Fachhochschule Westküste
Hochschule für Wirtschaft & Technik

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Richard Georg Albin Günther

Entwicklung eines Management- und Analysesystems für mobile eingebettete Datenerfassungssysteme

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Richard Georg Albin Günther

Entwicklung eines Management- und Analysesystems für mobile eingebettete Datenerfassungssysteme

Masterthesis eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Fachbereich Technik
der Fachhochschule Westküste
und
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Rasmus Rettig
Zweitgutachter : Prof. Dr.-Ing. Detlef Jensen

Abgegeben am 9. Mai 2016

Richard Georg Albin Günther

Thema der Masterthesis

Entwicklung eines Management- und Analysesystems für mobile eingebettete Datenerfassungssysteme

Stichworte

Eingebettete Systeme, Datenerfassung, Fernmanagement, Webservices, Datenverwaltung, Fahrzyklen, Nahverkehr

Kurzzusammenfassung

In dieser Masterthesis wird ein Management- und Analysesystem für mobile Datenerfassungssysteme realisiert. Das System erlaubt eine Fernwartung und -Diagnose der Datenerfassungssysteme, speichert die erfassten Messdaten zentral in einer Datenbank und stellt diese über einen Webservice zu Auswertungszwecken bereit. Im Anschluss wird das System verwendet, um eine statistische Analyse realer Verkehrsdaten von Bussen im öffentlichen Nahverkehr durchzuführen. Aus den Ergebnissen wird ein mittlerer Referenzfahrzyklus für einen Nahverkehrsbus entwickelt.

Richard Georg Albin Günther

Title of the paper

Development of a management and analysis system for mobile and embedded data acquisition systems

Keywords

embedded systems, data acquisition, remote management, web services, data management, driving cycles, public transportation

Abstract

This thesis describes the implementation of a management and analysis framework for mobile data aquisition systems. The framework provides remote management and diagnosis for the data aquisition, stores the measured data in a database and uses a web service to provide the measured data for analysis purposes. Subsequently the framework is used to perform a generic statistical analysis of real traffic information data of public transit busses. The results are used to compute an average driving cycle for such vehicles.

Inhaltsverzeichnis

1. Einführung

Die Hamburger Hochbahn AG, das zweitgrößte Nahverkehrsunternehmen Deutschlands, ist durch eine politische Vorgabe durch den Senat aufgefordert, ab 2020 nur noch emissionsfreie Busse anzuschaffen [2]. Das vom Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) geförderte Forschungsprojekt „Bewertung des Einsatzes von Elektrobussen mit Dezentraler Ladeinfrastruktur“ (BEEDeL) soll Indikatoren für die Planung eines Elektrobus-Teilnetzes in Hamburg liefern. Ziel ist die Ermittlung einer optimalen Struktur für ein Elektrobus-Teilnetz, sowie des entsprechenden Bedarfs an dezentralen Ladestationen [18]. Hierzu werden bestehende Buslinien vermessen, um deren Eignung für eine Elektrobuslinie beobachten zu können. Dies geschieht mit autonomer Messtechnik, welche die Verkehrsdaten der betrachteten Busse aufzeichnet und über das Mobilfunknetz automatisch an einen zentralen Server überträgt.

Insgesamt wurden im Laufe des Projekts seit Anfang 2015 bisher mehr als 15000 h Messdaten erhoben. Die dauerhafte Verkehrsdatenerfassung von Bussen im Linienverkehr in diesem Umfang ist eine Besonderheit. Die ersten Analyseergebnisse zeigen zudem deutlich, dass eine ungeahnte Fülle an wertvollen Informationen in den gewonnenen Messdaten steckt.

Durch die Verwendung von leistungsfähigen, eingebetteten Systemen zur Datenerfassung können Messdaten mit sehr hoher Datenrate und Auflösung erfasst werden. Auch die Übertragung von großen Datenmengen über das Mobilfunknetz ist aufgrund der sich stetig entwickelnden Mobilfunktechnik problemlos möglich. Da die Hardware-Kosten für solche Systeme gering sind, können kostengünstig sehr große Sensornetzwerke aufgebaut werden.

Die Verwaltung eines solchen Netzwerks ist jedoch nicht immer trivial. Die korrekte Funktion der Systeme kann nur dadurch erkannt werden, dass ein Upload auf den Server erfolgt ist. Fehlverhalten wird so zwar erkannt, doch muss die anschließende Wartung vor Ort durchgeführt werden. Das ist jedoch nur für eine begrenzte Anzahl an Datenerfassungssystemen effizient. Da die Systeme bereits in der Lage sind eine Verbindung zu einem Server aufzubauen, liegt es Nahe, über diesen Kanal nicht nur die Messdaten zu übertragen, sondern auch Status-Informationen, Konfigurationen oder Software-Updates. Auf diese Weise kann die Wartung und Fehleranalyse aus der Ferne durchgeführt werden und eine größere Flotte aus Datenerfassungssystemen verwaltet werden.

Ein weiterer Aspekt ist die Menge an Daten, die durch ein solches verteiltes Sensornetzwerk produziert wird. Diese Messdaten müssen organisiert und für die Auswertung zentral zur Verfügung gestellt werden. Die Datenmenge kann dabei schnell so groß werden, dass eine manuelle Auswertung kaum noch möglich ist. Je nach Größe des Netzwerkes, sind dann auch Methoden aus dem Bereich von „Big-Data“ zur automatisierten Datenauswertung notwendig.

Aufgabenstellung

In dieser Arbeit wird ein Management- und Analysesystem für mobile, autonome Datenerfassungssysteme entwickelt. Das System erlaubt Ferndiagnose und -Management dieser Systeme sowie die Verwaltung der erfassten Messdaten.

Die Datenaufnahmesysteme sind hierbei nur zu bestimmten Zeitpunkten mit einem Server verbunden. Zum Management der Systeme wird eine Erfassung und Übertragung von Statusinformationen sowie eine Anpassung der Konfiguration realisiert. Zudem besteht die Möglichkeit, die Software auf dem System aus der Ferne zu aktualisieren. Die von den Datenerfassungssystemen erzeugten Messdaten werden durch das Management- und Analysesystem verarbeitet, gespeichert und über eine standardisierte Schnittstelle anderen Anwendungen zur Verfügung gestellt.

Statusinformationen können über eine Benutzer-Schnittstelle auf dem Server eingesehen werden. Die Konfigurationen der Clients können geändert und gespeichert werden. Zusätzlich können die Messdaten über diese Schnittstelle zu Auswertungszwecken übertragen werden.

Das System wird flexibel für verschiedene Szenarien und mit unterschiedlicher Hardware eingesetzt. Auf den Clients wird daher ein einheitliches Betriebssystem verwendet, welches flexibel auf unterschiedlichen Hardware-Plattformen genutzt werden kann.

Im Rahmen dieser Arbeit wird die Funktion exemplarisch für ein Szenario umgesetzt und eine Messdatenanalyse durchgeführt. Die Analyse der Fahrzyklen von Bussen im öffentlichen Nahverkehr bilden hierfür die Basis. Es werden die Software-Komponenten für Client und Server implementiert, die ein Management- und Analysesystem umsetzen. Die erfassten Datensätze werden für die weitere statistische Analyse in ein zu definierendes Format migriert und anschließend statistisch analysiert. Abschließend wird hieraus ein bewerteter Referenz-Fahrzyklus entwickelt.

2. Grundlagen der verwendeten Technologien

2.1. Machine-to-Machine-Kommunikation (M2M)

Machine-to-Machine-Kommunikation (M2M) beschreibt allgemein den automatischen Informationsaustausch zwischen vernetzten intelligenten Geräten. Der Datenaustausch wird über ein Kommunikationsnetzwerk realisiert.

Grundsätzlich basieren M2M-Anwendungen auf drei Grundbausteinen:

1. Datenendpunkt (DEP)
2. Kommunikationsnetzwerk
3. Datenintegrationspunkt (DIP)

Abbildung 2.1.: End-to-End M2M-Anwendung [12, vgl. Abb. 3, S. 21]

Der Datenendpunkt (DEP) agiert wie ein Client der M2M-Anwendung und kann z.B. ein Datenaufnahmesystem sein. Über eine entsprechende Schnittstelle ist dieser an das Kommunikationsnetzwerk angeschlossen und kann so einen Prozess (z.B. die Datenerfassung), in eine IT-Anwendung am Datenintegrationspunkt (DIP) integrieren. Der DIP agiert demnach als Server. Die eigentlichen Funktionsendpunkte sind der Prozess A und die Anwendung B. Eine solche Anwendung wird auch als „End-to-End M2M“ bezeichnet [12, vgl. S. 20-21].

Häufig wird das Internet als Kommunikationsnetzwerk verwendet. Das birgt den Vorteil der Verwendung von standardisierten Kommunikationsprotokollen und hoher Verfügbarkeit. Weiterhin können auf diese Weise auch mobile Geräte in das Netzwerk integriert werden. So entsteht ein *Internet der Dinge*, auch *Internet of Things (IoT)* genannt [12, S. 18].

Es können beispielsweise großräumig verteilte Sensornetzwerke realisiert werden, welche über eine mobile Breitbandverbindung mit dem Internet verbunden sind. Typisch dafür sind Telemetrie-Anwendungen, wie autonome Wetterstationen, welche automatisch Wetterdaten an einen zentralen Server übertragen. Auch die Gebiete *Smart Metering* oder *Mobile Payment* sind Anwendungsfelder von M2M.

2.2. Computer-Netzwerke

2.2.1. Respresentational State Transfer (REST)

Representational State Transfer (REST) ist ein Entwurfsmuster für die Struktur eines Informationssystems und wurde von Roy Fielding in seiner Dissertation (siehe [9]) entwickelt. Dazu wählte Fielding verschiedene Eigenschaften und Randbedingungen von existierenden Netzwerkarchitekturen aus, um diese zu einer neuen Architektur zu kombinieren. REST beschreibt so eine System-Architektur zur Kommunikation in sehr großen Netzwerken. Daher wird REST auch oft als Basis für die Skalierbarkeit des WWW beschrieben.

Die grundlegenden Eigenschaften eines REST-Systems sind: [9, vgl. S. 76-84]:

- Client-Server-Architektur
- Zustandslose Kommunikation
- Verwendung von Cache
- Einheitliche Schnittstellen
- Verwendung von hierarchischen Ebenen
- Code-on-Demand
  (Herunterladen und Ausführung von zusätzlicher Software bei Bedarf)

Ein REST-System besteht aus den folgenden grundlegenden Elementen (vgl. [8, S. 52-56]):

- *Ressourcen*:
  Ressourcen sind adressierbare Endpunkte im System, welche Inhalte wie z.B. ein HTML-Dokument, ein Bild oder einen XML-Datenstrom repräsentieren. Dabei kann eine Ressource auch mehrere Repräsentationen besitzen. Sie können über einen Uniform Resource Identifier (URI) eindeutig adressiert werden, wobei eine Ressource auch unter mehreren URIs verfügbar sein kann.
- *Operationen*:
  Mit Hilfe der Operatoren kann der Nutzer des Systems mit den Ressourcen interagieren. Eine wichtige Grundeigenschaft ist dabei die zustandslose Interaktion. Die Ausführung einer Operation ist immer unabhängig von der vorherigen oder folgenden

Operation. Mit Operatoren können Ressourcen erstellt, gelöscht oder abgefragt werden.

- *Links*:
  Um aus der Ansammlung von Ressourcen ein Netzwerk zu erstellen, werden Links verwendet. Ein Link kann wie ein Zeiger interpretiert werden, an dessen Ende eine Ressource stehen kann, aber nicht muss. Links können auch Meta-Daten enthalten, z.B. auf welche Form der Repräsentationen einer Ressource der Link zeigt.

Das *Hypertext Transfer Protocol (HTTP)* implementiert diesen Ansatz und ist daher geeignet, um REST-konforme Anwendungen oder Dienste zu entwickeln [30, S. 9-11].

2.2.2. Netzwerkprotokolle

2.2.2.1. TCP/IP-Referenzmodell

Mit *TCP/IP (Transmission Control Protocol/Internet Protocol)* wird eine Familie von Protokollen beschrieben, welche für die Kommunikation im Internet verwendet werden. Zur Beschreibung wird ein Schichtenmodell verwendet, welches die unterschiedlichen Ebenen der Kommunikation beschreibt [3, S. 15]. Die vier Schichten des Modells sind in der nachfolgenden Tabelle 2.1 dargestellt.

Schicht	**Beschreibung**	**Protokolle**
Anwendungsschicht	Enthält Anwendungen, die auf das Netzwerk zugreifen	z.B. HTTP, FTP, SMTP
Transportschicht	Herstellung einer Ende-zu-Ende-Verbindung zw. Sender und Empfänger	TCP, UDP
Vermittlungsschicht	Aufbau von Datenpakten und Ermittlung der Daten-Route	Internetprotokoll (IPv4, IPv6)
Verbindungsschicht	Zugriff auf physikalische Netzwerk-Verbindung	z.B. Ethernet, PPP, UMTS

Tabelle 2.1.: TCP/IP-Schichtenmodell nach [3, S. 16]

2.2.2.2. Hypertext Transfer Protocol (HTTP)

Das *HTTP* ist ein Protokoll zur Datenübertragung, insbesondere im *World Wide Web (WWW)*. Es ist der Anwendungsschicht des TCP/IP-Schichtenmodells zuzuordnen und verwendet TCP auf der Transportschicht. Es handelt sich um ein zustandsloses Protokoll, denn

jede Anfrage eines Clients ist unabhängig von vorherigen Anfragen desselben oder anderer Clients. Um Ressourcen im Netzwerk eindeutig zu adressieren, verwendet HTTP eine standardisierte Adresse (*Uniform Resource Locator (URL)*) [3, S. 161-162].

Die aktuelle am häufigsten verwendete Version ist HTTP/1.1, welche im RFC 2616[1] der IETF spezifiziert ist. Im Mai 2015 wurde die Spezifikation der Version HTTP/2 im RFC 7540[2] veröffentlicht.

HTTP-Nachrichten

Es werden zwei unterschiedliche HTTP-Nachrichtentypen unterschieden. Der Client sendet eine Anfrage (*HTTP-Request*), auf die der Server mit einer *HTTP-Response* antwortet. Die Strukturen der Nachrichten ist in der folgenden Abbildung 2.2 dargestellt.

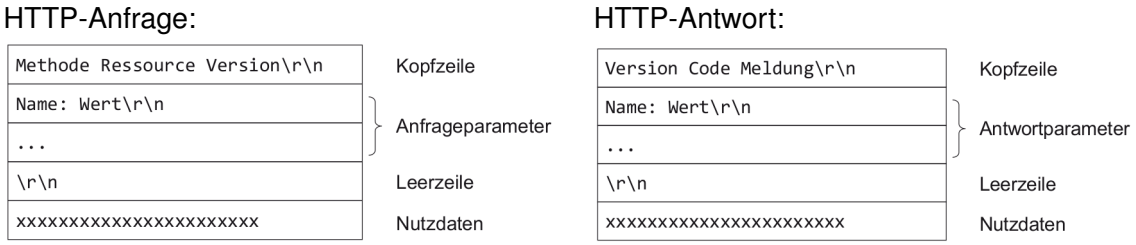

Abbildung 2.2.: Strukturen der HTTP-Nachrichten [3, S. 165,169]

Eine HTTP-Anfrage besteht aus einer Kopfzeile, welcher die Anfrage-Methode, den Namen der Ressource, sowie die verwendete HTTP-Version enthält. Es folgt ein optionaler Anfrage-Parametersatz, der z.B. den Server-Namen und weitere Informationen über die Anfrage enthält. Ebenfalls optional ist ein Nutzdatenteil, in dem Informationen an den Server gesendet werden können. (z.B. Daten eines Formulars)

HTTP stellt eine Reihe von möglichen Anfrage-Methoden bereit, wobei die Methoden „GET“ und „POST“ am häufigsten eingesetzt werden:

- GET
  Fordert eine Ressource an (z.B. eine HTML-Datei)
- POST
  Überträgt Nutzdaten an den Server (z.B. eine Text-Datei)

[1] https://tools.ietf.org/html/rfc2616
[2] https://tools.ietf.org/html/rfc7540

Weitere Methoden wie „PUT“ oder „DELETE“, werden von vielen Webservern aus Sicherheitsgründen nicht verarbeitet. Sollen Ressourcen gelöscht oder erzeugt werden, so wird dies ebenfalls mit Hilfe der „POST“-Methode durchgeführt.

Die HTTP-Antwort besteht ebenfalls aus einer Kopfzeile, welche die verwendete Protokoll-Version, einen Status-Code und eine optionale Status-Meldung enthält. Es folgen Antwortparameter, die zusätzliche Informationen über den Server und die Antwort bereitstellen. Der Nutzdatenteil enthält die vom Client angeforderte Ressource (z.B. eine HTML-Datei). Mit Hilfe des Status-Codes teilt der Server dem Client mit, wie die Anfrage bearbeitet wurde.

Beispiel einer HTTP-Kommunikation

HTTP-Anfrage:

```
GET /dlms/index.php HTTP/1.1
Host: urbandevelopment.gnet.haw-hamburg.de
User-Agent: curl/7.43.0
```

Listing 2.1: Beispiel einer HTTP-Anfrage

HTTP-Antwort:

```
HTTP/1.1 200 OK
Date: Tue, 12 Apr 2016 10:46:15 GMT
Server: Apache/2.4.7 (Ubuntu)
...
Content-Length: 3649
Content-Type: text/html; charset=UTF-8

<html> ... </html>
```

Listing 2.2: Beispiel einer HTTP-Antwort

2.2.3. Virtual Private Network (VPN)

Mit Hilfe eines Virtual Private Network (VPN) lassen sich räumlich getrennte Netzwerke miteinander verbinden (*Side-to-Side*) oder Einzelgeräte in ein entferntes Netzwerk integrieren (*End-to-Side*). Ein Anwendungsfall hierfür ist der Zugriff von Mitarbeitern im Außendienst auf Daten im firmeninternen Netzwerk [35].

Als Medium für die Verbindung zwischen den Teilnehmern wird das Internet verwendet, wobei die Datenübertragung durch einen verschlüsselten Kanal geleitet wird, sodass die Kommunikation nicht öffentlich ist.

Für den Aufbau eines VPN ist ein VPN-Server und ein VPN-Client notwendig. Der VPN-Server muss über das Internet erreichbar sein und eine gültige Internet-IP-Adresse besitzen. Der VPN-Client verbindet sich mit dem VPN-Server, sodass beide Teilnehmer ein virtuelles Netzwerk bilden. In diesem Netzwerk werden eindeutige IP-Adressen vergeben, welche nur im VPN gültig sind. Durch die entsprechende Konfiguration der *Firewall* des VPN-Servers, kann den Teilnehmern im VPN-Netzwerk Zugriff auf andere Netzwerke erteilt werden.

Die wichtigsten VPN-Protokolle sind PPTP, IPSec mit L2TP und OpenVPN. Alle Protokolle verwenden unterschiedliche Ports und Verschlüsselungstechnologien und sind untereinander inkompatibel [33].

2.3. Datenbank-Technologien

2.3.1. Relationale Datenbanken

Das Modell der relationalen Datenbank wurde 1970 von Edgar Frank Codd vorgestellt (siehe [6]) und ist eine der am häufigsten verwendeten Datenbank-Technologien.

2.3.1.1. Struktur

Relationale Datenbanken bestehen ausschließlich aus zweidimensionalen Tabellen, auch *Relationen* genannt, welche die Daten enthalten. Die einzelnen Spalten der Tabellen werden als *Attribute* bezeichnet. Diese haben einen festgelegten Wertebereich und es können Operationen auf Attribute angewendet werden. Die Zeilen einer Tabelle werden *Tupel* genannt und enthalten die tatsächlichen Werte der Relation [22, S. 55ff].

2.3.1.2. Schlüssel

Um die Datenintegrität zu gewährleisten werden verschiedene Schlüssel verwendet. Es werden *Primärschlüssel*, *Fremdschlüssel* und *Kandidatenschlüssel* unterschieden.

Mit einem Kandidatenschlüssel kann jedes Tupel einer Relation eindeutig identifiziert werden. Der Schlüssel wird durch ein einzelnes Attribut oder eine Menge von Attributen repräsentiert, wobei es für eine Relation auch mehrere Kandidatenschlüssel geben kann. In diesem Fall muss einer der Schlüssel als Primärschlüssel ausgewählt werden. Ein Fremdschlüssel ist ein Attribut, welches Primärschlüssel ist und auch in anderen Tabellen vorkommt. Der Fremdschlüssel dient dazu, Beziehungen zwischen den Tabellen aufzubauen

und wird häufig in Form einer eindeutigen Nummer (ID) repräsentiert. Besitzen zwei Tabellen verschiedene Informationen über ein Objekt, so können diese Informationen über ein gemeinsames Attribut (den Fremdschlüssel) verknüpft werden.

2.3.1.3. Beziehungen

Die Tabellen können auf unterschiedliche Arten miteinander in Beziehung gesetzt werden. Es wird dabei immer der Assoziationstyp einer Relation A zu einer Relation B betrachtet. Die Tabelle 2.2 zeigt die möglichen Assoziationstypen von Tabelle A zu Tabelle B.

Abkürzung	Assoziationstyp	Anzahl Tupel in Relation B
1	einfach	genau ein Tupel (1)
c	konditionell	kein oder genau ein Tupel (0/1)
m	multipel	mindestens ein Tupel (>=1)
mc	multipel-konditionell	beliebig viele Tupel (>=0)

Tabelle 2.2.: Assoziationstypen für die Verknüpfung von Tabellen nach [29, S. 18]

2.3.1.4. Datenabfrage

Um auf die Daten in einer relationalen Datenbank zuzugreifen, wird Structured Query Language (SQL) verwendet. Daher werden relationale Datenbank-Systeme auch häufig als „SQL“-Datenbanken bezeichnet. Die aktuelle Revision von SQL ist in ISO/IEC 9075:2011 spezifiziert.

Mit Hilfe von SQL kann eine Datenbank sehr flexibel abgefragt werden. Grundsätzlich gilt, dass immer nur ein SQL-Befehl benötigt wird, um einen bestimmten Datensatz zu erhalten. Daher können in einem Befehl einzelne oder mehrere Attribute aus Tabellen ausgelesen, gefiltert und sortiert werden. Auch die Verknüpfung von Tabellen (Joins) ist innerhalb eines Befehls möglich.

Die wichtigsten Elemente einer SQL-Abfrage werden anhand der folgenden einfachen Beispiel-Abfrage erläutert:

```
SELECT time, gps_lat, gps_lon
FROM gps_data_table
WHERE time > 2016-02-01 AND
WHERE time < 2016-02-05
ORDER BY Time DESC
LIMIT 0, 30;
```

Listing 2.3: Bespiel einer SQL-Abfrage

- SELECT:
  Es werden die Attribute „time“, „gps_lat“ und „gps_lon“ abgefragt.
- FROM:
  Die Tabelle „gps_data_table“ soll abgefragt.
- WHERE (optional):
  Filtert die Ergebnismenge. Es sollen nur Tupel zurückgegeben werden, bei denen das „time“-Attribut einen Wert zwischen „2016-02-01“ und „2016-02-05“ besitzt.
- ORDER BY (optional):
  Die Ergebnismenge soll nach dem Attribut „time“ absteigend sortiert werden.
- LIMIT (optional):
  Es sollen lediglich ein Teil der Ergebnismenge zurückgegeben werden (die ersten 30 Ergebnisse)

3. Analyse

In diesem Kapitel wird ein typisches Einsatzszenario für die Verwendung eines Management- und Analysesystems für mobile Datenlogger beschrieben. Dabei wird auf die Aufgaben eines solchen Systems eingegangen und es werden die Anforderungen an die verschiedenen Komponenten ermittelt.

3.1. Szenario: Verwaltung von autonomen Datenloggern in Nahverkerhrsbussen

Die im Forschungsprojekt „BEEDeL“ verwendeten Datenlogger zur Aufzeichnung realer Verkehrsdaten, werden fest in den Busse installiert und arbeiten autonom. Insgesamt sind bis zu 12 Datenlogger gleichzeitig im Dauereinsatz. Die Systeme zeichnen GPS-Position, Geschwindigkeit und Tür-Signale von unterschiedlichen Bus-Typen auf. Über ein UMTS-Modem stellt der Datenlogger eine temporäre Verbindung zu einem zentralen Server her, auf den die aufgezeichneten Daten hochgeladen werden. Von dort werden die Daten von Mitarbeitern der HAW Hamburg heruntergeladen, geprüft und an das Fraunhofer-Institut für Verkehrs- und Infrastruktursystem (IVI) zur Analyse weitergeleitet.

Da die Datenlogger in den Bussen fest verbaut sind, ist die Wartung aufwändig, da sie nur außerhalb der Betriebszeiten des Busses erfolgen kann. Die betroffenen Busse müssen daher für eine Wartung der Datenlogger gezielt umdisponiert werden, was einen zusätzlichen Verwaltungsaufwand erzeugt. Bei einer geringen Anzahl an Datenloggern (<10) ist diese Art der Wartung möglich. Soll jedoch die Flotte der Datenlogger vergrößert werden, müssen weitere Maßnahmen ergriffen werden.

Mit Hilfe eines Management- und Analysesystems kann die Verwaltung, Wartung und Messdatenerfassung stark vereinfacht werden. Während der Verbindung zum Server können die Datenlogger nicht nur Messdaten, sondern auch Status-Informationen, Konfigurationen oder Software-Updates bidirektional übertragen. Bei bestehender Verbindung zum Server könnte sogar ein direkter Fern-Zugriff auf die Datenerfassungssysteme für Wartungszwecke erfolgen.

Durch eine standardisierte Schnittstelle zur zentralen Messdatenspeicherung wird weiterhin der Zugang zu den Messdaten, sowie deren Analyse vereinfacht. So sind z.B. auch regelmäßige, automatische Analysen auf dem Server möglich.

3.2. Aufgaben des Systems

Nachfolgend werden die einzelnen Aufgaben eines Management- und Analysesystems kurz erläutert. Die Abbildung 3.1 zeigt die Funktionen des Gesamtsystems.

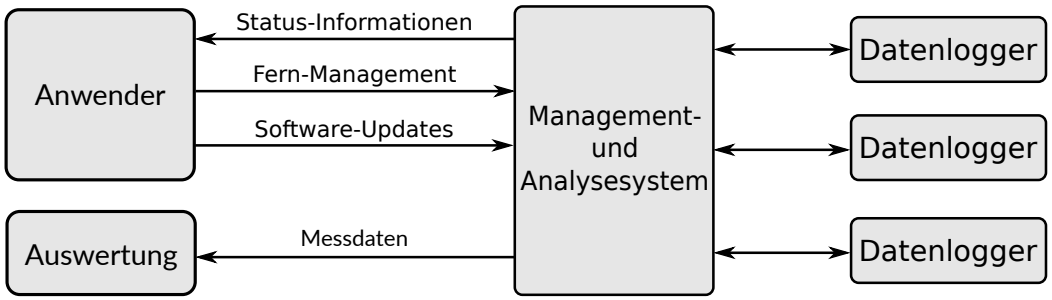

Abbildung 3.1.: Übersicht der System-Funktionen

3.2.1. Management-Funktionen

- **Erfassung von Statusinformationen der Clients**:

  Das System soll Statusinformationen über jeden Client sammeln und speichern. Damit hat der Anwender einen Überblick über die aktiven Datenlogger und deren Verhalten im Feld. Mit Hilfe der Status-Informationen können mögliche Probleme frühzeitig erkannt und Fehlerursachen schneller diagnostiziert werden. Durch Speicherung aller Status-Meldungen des Clients können auch Probleme aus der Vergangenheit analysiert und evtl. Muster erkannt werden, die zu Fehlverhalten führen.

- **Bereitstellung von Konfigurationen für Clients**:

  Um das Verhalten der Datenlogger beeinflussen zu können, soll mit Hilfe des Systems auch die Konfiguration eines Clients verändert werden können. Die Konfiguration enthält Details über die Verbindung zum Server, einer eindeutige Identifizierungsnummer, sowie die zu verwendenden Programme für die Messdatenerfassung und Messdatenverarbeitung. Ebenfalls enthalten sind Informationen über die Zeitpunkte, zu denen sich der Datenlogger mit dem Server verbindet und die entsprechenden Management-Funktionen durchführt.

- **Bereitstellung von Software-Updates**

  Während des Betriebs können Probleme auftreten, welche nicht durch die Anpassung der Konfiguration korrigiert werden können. Um dennoch reagieren zu können, soll das System die Installation neuer Software oder von Software-Updates auf den Clients ermöglichen. Auf diese Weise können auch Teile des Betriebssystems aktualisiert werden, was insbesondere für sicherheitskritische Pakete des Betriebssystems von enormer Wichtigkeit ist.

3.2.2. Analyse-Funktionen

- **Speicherung der Messdaten**

  Die von den Datenerfassungssystemen aufgezeichneten Messdaten sollen vom System gespeichert und verwaltet werden. Dazu gehört die Verarbeitung der Datensätze, die vom Client hochgeladen werden. Diese werden in den Datenspeicher integriert.

- **Bereitstellung der Messdaten**

  Die Messdaten aus dem Datenspeicher sollen zur externen Analyse zur Verfügung gestellt werden. Dabei soll ein standardisiertes, maschinenlesbares Datenformat verwendet werden, welches die automatisierte Analyse der Messdaten ermöglicht. Über diese Schnittstelle sollen auch Teil-Datensätze zur weiteren Auswertung angefordert werden können.

3.3. Anforderungsanalyse

Die Anforderungen an die einzelnen Teilsysteme werden im folgenden analysiert und, wenn möglich, quantifiziert. Dabei werden Server und Client separat betrachtet.

3.3.1. Anforderungen an den Server

Die Speicherung der Daten auf dem Server ist von der Entwicklung der Server-Applikation unabhängig. Daher werden die Anforderungen an den Server in die Teilbereiche Server-Applikation und Datenspeicherung unterteilt.

3.3.1.1. Datenspeicherung

Zunächst wird ermittelt, welche Datenmengen vom Datenspeicher bewältigt werden müssen. Die Datenlogger werden als Maß für die anfallenden Datenmengen verwendet. Es werden Annahmen für die durchschnittliche Messdauer pro Tag verwendet.

Die Messdaten werden mit einer Frequenz von 10 Hz aufgezeichnet [32, S. 21, Tab. 3.1], die aufgezeichnete Datenmenge beträgt 8898 kB pro Stunde [32, S. 69, Tab. 5.12].

Die Berechnung der zu erwarteten Datenmengen und Anzahl der Datenpunkte für einen Datenlogger ist den Gleichungen 3.1 bis 3.4 zu entnehmen. Es wird dazu eine durchschnittliche tägliche Messdauer von 8 h pro Tag und einer Gesamteinsatzdauer von 180 Tagen festgelegt.

$$\begin{aligned} Datenpunkt &= \frac{Datenmenge\ pro\ Stunde}{3600\ s} \cdot \frac{1}{10\ Hz} \\ &= \frac{8898\ kB}{3600\ s}\frac{1}{10\frac{1}{s}} = 247\ B \approx \underline{250\ B} \end{aligned} \tag{3.1}$$

$$\begin{aligned} Datenpunkte\ pro\ Tag\ &= Messdauer\ pro\ Tag\ \cdot Messfrequenz \\ &= 8 \cdot 60 \cdot 60 \cdot 10\ \frac{1}{s} = \underline{288.000} \end{aligned} \tag{3.2}$$

$$\begin{aligned} Anzahl\ Datenpunkte &= Datenpunkte\ pro\ Tag \cdot Messtage \\ &= 288000 \cdot 180\ = \underline{51.840.000} \end{aligned} \tag{3.3}$$

$$\begin{aligned} Datenmenge &= Anzahl\ Datenpunkte \cdot Datenpunkt \\ &= 51.840.000 \cdot 250\ B = \underline{12,96\ GB} \end{aligned} \tag{3.4}$$

Die Tabelle 3.1 zeigt die anfallenden Datenmengen, sowie die Anzahl der zu speichernden Messdatenpunkte für verschiedene Gesamteinsatzdauern von 100 Datenloggern mit einer täglichen Messzeit von 8 h.

Anzahl Messtage	Datenmenge	Anzahl Datenpunkte
1	7,2 GB	28.800.000
7	50,4 GB	201.600.000
30	216,0 GB	864.000.000
90	648,0 GB	2.592.000.000
180	**1,296 TB**	**5.184.000.000**

Tabelle 3.1.: Datenmengen für unterschiedliche Gesamteinsatzdauern (100 Datenlogger)

Die Tabelle 3.1 zeigt, dass der Datenspeicher über eine Kapazität von mindestens 1,296 TB verfügen muss, um 100 Datenlogger für 180 Tage zu verwalten. Der Datenzuwachs pro Tag beträgt dabei 7,2 GB, die vom Datenspeicher eingelesen werden müssen. Weiterhin wird angenommen, dass pro Tag die Messdaten von mindestens einer Woche (50,4 GB) aus dem Speicher ausgelesen werden können.

Um die Messdaten effizient auszuwerten, muss der Datenspeicher in der Lage sein, Teilergebnisse bereitzustellen. Es soll möglich sein, einen Teil-Datensatz anzufragen, in dem einzelne Messgrößen selektiert und im Wertebereich eingeschränkt werden können.

Für Datenspeicher werden außerdem die grundlegenden Eigenschaften *Atomicity, Consitency, Isolation, Durability (ACID)* gefordert, die häufig für Datenbanksysteme oder verteilte Systeme gewünscht sind. Diese Eigenschaften sind die Voraussetzung für die Verlässlichkeit des Systems. Nach jeder Interaktion mit den Daten müssen diese einen konsistenten Zustand aufweisen (*Consistency*) und dauerhaft gespeichert werden (*Durability*). Weiterhin ist gefordert, dass eine Sequenz von Daten-Operationen nur als vollständige Sequenz durchgeführt wird (*Atomicity*) und sich gleichzeitige Interaktionen nicht gegenseitig beeinflussen (*Isolation*).

Die Tabelle 3.2 fasst die Anforderungen an den Datenspeicher zusammen.

Eigenschaft	**Anforderung**
Zuverlässigkeit	• Erhaltung der Datenkonsistenz • Abgeschlossenheit der Daten-Operationen • Isolation der Daten-Operationen • Dauerhafte Speicherung der Daten
Kapazität	• min. 1,296 TB Speicher
Leistung	• Einlesen von 7,2 GB Messdaten pro Tag • Auslesen von 50,4 GB Messdaten pro Tag (Annahme)
Datenabfrage	• Selektion der auszugebenden Messgrößen • Filterung der auszugebenden Messgrößen nach Wertebereichen • Ausgabe von Datensätzen variabler Größe

Tabelle 3.2.: Anforderungen an die Datenspeicherung auf dem Server

3.3.1.2. Server-Applikation

Die Server-Applikation regelt die Kommunikation zwischen Client, Datenspeicher und Anwender und ist direkt mit dem Datenspeicher verbunden. Es werden Schnittstellen für Clients und Anwender implementiert, sodass diese Daten mit dem Datenspeicher austauschen können.

Client-Schnittstellen: Die Server-Applikation verarbeitet die Status-Meldungen und Messdatensätze der Clients. Zusätzlich werden Konfigurationen und Software-Updates den Clients zur Verfügung gestellt. Dabei soll ein standardisiertes Datenformat verwendet werden.

Auswertungsschnittstellen: Über ein standardisiertes und maschinenlesbares Datenformat sollen Messdaten, Status-Informationen und Konfigurationen zu Analysezwecken zur Verfügung gestellt werden. Diese Schnittstellen sollen von Analyse-Software genutzt werden, um automatisierte Datenauswertungen durchzuführen.

Benutzeroberfläche: Der Anwender hat über die grafische Benutzeroberfläche Zugriff auf die Verwaltungsfunktionen. Eine Übersicht über alle registrierten Clients soll helfen, den Überblick über die Datenlogger-Flotte zu behalten. Zusätzlich können auch Status-Informationen eingesehen und spezifischere Messdatensätze angefordert werden. Über eine Maske ist es möglich, die Konfigurationen der Clients anzupassen. Der Zugang zur Benutzeroberfläche soll durch eine Benutzer-Verwaltung geregelt werden. Dabei soll es verschiedene Benutzer-Gruppen geben, die unterschiedliche Zugriffsrechte für die vorhandenen Funktionen besitzen.

Komponente	**Anforderung**
Client-Schnittstellen	• Standardisiertes Datenformat • Verarbeitung von Status-Nachrichten • Verarbeitung von Messdatensätzen • Bereitstellung von Konfigurationen für Clients • Bereitstellung von Software-Updates für Clients
Auswertungs-Schnittstellen	• Standardisiertes Format zum Datenaustausch • Bereitstellung von Messdaten • Bereitstellung von Status-Informationen • Bereitstellung von Konfigurationen
Benutzer-Oberfläche	• Grafische Oberfläche • Übersicht aller registrierten Clients • Ansicht von Konfigurationen und Status-Meldungen der Clients • Bearbeitung der Konfiguration eines Clients • Anforderung von Messdaten aus dem Datenspeicher • Benutzer-Verwaltung mit Benutzer-Gruppen

Tabelle 3.3.: Anforderungen an die Applikation auf dem Server

3.3.2. Anforderungen an den Client

Damit die Clients durch das System verwaltet werden können, müssen die folgenden Anforderungen erfüllt sein. Es werden die zwei Komponenten Betriebssystem und Client-Applikation unterschieden.

Betriebssystem: Es soll ein Linux-basiertes System verwendet werden, welches nur die Software-Pakete enthält, die für den Betrieb des Clients benötigt werden. Durch das Betriebssystem werden die Schnittstellen für den Hardware-Zugriff bereitgestellt und die Laufzeitumgebung für die Client-Applikation zur Verfügung gestellt. Es muss zusätzlich gewährleistet sein, dass ein Internet-Zugang über eine UMTS-Verbindung hergestellt werden kann. Mit einem Paket-Manager sollen auf dem Client-Betriebssystem die Software-Pakete verwaltet und aktualisiert werden können.

Management-Client-Applikation: Die Client-Applikation interagiert mit dem Server und realisiert auf diese Weise das Fern-Management. Dazu zählt der Upload von Status-Meldungen, sowie der Download von Konfigurationen und Software-Updates über den Server. Die Messdatenerfassung wird durch die Client-Applikation gestartet und überwacht. Ist die Datenerfassung beendet, werden die Messdaten in ein standardisiertes Format konvertiert und auf den Server hochgeladen.

Die Tabelle 3.4 stellt die Anforderungen an den Client zusammengefasst dar.

Komponente	**Anforderung**
Betriebssystem	• Embedded Linux • Begrenzung auf die für den Betrieb notwendigen Software-Pakete • Bereitstellung der Schnittstellen zur Hardware (Treiber) • Laufzeitumgebung für Client-Applikation • Möglichkeit zur Herstellung einer Internetverbindung • Paket-Manager zur Software-Verwaltung
Client-Applikation	• Start und Überwachung der Software zur Messdatenerfassung • Upload von Status-Nachrichten auf den Server • Download und von Konfigurationen vom Server • Durchführung von Software-Updates über den Server • Konvertierung der Datensätze in ein standardisiertes Format • Upload der Messdatensätze auf den Server

Tabelle 3.4.: Anforderungen an den Client

4. Konzept

In diesem Kapitel wird ein Konzept für das zu entwickelnde System erläutert. Hierzu wird zunächst ein Überblick über die Komponenten und deren Aufgaben im Gesamtsystem gegeben. Anschließend werden Konzepte für die einzelnen Komponenten entwickelt. Es wird insbesondere auch auf die Speicherung der Messdaten eingegangen, da eine sehr große Datenmenge gespeichert und bereitgestellt werden soll.

4.1. Management- und Analysesystem

4.1.1. Client-Server-Architektur

Das System wird mit einer Client-Server-Architektur umgesetzt. Die Clients rufen verschiedene Funktionalitäten über ein Netzwerk auf dem Server auf, der diese zur Verfügung stellt. Es werden zwei Arten von Clients unterschieden: Anwender und Datenerfassungssysteme. Diese können über unterschiedliche Netzwerke mit dem Server verbunden sein. Der Datenaustausch zwischen allen Teilnehmern ist dabei bidirektional.

Somit besteht das System aus den Hauptkomponenten Server, Datenerfassungssystemen, im folgenden als „Clients“ bezeichnet, und dem Anwender, die über Kommunikationsnetzwerke miteinander verbunden sind. Der Anwender kann über ein weiteres Netzwerk mit dem Server verbunden sein. Die Abbildung 4.1 zeigt die Struktur des Gesamtsystems.

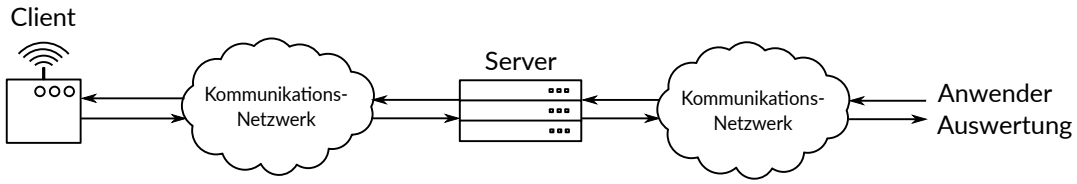

Abbildung 4.1.: Architektur des Gesamtsystems

4.1.2. Funktionsweise des Systems

Das Ziel ist es, dem Anwender Zugriff auf die durch das System erfassten Messdaten zu ermöglichen. Die Datenerfassungssysteme besitzen jedoch aus energietechnischen Gründen keine permanente Verbindung zum Kommunikationsnetzwerk, sodass ein direkter Zugriff nicht jederzeit möglich ist. Daher wird ein Server in die Kommunikation einbezogen und dient als „Mittelsmann“ zwischen Anwender und Client. Die von den jeweiligen Teilnehmern übertragenen Daten werden auf dem Server zentral gespeichert und zum Abruf bereitgestellt. Damit wird die Kommunikation unabhängig vom Verbindungsstatus der Clients. Der Server ist, im Gegensatz zu Anwender und Client, permanent mit dem Kommunikationsnetzwerk verbunden.

In Abbildung 4.2 ist ein beispielhafter Ablauf der Kommunikation dargestellt. Sobald der Client mit dem Server verbunden ist, werden eine Status-Meldung *S1* und aufgezeichnete Messdaten *M1* auf den Server übertragen. Danach wird die Verbindung zwischen Client und Server wieder getrennt. Der Anwender kann unabhängig vom Verbindungsstatus des Clients nun die hochgeladenen Daten *S1* und *M1* auf dem Server einsehen und herunterladen.

Das Verhalten des Clients wird durch eine Konfiguration bestimmt. Der Anwender kann daher Einfluss auf das Verhalten des Clients nehmen, indem dessen Konfiguration aktualisiert wird. Die aktualisierte Konfiguration *K1* wird vom Anwender auf den Server übertragen. Sobald der Client das nächste mal eine Verbindung zum Server aufgebaut hat, wird diese aktualisierte Konfiguration *K1* heruntergeladen und anschließend aktiviert. Zusätzlich kann der Client vom Server Software-Updates für das Betriebssystem herunterladen und installieren. Diese Operation wird ebenfalls über die Konfiguration des Clients gesteuert und ist somit ebenfalls durch den Anwender konfigurierbar.

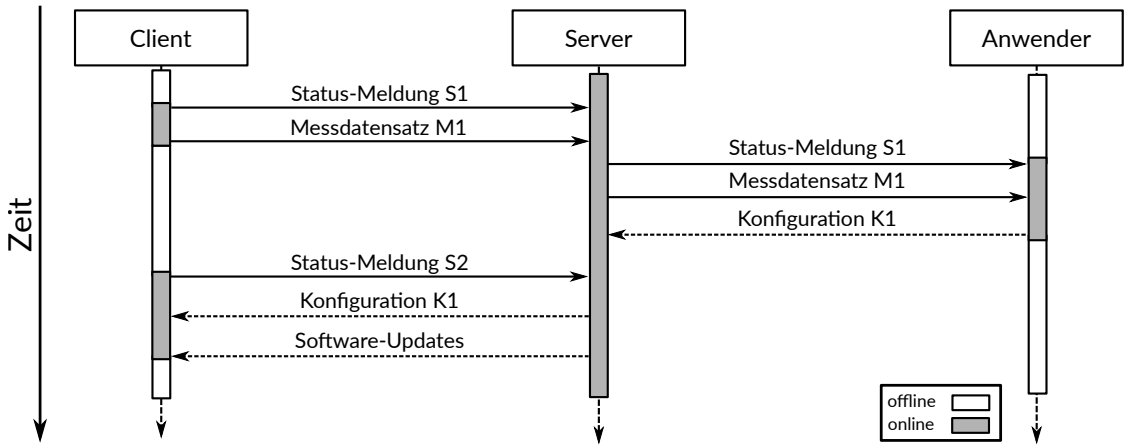

Abbildung 4.2.: Zeitlicher Ablauf der Management-Funktionen

4.2. Kommunikationsnetzwerke

Es werden zwei unterschiedliche Kommunikationsnetzwerke unterschieden. Über eines der Netzwerke sind die Clients mit dem Server verbunden, während der Anwender über ein weiteres Netzwerk mit dem Server interagiert.

Da die Datenerfassungssysteme autonom arbeiten und mobil sind, können diese nur über eine drahtloses Schnittstelle mit einem Netzwerk in Verbindung treten. Da es keine Beschränkungen gibt, wo sich die Clients aufhalten dürfen, verwenden die Clients das Mobilfunknetz, um eine Internetverbindung aufzubauen. Durch die weltweite und flächendeckende Verbreitung des Mobilfunknetzes können so auch geografisch weit entfernte Clients mit dem Server in Verbindung treten. Der Server wiederum muss ebenfalls mit dem Internet verbunden sein, damit die Clients eine Verbindung über das Internet aufbauen können.

Da zwischen Server und Client auch sensible Daten wie Konfigurationen oder Messdaten übertragen werden sollen, muss die Verbindung zwischen Client und Server verschlüsselt werden.

Der Anwender verbindet sich mit dem Server über ein lokales Netzwerk mit dem Server.

Die folgende Abbildung 4.2 zeigt die Struktur der Anwendung mit der Kommunikation über mobiles Internet und Verwendung eines VPN.

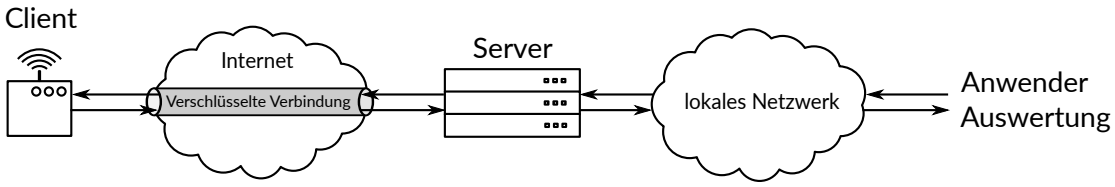

Abbildung 4.3.: Konzept für die Datenübertragung zwischen Server und Client

4.3. Server

Auf dem Server werden alle Ressourcen gespeichert und verwaltet. Anwender und Clients können die Ressourcen anfragen oder auf dem Server ablegen. Es wird eine Applikation entworfen, welche die Interaktion ermöglicht. Nachfolgend werden die Konzepte für die Datenspeicherung und die Applikation auf dem Server detailliert beschrieben.

Die Abbildung 4.4 zeigt das Konzept für die Server-Struktur.

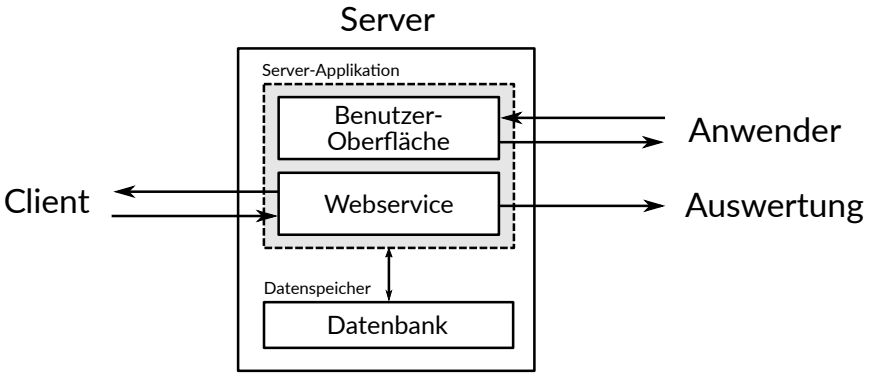

Abbildung 4.4.: Konzept für die Server-Struktur

4.3.1. Datenspeicherung

Die Daten auf dem Server werden durch ein Datenbanksystem verwaltet, um die Konsistenz der Daten zu gewährleisten und eine standardisierte Schnittstelle für alle Ressourcen zur Verfügung zu stellen.

Zunächst wird ermittelt, welche Informationen auf dem Server gespeichert werden müssen. Hierzu wird das Datenmodell entworfen, welches einen Client repräsentiert. Außerdem wird betrachtet, wie diese Informationen in Beziehung zueinander stehen müssen, damit eine eindeutige Repräsentation eines Clients vorliegt.

Die folgenden Ressourcen werden im System gespeichert und verwaltet:

- **Client-Informationen**:
  Allgemeine Informationen über einen Client, die der Identifikation und Organisation dienen.
- **Status-Informationen**:
  Enthält Informationen über den aktuellen Zustand eines Clients. Dazu zählen z.B. der freie Speicherplatz auf dem Massenspeicher, Ladezustand des Akkus oder der aktuelle Status der Mobilfunkverbindung.
- **Konfigurationen**:
  Die Konfiguration bestimmt das Verhalten des Clients. Hier werden Daten wie die zu verwendende Datenerfassungssoftware, Zeitpunkte für geplante Aufgaben, Zugangsdaten für VPN und Informationen über die Kommunikation mit dem Server abgelegt.

- **Messdatenpunkte**:
  Ein Datenpunkt enthält genau einen Messwert für jeden Sensor, der vom Client ausgewertet wurde. Idealerweise wurden alle Sensorwerte eines Clients zum exakt gleichen Zeitpunkt aufgezeichnet.
- **Messdatensätze**:
  Ein Datensatz besteht aus einer Reihe aneinander hängender Datenpunkte. Die separate Speicherung von Datensätzen ist erforderlich, da möglicherweise Messwerte in der Reihenfolge ausgewertet werden sollen, in der sie aufgezeichnet wurden (z.B. GPS-Position).

Durch Verknüpfung dieser Ressourcen entsteht ein Datenmodell für einen Client, welches in Abbildung 4.5 dargestellt ist.

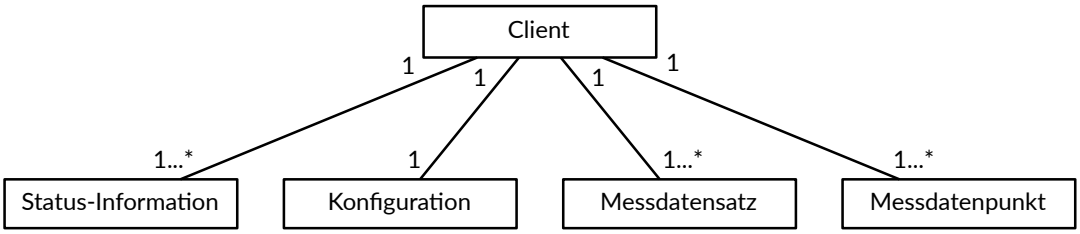

Abbildung 4.5.: Datenbanklayout für Datenlogger-Informationen

Zu jeden Client gehört demnach immer genau eine Konfiguration und es können mehrere Status-Informationen vorliegen. Es gilt außerdem, dass mehrere Datensätze und Datenpunkte zu einem Client zugeordnet werden. Umgekehrt gehören Status-Informationen und Konfiguration immer nur zu einem Client. Auch Datensätze und Datenpunkte werden exakt einem Client zugeordnet.

4.3.2. Management-Server-Applikation

Für die Interaktion mit dem Server wird eine eigenständige Applikation entworfen, die mit der Datenbank verbunden ist. Über die Server-Applikation können Clients und Anwender so Daten mit dem Datenspeicher austauschen. Die Applikation ist in zwei Bereiche unterteilt: eine grafische Benutzerschnittstelle für den Anwender und eine Schnittstelle zur Kommunikation mit den Clients (API).

4.3.2.1. Benutzer-Schnittstelle

Der Anwender verwendet das System ausschließlich über die grafische Oberfläche. Diese muss daher alle geforderten Funktionalitäten zur Verfügung stellen.

Über die Benutzer-Schnittstelle werden alle im System registrierten Clients überwacht und deren Status-Informationen angezeigt werden. Es kann außerdem die auf dem Server hinterlegte Konfiguration für einen Client bearbeitet und gespeichert werden. Die Benutzer-Schnittstelle bietet die Möglichkeit einen gefilterten Datensatz anzufragen, der anschließend heruntergeladen werden kann. Über eine Benutzer-Oberfläche können die gewünschten Messgrößen ausgewählt und eingeschränkt werden. Weiterhin werden die Software-Updates verwaltet, die auf den Clients installiert werden sollen.

Der Zugang zu den Daten über die Benutzeroberfläche wird durch eine Benutzer-Verwaltung geregelt, wobei zwischen verschiedenen Benutzer-Gruppen unterschieden wird. Die Benutzer-Gruppen erhalten unterschiedliche Zugriffsrechte auf die Funktionen der Schnittstelle, sodass ein unbefugter Datenzugriff verhindert wird. Das ist insbesondere für die Bearbeitung der Konfigurationen wichtig, da durch eine Fehlkonfiguration Clients außer Betrieb gesetzt werden können.

Die Abbildung 4.6 zeigt die Funktionen der Benutzeroberfläche.

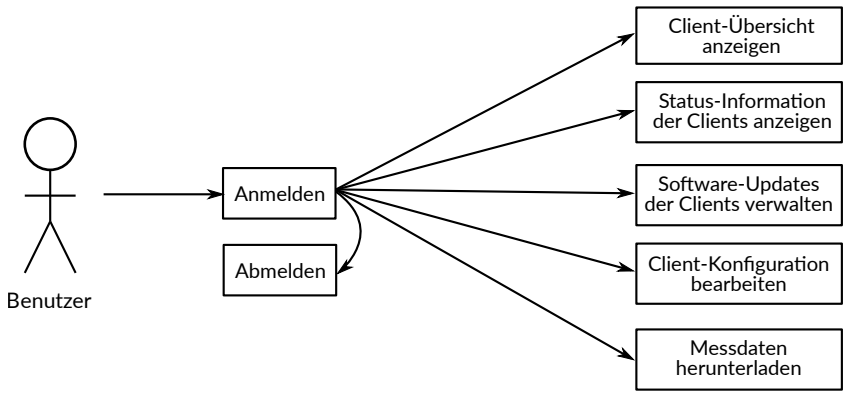

Abbildung 4.6.: Mögliche Interaktionen über die Benutzer-Schnittstelle des Servers

4.3.2.2. Webservice (API)

Für die Clients wird eine weitere Schnittstelle vorgesehen, über die in einem standardisierten Format Daten mit dem Server ausgetauscht werden können. Diese Schnittstelle kann auch von anderen Software-Komponenten verwendet werden, um z.B. eine automatisierte Messdatenauswertung zu realisieren. Da der Datenaustausch über das Netzwerk erfolgt,

wird diese Schnittstelle als „Webservice“ oder „Web-API“ (Application Programming Interface) bezeichnet.

Über den Webservice können die folgeden Interaktionen mit dem Server durchgeführt werden:

- Status-Meldungen hochladen
- Messdaten hochladen
- Konfigrationen herunterladen
- Software-Updates herunterladen

Für die externe Auswertung der Daten stehen zusätzlich die folgenden Funktionen zur Verfügung:

- Status-Meldungen herunterladen
- Messdaten herunterladen

Alle möglichen Operationen, die über den Webservice möglich sind, werden in Abbildung 4.7 dargestellt.

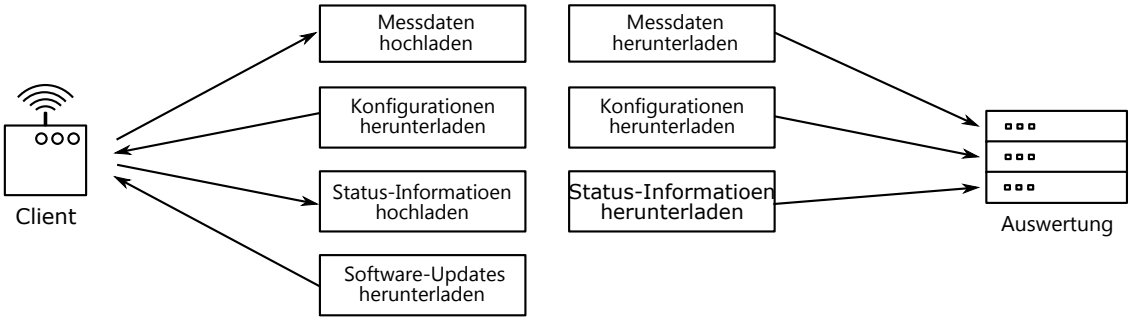

Abbildung 4.7.: Mögliche Interaktionen über den Webservice des Servers

4.4. Client

Die Clients sind für die automatisierte Messdatenerfassung im Gesamtsystem zuständig. Diese arbeiten autonom und verwenden zur Ressourcen- und Speicherverwaltung ein Betriebssystem. Über eine Datenerfassungssoftware werden die Sensoren ausgelesen, lokal gespeichert und anschließend auf den Server hochgeladen. Damit die Clients über das Management-System verwaltet werden können, ist außerdem eine Management-Applikation notwendig, die das entsprechende Gegenstück zum Server darstellt.

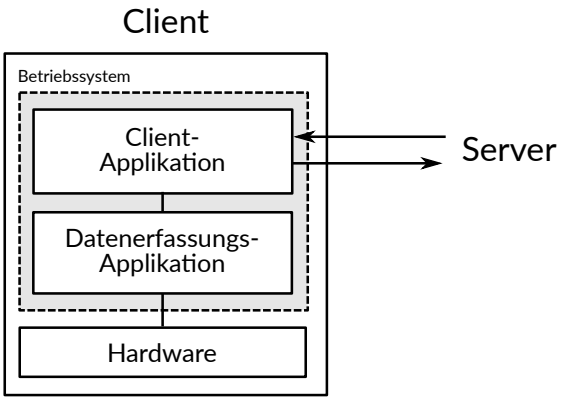

Abbildung 4.8.: Konzept für die Client-Applikation

4.4.1. Betriebssystem

Da bis zu 100 Clients verwaltet werden sollen, ist es sinnvoll, ein einheitliches Betriebssystem zu verwenden. Zusätzlich sollen unterschiedliche Hardware-Plattformen im Gesamtsystem zusammenarbeiten können. So muss das Betriebssystem trotzdem einen einheitlichen Funktionsumfang liefern, damit auf allen Clients die gleiche Software zu Einsatz kommen kann.

Das Betriebssystem muss die Hardware-Treiber für Sensorik des Datenerfassungssystems und eine Laufzeitumgebung für die Client-Applikation und die Datenerfassungssoftware bereitstellen. Weiterhin muss eine verschlüsselte Verbindung zu einem entfernten Server über das Mobilfunknetz hergestellt werden können.

Es wird ein Linux-basiertes Betriebssystem verwendet, da dieses gut skalierbar ist und somit für eine große Auswahl an unterschiedlichen Hardwares zur Verfügung gestellt werden kann. So ist die Plattformunabhängigkeit des Gesamtsystems gewährleistet. Linux-Systeme lassen sich außerdem sehr gut an die jeweiligen Bedürfnisse und das Anwendungsgebiet anpassen.

4.4.2. Management-Client-Applikation

Die Management-Applikation ist die zentrale Software auf den Clients, welche die Messdatenerfassung steuert und den Datenaustausch mit dem Server realisiert. Somit ist die Applikation für das Gesamtverhalten des Clients verantwortlich.

Die Datenerfassung wird von einer externen Software übernommen, die lediglich von der Client-Applikation gestartet und überwacht wird. Der eigentliche Zugriff auf die Sensorik des Datenerfassungssystems ist daher unabhängig von der Management-Client-Applikation.

Die Management-Applikation verwendet eine zentrale Konfigurationsdatei, in der alle für den Betrieb des Clients notwendigen Einstellungen vorgenommen werden. Über den Webservice kann diese Konfiguration durch eine vom Anwender auf dem Server hinterlegte Konfiguration ausgetauscht werden. So wird das Verhalten des Clients indirekt durch den Anwender beeinflusst.

Grundsätzlich werden die folgenden vier Aufgaben durch die Applikation durchgeführt:

- Steuerung der Datenerfassung
- Ermittlung und Senden des System-Status an den Server
- Herunterladen und aktivieren von Konfigurationen
- Durchführung von Software-Updates über den Server

Es wird der Zugang zu Betriebssystem-Funktionen benötigt, die zum Aufbau der Verbindung zum Server und zur Ermittlung des System-Status erforderlich sind. Die Interaktion mit dem Server wird über ein Webservice-Interface durchgeführt, welches zu dem Webservice auf dem Server kompatibel ist.

Die Funktionen der Client-Applikation sind in der Abbildung 4.9 dargestellt.

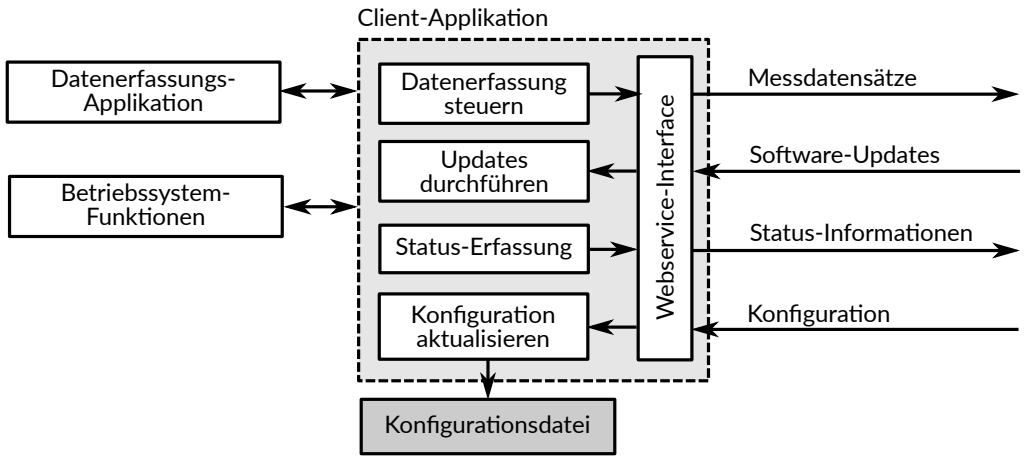

Abbildung 4.9.: Konzept der Management-Client-Applikation

4.5. Zusammenfassende Übersicht über das Konzept

Das System wird als Client-Server-System mit Hilfe von Web-Technologien umgesetzt. Dadurch ist das System sehr gut skalierbar und kann um weitere Clients und Anwender erweitert werden.

Den Kern des Systems stellt ein Webservice auf dem Server dar, über den die Interaktion mit den Ressourcen durchführt wird. Für die Verwaltung der Ressourcen, verwendet der Server eine Datenbank.

Die Clients nutzen den Webservice, um die erfassten Messdaten und Status-Informationen auf den Server hochzuladen. Außerdem werden auf diese Weise auch Konfigurationen und Software-Updates vom Server heruntergeladen.

Der Anwender kann die Status-Informationen der Clients über eine grafische Benutzeroberfläche auf dem Server einsehen und die dort gespeicherten Konfigurationen bearbeiten und Messdatensätze herunterladen. Ebenfalls über den Webservice können die Messdaten auch in maschinenlesbarem Format heruntergeladen werden. Ebenso ist der Zugriff auf Status-Informationen und Konfigurationen über den Webservice möglich. Auf diese Weise können Messdaten, Status-Informationen und Konfigurationen auch automatisch ausgewertet werden.

Die Kommunikation zwischen Client und Server ist somit vollständig entkoppelt von der Kommunikation des Anwenders mit dem Server. Die Abbildung 4.10 zeigt den Fluss der Daten im System.

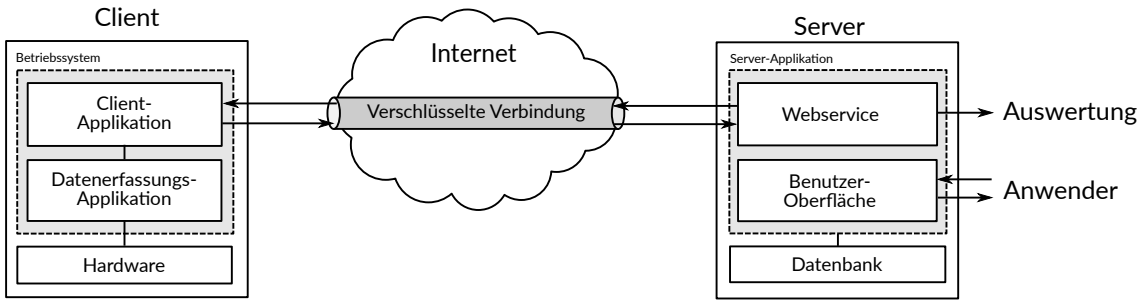

Abbildung 4.10.: Konzept des Gesamtsystems

5. Realisierung des Systems

Im folgenden Kapitel wird die Realisierung des zuvor beschrieben Konzepts erläutert. Zunächst werden die verwendeten Komponenten vorgestellt. Anschließend wird die Umsetzung der Datenbank und einer Web-Applikation auf dem Server beschrieben. Es wird die Inbetriebnahme und Anpassung eines Betriebssystems für den verwendeten Client, sowie die Implementierung einer Management-Applikation, die eine Fernwartung des Clients ermöglicht.

5.1. Übersicht der verwendeten Komponenten

5.1.1. Server: Virtual Machine

Als Server dient eine Virtual Machine (VM), die im Rechenzentrum an der HAW Hamburg läuft. Der Server ist 24 h am Tag in Betrieb und damit hochverfügbar. Über eine feste IP-Adresse ist die VM auch außerhalb des HAW-Netzwerkes erreichbar.

Die Hardware und das verwendete Betriebssystem des Servers sind in der Tabelle 5.1 aufgelistet.

Prozessor	Typ:	Intel Xeon E5-2670 v2
	Taktrate:	2,5 GHz
	Verfügbare Kerne:	2
Speicher	Arbeitsspeicher:	4 GB
	Festplatte:	100 GB
	NFS-Laufwerk:	200 GB
Betriebssystem	System/Version	Ubuntu 14.04.4 LTS (Linux-basiert)
	Kernel	3.16.0-70

Tabelle 5.1.: Übersicht: Virtual Machine (Server)

5.1.2. Client: „BEEDeL“ Datenlogger

Als Client steht der im Projekt „BEEDeL“ eingesetzte Datenlogger zur Verfügung. Dieser arbeitet autonom und ist für den Einbau in einen Nahverkehrsbus konzipiert. Über ein UMTS-Modem hat der Datenlogger einen mobilen Internet-Zugang. Die Energieversorgung wird im laufenden Betrieb durch den Nahverkehrsbus übernommen und durch einen Akkumulator gepuffert.

Der Datenlogger zeichnet die folgenden Messgrößen auf:

- GPS-Informationen (Position, Anzahl der Satelliten, Kurs)
- Geschwindigkeit des Busses (Tacho-Signal)
- Beschleunigung (Inertialsensor, 3 Achsen)
- Drehrate (Inertialsensor, 3 Achsen)
- Erd-Magnetfeld (3 Achsen)
- Temperatur
- Luftdruck

Der Datenlogger basiert auf dem Einplatinen-Computer „Raspberry Pi“, an welchen die Sensoren über verschiedene Schnittstellen wie UART, I2C und SPI angeschlossen sind. Als Betriebssystem wird „Raspian“, ein Debian-basiertes Linux-System, das speziell für den Raspberry Pi entwickelt wurde, eingesetzt. Als drahtlose Debug-Schnittstelle steht ein WLAN-Accesspoint zur Verfügung, über den anschließend ein Login über SSH auf dem Datenlogger möglich ist.

Eine Software, welche die genannten Messgrößen erfasst ist ebenfalls vorhanden. Diese zeichnet die Messdaten auf und wandelt die Rohdaten anschließend in einen Datensatz um, wobei eine MATLAB-kompatible Datei erzeugt wird. Anschließend werden die Rohdaten und die MATLAB-Datei komprimiert. Dieser Datensatz wird über SSH an den Server übertragen.

In Abbildung 5.1 ist der verwendete Datenlogger abgebildet. Weitere Informationen zu der Funktionsweise können [32] entnommen werden.

Abbildung 5.1.: Client: BEEDeL Datenlogger für Nahverkehrsbusse

5.2. Datenübertragung im Netzwerk

Nachfolgend werden ein Übertragungsprotokoll und die Datenformate ausgewählt, über welche die verschiedenen Teilnehmer im System Daten austauschen. Wie bereits in Abschnitt 4.2 erwähnt, kommunizieren die Datenerfassungssysteme über das Mobilfunknetz. Da große Datenmengen an den Server übertragen werden müssen, wird die UMTS-Technologie für mobiles Internet verwendet. Mit UMTS sind Übertragungsgeschwindigkeiten von bis zu 384 kBit/s möglich. Mit der Erweiterung HSPA können 14,4 MBit/s übertragen werden.

5.2.1. Sicheres Transport- und Netzwerkprotokoll

Damit die Übertragung der Daten über das Internet nicht öffentlich ist, wird für die Verbindung zwischen Client und Server ein VPN verwendet. Auf diese Weise wird die komplette Datenübertraung zwischen Server und Client verschlüsselt. Da IP-Adressen im Mobilfunknetz dynamisch und durch den jeweiligen Provider vergeben werden, ist eine eindeutige Identifikation des verbundenen Clients nicht immer möglich. Durch die interne IP-Adresse im VPN können die Clients jedoch eindeutig identifiziert werden. Dadurch ist zusätzlich die Möglichkeit des Zugriffs auf weitere Netzwerkdienste des Servers gegeben.

Zum Aufbau der virtuellen privaten Netzwerks wird *OpenVPN* eingesetzt. Dies erlaubt die Herstellung einer verschlüsselten TLS-Verbindung und verwendet TCP oder UDP für den Datentransport. Die Authentifizierung erfolgt dabei über die Public Key Infrastructure (PKI), also durch öffentliche und private Schlüssel. Diese werden für den VPN-Server und die VPN-Clients im Voraus generiert und auf die Endgeräte übertragen. Zusätzlich werden die Schlüssel mit einem Zertifikat signiert, welches sich auf dem Server befindet. Bei einem Verbindungsaufbau durch den Client kann so leicht geprüft werden, ob der Client-Schlüssel mit dem korrekten Zertifikat signiert wurde [1].

5.2.2. Anwendungsprotokoll

Als Übertragungsprotokoll für den Datentransport im System wird HTTP/1.1 verwendet. Da HTTP ein Plattform-übergreifender Standard ist, können so Systeme mit unterschiedlicher Systemarchitektur zusammenarbeiten. Da HTTP sehr universell einsetzbar ist, kann sowohl die grafische Benutzeroberfläche, also auch die Datenkommunikation zwischen Client und Server realisiert werden.

Dabei werden die HTTP-Methoden direkt auf Adressen des Servers angewendet. Mit HTTP-GET kann so z.B. eine Ressource angefragt werden. Durch diese Art der Verwendung von HTTP kann das System als RESTful bezeichnet werden (siehe dazu auch Abschnitt 2.2.1).

5.2.3. Datenformate

Für den Informationsaustausch zwischen Client und Server wird Javascrip Object Notation (JSON) verwendet. JSON ist ein Text-basiertes Dateiformat und ist in ECMA-404[1] spezifiziert. Durch die im Vergleich zu XML kompakte Syntax entsteht wenig Overhead bei der Beschreibung von Datenobjekten. Das Format besteht aus zwei grundsätzlichen Strukturen: Name/Wert-Paare (Objekte) und Arrays (Listen). Diese Datenstrukturen werden von allen modernen Programmiersprachen unterstützt. Daher kann das JSON-Format von vielen Programmiersprachen leicht erzeugt und gelesen werden.

Als Ausnahme werden für die Messdaten auch das Comma-separated Values (CSV)-Format für den Up- und Download zugelassen. Da die Messdaten den größten Teil der zu übertragenden Daten ausmachen, ist ein minimaler Overhead für die Datenbeschreibung erforderlich. Das CSV-Format ist in RFC-4180[2] der IETF spezifiziert. Das ebenfalls textbasierte Format ist wie eine Tabelle zu betrachten: Die Zeilen enthalten Werte, die mit einem Komma getrennt sind. So lassen sich Daten mit einfacher Struktur sehr effizient darstellen. Ein

[1] http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf
[2] https://tools.ietf.org/html/rfc4180

weiterer Vorteil von CSV ist einfache Generierung und Dekodierung. So unterstützen viele Programme den Daten-Import über das CSV-Format, was für die Auswertung der Messdaten von großem Vorteil ist.

5.3. Web-Applikation auf dem Server

Auf dem Server wird eine Web-Applikation implementiert, welche die Interaktion mit den Clients und dem Anwender durchführt. Die Applikation ist mit einer Datenbank verbunden, die alle Daten verwaltet. Nachfolgend wird die Struktur der implementierten Applikation und der Datenbank erläutert.

Für die Implementierung der Web-Applikation wird das PHP-Framework *Laravel* [3] verwendet. *Laravel* ist ein *Open-Source*-PHP-Framework, welches dem MVC-Entwurfsmuster (Model-View-Controller) folgt. Im Projekt wird die Version 5.1 verwendet.

Das Framework erlaubt die Implementierung von *RESTful*-Applikationen, welche über *REST*-konforme Schnittstellen verfügen (siehe auch 2.2.1).

Im Anhang B.2 kann der Quellcode der Webapplikation eingesehen werden.

5.3.1. Struktur der Applikation

Die Applikation wird nach dem MVC-Muster entwickelt und in die Komponenten Datenmodell (Model), Präsentation (View) und Steuerung (Controller) unterteilt. Die Abbildung 5.2 zeigt das MVC-Entwurfsmuster.

- **Model:**
  Das Modell enthält die Daten und ist im Datenspeicher persistent abgelegt. Das Modell ist von Präsentation und Steuerung unabhängig.
- **View:**
  Die Präsentation stellt die Daten des Modells repräsentativ dar und liefert dies an den Benutzer aus (z.B. in Form einer Website).
- **Controller:**
  Der Controller ist für die Interaktion mit dem Benutzer zuständig und verarbeitet dessen Anfragen. Je nach Anfrage werden so die Daten des Modells verändert oder Views aktualisiert.

[3] https://laravel.com/

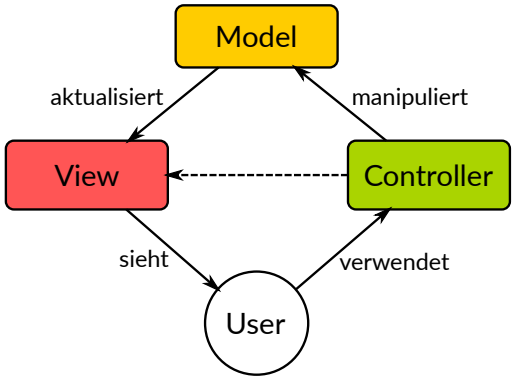

Abbildung 5.2.: Model-View-Controller (MVC)-Entwurfsmuster

Da sowohl Client, als auch Anwender mit den Daten aus den Modellen interagieren, werden diese unabhängig erläutert. Die Implementierungen von Controllern und Views werden nach Verwendungszweck (Webservice und Webseite) getrennt betrachtet.

5.3.2. Datenbank

5.3.2.1. Technologie

Für die Verwaltung der Daten auf dem Server wird Oracle MySQL 5.5 verwendet. Dabei handelt es sich um ein relationales Datenbanksystem.

Der Grund für diese Wahl ist die hohe Flexibilität bei der Datenabfrage, die in Abschnitt 3.3 gefordert wurde. Da ein eine solche Datenbank das Daten-Schema vorgibt, sind die Voraussetzungen für die Erfüllung der ACID-Eigenschaften (siehe Abschnitt 3.3.1.1) gegeben. Weiterhin sind relationale Datenbank-Systeme zurzeit sehr weit verbreitet und es gibt eine Vielzahl an Schnittstellen zu anderen Anwendungen.

Die zu speichernde Datenmenge ist groß, kann jedoch ohne Probleme durch ein solches System bewältigt werden. Da das Datenmodell sehr einfach ausfällt, ist zu erwarten, dass die Abfrage-Geschwindigkeit trotz der großen Datenmenge ausreichend schnell ist.

5.3.2.2. Datenmodell

Dem Datenmodell aus Abschnitt 4.5 entsprechend, werden die einzelnen Modelle für die zu speichernde Ressource angelegt. Dafür stellt *Laravel* eine Basis-Klasse „Model“ zur Verfügung. Diese Klasse enthält Methoden, welche die Interaktion mit der Datenbank implementieren.

Die folgenden Model-Klassen werden erzeugt:

- Datalogger: Informationen über einen Datenlogger (Client)
- DataloggerConfig: Konfiguration
- DataloggerStatus: Status-Informationen
- DataloggerDataset: Messdatensatz
- DataloggerDatapoint: Ein Messwert aller Sensoren

Laravel verwendet „Migrations“, um Datenbank-Tabellen für Modelle automatisch anzulegen. Darin sind auch die Abhängigkeiten und Beziehungen zwischen den einzelnen Tabelle beschrieben. In der folgenden Abbildung 5.3 ist das vollständige Datenbank-Layout dargestellt, welches die Modelle repräsentiert.

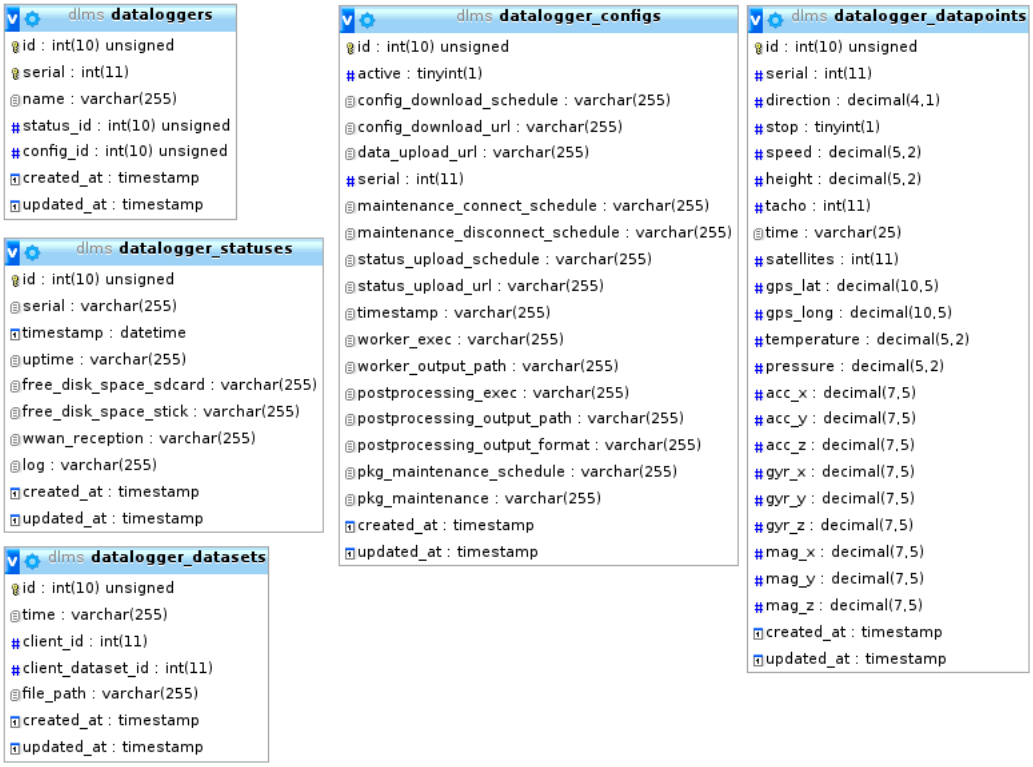

Abbildung 5.3.: Datenbanklayout für die Datenlogger-Informationen

5.3.3. RESTful Webservice

Über den Webservice (oder Web-API) können Status-Informationen, Konfigurationen und Messdaten über das JSON- bzw. CSV-Format mit dem Server ausgetauscht werden. Die

Bezeichnung *RESTful* bedeutet hier, dass nur die zustandslosen HTTP-Methoden für die Kommunikation verwendet werden. (siehe auch 2.2.1)

Es werden Controller für Status-, Konfigurations- und Messdateninteraktion implementiert, welche die jeweiligen geforderten Funktionen realisieren. In der Tabelle 5.2 sind alle Funktionen des Webservice zusammengefasst. Die einzelnen Funktionen und deren Parameter (in eckigen Klammern) werden im folgenden erläutert.

HTTP-Methode	**Adresse**	**Funktion**
Status-Informationen		
POST	/api/status/client/[id]/upload	Status-Meldung hochladen
GET	/api/status/client/[id]/status	Status eines Clients herunterladen
GET	/api/status/[id]	Status-Meldungen herunterladen
Konfigurationen		
GET	/api/config/client/[id]	Konfiguration herunterladen
Messdaten		
POST	/api/data/client/[id]/dataset	Datensatz hochladen
GET	/api/data/client/[id]/dataset/[id]/csv	Messdatensätze herunterladen
GET	/api/data/datapoints/[format]	Einzelne Messpunkte herunterladen

Tabelle 5.2.: Übersicht der Webservice-Funktionen

5.3.3.1. Status-Informationen austauschen

Zum Upload einer Status-Datei ist die Angabe der Client-ID in der URL notwendig. Der Inhalt der Status-Meldung wird im *file*-Parameter der POST-Methode übertragen. Nach der Prüfung der Datei auf Konsistenz, speichert der Controller die Datei auf dem Server und trägt die Daten aus der Datei in die Datenbank ein. Anschließend wird die Verknüpfung des entsprechenden Clients aktualisiert, sodass der hochgeladene Status nun als aktueller Status des Clients eingetragen ist.

Zum Download von Status-Meldungen wird nur die Client-ID benötigt, da jeder Client nur einen aktuellen Status besitzt. Alternativ können die Status-Informationen durch Angabe der laufenden Nummer aller Status-Meldungen heruntergeladen werden.

5.3.3.2. Konfigurationen herunterladen

Konfigurationen können ebenfalls unter Angabe der Client-ID heruntergeladen werden. Da jeder Client immer mit nur einer Konfiguration verknüpft ist, sind keine weiteren Parameter nötig. Der Controller für Konfigurationen findet die Konfiguration in der Datenbank über die eindeutige Verknüpfung zwischen Client-Modell und Konfigurationen.

5.3.3.3. Messdaten Up- und Download

Die Messdaten können in beiden Richtungen ausgetauscht werden. Zum Upload eines Messdatensatzes ist ebenfalls die Angabe der Client-ID notwendig. Da Messdatensätze sehr viele Einträge enthalten können, kann die Eintragung in die Datenbank einen längeren Zeitraum in Anspruch nehmen (mehrere Minuten). Da die Clients aber nur für kurze Zeit mit dem Server verbunden sind, wird bereits direkt nach dem Upload der Messdatendatei ein entsprechender Return-Code gesendet, um dem Client die Datenübertragung zu quittieren. Die Eintragung der Messdaten in die Datenbank wird anschließend mit Hilfe von *Background-Jobs* durchgeführt, die über eine *Job-Queue* verwaltet werden. Die *Job-Queue* wird ebenfalls in der Datenbank gespeichert, doch werden nur Informationen über die aufzurufende Funktion und deren Parameter gespeichert. Mit einem *Worker* kann die *Queue* abgearbeitet werden, wobei auch mehrere *Worker* auf einer *Queue* arbeiten können und so die Messdaten parallelisiert eintragen.

Über den Webservice können die Messdatensätze, wie sie von den Clients hochgeladen wurden, wieder heruntergeladen werden. Dazu ist die Angabe der Client-ID und die laufende Nummer des Datensatzes nötig. Diese Art des Messdatendownloads ist besonders gut für die automatisierte Auswertung, da Client-IDs und Datensatz-IDs permutiert werden können. Die Datensätze werden dabei im CSV-Format zurückgeliefert.

Zusätzlich können auch einzelne oder gefilterte Messdaten angefordert werden. In der Route wird lediglich das gewünschte Datenformat (JSON oder CSV) angegeben. Zur Filterung stehen eine ganze Reihe von Parametern zur Verfügung, die als HTTP-GET-Parameter übergeben werden. Dabei ist es möglich, jede Messgröße einzeln auszuwählen und einen Wertebereich für diese Größen anzugeben. Weiterhin kann die Anzahl der gewünschten Messdatenpunkte und eine Sortierreihenfolge angegeben werden.

5.3.4. Benutzeroberfläche (Webseite)

Der Anwender kann über die grafische Benutzeroberfläche mit den Modellen interagieren. Die Oberfläche wird als Webseite realisiert, sodass der Anwender lediglich einen Brow-

ser benötigt, um die Benutzeroberfläche verwenden zu können. Es wird eine Benutzer-Verwaltung, die unterschiedliche Benutzer-Gruppen mit verschiedenen Zugriffsrechten verwaltet, implementiert.

Zum Erstellen der Views bietet das *Laravel*-Framework die *Template-Engine* „Blade“. Damit lässt sich PHP-Code direkt in HTML-Seite integrieren, sodass einzelne HTML-Objekte dynamisch erzeugt werden können. Der für die entsprechende Adresse zuständige Controller liest entsprechend der Anfrage Daten aus der Datenbank aus und übergibt diese der *View*. Auf diese Weise werden z.B. die Listen für registrierte Clients und Konfigurationen dynamisch erzeugt, je nachdem wie viele Einträge in der Datenbank vorhanden sind.

Das Design der Website wurde mit Hilfe des *Bootstrap* Frameworks entwickelt. Das Framework stellt unter Anderem HTML- und CSS-Vorlagen für Tabellen, Buttons und Menüs zur Verfügung.

In der Tabelle 5.3 sind alle auf der Website verfügbaren Ansichten und Funktionen dargestellt.

HTTP-Methode	**Adresse**	**Funktion**
Benutzer-Verwaltung		
GET	/auth/login	Login-Formular
GET	/auth/register	Formular zum Erstellen von Benutzern
POST	/auth/logout	Abmeldung vom System
Client-Übersicht		
GET	/dl/list	Liste aller registrierten Clients
Status-Informationen		
GET	/status/list	Liste aller Status-Informationen
GET	/status/list/[id]	Liste der Status-Informationen eines Clients
GET	/status/by-id/[id]	Anzeige eines Status
Konfigurationen		
GET	/config/list	Liste aller Konfigurationen
GET	/config/create	Formular zum Erstellen einer Konfiguration
GET	/config/edit/id	Formular zum editieren der Konfiguration [id]
GET	/config/by-id/id	Anzeige der Konfiguration [id]
POST	/config/store	Konfigurationen speichern (intern)
Messdaten		
GET	/data/export/[format]	Formular zum herunterladen von Messdaten

Tabelle 5.3.: Übersicht der Funktionen der Benutzeroberfläche

5.3.4.1. Benutzer-Verwaltung

Der Zugang zu den Funktionen der Webseite ist durch eine Benutzer-Verwaltung geregelt. Es existieren drei Benutzergruppen: passive, operative und administrative Benutzer. Die passiven Benutzer können lediglich Client- und Status-Informationen einsehen. Die Bearbeitung von Konfigurationen und Herunterladen von Messdaten ist nur operativen Nutzern erlaubt. Die Administratoren haben zusätzlich die Möglichkeit neue Benutzer für alle Gruppen anzulegen.

5.3.4.2. Ansicht von Client-Informationen

Über die Webseite können alle registrierten Clients und deren Status-Informationen eingesehen werden. Dazu stehen eine Liste für alle Clients und Status-Informationen, sowie Detailansichten zur Verfügung. Die Liste der Status-Meldungen kann gefiltert werden, sodass nur die Meldungen eines einzelnen Clients angezeigt werden.

5.3.4.3. Bearbeitung von Konfigurationen

Eine für das Fernmanagement elementare Funktion der Webseite ist die Bearbeitung von Konfigurationen der registrierten Clients. Zur Erstellung und Bearbeitung wird daher ein Formular zur Verfügung gestellt, über welches die einzelnen Felder der Konfiguration eingetragen werden können. Durch Absenden des Formulars wird ein HTTP-POST-Request an die entsprechende Controller-Funktion geschickt. Dort werden die eingegebenen Daten validiert und schließlich in die Datenbank eingetragen. In Anhang A.3 ist die Ansicht der Benutzer-Oberfläche für eine Client-Konfiguration dargestellt.

5.3.4.4. Download von Messdaten

Die Webseite enthält ein Formular, über das ein gefilterter Datensatz heruntergeladen werden kann. Die im Formular erfassten Daten werden an den Webservice weitergeleitet, um die Messdaten anzufordern (siehe 5.3.3.3). Dabei ist es ebenfalls möglich die gewünschten Messgrößen zu selektieren, zu filtern und zu sortieren. Außerdem kann eine bestimmte Anzahl von Messpunkte eingetragen werden. In Anhang A.3 ist ein Ausschnitt des Formulars zur Messdatenanfrage abgebildet.

5.4. Client

Für die Clients wird ein Betriebssystem generiert und in Betrieb genommen. Anschließend wird die Applikation entwickelt, welche das Fernmanagement auf den Clients ermöglicht.

5.4.1. Betriebssystem: Yocto Embedded Linux

Es wird ein eigenständiges Linux-basiertes Betriebssystem für die Verwendung auf den Clients generiert. Das hat den Vorteil gegenüber vorhandenen Linux-Distributionen, dass nur die Pakete im System enthalten sind, die für die Funktionalität der Clients tatsächlich benötigt werden. Auf diese Weise können die Ressourcen der verwendeten Hardware besser für die eigentliche Anwendung genutzt werden.

Zur Generierung des Betriebssystems wird das *Yocto-Project* verwendet. Das *Open-Source* Projekt stellt Werkzeuge und Methoden für die Generierung von benutzerdefinierten Linux-basierten Betriebssystemen zur Verfügung. Dabei ist das erzeuge Betriebssystem unabhängig von der Architektur der Zielhardware. Es kommt die Version 2.0 „Jethro“ zum Einsatz.

Das *Yocto-Project* stellt dazu Methoden bereit, mit denen Software-Pakete zu einem Betriebssystem kombiniert werden können. Dazu werden Anleitungen (*Recipes*) verwendet, in denen z.B. festgelegt ist, wo der Quellcode heruntergeladen wird, wie das Paket übersetzt wird und von welchen anderen Software-Paketen es abhängt. Das wichtigste Paket stellt dabei der *Linux-Kernel* dar, die anderen Pakete können z.B. Bibliotheken, Laufzeitumgebungen oder Werkzeuge sein. Die gewünschten Pakete werden in einer *User-Configuration* gesammelt. Die Hardware-spezifischen Einstellungen und Pakete (Board Support Package (BSP)) für den Raspberry Pi werden ebenfalls durch das *Yocto-Project* zur Verfügung gestellt.

Das *Yocto-System* generiert das Betriebssystem, indem es alle ausgewählten und die von diesen abhängigen Pakete mit einem geeigneten *Compiler* übersetzt. Der *Compiler* hängt dabei von der Ziel-Hardware ab und wird daher ebenfalls vom *Yocto-System* generiert.

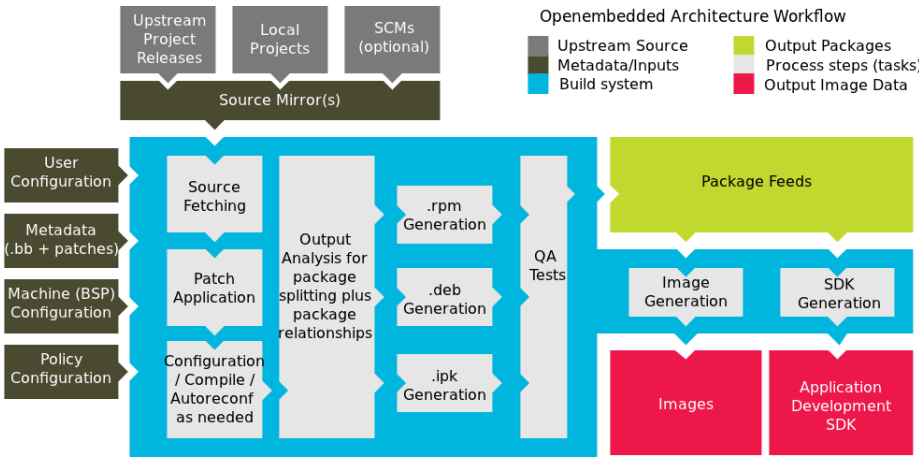

Abbildung 5.4.: Ablauf der Betriebssystemgenerierung mit dem *Yocto-Project* [15]

5.4.1.1. Besonderheiten des Betriebssystems

Nachfolgend werden die Besonderheiten erläutert, die das Betriebssystem aufweisen soll und welche Software zur Realisierung bestimmter Funktionen ausgewählt wurden.

Das Betriebssystem für die Clients wird „headless“, also ohne grafische Schnittstelle, konfiguriert. Die Interaktion mit dem Benutzer erfolgt über das Netzwerk. Dazu wird ein *WLAN-Access-Point* in das System integriert, über den eine Netzwerkverbindung mit dem System hergestellt werden kann. Ein *SSH*-Server erlaubt anschließend den Login auf dem Client.

Für die Kommunikation mit dem UMTS-Modem wird das *QMI*-Protokoll verwendet. Dies ist ein proprietäres Protokoll zur Kommunikation mit Modems des Herstellers *Qualcom*. Das *QMI*-Protokoll ist eine Alternative zur Verwendung von AT-Befehlen. Zur Herstellung der VPN-Verbindung wird außerdem die *OpenVPN*-Client-Software benötigt.

Als Laufzeitumgebung für die Client-Applikation, wird die Skriptsprache *Python 3* verwendet. Zusätzlich wird das *Python-Requests*-Paket für die komfortablen Verwendung von HTTP-Requests mit *Python* genutzt.

Um die Software auf dem Betriebssystem aktualisieren zu können, wird ein Paket-Management-System verwendet. Durch die Auswahl dieser Option erstellt *Yocto* während der Generierung des Betriebssystems automatisch ein Paket-*Repository*, welches alle Pakete enthält, aus denen das System besteht. Zusätzlich wird der Paket-Manager *OPKG* (Open Package Management) in das System integriert, der für die Verwaltung der Software auf dem System zuständig ist.

5.4.1.2. Anpassung an den verwendeten Client

Das *Yocto-Project* kann zunächst nur Systeme für x86-Architekturen oder Emulatoren generieren. Für den Betrieb auf dem Client sind spezifische Einstellungen und Anpassungen notwendig. Daher müssen einige Software-Pakete in das *Yocto-System* integriert werden.

Damit das Betriebssystem auf dem Client lauffähig ist, werden Hardware-spezifische Dateien und Pakete benötigt. Für *Yocto* ist dazu ein BSP für den RaspberryPi in den offiziellen *Yocto*-Quellen verfügbar [4]. Dieses bietet Unterstützung für die Modelle A, B, B+ des Raspberry Pi und den Raspberry Pi 2.

Für die Funktion als Datenlogger wird außerdem die vorhandene Datenerfassungssoftware des Clients in das *Yocto-System* integriert. Da die vorhandene Version für eine andere *Compiler*-Version entwickelt wurde, wird diese zunächst portiert, sodass diese auch mit dem von *Yocto* verwendeten *Compiler* übersetzt werden kann.

Weiterhin wird ein Paket erstellt, welches alle Client-spezifischen Einstellungen enthält. Dazu zählen unter Anderem die zu ladenden Geräte-Treiber, die Konfiguration der Netzwerkschnittstellen (WLAN-Access-Points und UMTS) und die Eintragung der Paket-*Repositories* für den Paket-Managers.

5.4.2. Client-Applikation

Die Client-Applikation wird in der Programmiersprache *Python* entwickelt. Dabei handelt es sich um eine universelle Skriptsprache, welche jedoch auch zur Implementierung von objektorientierter Software geeignet ist. *Python*-Code zeichnet sich durch eine gute Lesbarkeit und Übersichtlichkeit aus. Zusätzlich ist es für die meisten gängigen Betriebssysteme verfügbar, sodass die Plattformunabhängigkeit gegeben ist. Es kommt die Version 3.4.4 zum Einsatz.

Der Quellcode der Anwendung befindet sich in Anhang B.2. Weiterhin können Verwendungshinweise und Aufrufparameter dem Anhang A.1 entnommen werden.

5.4.2.1. Aufbau der Applikation

Die Applikation steuert das generelle Verhalten des Clients, z.B. bei der Messdatenerfassung. Dazu wird eine zentrale Konfigurationsdatei verwendet, in der alle Randbedingungen für den Betrieb des Clients eingetragen werden. Über den Webservice kann diese Konfiguration durch eine vom Anwender auf dem Server hinterlegte Konfiguration ausgetauscht werden. So wird das Verhalten des Clients aus der Ferne beeinflussbar.

[4] RaspberryPi BSP: http://git.yoctoproject.org/cgit/cgit.cgi/meta-raspberrypi/

Über eine entsprechende System-API hat die Applikation Zugriff auf System-Funktionen zur Herstellung der Internet-Verbindung und der Ermittlung des aktuellen System-Status. Dazu zählt die Steuerung des OpenVPN-Clients zur Verbindung mit dem Server, der Netzwerk-Geräte und des Paket-Managers. Alle Aktionen, die durch die Client-Applikation angestoßen werden, sind über eine Aufgaben-Planung organisiert. Die Planung erfolgt dabei ebenfalls über die Konfigurationsdatei und kann somit auch aus der Ferne aktualisiert werden.

Die Interaktionen mit dem Server werden durch die Ausführung von HTTP-Methoden durchgeführt. Konkret wird dafür die „requests“-Bibliothek von Kenneth Reitz [21] verwendet. Da die Webanwendung RESTful entworfen wurde, sind keine weiteren Protokolle notwendig.

Der Aufbau und die Funktionen der Applikation sind in der Abbildung 5.5 dargestellt.

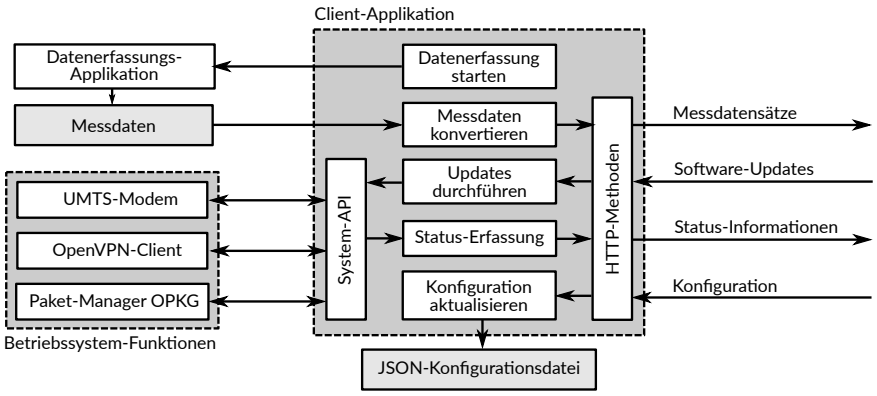

Abbildung 5.5.: Konzept der Management-Client-Applikation

5.4.2.2. Funktionsweise

Die Applikation wird automatisch nach dem Hochfahren des Datenloggers gestartet. Als erstes wird die externe Datenerfassungssoftware in einem separaten Prozess gestartet.

Während die Datenerfassung läuft, werden die Management-Aufgaben durchgeführt. Dazu zählen der Upload von Status-Meldungen, die Aktualisierung der Konfiguration, die Durchführung von Software-Updates und die Herstellung einer Server-Verbindung. Um eine größtmögliche Flexibilität zu erreichen, werden diese Aufgaben geplant und in einem separaten Thread ausgeführt. Auf diese Weise sind die Management-Funktion unabhängig von der eigentlichen Datenerfassung. Zur Planung werden in der Client-Konfiguration explizit Uhrzeiten angegeben, zu denen eine bestimmten Aufgabe durchgeführt werden soll. Durch die Aktualisierung kann so auch die Aufgabenplanung beeinflusst werden.

Nachdem die Datenerfassung beendet wurde, werden die erzeugten Messdaten mit einer separaten Konvertierungssoftware in das CSV-Format umgewandelt. Die Messdaten im CSV-Format werden anschließend über den Webservice an den Server übertragen.

Die Funktionsweise der Client-Applikation ist in der Abbildung 5.6 als Ablaufdiagramm dargestellt.

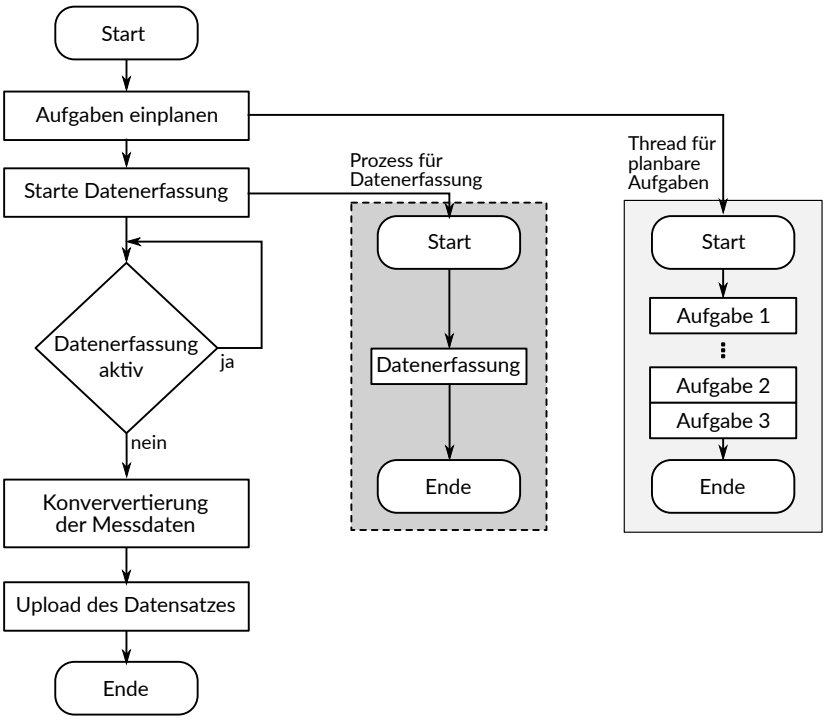

Abbildung 5.6.: Abläufe in der Client-Applikation

Nachfolgend werden die Funktionen und Eigenschaften der Applikation detailliert erläutert.

5.4.2.3. Konfigurationsdatei eines Clients

Die Konfigurationsdatei wird im JSON-Format auf dem Client gespeichert und besteht aus zwei Arten von Einträgen: persistente und nicht-persistente Optionen. Die nicht-persistenten Einstellungen sind jene, die über den Webservice aktualisiert werden können.

Zu den persistenten Einstellungen gehören in erster Linie die Netzwerkeinstellungen. Darin sind die zu verwendenden Netzwerkgeräte und Informationen zum VPN-Zugang zum Server angegeben. Hier werden auch die Zugangsdaten für die Mobilfunk-Verbindung (Access Point Name - APN) und die PIN für die SIM-Karte eingetragen. Diese sollen nicht über den

Webservice aktualisiert werden, denn im Falle eine Fehlkonfiguration wäre der Client vom Netzwerk ausgeschlossen.

```
"network": {
    "iface": "wwan0",
    "vpn_iface": "tun0",
    "vpn_service": "openvpn",
    "wwan_apn": "internet.apn.de",
    "wwan_pin": "1234"
}
```

Listing 5.1: Auszug aus der Konfigurationsdatei: Persistente Optionen

Alle weiteren Einstellungen werden unter der „dlmconfig“-Option (Datalogger Management Configuration) gespeichert. Hier werden alle Einstellungen vorgenommen, die das Verhalten des Clients widerspiegeln und die über das System verändert werden können. Wichtige Einträge sind hier die URLs für die Webservice-Interaktion, sowie die Datenerfassungssoftware. Weiterhin werden hier auch die Zeitpunkte für die Durchführung der nebenläufigen Aufgaben eingetragen.

Nachfolgend ist ein Auszug aus der Konfigurationsdatei dargestellt, welche die dynamisch änderbaren Optionen enthält.

```
"dlmconfig": {
    "active": "true",
    "serial": "123456",
    "data_upload_url": "http://urbandevelopment.gnet.haw-hamburg.de/dlms/api/data/client/1/dataset",
    "config_download_url": "urbandevelopment.gnet.haw-hamburg.de/dlms/api/config/client/1",
    "status_upload_url": "http://urbandevelopment.gnet.haw-hamburg.de/dlms/api/status/client/1/upload
        ",
    "worker_exec": "/usr/bin/gps-logger",
    "worker_output_path": "/home/root/databuffer",
    "postprocessing_exec": "/usr/bin/dataset2json",
    "postprocessing_output_path": "/home/root/datasets/json"
    "status_upload_schedule": "11:47",
    "config_download_schedule": "11:47",
    "maintenance_connect_schedule": "11:30",
    "maintenance_disconnect_schedule": "12:00",
    "timestamp": "2016/21/11 11:12:13",
}
```

Listing 5.2: Auszug aus der Konfigurationsdatei: Dynamische Optionen

5.4.2.4. Steuerung der Messdatenerfassung

Die Messdatenerfassung ist die eigentliche Aufgabe der Clients. Um eine maximale Flexibilität bei der zu verwendenden Datenerfassungs-Software zu gewährleisten, wird von der

Applikation aus keine weitere Interaktion durchgeführt. Es wird lediglich geprüft, ob der Prozess beendet wurde, also die Datenerfassung abschlossen ist.

5.4.2.5. Planbare Aufgaben

Die folgenden Aufgaben werden vom System zu den in der Konfiguration eingetragenen Uhrzeiten durchgeführt:

1. Erfassung des Status-Informationen
2. Aktualisierung der Konfigurationen
3. Durchführung Software-Updates
4. Serververbindung herstellen (Wartungsfenster)

1. Erfassung des Systemstatus

Die Applikation ist in der Lage, die für den Datenlogger-Betrieb wichtigen Systeminformationen zu sammeln. Dazu zählt die Ermittlung der vergangenen Zeit seit dem letzten System-Boot, sowie die mittlere System-Last. Außerdem werden Informationen über den verbleibenden freien Speicherplatz auf dem Client abgefragt und die Signalstärke des Mobilfunknetzes aufgezeichnet.

Die erfassten Daten werden anschließend in JSON-Format kodiert und an den Server übertragen. Es wird jedoch auch eine lokale Kopie von der Datei erstellt.

2. Aktualisierung der Konfigurationen

Über den Webservice wird die zum Client gehörige Konfiguration vom Server heruntergeladen. Anschließend wird die aktive Konfiguration mit den Werten aus der heruntergeladenen Datei aktualisiert. Die Konfiguration ist damit direkt nach dem Beenden dieser Aufgabe aktualisiert.

3. Durchführung von Software-Updates

Die Software auf dem Client-Betriebssystem wird durch den Paket-Manager OPKG (Open Package Management) verwaltet. Dieser bezieht Informationen zu verfügbaren Paketen und Updates aus Paket-*Repositories*. Mit Hilfe des Yocto-Systems wurde ein solches *Repository* erzeugt (siehe 5.4.1) und auf dem Server zur Verfügung gestellt.

In der Client-Konfiguration ist ein Feld vorgesehen, in welches die Namen von Software-Paketen eingetragen werden können. Die Applikation ruft den Paket-Manager beim Update-*Task* so auf, dass für diese Pakete ein *upgrade* ausgeführt wird. Die Pakete werden so auf die neuste Version aktualisiert oder installiert, wenn diese noch nicht installiert waren.

4. Serververbindung herstellen (Wartungsfenster)

Über die Konfigurations-Einträge „maintenance_connect“ und „maintenance_disconnect“ können zusätzliche Online-Zeiten für den Client konfiguriert werden. So entstehen Wartungsfenster, in denen der Client mit dem Server auch für längere Zeit verbunden bleibt.

5.4.3. Konvertierungssoftware für Messdaten

Wie bereits zuvor erläutert wurde, müssen die Messdaten in JSON- oder CSV-Format vorliegen, damit diese über den Webservice an den Server übertragen können. Die auf dem verwendeten Datenlogger implementierte Software erzeugt jedoch ein eigenes Datenformat. Daher wird eine Konvertierungssoftware implementiert, die einen BEEDeL-Datensatz in das CSV-Format umwandelt.

Hinweise zur Verwendung der Software befinden sich in Anhang A.2.

5.4.3.1. BEEDeL Datenformat

Die erzeugten Messdaten liegen in Form von Datensätzen mit variabler Länge vor. Ein Datensatz besteht dabei aus einem ZIP-komprimierten Ordner, der die gemessenen Rohdaten enthält. Jede Messgröße ist zeilenweise in einer Textdatei abgespeichert, wobei die Aufzeichnung zeilensynchron erfolgt. Somit kann die Zeilennummer in den Textdateien als Datenpunkt-Index interpretiert werden. Grundsätzlich sind diese Datensätze nur als vollständiger Datensatz gültig, da es nur einen Zeitstempel zu Beginn des Datensatzes gibt.

Die folgenden Dateien mit Rohdaten sind in einem BEEDeL-Datensatz enthalten

- ACC.txt: Beschleunigungen (3 Achsen)
- GYR.txt: Drehrate (3 Achsen)
- MAG.txt: Erd-Magnetfeld (3 Achsen)
- POINTS.bin: Anzahl der Messpunkte
- PR_TE.txt: Luftdruck und Temperatur
- Tacho.bin: Odometriesignal des Busses
- Time.bin: Startzeitpunkt des Datensatzes
- Halt.txt: Türsignal des Busses

Um die Daten später in der Datenbank zu speichern, müssen die Daten so umgewandelt werden, dass jeder Messpunkt allein gültig und identifizierbar ist. Jede Zeile muss daher mit einem Zeitstempel versehen werden, außerdem wird die Client-ID als zusätzliche Information integriert.

5.4.3.2. Umsetzung der Konvertierung

Die Konvertierungssoftware ist ebenfalls in *Python* implementiert. Das hat den Vorteil der Plattformunabhängigkeit, sodass die Software auch auf dem Server zur Datensatz-Konvertierung verwendet werden kann.

Die Applikation ist so implementiert, das lediglich ein Ordner angegeben wird, in dem sich die BEEDeL-Datensätze befinden. Das Programm findet anschließend alle Dateien, die eine „.zip“-Dateiendung besitzen und verarbeitet diese. So kann mit einem Aufruf eine ganze Reihe von Datensätzen konvertiert werden.

Zusätzlich ist die Konvertierung parallelisiert. Da die Datensätze nicht von einander abhängen, ist auch die Konvertierung unabhängig. Zur Parallelisierung wird das *multiprocessing*-Modul von *Python* verwendet. Die Anzahl an verfügbaren Prozessoren wird automatisch bestimmt und entsprechend viele Konvertierungen parallel ausgeführt.

Ablauf der Konvertierung eines Datensatzes

Zunächst muss der Datensatz dekomprimiert werden, um die Rohdaten auslesen zu können. Dann wird die Datei „POINTS.bin“ ausgelesen, um die Anzahl der im Datensatz enthaltenen Messdatenpunkte zu ermitteln. Im Anschluss wird die Startzeit aus der Datei „Time.bin“ entnommen. Die Startzeit ist in Form eines Datums mit Uhrzeit angegeben und benennt den Zeitpunkt des ersten Messwertpunktes. Da die Datenerfassung mit einer festen Frequenz von 10 Hz durchgeführt wird, kann mit Hilfe der Information über die Anzahl der Datenpunkte der Zeitstempel für jeden Datenpunkt berechnet werden.

Es wird nun aus den Rohdaten immer eine Zeile, beginnend bei Zeile 1, ausgelesen und zwischengespeichert. Eine Zeile repräsentiert so genau einen Datenpunkt, der alle Messwerte enthält. Im Anschluss wird der Zeitstempel berechnet. Bevor der Datenpunkt in die Ziel-Datei geschrieben wird muss dieser entsprechend formatiert werden.

Nachfolgend ist der Pseudocode für die Konvertierung eines Datensatzes angegeben.

```
1 Datensatz dekomprimieren
2 Anzahl der Messpunkte auslesen
3 Startzeit auslesen
4 FOR i=1 TO Anzahl Messpunkte DO:
5     Auslesen der Zeile i der Rohdaten
6     Zeitstempel berechnen
7     Messpunkt in das Ziel-Format konvertieren
8     Messpunkt in Zieldatei eintragen
```

Listing 5.3: Ablauf der Messdaten-Konvertierung

5.5. Bewertung der Realisierung

5.5.1. Vergleich mit den Anforderungen

Im folgenden werden die Anforderungen an das System aus dem Kapitel 3 mit den realisierten Funktionen vergleichen.

Anforderung	**erfüllt?**	**Bemerkung**
Zuverlässigkeit		
Erhaltung der Datenkonsistenz	Ja	–
Abgeschlossenheit der Daten-Operationen	Ja	–
Isolation der Daten-Operationen	Ja	–
Dauerhafte Speicherung der Daten	Ja	–
Kapazität		
min. 1,296 TB Speicher	Nein	Für den Test-Betrieb nur 300 GB notwendig
Client-Schnittstellen		
Einlesen von 7,2 GB Messdaten pro Tag	Ja	–
Auslesen von 50,4 GB Messdaten pro Tag	Ja	–
Datenabfrage		
Selektion der auszugebenden Messgrößen	Ja	–
Filterung der Messgrößen nach Wertebereichen	Ja	–
Ausgabe von Datensätzen variabler Größe	Ja	–

Tabelle 5.4.: Vergleich mit den Anforderungen an den Datenspeicher

Anforderung	**erfüllt?**	**Bemerkung**
Client-Schnittstellen		
Verarbeitung von Status-Nachrichten	Ja	–
Verarbeitung von Messdatensätzen	Ja	–
Bereitstellung von Konfigurationen für Clients	Ja	–
Bereitstellung von Software-Updates für Clients	Ja	–
Auswertungs-Schnittstellen		
Standardisiertes Format zum Datenaustausch	Ja	–
Bereitstellung von Messdaten	Ja	–
Bereitstellung von Status-Informationen	Ja	–
Bereitstellung von Konfigurationen	Ja	–

Benutzer-Schnittstelle		
Grafische Oberfläche	Ja	–
Übersicht aller registrierten Clients	Ja	–
Ansicht von Konfigurationen und Status-Meldungen der Clients	Ja	–
Bearbeitung der Konfiguration eines Clients	Ja	–
Anforderung von Messdaten aus dem Datenspeicher	Ja	–
Benutzer-Verwaltung mit Benutzer-Gruppen	Ja	–

Tabelle 5.5.: Vergleich mit den Anforderungen an die Server-Applikation

Anforderung	**erfüllt?**	**Bemerkung**
Betriebssystem		
Embedded Linux	Ja	–
enthält nur die für den Betrieb notwendigen Software-Pakete	Ja	–
Bereitstellung der Schnittstellen zur Hardware (Treiber)	Ja	–
Laufzeitumgebung für Client-Applikation	Ja	–
Möglichkeit zur Herstellung einer Internetverbindung	Ja	–
Paket-Manager zur Software-Verwaltung	Ja	–
Client-Applikation		
Start und Überwachung der Software zur Messdatenerfassung	Ja	–
Upload von Status-Nachrichten auf den Server	Ja	–
Download und von Konfigurationen vom Server	Ja	–
Durchführung von Software-Updates über den Server	Ja	–
Konvertierung der Datensätze in ein standardisiertes Format	Ja	–
Upload der Messdatensätze auf den Server	Ja	–

Tabelle 5.6.: Vergleich mit den Anforderungen an den Client

5.5.2. Realisierbarkeit für ähnliche Problemstellungen

Das System wurde prototypisch für das in 3.1 genannte Szenario entwickelt. Doch ist der Einsatz des Systems auch für ähnliche Problemstellungen denkbar. Die für die Realisierung

ausgewählten Software-Komponenten wurden daher auch unter dem Aspekt der Verfügbarkeit und verschiedenen Betriebssystemen ausgewählt.

Weiterhin wurden alle Applikationen in Programmiersprachen (PHP, Python) implementiert, welche für unterschiedliche Betriebssysteme verfügbar sind. Auch die verwendete MySQL-Datenbank läuft auf verschiedenen Systemen. Hier muss lediglich das Datenmodell entsprechend der Problemstellung angepasst werden.

Das für die Clients generierte Linux-Betriebssystem kann leicht für eine andere Client-Hardware konfiguriert werden. Die Messdatenerfassung ist vom implementierten System unabhängig. Die gewünschte Datenerfassungssoftware kann in der Konfiguration eingestellt werden und wird durch die Client-Applikation lediglich gesteuert. Durch die frei konfigurierbaren Aufgaben für Status-Meldungen, Konfigurationsaustausch und Wartungsfenster können verschiedene Betriebsstrategien mit dem realisierten System umgesetzt werden.

Das System kann daher gut für ähnliche Problemstellungen ohne eine vollständige Neuimplementierung verwendet werden.

6. Einsatz des Systems

Nach der Realisierung soll nun eine reale Analyse von Messdaten mit Hilfe des entwickelten System durchgeführt werden. Dazu werden zunächst bereits vorhandene Messdaten in das System migriert. Anschließend werden die Daten für eine statistische Analyse verwendet.

6.1. Migration vorhandener Messdaten

Im Laufe des Forschungsprojekts „BEEDeL“ wurde bereits eine große Menge an Messdaten erzeugt. Die Datenmenge beläuft sich auf ca. 15000 Messstunden in 2000 Datensätzen. Insgesamt belegen die komprimierten Datensätze in dieser Form 21 GB Speicher. Die Daten sollen nun in das System migriert werden, um die Analyse zu vereinfachen.

Dazu werden die Datensätze zunächst mit Hilfe der in Abschnitt 5.4.3 implementierten Konvertierungssoftware in das CSV-Format umgewandelt. Die erstellten CSV-Datensätze belegen zusammen ca. 80 GB Speicher.

Nach der Konvertierung sollen die Daten in die Datenbank eingetragen werden. Dazu wird der in Abschnitt 5.3.3 implementierte Webservice verwendet. Mit einem Skript werden die Messdatensätze nacheinander über einen HTTP-POST-Request an die Webapplikation übergeben, welche die Daten automatisiert und im Hintergrund in die Datenbank einträgt.

Das Eintragen der 80 GB Rohdaten in die Datenbank dauerte auf dem verwendeten Server insgesamt 144,8 h (ca. 6 Tage). Die Anzahl der Datenpunkte in der Datenbank beläuft sich auf ca. 500 Mio Einträge.

6.2. Statistische Auswertung von Fahrzyklen von Nahverkehrsbussen

Anhand der aufgezeichneten Verkehrsdaten sollen die Fahrzyklen von Stillstand bis Stillstand statistisch analysiert werden. Dazu wird ein Modell entwickelt, welches einen Fahrzy-

klus vereinfacht repräsentiert. Anschließend wird das Modell auf die gemessenen Fahrzyklen angewendet und die Parameter werden statistisch analysiert.

6.2.1. Motivation

Für die Planung einer Elektrobuslinie spielen nicht nur zu planende Ladeinfrastruktur eine große Rolle, sondern auch die elektrische Antriebstechnik. Für die Entwicklung von entsprechend leistungsstarken Akkumulatoren ist es wichtig die Lebensdauer vorhersagen zu können. Durch die Kenntnis über typische Fahrzyklen, kann ermittelt werden, wie stark die Batterie im Mittel bei der Beschleunigung belastet wird. Ebenso können durch Wissen über typische Brems-Beschleunigungen, Rückschlüsse auf die zu erwartende Energie-Rekuperation getroffen werden. Mit Hilfe des mittleren Fahrzyklus können daher Akkumulatoren mit realistischen Lade- und Entladeströmen zur Lebensdauervorhersage belastet werden.

Ein weiterer Anwendungsfall ist die Beurteilung der mechanischen Belastung des gesamten Busses.

6.2.2. Ziel der Auswertung

Ziel ist die Entwicklung eines Referenz-Fahrzyklus für einen Nahverkehrsbus. Der Referenzzyklus wird in drei Bereich unterteilt: Beschleunigung, Fahrt und Bremsen. Für jeden dieser Bereiche werden Mittelwerte und Standardabweichungen angegeben. Dabei werden sowohl die Werte für Beschleunigungen und Geschwindigkeiten analysiert, als auch die zeitliche Dauer der einzelnen Sektionen.

6.2.3. Modellbildung für einen Referenzzyklus

Im folgenden wird ein Modell für einen Fahrzyklus entwickelt. Dazu werden Parameter für jede Sektionen ausgewählt, die anschließend aus den realen Messdaten bestimmt werden.

Das Modell des vereinfachten Fahrzyklus wird durch sechs Parameter beschrieben: Die Beschleunigungen beim Anfahren und Bremsen, die Geschwindigkeit in der Fahrphase und die Dauer der jeweiligen Phasen.

Die Beschleunigungs- und Bremsphase werden mit einem Polynom ersten Grades angenähert, während die Fahrphase durch einen konstanten Wert (Polynom nullten Grades) modelliert wird. Setzt man alle Sektionen aneinander, so entsteht ein trapezförmiges Geschwindigkeitsprofil.

Die Abbildung 6.1 zeigt den Geschwindigkeits- und Beschleunigungsverlauf im Modell des Fahrzyklus.

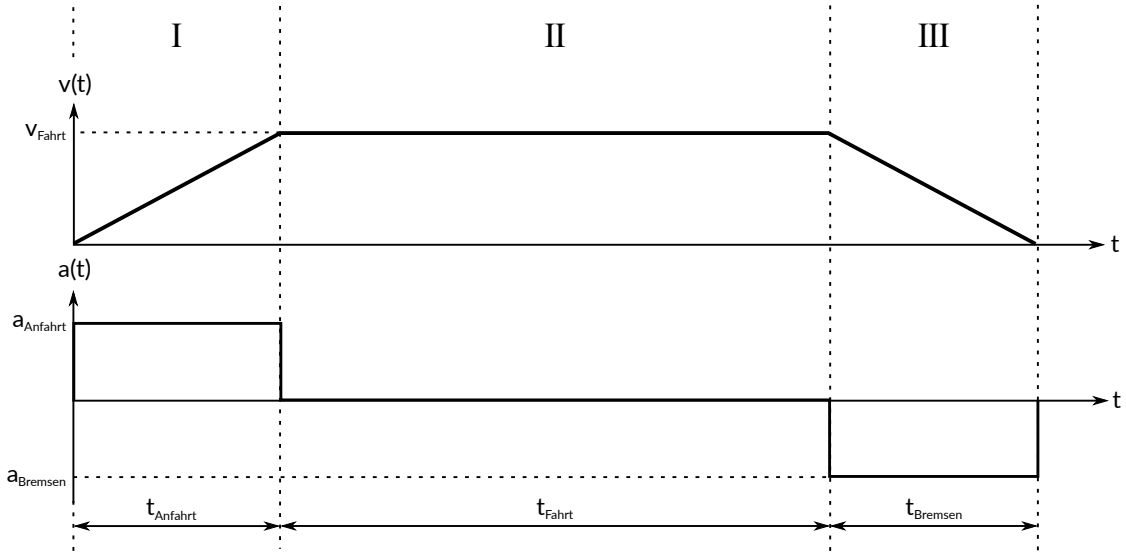

Abbildung 6.1.: Geschwindigkeits- und Beschleunigungsverlauf im Modell des Fahrzyklus

6.2.4. Durchführung der Messdatenauswertung

Um einen mittleren Referenzzyklus zu entwickeln, werden alle Datensätze und die darin gemessenen Fahrzyklen analysiert. Die Datensätze werden über den in 5.3.3 beschriebenen Webservice heruntergeladen. Zur Analyse der Messdatensätze wird *MathWorks MATLAB* verwendet.

Der Pseudo-Code für die Auswertung der Messdaten eines Datensatzes ist im folgenden dargestellt.

```
1 Berechne Geschwindigkeitsprofil aus Odometriesignal
2 Extrahiere Fahryklen
3 FOREACH Zyklus OF Fahrzyklen DO:
4     Trenne Sektionen
5     FOREACH Sektion of Sektionen DO:
6         Näherung mit Polynom
7         Bestimme Näherungsfehler
8         Speichere Länge, Polynom, Fehler
```

Listing 6.1: Ablauf der Messdatenanalyse für einen Datensatz

6.2.4.1. Ermittlung des Geschwindigkeitsprofils

Das Geschwindigkeitsprofil des Busses wird mit Hilfe des aufgezeichneten Tacho-Signals bestimmt. Der Sensor ist direkt an den Antriebsstrang des Busses gekoppelt und erzeugt alle 0,25 m einen Impuls. Diese Impulse werden während der Datenerfassung gezählt und mit einer festen *Samplerate* von 10 Hz aufgezeichnet. Auf diese Weise repräsentiert das Signal den zurückgelegten Weg des Busses mit einer Quantisierung von 25 cm.

Durch Ableitung des Odometriesignals kann die Geschwindigkeit bestimmt werden. Für die Umrechnung in $m \cdot s^{-1}$ wird mit dem Faktor $4 \cdot 10^{-1}$ multipliziert ($4\ Pulse = 1\ m$ und $10\ Zeitintervalle = 1s$).

$$s(t) = n_{Pulse} \tag{6.1}$$

$$v(t) = \frac{ds}{dt} \tag{6.2}$$

$$v(t) = \frac{n_{Pulse}}{0.1} \cdot \frac{4}{10} \frac{m}{s} \tag{6.3}$$

Da das Odometriesignal mit nur 4 Pulsen auf 1 m aufgelöst ist, wird das Signal durch einen gleitenden Mittelwert über 10 Werte geglättet.

6.2.4.2. Extraktion der Fahrzyklen

Aus dem Geschwindigkeitsprofil müssen nun die einzelnen Fahrzyklen extrahiert werden. In einem Datensatz sind im Normalfall mehrere Fahrzyklen enthalten. Standzeiten sollen nicht analysiert werden.

Um die Zyklen zu finden, wird der Geschwindigkeitsverlauf in ein binäres Format umgewandelt: Alle Zeitpunkte, zu denen die Geschwindigkeit größer als 0 m/s ist, werden eine logische 1, alle anderen Werte eine logische 0. Anschließend werden die steigenden und fallenden Flanken in dem binären Geschwindigkeitsprofil detektiert. Alle Zeitpunkte, an denen eine steigende Flanke detektiert wurde, zeigen so den Beginn eines Fahrzyklus an, während fallenden Flanken das Ende eines Zyklus beschreiben.

Dadurch werden alle Fahrzyklen im Datensatz bestimmt, jedoch auch solche, die nicht für Auswertung geeignet sind. Ein Fahrzyklus wird nur dann für die weitere Analyse verwendet, wenn dieser eine Gesamtdauer von mindestens 2 Sekunden besitzt.

Durch die Filterung des Tacho-Signals mit einem Mittelwertfilter, entstehen vereinzelt Filterartefakte. Ein einzelner Störimpuls der Länge 1 wird durch diesen Filter auf eine Länge von 10 Werten (1 Sekunde) vergrößert. Durch die Art der Zyklus-Detektierung würde diese Störung ebenfalls als Zyklus erkannt werden.

6.2.4.3. Näherung eines Fahrzyklus durch das Modell

Das Modell soll nun den realen Geschwindigkeitsverlauf eines Fahrzyklus möglichst gut nachempfinden. Zunächst wird der Fahrzyklus daher in die drei Sektionen unterteilt. Anschließend wird jede Sektion durch ein Polynom angenähert und der Näherungsfehler ermittelt.

Trennung der Sektionen

Es muss definiert werden, wann die Beschleunigungsphase beendet ist und wann die Bremsphase beginnt. Zur Trennung der Sektionen wird ein Geschwindigkeitsschwellwert verwendet. Es wird definiert, dass die Beschleunigungsphase beendet ist, sobald dieser Schwellwert das erste mal überschritten wird. Der Beginn der Bremsphase wird durch das letztmalige unterschreiten des Schwellwerts definiert.

Um die optimale Höhe des Schwellwerte einzustellen, wurde ein Teildatensatz, bestehend aus 10 Datensätzen, analysiert. Bei der Näherung der Zyklen durch das Modell wurde dabei der Schwellwert von 50% in 1%-Schritten bis 99% verschoben und die resultierenden Abweichungen ausgewertet. Anschließend wurde der Mittelwert aus den Schwellwerten gebildet, die zu einer minimalen Abweichung im jeweiligen Datensatz geführt haben.

Im Mittel erzielte ein Schwellwert von 89,5% der maximalen Geschwindigkeit in einem Zyklus die geringste Abweichung. Daher wird dieser Schwellwert für alle Zyklen verwendet.

Die Abbildung 6.2 zeigt einen realen Fahrzyklus und den ermittelten Schwellwert.

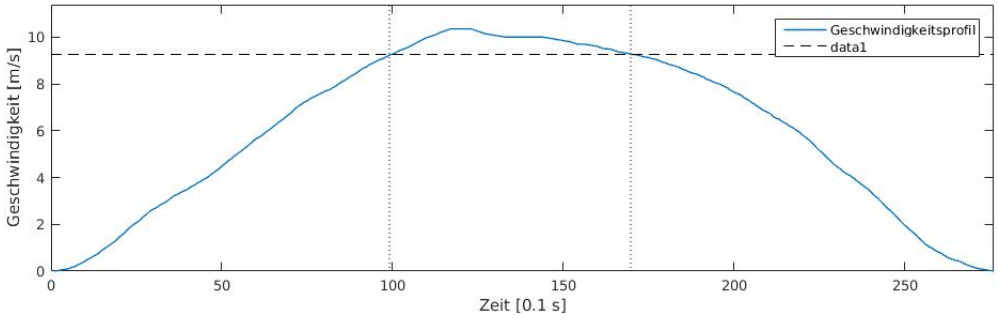

Abbildung 6.2.: Fahrprofil und berechneter Schwellwert für die Trennung der Sektionen

Näherung der Sektionen durch Polynome

Die einzelnen Sektionen werden durch Polynome angenähert. Dazu wird der Grad des Polynoms definiert und anschließend mit Verfahren der minimalen Quadrate die Parameter des Polynoms bestimmt.

Für die Beschleunigungsphase wird als Randbedingung festgelegt, dass das Polynom durch den Punkt (0,0) laufen muss. Das Polynom, welches die Bremsphase annähert, muss außerdem durch den Endpunkt des Fahrzyklus laufen.

Zur Durchführung der Näherung wird die *MATLAB*-Funktion *polyfix*[1] von Are Mjaavatten verwendet. Diese berechnet die Näherung für Polynome beliebige Grades, jedoch können auch Punkte angegeben werden, durch die das resultierende Polynom laufen muss. Die ermittelten Polynom-Parameter werden für jeden Zyklus separat abgespeichert. Außerdem wird die Länge der Sektionen ermittelt und ebenfalls gespeichert.

Das Ergebnis der Näherung des in 6.2 gezeigten Fahrzyklus ist in der Abbildung 6.3 dargestellt.

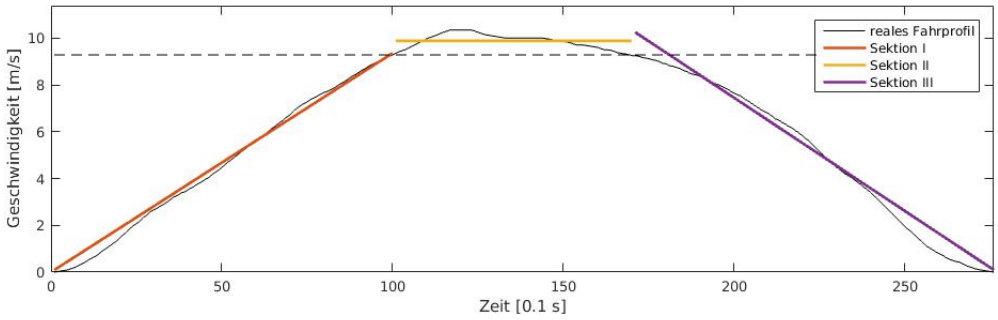

Abbildung 6.3.: Näherung der Sektionen des Fahrprofils durch Polynome

Bestimmung des Näherungsfehlers

Zur Beurteilung der Güte der Näherung wird der mittlere quadratische Fehler, auch Root Mean Square (RMS)-Fehler, für jede Sektion bestimmt. Dieser wird mit der in 6.4 angegebenen Gleichung für jede Sektion berechnet und gespeichert.

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x_i})^2} \tag{6.4}$$

[1] http://www.mathworks.com/matlabcentral/fileexchange/54207-polyfix-x-y-n-xfix-yfix-xder-dydx-

n ist die Anzahl der Datenpunkte der jeweiligen Sektion, die Werte x_i entsprechen denen des realen Fahrprofils und $\bar{x}_i$ sind die Werte der Näherung. Das Ergebnis ist ein Zahlenwert, der zur Bewertung der Näherung verwendet werden kann.

6.2.5. Ergebnisse

Insgesamt wurden 463958 Fahrzyklen aus den Messdaten extrahiert und durch das Modell angenähert. Nachfolgend werden valide Ergebnisse ausgewählt und deskriptive Parameter bestimmt. Für jede Sektion werden anschließend die Verteilungen der ermittelten Werte in Form von Histogrammen dargestellt.

Selektion der Ergebnisse nach RMS-Fehler

Für die statistische Auswertung werden die Ergebnisse verwendet, deren RMS-Fehler einen Grenzwert nicht überschreitet. So wird gesichert, dass nur die Fahrzyklen analysiert werden, für die das entwickelte Modell eine ausreichende Näherung darstellt.

Die folgende Abbildung 6.4 zeigt die Prozentzahl verbleibender Ergebnisse in Abhängigkeit von dem maximal zugelassenen RMS-Fehler:

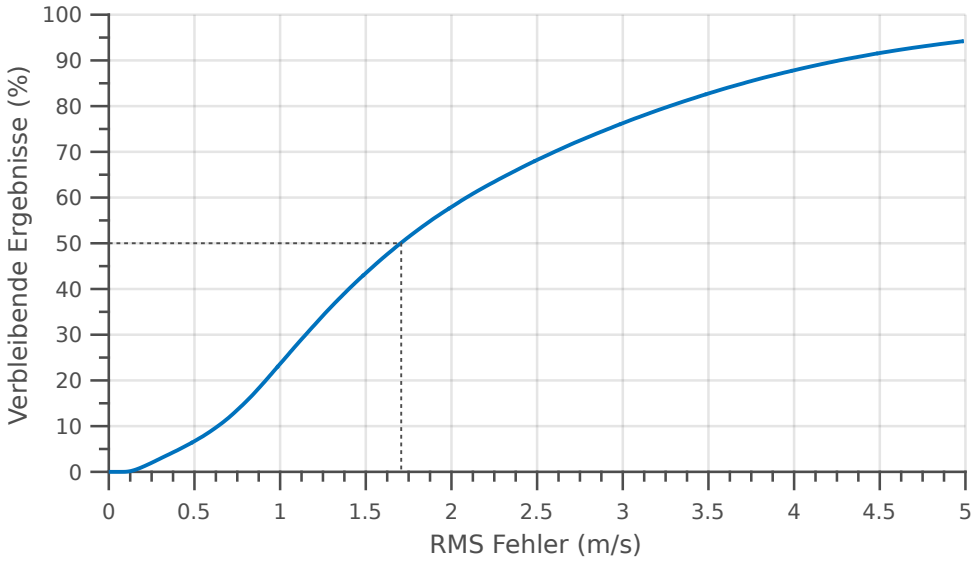

Abbildung 6.4.: Verbleibende Ergebnisse in Abhängigkeit vom maximal zugelassenen RMS-Fehler

Es wird gefordert, dass 50% der Ergebnisse verbleiben müssen, daher wird der RMS-Fehler auf einen Wert von 1,7 m/s begrenzt. Nach der Selektion verbleibt eine Anzahl von 231531 modellierten Fahrzyklen für die statistische Auswertung.

Deskriptive Parameter

Mit Hilfe der folgenden Parameter werden die Ergebnisse statistisch erfasst. Diese deskriptiven Parameter werden anschließend für die Entwicklung des Referenzzyklus verwendet.

Die Ergebnisse werden auf Normalverteilung durch den *Kolmogorov-Smirnov-Test* [2] (K-S-Test) geprüft. Die Verteilungen aller Parameter bestehen den Test und werden daher als normalverteilt betrachtet. Anschließend werden der arithmetische Mittelwert (Gl. 6.5) und die Standardabweichung (Gl. 6.6) berechnet.

$$\bar{x}_{arithm} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.5}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{6.6}$$

Die folgende Tabelle 6.1 zeigt die Ergebnisse der Berechnung der Parameter für alle Sektionen.

Sektion	**Parameter 1**		**Parameter 2**	
Anfahrphase (I)	$a_{1,mittel}$ $a_{1,\sigma}$	$0.72\ ms^{-2}$ $0.26\ ms^{-2}$	$t_{1,mittel}$ $t_{1,\sigma}$	$10.04\ s$ $5.38\ s$
Fahrphase (II)	$v_{2,mittel}$ $v_{2,\sigma}$	$7.23\ ms^{-1}$ $3.47\ ms^{-1}$	$t_{2,mittel}$ $t_{2,\sigma}$	$8.26\ s$ $7.91\ s$
Bremsphase (III)	$a_{3,mittel}$ $a_{3,\sigma}$	-0.77 ms^{-2} $0.31\ ms^{-2}$	$t_{3,mittel}$ $t_{3,\sigma}$	$9.81\ s$ $4.95\ s$

Tabelle 6.1.: Mittelwerte und Standardabweichungen aller ermittelten Modellparameter

Aus den Standardabweichungen lassen sich die Maximal- und Minimalwerte für jeden Parameter des Modells berechnen.

[2] Ausführliche Beschreibung siehe [13, S. 461ff]

6.2.5.1. Beschleunigungsphase

Verteilung der Beschleunigungswerte

Die Verteilung der Beschleunigungen in der Anfahrtsphase (Abb. 6.5) ist asymmetrisch und fällt zur rechten Seite steiler ab als auf der linken Seite. Es können keine Ausreißer in der Verteilung beobachtet werden.

Signifikante Werte sind zwischen 0,05 m/s^2 und 1,4 m/s^2 erkennbar. Mit einer Anzahl von 19000 Beobachtungen, treten dabei Beschleunigungen im Bereich von 0,9 m/s^2 am häufigsten auf. Es ergibt sich ein arithmetischer Mittelwert von 0.72 m/s^2 und eine Standardabweichung von 0,26 m/s^2.

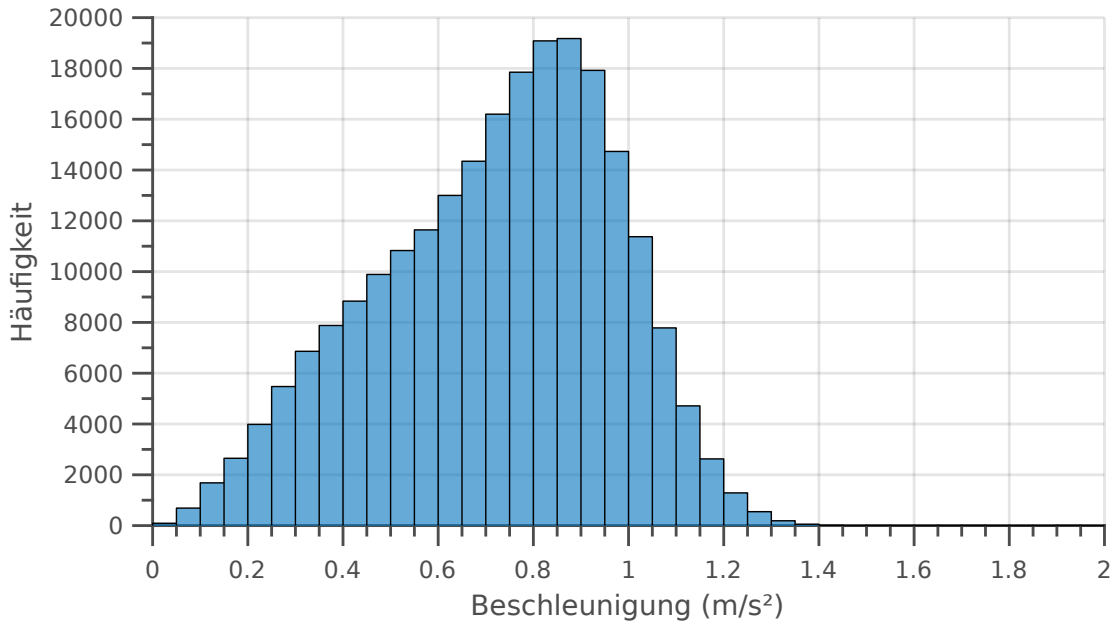

Abbildung 6.5.: Histogramm der Beschleunigungen in der Anfahrtsphase (I)

Verteilung der Dauer der Beschleunigungsvorgänge

Für die Dauer der Beschleunigungsvorgänge (Abb. 6.6) ergibt sich eine deutlich linksschiefe Verteilung. Neben dem Maximum im Bereich zwischen 6 s und 7 s, kann ein lokaler Wendepunkt zwischen 10 s und 12 s beobachtet werden. Der Mittelwert beläuft sich auf 10.04 s und die Standardabweichung beträgt 5.38 s.

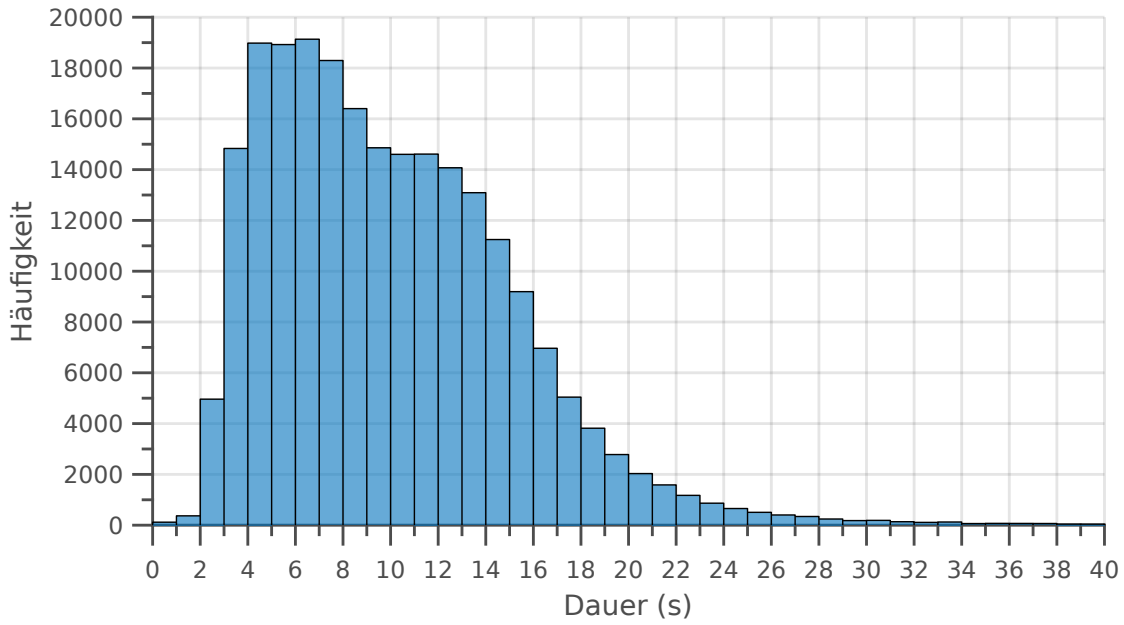

Abbildung 6.6.: Histogramm der Dauer der Anfahrtsphase (I)

6.2.5.2. Fahrphase

Verteilung der Geschwindigkeitswerte

Die Verteilung der Geschwindigkeiten zwischen Anfahrt- und Bremsphase unterscheidet sich deutlich von den anderen Histogrammen und ist in Abb. 6.7 dargestellt. So ist die Verteilung sehr breit, hat aber einen nahezu symmetrischen Charakter.

In der breiten Verteilung sind zwischen 1 m/s (3.6 km/h) und 16 m/s (57.6 km/h) Werte mit einer signifikanten Häufigkeit vorhanden. Die Spitze des Histogramms befindet sich bei 7,6 m/s (27.36 km/h), es sind jedoch weitere lokale Maxima bei 1.5 m/s (5.4 km/h), sowie bei 10 m/s (36.0 km/h) erkennbar. Der Mittelwert ergibt sich zu 7.23 m/s (26.0 km/h) und die Standardabweichung umfasst 3.47 m/s (12.49 km/h).

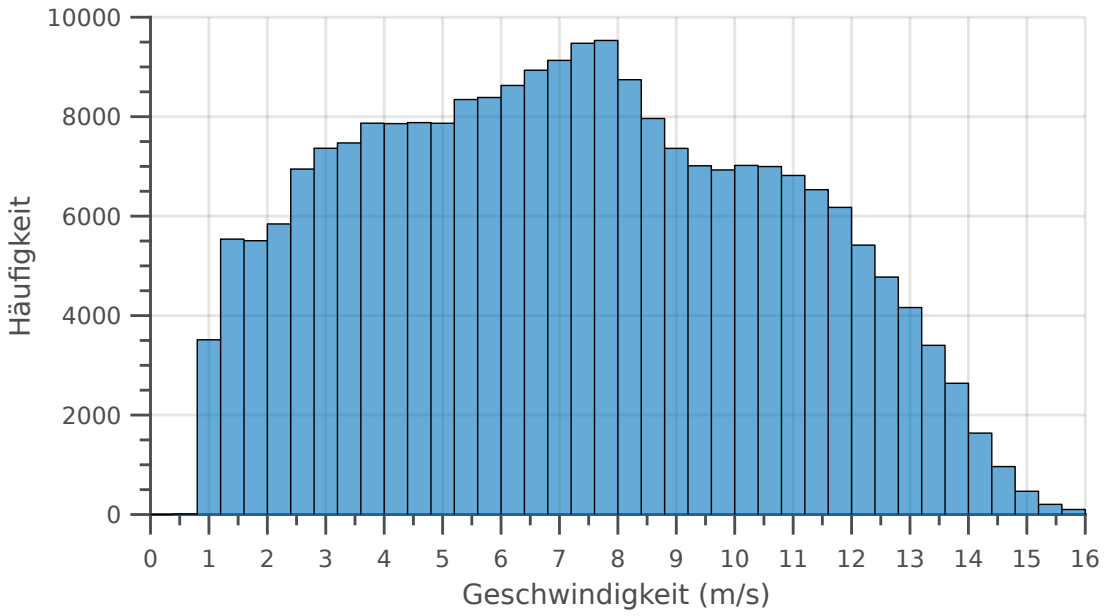

Abbildung 6.7.: Histogram der Geschwindigkeiten in der Fahrphase (II)

Verteilung der Dauer der Fahrten mit konstanter Geschwindigkeit

Die Häufigkeiten der Dauer der Fahrphase in Abb. 6.8 weisen eine Form mit einem einzigen globalen Maximum auf. Auffällig ist dabei die rechtsseitige Schiefe und die positive Wölbung (Spitzgipfligkeit). Die Verteilung fällt zur linken Seite sehr steil ab, während die Häufigkeiten zur rechten Seite langsam abklingen. Auch in dieser Verteilung sind keine Ausreißer zu beobachten.

Es treten Werte zwischen 0 s und 40 s mit nennenswerter Häufigkeit auf. Die Verteilung hat ein Maximum bei 4 s, der Mittelwert beträgt 8.26 s. Die Standardabweichung von 7.91 s ist dabei auffällig hoch, was jedoch durch die starke Asymmetrie der Verteilung zu begründen ist.

Die Dauer der Fahrphase hängt dabei mit dem gewählten Schwellwert für die Trennung der Sektionen zusammen.

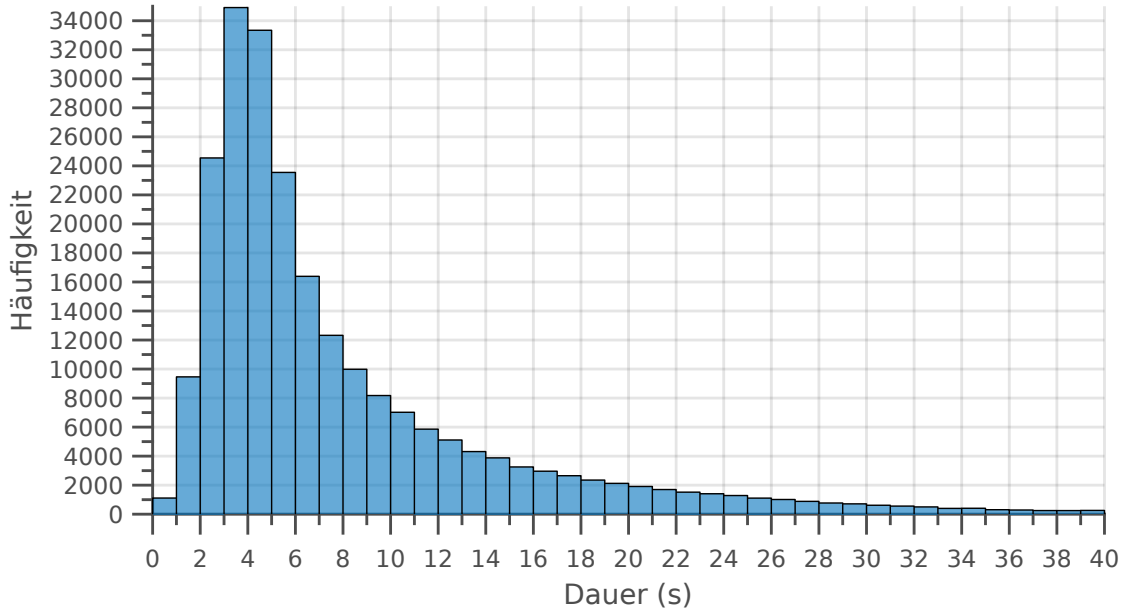

Abbildung 6.8.: Histogramm der Dauer der Fahrphase (II)

6.2.5.3. Bremsphase

Verteilung der Beschleunigungswerte

Die Verteilung der Bremsvorgänge ist leicht asymmetrisch und hat eine linksseitige Schiefe. Das Histogramm der Verteilung ist in Abb. 6.9 dargestellt.

Werte mit einer relevanten Häufigkeit liegen dabei zwischen -2 m/s^2 und nahe 0 m/s^2. Das Maximum der Verteilung befindet sich bei -0,82 m/s^2. Für den Mittelwert wurde -0.77 m/s^2 berechnet, die Standardabweichung beträgt 0.31 m/s^2.

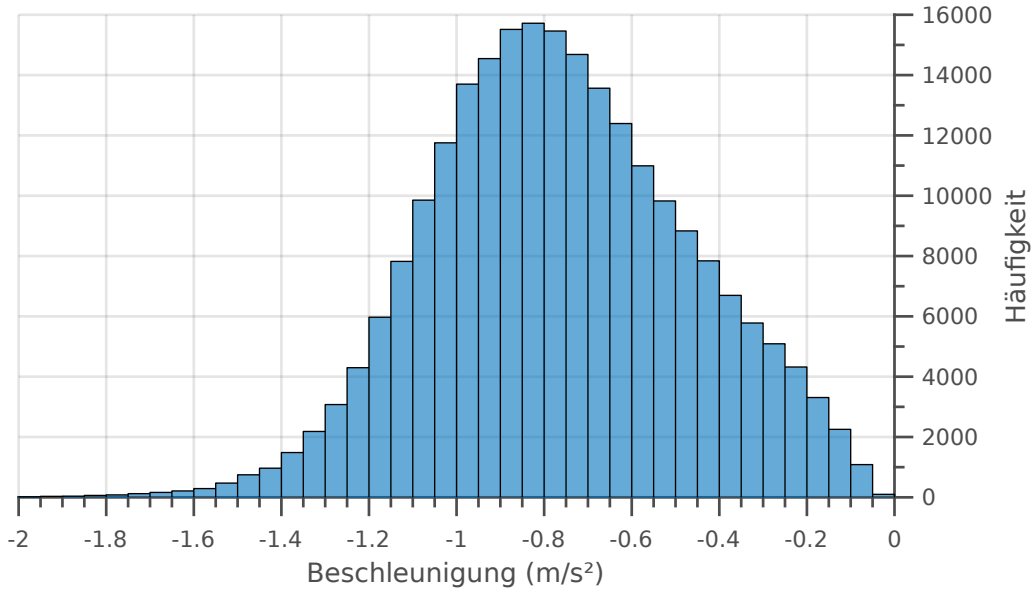

Abbildung 6.9.: Histogramm der Beschleunigungen in der Bremsphase (III)

Verteilung der Dauer der Beschleunigungsvorgänge

Die Verteilung der Dauer der Bremsphase hat weitgehend den Charakter einer Normalverteilung und ist in Abbildung 6.10 dargestellt. Es ist eine leicht Asymmetrie mit rechtsseitiger Schiefe festzustellen.

Das globale Maximum liegt bei 8 s, nennenswerte Häufigkeiten können zwischen 1 s und 30 s beobachtet werden. Für diese Verteilung ergibt sich ein Mittelwert von 9.81 s und eine Standardabweichung von 4.95 s.

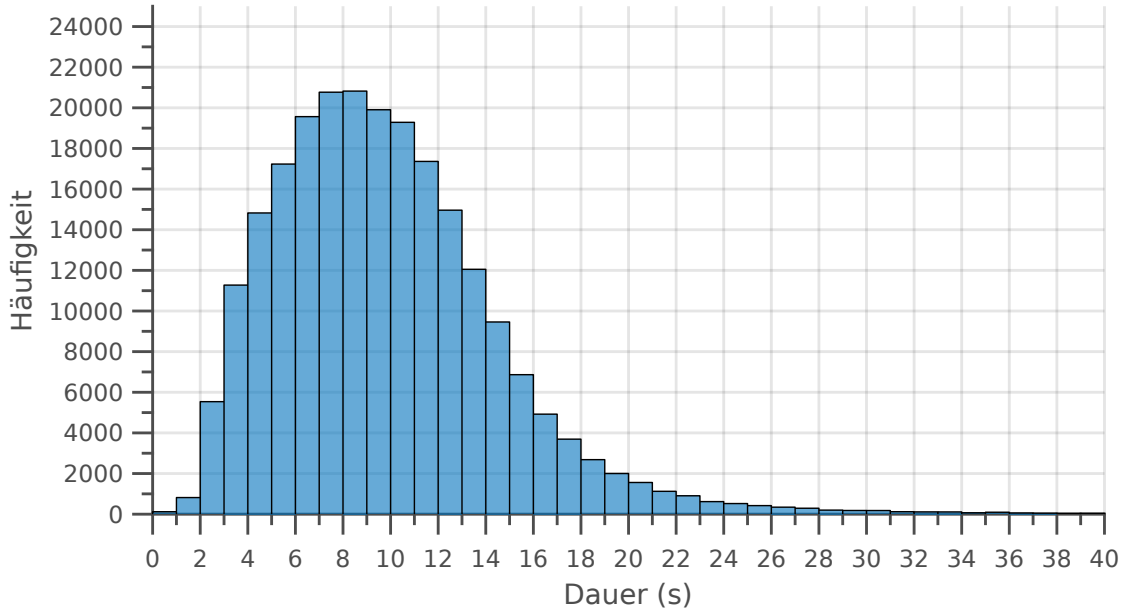

Abbildung 6.10.: Histogramm der Dauer der Bremsphase (III)

6.2.6. Auswertung

Aus den Ergebnissen wird nun ein Referenz-Fahrzyklus entwickelt. Dazu werden die berechneten Mittelwerte und Standardabweichungen der zuvor dargestellten Ergebnisse verwendet. Es werden die Dauern und Geschwindigkeitsprofile der einzelnen Phasen separat betrachtet und anschließend zu einem Referenzzyklus kombiniert.

6.2.6.1. Gesamtdauer eines Fahrzyklus

Für die Verteilung der Gesamtdauer eines Fahrzyklus werden die Dauern aller Sektionen addiert und anschließend die Häufigkeiten bestimmt.

Das ermittelte Histogramm (Abb. 6.11) ist asymmetrisch und rechtsschief, die Häufigkeiten fallen auf den linken Seite deutlich stärker ab, als auf der rechten Seite. Sehr kurze Zyklen unter 5 s treten nur mit einer geringen Häufigkeit auf, signifikante Werte sind ab 5 s bis 7 s erkennbar. Die meisten Beobachtungen wurden für eine Gesamtzyklusdauer zwischen 20 s und 22,5 s erfasst.

Der Mittelwert dieser Verteilung beträgt 28.11 s und die Standardabweichung wurde zu 14.11 s berechnet.

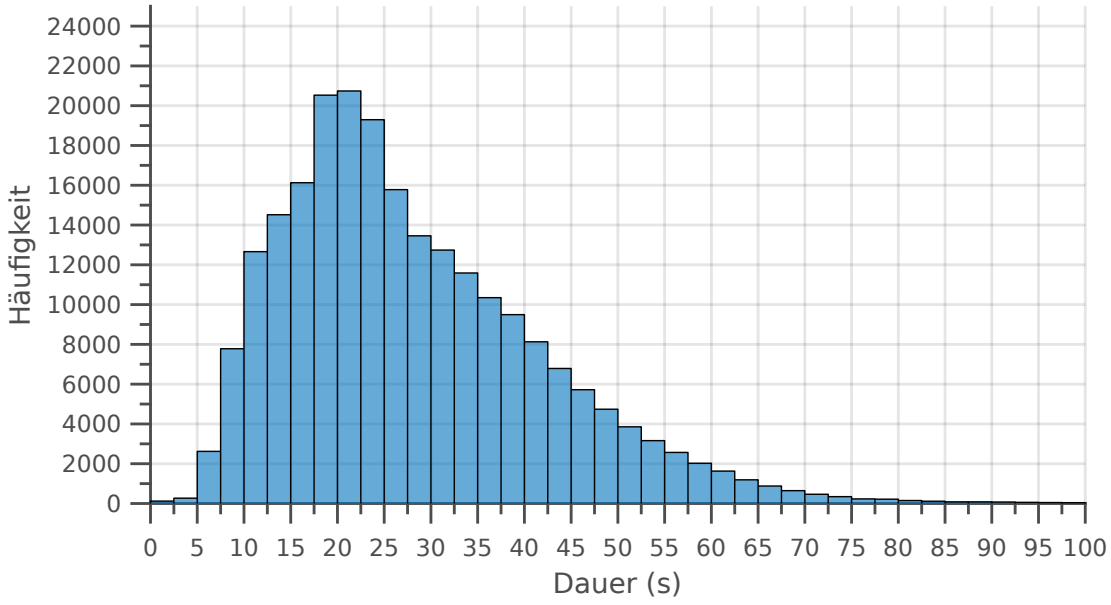

Abbildung 6.11.: Histogramm der Dauer eines gesamten Fahrzyklus

6.2.6.2. Mittlerer Fahrzyklus

Nachfolgend wird das mittlere Geschwindigkeitsprofil für einen Gesamtzyklus aus den Mittelwerten der einzelnen Sektionen ermittelt. Die Abschnitte des Profils werden mit den Formeln 6.7 bis 6.9 berechnet. Der Index m steht für den Mittelwert und die Indizes 1 - 3 für die jeweilige Sektion.

$$v_{1,m}(t) = \int_0^{t_{1,m}} a_{1,m}(t)dt \tag{6.7}$$

$$v_{2,m}(t) = v_{2,m}(t) = const. \tag{6.8}$$

$$v_{3,m}(t) = \int_{t_{3,m}}^{t_{3,m}} a_{3,m}(t)dt \tag{6.9}$$

Die Abbildung 6.12 zeigt das resultierende Geschwindigkeitsprofil für die Mittelwerte aller ermittelten Modellparameter.

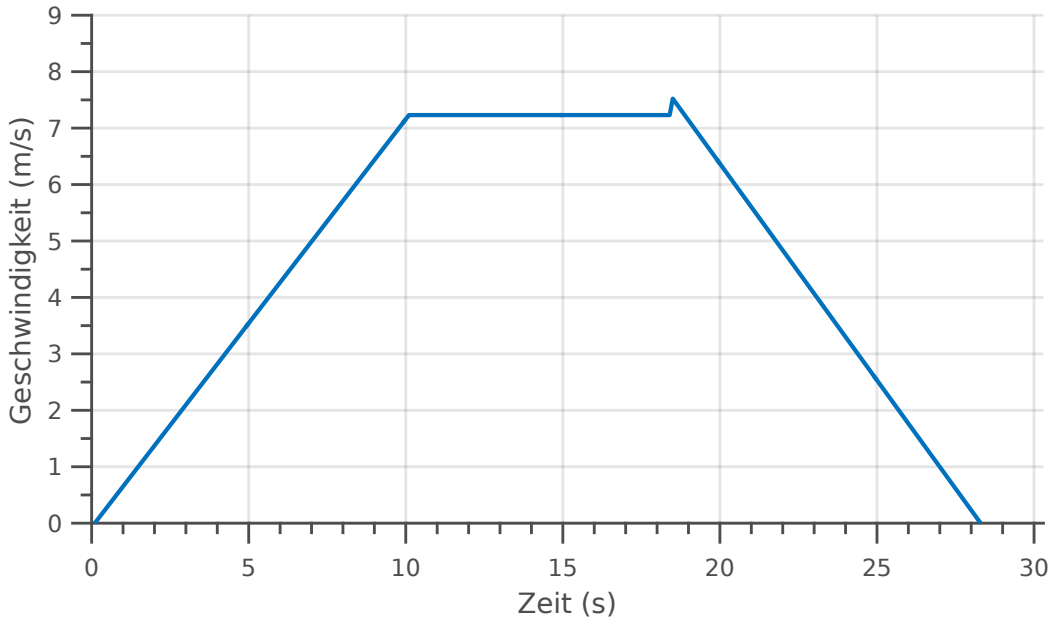

Abbildung 6.12.: Geschwindigkeitsprofil des mittleren Fahrzyklus

Es ist erkennbar, dass der mittlere Fahrzyklus dem Modellzyklus sehr Nahe kommt.

Nach der Beschleunigungsphase mit 0,72 m/s^2 wird eine Geschwindigkeit von 7,23 m/s (ca. 26 km/h) erreicht. Diese Geschwindigkeit wird für 8.23 s gehalten, bevor die letzte Phase des Zyklus beginnt. In der Bremsphase wird mit einer Verzögerung von 0,77 m/s^2 bis zum Stillstand gebremst.

Der mittlere Fahrzyklus hat eine Gesamtdauer von 28,1 s. Die Beschleunigungsphase dauert 10 s, während die Bremsphase mit 9,8 s etwas kürzer ausfällt. Dies kann auch in den Beschleunigungswerte der Anfahrts- und Bremsphase erkannt werden.

Auffällig ist der Sprung zwischen Fahrphase und Bremsphase. Dieser entsteht dadurch, dass im Modell keine Randbedingung vorgegeben wurde, die einen direkten Übergang zwischen den einzelnen Sektion fordert.

6.2.6.3. Minimal- und Maximalwerte

Für die Entwicklung des Referenzzyklus wird nicht nur der mittlere Fahrzyklus analysiert, sondern auch die Maximal- und Minimalwerte.

Mit Hilfe der 1σ-Standardabweichung werden die maximalen und minimalen Werte der Modellparameter bestimmt.

Dazu werden zunächst die Minimal- und Maximalwerte für die Beschleunigungen in Phase I und III, sowie für die Geschwindigkeit in Phase II berechnet. Auch für die Dauern der einzelnen Phasen werden die Minimal- und Maximalwerte ermittelt. Die Berechnung erfolgt mit den Formeln 6.10 und 6.11. Es werden die Beträge der Parameter verwendet.

$$\begin{aligned} Maximum &= Betrag\ des\ Mittelwerts + Standardabweichung(1\sigma) \\ x_{max} &= |\bar{x}| + \sigma(x) \end{aligned} \tag{6.10}$$

$$\begin{aligned} Minimum &= Betrag\ des\ Mittelwerts - Standardabweichung(1\sigma) \\ x_{min} &= |\bar{x}| - \sigma(x) \end{aligned} \tag{6.11}$$

Die Tabelle 6.2 zeigt die berechneten Durchschnitts-, Minimal- und Maximalwerte für alle Sektionen.

Sektion	**Parameter 1**		**Parameter 2**	
Anfahrphase (I)	$a_{1,min}$	$0.47\ ms^{-2}$	$t_{1,min}$	$4.66\ s$
	$a_{1,mittel}$	$0.72\ ms^{-2}$	$t_{1,mittel}$	$10.04\ s$
	$a_{1,max}$	$0.98\ ms^{-2}$	$t_{1,max}$	$15.42\ s$
Fahrphase (II)	$v_{2,min}$	$3.76\ ms^{-1}$	$t_{2,min}$	$1.35\ s$
	$v_{2,mittel}$	$7.23\ ms^{-1}$	$t_{2,mittel}$	$8.26\ s$
	$v_{2,max}$	$10.70\ ms^{-1}$	$t_{2,max}$	$16.17\ s$
Bremsphase (III)	$a_{3,min}$	$-0.46\ ms^{-2}$	$t_{3,min}$	$4.86\ s$
	$a_{3,mittel}$	$-0.77\ ms^{-2}$	$t_{3,mittel}$	$9.81\ s$
	$a_{3,max}$	$-1.08\ ms^{-2}$	$t_{3,max}$	$14.76\ s$
Gesamtzyklus			$t_{g,min}$	$10.87\ s$
			$t_{g,mittel}$	$28.38\ s$
			$t_{g,max}$	$46.35\ s$

Tabelle 6.2.: Minimal- und Maximalwerte aller ermittelten Modellparameter

6.2.6.4. Entwicklung eines Referenzzyklus

Für die Entwicklung eines Referenzzyklus müssen nun die ermittelten Parameter ausgewertet werden. Dazu müssen die Parameter geeignet miteinander kombiniert werden. Theoretisch sind 18 solcher Kombinationen möglich. Jede dieser Kombination führt dabei zu einem anderen Fahrzyklus, welcher entsprechend auf seine Eigenschaften untersucht werden muss. Um einen vollständig definierten Referenzzyklus zu erhalten, müssen daher alle Kombinationen ausgewertet und die Ergebnisse überlagert werden.

In dieser Arbeit wird die Auswertung exemplarisch anhand einzelner Parameterkombinationen durchgeführt. In den folgenden Abbildungen ist jeweils zusätzlich der mittlere Zyklus (blau) zur Orientierung abgebildet.

Die ersten zwei Parameter-Kombinationen sollen Aufschluss über die maximalen Beschleunigungen und Geschwindigkeiten eines Fahrzyklus geben. Dazu werden die Dauern der Sektionen auf die Mittelwerte festgelegt und die Beschleunigungen und Fahrgeschwindigkeiten variiert.

- Zyklus 1 (Abb. 6.13, rot):
  - Sektion I: mittlere Dauer: $t_{1,mittel}$, minimale Beschleunigung: $a_{1,min}$
  - Sektion II: mittlere Dauer: $t_{2,mittel}$, minimale Fahrgeschwindigkeit: $v_{2,min}$
  - Sektion III: mittlere Dauer: $t_{3,mittel}$, minimale Bremsbeschleunigung: $a_{3,min}$
- Zyklus 2 (Abb. 6.13, gelb):
  - Sektion I: mittlere Dauer: $t_{1,mittel}$, maximale Beschleunigung: $a_{1,max}$
  - Sektion II: mittlere Dauer: $t_{2,mittel}$, maximale Fahrgeschwindigkeit: $v_{2,max}$
  - Sektion III: mittlere Dauer: $t_{3,mittel}$, maximale Bremsbeschleunigung: $a_{3,max}$

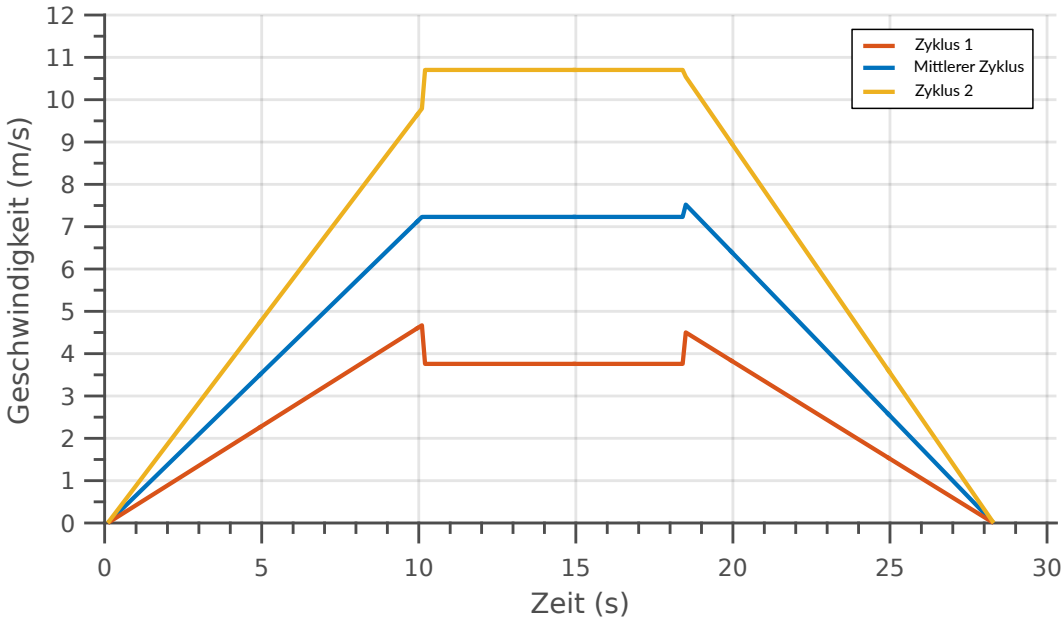

Abbildung 6.13.: Fahrzyklen bei mittlerer Phasendauer und Variation der Beschleunigungen

Um auch Informationen über die mögliche Variation der Dauern der jeweiligen Sektionen zu erhalten, werden zwei weitere Parameter-Kombinationen untersucht. Dabei werden die

Beschleunigungen in Anfahrts- und Bremsphase auf die Mittelwerte festgelegt, während die Dauer der einzelnen Sektionen so gewählt wird, sodass diese zu einem minimal kurzen und einem maximal Langen Fahrzyklus führen.

- Zyklus 3 (Abb. 6.14, rot):
  - Sektion I: minimale Dauer: $t_{1,min}$, mittlere Beschleunigung: $a_{1,mittel}$
  - Sektion II: minimale Dauer: $t_{2,min}$, minimale Fahrgeschwindigkeit: $v_{2,min}$
  - Sektion III: minimale Dauer: $t_{3,min}$, mittlere Bremsbeschleunigung: $a_{3,mittel}$
- Zyklus 4 (Abb. 6.14, gelb):
  - Sektion I: maximale Dauer: $t_{1,max}$, mittlere Beschleunigung: $a_{1,mittel}$
  - Sektion II: maximale Dauer: $t_{2,max}$, maximale Fahrgeschwindigkeit: $v_{2,max}$
  - Sektion III: maximale Dauer: $t_{3,max}$, mittlere Bremsbeschleunigung: $a_{3,mittel}$

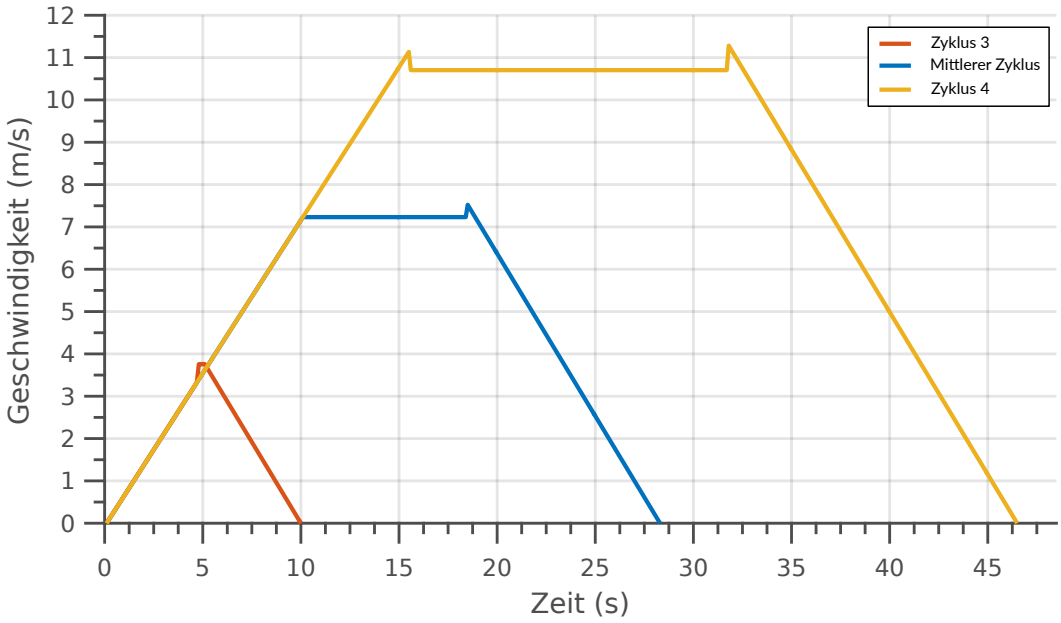

Abbildung 6.14.: Fahrzyklen bei mittleren Beschleunigungen und Variation der Phasendauer

In allen vier Kombinationen werden grundsätzlich valide Fahrzyklen erzeugt. Es gibt Sprünge beim Übergang der einzelnen Sektionen. Die Sprünge haben eine Höhe von ± 1 m/s und fallen somit nicht ins Gewicht. Die Sprünge können damit begründet werden, dass im Modell keine Randbedingung für einen fließenden Übergang der Sektionen angegeben wurde.

Um den Referenzzyklus zu entwickeln, müssen Randbedingungen für die Kombination der Parameter definiert werden. Das Ziel ist Informationen darüber zu erhalten, welche Parameter voneinander abhängen und welche unabhängig sind.

Die Parameter-Kombinationen 1 und 2 geben Aufschluss über den direkten Zusammenhang zwischen Beschleunigung und Fahrgeschwindigkeit, sowie zwischen Bremsbeschleunigung und Fahrgeschwindigkeit. Dies gilt, solange alle Sektionen eine mittlere Dauer aufweisen.

In der Kombination 3 und 4 wurden mittlere Beschleunigungen verwendet, um den längsten und kürzesten Gesamtzyklus zu realisieren. Die Fahrgeschwindigkeit richtet sich dabei nach den Dauern der jeweiligen Sektion (Minimale Dauer der Beschleunigungs- und Bremssektionen bedingt eine minimale Fahrgeschwindigkeit). Diese Informationen sind jedoch auch nur gültig, solange mittlere Beschleunigungen in Anfahrts- und Bremsphase verwendet werden.

Für eine vollständige Spezifikation eines Referenzzyklus müssen noch weitere Kombinationen ausgewertet werden. Dazu zählt die unabhängige Variation von Beschleunigungswerten in der Anfahrts- und Bremsphase, sowie die unabhängige Variation von Dauern in allen Phasen.

6.2.6.5. Zyklierung von Akkumulatoren mit dem Referenzzyklus

Der entwickelte Referenzzyklus kann nun verwendet werden, um Akkumulatoren für Elektrobusse zu „zyklieren“. Dabei werden die Akkumulatoren mit vielen Lade- und Entladezyklen belastet, um Alterserscheinungen wie Kapazitätsverluste zu untersuchen. Das ist notwendig, um die voraussichtliche Lebensdauer des Akkumulators zu bestimmen.

Wird eine Lebensdauer eines Busses mit 15000 Betriebsstunden angenommen (nach [31]), so kann die Anzahl der Referenzzyklen berechnet werden, die in dieser Zeit durchlaufen wird. Mit dieser Information kann ein Akkumulator so zykliert werden, dass ein realistischer Fahrbetrieb eines Busses simuliert wird.

$$\begin{aligned} Anzahl\ Referenzzyklen &= \frac{Lebensdauer}{Dauer\ des\ Referenzzyklus} \\ &= \frac{15000\ h \cdot 3600}{28.4\ s} = 1901408,45 \approx \underline{1901408} \end{aligned} \tag{6.12}$$

Während der gesamten Lebensdauer werden demnach ca. 1.9 Mio. Standardzyklen durchlaufen. Geht man davon aus, dass ein Nahverkehrsbus im realen Betrieb durchschnittlich

einen Zyklus pro Minute fährt und pro Tag 8 h im Einsatz ist, entspricht das einer Gesamteinsatzdauer von ca. 10 Jahren (vgl. [23]).

Im folgenden wird das zum Referenzzyklus gehörige Stromverlaufs-Profil qualitativ bestimmt. Es wird angenommen, dass die Batterie nur während der eigentlichen Fahrt belastet wird. Belastungen durch Nebenaggregate und weitere Verluste werden nicht berücksichtigt.

Dazu wird angenommen, dass der Ladestrom der Batterie sich proportional zur Beschleunigung verhält. Demnach wird die Batterie während der Beschleunigung mit einem konstanten Strom I_1 belastet, in der Fahrphase fließt ein sehr kleiner Laststrom I_2 zur Erhaltung der Geschwindigkeit und in der Bremsphase wird die Batterie durch einen konstanten Ladestrom I_3 aufgeladen. Die Abbildung 6.15 zeigt den qualitativen Stromverlauf während des mittleren Fahrzyklus. Die Zeiten t_1 bis t_3 stellen die Dauern der einzelnen Phasen I bis III dar.

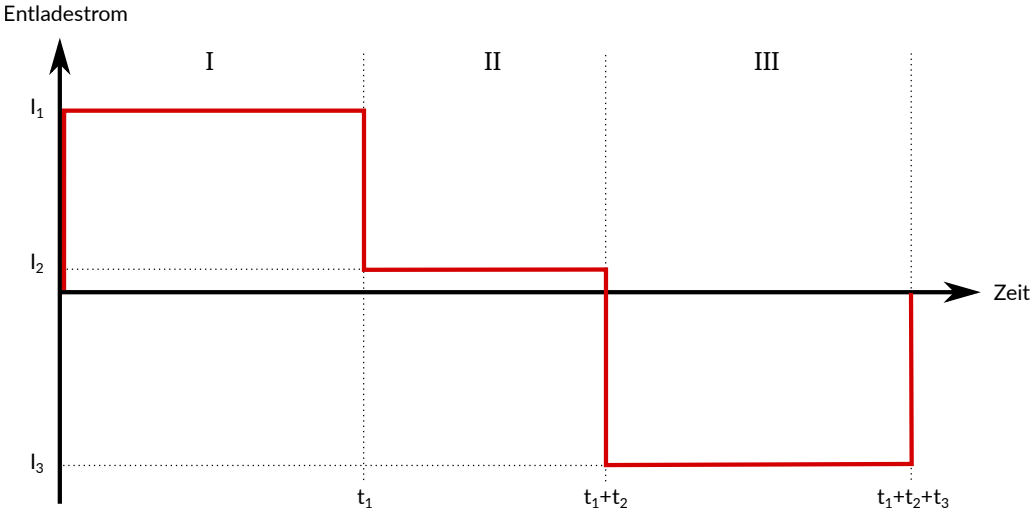

Abbildung 6.15.: Qualitativer Verlauf des Entladestroms der Batterie während des Referenzzyklus

Die Fläche unter der Kurve stellt die entnommene und zugeführte Energiemenge dar. Um einen idealen Fahrzyklus ohne Verluste zu simulieren, muss der Stromverlauf daher so gewählt werden, dass die Gesamtfläche unter der Kurve gleich null ist.

Mit Hilfe der zuvor berechneten Standardabweichungen und Mittelwerte lassen sich auf diese Weise Stromverläufe erzeugen, welche unterschiedliche Fahrzyklen repräsentieren. Dabei kann das Maß angegeben werden, in dem die erzeugten Fahrzyklen vom mittleren Standardzyklus abweichen. Durch Erzeugung entsprechenden Parameterverteilung können Serien von Batteriezyklen unterschiedlicher Länge mit realistischem Charakter durchgeführt werden.

7. Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Management- und Analysesystem für mobile Datenerfassungssysteme realisiert. Das System erlaubt Ferndiagnose und -Management der Datenerfassungssysteme (Clients) und verwaltet die erzeugten Messdaten, die über eine standardisierte Schnittstelle zu Auswertungszwecken bereitgestellt werden. Im Anschluss wurde das System für die statistische Analyse von Fahrzyklen von Bussen im öffentlichen Nahverkehr eingesetzt und aus den Ergebnissen ein Referenzfahrzyklus entwickelt.

Das entwickelte Gesamtsystem ist nach einer Client-Server-Struktur aufgebaut. Die mobilen Datenerfassungssysteme haben aus energietechnischen Gründen nur temporär Zugang zu einem Server. Daher speichert der Server die Daten für die Verwaltung und Fernwartung, die von Anwendern oder Clients übertragen wurden. Die Clients senden bei einer aktiven Verbindung Statusinformationen und Messdaten an den Server, außerdem werden Konfigurationen und Software-Updates heruntergeladen, welche das Verhalten des Clients beeinflussen. Über eine Benutzer-Schnittstelle können die Daten auf dem Server eingesehen, heruntergeladen oder aktualisiert werden.

Zur Datenübertragung zwischen den Teilnehmern des Systems werden Web-Technologien verwendet. Im Mittelpunkt steht ein Webservice auf dem Server, der den Datenaustausch in standardisierten Formaten ermöglicht. Über diese Schnittstelle tauschen die Clients Daten mit dem Server aus, außerdem können Messdaten an den Anwender übertragen werden. Um die Daten im Gesamtsystem zu verwalten, verwendet der Server ein relationales Datenbanksystem. Die Benutzer-Oberfläche ist in Form einer Website realisiert, wodurch der Anwender lediglich einen Browser zur Verwendung des Systems benötigt.

Für die Clients wurde ein einheitliches Linux-basiertes Betriebssystem generiert und eine Anwendung implementiert, welche das Management des Clients ermöglicht. Diese Applikation steuert die Messdatenerfassung und führt, Kommunikation mit dem Server durch. Das Verhalten des Clients wird durch eine Konfigurationsdatei beeinflusst, welche über den Webservice aktualisiert werden kann. Um die Messdaten an den Server zu übertragen, werden diese mit einer Konvertierungssoftware auf den Clients in ein standardisiertes Format umgewandelt.

Im Anschluss an die Realisierung wurde das System für eine exemplarische Analyse verwendet. Dazu wurden 15000 h Messdaten aus dem Forschungsprojekt „BEEDeL“ in das

System migriert und anschließend statistisch ausgewertet. Der Zugriff auf die Messdaten erfolgte dabei über den Webservice des realisierten Systems.

Das Ziel der Auswertung war die Entwicklung eines mittleren Fahrzyklus eines Nahverkehrsbusses mit statistischer Grundlage. Dazu werden alle Fahrzyklen aus den Messdaten extrahiert und durch ein Modell angenähert. Die Modellparameter werden anschließend statistisch analysiert und aus den Ergebnissen ein Referenzzyklus entwickelt. Dieser Zyklus beschreibt das mittlere Fahrverhalten eines Busses über einen Zeitraum von 15000 Betriebsstunden im öffentlichen Nahverkehr. Dieser Referenzzyklus kann so zur Zyklierung von Akkumulatoren für Elektrobusse verwendet werden.

Es wurde gezeigt, dass das realisierte Gesamtsystem geeignet ist, um die Datenerhebung und Konvertierung auf den Clients zu steuern und Fernwartungsaufgaben über eine grafische Benutzer-Oberfläche durchzuführen. Das System wurde weiterhin für die zentrale Speicherung realer Messdaten in einer Datenbank und deren Bereitstellung über einen Webservice im Zuge einer exemplarischen Datenauswertung erfolgreich eingesetzt.

8. Ausblick

8.1. Fernaktivierung und -Deaktivierung von Clients

In der aktuellen Umsetzung werden die Clients vor dem Beginn des Betriebs mit der notwendigen Software ausgestattet und anschließend installiert. Sobald der Client eingeschaltet wird, startet dann bereits die Messdatenerfassung. Die Clients müssen auf diese Weise direkt nach der Installation aktiviert werden und liefern ab diesem Zeitpunkt bereits Messdaten an den Server.

Manchmal ist es jedoch gewünscht die Datenerfassung erst zu einem bestimmten Zeitpunkt zu starten, um nachvollziehbare Messdaten zu erhalten. Es wäre in diesem Fall wünschenswert die Datenerfassung auf den Clients durch eine Fernaktivierung zu starten. Dies könnte über den vorhandenen Server oder über einen weiteren Kommunikationskanal (z.b. SMS) realisiert werden. Die Clients könnten dann so konfiguriert sein, dass diese erst nach der Fernaktivierung die eigentliche Konfiguration oder Datenerfassungssoftware herunterladen. So können alle Clients mit der gleichen Software und Konfiguration ausgeliefert werden. Ebenso kann eine Fern-Deaktivierung gewünscht sein, wenn ein Client keine validen Daten erzeugt oder die Datenerfassung insgesamt beendet wird.

8.2. Automatisierte Messdatenauswertung

Die manuelle Auswertung ist nur bis zu einer begrenzten Anzahl an Messdaten sinnvoll. Die Datenübertragung zwischen Server und der lokalen Auswertungssoftware kann, je nach Verbindung zum Server, sehr viel Zeit in Anspruch nehmen.

Alternativ könnte daher die Datenauswertung auch automatisiert auf dem Server stattfinden. Dies hätte den Vorteil, dass lediglich die Ergebnisse vom Server zum Anwender übertragen werden müssen. Durch eine regelmäßige Auswertung können zusätzlich Tendenzen in den Messdaten erkannt werden. Mit Hilfe von Frameworks wie *Hadoop*, die sich auf die Auswertung und Verwaltung von sehr großen Datenmengen spezialisiert haben, lassen sich solche Auswertungen automatisieren.

8.3. Optimierung der Datenspeicherung

Das auf dem Server eingesetzte relationale Datenbanksystem kann die Messdaten von 100 Datenerfassungssystemen verwalten, jedoch sind die Datenabfragen unter Umständen sehr langsam. Dies ist insbesondere der Fall, wenn eine Abfrage mit einer Einschränkung des Wertebereichs für mehrere Messgrößen durchgeführt werden soll. Für sehr lange Tabellen ist dies sehr ineffizient. Da das verwendete Datenmodell sehr einfach ausfällt, können auch andere Datenbanksysteme für die Verwaltung der Daten verwendet werden.

Nicht-relationale Datenbanksysteme (NoSQL-Datenbanken) sind für die Verwaltung von derart großen Datenmengen sehr gut geeignet, insbesondere, wenn viele Daten von gleichem Format gespeichert werden sollen. Bei solchen Datenbanksystemen wird das Datenmodell nicht durch die Datenbank, sondern durch die Daten selbst vorgegeben. Die Daten werden dabei z. B. als einfache *Key/Value*-Paare abgespeichert (*Key/Value Store*), wobei die *Values* Daten von unterschiedlichster Art sein können. So ist ein schneller Datenzugriff möglich, jedoch ist nicht vorgegeben, welches Format die zurückgegeben Daten aufweisen. In diesem Fall muss die Auswertungssoftware die Abfrageergebnisse auf Konsistenz prüfen.

8.4. Optimierung der statistischen Analyse von Fahrzyklen

Durch das verwendete Modell werden alle Fahrzyklen abgebildet, die ein trapezförmiges Geschwindigkeitsprofil aufweisen. Im realen Fahrbetrieb treten jedoch eine Vielzahl unterschiedlicher Zyklus-Formen auf. Beispielsweise kann die Fahrphase durch eine plötzliche Bremsung unterbrochen werden oder die Beschleunigungs- und Bremsphase mehrere lokale Maxima oder Minima aufweisen.

Um auch solche Zyklen in die Analyse mit einfließen zu lassen, müssen dem Modell weitere Parameter hinzugefügt werden. Dazu könnten die einzelnen Sektionen des Geschwindigkeitsprofils zusätzlich unterteilt werden, um ein detaillierteres Abbild zu erhalten. Ein weiterer Ansatz wäre die Klassifikation der unterschiedlichen Fahrzyklus-Formen. Im Anschluss können dann die einzelnen Klassen der Zyklen durch unterschiedliche Modelle angenähert werden.

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

Abkürzungsverzeichnis

ACID	Atomicity, Consitency, Isolation, Durability
API	Application Programming Interface
APN	Access Point Name
BEEDeL	Bewertung des Einsatzes von Elektrobussen mit Dezentraler Ladeinfrastruktur
BMVI	Bundesministerium für Verkehr und digitale Infrastruktur
CSV	Comma-separated Values
GSM	Global System for Mobile Communication
HAW	Hochschule für Angewandte Wissenschaften Hamburg
HHA	Hamburger Hochbahn AG
HSPA	High Speed Packet Access
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
I2C	Inter-Integrated Circuit
IoT	Internet of Things
IP	Internet Protocol
IVI	Fraunhofer-Institut für Verkehrs- und Infrastruktursystem
JSON	Javascrip Object Notation
M2M	Machine-to-Machine
MVC	Model-View-Controller
NFS	Network Filesystem
OPKG	Open Package Management
PHP	PHP: Hypertext Preprocessor

PKI Public Key Infrastructure

RDBMS Relational Database Management System

RMS Root Mean Square

REST Representational State Transfer

SCP Secure Copy

SOAP Simple Object Access Protocol

SPI Serial Peripheral Interface

SQL Structured Query Language

SSH Secure Shell

TCP Transmission Control Protocol

TLS Transport Layer Security

UART Universal Asynchronus Receiver/Transmitter

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunications System

URI Uniform Resource Identifier

URL Uniform Resource Locator

VM Virtual Machine

VPN Virtual Private Network

WLAN Wireless Local Area Network

WWW World Wide Web

XML Extensible Markup Language

Literaturverzeichnis

[1] *OpenVPN: Security Overview*. – URL https://openvpn.net/index.php/open-source/documentation/security-overview.html. – Zugriffsdatum: 03.05.2016

[2] *Regierungserklärung des Ersten Bürgermeisters Olaf Scholz*, 2015. – URL http://www.hamburg.de/senatsthemen/4491844/regierungserklaerung-mai-2015/

[3] ABTS, Dietmar: *Masterkurs Client/Server-Programmierung mit Java*. Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – ISBN 978-3-658-09921-3

[4] BAYER, Thomas: *REST Web Services - Einführung u. Vergleich mit SOAP*. – URL http://www.oio.de/public/xml/rest-webservices.htm. – Zugriffsdatum: 08.12.2015

[5] BENGEL, Günther: *Grundkurs Verteilte Systeme*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014. – ISBN 978-3-8348-2150-8

[6] CODD, Edgar F.: A Relational Model of Data for Large Shared Data Banks. URL http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf. – Zugriffsdatum: 07.05.2016, 1970. – Forschungsbericht

[7] DAIGNEAU, Robert: *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services*. 1 edition. Addison-Wesley Professional, 2011. – ISBN 978-0-321-54420-9

[8] EDLICH, Stefan ; FRIEDLAND, Achim ; HAMPE, Jens ; BRAUER, Benjamin ; BRÜCKNER, Markus: *NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. 2. Carl Hanser Verlag GmbH & Co. KG, 2011. – ISBN 978-3-446-42753-2

[9] FIELDING, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, Dissertation, 2000. – URL http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. – Zugriffsdatum: 07.05.2016

[10] FRAUNHOFER IVI: *News 2015*. – URL http://www.ivi.fraunhofer.de/de/news_2015.html. – Zugriffsdatum: 07.04.2016

[11] FRAUNHOFER IVI: *PB Elektrobuseinführung 2015 Hamburg*. – URL http://www.ivi.fraunhofer.de/content/dam/ivi/de/documents/PB_Elektrobuseinfuehrung_2015_web_de.pdf. – Zugriffsdatum: 07.04.2016

[12] GLANZ, Axel ; JUNG, Oliver: *Machine-to-Machine-Kommunikation*. Frankfurt am Main u.a : Campus, 2010. – ISBN 978-359-33922-4-0

[13] HEDDERICH, Jürgen ; SACHS, Lothar: *Angewandte Statistik Methodensammlung mit R*. BerlinHeidelberg : Springer Spektrum, 2016. – ISBN 978-3-662-45690-3

[14] JAEKEL, Michael ; BRONNERT, Karsten: *Die digitale Evolution moderner Großstädte*. Wiesbaden : Springer Fachmedien Wiesbaden, 2013. – ISBN 978-3-658-00171-1

[15] LINUX FOUNDATION: *Yocto Project Mega-Manual*. – URL http://www.yoctoproject.org/docs/2.0/mega-manual/mega-manual.html. – Zugriffsdatum: 10.01.2016

[16] MELZER, Ingo: *Service-orientierte Architekturen mit Web Services: Konzepte - Standards - Praxis*. 4. Aufl. 2010. Spektrum Akademischer Verlag, 2010. – ISBN 978-3-8274-2549-2

[17] MÜLLER, Klaus-Rainer: *IT-Sicherheit mit System*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014. – ISBN 978-3-658-04333-9

[18] NOW GMBH: *Modellregion Hamburg: BEEDeL*. 2014. – URL https://www.now-gmbh.de/de/modellregionen-elektromobilitaet/projektfinder/modellregionen/hamburg/beedel. – Zugriffsdatum: 07.04.2016

[19] OECD: Machine-to-Machine Communications. URL http://www.oecd-ilibrary.org/science-and-technology/machine-to-machine-communications_5k9gsh2gp043-en. – Zugriffsdatum: 08.12.2015, 2012 (192). – OECD Digital Economy Papers

[20] ORACLE CORPORATION: *MySQL 5.5 Reference Manual*. : . – URL http://dev.mysql.com/doc/refman/5.5/en/. – Zugriffsdatum: 25.03.2016

[21] REITZ, Kenneth: *Requests: HTTP for Humans — Requests 2.9.1 documentation*. – URL http://docs.python-requests.org/en/v2.9.1/. – Zugriffsdatum: 02.02.2016

[22] ROLLAND, Fred: *Datenbanksysteme im Klartext*. Müchen u.a : Pearson Studium, 2003. – ISBN 3-8273-7066-3

[23] SCHÄDLICH, Gunter: *Moderne Batterietechnologien für den Einsatz in Elektrobussen des ÖPNV*. 2010. – URL http://www.trolley-project.eu/fileadmin/user_upload/Library/6S_Gunter_Schaedlich.pdf. – Zugriffsdatum: 07.05.2016

[24] SCHERFF, Jürgen: *Grundkurs Computernetzwerke: Eine kompakte Einführung in Netzwerk- und Internet-Technologien. Mit Online-Service*. 2., überarb. und erw. Aufl. 2010. Vieweg+Teubner Verlag, 2010. – ISBN 978-3-8348-0366-5

[25] SCHICKER, Edwin: *Datenbanken und SQL*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014 (Informatik & Praxis). – ISBN 978-3-8348-2185-0

[26] SCHILL, Alexander ; SPRINGER, Thomas: *Verteilte Systeme*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2012 (eXamen.press). – ISBN 978-3-642-25795-7

[27] SCHWENK, Jörg: *Sicherheit und Kryptographie im Internet*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014. – ISBN 978-3-658-06544-7

[28] SENDLER, Ulrich (Hrsg.): *Industrie 4.0*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013 (Xpert.press). – ISBN 978-3-642-36917-9

[29] STEINER, René: *Grundkurs Relationale Datenbanken*. Wiesbaden : Springer Fachmedien Wiesbaden, 2014. – ISBN 978-3-658-04287-5

[30] TILKOV, Stefan ; EIGENBRODT, Martin ; SCHREIER, Silvia ; WOLF, Oliver: *REST und HTTP*. dpunkt.verlag, 2015. – ISBN 3864901200

[31] TROLLEY MOTION: *Rückblick auf die 4. Internationale E-Bus Konferenz in Hamburg*. – URL `http://www.trolleymotion.eu/www/index.php?id=38&L=0&n_ID=2069`. – Zugriffsdatum: 07.05.2016

[32] WEGNER, Mario: *Entwicklung eines autonom arbeitenden GPS-Datenloggers mit hoher Updatefrequenz für die Anwendung in Nahverkehrsbussen*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorthesis, 2015

[33] WENDZEL, Steffen: *Tunnel und verdeckte Kanäle im Netz Grundlagen, Protokolle, Sicherheit und Methoden*. Wiesbaden : Springer Vieweg, 2012. – ISBN 978-3-8348-1640-5

[34] ZEPPENFELD, Klaus ; FINGER, Patrick: *SOA und WebServices*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009 (Informatik im Fokus). – ISBN 978-3-540-76990-3

[35] ZISLER, Harald: *Computer-Netzwerke*. Galileo Press GmbH, 2015. – ISBN 978-3-8362-3479-5

A. Anhang

A.1. Verwendung der Client-Applikation

Die Client-Applikation benötigt eine *Python 3* Laufzeitumgebung. Als zusätzliches Paket muss das *requests*-Modul installiert sein.

Während das Programm ausgeführt wird, wird automatisch eine Log-Datei erzeugt. Diese enthält Nachrichten über Ereignisse, Fehler oder Warnungen. Dabei wird ein *Rotating-Log* verwendet. Das bedeutet, sobald die Log-Datei eine gewisse Größer erreicht, wird die Datei archiviert und eine neue Log-Datei angelegt. Es werden bis zu fünf Log-Dateien archiviert.

Beim Aufruf des Programms können optionale Parameter angegeben werden. Diese sind in Tabelle A.1 zusammengefasst. Nachfolgend ist der Aufruf-Prototyp dargestellt:

```
:~# python3 dlmclient-main.py -c <configfile> -v <loglevel> -l <logfile>
```

Listing A.1: Aufruf-Prototyp der Client-Applikation

Parameter	**Funktion**
-c <configfile>	Pfad zur Konfigurationsdatei (Standard: /etc/dlmconfig.json)
-v <loglevel>	Level der Nachrichten, die in die Log-Datei geschrieben werden Mögliche Werte: 10, 20, 30, 40, 50
-l <logfile>	Pfad zur Log-Datei (Standard: /var/log/dlmclient.log)
--stats	Ermittelt den Systemstatus, zeigt diesen an und beendet das Programm
-h, --help	Optional: Zeigt eine Hilfe, sowie die möglichen Parameter an und beendet das Programm

Tabelle A.1.: Optionale Aufrufparameter für die Client-Applikation

Der folgende Beispielaufruf startet die Client-Applikation, verwendet jedoch eine alternative Konfigurationsdatei. Zusätzlich wird das *Log-Level* auf 10 gesetzt, wodurch alle Nachrichten mit der Einstufung „DEBUG“ und höher in die Log-Datei geschrieben werden.

```
:~# python3 dlmclient-main.py -c /testrun/config.json -v 10
```

Listing A.2: Beispielaufruf der Client-Applikation

A.2. Verwendung der Konvertierungssoftware

Die Software kann auf allen Betriebssystemen verwendet werden, auf denen eine Laufzeitumgebung für *Python 3* installiert ist.

Es werden verschiedene Parameter für den Aufruf des Konverters benötigt. Diese sind in der folgenden Tabelle A.2 aufgelistet. Die Parameter werden in der folgenden Reihenfolge übergeben:

```
:~# dsc <in_dir> <out_dir> <processed_dir> <out_format>
```

Listing A.3: Aufruf-Prototyp der Konvertierungssoftware

Parameter	**Funktion**
in_dir	Ordner, in dem sich die BEEDeL-Datensätze befinden
out_dir	Ordner, in welchen die konvertierten Datensätze gespeichert werden
processed_dir	Ordner, in den die bereits konvertierten BEEDeL-Datensätze verschoben werden
out_format	Format, in welches die Daten konvertiert werden sollen: csv, json, xml
--serial <ID>	Optional: Client-ID, die in jeden Datenpunkt eingetragen werden soll
-h, --help	Optional: Zeigt eine Hilfe, sowie die möglichen Parameter an und beendet das Programm

Tabelle A.2.: Aufrufparameter für die Konvertierungssoftware

Der folgende Beispielaufruf konvertiert alle Datensätze in das CSV-Format, die sich im Ordner „/data/datasets“ befinden, speichert die generierten CSV-Datensätze in „/data/csv“ und verschiebt die verarbeiteten BEEDeL-Datensätze in den Ordner „/data/datasets/done“.

```
:~# dsc /data/datasets /data/csv /data/datasets/done csv
```

Listing A.4: Beispielaufruf der Konvertierungssoftware

A.3. Screenshots der Benutzer-Oberfläche

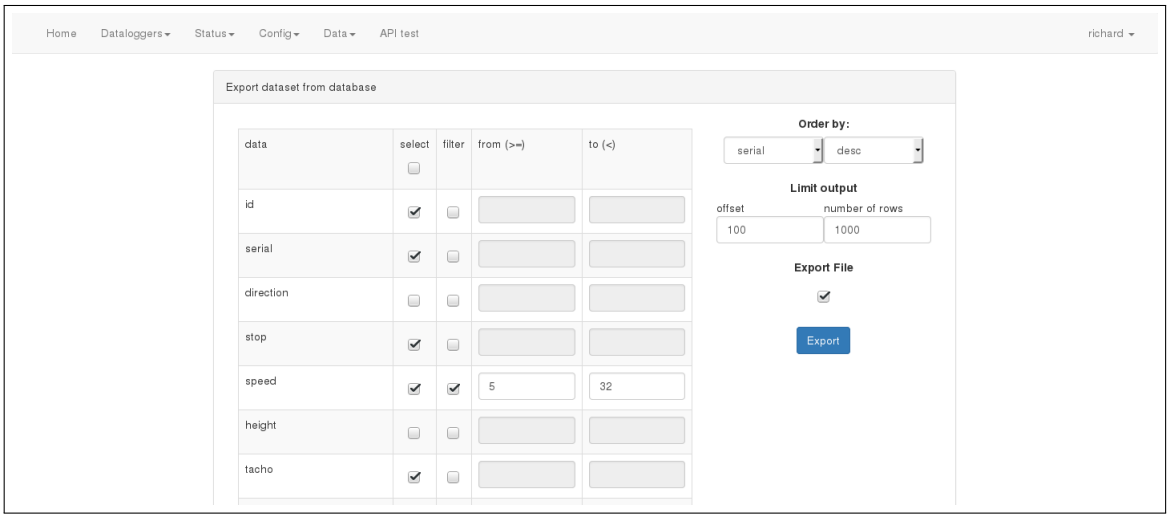

Abbildung A.1.: Benutzer-Oberfläche: Formular zur Messdatenanfrage

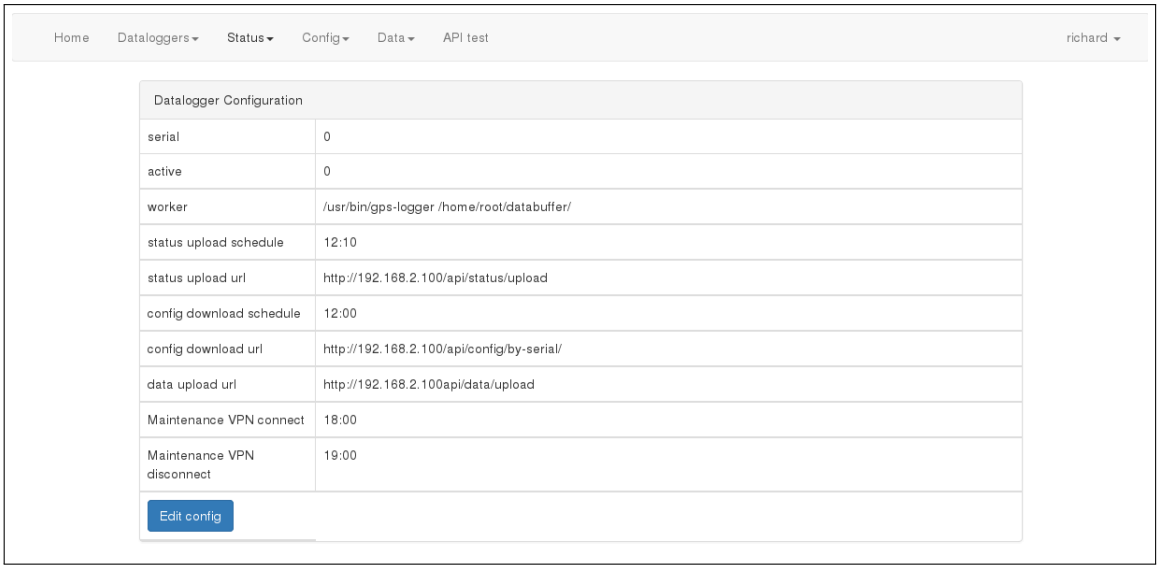

Abbildung A.2.: Benutzer-Oberfläche: Ansicht einer Client-Konfiguration

B. DVD

Die folgenden Anhänge der Arbeit befinden sich auf einer DVD, welche bei Prof. Dr. rer. nat. Rasmus Rettig (HAW-Hamburg) und Prof. Dr. -Ing. Detlef Jensen (FH-Westküste) eingesehen werden kann.

B.1. Masterthesis

- Masterthesis im PDF-Format

B.2. Quellcode der realisierten Software

1. Webapplikation des Servers: „dlms-server“
2. Client-Management-Applikation: „dlms-client“
3. Konvertierungssoftware für „BEEDeL“-Datensätze: „dlms-dataset-converter“
4. Yocto-Layer für die Erstellung des Betriebssystems: „yocto-meta-datalogger“
5. *MATLAB*-Funktion zum herunterladen eines Datensatzes über den Webservice: „dlms_dataset_download.m“
6. *MATLAB*-Skripte und Funktionen zur statistischen Analyse von Fahrzyklen: „driving-cycle-analysis“

B.3. Auswertungsergebnisse

- Ermittelte Modellparameter der Fahrzyklen (MATLAB-Datei)

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 9. Mai 2016	
Ort, Datum	Unterschrift