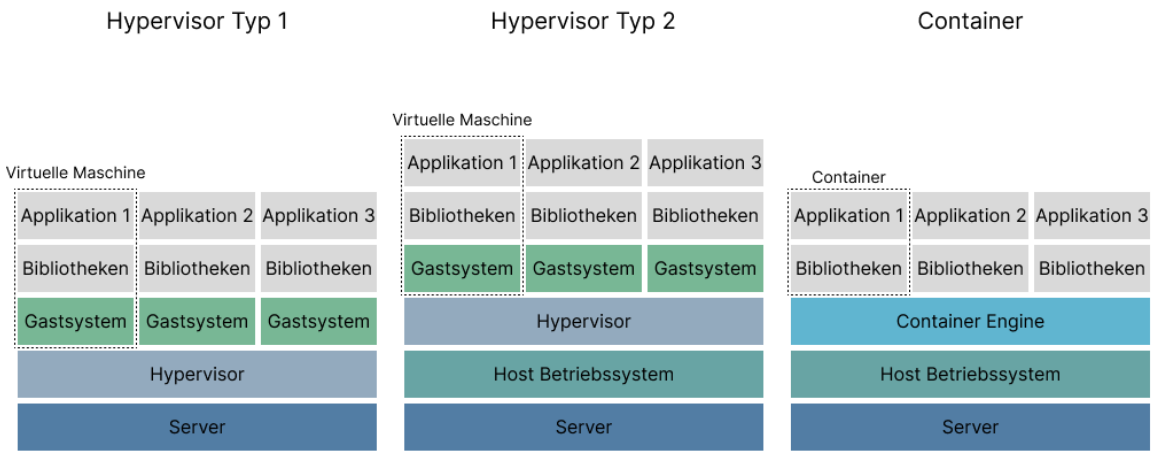


Abbildung 2.3.: Darstellung der Virtualisierungsmethoden *Hypervisor Typ 1*, *Hypervisor Typ 2* und *Container* [29]. Bei dem *Typ 1 Hypervisor* werden Gast-Betriebssysteme von dem *Hypervisor* auf einem Server ausgeführt, ohne das ein vollständiges Host-Betriebssystem installiert sein muss. Der *Typ 2 Hypervisor* hingegen benötigt ein Host-Betriebssystem. Bei der Containervirtualisierung werden auf einem Host-Betriebssystem über die sogenannte *Container Engine* Container ausgeführt, welche kein vollständiges Gast-Betriebssystem benötigen.

2.4. Mikrocontroller und Sensoren

Mikrocontroller kennzeichnen sich durch die Integration von Prozessor und Peripherie in einem einzigen Halbleiterchip. In einigen Fällen werden ebenfalls volatile oder non-volatile Speicher, Grafikprozessoren und auch Wi-Fi in den Chip integriert. Diese sogenannten *SoCs* (*System on a Chip*) zeichnen sich durch ihre hohe Rechenleistung pro Watt und Größe aus. Zumeist bieten Mikrocontroller digitale, je nach Anwendung aber auch analoge Schnittstellen zur Anbindung von Sensoren und weiterer Hardware. Die Kombination von analogen und digitalen Sensoren ermöglicht eine schnelle, skalierbare und kostengünstige Überwachung von Maschinen, Räumen und Experimenten. In Verbindung mit einfacher Netzwerkfunktionalität wie WLAN, Bluetooth oder Ethernet lassen sich Daten an einem zentralen Punkt sammeln, anreichern und überwachen.

Entwicklungsplatinen Entwicklungsplatinen sind mit einem Mikrocontroller bestückte Leiterplatten, welche über Zusatzhardware verfügen, um die Entwicklung eines Prototypen zu vereinfachen und zu beschleunigen. Der starke Anstieg von dem sogenannten *Internet of Things* (IoT) hat preisgünstige, netzwerkfähige Entwicklungsplatinen und passende Entwicklungsumgebungen hervorgebracht [30]. Die Programmierung erfolgt meist mit hardwarenahen Programmiersprachen wie zum Beispiel *C*, *C++* oder *Assembler*.

Single Board Computer Bei einem Einplatinencomputer, auch *Single Board Computer* (*SBC*), befinden sich alle zum Betrieb benötigten Komponenten auf einer einzigen Leiterplatte. Zu den Komponenten zählen zumeist ein Mikroprozessor, volatiler Speicher, in manchen Fällen auch nicht volatiler Speicher und Hardwareschnittstellen. Im Gegensatz zu den Entwicklungsplatinen sind *SBCs* leistungsfähiger und häufig mit Linux-Derivaten betrieben. Zusammen mit dem meist geringeren Stromverbrauch stellen *SBCs* in kleinen Anwendungen eine Alternative zu herkömmlichen Serverlösungen dar.

3. Stand der Technik

In diesem Kapitel werden Funktionsweisen von mobilen Messgeräten sowie die verschiedenen Messgrößen, welche von diesen erfasst werden, behandelt. Weiterführend werden aktuell am Markt verfügbare Lösungen und Produkte analysiert und ihre Stärken und Schwächen aufgezeigt. Auf die Art der Datenhaltung wird in diesem Zusammenhang ebenfalls eingegangen. Im Anschluss werden die Funktionalitäten, welche die verschiedenen Lösungen der Hersteller bieten, verglichen, um Anforderungen für das Gerät, welches in dieser Arbeit entwickelt wird, zu erheben.

3.1. Funktionsweisen der Messsysteme

Zur Erfassung von verschiedenen Messgrößen beim Transport von Produkten gibt es verschiedene Ansätze, welche sich in ihrer Komplexität, aber auch in der Nutzbarkeit der erzeugten Informationen unterscheiden. Die einfachste Form der Überwachung stellen selbstklebende analoge Indikatoren dar, welche anzeigen, ob ein gewisser Grenzwert der Messgröße überschritten wurde. Zumeist sind diese Indikatoren (siehe Abb. 3.1) nicht wiederverwendbar, was den Vorteil der geringen Anschaffungskosten mindert. Zudem lassen diese Indikatoren keine Rückschlüsse darauf zu, zu welchem Zeitpunkt oder an welchem Ort beim Transport die Grenzwerte überschritten wurden. Dies erschwert die Ursachenforschung und Optimierung der Transportprozesse.
Eine weitere Kategorie sind digitale Messgeräte. Durch die digitale Erfassung lassen sich mehrere Messwerte mit nur einem Gerät aufnehmen. Besitzen diese eine Echtzeituhr, lassen sich die Messwerte mit einem Zeitstempel abspeichern, was den potentiellen Informationsgehalt der Daten steigert. Das manuelle Auslesen der Daten stellt jedoch

einen extra Schritt in der Nachbearbeitung der Daten dar. Anzumerken ist, dass einige Hersteller digitale Messgeräte auch als nicht wiederverwendbare Lösungen vertreiben. Preislich sind diese Geräte um ein vielfaches teurer als analoge Indikatoren [31].

Echtzeitfähige Geräte zur Messdatenerfassung übertragen die Messwerte meist über das Mobilfunknetz an einen zentralen Server und umgehen so das manuelle Auslesen der Daten. Einige dieser Geräte ermöglichen zudem die Erfassung der Positionsdaten über das *GNSS*. Die Anbindung an eine zentrale Datenbank bringt dabei umfängliche Anforderungen an die Sicherheit der Daten mit sich. Zudem entstehen laufende Kosten für das Betreiben der Datenbank, sowie die Sim-Karten der Messgeräte. Der gesteigerte Enerigebedarf verringert zudem die potenzielle Laufzeit [32].

Abbildung 3.1.: Analoger Schockindikator der Firma Shockwatch, welcher bei überschreiten des Grenzwertes von 75 g die Farbe des Messstreifens verändert [33].

3.2. Messgröße der Messsysteme

Die zu erfassenden Messgrößen bei dem Transport richten sich je nach Transportgut. Verbreitet ist die Erfassung von Stößen oder Beschleunigungen, welche auf den Sensor wirken. Indikatoren gibt es in verschiedenen festen Sensitivitäten, bei welchen sie auslösen. Die Sensitivitäten liegen bei dem Hersteller *ShockWatch* im Bereich von 0.5 bis 100 g über 50 ms und weisen, je nach Art, Toleranzen zwischen 10 - 15 % auf [34]. Bei den digitalen Sensoren reicht die Sensitivität bis 200 g [35].

Die Messung von Über- oder Unterschreitung einer gewissen **Temperatur** lässt sich ebenfalls mittels Indikatoren realisieren. Einige Indikatoren ermöglichen zudem die Abschätzung der Über- oder Unterschreitungsdauer. Je nach Temperaturbereich beträgt die erfassbare Dauer der Überschreitung mindestens 30 Minuten. Digitale Geräte decken, je nach Ausführung, einen Bereich von -40 bis +85 °C ab [36].

Feuchtigkeitsindikatoren gibt es sowohl reversibel als auch irreversibel. Diese decken in gewissen Abständen einen Bereich von fünf bis 95 % relative Luftfeuchte ab [37] [38]. Eine weitere Variante sind Nässeindikatoren, welche den Kontakt mit Wasser über verlaufende Farbe irreversibel nachweisen. Digitale Geräte sind in der Lage, relative Feuchte im Bereich null bis 100 % (nicht kondensierend) zu erfassen [39].

Kipp- oder **Neigungsindikatoren** können irreversibel nachweisen, ob das Transportgut über einen spezifischen Winkel hinaus gekippt wurde [40]. Die digitale **Erfassung der Lage** kann über einen Beschleunigungssensor und die Erdbeschleunigung ermittelt werden [41].

Ob das Transportgut während des Transportes **Licht** ausgesetzt war, lässt sich mittels digitaler Sensoren erfassen, diese sind jedoch in den meisten Geräten nicht zu finden [42].

Zur **Positionsbestimmung** kann unter Umständen das Mobilfunknetz verwendet werden, verbreiteter und genauer ist jedoch die Bestimmung über *GNSS* [43].

3.3. Kommerzielle Lösungen

In Tabelle 3.1 sind einige fertig zu kaufende Messsysteme aufgeführt. Bis auf das Messgerät *ASPION G-LOG 2* besitzen alle die Möglichkeit, Messwerte über Funk zu übertragen. Das Messgerät *ASPION G-LOG 2* muss am Ende des Transportes ausgelesen werden. Die Batterielaufezit befindet sich jedoch nur im Mittelfeld, was auf eine deutlich höhere Messfrequenz von 50 Messungen in der Sekunde zurückzuführen ist. Die Batterielaufzeit der restlichen Systeme wird anhand ihrer möglichen Sendevorgänge verglichen, da eine Betrachtung der möglichen Laufzeit stark von den Mess- und Sendeintervallen abhängig ist. Der *EFENTO NB-IoT* besitzt die beste Effizienz, jedoch bietet dieser keine Möglichkeit der Positionsbestimmung über *GNSS*. Theoretisch ist eine Positionsbestimmung über *NB-IoT* oder *LTE Cat-M1* möglich, in der Praxis wird jedoch nur eingeschränkte Genauigkeit erreicht [17] [43]. Ausschließlich der *trusted T10 LTE* verfügt über einen wiederaufladbaren Akku, alle anderen Systeme setzten auf auswechselbare Batterien.

3.4. Fazit

Die Betrachtung der verschiedenen Hersteller zeigt, das keines der Produkte ein vollumfängliches Mess- und Datenhaltungskonzept bietet. Einige der Geräte besitzen umfangreiche Sensorik, welche jedoch nicht erweiterbar ist. Herauszuheben sind die Produkte von *EFENTO*, welche mit guter Akkulaufzeit herausstechen, jedoch keine Positionsbestimmung über *GNSS* ermöglichen. Die Produkte von *trusted* bieten die umfangreichste Sensorik in einem Gerät. Problematisch ist die entstehende Abhängigkeit von der cloundbasierten Datenhaltung der Hersteller. Undurchsichtige Systemarchitekturen erschweren die Bewertung der Sicherheit und Langlebigkeit der Produkte. Eine quelloffene Lösung, welche sowohl cloudbasierte, als auch selbst-verwaltete Datenhaltung ermöglicht und eine höhere Flexibilität im Bezug auf die Sensorik bietet, wird von keinem Hersteller angeboten.

Name	***ASPION G-Log 2***	***trusted T10 LTE***	***SpotBot Cellular***	***EFENTO NB-IoT***
Art	offline	online	online	online
Temp [°C]	-30 bis +60	-30 bis +85	-40 bis +85	-35 bis +70
Feuchte [%]	0 bis 85	-	-	0 bis 100
Stoß [g]	24	8	65	-
Druck [hPa]	-	-	-	300 bis 1100
Position	ja	ja	ja	-
Bewegung	-	ja	-	-
Licht	-	ja	-	-
Neigung	-	ja	-	-
Batterie	CR2032	94Wh Lithium	einweg	2x 2100mah einweg
Laufzeit [Tage]	450	2555	243	3650
Sendevorgänge	-	2555	2400	9636

Tabelle 3.1.: Auswahl kommerzieller Messsysteme mit Sensorik und Messbereichen

4. Anforderungen

In diesem Kapitel werden die Anforderungen an das zu entwickelnde Messsystem ermittelt. Dabei werden im Austausch mit den Mitarbeitenden des Institutes Anwendungsfälle aus dem betrieblichen Alltag analysiert und daraus Use Case Diagramme erstellt. Darauf folgend wird anhand der erarbeiteten Informationen eine Anforderungsliste generiert, welche im folgenden Kapitel zur Auslegung und Validierung des Messystems herangezogen wird.

4.1. Anwendungsfälle am Institut

Zusammen mit Mitarbeitenden des Institutes wurden zwei mögliche Anwendungsfälle für das in dieser Arbeit zu entwickelnde Messsystem erarbeitet. Die Anwendungsfälle orientieren sich dabei an wiederkehrenden Situationen im Betriebsalltag des Institutes, in welchen eine echtzeitfähige Datenerfassung einen Mehrwert zur Prozessoptimierung beiträgt. Die Generalisierung und Abstraktion dieser Situationen ergaben den Anwendungfall der Transportüberwachung, die eine Standortermittlung, Erfassung der Temperatur, Luftfeuchte und Luftdruck sowie die Erfassung von Beschleunigungen, die auf das Transportgut wirken, umfasst. Diese Messdaten sollen in Echtzeit an einen zentralen Server geleitet werden, mit welchem sich die Daten durchsuchen, visualisieren und exportieren lassen. Die Frequenz der Messdatenerfassung während der Transportüberwachung soll dabei so einstellbar sein, dass der komplette Transportprozess abgedeckt werden kann.
Ein weiterer generalisierter Anwendungsfall ist die Aufnahme von Beschleunigungswerten und das Abspeichern dieser direkt auf dem Gerät, zur späteren Auswertung.

4.2. Use Case Diagramme

Use Case Diagramme stellen in einer abstrahierten grafischen Darstellung Interaktionen von Akteurinnen oder Akteuren mit einem System dar. Sie eignen sich zur visuellen Repräsentation von vom Akteur oder von der Akteurin ausführbaren Aktionen an einem System und dessen Reaktion darauf. Dabei werden mögliche Verzweigungen vom System nach einer Aktion des Akteurs oder der Akteurin mittels vereinfachter Relationen dargestellt. Diese Relationen umfassen einschließen und erweitern, je nach dem ob der Anwendungsfall zwingend einen weiteren Anwendungsfall einschließt oder nur unter bestimmten Voraussetzungen erweitert wird.

Messdatenerfassung - Use Case Diagramm Das Use Case Diagramm für die Messdatenerfassung (siehe Abb. 4.1) besitzt eine Akteurin oder einen Akteur, welcher mit dem Messsystem Messdaten erfassen möchte. Dabei gibt es zwei Arten der Messwerterfassung: Zum einen Beschleunigungsmessung, welche einschließt, dass die gemessenen Sensordaten auf dem Gerät gespeichert werden. Zum anderen gibt es die Transportüberwachung, welche durch die Verschlüsselung der gemessenen Sensordaten erweitert werden kann. Die Anwendungsfälle des Verbindungsaufbaus und der Messdatenweiterleitung werden mit oder ohne Verschlüsselung eingeschlossen.

Server - Use Case Diagramm Das Use Case Diagramm für die Interaktion mit dem Server (siehe Abb. 4.2) kann einen oder mehrere Akteurinnen oder Akteure beinhalten. Die Akteurinnen und Akteure können mit dem System Daten exportieren, Daten visualisieren oder Daten durchsuchen. Jeder dieser Anwendungsfälle schließt den Anwendungsfall der Bereitstellung von den Daten ein. Das Empfangen von Daten stellt einen Anwendungsfall ohne direkten Akteur oder Akteurin dar. Das Empfangen schließt die Kontrolle, Konsolidierung und Speicherung der empfangenen Daten ein. Das Speichern der Daten wird wiederum erweitert durch das Bereitstellen der Daten.

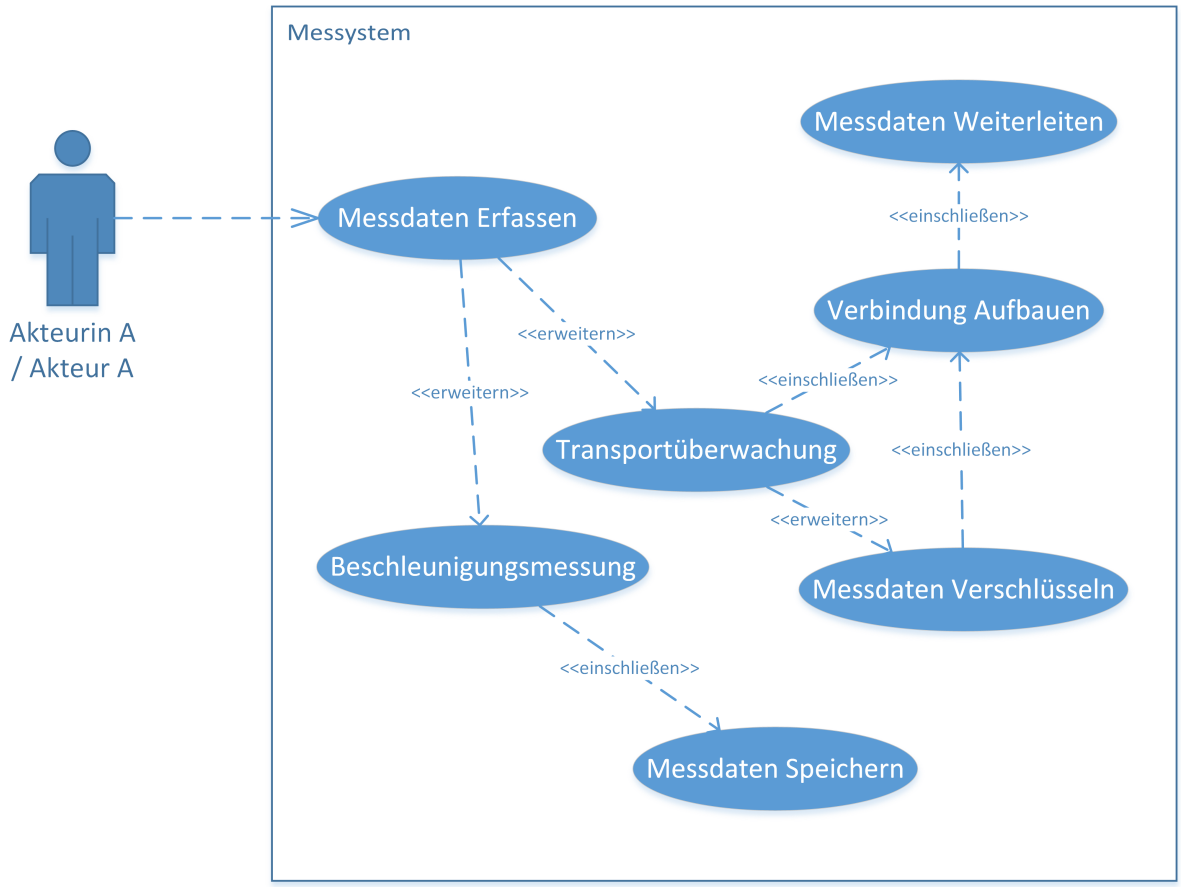


Abbildung 4.1.: Das Use Case Diagramm für das Messsystem besteht aus einer Akteurin oder einem Akteur, welche oder welcher mit dem System Messdaten erfassen will. Das Messsystem kann dabei auf eine Beschleunigungsmessung gestellt werden, in welcher die erfassten Messdaten direkt abgespeichert werden. Wird es auf die Transportüberwachung gestellt, werden die Messdaten optional verschlüsselt, bevor das Messsystem Verbindung mit dem Netzwerk aufbaut. Anschließend werden die Daten an einen Server weitergeleitet.

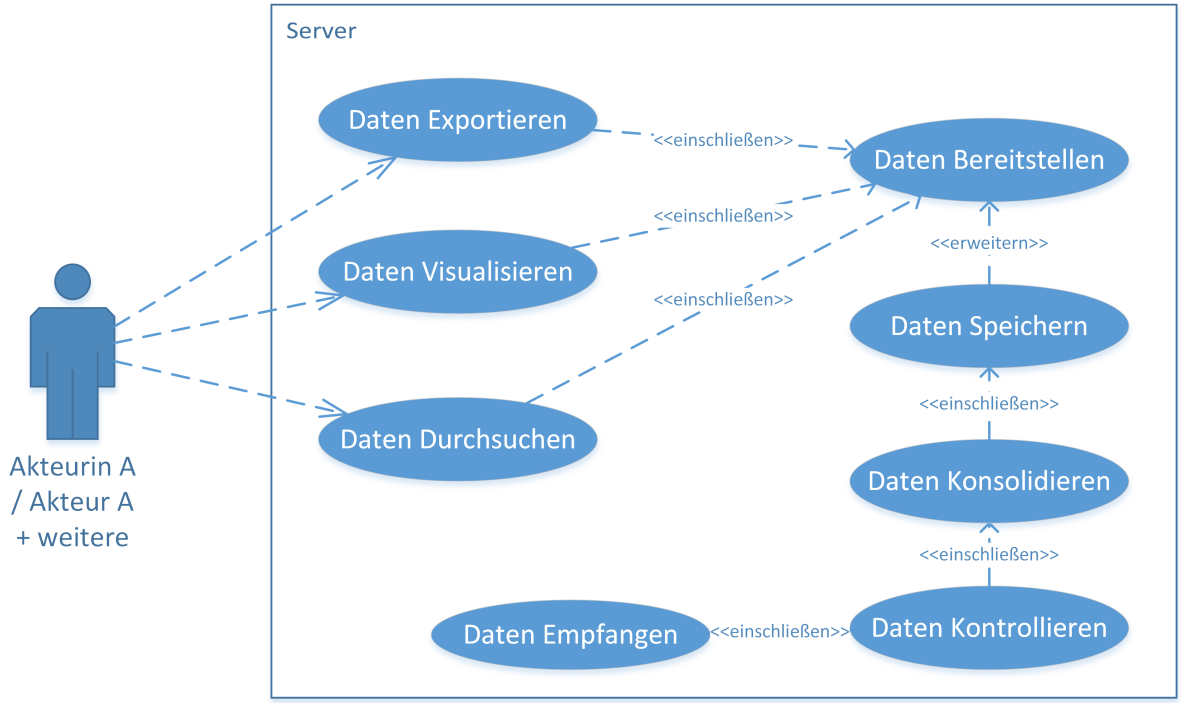


Abbildung 4.2.: Das Use Case Diagramm für den Server besteht aus einer oder mehreren Akteurinnen oder Akteuren, welche mit dem System Daten exportieren, visualisieren oder Daten durchsuchen möchten. Bei allen Anwendungsfällen werden die Daten vom Server bereitgestellt. Auf der anderen Seite muss der Server die Daten der Messgeräte empfangen können. Diese werden im weiteren kontrolliert und konsolidiert und anschließend abgespeichert.

4.3. Anforderungsliste

Aus den vorangegangenen Use Case Diagrammen und aus den mit den mitarbeitenden Personen am Institut generierten Anwendungsfällen ergeben sich die Anforderungen in Tabelle 4.1. In Abstimmung mit den Mitarbeitenden des Institutes wurden die Anforderungen in erforderlich und optional unterteilt.

Nr	**Erforderlich/Optional**	**Beschreibung/Name der Anforderung**
1.1	E	Mobile sensorische Erfassung von Messwerten
1.2	E	Erfassung von Temperatur, Luftfeuchte und Druck
1.3	E	Standortermittlung über GPS
1.4	E	Erfassung von Beschleunigung und Rotation
1.5	E	Kommunikation über Ländergrenzen hinaus
1.6	E	Konsolidierung der Messwerte an einer Stelle
1.7	E	Lokale Speicherung der Messwerte bei Kommunikationsproblemen
1.8	E	generische Codebasis
1.9	E	verständliche Dokumentation
1.10	E	Export der Daten in CSV-Format
1.11	E	Standardisierte Kommunikation
1.12	O	Einbinden neuer Sensorik
1.13	O	Übermittlung von Systeminformationen
1.14	O	Visualisierung der Daten in Dashboards

Tabelle 4.1.: Anforderungsliste, abgeleitet aus den Use Case Diagrammen und den mit den Mitarbeitenden des Institutes generierten Anwendungsfällen.

5. Material und Methoden

In diesem Kapitel wird ein Konzept für das Messsystem entwickelt. Dies umfasst neben der grundlegenden Konzeption auch die Auswahl geeigneter Einzelkomponenten des Systems. Weiterführend werden Prototypen gefertigt und anhand von Tests validiert. Diese Tests wurden aus den Anforderungen und den definierten Anwendungsfällen abgeleitet.

5.1. Konzept

In dem Kapitel Konzept wird das Grundlegende Messkonzept wie in Abb. 5.1 definiert. Des weiteren werden Lösungen für die Teilaufagben des Messsystems ausgewählt und diese Entscheidungen begründet.

5.1.1. Auswahl der Technologien

Datenbank Für die Umsetzung des Prototypen wird die Dokumentendatenbank *Elasticsearch* verwendet. Der Ansatz, Daten semistrukturiert und schemafrei zu speichern, erlaubt den anfänglichen Fokus der Entwicklung vom Definieren genauer Datenstrukturen auf andere Aspekte zu verlegen. Die Einsparung beim Arbeitsaufwand begünstigt die Entwicklungsgeschwindigkeit des Prototypen. *Elasticsearch* zeichnet sich zudem durch eine ausgesprochen gute vertikale Skalierbarkeit aus. Zudem bietet *Elasticsearch* Schnittstellen für die Abfrage und Auswertung von Geodaten direkt in der Datenbank. In Kombination mit der grafischen Oberfläche *Kibana* bietet die Datenbank eine intuitiv zu bedienende und leistungsstarke Möglichkeit, Daten zu visualisieren und zu exportieren.

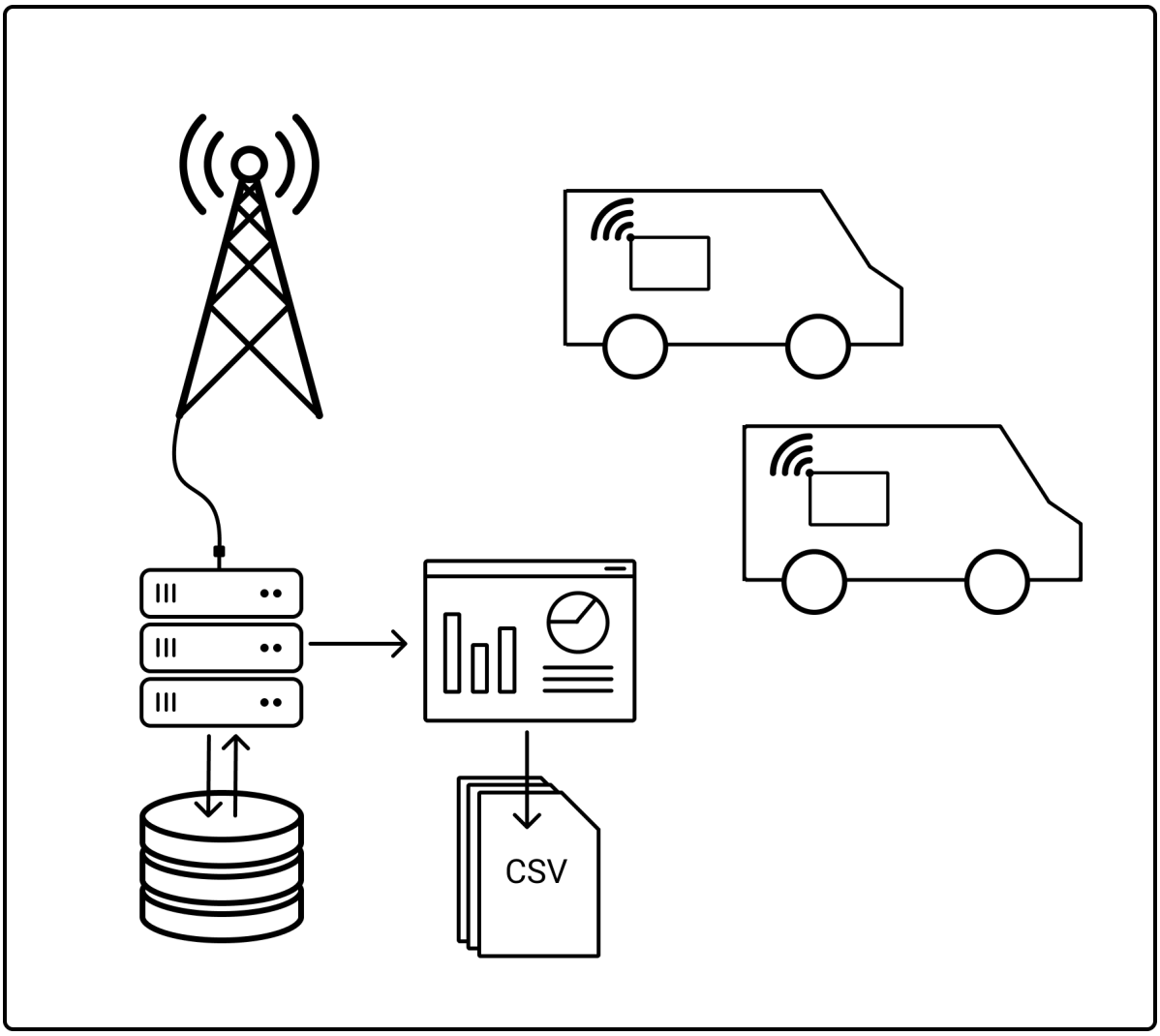


Abbildung 5.1.: Schematische Darstellung des Messsystemkonzeptes. Dargestellt sind die mobilen Messgeräte in Transportern, sowie wie ein Sendemast und der zentrale Server mit Datenbank, Benutzeroberfläche und Datenexport.

Visualisierung und Export Die Visualisierung und der Export der Daten ist eine wichtige Anforderung an die Umsetzung. *Kibana* bietet durch die direkte Schnittstelle zu *Elasticsearch* eine Möglichkeit, ohne zusätzlichen Aufwand eine Visualisierungs- und Analyseoberfläche für die Daten in das System zu integrieren. Die Benutzeroberfläche bietet durch vorgefertigte Visualisierungsarten unerfahrenen Nutzerinnen und Nutzern die Mittel, Daten schnell zu visualisieren. Erfahrene Nutzerinnen und Nutzer können mittels verschachtelter Aggregationen und programmierbaren Pipelines komplexe Auswertungslogiken abbilden und übersichtlich darstellen. Das Zusammenfügen einzelner Visualisierungen zu interaktiven Dashboards ermöglicht es, aus großen Datenmengen in Echtzeit Informationen aus den Messdaten des Systems zu ziehen. Da *Kibana* als Webapplikation angelegt ist, bestehen keine Anforderungen bezüglich des Betriebssystems. Dies ermöglicht die Nutzung auf mobilen Endgeräten. Neben der Visualisierung stellt *Kibana* einen Weg bereit, die Datenbank in Echtzeit zu durchsuchen und die Ergebnisse weiter zu filtern. Diese Suchen sind in *Kibana* speicherbar und im Anschluss sowohl als CSV Datei zu exportieren als auch über einen permanenten Link in anderen Programmen nutzbar zu machen. Zusätzlich können Nutzer und Rollen definiert werden, welche nur Zugriff auf bestimmte Bereiche und Daten haben. Diese können sich eigene Suchen, Visualisierungen und Dashboards abspeichern und später darauf zugreifen. Zudem bietet *Kibana* die Möglichkeit, Geodaten auf einer Karte zu visualisieren. Dabei können verschiedene Aggregationen, welche über den Standort ausgeführt werden, überlagert werden. Dazu gehören unter anderem Heatmaps oder Rasterkarten. *Kibana* ermöglicht ebenfalls das Einbinden vorhandener Straßenkarten.

Mikrocontroller Die Auswahl des Mikrocontrollers wird maßgeblich durch die Anforderungen an die Konnektivität beeinflusst. Für dieses Projekt wird der Mikrocontroller *ESP32* von der Firma *Espressif* verwendet. Dieser bietet mit seinem Dualcore Prozessor eine gute Basis. Bedingt durch die große Community dieses Mikrocontrollers gibt es zahlreiche gut dokumentierte Programmbibliotheken, welche die Programmierung erleichtern und erheblich beschleunigen. Verschiedene Hersteller bieten zudem auf spezielle Anwendungen ausgelegte Entwicklungsplatinen auf Basis des *ESP32* an.

Funktechnologie Für die Übertragung der Daten an die Datenbank werden die Mobilfunkstandards *GSM* und *LTE* verwendet. Das in der Entwicklungsplatine verwendete Funkmodul beherrscht die Mobilfunkstandarts *LTE, GSM, EDGE* und die *IoT* Standards *LTE CAT-M1* und *NB-IoT*. Der verwendete Anbieter *1NCE* benutzt das Mobilfunknetz der *Telekom*, somit kann in dieser Arbeit auf ein gut ausgebautes Mobilfunknetzwerk zurückgegriffen werden.

Sensoren Für die Messung der Umgebungsvariablen wird ein Feuchtigkeitssensor des Typs *Bosch BME280* verwendet. Dieser Sensor bietet die Möglichkeit, mit geringen Abmaßen Temperatur, relative Feuchte und den Umgebungsdruck zu messen und diese via *Inter-Integrated Circuit (I^2C)* oder *Serial Peripheral Interface (SPI)* digital weiterzuleiten. Dies ermöglicht den Betrieb an Geräten ohne eigenen *Analog Digital Converter (ADC)*.

Reverse Proxy Zum Schutz der Datenbank und der API vor unbefugten Zugriffen wird der Reverse Proxy *nginx* verwendet. Dieser zeichnet sich vor allem durch die hohe Geschwindigkeit aus. *Nginx* ist einer der am weitest verbreiteten Reverse Proxies und bietet außerdem die Möglichkeit, als Webserver statische Inhalte wie Webseiten bereitzustellen.

Application Programming Interface Für die Umsetzung der API wird *Node.js* verwendet, welches die Javascript-Laufzeitumgebung *V8* des *Chrome* Webbrowsers nutzt, um Javascript-Programme auszuführen. Über Programmbibliotheken lassen sich die Grundfunktionen von Javascript erweitern. Die Programmbibliothek *Express.js* ermöglicht eine einfache und schnelle Erstellung auch komplexer APIs. Zur Laufzeitüberwachung des *Node.js* Programmes wird der Prozessmanager *PM2* verwendet, welcher das Programm bei Fehlern automatisiert erneut starten kann und zudem eine Ressourcenüberwachung ermöglicht.

Virtualisierung Zur Virtualisierung wird die Containervirtualisierung *Docker* verwendet, welche durch das Bereitstellen einheitlicher Container für die einzelnen Bestandteile des Systems sowohl die Entwicklung als auch den Übergang in die Produktivumgebung erleichtert und beschleunigt. Die durch die Container geschaffene Abstraktion der Laufzeitumgebungen vereinfacht ebenfalls die Überwachung der einzelnen Prozesse.

Server Der Server wird als "in house"-Lösung konzipiert und an das Institutsnetzwerk angeschlossen. Als Betriebssystem wird die Linuxdistribution *Debian* in der Version *Buster* verwendet. Die einzelnen Komponenten der Toolchain werden separat in Containern virtualisiert. Jeglicher Netzwerkverkehr wird über den Reverse Proxy geleitet, welcher die Anfragen an API und Datenbank umleitet.

5.1.2. Hardware

Für die Geräte wird die Entwicklungsplatine *TTGO T-SIM7000G* der Firma *Xinyuan-LilyGO* verwendet. Diese besitzt neben dem *ESP32* Mikroprozessor zusätzlich ein *SIMCOM SIM7000G* Mobilfunkmodem mit NanoSim-Steckplatz sowie eine *LTE* und *GPS* Antenne, einen USB-C Anschluss zur Stromversorgung sowie zum Aufspielen der Firmware, einen Laderegler, eine Aufnahme für einen *18650* Lithium-Ionen Akku und einen Steckplatz für eine microSD-Karte. Die Pinzuordnung kann im Abblidung 5.1.2 entnommen werden, die restlichen Pins werden einerseits zum Anschließen des *BME280* Sensors sowie eines *MP-6050* Beschleunigungssensors verwendet.

Programmierung Die Programmierung des Mikrocontrollers wird mittels C/C++ realisiert. Zur besseren Strukturierung wurden die Funktionen vorab als Programmablaufplan visualisiert (siehe Anhang A.1). Die verschiedenen Funktionen werden nach Möglichkeit über vorhandene Bibliotheken realisiert.

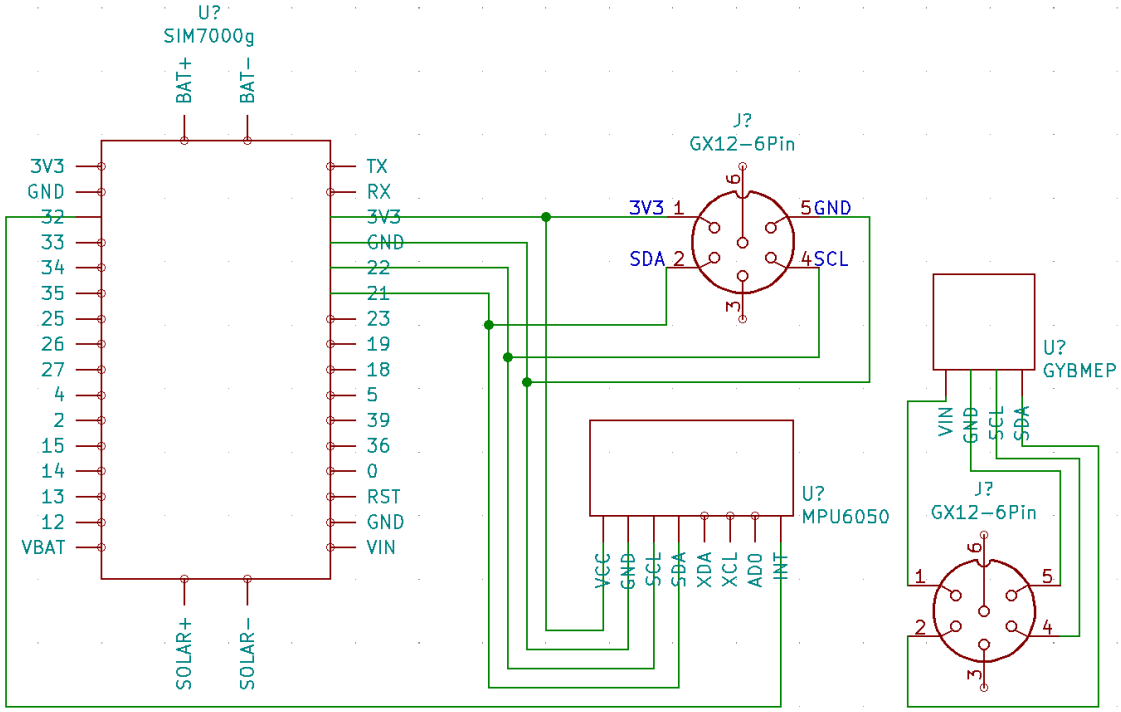


Abbildung 5.2.: Pinbelegungsdiagramm des *T-SIM7000G* Entwicklungsboards. Die für das *I2C* Protokoll benötigten Pins *SDA*, *SCL*, *GND* und *3V3* werden sowohl mit dem Beschleunigungssensor als auch mit der Buchse für den externen Sensor verbunden. Zusätzlich wird Pin 32 der Entwicklungsplatine mit dem Interruptpin des Beschleuniungssensors verbunden.

Schnittstellen Die Definition von Schnittstellen kann dabei helfen, verschiedene Systemkomponenten parallel zu entwickeln. In dem zum Beispiel die Datenstruktur der zu übermittelnden Messdaten vorab definiert wird, können die sendende und die empfangende Komponente voneinander losgelöst entwickelt werden, da im Vorhinein klar definiert wurde, welche exakte Struktur und Form die Daten bei der Übergabe haben müssen. Neben den eigentlichen Messdaten umfassen die übertragenen Datenpakete zusätzlich Konfigurationsparameter wie das durch die Nutzerin oder den Nutzer eingestellte Sendeintervall. In Abb. 5.4 ist die Schnittstellendefinition der von dem Messgerät an den Server gesendeten Datenpakete definiert. Neben den Standarddatentypen werden auch eigens definierte Datenformate eingebunden, diese sind in Abb. 5.3 und Abb. 5.5 zu sehen. Das Darstellungsformat dieser Definition entstammt der Programmiersprache *TypeScript*. Ein Interface wird in diesem Format mittels *interface* und dem Namen der zu definierenden Schnittstelle begonnen. In den geschweiften Klammern folgen die einzelnen Definitionen der Schlüssel-Wert-Paare. Diese beginnen mit dem Namen des Schlüssels, gefolgt von einem Doppelpunkt und der Angabe des Datentyps und endet mit einem Semikolon. Folgt auf den Namen des Schlüssel direkt ein Fragezeichen, so ist dieser optional. Werden zusätzlich zu den in *TypeScript* verfügbaren *primitiven Datentypen* anderweitige Datenstrukturen in einer Schnittstelle benötigt, können diese in einem separaten *interface* definiert werden. In der Schnittstelle *Daten* werden die in Abb. 5.3 und Abb. 5.5 definierten Datentypen *Location* und *Vector* verwendet.

```
1 interface Location {
2     lat: number;
3     lon: number;
4 }
```

Abbildung 5.3.: Schnittstellendefinition des zusätzlichen Typen *Location*.

```typescript
interface Data {
    type: string; // Typ des Messgerätes,
    deviceID: string; // nicht veränderbare Hardware-ID
    sdStatus: boolean; // SD-Karte eingesteckt und erkannt
    sendInterval: number; // nutzer definierbares Sendeintervall
    millis: number; // Millisekunden seit Start des Gerätes
    gprsConnected: boolean; // Mobilfunkverbindung erfolgreich
    ccid?: string; // Mobilfunk-Identifikationsnummern
    imei?: string; // Mobilfunk-Identifikationsnummern
    imsi?: string; // Mobilfunk-Identifikationsnummern
    operator?: string; // Funknetzbetreiber
    signalQuality?: number; // Zahlenwert für die Dämpfung
    gsmIP?: string; // IP-Adresse des Gerätes
    gsmTime?: Date; // Uhrzeit des Funkmasten
    gsmLocation?: Location; // Standort des Funkmasten
    gsmLocationSwitched?: Location; // Standort des Funkmasten
    gsmAccuracy?: number; // Genauigkeit des Funkmaststandortes
    envSensorStatus: boolean; // BME280 Sensor initialisiert
    temperature?: number; // Temperaturmesswert des BME280
    pressure?: number; // Luftdruckmesswert des BME280
    humidity?: number; // rel. Luftfeuchtemesswert des BME280
    accSensorStatus: boolean; // Beschleunigungsensor initialisiert
    acceleration?: Vector; // Beschleunigung in alle Raumrichtungen
    gyroscope?: Vector; // Drehbewegung um alle Raumachsen
    gpsConnected: boolean; // GPS-Signal empfangen
    location?: Location; // Positionsdaten
    altitude?: number; // Höhe über Meeresspiegel
    GPSspeed?: number; // Geschwindigkeit
    GPSaccuracy?: number; // Genauigkeit GPS-Signal
    GPSage?: number; // Dauer bis zur erfolgreichen GPS-Verbindung
    GPStime?: Date; // Datum und Uhrzeit der GPS-Daten
    batteryVoltage: number; // Batteriespannung
}
```

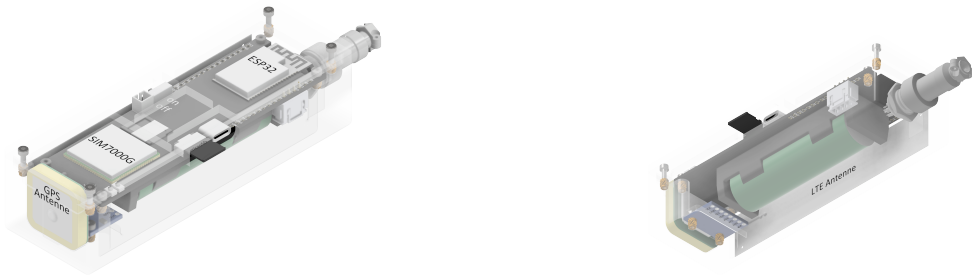
Abbildung 5.4.: Schnittstellendefinition des Messdatenpaketes.

5.1.3. Konstruktion der Gehäuse

Bei der Konstruktion der Prototypen wurde die Anforderung der zerstörungfreien Demontierbarkeit ergänzt, da dies eine langfristige Reparierbarkeit der Geräte begünstigt und Anpassungen erleichtert. Alle Teile sollen ebenfalls preisgünstig und möglichst ohne weitere Bearbeitungsschritte additiv fertigbar sein. Durch die Nutzung des *TTGO T-SIM7000G* Entwicklungsboards ist eine gewisse Positionierung der restlichen Bauteile vorgegeben. Vor allem durch den auf der Rückseite befindlichen Steckplatz des *18650* Akkus ist der Bauraum unterhalb der Entwicklungsplatine eingeschränkt. Zudem sind Bereiche, mit welchen der Nutzer beim Gebrauch des Messgerätes interagiert, ausreichend erreichbar zu gestalten. Der An-Aus-Schalter des *T-SIM7000G* Entwicklungsboards ist durch das Gehäuse nur mit einem Hilfsmittel zu bedienen, um versehentliches Ausschalten während des Transportes zu verhindern. Aufgrund der Tatsache, dass der USB-C Anschluss zum Laden und Programmieren des Entwicklungsboards direkt an den microSD-Steckplatz angrenzt, gibt es zwei Varianten des Gehäusedeckels. Eine Variante bietet keinen Schutz für die microSD-Karte, erleichtert jedoch den Zugang zum USB-C Anschluss. Die andere Variante schützt die microSD-Karte vor ungewollter Krafteinwirkung bei dem Transport, verdeckt dadurch aber den USB-C Anschluss. Somit muss zum Laden und zum Auslesen der microSD-Karte der Deckel mit vier Schrauben entfernt werden. Aufgrund der Größe der *LTE*- und *GPS*-Antennen belegen diese alleine eine Längs- und Stirnseite. Auf der zweiten Stirnseite befindet sich der Anschluss für den externen Sensor. Der Beschleunigungssensor findet unter der Entwicklungsplatine, zwischen Akku und *GPS*-Antenne, platz. Alle Schraubverbindungen werden über Einschmelzgewinde realisiert, um mehrmaliges Öffnen der Geräte zerstörungsfrei zu gewährleisten.
Der *BME280* Umweltsensor befindet sich in einem sepaten Gehäuse, bestehend aus Ober- und Unterseite. Über die Verschraubung der beiden Gehäusehälften wird das Kabel des Sensors verklemmt, um eine Zugentlastung der Lötstellen zu erzeugen. Beide Gehäuse werden mittels *Fused Deposition Modeling (FDM)* aus *Acrylnitril-Butadien-Styrol-Copolymer (ABS)* gedruckt.

```
interface Vector {
    x: number;
    y: number;
    z: number;
}
```

Abbildung 5.5.: Schnittstellendefinition des zusätzlichen Typen *Vector*.



(a) Gehäuse oben

(b) Gehäuse unten

Abbildung 5.6.: Das entworfene Gehäuse der Messgeräte. An der Stirnseite befindet sich der Anschluss für die externe Sensorik. Die Antennen für *GPS* und *LTE* befinden sich an den Seiten des Gerätes, um einen möglichst ungestörten Empfang zu ermöglichen. Unter der Platine des *SIM700g* sitzen der *MPU-6050* Beschleunigungssensor sowie ein *18650* Lithium-Ionen-Akku. Der Anschluss des Beschleunigungssensors und der Buchse für die externe Sensorik mit der Hauptplatine erfolgt durch einen Stecker, um den Zusammenbau zu erleichtern.

5.2. Validierung

In diesem Unterkapitel werden die Use Cases in einzelne Funktionalitäten abstrahiert und daraus validierbare Testszenarien generiert. Im weiteren Verlauf werden Aufbauten und Abläufe der Versuche beschreiben.

5.2.1. Energieverbrauchsmessung

Zur Auslegung der Batterielaufzeit der Geräte wird der Stromverbrauch über einen Messvorgang erfasst. Durch das Abspeichern der Zeit beim Durchlaufen der verschiedenen Codebausteine durch den Mikrocontroller können diese über die Stromverlaufskurve gelegt werden. Somit ergibt sich ein Graph, welcher Aufschluss über die Strombedarfe der einzelnen Bauteile der Entwicklungsplatine ermöglicht.

Versuchsablauf Der *18650* Akku wird aus dem Gerät entnommen und anstatt dessen wird eine Stromquelle angeschlossen, welche die Möglichkeit besitzt, Ströme und Spannungen über den Versuchszeitraum aufzunehmen. Das Gerät wird auf die Standardeinstellungen konfiguriert (siehe Abb. 5.7). Über einen Computer wird ein Debugmodus aktiviert, welcher die Zeitstempel sowie die Namen der aktuellen Funktion, bei welcher sich das Gerät im Programmablauf befindet, auf die microSD-Karte schreibt. Das Gerät wird vom Computer getrennt und anschließend eingeschaltet. Nachdem drei Sendevorgänge durchlaufen wurden, wird das Geräte ausgeschaltet und die microSD-Karte entnommen.

Auswertung Zur Auswertung werden die Messwerte über einen Sendevorgang erfasst. Die über den Debugmodus geschriebenen Zeitpunkte werden mittels eines Pythonskriptes mit den Messungen der Stromquelle überlagert (siehe Abb. 6.1). Es werden sowohl der durchschnittliche Strombedarf des Sendevorganges berechnet als auch der durchschnittliche Strombedarf während der Standortbestimmung. Da dies programmatisch bedingt einen Zeitraum von 0 - 180 Sekunden benötigt, lassen sich durch die Bestimmung des minimalen und maximalen Strombedarfs für die Standortbestimmung ein Best- und ein Worst-Case-Szenario errechnen.

```
{
    "name":"Test1",
    "measureInterval":60,
    "sendInterval":60,
    "GPSenabled":1,
    "accelerometerRange":16,
    "gyroRange":2000,
    "motionDetection":0,
    "encryption":1,
    "accelMode":0
}
```

Abbildung 5.7.: Gerätekonfiguration für die Energieverbrauchsmessung im JSON-Format. Festlegung des Namens, des Messintervalls, des Senseintervalls, der GPS Positionsbestimmung, der Grenzwerte des Beschleunigungssensors und des Gyroskops, der Bewegungserkenung, der Verschlüsselung und des speziellen Modus zur Beschleunigungsmessung.

5.2.2. Beschleunigungsmessung

Eine der möglichen Anwendungen für das in dieser Arbeit konzeptionierten Messsystems ist die Erfassung von Beschleunigungen in alle Raumrichtungen. Einerseits können über die Summe der absoluten Beschleunigungswerte für die drei Raumrichtungen Stöße ermittelt werden, welche auf das Gerät bei einem Sturz wirken. Andererseits lässt sich anhand der Erdbeschleunigung die Orientierung der Geräte eindeutig feststellen und so ein Umkippen des Transportbehältnisses festzustellen. Das Feststellen der Orientierung stellt hierbei weniger Anforderungen an das Messsystem, da die Erdbeschleunigung konstant wirkt und somit kein spezieller Messzeitpunkt einzuhalten ist. Die Erfassung eines Stoßes hingegen erfordert entweder, einen genauen Messzeitpunkt erfassen zu können, oder eine hinreichend hohe Messfrequenz. Zur Validierung der Geräte wird nachstehend ein Versuchsaufbau und dessen Durchführung definiert. Die Prüfung orientiert sich bei der Fallhöhe an der *DIN EN 22248 Vertikale Stoßprüfung (freier Fall)* [44]. Die anderen Parameter sowie die Durchführung wurden angepasst.

Versuchsablauf Zur Durchführung des Versuches werden drei Geräte mit gleicher Orientierung in ein ausreichend großes Paket gelegt und von allen Seiten mit Knüllpapier gepolstert. Das Paket wird auf eine 80 cm hohe, ebene Fläche gelegt. Da zur Erfassung von Beschleunigungswerten eine ausreichend hohe Messfrequenz benötigt wird, werden die Geräte über die Konfigurationsdatei 5.9 in einen Modus versetzt, bei welchem keine *GPS*-Daten erfasst werden und keine Kommunikation über *LTE* stattfindet. Die Geräte erfassen die Daten und schreiben diese auf die SD-Karte. Die Geräte eingeschaltet und das Paket zugeklebt. Um die Daten der Geräte bei der Auswertung zu synchronisieren, wird das Paket kurzzeitig auf den Kopf gestellt. Danach wird das Paket wieder richtig herum hingelegt und für kurze Zeit in Ruhe gelassen. Anschließend wird das Paket mit der kurzen Seite von der Fläche geschoben. Nach einer kurzen Ruhezeit wird das Paket wieder richtig herum auf die Fläche gelegt und nach kurzer Zeit erneut mit der kurzen Seite über die Kante geschoben. Diese Schritte werden für lange Seite des Paketes wiederholt. So werden insgesamt vier Stürze aufgezeichnet.

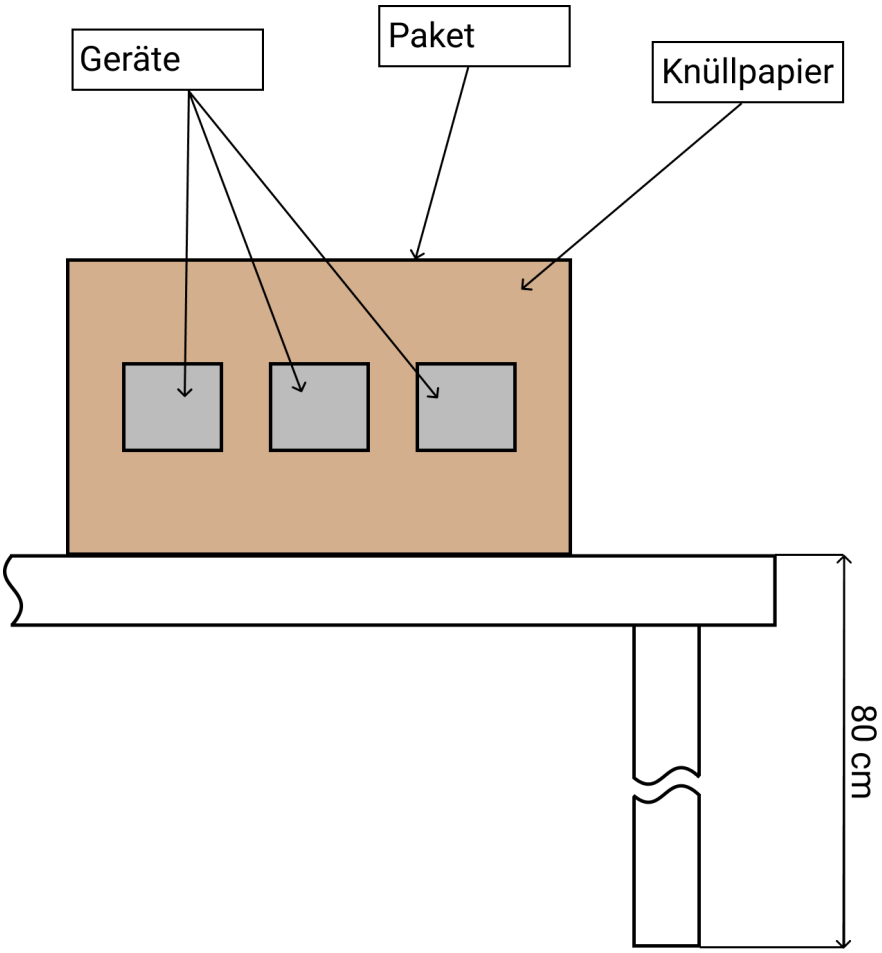


Abbildung 5.8.: Versuchsaufbau der Beschleunigungsmessung. Drei Messgeräte befinden sich in gleicher Orientierung in einem Paket und sind mittels Knüllpapier in alle Richtungen gepolstert. Das Paket befindet sich auf einer 80 cm hohen, ebenen Fläche.

```
{
    "name":"Test1",
    "measureInterval":0,
    "sendInterval":0,
    "GPSenabled":0,
    "accelerometerRange":16,
    "gyroRange":0,
    "motionDetection":0,
    "encryption":0,
    "accelMode":1
}
```

Abbildung 5.9.: Gerätekonfiguration für den Beschleunigungstest im *JSON* Format. Festlegung des Namens, des Messintervalls, des Sendeintervalls, der *GPS* Positionsbestimmung, der Grenzwerte des Beschleunigungssensors und des Gyroskops, der Bewegungserkennung, der Verschlüsselung und des speziellen Modus zur Beschleunigungsmessung.

5.2.3. Validierung der Transportüberwachung

Die Erfassung von Umgebungsparametern bei einem Transport von Waren stellt einen wichtigen Anwendungsfall des Messsystems dar. Während die Erfassung von Temperatur, Luftfeuchte und Luftdruck direkte Aussagen über die Einhaltung gewisser Grenzwerte beim Transport zulassen, können weitere Metainformationen wie der Standort der Geräte helfen, Transportprozesse zu optimieren. Ziel dieses Versuches ist die Validierung der grundlegenden Funktionalität des Messsystems anhand eines realitätsnahen Transportprozesses. Die Geräte werden so konfiguriert, dass sie alle 60 Sekunden Messwerte aufnehmen und an die Datenbank übertragen (siehe Abb. 5.10).

Versuchsablauf Zur Durchführung des Versuches werden drei Geräte in den Kofferraum eines Autos gelegt. Die Umweltsensoren werden in einer Kühlbox plaziert, in welcher sich ein Kühlakku befindet. Anschließend werden die Geräte eingeschaltet. In einem Zeitraum von fünf Stunden werden verschiedene Ziele im Großraum Hannover angefahren. Dabei werden Landstraßen, Bundesstraßen, Autobahnen und innerstädtische Straßen befahren. Nach 90 Minuten werden am ersten Standort Proben in die Kühlbox gelegt und über den weiteren Verlauf des Tages überwacht. Die Messgeräte übermitteln dabei kontinuierlich verschlüsselte Messwerte an eine *Elasticsearch*-Instanz. Über ein Dashboard in *Kibana* lassen sich die Standortdaten und weitere Messwerte in Echtzeit überwachen.

```
{
    "name":"Test1",
    "measureInterval":60,
    "sendInterval":60,
    "GPSenabled":1,
    "accelerometerRange":16,
    "gyroRange":2000,
    "motionDetection":0,
    "encryption":1,
    "accelMode":0
}
```

Abbildung 5.10.: Gerätekonfiguration für die Validierung der Transportüberwachung im *JSON* Format. Festlegung des Namens, des Messintervalls, des Sendeintervalls, der *GPS* Positionsbestimmung, der Grenzwerte des Beschleunigungssensors und des Gyroskops, der Bewegungserkennung, der Verschlüsselung und des speziellen Modus zur Beschleunigungsmessung.

6. Ergebnisse

In diesem Kapitel werden die Ergebnisse aus den vorangegangenen Versuchen aufbereitet und abgebildet. Eingegangen wird auf die Energieverbrauchsmessung, die Beschleunigungsmessung und die Transportüberwachung. Bei der Transportüberwachung wird zur Visualisierung das Dashboard direkt aus *Kibana* herangezogen, da dies einen wichtigen Bestandteil des Messsystems darstellt.

6.1. Energieverbrauchsmessung

Der Graph der Energieverbrauchsmessung zeigt den Stromfluss in Ampere über die Zeit in Sekunden für einen Messvorgang, vom Starten des Gerätes bis zur Übertragung der Daten, inklusive Standortermittlung. Die durchlaufenen Abschnitte im Programm der Messgeräte sind mit vertikalen Linien angezeigt. Die Standortbestimmung ist grün hinterlegt. Ebenfalls wurde die durchschnittliche Stromaufnahme des Gerätes mittels einer schwarzen Linie dargestellt. Für den Bereich der Standortbestimmung wurde aufgrund der variablen Länge eine separate durchschnittliche Stromaufnahme in grün eingezeichnet. Durch den Graphen wird verdeutlicht, dass das *SIM7000g* Modem Stromspitzen beim Verbindungsaufbau zum *LTE*-Netzwerk und beim Einschalten des *GPS*-Empfängers hervorruft. Das Warten auf das *GPS*-Signal verbraucht hingegen unterdurchschnittlich viel Strom. Für das Best- und das Worst-Case-Szenario betragen die Strombedarfe pro Messvorgang 1.49 mAh bis 5.66 mAh. Wird der Stromverbrauch im Tiefschlaf sowie die Eigenentladung des Akkus vernachlässigt, ergeben sich bei einem 3000 mAh Akkus zwischen 529 und 4474 Messungen.

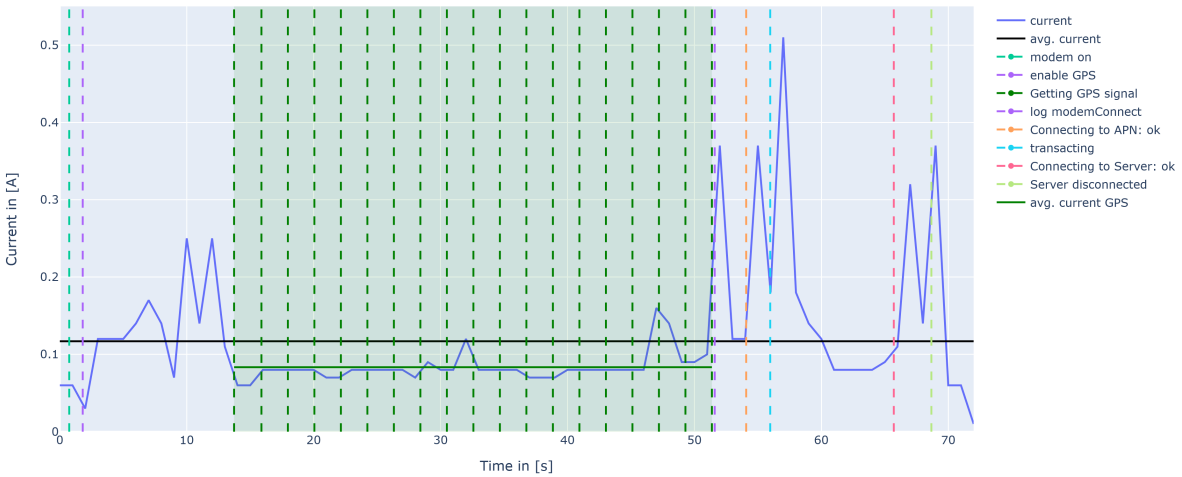


Abbildung 6.1.: Stromverbrauchsmessung bei einem Messvorgang inkl. Übermittlung der Daten. In blau der gemessene Strom und in schwarz der Durchschnittswert über die gesamte Laufzeit. Vertikale Linien stellen Printbefehle dar, welche zu Beginn der Funktionsaufrufe mit Zeitstempel in eine Datei gespeichert wurden. Funktionen, welche mehr als eine Sekunde Laufzeit besitzen, wurden damit einzelnen Bereichen des Graphen zugeordnet. Die grüne horizontale Linie ist der Durchschnittsverbrauch beim Herstellen der *GPS*-Verbindung. Da dies einen stark variablen Zeitraum darstellt, können mit dem Durchschnittswert Best- und Worst-Case-Szenarien ermittelt werden.

6.2. Beschleunigungsmessung

Mit der standardmäßigen Datenhaltung der Geräte ist eine Messfrequenz von ca. 5 Hz möglich. Dies reichte jedoch nicht aus, um die Stöße beim Aufprall eindeutig zu erfassen. Der freie Fall der Geräte war bei allen Messungen sichtbar, jedoch wird nur ein Aufprall bei drei Geräten und zwei Messungen erfasst. Eine Optimierung des Ablaufes für den speziellen Fall der Beschleunigungsmessung bringt eine maximale Messfrequenz von 130 Hz. Da der digitale Beschleunigungssensor jedoch nur Daten in einer Frequenz von 40 Hz bereitstellt, wird der Ablauf dahingehen verlangsamt. Der Versuch wird mit der überarbeiteten Datenverarbeitung durchgeführt.

Durch das kurzzeitige auf den Kopf drehen des Paketes werden in den Verläufen bei allen Geräten negative Werte in Höhe der Erdbeschleunigung in Richtung der Z-Achse erzeugt. So können die Messwerte der Geräte mittels eines Pythonskriptes übereinandergelegt werden. Zudem werden in der Visualisierung vertikale Linien eingefügt, um die einzelnen Stürze zu separieren. Die beiden ersten Bereiche des Diagramms zeigen Stürze, bei welchen das Paket mit der kurzen Seite von der Fläche geschoben wurde. Die beiden folgenden Bereiche zeigen Stürze, bei welchen es mit der langen Seite vom Tisch geschoben wurde. Der Verlauf der Beschleunigungssummen zeigt den Zustand des in Ruhe liegens durch Werte, die ungefähr der Erdbeschleunigung entsprechen. Darauf folgte eine leichte Erhöhung der Messwerte durch das Schieben von der ebenen Fläche. Beim Herunterfallen befinden sich die Geräte im freien Fall, in den Messwerten zeigt sich dies durch Beschleunigungssummen von ungefähr Null. Im Anschluss an den freien Fall folgt der Aufschlag des Paketes auf den harten Untergrund. Nach dem zur Ruhe kommen folgt eine verlängerte Ruhephase, gefolgt von einem sprunghaften Verlauf der Kurven, ausgelöst durch das Aufheben und Platzieren des Paketes auf der 80 cm hoch gelegenen Fläche.

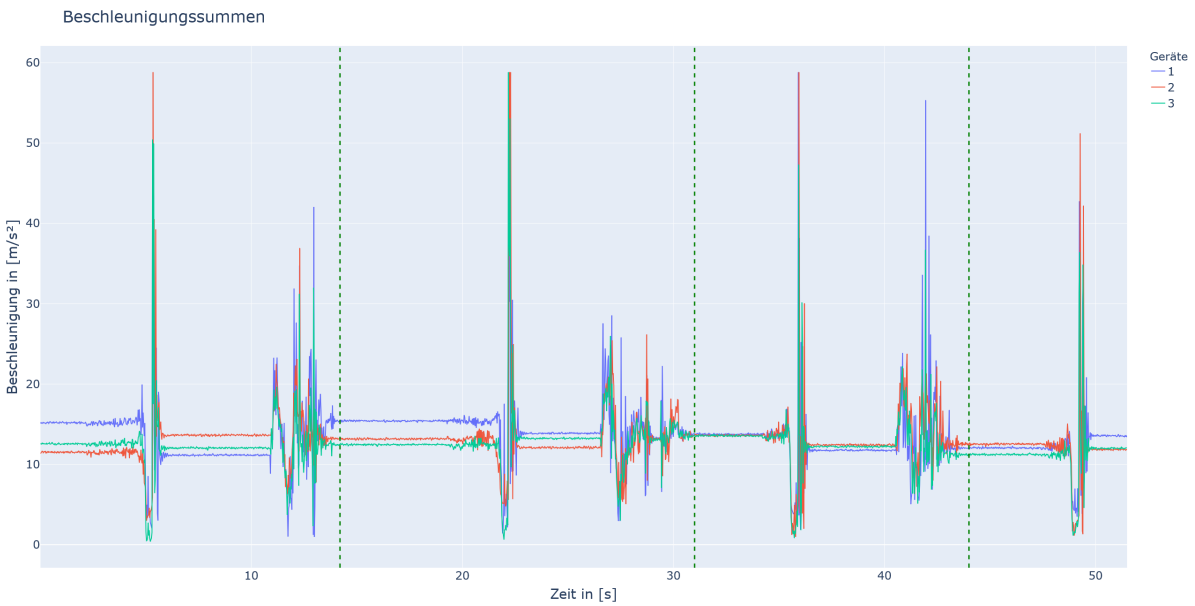


Abbildung 6.2.: Graphen der Beschleunigungsmessung. Abgebildet sind die Summen der Beschleunigungen über alle Raumrichtungen bei freiem Fall aus 80 cm Höhe und Aufschlagen auf harten Boden. Bei den ersten zwei Peaks der Graphen wurde das Paket mit der kurzen Seite von der Fläche geschoben, in den letzten zwei Peaks mit der langen Seite. Die einzelnen Stürze zeigen einen in Ruhe befindlichen Wert, welcher der Erdbeschleunigung entspricht. Anschließend eine leichte Erhöhung durch das Anschieben des Paketes, gefolgt von einem freien Fall. Darauf folgend der Aufschlag und ein Zurückfedern der Geräte. Danach eine weitere Phase in Ruhe und das Aufheben und das erneute Anheben auf die 80 cm hohe ebene Fläche.

6.3. Transportüberwachung

Die an den Server übermittelten Messwerte umfassen, neben den über die Sensoren ermittelten Messgrößen, Temperatur, relative Luftfeuchte, Luftdruck, Beschleunigungen und Position sowie weitere Daten, welche den Informationsgehalt anreichern. Dazu gehören unter anderem Metriken, welche quantifizieren, wie lange die Erfassung und Übermittlung der Messdaten gedauert hat. In Abb. 6.3 werden die übermittelten Standorte der Messgeräte auf einer Straßenkarte dargestellt. Ebenfalls abgebildet sind die Standorte der Funkmasten, mit welchen sich die Geräte über den Versuchszeitraum entlang der Strecke verbunden haben. Über eine Heatmap werden Häufungen von Messwerten an Standorten dargestellt. Diese entsprechen den Standorten, an welchen das Fahrzeug länger gehalten hat. In farbigen Kacheln wird die Zeit dargestellt, welche die Messgeräte benötigt haben, um ihren Standort über ein *GPS*-Signal zu ermitteln. In Abb. 5.10 sind neben den Namen der Messgeräte sowohl die Verläufe von Temperatur, relativen Luftfeuchte und Luftdruck gezeigt, als auch die der Batteriespannung und Signalqualität des Mobilfunksignals zu sehen. In Abb. 6.4 ist zu sehen, dass die Verläufe der gemessenen Temperaturen nach dem Einlegen der Sensoren in die Kühlbox sinken und danach wieder ansteigen. Beim Einlegen der Proben um 15:35 sind Anstiege bei der relativen Luftfeuchte sowie der Temperatur zu verzeichnen. Im weiteren Verlauf bleiben die Werte annähernd konstant. Die Batteriespannung verläuft annähernd linear über den gesamten Versuch. Die Signalqualität der Mobilfunkverbindung schwankt über den Versuchszeitraum.

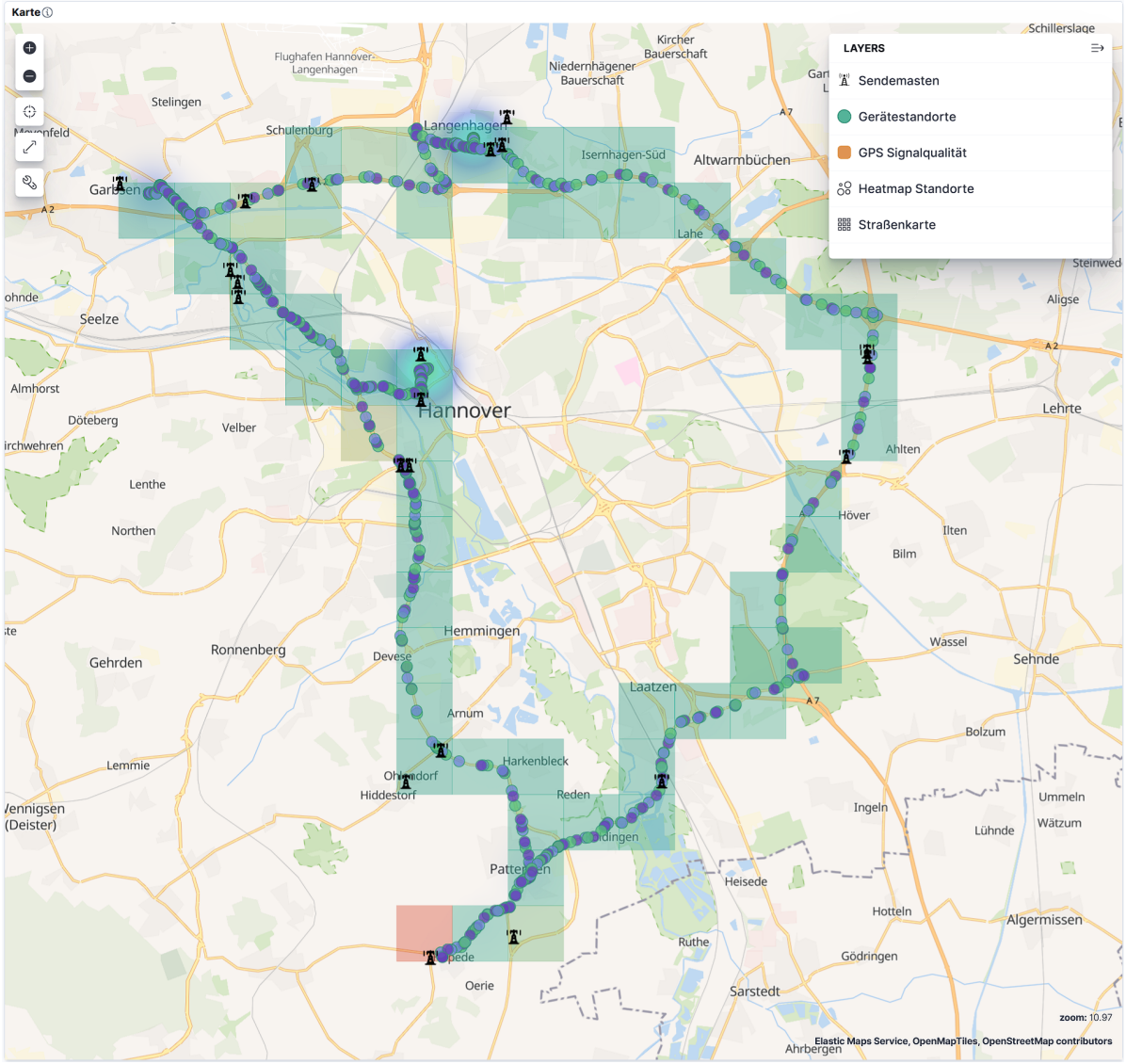


Abbildung 6.3.: Kartendarstellung aus *Kibana*, mit Standorten und Messwerten aus dem Versuch zur Validierung der Transportüberwachung. Die Kreise bilden übermittelte Standorte und Messwerte der drei Geräte ab. Die Heatmap visualisiert Anhäufungen von Messwerten mit ähnlichem Standort, um Standzeiten darzustellen. Die Kacheln entlang der gefahrenen Strecke sind auf der Basis eingefärbt, wie lange die Positionsermittlung über GPS gedauert hat. In schwarz dargestellt sind die Standorte der Funkmasten, mit welchen sich die Geräte entlang der Strecke verbunden haben.

Abbildung 6.4.: Graphendarstellung aus *Kibana*, Messwerte aus dem Versuch zur Validierung der Transportüberwachung. (Links nach Rechts) Die Auflistung der für den Versuch relevanten Geräte mittels einer *Tagcloud*-Darstellung. Zeitlicher Verlauf der Batteriespannung aller Geräte in Volt. Graphendarstellung der *LTE*-Signalqualität (*Received Signal Strength Indication*) in dbm (Dezibel mW). Temperatur-, Luftdruck- und relativer Leuftfeuchteverlauf des Transportgutes

7. Diskussion

In diesem Kapitel werden die Ergebnisse aus den vorangegangenen Tests analysiert. Anschließend werden die erhobenen Anforderungen an das System mit den erreichten Leistungen des Systems abgeglichen und validiert, ob die Anwendungsfälle vollständig durch das System abgedeckt werden. Zudem werden die Ergebnisse mit den technischen Angaben der am Markt käuflichen Systeme verglichen. Dabei wird auf die beiden Messwerterfassungsarten des Gerätes, die Transportüberwachung und die Beschleunigungsmessung, separat eingegangen. Es werden die zur Lösung der Teilaufgaben des Messsystems ausgewählten Systemkomponenten durchgegangen und analysiert, ob die einzelnen Komponenten den in der Theorie erdachten Mehrwert bringen. Weiterführend werden mögliche Schwachstellen des Systems aufgezeigt und deren Auswirkungen auf das Gesamtsystem in einen Kontext gesetzt.

7.1. Beschleunigungsmessung

Der Anwendungsfall der Beschleunigungsmessung zeigt die technischen Beschränkungen des Systems auf. Eine Echtzeitübermittlung der Messdaten war in diesem Anwendungsfall nicht zu realisieren. Die Latenz beim Aufbau einer Netzwerkverbindung ist zu groß, um eine unterbrechungsfreie Erfassung der Beschleunigungswerte zu gewährleisten. Rechnerisch ergeben sich bei einer Messfrequenz von 40 Hz nur 25 ms zur weiteren Verarbeitung der Messdaten. Das Erfassen und Speichern der Messdaten im CSV-Format auf die microSD-Karte benötigt ungefähr acht Millisekunden, wohingegen zum Verbindungsaufbau zu einem Mobilfunknetz und zum Senden der Daten in der aktuellen Implementierung allein über zehn Sekunden benötigt werden. Durch den speziellen Modus

zur Beschleunigungsmessung wurde jedoch eine Messfrequenz von 40 Hz erreicht, was durch die Limitierung des verwendeten Sensors bedingt ist. Dahingehend wurden die für diesen Anwendungsfall definierten Anforderungen erreicht. Ob diese Messfrequenz ausreichend ist um alle Szenarien abzudecken, in denen eine Beschleunigungsmessung benötigt wird, müssen weitere Tests zeigen. Die Beschleunigungsspitzen beim Versuch wurden von allen Geräten detektiert. Ebenfalls wird ersichtlich, dass das zur Polsterung genutzte Knüllpapier eine federnde und dämpfende Wirkung auf die Messgeräte besitzt, da diese nach dem Aufprall überschwingen und erst nach kurzer Zeit in Ruhe liegen. Die Einstellbarkeit zwischen ±250, ±500, ±1000 und ±2000 Grad pro Sekunde bei der Rotationsmessung und ±2 g, ±4 g, ±8 g, und ±16 g bei der Beschleunigungsmessung zur besseren Nutzung der Auflösung ist eine gute Ergänzung zu den definierten Anforderungen. Der verwendete *MPU-6050* 6-Achsen Gyro und Beschleunigungssensor der Firma *TDK InvenSense* bietet aufgrund seines Alters eingeschränkte Leistungsmerkmale im Vergleich zu neueren Sensoren. Für die Umsetzung im Rahmen dieser Masterarbeit sind jedoch andere Kriterien wichtiger als die potenzielle Leistungsfähigkeit. So stellen neben einer guten Verfügbarkeit gut gepflegte Programmbibliotheken die wichtigste Entscheidungsgrundlage für den *MPU-6050*. Die Möglichkeit, den Interrupt-Pin des Sensors an den Mikrocontroller zu koppeln, ermöglicht das Aufwachen aus dem Tiefschlafmodus durch Bewegung. Die Zeit, die der Mikrocontroller vom Erhalten des Interruptsignals bis hin zum Auslesen des Sensors benötigt, ist zu lang, um die Interrupt auslösende Beschleunigung zu messen. Für diesen Anwendungsfall müssen weitere Untersuchungen zeigen, ob dieses Problem softwareseitig behoben werden kann, oder ob Anpassungen in der Hardware nötig sind. Der benötigte Speicherbedarf einer Messung beträgt ca. 30 Byte, was bei der verwendeten 32 GB microSD-Karte mehr als 10^9 Messvorgänge ermöglicht. Die Auswertung der Messvorgänge kann, dank gängigem CSV-Format, unter anderem in einer Applikation zur Tabellenkalkulation erfolgen (z.B. Excel).
Auch wenn viele der umgesetzten Funktionalitäten des Messsystems beim Anwendungsfall der Beschleunigungsmessung durch die technischen Limitierungen nicht zum Tragen kommen, bieten die Erfahrungen aus der Umsetzung potenziellen Mehrwert für den Anwendungsfall der Transportüberwachung.

7.2. Transportüberwachung

Die Transportüberwachung stellt im Messsystem den Anwendungsfall mit dem größten Umfang an Funktionalität dar. Die Messwerte müssen zuverlässig von den Messgeräten aufgenommen und für den Datentransport vorbereitet werden. Dazu werden neben der eigentlichen Messwerterfassung von Temperatur, relativer Luftfeuchte, Luftdruck und Beschleunigungen auch der Standort des Messgerätes aufgenommen. Ebenfalls werden Metainformationen wie die Batteriespannung, Angaben zur Signalqualität oder netzwerkspezifische Informationen erfasst. Das Messgerät verschlüsselt diese Daten und überträgt diese gesichert an einen zentralen Server, welcher die Daten durch die API entschlüsselt und kontrolliert. Dazu wird eine Verbindung zu einem *LTE* oder *GPRS* Netzwerk aufgebaut. Die API, welche auf dem Server läuft, vereinheitlicht nach der Kontrolle die Daten. Anschließend werden die Daten konsolidiert in *Elasticsearch* abgespeichert. Mit dem Visualisierungstool *Kibana* lassen sich die Messwerte über eine Weboberfläche durchsuchen oder exportieren. Ebenfalls ermöglicht *Kibana*, die Messdaten zu visualisieren und diese Visualisierungen in Dashboards zu gruppieren, um die Echtzeit-Datenübertragung des Systems auszunutzen (siehe Abbildung 7.1). Mitarbeitenden des Institutes ist es so möglich, effizient Informationen aus den Messdaten zu generieren, ohne sich explizit mit der Datenhaltung beschäftigen zu müssen. Festzuhalten ist, dass die absolute Genauigkeit der durch das System ermittelten Messwerte nicht im Fokus dieser Arbeit lag. Die relative Genauigkeit der Messwerte lässt nichtsdestotrotz belastbare Rückschlüsse über die Bedingungen beim Transport der Güter zu. Zum Erreichen einer gesteigerten absoluten Genauigkeit wären wiederkehrende Kalibrierungen der Messgeräte vorausgesetzt. Die vorher definierten Anforderungen an die Erfassung von Temperatur, relative Luftfeuchte und Luftdruck sowie Beschleunigung und Rotation werden voll erfüllt. Der gewählte Bosch *BME280* Umweltsensor bietet durch die umfassende Dokumentation und gute Arbeitsbereiche (Temperatur: -40 bis +85 °C, rel. Luftfeuchte 0 - 100 %, Luftdruck 300 - 1100 hPa) eine effektive Basis für das Messsystem. Zusätzlich bietet der Sensor eine geringe Ansprechzeit von einer Sekunde für die Messung der relativen Feuchtigkeit bei einer ausreichenden Genauigkeit von ±3 %.

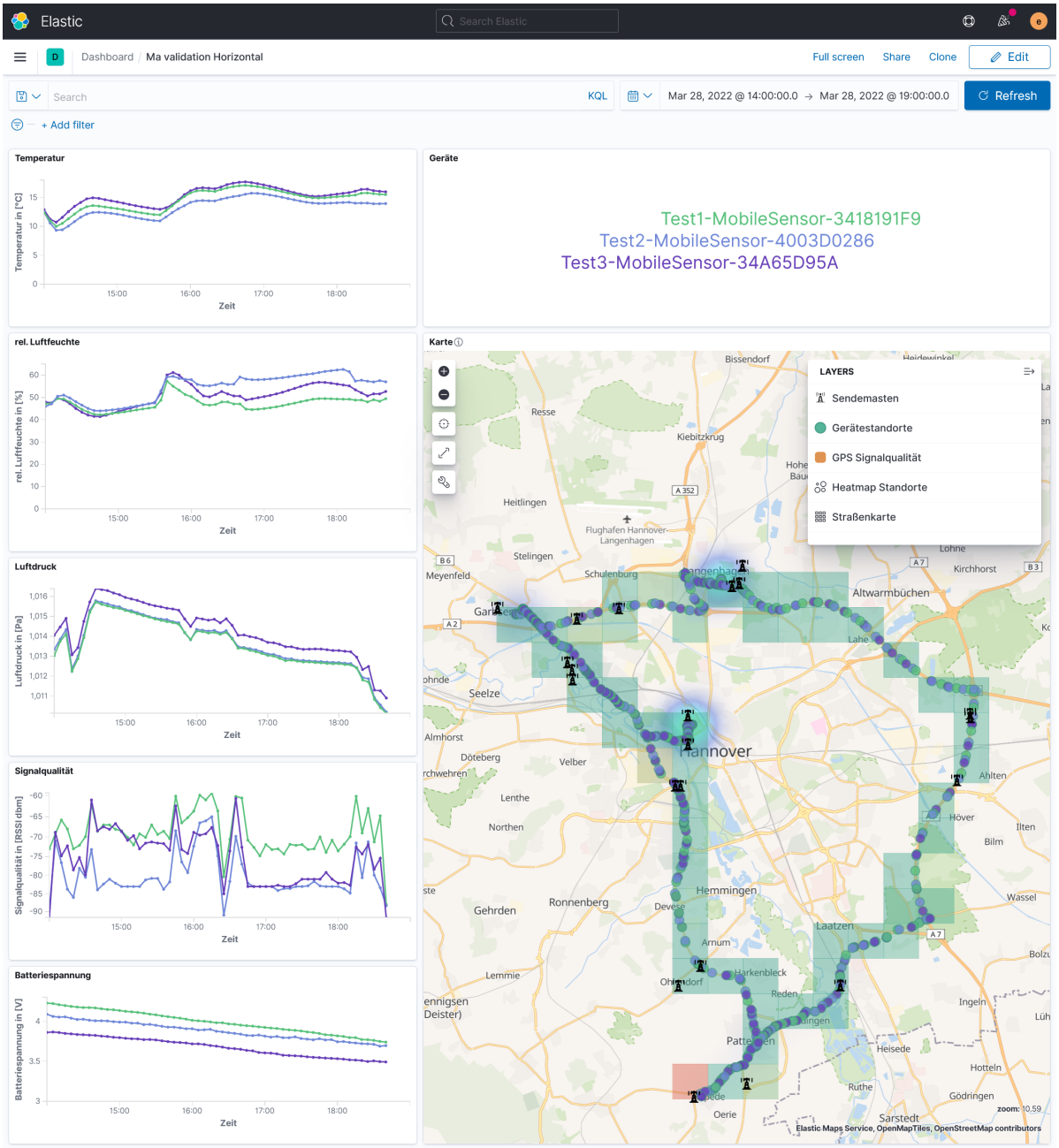


Abbildung 7.1.: *Kibana* Dashboard mit Daten des Messsystems. Dargestellt ist rechts eine Auflistung der einzelnen Messgeräte und eine Karte, welche sich automatisch an die erfassten Messdaten anpasst und neben den einzelnen Standorten auch die Standorte der Sendemasten darstellt. Auf der rechten Seiten sind Diagramme mit den Messwerten der einzelnen Geräte in Graphen dargestellt.

Die unterschiedlichen gemessenen Temperaturen der einzelnen Messgeräte werden maßgeblich durch das Fehlen einer Kalibrierung der Sensoren beeinflusst. Einflüsse durch unterschiedliche Positionen in der Kühlbox sind ebenfalls möglich. Die Nutzung von Sensoren mit der *I2C*-Schnittstelle und eine vorausschauende Dimensionierung der Sensoranschlüsse ermöglicht die Erweiterung des Systems um weitere anwendungsspezifische Sensorik, ohne neue Geräte fertigen zu müssen. Die Standortermittlung der Messgeräte erfolgte in den Tests schnell und zuverlässig. Die Genauigkeit der Standorte ist ebenfalls als gut zu bewerten. Durch die Nutzung der internationalen Mobilfunkstandards *LTE CAT-M* und *NB-IoT* können die Geräte je nach Mobilfunkanbieter auch in anderen Ländern vollumfänglich genutzt werden. Die Dämpfung, mit welcher die Mobilfunksignale von den Geräten empfangen werden, ist über den Versuch wie zu erwarten schwankend. Jedoch ist die Signalqualität im gesamten Versuchszeitraum ausreichend, um eine Kommunikation mit dem Server zu gewährleisten.

Die weit verbreitete Plattform, basierend auf dem *ESP32* Mikrocontroller der Firma *Espressif*, ermöglicht die Nutzung verschiedener, gut gepflegter Programmbibliotheken. Dadurch lassen sich einzelne Bausteine des Programmcodes ersetzen, um das System mit anderen Kommunikationsnetzwerken zu betreiben. Durch die zugrundeliegende modulare Serverarchitektur lassen sich auch bei der serverseitigen Verarbeitung und Speicherung der Daten leicht Änderungen implementieren, um das System auch mit geänderten Anforderungen zu betreiben. Durch die Übermittlung der Messdaten des Beschleunigungssensors kann mit Hilfe der Erdbeschleunigung die Lage und Orientierung des Messgerätes ermittelt werden. Somit lassen sich Rückschlüsse über die Verkippung des Transportgutes ziehen.

Die in der Arbeit gewählte Datenbank *Elasticsearch* bietet durch ihre unkomplizierte Inbetriebnahme und sehr gute Skalierbarkeit eine solide Basis zur Datenhaltung. Die dazugehörige Webapplikation *Kibana* ermöglicht auch nicht geschulten Benutzerinnen und Benutzern eine einfache Möglichkeit, die Messdaten zu visualisieren und zu exportieren. Die erstellbaren interaktiven Dashboards bieten zudem die Möglichkeit, schnell Informationen aus den Messdaten zu erhalten.

Das Zwischenspeichern der Daten auf den Messgeräten bei Verbindungsproblemen wurde ebenfalls realisiert. Die durchschnittliche Größe eines Messdatenpaketes beträgt um 3.5 KB, somit lassen sich auf einer handelsüblichen microSD-Karte mit 32 GB Speicher über neun Millionen Messpunkte zwischenspeichern. Durch die Übermittlung von Systeminformationen der Messgeräte lassen sich die zusätzliche Informationen für die Transportprozesse gewinnen. Durch die Übermittlung der Batteriespannung lässt sich ebenfalls die restliche Laufzeit abschätzen. Die Unterschiede in den Verläufen der Batteriespannung haben einerseits mit dem Ladezustand zu Beginn der Messung zu tun, andererseits ist eine unterschiedlich stark ausgeprägte Degradation der einzelnen Akkus denkbar. Die erreichten Akkulaufzeiten sind mit bis zu 4400 Messvorgängen mit einer Akkuladung bereits wettbewerbsfähig. Zudem bietet das verwendete Entwicklungsboard die Möglichkeit, über den vorhandenen Anschluss für eine Solarzelle die Laufzeit weiter zu verlängern.

Abschließend ist festzuhalten, dass die Anforderungen an die Transportüberwachung vollumfänglich erfüllt werden. Im direkten Vergleich mit käuflich zu erwerbenden Geräten zeigt sich die Leistungfähigkeit der in dieser Masterarbeit entwickelten Transportüberwachung.

8. Fazit

Die Aufgabenstellung dieser Masterarbeit war die Konzeption und prototypische Umsetzung einer mobilen Plattform zur Erfassung von Sensordaten in der Biomedizintechnik. Die Fragestellung lässt sich dahingehend auslegen, ob es im Rahmen einer Masterarbeit möglich ist, ein Messsystem zu entwickeln, zu fertigen, zu programmieren und validieren. Nach Analyse der am Markt befindlichen kaufbaren Lösungen lassen sich die Vorteile für ein in wissenschaftlicher Umgebung entwickeltes System vorallem in der offenen Hardware- und Softwareplatform verorten. Die am Markt befindlichen kommerziellen Messsysteme besitzen eine undurchsichtige Datenhaltung und wenig bis keine Flexibilität bei der Integration neuer Sensorik. Somit wird bei Benutzung dieser Systeme eine Abhängigkeit vom Anbieter geschaffen, die bei fehlender Schnittstellen zum Export der Daten nur schwer zu lösen sind. Somit lässt sich die Fragestellung weiter spezifizieren, ob sich ein leistungstechnisch mindestens gleichwertiges System mit offener Datenhaltung und offener Hardware umsetzen lässt, um langfristig schneller auf geänderte Anforderungen und neue Sensorik reagieren zu können. Das im Rahmen dieser Arbeit entwickelte Echtzeitmesssystem erfüllt alle vorab definierten Anforderungen, welche aus konkreten Anwendungsfällen abgeleitet wurden. Gut dimensionierte, hardwareseitige Schnittstellen ermöglichen das Einbinden neuer Sensorik am selben Messgerät mit wenig Aufwand. Frühzeitig definierte Schnittstellen in der Datenhaltung ermöglichen einfache Anpassungen der gemessenen Parameter, ohne tiefgreifende Änderungen an weiteren Systemkomponenten durchführen zu müssen. Ebenfalls lassen sich einzelne Systemkomponenten wie die Datenbank schnell realisieren, um geänderten Anforderungen gerecht zu werden. Die verwendete Datenbank *Elasticsearch* ermöglicht bereits, auf die strikte Definition der Daten zu verzichten und bietet dank der hervorragenden Skalierbarkeit nicht nur in der

Entwicklung eine gute Basis für die Datenhaltung. In Kombination mit der dazugehörigen Webapplikation *Kibana* lassen sich zudem die Messwerte in der Datenbank auf vielfältige Weise visualisieren, ohne der Nutzerin oder dem Nutzer tiefgreifendes Wissen über die Datenhaltung und Struktur aufzubürden. Die Kombination von Visualisierungen zu Dashboards und der Export der Daten im CSV-Format sind zudem ebenfalls gegeben.

Die Anforderungen, welche durch den zweiten Anwendungsfall der Beschleunigungsmessung an das System gestellt wurden, konnten erfüllt werden. Dabei sind sowohl Messfrequenz als auch die anpassbaren Messbereiche für Beschleunigung und Rotation hervorzuheben. Die Leistungsmerkmale des entwickelten Systems sind dabei auf einem Niveau mit den kaufbaren Lösungen. Im Bezug auf die Batterielaufzeit der Messgeräte ließ sich im Vergleich zu anderen Geräten eine überdurchschnittliche Anzahl von Messungen realisieren. Die im ersten Anwendungsfall beschriebene Transportüberwachung wurde in dem dazugehörigen Versuch vollumfänglich validiert. Die Messgeräte übermitteln nicht nur zuverlässig die Messdaten der Sensoren, sondern geben durch die Übermittlung von Metadaten wie Batteriespannung, Mobilfunkempfang und den Standorten von Mobilfunkmasten die Möglichkeit, wertvolle Informationen über Transportrouten zu gewinnen. Die Übertragung der Daten in Echtzeit ermöglicht zudem, schnell auf auftretende Probleme zu reagieren. Somit stellt das Messsystem Informationen bereit, die zur kurz- und langfristigen Optimierung vom Transport von Gütern beitragen können. Das im Rahmen dieser Arbeit entwickelte System stellt zudem eine offene Plattform dar, welche für verschiedene Anwendungsfälle adaptiert werden kann.

Abschließend ist zu sagen, dass die bisherigen Anwendungsfälle nur einen Teil des Potenzials dieses entwickelten Echtzeitmesssystems ausnutzen. Das geringe Vorwissen, welches notwendig ist, um aus diesem System einen effektiven Nutzen zu ziehen, in Kombination mit der guten Skalierbarkeit und der hohen Anpassbarkeit der Datenhaltung und Sensorik, sollten zu einer hohen Akzeptanz und langfristig zur Weiterentwicklung dieser Messplattform führen.

9. Ausblick

In diesem Kapitel werden mögliche weiterführende Entwicklungen für das Messsystem umrissen und die weiteren Schritte aufgezählt, welche dafür sorgen sollen, dass das System zu einer nachhaltigen Verbesserung der Messwerterfassung in der Biomedizintechnik beiträgt. Die Kombination der guten Skalierbarkeit und der Benutzerfreundlichkeit des Systems stellen dabei eine gute Basis für Weiterentwicklungen dar. Dabei sollen mit Hilfe des *Open Source* Ansatz, also der Offenlegung von Hardware und Programmcode, Wissenschaftlerinnen und Wissenschaftler dazu motiviert werden, das System langfristig wettbewerbsfähig zu halten.

9.1. Mögliche Anwendungsfälle

Die erreichte Leistungsfähigkeit des in dieser Arbeit entwickelten Messsystems kann mit weiteren Anwendungfällen ergänzt und erweitert werden. Dahingehend werden im Anschluss zwei mögliche Anwendungsfälle umschrieben, welche in weiterführenden Arbeiten umgesetzt werden können.

Containerüberwachung Interessant wäre ein Anwendungsfall, welcher durch die Nutzung der bereits vorhandenen Möglichkeit des Ladens über ein Photovoltaikmodul eine stark verlängerte, potenziell unbegrenzte Laufzeit ausnutzen könnte. Dabei könnte das Messsystem an einem Container befestigt werden, um autark Messdaten im Inneren zu erfassen (siehe Abb. 9.1). Dabei müssen die Größe des Solarmoduls und des verbauten Akkus so gewählt werden, dass eine Messwerterfassung bei einer bestimmten Einstrahlung von Sonnenenergie gewährleistet wird.

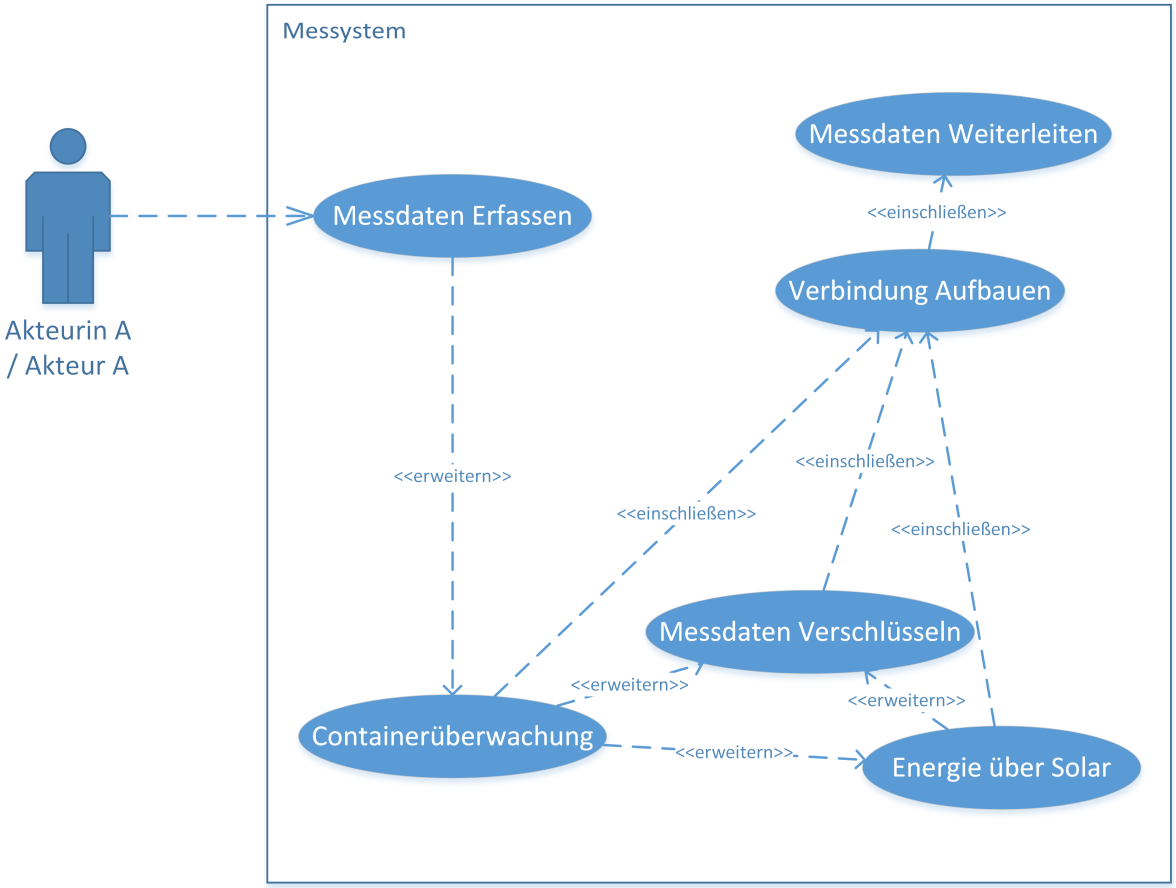


Abbildung 9.1.: Das Use Case Diagramm für die Containerüberwachung besteht aus einer Akteurin oder einem Akteur, welche oder welcher mit dem System Messdaten erfassen will. Das Messsystem kann dabei auf die Containerüberwachung gestellt werden, in welchem optional Energie über ein Solarmodul genutzt wird. Die erfassten Messdaten werden wahlweise verschlüsselt, bevor das Messsystem Verbindung mit dem Netzwerk aufbaut. Anschließend werden die Daten an einen Server weitergeleitet.

Stoßerkennung Eine mögliche Weiterentwicklung für das System ist die Implementierung einer Stoßerkennung. Dabei würde der Beschleunigungssensor kontinuierlich Messwerte aufnehmen. Idealerweise eigentständig, ohne auf den Mikrocontroller angewiesen zu sein. Somit kann der Beschelunigungssensor das restliche Messsystem aus dem Tiefschlafmodus aufwecken und in einen Modus versetzen, bei welchem für einen bestimmten Zeitraum die Beschleunigungswerte erfasst, gespeichert und im Anschluss übermittelt werden. Potenziell ließen sich so unsachgemäße Handhabung des Transportgutes oder anderweitige Ereignisse auf dem Transportweg nachweisen (siehe Abb. 9.2). Die Schwierigkeit wird voraussichtlich darin besetehen, dass der Mikrocontroller nur wenig Zeit hat, um aufzuwachen und Messdaten aufzunehmen. Im besten Fall würde der Beschleunigungssensor schon vor dem eigentlichen Stoß den Mikrocontroller aufwecken, möglicherweise bei der Detektion eines freien Falls. Dieser Anwendungfall benötigt möglicherweise Anpassungen an der Hardware der Messgeräte und wäre somit umfangreicher als der vorher beschriebene Anwendungsfall der Containerüberwachung.

9.2. Weiterentwicklung

Der modulare Aufbau des Messsystems ermöglicht die Weiterentwicklung einzelner Komponenten, ohne Vorwissen in allen Bereichen vorauszusetzen. Wissenschaftlerinnen und Wissenschaftler können so entsprechend ihrer Kompetenzen dazu beitragen, das System möglichst leistungsfähig zu gestalten. Ebenfalls relevant sind Rückmeldungen von Nutzerinnen und Nutzern des Systems, welche auf Probleme oder neue Anwendungsfälle aufmerksam machen. Um sowohl die parallele Entwicklung des Systems zu ermöglichen, als auch eine Schnittstelle zwischen Nutzerinnen und Nutzern und Entwicklerinnen und Entwicklern zu schaffen, wird die Arbeit mit dem Programmcode, Schaltplänen und Zeichnungen in einem öffentlichen *GitHub Repository* zur Verfügung gestellt.

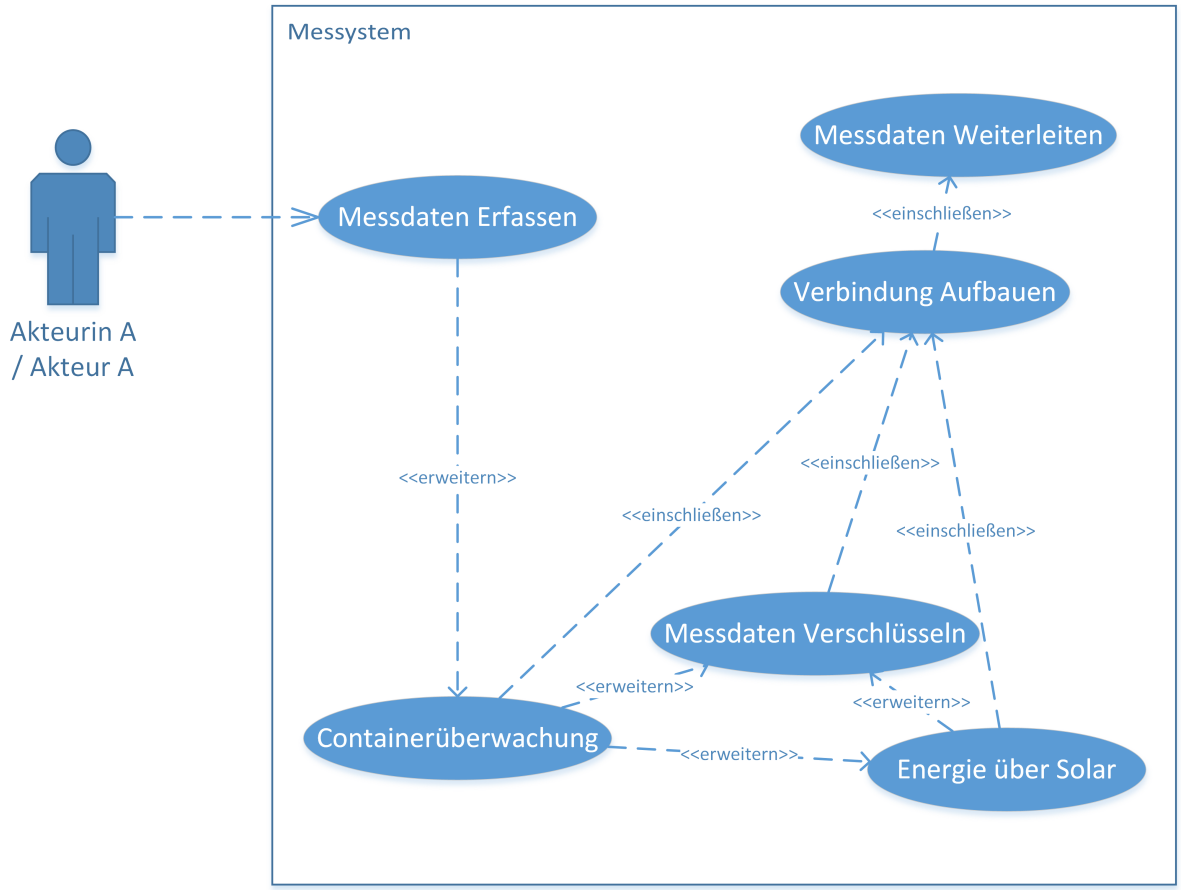


Abbildung 9.2.: Das Use Case Diagramm für die Stoßerkennung besteht aus einer Akteurin oder einem Akteur, welche oder welcher mit dem System Messdaten erfassen will. Das Messsystem kann dabei auf die Transportüberwachung gestellt werden. Optional können die Messdaten verschlüsselt werden, bevor das Messsystem Verbindung mit dem Netzwerk aufbaut. Anschließend werden die Daten an einen Server weitergeleitet. Wahlweise lässt sich eine Stoßerkennung zuschalten, welche bei der Detektion eines Stoßes für einen Zeitraum eine Beschleunigungsmessung auslöst. Am Ende der Messwerterfassung werden die Daten optional verschlüsselt an den Server weitergeleitet.

Literatur

[1] *Hapag-Lloyd stattet als erste Reederei alle Container mit Echtzeit-Tracking aus.* 2022. URL: https://www.hapag-lloyd.com/de/company/press/releases/2022/04/hapag-lloyd-first-mover-in-equipping-all-containers-with-real-ti.html. (aufgerufen: 19.05.2022).

[2] Ng, C. Z. u. a. „Cold chain time- and temperature-controlled transport of vaccines: a simulated experimental study“. eng. In: *Clinical and experimental vaccine research* 9.1 (Jan. 2020). PMC7024728[pmcid], S. 8–14. DOI: 10.7774/cevr.2020.9.1.8.

[3] Khandpur, R. „Blood Time–Temperature Indicator“. In: Dez. 2019, S. 275–278. ISBN: 9781119288121. DOI: 10.1002/9781119288190.ch50.

[4] Lusti, M. *Dateien und Datenbanken.* Springer, Berlin, Heidelberg, 1989. ISBN: 978-3-540-51035-2.

[5] Hoff, T. *How Twitter Uses Redis To Scale - 105TB RAM, 39MM QPS, 10,000+ Instances. (Englisch).* 2014. URL: highscalability.com/blog/2014/9/8/how-twitter-uses-redis-to-scale-105tb-ram-39mm-qps-10000-ins.html. (aufgerufen: 04.04.2022).

[6] Garcia-Molina, H. u. a. *DATABASE SYSTEMS The Complete Book Second Edition.* Upper Saddle River, NJ 07458: Pearson Prentice Hall, 2009. ISBN: 978-0-13-606701-6.

[7] Codd, E. F. „Extending the Database Relational Model to Capture More Meaning“. In: *ACM Transactions on Database Systems* 4 (1979), S. 397–434.

[8] Buneman, P. „Semistructured Data“. In: *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. PODS '97. Tucson, Arizona, USA: Association for Computing Machinery, 1997, S. 117–121. ISBN: 0897919106. DOI: 10.1145/263661.263675.

[9] Ajay Kulkarni, R. B. *Hapag-Lloyd stattet als erste Reederei alle Container mit Echtzeit-Tracking aus*. 2020. URL: https://www.timescale.com/blog/what-the-heck-is-time-series-data-and-why-do-i-need-a-time-series-database-dcf3b1b18563/. (aufgerufen: 19.05.2022).

[10] TH Köln, C. G. *Graphenmodell*. 2011. URL: https://wikis.gm.fh-koeln.de/Datenbanken/Graphenmodell. (aufgerufen: 13.04.2022).

[11] Holleczek, P. und Vogel-Heuser, B. *Echtzeitkommunikation und Ethernet/Internet*. Springer, Berlin, Heidelberg, 2001. ISBN: 978-3-540-42706-3.

[12] Sauter, M. *Grundkurs Mobile Kommunikationssysteme*. Springer Vieweg, Wiesbaden, 2018. ISBN: 978-3-658-21646-7.

[13] Linnemann, M. u. a. „LoRaWAN“. In: *Einsatzpotentiale von LoRaWAN in der Energiewirtschaft: Praxisbuch zu Technik, Anwendung und regulatorischen Randbedingungen*. Wiesbaden: Springer Fachmedien Wiesbaden, 2019, S. 25–39. ISBN: 978-3-658-26917-3. DOI: 10.1007/978-3-658-26917-3_3.

[14] Rudd, R. u. a. *Building Materials and Propagation. (Englisch)*. 2014. URL: ofcom.org.uk/__data/assets/pdf_file/0016/84022/building_materials_and_propagation.pdf. (aufgerufen: 04.04.2022).

[15] Zong, X. u. a. „Corrections to “Numerical Analysis of Cone/Pyramid-Shaped Antennas Using Wires/Surface-to-Surface Junction Basis” [Apr 09 1150-1157]“. In: *IEEE Transactions on Antennas and Propagation* 58.5 (2010), S. 1828–1828. DOI: 10.1109/TAP.2010.2044358.

[16] Rodriguez, I. u. a. „Radio Propagation into Modern Buildings: Attenuation Measurements in the Range from 800 MHz to 18 GHz“. In: *2014 IEEE 80th Vehicular*

Technology Conference (VTC2014-Fall). 2014, S. 1–5. DOI: 10.1109/VTCFall.2014.6966147.

[17] Reininger, P. *3GPP Standards for the Internet-of-Things. (Englisch)*. 2016. URL: 3gpp.org/images/presentations/2016_11_3gpp_Standards_for_IoT.pdf. (aufgerufen: 05.02.2022).

[18] Rochkind, M. J. „The source code control system“. In: *IEEE Transactions on Software Engineering* SE-1.4 (1975), S. 364–370. DOI: 10.1109/TSE.1975.6312866.

[19] Synopsys, I. *Compare Repositories. (Englisch)*. URL: https://www.openhub.net/repositories/compare. (aufgerufen: 14.06.2022).

[20] git. *git Git –everything-is-local. (Englisch)*. URL: https://git-scm.com/. (aufgerufen: 14.06.2022).

[21] GitHub. *About GitHub. (Englisch)*. URL: https://github.com/about. (aufgerufen: 14.06.2022).

[22] Zahedi, P. *Git ist ein System zur Versionsverwaltung von Projekten*. 2019. URL: https://praxistipps.chip.de/git-und-github-unterschiede-einfach-erklaert_110683. (aufgerufen: 14.06.2022).

[23] T. Dierks, C. A. *The TLS Protocol Version 1.0*. 1999. URL: https://datatracker.ietf.org/doc/html/rfc2246. (aufgerufen: 19.06.2022).

[24] contributors, M. *Proxy servers and tunneling*. 2022. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Proxy_servers_and_tunneling. (aufgerufen: 19.06.2022).

[25] Fielding, R. T. und Taylor, R. N. „Architectural Styles and the Design of Network-Based Software Architectures“. AAI9980887. Diss. 2000. ISBN: 0599871180.

[26] Baun, C. „Virtualisierung“. In: *Betriebssysteme kompakt: Grundlagen, Daten, Speicher, Dateien, Prozesse und Kommunikation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2020, S. 253–263. ISBN: 978-3-662-61411-2. DOI: 10.1007/978-3-662-61411-2_10.

[27] *what-is-a-hypervisor*. 2020. URL: https://www.redhat.com/de/topics/virtualization/what-is-a-hypervisor. (aufgerufen: 19.05.2022).

[28] Dr. Dietmar Müller, F. K. *grundlagen-der-container-virtualisierung*. 2016. URL: https://www.cloudcomputing-insider.de/grundlagen-der-container-virtualisierung-a-521961/. (aufgerufen: 19.05.2022).

[29] Kaschke, M. *Virtual Machine (VM) vs. Container*. 2020. URL: https://mkaschke.medium.com/virtual-machine-vm-vs-container-13ab51f4c177. (aufgerufen: 14.06.2022).

[30] Dahlqvist, F. *Growing opportunities in the Internet of Things. (Englisch)*. 2010. URL: perma.cc/8PQJ-EKJ9. (aufgerufen: 05.04.2022).

[31] *spotbot transport datenlogger*. URL: https://monitoring-shop.de/spotbot-transport-datenlogger. (aufgerufen: 19.05.2022).

[32] *spotbot cellular*. URL: https://spotsee.io/impact/spotbot-cellular. (aufgerufen: 19.05.2022).

[33] *HIGH IMPACT LABELS BY SHOCKWATCH*. URL: https://www.shockwatch.com.au/product/shockwatch-labels-impact-indicators/. (aufgerufen: 19.05.2022).

[34] *ShockWatch Label Impact Indicators Overview*. URL: https://www.shockwatch.com.au/wordpress/wp-content/uploads/2018/08/ShockWatch-Label-Impact-Indicators-Overview-SWPTY-1.pdf. (aufgerufen: 19.05.2022).

[35] *ShockLog 298 Datenblatt*. URL: https://www.shockwatch.com.au/wordpress/wp-content/uploads/2021/01/ShockLog-298-%E2%80%93-Sales-Sheet.pdf. (aufgerufen: 19.05.2022).

[36] *Warmmark Datenblatt*. URL: https://www.shockwatch.com.au/wordpress/wp-content/uploads/2020/04/WarmMark-%E2%80%93-Sales-Sheet.pdf. (aufgerufen: 19.05.2022).

[37] StaticControl. *SCS Humidity Indicator Cards and Desiccant: Product Selection Guide*. URL: http://staticcontrol.descoindustries.com/pdf/HIC_Desiccant_Product_Selection_Guide.pdf. (aufgerufen: 19.05.2022).

[38] Limited, B. *Humidity Indicator 9 Level - 15-95%*. URL: https://brownell.co.uk/9-level-15-95.html. (aufgerufen: 19.05.2022).

[39] Sensortec, B. *BME280 Data sheet*. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf. (aufgerufen: 19.05.2022).

[40] *ShockWatch Tilt Indicators*. URL: https://www.shockwatch.com.au/product-category/impact-tilt/tilt-indicators/. (aufgerufen: 19.05.2022).

[41] InvenSense. *MPU-6000/MPU-6050 Product Specification*. URL: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. (aufgerufen: 19.05.2022).

[42] ams. *TSL2561 Light-to-Digital Converter*. URL: https://ams.com/documents/20143/36005/TSL2561_DS000110_3-00.pdf/18a41097-2035-4333-c70e-bfa544c0a98b. (aufgerufen: 19.05.2022).

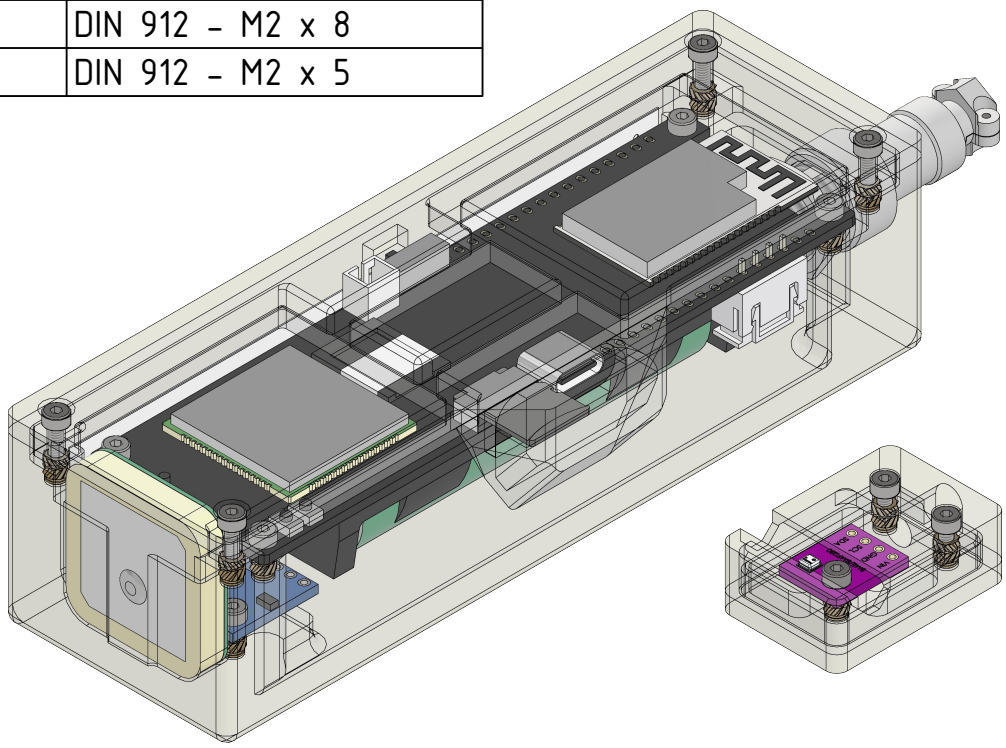
[43] Spirito, M. u. a. „Experimental performance of methods to estimate the location of legacy handsets in GSM“. In: *IEEE 54th Vehicular Technology Conference. VTC Fall 2001. Proceedings (Cat. No.01CH37211)*. Bd. 4. 2001, 2716–2720 vol.4. DOI: 10.1109/VTC.2001.957254.

[44] GmbH, B. V. *DIN EN 22248:1993-02, Verpackung; Versandfertige Packstücke; Vertikale Stoßprüfung (freier Fall) (ISO 2248:1985)*.

A. Anhang

BAUTEILLISTE		
OBJEKT	ANZAHL	BAUTEILNUMMER
1	1	T-Sim7000g
2	1	18650 Lilon-Akku
3	1	GPS-Antenne
4	1	LTE-Antenne
5	1	MPU-6050
6	1	GY-BME280
7	1	microSD Karte
8	1	GX12-Buchse
9	1	GX12-Mutter
10	1	GX12-Stecker
11	1	Gehäuse
12	1	JST-XH-4 Buchse
13	1	JST-XH-4 Stecker
14	1	Deckel
15	1	Sensor-Gehäuse
16	1	Sensor-Deckel
17	13	M2 Einschmelzgewinde
18	4	DIN 912 - M2 x 8
19	9	DIN 912 - M2 x 5

Zusätzliche Interaktionen	
Buchstabe	Name
A	An-Aus Schalter
B	nanoSIM Steckplatz
C	USB-C Buchse



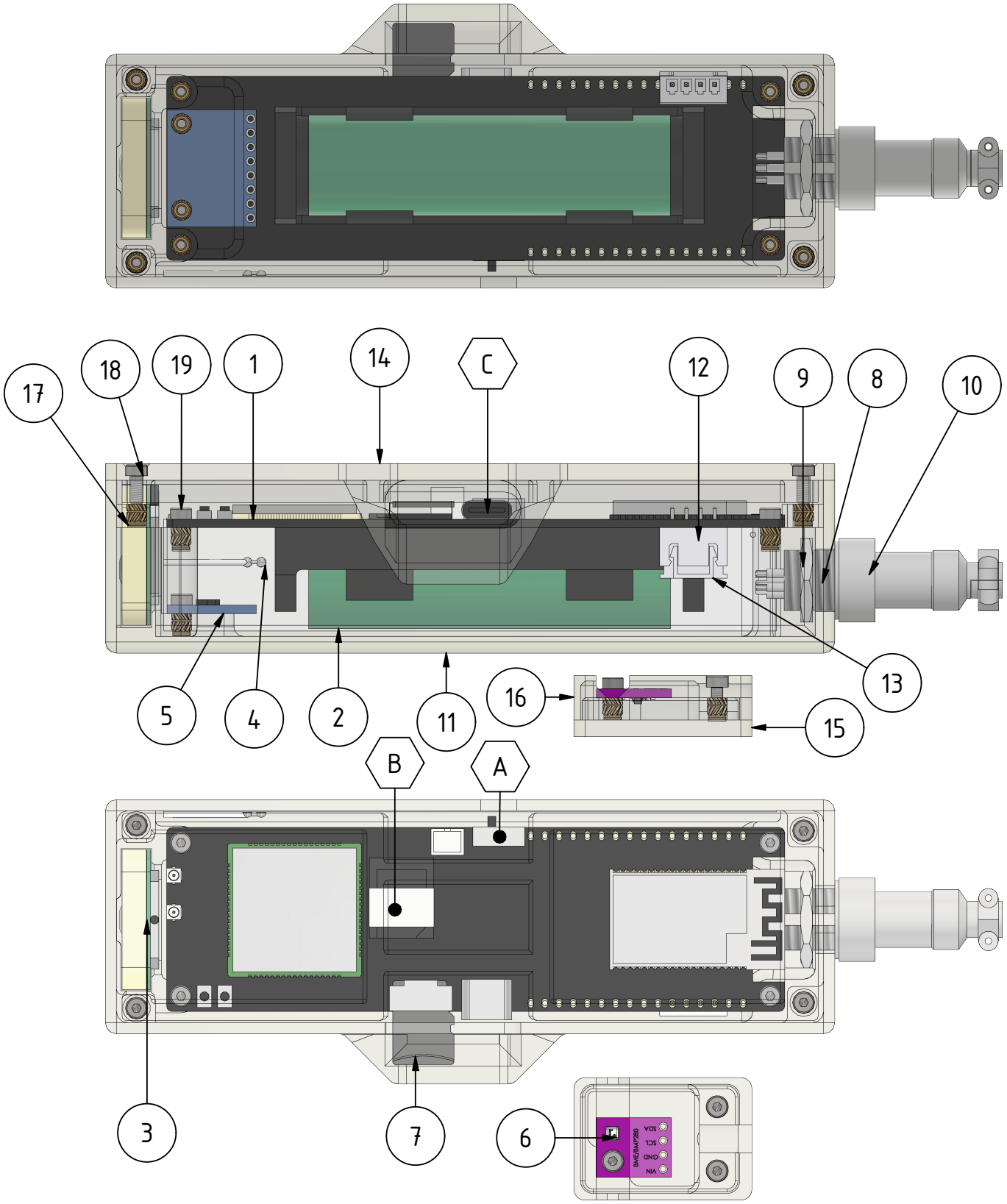
					Datum	Name
				Gezeich	27.04.2022	Lukas Tilch
				Kontroll		
				Norm		
Status	Änderungen	Datum	Name			

Messgerät

Gehäuse

1

A4



					Datum	Name
				Gezeich	27.04.2022	Lukas Tilch
				Kontroll		
				Norm		
Status	Änderungen	Datum	Name			

Messgerät

Gehäuse

2

A4

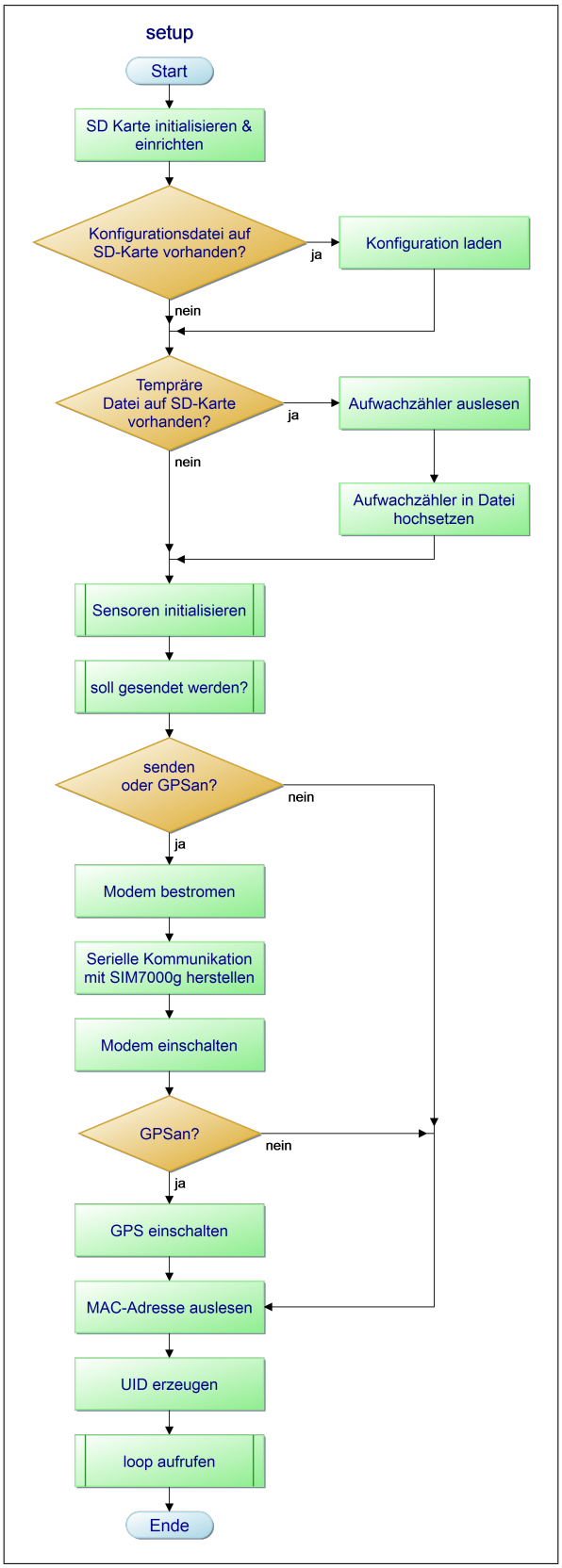


Abbildung A.1.: Programmablaufplan Setup

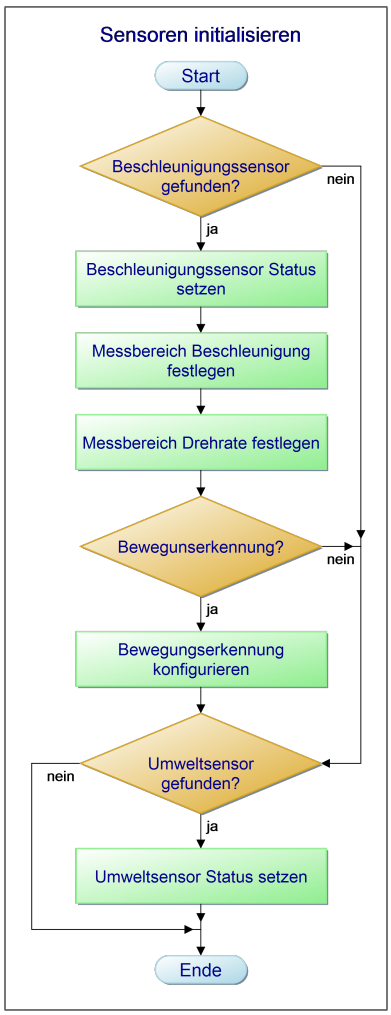


Abbildung A.2.: Programmablaufplan Sensoren initialisieren

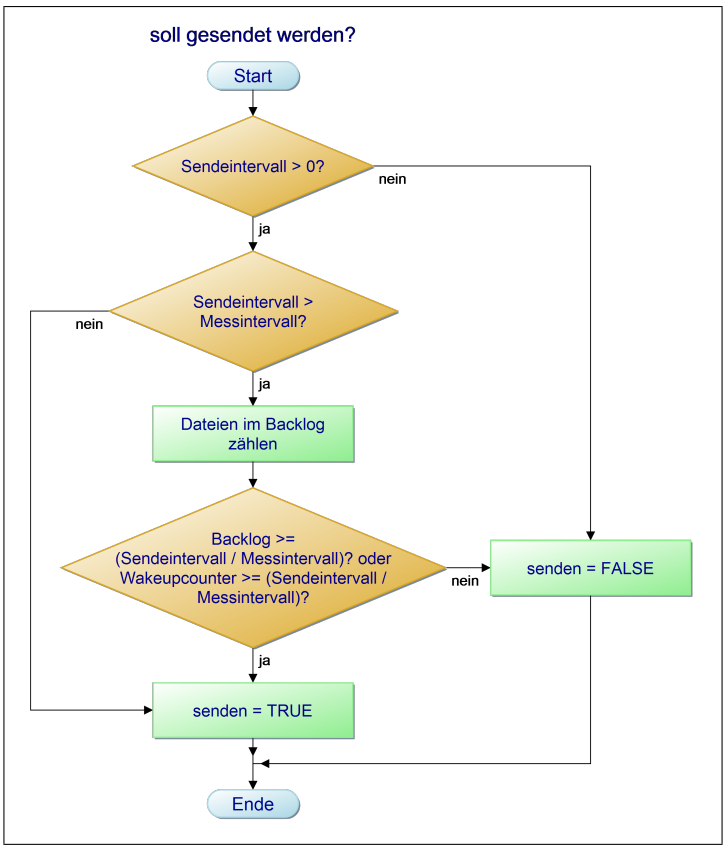


Abbildung A.3.: Programmablaufplan Senden?

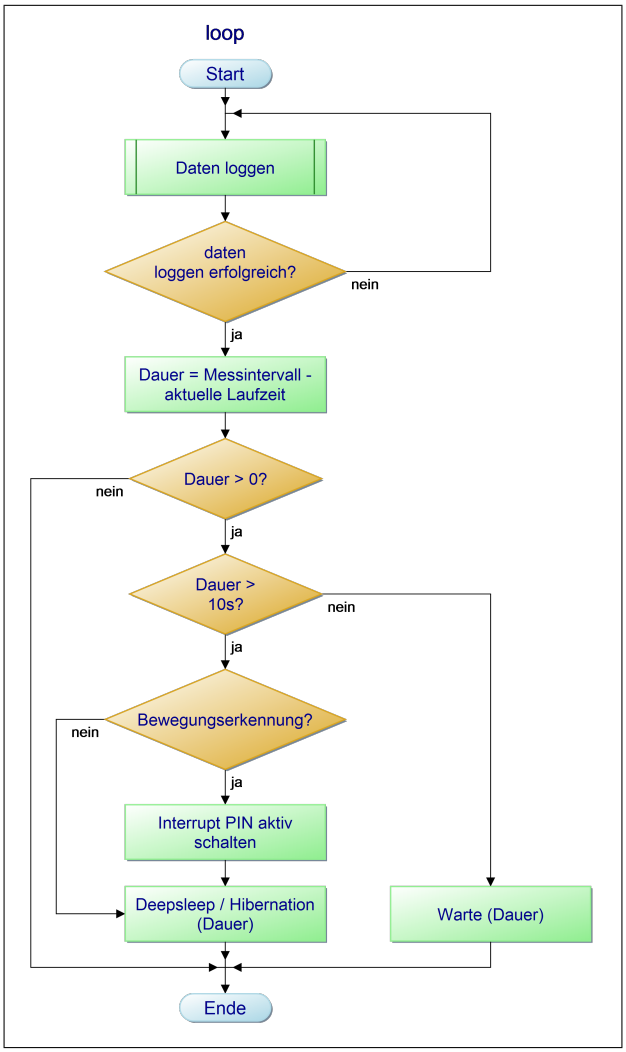


Abbildung A.4.: Programmablaufplan Loop

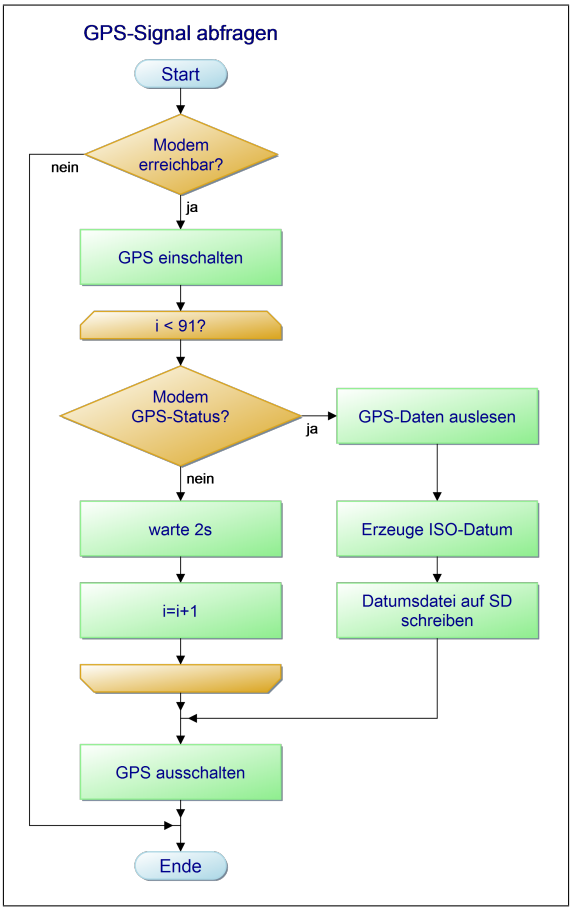


Abbildung A.5.: Programmablaufplan GPS-Signal abfragen

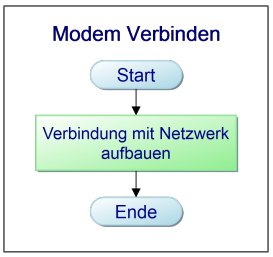


Abbildung A.6.: Programmablaufplan Modem Verbinden

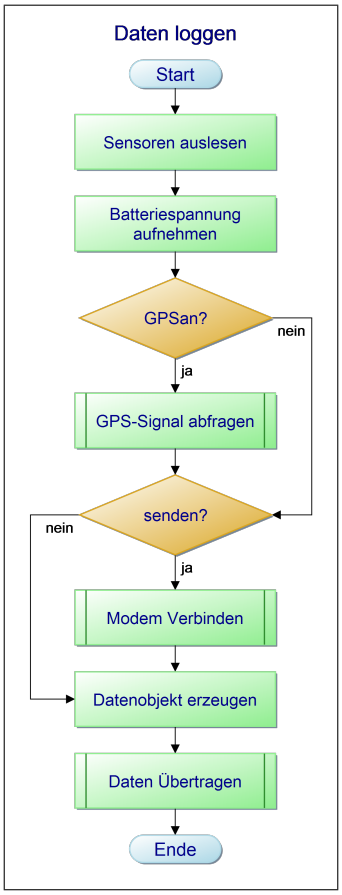


Abbildung A.7.: Programmablaufplan Programmablaufplan Daten loggen

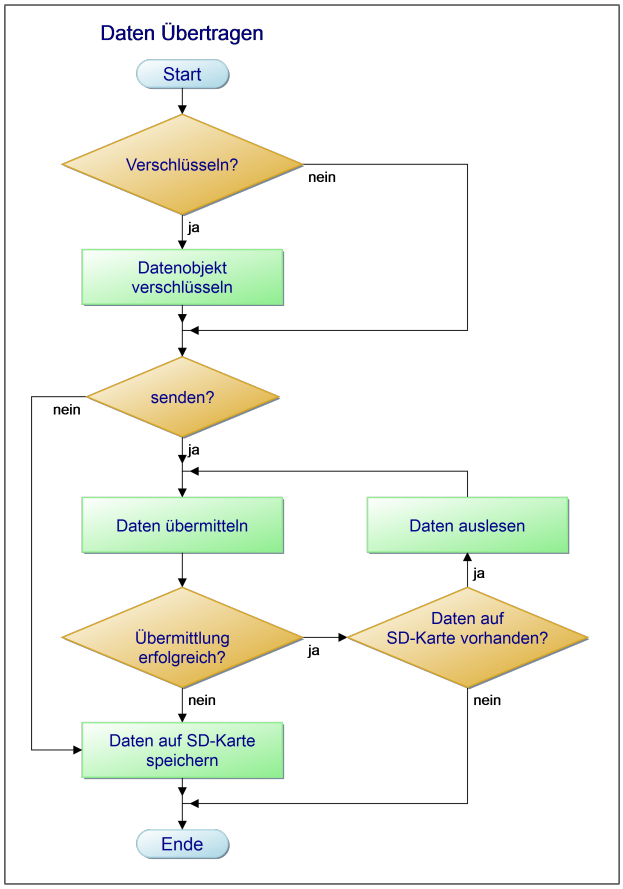


Abbildung A.8.: Programmablaufplan Daten Übertragen