Eva Rott, BSc

Design und Implementierung von Methoden zur Positionsbestimmung im Bluetooth Low Energy Mesh

Masterarbeit
zur Erlangung des akademischen Grades
Diplom-Ingenieur
Masterstudium: Computer Science

eingereicht an der
Technische Universität Graz

Betreuer
Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger
Amtsdirektor Andreas Zobl (quadratic GmbH)

Institut für Technische Informatik

Graz, Mai 2020

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Abstract

In the area of indoor positioning in large indoor structures, finding the exact position of objects can become quite a challenge due to the lack of conventional positioning methods such as the Global Positioning System (GPS). There exist a number of approaches for achieving indoor localization. Depending on the indoor environment and its infrastructure, a well performing technique or a combination of such is required.

This work focuses primarily on methods that can be applied to indoor environments consisting of multiple, partially underground locations. The vehicles that are parked in these locations represent the objects whose position shall be computed. The parking area as well as the vehicles are equipped with small hardware devices that communicate over Bluetooth Low Energy (BLE) and BLE Mesh. The devices are part of an already existing localization framework with multiple groups of hardware and software components. Over the course of this work, their functionality needs to be extended in the following way. First, the framework is enhanced with the ability to identify the broader area in which the vehicle can be found. Second, the computation of the exact position of the vehicle is implemented. Approaches, such as detecting the vehicle's state of motion for computing the position only when it is not moving, are used for increasing the quality of the data. For computing the exact position from this data a number of positioning techniques are researched and their performance is evaluated.

Four different approaches are studied. First, some well known, as well as some more recently developed trilateration methods with pre-filtered Received Signal Strength Indicator (RSSI) measurements as inputs are implemented and tested. Second, the enhancement of the localization framework with the new Bluetooth standard Angle of Arrival (AoA) is designed. In a third step, the potential improvements based on the application of the supervised machine learning algorithm K-Nearest Neighbors (KNN) is examined. Fourth, the impact of using additional data sources, such as the accelerometer sensor, for increasing the position accuracy is examined. In the last sections of this work, the implemented methods are evaluated and a good solution for indoor positioning is proposed.

Kurzfassung

Im Bereich der Innenraumlokalisierung in großen, teilweise unterirdischen Hallen stellt die Berechnung der Position eines Objektes eine Herausforderung dar, da konventionelle Methoden wie die Verwendung des GPS hier nicht verfügbar sind. Da das Thema der Innenraumlokalisierung schon seit längerem erforscht wird, wurden bereits verschiedene Methoden dafür entwickelt. Abhängig vom Aufbau und der infrastrukturellen Ausstattung der Lokalisierungsumgebung ergeben sich verschiedene Ansprüche an die Positionierungsmethode.

Diese Arbeit beschäftigt sich mit Methoden, die in großen Parkhallen angewandt werden können. Die Zielobjekte, deren Position berechnet werden soll, sind Fahrzeuge verschiedener Typen mit unterschiedlichem Aufbau. Sowohl die Parkhallen als auch die Fahrzeuge wurden zu diesem Zweck mit zusätzlichen Hardwareelementen ausgestattet. Diese Geräte verwenden BLE und BLE Mesh zur Kommunikation. Sie sind Teil des Ortungsframeworks, dessen Basis bereits zu Beginn dieser Arbeit besteht. Im Zuge dieser Arbeit werden die Hard- und Softwarekomponenten des Frameworks in mehreren Schritten weiter ausgebaut. Zuerst wird das Framework durch die Fähigkeit der Identifikation der groben Standortbereiche der Fahrzeuge erweitert. Erst danach wird die Berechnung der genauen Position der Fahrzeuge implementiert. Zusätzlich werden weitere Verfahren eingebaut, welche die Datenqualität verbessern sollen. Im Detail gibt es die Möglichkeit zu erkennen, wann das Fahrzeug nicht mehr in Bewegung ist, denn nur dann sollen Daten gesammelt werden. Um aus diesen anschließend die Position berechnen zu können werden die folgenden vier Methoden genauer untersucht.

Zuerst werden verschiedene Trilaterationsalgorithmen, bewährte ebenso wie neuere, in Kombination mit der Vorfilterung der RSSI Messwerte zur Positionsberechnung umgesetzt. Zweitens wird die Erweiterung des Ortungsframeworks durch den neuen Bluetooth Standard AoA entworfen. In einem dritten Schritt wurde der Machine Learning Algorithmus KNN verwendet, um die Position auf Basis der generierten Fingerprints zu berechnen. Viertens wird die Verbesserung der Positionsgenauigkeit durch Hinzunehmen von weiteren Inputs wie dem Beschleunigungssensor untersucht.

Am Ende dieser Arbeit werden die Methoden zur Verbesserung der Ortung evaluiert. Zusätzlich wird eine gute Lösung zur Positionsberechnung von Fahrzeugen im Innenbereich vorgeschlagen.

Danksagung

Diese Diplomarbeit wurde am Institut für Technische Informatik an der Technischen Universität Graz durchgeführt.

Zuerst möchte ich mich bei Ass.Prof. Dipl.-Ing. Dr.techn. Christian Steger sowohl für die fachliche Unterstützung, als auch für die Geduld und im Besonderen für die faire Behandlung bedanken.

Ebenfalls möchte ich mich bei den Innsbrucker Verkehrsbetrieben und bei DI Christoph Friedrich für die gute Zusammenarbeit und die Möglichkeit, zu so einem interessanten Thema meine Masterarbeit verfassen zu dürfen, bedanken.
Amtsdirektor Andreas Zobl möchte ich für die umfassende fachliche und technische Unterstützung bei der Entwicklung des Projekts danken. Ihm und Martin Moser, beide Gründer der Firma quadratic, möchte ich weiters dafür danken, dass sie mir die erste Chance als Softwareentwicklerin gegeben haben. Dank gilt auch jedem Einzelnen im Team quadratic für die gute Zusammenarbeit und schnelle Hilfe bei Exceptions und I2C Bus Problemen.

Am meisten muss ich mich bei meiner Familie bedanken, ohne die ich gar nicht bis zur Masterarbeit gekommen wäre. Hätte meine Schwester mir vor vielen Jahren nicht das Vertrauen mitgegeben, dass ich ein technisches Studium anfangen kann, so hätte ich die Begeisterung für die Softwareentwicklung nie entdeckt. Ohne sie, die mir Zusammenhänge erklärt hat, obwohl ich sie manchmal lieber nicht verstanden hätte und deswegen nicht die beste Gesellschaft war, hätte ich einige schwere Prüfungen nicht geschafft. Dafür möchte ich Danke sagen und ihr nur halb so viele faire Chancen wünschen wie ich das Glück hatte bekommen zu dürfen.
Bei meinen Eltern möchte ich mich für die persönliche Unterstützung rund ums Studium bedanken inklusive offenem Ohr, finanzieller Unterstützung und der Freiheit, mein Studium so zu gestalten wie es für mich am besten gepasst hat mit vielen Nachtschichten und verschobenen Prüfungen.
Zuletzt möchte ich mich noch bei einem der tollsten „Features“ meines Studiums, meinem Freund, bedanken. Begonnen bei so mancher Übung, die ich ohne seine Hilfe wohl noch mal gemacht hätte, über die seelische Unterstützung beim Tafelrechnen bis hin zu gemeinsamen Prüfungen, bei denen er mir geholfen hat meine Nerven zu bewahren, muss ich Danke sagen.

Graz, im Mai 2020 Eva Rott

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

2D 2-Dimensional. 14

3D 3-Dimensional. 14, 64

AJAX Asynchronous JavaScript and XML. 58

AoA Angle of Arrival. 2, 3, 5, 7, 12, 13, 15, 16, 23, 45, 46, 63, 66, 73, 74

AP Access Point. 26–28, 47

API Application Programming Interface. 14, 15, 29, 31, 37, 41, 44, 48–51, 58–61

ATT Attribute. 34

BGM Blue Gecko Module. 34, 36, 37, 45, 53, 58, 61–63

BHM Betriebshofmanagement. 28, 31, 37, 41, 44, 49, 50, 58–60

BLE Bluetooth Low Energy. 2, 3, 7, 11, 13–17, 24, 27, 29, 31, 33, 34, 36–39, 41, 42, 44, 46, 48, 49, 56, 58–61, 74

CLI Command Line Interface. 45, 63

CRC Cyclic Redundancy Check. 34

CTE Constant Tone Extension. 7, 23–25, 45, 46, 63

dBm Dezibel Milliwatt. 19

DFU Device Firmware Upgrade. 34, 37, 62, 63

ECA Error Correction Algorithm. 7, 9, 23, 24, 67–71, 73

FPZ Zuständen von Ortungsereignissen mit explizitem oder implizitem Fingerprintlevel. 42, 43

GATT Generic Attribute. 35, 36, 61, 63

GPS Global Positioning System. 2, 3, 14–16

GUI Graphical User Interface. 41, 60

HTTP Hypertext Transfer Protocol. 55, 56, 58–60

ID Identifier. 37, 41, 61

IMU Inertial Measurement Unit. 15

IP Internetprotokoll. 37, 58

IQ In Phase and Quadrature. 7, 24–26

IVB Innsbrucker Verkehrsbetriebe und Stubaitalbahn GmbH. 10, 28, 29, 49, 50, 58

IVI Initialization Vector Index. 62

KNN K-Nearest Neighbors. 2, 3, 5–7, 25–27, 41, 46, 48, 51, 52, 60, 63, 64, 73, 74

LAN Local Area Network. 37, 58

LoRa Long Range. 13–15

MSE Mean Squared Error. 9, 67, 69

MSSQL Microsoft SQL Server. 58

MTU Maximum Transmission Unit. 34

MVB Movementbyte. 7, 51, 52, 64, 65, 74

NPM Node Package Manager. 58

OTA Over The Air. 7, 34, 36, 37, 62, 63

PHP Hypertext Preprocessor. 56, 58

REST Representational State Transfer. 14, 15, 58

RFID Radio-Frequency Identification. 13–15

RSSI Received Signal Strength Indicator. 2, 3, 8, 12, 15, 16, 18, 19, 21, 23, 28, 34, 36, 37, 40–44, 47–51, 59, 66, 68–73

RTLS Real Time Localization Systems. 5, 9, 13–17, 73

SAP Systeme, Anwendungen und Produkte in der Datenverarbeitung. 14, 15

SDK Software Development Kit. 53

SDMC Secure Digital Memory Card. 61

SIG Special Interest Group. 23

SOAP Simple Object Access Protocol. 14, 15

SSH Secure Shell. 58

TCP Transmission Control Protocol. 55, 56, 58

TDoA Time Difference of Arrival. 15

TLZ Zuständen von Ortungsereignissen mit explizitem oder implizitem Trilaterationslevel. 42, 43

ToF Time of Flight. 14

TX Transmit. 33, 34, 53

UART Universal Asynchronous Receiver Transmitter. 36, 55, 56, 58, 61

UWB Ultra-Wideband. 13–17

WCL Weighted Centroid Localization. 8, 9, 22, 67–69, 71, 73

Wi-Fi Wireless Fidelity. 13–15

WLAN Wireless Local Area Network. 11, 28, 37, 58

Kapitel 1

Einleitung

Die Lokalisierung von Fahrzeugen innerhalb von Gebäuden spielt gerade bei Mobilitätsbetrieben eine große Rolle. Hierbei ergeben sich verschiedene Anforderungen und Herausforderungen. Die Anforderungen bestehen darin, dass die Fahrzeuge am Beginn eines Arbeitstages aufgrund der zuvor berechneten Position vom Fahrer schnell gefunden werden können. Die Herausforderungen dabei sind die Art der Umgebung in denen die Fahrzeuge stehen, nämlich die verschiedenen Hallen, die vielen Oberflächen, die viel Potenzial für Signalreflexionen bieten, sowie die verschiedenen Arten von Fahrzeugen. Grob lassen sich diese in Busse und Straßenbahnen einteilen, wobei alle Fahrzeuge generell sehr dicht beieinander abgestellt werden.

Es gibt zu diesem Thema und dieser Anwendung bereits bei den Innsbrucker Verkehrsbetriebe und Stubaitalbahn GmbH (IVB) ein Vorgängerprojekt, das auf Basis von einer anderen Drahtlostechnologie durchgeführt wurde. Auf die Unterschiede zum Vorgängerprojekt wird im nächsten Kapitel genauer eingegangen.

Für das neue Projekt wurde ein anderer Ansatz gewählt. Die Fahrzeuge so wie die Hallen wurden mit neuer Hardware auf Basis der BLE Technologie ausgestattet. Zusätzlich dazu wurden die Webanwendungen für die Datenverarbeitung und die Visualisierung neu aufgebaut. In den folgenden Abschnitten wird auf die Motivation hinter dem Projekt sowie die Voraussetzungen und Herausforderungen genauer eingegangen.

1.1 Motivation

Das Ziel der Arbeit ist die automatische Standortbestimmung von Fahrzeugen. Sobald ein Fahrzeug in eine Halle einfährt, sollte dieses erkannt werden. Danach wird das Fahrzeug verschiedene Kontrollpunkte passieren. Dies wird ebenfalls vom System registriert werden. Sobald das Fahrzeug am Parkplatz angekommen ist, sendet es noch mehrere Minuten lang ein Signal aus. Zuerst wird auf Basis von den passierten Kontrollpunkten der Hallenabschnitt, in dem sich das Fahrzeug befindet bestimmt. Danach wird die genaue Position des Fahrzeugs mithilfe der letzten Signale, die vom Fahrzeug in dem Hallenabschnitt gemessen wurden berechnet.

Verwendet wird der Standort für eine Live-Übersicht über alle Fahrzeuge und wo sich diese gerade befinden. Das kann zum Beispiel das Liniennetz sein, falls ein Fahrzeug gerade im Diensteinsatz ist, oder auch der genaue Stellplatz im Betriebsgelände. Diese Informationen werden zum Beispiel zum Finden des Fahrzeugs zu Dienstbeginn verwendet. Weiters lässt sich erkennen, wenn ein Fahrzeug in der Werkstatt ist oder gerade in der Waschstraße gereinigt wird.

Die dafür benötigten Daten werden über Bluetooth Low Energy (BLE) gesendet bzw. empfangen. Anschließend werden sie an den Server weitergeleitet, der diese Daten weiter verarbeitet und Positionsereignisse auslöst. Daraufhin folgt eine genaue Standortberechnung.

Dieses System lässt sich aufgrund der BLE Technologie und der flexiblen Hardware Bauart in Zukunft um zusätzliche (Sensor-)Daten erweitern. Zum Beispiel könnte ein Sensor zur Füllstandsüberwachung ausgelesen werden und die daraus gewonnene Information über das Ortungsframework an Monitoringsysteme weitergeleitet werden.

1.2 Voraussetzungen und Herausforderungen

Die Umgebung, in der das System läuft, besteht aus mehreren großen, mehrere Meter hohen Hallen. Verschiedene Fahrzeugtypen sollen geortet werden. Sowohl Busse als auch Straßenbahnen parken in den Hallen. Der Großteil der Hardware Module wird in den Fahrzeugen angebracht. Die verbleibenden Module werden hoch an der Decke und an den Wänden der Hallen montiert. Dadurch soll eine möglichst direkte Linie für die BLE Signale geboten werden.

Sowohl die Wände und Säulen als auch die Oberfläche der Fahrzeuge, welche aus verschiedenen Materialien bestehen, bieten viel Fläche für Reflexionen. Eine Folge der Reflexionen ist eine falsche Standortberechnung. Je mehr Reflexionen umso größer die Ungenauigkeit. Dies bedeutet auch je mehr Fahrzeuge sich in einer Halle befinden umso ungenauer der Standort des einzelnen Fahrzeugs. Neben den Signalreflexionen sind auch Signalauslöschungen ein Problem für die Ortung. Diese kommen öfter vor je mehr drahtlose Kommunikation stattfindet. Der Verlust von Signalen kann ebenfalls die Genauigkeit des Ortungsergebnisses beeinflussen. Daher ist es auch wichtig, die Anzahl der gesendeten Datenpakete so gering wie möglich zu halten. Bluetoothsignale können jedoch nicht nur von anderen Bluteoothsignalen sondern auch von Wireless Local Area Network (WLAN) oder ähnlichen Drahtloskommunikationstechnologien im selben Frequenzbereich beeinflusst werden. Da das jedoch außerhalb vom Einflussbereich des Ortungsframeworks liegt, muss hier die Bluetoothkommunikation auf weitere Kanäle ausgeweitet werden.

Eine weitere Herausforderung ist die limitierte Sendezeit der Signale in einem großen Teil der Fahrzeuge. Nachdem das Fahrzeug abgestellt wurde, besteht die Stromversorgung der Hardwaremodule nur noch für 20 Minuten. Das bedeutet, dass in der Zeit genug Daten von diesem Fahrzeug gesammelt werden müssen. Im Allgemeinen ist diese Zeitspanne ausreichend. Zu Stoßzeiten jedoch gehen aufgrund der vielen Fahrzeuge Signale verloren. Zusätzlich dazu müssen verschiedene Routineaufgaben in dieser Zeitspanne ebenfalls aus-

geführt werden. Das sind zum Beispiel das Auslesen von Charakteristiken die Sensorwerte enthalten oder auch Firmwareupdates, die durchgeführt werden müssen. Dies ist notwendig um den aktuellen Stand des Systems pro Fahrzeug zu überprüfen und aktuell zu halten.

1.3 Gliederung

Diese Arbeit ist in mehrere Kapitel aufgeteilt. In Kapitel 1 wurde die Motivation dieser Arbeit sowie ihre Voraussetzungen und Herausforderungen genauer beschrieben.

Das folgende Kapitel 2 beschreibt zuerst bestehende Systeme und vergleicht diese anschließend miteinander und mit dem in dieser Arbeit vorgestellten System zur Positionsbestimmung der Fahrzeuge, dem Ortungsframework. Weiters werden vier Methoden zur Verbesserung der Ortungsergebnisse untersucht und bestehende Lösungsansätze erklärt. Diese Methoden sind Received Signal Strength Indicator (RSSI)-basierte Trilateration, Angle of Arrival, Machine Learning und zusätzliche Quellen. Sie werden in den folgenden Kapiteln Design und Implementierung in dieser Reihenfolge wieder vorkommen.

Kapitel 3 beschreibt zuerst die Anforderungen an das Ortungsframework sowie seinen Aufbau im Bezug auf die Hard- und Softwarearchitektur. Danach wird die Umsetzung und Einbindung der vier Verbesserungsmethoden im Ortungsframework erörtert.

Im darauf folgenden Kapitel 4 findet man eine Beschreibung des Zieleinsatzgebietes der Ortung sowie Beispiele für Pläne. Ebenfalls wird auf die technische Umsetzung im Bezug auf die vom Ortungsframework verwendeten Hard- und Softwaretechnologien genauer eingegangen, sowie der Ablauf und die Abhängigkeiten von Systemupdates erläutert. Zuletzt wird die softwareseitige Umsetzung der verbesserten Ortungsmethoden beschrieben.

Auf die Beschreibung der Umsetzung der in dieser Arbeit entwickelten Komponenten folgt die Präsentation der Ergebnisse im Kapitel 5. Nach einer ersten Beschreibung der Testumgebung werden die Trilaterationsfilter und -algorithmen untereinander verglichen. Anschließend erfolgt ein Vergleich mit den Ergebnissen des Machine-Learning Ansatzes.

Zuletzt gibt es im Kapitel 6 die Zusammenfassung der wichtigsten Punkte und Erkenntnisse dieser Arbeit und es werden zukünftiger Schritte zur Weiterentwicklung des Ortungsframeworks genannt.

Kapitel 2

Grundlagen

Die Ortung von Objekten in Gebäuden ist ein aktuelles Thema. Besonders im Bereich Industrie 4.0 und Logistik wurden Real Time Localization Systems (RTLS) entwickelt. Diese basieren auf verschiedenen Technologien und Ansätzen. Dieses Kapitel beschäftigt sich mit bestehenden RTLS und den verschiedenen Ansätzen, welche die Positionsbestimmung im BLE-basierten System verbessern sollen.

2.1 Existierende RTLS

Die Lokalisierung von Objekten im Allgemeinen ist ein aktuelles, aber kein neues Thema. Aus diesem Grund gibt es bereits bestehende Systeme, die sich mit der Positionsbestimmung von Objekten im Bereich der Industrie oder des Handels beschäftigen. Im Folgenden werden die wichtigsten bestehenden Lokalisierungssysteme vorgestellt und mit dem in dieser Arbeit weiterentwickelten Ortungsframework verglichen.

infsoft Smart Connected Locations

infsoft [inf] hat ein Echtzeitlokalisierungs- und Trackingsystem für Innenräume entwickelt. Neben dem Tracking wird auch die Navigation, Objektidentifikation und Systemanalyse mit Statusüberwachung durch Sensortags in Echtzeit unterstützt. Hierfür werden mehrere Gruppen von Hardwareelementen verwendet. Die erste Gruppe sind Locator Tags, welche auf Basis von Long Range (LoRa) oder BLE funktionieren. Die zweite Gruppe sind Sensor Tags, die zusätzlich noch Sensordaten sammeln. Beacons mit E-Ink Display bilden die dritte Gruppe, welche zusätzlich Daten zum Objekt am Display darstellen. Zuletzt gibt es noch die Locator Nodes, welche die Positionsdaten sammeln. Auf der Softwareseite wird ein Data Hub und die LocAware Plattform zur Verfügung gestellt, welche die Systemdaten sammeln. Über einen Map Editor kann der Benutzer die Pläne von neuen Standorten erstellen und die Positionen der fixen Komponenten eintragen und aktualisieren. Zuletzt gibt es noch das Beacon Management, welches die Verwaltung der Beacons im System ermöglicht. Zusätzlich zu BLE und Wireless Fidelity (Wi-Fi) werden auch noch Ultra-Wideband (UWB) für eine noch genauere Positionsbestimmung und Radio-Frequency Identification (RFID) für die Objektidentifikation unterstützt. Neben der Positionsbestimmung und Systemüberwachung werden noch weitere Features optional

geboten. Diese sind unter anderem eine 360° Antenne für die Verwendung von Angle of Arrival (AoA) mit externen Drittanbietersystemen, Datenschutz durch die Verwendung von Hashes zur Identifikation und eine Representational State Transfer (REST) bzw. Simple Object Access Protocol (SOAP) Application Programming Interface (API) zur Integration in andere Systeme. Die Genauigkeit dieser Lösung beträgt aufgrund der Verwendung von UWB weniger als 30 cm und findet in Echtzeit statt.

eliko KIO RTLS

Ein UWB-basiertes Indoor Positionierungssystem wurde von eliko [eli] entwickelt. Es dient zum Objekttracking und verwendet batteriebetriebene Hardwaretags mit verschiedenen Montagemöglichkeiten inklusive eines Beschleunigungssensors, sowie Anchors zum Empfang der Tagdaten in bis zu 70 m Entfernung. Die Anchors leiten die Daten an den Server weiter, welcher diese sammelt. Auf der Softwareseite dient der KIO RTLS Manager zum Einrichten des Systems und zur Visualisierung der Daten. Die Kommunikation zwischen Anchor und Server erfolgt über Ethernet oder Wi-Fi. Neben der Positionsbestimmung wirbt das System mit Tracking bzw. der Verbesserung der Verarbeitungstransparenz im Zielsystem und einer API zur Integration in andere Systeme. Die Distanzbestimmung erfolgt mittels Time of Flight (ToF) und wird anschließend in der Trilateration weiterverarbeitet. Die Wahl des Anchors und des Bereichs findet automatisch statt. Die Genauigkeit des Systems beträgt dank UWB 5-30 cm.

INTRANAV Tracking Industrial Assets in Motion

Das dritte hier vorgestellte System wurde von INTRANAV [int] entwickelt und dient ebenfalls dem Objekttracking in Echtzeit sowohl in 2-Dimensional (2D) als auch in 3-Dimensional (3D) und unterstützt die Ortung im Innen- und Außenbereich. Es bietet eine Middleware für die Einbindung von Technologien wie RFID, UWB, BLE, Global Positioning System (GPS), LoRa und Wi-Fi. Zusätzlich können Nutz- und Sensordaten gesammelt werden. Ein Dashboard bietet eine Übersicht über den Systemzustand bzw. die Erkenntnisse durch die Big-Data IntraLytics. Ebenfalls können Bereichszonen definiert werden und Trigger für diese angelegt werden. Ein Sicherheitsassistenzsystem kann helfen, Kollisionen von Fahrzeugen wie Staplern zu vermeiden. Neben vielen weiteren Features wird auch ein Flottenmanagement und dynamisches Auftragspooling unterstützt sowie die Integration in Systeme, Anwendungen und Produkte in der Datenverarbeitung (SAP) ermöglicht. Die Genauigkeit beträgt ±8.5 cm in 3D.

sewio Real Time Location System

Ein weiteres Echtzeit-RTLS wurde von sewio [sew] zur Anwendung im Bereich Industrie 4.0 entwickelt. Es verwendet Tags und Anchors mit UWB zur Kommunikation und bietet viele verschiedene Softwareanwendungen: Den RTLS Manager für Deployment und Kommunikation des RTLS, den RTLS Planner um den optimalen Systemaufbau zu identifizieren, den RTLS Monitor zur Ortungssystemüberwachung, den RTLS Player für die Datenanalyse und zur Optimierung der Zielanwendung, Sensmap um aktuelle Vorgänge im Zielsystem zu sehen, z. B. welche Produkte gerade wie verarbeitet werden, und SAGE Analytics welche Heatmaps und ähnliche Diagramme zur Visualisierung von Vorgängen

im Zielsystem darstellt. Neben dem Objekttracking wird auch die Navigation zum Objekt unterstützt. Zusätzlich dazu gibt es ein Flottenmanagement, Softwarezonen mit Eingangsdetektion, Geschwindigkeitskontrolle und Kollisionsvermeidung für mehr Sicherheit. Die Position selbst wird mittels Time Difference of Arrival (TDoA) bzw. Two Way Ranging berechnet. Aufgrund der Verwendung von UWB beträgt die Positionsabweichung maximal 10 cm.

leantegra RTLS

leantegra [lea] hat ein weiteres RTLS entwickelt welches für Objekt- oder Personentracking verwendet werden kann und mit verschiedenen Technologien wie BLE, UWB, Mobile, Wi-Fi, GPS oder Tetra funktioniert. Zur Positionsbestimmung werden sowohl Beacons und Anchors aus eigener Herstellung als auch Hardwareelemente von Drittanbietern unterstützt. Die Berechnung der Position erfolgt durch Multilateration mithilfe von verschiedenen Algorithmen und Filtern. Eine Inertial Measurement Unit (IMU) erlaubt Motion Tracking für batteriebetriebene Geräte wie die leantegra BLE Beacons. Zusätzlich dazu wird das Anlegen von Softwarezonen mit Triggern und deren Analyse unterstützt. Die Integration von AoA, welche eine Genauigkeit von 0.5 - 1 m liefert, ist noch in Arbeit. Leantegra bietet verschiedene Analyseanwendungen zur Optimierung des Zielsystems. Die Genauigkeit des RTLS beträgt 1 - 3 m bei der Verwendung von BLE und 10 - 30 cm im Fall von UWB.

blukii Location Based Services

blukii [blua] hat Beacons entwickelt welche nicht nur für die Lokalisierung von Objekten, sondern auch für Info- und Proximity Marketing Systeme verwendet werden können. Zusätzlich zu den Beacons wurden noch Wearables, welche die Beacons in ihrer Nähe erkennen, und ein Hub entwickelt. Es gibt drei verschiedene Beacons: Smart Beacons für die Bereichserkennung und Positionsbestimmung, Sensor Beacons zum Sammeln von Messdaten und Smart Keys um Zugriff auf Ressourcen zu verwalten. Der Hub sammelt die Daten der Beacons und visualisiert die Objektpositionen. Der blukii Manager dient zur Verwaltung und Konfiguration des Systems. Die von den Beacons verwendete Kommunikationstechnologie ist BLE mit der Unterstützung der Advertising Protokolle iBeacon und Eddystone. Es können drei verschiedene Sensorpakete gewählt werden: Environmental (Temperatur, Luftfeuchtigkeit, Luftdruck, Licht), Magnetometer (Magnetfeld), Beschleunigung und Neigung. Eine Distanzmessung kann über RSSI erfolgen, welcher über eine API zur Integration in ein externes System zur Verfügung gestellt wird. Da das System nicht selbst die Lokalisierung von Objekten durchführt, wird keine Genauigkeit für die Lokalisierung angegeben.

2.1.1 Vergleich der RTLS

Tabelle 2.1 zeigt einen Vergleich der zuvor beschriebenen Systeme. Wie in der Tabelle zu sehen, haben die existierenden Systeme durch die Verwendung von UWB eine sehr hohe Positionsgenauigkeit. Alle Systeme liefern das Ergebnis in Echtzeit. Die meisten von ihnen unterstützen noch weitere Kommunikationstechnologien wie BLE, Wi-Fi, RFID oder LoRa. Oft können auch weitere Sensoren an das Beacon angeschlossen werden. Diese

sind vor allem Bewegungssensoren und Umweltsensoren. Neben REST und SOAP APIs wird bei einem RTLS auch die Möglichkeit zur Integration in SAP geboten. Der Systemeinrichtungsaufwand ist eher schwieriger zu vergleichen, viele Lösungen stellen jedoch Softwareanwendungen zur Systemkonfiguration zur Verfügung. Die meisten Beacons sind batteriebetrieben mit einer Lebensdauer von ein bis zehn Jahren. Zusätzlich dazu enthalten fast alle Systeme weitere Analysewerkzeuge, welche auch die Optimierung des Zielsystems ermöglichen sollen. Weitere Features sind Flottenmanagement, Sicherheitsassistenzsysteme sowie eine Bereichsverwaltung mit Triggern.

Der Vergleich mit bestehenden Systemen zeigt sowohl die Konkurrenzfähigkeit von RTLS mit UWB, sowie den großen Featureumfang der bestehenden RTLS. Jedoch muss festgehalten werden, dass die RTLS zum Großteil für die Verwendung im Bereich Industrie 4.0 entwickelt wurden. Ihre Anforderungen, welche den Umfang und die Funktionalität betreffen, überschneiden sich also nur teilweise mit dem in dieser Arbeit weiterentwickelten Ortungsframework. Im Gegensatz zu einer benötigten Genauigkeit von wenigen Zentimetern für kleine Objekte ist bei der Ortung von Fahrzeugen eine Genauigkeit von bis zu drei Metern ausreichend. Zusätzlich dazu wird kein positionsgenaues Tracking der Fahrzeuge, sondern nur ein bereichsgenaues benötigt. Die Koordinaten des Fahrzeugs müssen erst bestimmt werden, wenn es nicht mehr in Bewegung ist. Das bedeutet auch, dass in der ersten Version des Ortungsframeworks keine Tools zur Analyse des Fahrverhaltens oder Ähnliches benötigt werden. Gleichzeitig ist das Ortungsframework jedoch offen für eine Erweiterung in diesem Bereich. Durch die Verwendung von BLE bietet es auch die Möglichkeit, die Fahrzeuge mit weiteren Sensoren auszustatten und damit den Zustand der Fahrzeuge gleichzeitig mit ihrem Standort überwachen zu können.

2.2 Methoden zur Verbesserung der Ortung

Durch die fehlende Möglichkeit der Verwendung von GPS müssen Algorithmen und andere Methoden zur Positionsbestimmung verwendet werden. Besonders im Bereich Bluetooth gibt es bereits viele Lösungsansätze dafür. Diese lassen sich in mehrere Gruppen einteilen. Die erste und größte Gruppe der Lösungen basiert auf Tri- bzw. Multilaterationsalgorithmen inklusive Datenfilterung zur Reduktion von Messfehlern wie z. B. Signalauslöschungen. Die zweite Gruppe, die genannt werden muss, basiert auf AoA Bestimmung. Diese Methode ist sehr neu und derzeit noch in Entwicklung. Sie erfordert zusätzliche Hardwarekomponenten. Eine weitere Möglichkeit zur Positionsbestimmung bildet Machine Learning unter Verwendung von Fingerprinting. Sie ist eine eher berechnungsaufwendige Methode, die genaue Informationen über das Umfeld, in dem die Positionsbestimmung stattfindet, benötigt. Zuletzt ist es noch möglich zusätzliche Hardwarekomponenten wie einen Beschleunigungssensor zu verwenden, um die Genauigkeit bisheriger Berechnungen weiter zu verbessern. Im Folgenden wird auf die Grundlagen aller hier genannten Methoden genauer eingegangen.

2.2.1 RSSI-basierte Trilateration

Trilateration ist eine Methode um die Position von einem Objekt auf Basis der Distanz zwischen dem Objekt und drei anderen Punkten, deren Position bekannt ist, zu berechnen.

	infsoft	eliko	INTRANAV	sewio	leantegra	blukii	Ortungsframework
Positionsgenauigkeit	++	++	++	++	+	keine	+/ - [ab]
Positionsergebnis Berechnungsdauer	++	++	++	++	++	keine	+ [c]
Ortungsbasistechnologie	UWB	UWB	UWB	UWB	BLE, UWB	keine	BLE
unterstützte Kommunikationsstandards	++	-	++	-	++	+	+
unterstützte Sensoren	+	+	+		+	++	++
APIs	++	++	++	++	++	++	++
Systemsetup	+	+	++	+	+	+	~ [d]
Energieverbrauch	+	+	unbekannt	+	++	++	++ [e]
Datenanalyse	+	+	++	++	+	-	+
Systemumfang	++	unbekannt	++	++	+		++

Tabelle 2.1: Vergleich von existierenden RTLS untereinander und mit dem Ortungsframework.

[a] abhängig von den verwendeten Methoden
[b] noch nicht alle Verbesserungen evaluiert
[c] ab 30 s, konfigurierbar
[d] benötigt derzeit noch Vorbereitungszeit, Setup noch in Entwicklung
[e] Deep Sleep Mode bei Beacons in Entwicklung

Bei RSSI-basierter Trilateration wird die Distanz aus der Signalstärke berechnet. Werden mehr als drei Distanzen verwendet nennt man diese Methode auch Multilateration.

RSSI-basierte Trilateration besteht aus mehreren Schritten. Zuerst müssen die RSSI Daten gefiltert werden, um den Effekt der Mehrwegsignalausbreitung zu reduzieren. Anschließend müssen die übrigen RSSI Werte in eine Distanz, z. B. Meter, umgerechnet werden, welche im letzten Schritt von Algorithmen zur Positionsberechnung verwendet werden.

Filterung der Daten

Um eventuelle Signalreflexionen aufgrund von Mehrwegsignalausbreitung herauszufiltern, gibt es verschiedene Filter, die vor der Weiterverarbeitung der Daten verwendet werden können. In den folgenden Abschnitten wird eine Auswahl an Filtern genauer beschrieben. Diese reichen von einfachen, wenig aufwendigen Filtern wie dem Median, Averaging und Moving Average Filter bis hin zum komplexeren Kalman Filter.

Der **Median Filter** ist der einfachste Filer. Hierbei werden die RSSI Werte sortiert und der Wert in der Mitte der Liste genommen.

Beim **Averaging Filter** bildet man den Mittelwert aller gemessenen RSSI Werte und erhält daraus einen einzigen RSSI Wert. Die Formel für die Berechnung des Filters sieht wie folgt aus. [NJNR17]

$$\overline{RSSI} = \frac{1}{m} \sum_{i=1}^{m} RSSI_i \tag{2.1}$$

Der **Moving Average Filter** funktioniert ähnlich wie der zuvor beschriebene Filter. Die Autoren von [SKS+16] wenden den Moving Average Filter immer auf die letzten zehn Messwerte an. Das bedeutet bei n Messwerten erhält man $n - 9$ gefilterte Messwerte.

Durch Verwendung des **Standard Deviation Filters** werden alle Messwerte, die mehr als n Mal die Standardabweichung σ vom Mittelwert abweichen, eliminiert. In [DVG+17] wird $n = 2$ genommen und somit werden alle RSSI Werte größer oder kleiner als zweimal σ aus dem Datensatz entfernt.

Der **Gauß-Filter** funktioniert gleich wie der Standard Deviation Filter. Der einzige Unterschied hierbei ist, dass $n = 1$ gesetzt wird. Der Filter wird sowohl in [DVG+17] als auch in [NJNR17] eingesetzt.

Beim **Outlier Filter** werden Werte die unter der doppelten Standardabweichung liegen entfernt. [SS18]

Einer der komplexeren Filter ist der **Kalman Filter**. Dieser Filter entfernt normalverteiltes Rauschen aus den Daten. Die Berechnung eines gefilterten Wertes aus der Menge aller Messwerte erfolgt iterativ und kann in zwei Phasen aufgeteilt werden. Die erste Phase geht zum nächsten geschätzten Zustand $\hat{x}_k$ und aktualisiert die Fehlervarianz $P_{\overline{k}}$.

$$\hat{x}_{\overline{k}} = \hat{x}_{k-1} \tag{2.2}$$
$$P_{\overline{k}} = P_{k-1} + Q \tag{2.3}$$

Hierbei gibt k den Zeitschritt an und Q ist die Prozessvarianz. Die zweite Phase aktualisiert das Kalman Gain K_k, berechnet den tatsächlichen Zustand $\hat{x}_k$ auf Basis vom geschätzten Zustand, dem Kalman Gain und dem letzten Messwert z_k, sowie den tatsächlichen Fehler P_k. Das Kalman Gain bestimmt, wie vertrauenswürdig die Werte der vorhergegangenen Iteration sind. [SKS+16, KAP06]

$$K_k = \frac{P_{\overline{k}}}{P_{\overline{k}} + R} \tag{2.4}$$
$$\hat{x}_k = \hat{x}_{\overline{k}} + K_k(z_k - \hat{x}_{\overline{k}}) \tag{2.5}$$
$$P_k = (1 - K_k)P_{\overline{k}} \tag{2.6}$$

RSSI Umrechnung zu Distanz

Die Umrechnung der RSSI Werte zu einer Distanz ist nicht linear und hängt von mehreren Faktoren ab. Diese Faktoren sind zum Beispiel die Signalstärke der Geräte oder der Rauschfaktor der Umgebung, auch Pathloss Exponent genannt. Für eine korrekte Umrechnung ist daher eine gute Initialisierung der Parameter wichtig. Dies lässt sich zum Beispiel durch Messungen in der Zielumgebung umsetzen, wobei Groundtruth Daten vorhanden sein müssen. Zusätzlich zu einer guten Initialisierung ist die kontinuierliche Überprüfung des RSSI Wertes vom Sendegerät in 1 m Entfernung wichtig, um die Parameter aktuell zu halten, da die Signalstärke über die Einsatzdauer des Gerätes abnimmt. Dies ermöglicht es, dass die Positionsgenauigkeit vom Anfang der Messungen auch nach längerer Laufzeit noch gegeben ist.

In bisherigen Arbeiten findet man bereits Methoden zur Umrechnung von RSSI in Dezibel Milliwatt (dBm) zur Distanz in Meter. Die am häufigsten verwendete Formel ist in Gleichung 2.7 gegeben.

$$RSSI = A - 10 * \eta * log_{10}(d) \tag{2.7}$$

Um die Distanz d zu berechnen, lässt sich die zuvor genannte Formel umformen.

$$d = 10^{\frac{RSSI - A}{-10 * \eta}} \tag{2.8}$$

Der $RSSI$ ist der gemessene RSSI Wert in dBm auf die Distanz d in Meter, A ist der gemessene RSSI in 1 m Entfernung und η repräsentiert den Rauschfaktor der Umgebung. Die Bestimmung von A und η kann mithilfe von linearer Regression erfolgen. Hierbei werden mehrere Messungen in einer bestimmten Distanz, z. B. 1 m oder 3 m, genommen. Die gemessenen RSSI Werte und die Distanz werden dann in folgende Gleichung eingesetzt, die mit Regression gelöst wird. [NJNR17, SKS+16]

$$[RSSI]_n = \eta * [d]_n + A \tag{2.9}$$

Im Fall von Innenräumen liegt der Rauschfaktor η zwischen 2 und 6 [DVG+17].

Trilateration Berechnung

Die Grundidee der Trilateration besteht darin, dass Abstände von drei Punkten zum gesuchten Objekt bekannt sind. Diese werden als Kreise mit dem Punkt als Mittelpunkt und dem Abstand als Radius dargestellt. Für die Kreise in Abbildung 2.1 erhält man also folgende Gleichungen:

$$d_A^2 = (x - x_A)^2 + (y - y_A)^2 \tag{2.10}$$
$$d_B^2 = (x - x_B)^2 + (y - y_B)^2 \tag{2.11}$$
$$d_C^2 = (x - x_C)^2 + (y - y_C)^2 \tag{2.12}$$

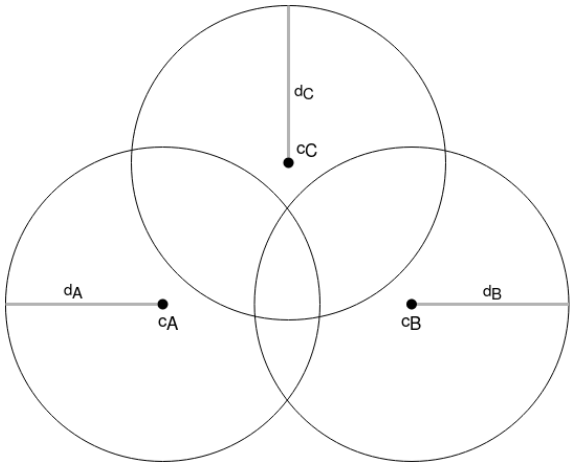


Abbildung 2.1: Die Trilaterationsberechnung basiert auf dem Schneiden von drei Kreisen mit Mittelpunkt c_X und Radius d_X.

Die gesuchte Position ist (x, y), also zwei Unbekannte in drei Gleichungen. Schneidet man nun die drei Kreise, erhält man im optimalen Fall genau einen Schnittpunkt. Aufgrund von Messungenauigkeiten in den Daten ist es möglich, dass man mehr als einen Schnittpunkt oder auch keinen erhält. Wenn nun die Information zu mehr als drei Punkt-Abstandspaaren vorhanden ist, verwendet man diese oft um die Genauigkeit des Ergebnisses weiter zu erhöhen.

Diese Arbeit beschäftigt sich mit fünf verschiedenen Trilaterationsalgorithmen, welche im folgenden genauer beschrieben werden.

Der wohl bekannteste Algorithmus ist **Least Squares Estimation**. Wie in [MJH99, NJNR17] beschrieben ist das Prinzip von Least Squares Estimation eine Lösung für n Unbekannte mithilfe von m Gleichungen zu finden und dabei den quadratischen Berechnungsfehler zu minimieren, wobei $n < m$. Es gibt also mehr Gleichungen als Unbekannte, da pro Messwert eine Gleichung erstellt wird. Bei Least Squares Estimation werden die Gleichungen als Zeilen einer Matrix notiert. Die Formel für Least Squares Estimation sieht daher wie folgt aus:

$$y = Ax + \epsilon \tag{2.13}$$

wobei y den Messwerten entspricht. A ist die Formel zur Berechnung der Outputwerte mit gesetzten Parametern und x ein Vektor mit den zu berechnenden Unbekannten. ϵ steht für den Fehler, der minimiert werden soll.

Im Fall der Kreisgleichungen von Gleichung 2.10 würde die Variablenbelegung wie folgt aussehen, wobei die Matrix A die Kreisgleichungen von x_B und x_C nach dem Schnitt mit x_A enthält.

$$y = \begin{bmatrix} (d_A^2 - x_A^2 - y_A^2) - (d_B^2 - x_B^2 - y_B^2) \\ (d_A^2 - x_A^2 - y_A^2) - (d_C^2 - x_C^2 - y_C^2) \\ (d_B^2 - x_B^2 - y_B^2) - (d_C^2 - x_C^2 - y_C^2) \end{bmatrix} \tag{2.14}$$

$$A = \begin{bmatrix} (x_A - x_B) + (y_A - y_B) \\ (x_A - x_C) + (y_A - y_C) \\ (x_B - x_C) + (y_B - y_C) \end{bmatrix} \tag{2.15}$$

$$x = \begin{bmatrix} x_{pos} \\ y_{pos} \end{bmatrix} \tag{2.16}$$

Durch Umformen der Least Squares Gleichung erhält man den Wert von x:

$$x = (A^T A)^{-1} A^T y \tag{2.17}$$

Wie am Ergebnis von x zu sehen, setzt dieser Algorithmus voraus, dass $A^T A$ invertierbar ist. Das bedeutet, dass er nicht für alle Daten eine Lösung liefert.

Eine Erweiterung des Least Squares Estimation Algorithmus ist **Weighted Least Squares Estimation**. Hierbei wird der Einfluss von einem Messwert auf das Gesamtergebnis anhand der Distanz d_i gewichtet. Je größer die Distanz, umso ungenauer die Messung desto weniger Einfluss soll sie auf das Ergebnis haben. Daher hat ein kleiner RSSI Wert mit großer Distanz weniger Einfluss auf das Gesamtergebnis als ein großer RSSI Wert, wo die gesuchte Position nahe bei der Messstelle liegt. Wie in [LQ13] beschrieben verändert sich also die Formel zur Berechnung von x:

$$x = (A^T W A)^{-1} A^T W y \tag{2.18}$$

wobei die Matrix W eine Diagonalmatrix ist die für das bisherige Beispiel von Abbildung 2.1 wie folgt aussieht:

$$W = \begin{bmatrix} \frac{1}{d_A^2} + \frac{1}{d_B^2} & 0 \\ 0 & \frac{1}{d_A^2} + \frac{1}{d_C^2} \end{bmatrix} \tag{2.19}$$

Es gibt verschiedene Ansätze für die Berechnung von W_{ii}. Diese Arbeit verwendet die Summe der invertierten Distanzen als Gewichtung, wie am Beispiel der Gleichung 2.19 zu sehen.

Ein weiterer Algorithmus ist **Trilateration Centroid**. Bei seiner unkomplizierten Berechnung wird die Position als Mittelwert der Koordinaten aller Schnittpunkte (sx_i, sy_i) gewählt [NJNR17].

$$(x, y) = \left(\frac{\sum_{i=1}^{n} sx_i}{n}, \frac{\sum_{i=1}^{n} sy_i}{n} \right) \tag{2.20}$$

Gibt es nun zwischen zwei Kreisen mehr als einen Schnittpunkt, so wird jener von beiden genommen, der dem Zentrum des dritten Kreises näher liegt. Zwei Beispiele hierfür sind in Abbildung 2.2 zu sehen.

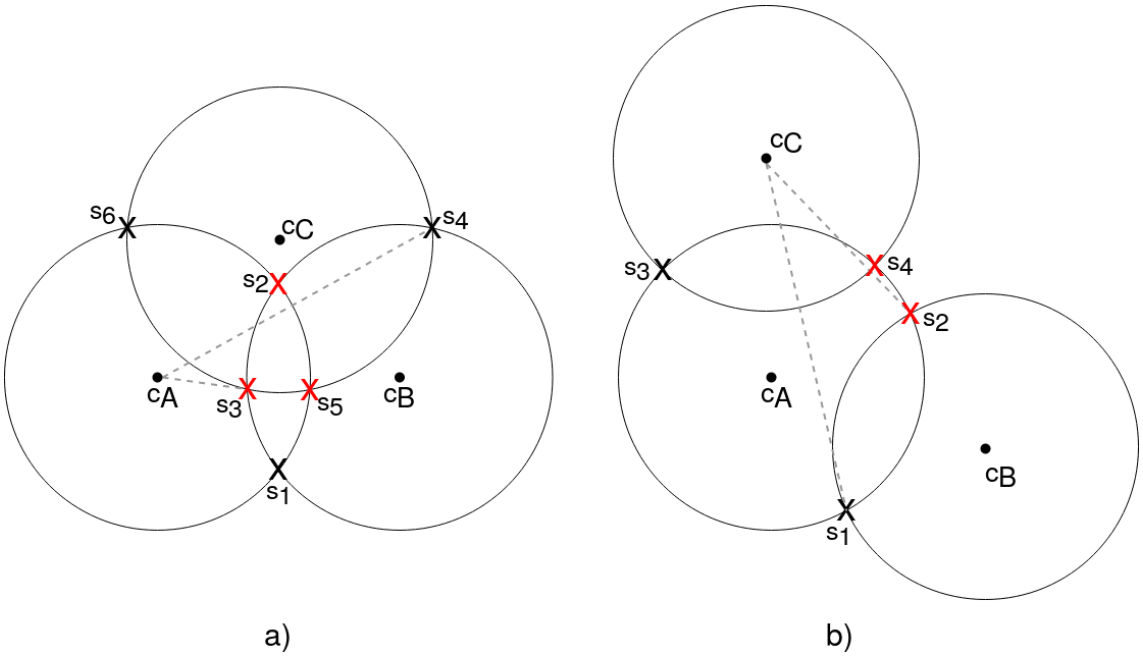


Abbildung 2.2: Der Trilateration Centroid Algorithmus wählt bei mehr als zwei Schnittpunkten zwischen zwei Kreisen jenen, der am nächsten beim Zentrum vom dritten Kreis liegt. So wird im Beispiel a) s_3 statt s_4 genommen und in Beispiel b) s_2 anstelle von s_1.

Der **Weighted Centroid Localization (WCL)** Algorithmus berechnet die Position, indem ein gewichteter Mittelwert der bekannten Positionen berechnet wird. Im Gegensatz zum ähnlich benannten Trilateration Centroid Algorithmus verwendet WCL die Position der Basisstationen anstelle der Schnittpunkte. Zuerst werden die Distanzgewichte bestimmt, wobei g ein konfigurierbarer Faktor ist.

$$w_i = \frac{1}{d_i^g} \tag{2.21}$$

Die finale Position ergibt sich also wie in [SKS+16] beschrieben:

$$x = \frac{\sum_{i=1}^{m} x_i w_i}{\sum_{i=1}^{m} w_i} \tag{2.22}$$

$$y = \frac{\sum_{i=1}^{m} y_i w_i}{\sum_{i=1}^{m} w_i} \tag{2.23}$$

Ein Algorithmus, der auf die verschiedenen Überschneidungskonstellationen von drei Kreisen eingeht, ist der **Error Correction Algorithm (ECA)**. Die Autoren von [MSA10] haben vier Überschneidungszustände identifiziert.

1. Jeder Kreis schneidet die anderen beiden in zwei Punkten. Es gibt also sechs Schnittpunkte. Der gesuchte Punkt ist der Mittelpunkt der drei Schnittpunkte, die am nächsten beieinanderliegen.

2. Zwei Kreise haben keinen Schnittpunkt zwischen einander. Es gibt also insgesamt vier Schnittpunkte. Der gesuchte Punkt liegt zwischen den zwei Schnittpunkten, die am nächsten beieinanderliegen.

3. Nur zwei Kreise schneiden sich. Das passiert, wenn entweder ein Kreis so klein ist, dass er die anderen nicht erreicht oder wenn er so groß ist, dass die anderen beiden innerhalb seines Radius liegen. Um diese Situation zu lösen, wird zuerst der Radius von dem Kreis, der keine Schnittpunkte hat, verändert. Bringt das keine Veränderung zu Zustand 1, so werden auch die Radien der anderen beiden Kreise verändert, und zwar auf umgekehrte Weise zur Veränderung des ersten Radius. Wurde dieser vergrößert so werden die beiden anderen verkleinert und umgekehrt. Bringt das keine Veränderung, so ist eine korrekte Positionsberechnung vom Algorithmus nicht gegeben.

4. Es gibt keine Schnittpunkte. In diesem Fall werden alle drei Radien in einem Zug verändert abhängig von der ursprünglichen Distanz zu den anderen beiden Kreisen.

Die vier Zustände und ihre Verarbeitung sind in Abbildung 2.3 aus der Arbeit von [MSA10] dargestellt.

2.2.2 Angle of Arrival

AoA ist eine Funkpeilmethode. Sie ist seit der Bluetooth Version 5.1 ein optionaler Teil der Special Interest Group (SIG) Bluetooth Kernspezifikation. Im Vergleich zu bisherigen Lokalisierungsmöglichkeiten mittels Bluetooth und RSSI-basierter Trilateration stellt AoA eine deutlich genauere Lokalisierungsmethode dar. Diese Form der Positionsbestimmung soll eine Genauigkeit von bis zu wenigen Zentimetern liefern [Blub].

Die Grundidee von AoA ist es einen Transmitter und einen Receiver zu haben, wobei der Transmitter das Gerät ist, welches man lokalisieren will. Um AoA nun durchführen zu können, müssen bestimmte Hardware und Software Voraussetzungen erfüllt sein. Zum einen benötigt der Receiver ein Antennenarray als zusätzliches Hardwareelement. Zum anderen muss der Transmitter eine spezielle Paketerweiterung, die Constant Tone Extension (CTE), aussenden während der Receiver sie empfangen können muss.

Im Folgenden werden Transmitter und Receiver sowie die Winkelberechnung und ihre weitere Anwendung genauer beschrieben.

Der **Transmitter** sendet eine spezielle Signalerweiterung, die CTE, aus welche vom Receiver empfangen wird. [Blu19a]

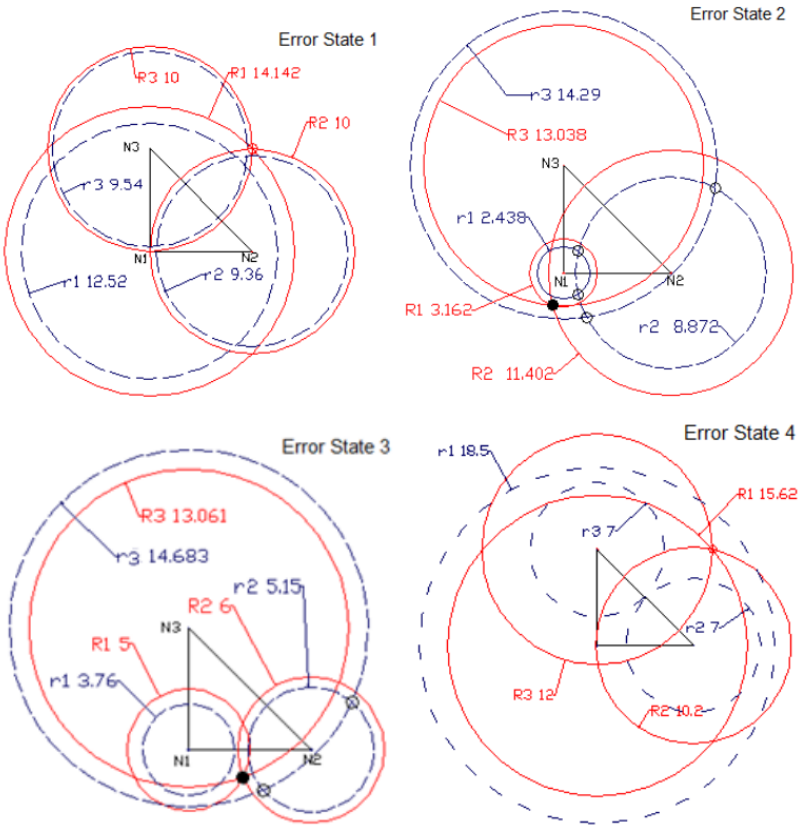


Abbildung 2.3: Die vier Zustände des ECA. Die blauen Linien geben die originalen Radien an während die roten Linien die vom Algorithmus adaptierten anzeigen. [MSA10]

Die **Constant Tone Extension** ist eine Reihe von Bits, die zusätzlich an Bluetooth Pakete angehängt wird. Sie hat eine variable Länge zwischen 16μs und 160μs und enthält eine konstant modulierte Folge aus Einsen ohne Whitening. Das ist ein Vorgang bei dem die Bitfolge der Nachrichten so verändert wird, dass lange Folgen von 0 und 1 vermieden werden. Whitening kann dazu führen, dass das Signal so verändert ist, dass der Receiver das Signal verliert. Um dies zu verhindern muss Whitening unterbunden werden. Zusätzlich dazu muss gewährleistet sein, dass sich die Frequenz des Signals auch nicht durch Modulation verändert. [Blu19b]

Im Allgemeinen besteht CTE aus einer Guard Period, einer Reference Period und einer Folge von abwechselnd Switch und Sample Slots, welche entweder 1μs oder 2μs lang sind, wie in Abbildung 2.4 zu sehen. [Blu19a]

Bei der CTE muss man zwei Arten unterscheiden: connection und connectionless. Erstere funktioniert mithilfe von Periodic Advertising, indem die CTE an die Advertising Pakete angehängt wird. Bei der zweiten Möglichkeit werden in einer aktiven BLE Verbindung zwischen zwei Geräten Pakete mit CTE gesendet. Das Mitsenden von CTE passiert nicht standardmäßig bei allen Paketen, sondern muss extra über Funktionsaufrufe in der Software aktiviert werden. [Blu19a, Blu19b]

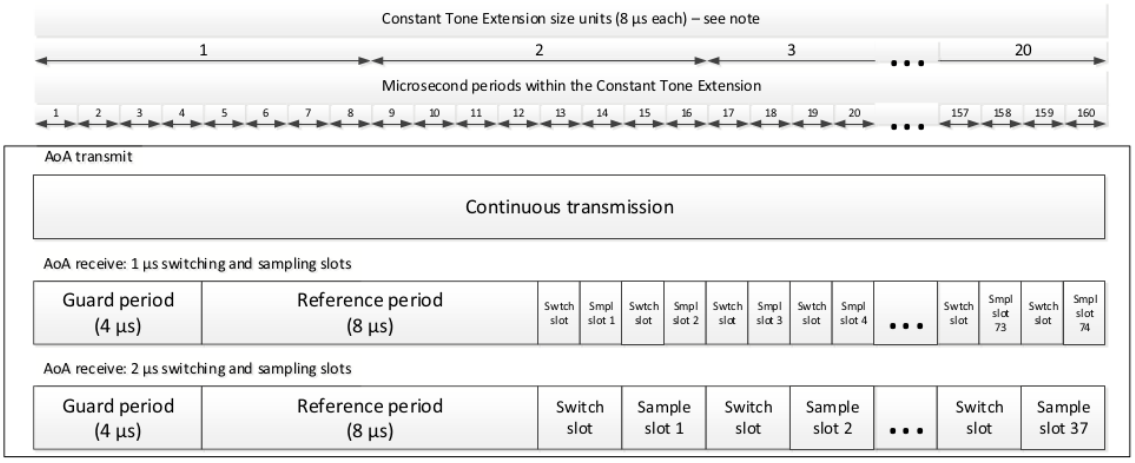


Abbildung 2.4: Die CTE besteht aus Guard Period, Reference Period und abwechselnd Switch und Sample Slots welche verschieden lang sein können. [Blu19a]

Um nun den Einfallswinkel bestimmen zu können, muss der **Receiver** mit einem Antennenarray ausgestattet sein. Auf diesem wird Antennenswitching durchgeführt. Hierbei werden In Phase and Quadrature (IQ) Samples an verschiedenen Antennen in verschiedenen Slots gesammelt. Das IQ Verfahren ermöglicht es, neben der Amplitude der Welle auch die Phaseninformation, also die Position innerhalb der Periode des Signals zum Empfangszeitpunkt, zu bestimmen. Auf Basis von IQ Samples kann dann der Winkel berechnet werden. [Blu19a]

Das **Antennenswitching** findet in periodischen Abständen, sogenannte Switch Slots, statt. CTE ist aufgeteilt in 4μs Guard Period und 8μs Reference Period. Während der Reference Period und in den Sample Slots sammelt die Link Layer IQ Samples, und zwar immer mit einer anderen Antenne. [Blu19a]

Ein **IQ Sample** soll jede Mikrosekunde während der Reference Period und in jedem Sample Slot genommen werden. Wie in Abbildung 2.5 zu sehen, besteht ein IQ Sample immer aus dem Winkel und der Amplitude vom Signal umgerechnet in kartesische Koordinaten. Diese zwei Werte sind verschieden, je nachdem zu welchem Zeitpunkt in einer Wellenlänge das Signal bei welcher Antenne ankommt. Ein Beispiel für drei Antennen und ihr Signalsampling ist in Abbildung 2.6 dargestellt. [Blu19a, Blu19b]

Der Winkel wird über die Phasenverschiebung, welche aus den IQ Samples berechnet wird, bestimmt. Die Formel für den Winkel ist in [Blu19a] wie folgt beschrieben.

$$\Phi = arccos\left(\frac{\psi\lambda}{2\pi d}\right) \tag{2.24}$$

ψ ist die Phasenverschiebung, λ ist die Wellenlänge und d ist die Distanz zwischen zwei Antennen.

Kennt man nun zumindest zwei Einfallswinkel, die von verschiedenen Punkten aus gemessen wurden, kann man die Position triangulieren.

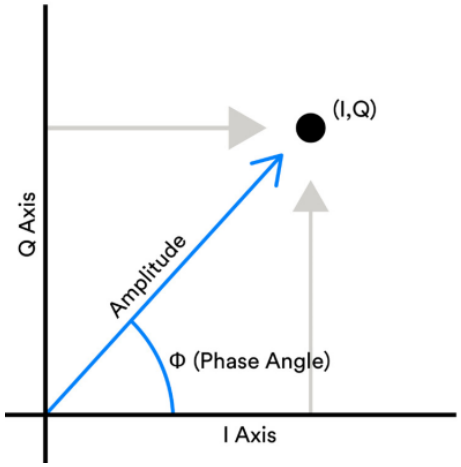


Abbildung 2.5: Ein IQ Sample besteht aus dem Winkel Φ und der Amplitude, mit der das Signal empfangen wurde. [Blu19b]

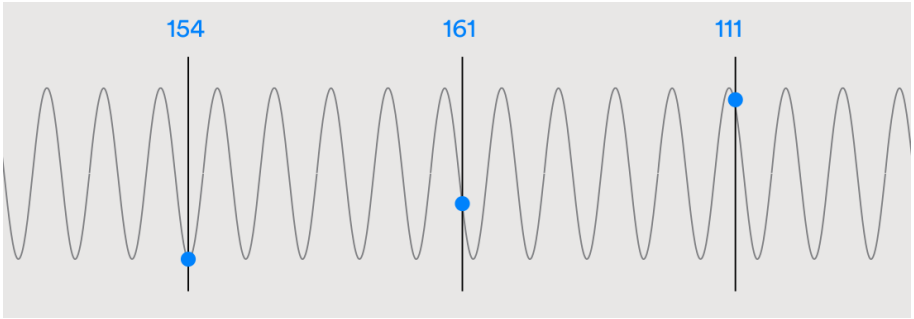


Abbildung 2.6: Die verschiedenen Antennen empfangen das Signal zu verschiedenen Zeitpunkten in der Wellenlänge. Dies ist mit blauen Punkten dargestellt. [Blu19b]

2.2.3 Machine Learning: K-Nearest Neighbors

Mit der in den letzten Jahren steigenden Präsenz von Machine Learning ist auch dessen Anwendung im Bereich der Positionsbestimmung immer wichtiger geworden. Wie häufig im Bereich Machine Learning verwendet, wird auch in dieser Arbeit Fingerprinting durchgeführt. Die dabei gewonnen Daten werden mithilfe des K-Nearest Neighbors (KNN) Algorithmus zu einer Position weiterverarbeitet. KNN ist ein bekanntes Verfahren, das bereits oft eingesetzt wurde. Wie auch in [LLH17] beschrieben besteht der Ansatz aus zwei Schritten: die offline Trainingsphase und die online Positionierungsphase. Der genaue Ablauf der zwei Phasen wird in den folgenden zwei Abschnitten beschrieben.

Trainingsphase

In der Trainingsphase werden die Fingerprints erstellt. In bisherigen Arbeiten [LLH17] und [CLZW18] wurde die Umgebung, in der die Positionsbestimmung stattfinden soll, zuerst in ein regelmäßiges Gitter aufgeteilt. Auf Basis von diesem wurden in festen Abständen sogenannte Access Points (APs) aufgestellt. Die Position der APs wurde genau vermessen.

Anschließend wurden die Fingerprints erstellt. Grundsätzlich besteht ein Fingerprint aus je einer Signalstärke pro AP, die an einer bekannten Position (Groundtruth) gemessen wurde. Gleich wie die APs werden die Fingerprints in festen Abständen gesammelt. Dies soll eine optimale Abdeckung des Geländes liefern. Laut der Arbeit von [Wil15] sollten nicht alle gemessenen Werte in den Fingerprint einfließen. Eine Untergrenze von -80 dBm für Bluetooth hat sich als effizienter Grenzwert zur Verbesserung der Positionsergebnisse herausgestellt. Zusätzlich gilt, dass eine gewisse Menge an Daten vorhanden sein muss, um ein verlässliches Auswertungsergebnis zu erreichen. Im Fall von KNN wurde von den Autoren von [SS18] eine Mindestanzahl von 300 Datensätzen empfohlen.

KNN-Phase

Nach erfolgreichem Sammeln der Datensätze können diese nun für eine Online-Positionierung verwendet werden. Dies passiert, indem für ein Objekt mit unbekannter Position Daten gesammelt werden, aus denen dann ein Fingerprint erstellt wird. Dieser wird anschließend mithilfe von KNN zu einer Position umgerechnet.

Die Grundidee des KNN Algorithmus besteht darin, dass für einen Inputfingerprint die K ähnlichesten Fingerprints im Datensatz gefunden und aus diesen dann die gesuchten Größen berechnet werden. Grundsätzlich gibt es hierbei zwei Möglichkeiten: Klassifikation für diskrete sowie Regression für kontinuierliche Größen. Ein Beispiel für eine diskrete Größe wären Standplätze. Im Gegensatz dazu sind absolute Positionen in Form von (x, y) Koordinaten z. B. in Metern ein Beispiel für kontinuierliche Größen. Diese Arbeit beschäftigt sich mit Letzterem.

Der erste Schritt bei KNN ist die Suche von ähnlichen Fingerprints. Nachdem ein Fingerprint nicht mehr als ein mehrdimensionaler Vektor, auch Featurevektor genannt, ist, wird die Distanz vom Inputvektor zu allen Vektoren aus dem Trainingsdatensatz berechnet. Danach werden jene K Vektoren genommen, welche zum Inputvektor die kleinste Distanz haben. Die Distanz kann auf mehrere Arten berechnet werden, zum Beispiel als Euklidische oder Manhatten Distanz [sci]. Der letzte Schritt bei KNN ist die Berechnung der gesuchten Größen als Mittelwert der Groundtruth Werte der gewählten K Nachbarn. Der Mittelwert kann optional auch gewichtet werden, sodass nähere Nachbarn mehr Einfluss auf das Ergebnis haben als weiter entfernte.

Herausforderungen

Dieser Ansatz zu Verbesserung der Ortung bringt einige Herausforderungen mit sich. Zum einen gilt, um Fingerprints vergleichen zu können, müssen diese gleich aufgebaut sein. Kommt also ein AP dazu oder weg so müssen die Fingerprints aktualisiert werden. Ein weiterer Grund für die Aktualisierung der Fingerprints ist der Signalstärkeverlust bei BLE Beacons im Laufe der Zeit. Im Fall von BLE, bei dem die Reichweite der Signale eher begrenzt ist, muss auch noch der Fall beachtet werden, bei dem nicht von allen APs Signale empfangen werden können und sich dadurch leere Felder in den Fingerprints ergeben. Zuletzt ist auch noch ein effizientes Speichern und Durchsuchen der in der Trainingsphase erstellten Fingerprints wichtig, um die Performance der Auswertung in einem akzeptablen Bereich zu halten.

2.2.4 Zusätzliche Quellen

Bisherige Arbeiten besonders im Bereich der Personenortung mittels Smartphone haben sich auf die Verwendung von Sensoren zur Verbesserung der Ortung konzentriert. Die Arbeit [LK16] verwendet Beschleunigungssensor, Magnetometer und Gyroskop, um die Position der Person sowie den Trend ihrer Bewegung zu berechnen. Die dabei gewonnen Daten werden verwendet, um die zuvor mittels Multilateration berechnete Position zu validieren und zu verbessern.

2.3 Vorgängerprojekt

Die aktuelle Arbeit ist Teil eines Nachfolgeprojekts von [Fri15]. Das Ziel des Projekts war ebenfalls die Indoor-Ortung von Fahrzeugen in den Hallen der IVB.

Die Grundidee war es die Ortung so aufzubauen, dass bestehende Hardware und Infrastruktur vollständig zur Positionsbestimmung verwendet werden konnten und dadurch Ressourcen gespart wurden. Die Fahrzeuge waren bereits mit einem WLAN Modul ausgestattet. Es wurden also WLAN Signale von bestehenden APs als Input für die Positionsberechnung genommen. Die Positionsberechnung basierte auf einer Kombination aus verschiedenen Ortungsmethoden, um möglichst gute Ergebnisse zu erzielen. Die Methoden waren unter anderem Cell of Origin, RSSI-basierte Trilateration und Fingerprinting.

Diese Lösung hat gute Ergebnisse geliefert, jedoch haben sich aufgrund der Verwendung von WLAN bestimmte Probleme ergeben, wie zum Beispiel die nicht kontinuierliche Verfügbarkeit von WLAN Signalen von allen APs in Fahrzeugnähe.

Gemeinsamkeiten des Projektes von [Fri15] zum Aktuellen sind die gleiche Ortungsumgebung und eine Schnittstelle zum Betriebshofmanagement (BHM) System der IVB sowie die Anwendung von Trilateration und Fingerprinting. Das aktuelle Projekt versucht, durch die Verwendung von Bluetooth und weiteren Ortungsmethoden die Positionsgenauigkeit weiter zu verbessern und Probleme, die sich im Vorgängerprojekt ergeben haben, zu lösen. Zum Beispiel die Datenverfügbarkeit, die nun durch die neue Bluetooth-Infrastruktur, die nur für die Ortung verwendet wird, höher ist.

Kapitel 3

Design

Das Ortungsframework wurde entwickelt, um automatisch die Standorte von Fahrzeugen zu berechnen. Seine Aufgaben lassen sich, wie in Abbildung 3.1 zu sehen, in drei Bereiche einteilen. Die primäre Aufgabe des Frameworks ist die Datenverarbeitung. Zusätzlich dazu sind die Wartung und Verwaltung des Systems sowie die Darstellung der berechneten Ergebnisse noch essenzielle Aufgaben des Ortungsframeworks.

Dieses Kapitel beschäftigt sich mit den Systemanforderungen und dem Systemaufbau aus Hard- und Softwaresicht. Weiters werden die wichtigsten Abläufe und Prinzipien genauer beschrieben. Zuletzt wird in diesem Kapitel auf die Methoden zur Verbesserung der Positionsgenauigkeit eingegangen.

3.1 Anforderungen

Die zentrale Anforderung an das Ortungsframework ist die Positionsbestimmung von unbewegten Fahrzeugen in Innenräumen mit einer Genauigkeit von weniger als drei Metern. Die Ortung von verschiedenen Fahrzeugtypen mit begrenzter (Busse) oder dauerhafter (Straßenbahnen) Stromversorgung soll möglich sein. Sie muss nicht in Echtzeit funktionieren, sollte jedoch möglichst zeitnah passieren. Die Technologie, die zur Ortung verwendet wird, ist BLE. Die Installation des Systems in einem neuen Gebäude soll mittels einer Webanwendung schnell und einfach funktionieren und Anzahl und Position der Hardwareelemente sollen frei konfigurierbar sein. Es wird nur die Positionsbestimmung jedoch nicht ihre Visualisierung benötigt, da Letzteres bereits vom bestehenden Fahrzeuginformationssystem der IVB umgesetzt wird. Die Ortungsergebnisse sowie eine Auswahl an Systeminformationen sollen dem Fahrzeuginformationssystem über eine API zur Verfügung stehen. Wichtig ist ebenfalls, dass Ortungsdaten effizient gesammelt werden. Dies bedeutet, dass Daten von Geräten, welche nicht Teil des Ortungsframeworks sind, wie zum Beispiel Smartphones, nicht gesammelt werden. Der Grund hierfür ist die verschlüsselte Kommunikation im BLE Mesh sowie die Auslöschung von BLE Paketen bei hohem Datenverkehr. Ein Monitoring über den Zustand der Komponenten des Ortungsframeworks mit Benachrichtigungssystem wird ebenfalls benötigt.

Da das Ortungsframework noch in Entwicklung ist, soll auch die Möglichkeit der Erweiterung durch zusätzliche Anwendungen existieren, zum Beispiel für die Analyse der

Fahrzeugbewegungen, sowie zur Durchführung von automatischen Firmware Updates. Neben der Erweiterbarkeit der Software soll es auch möglich sein das Fahrzeug mit weiteren Sensoren auszustatten, falls sein Zustand ebenfalls überwacht werden soll. Da keine Hardwarekomponente batteriebetrieben ist, ist die Minimierung des Energieverbrauchs keine primäre Anforderung für die erste Version des Ortungsframeworks.

Die genannten Anforderungen finden sich in den drei verschiedenen Aufgabenbereichen aus Abbildung 3.1 wieder, welche im Folgenden näher beschrieben werden. Der erste Aufgabenbereich umfasst die Verwaltung von Daten. Sie besteht aus der Generierung und dem Senden der Daten bzw. Signale in den Fahrzeugen, ihres Empfangs und ihrer Auswertung. Nachdem das System mehrere Komponenten umfasst, ist die Verwaltung der einzelnen Komponenten sowie die Wartung in Form von Updates für dauerhafte Komponenten bzw. das Hinzufügen und Entfernen von austauschbaren Komponenten wichtig. Die letzte Aufgabe ist die Darstellung der Ergebnisse.

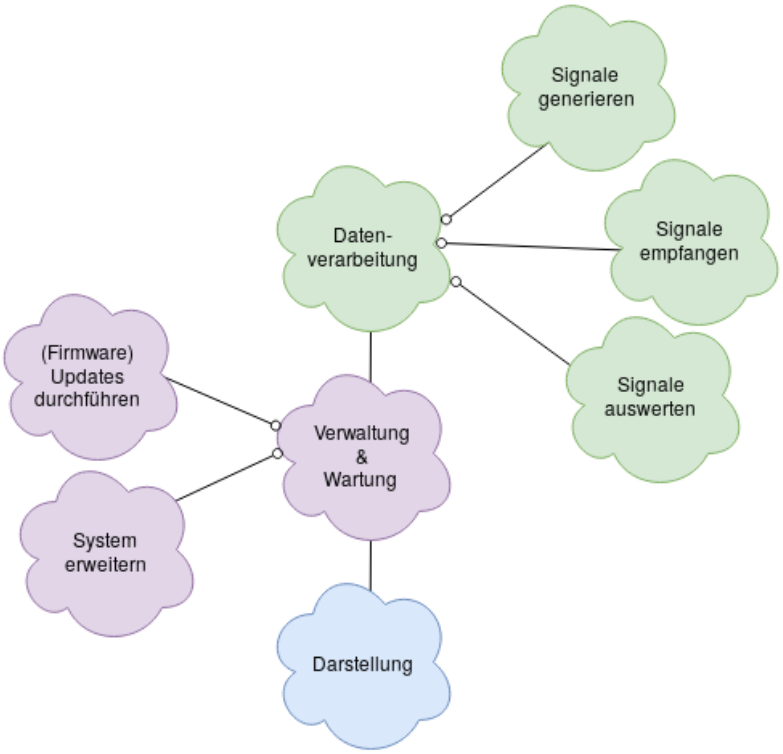


Abbildung 3.1: Das Ortungsframework muss drei Hauptaufgaben erfüllen: die Datenverarbeitung in Grün, die Systemverwaltung und -wartung in Violett und die Darstellung der Ergebnisse in Blau.

Abbildung 3.2 bietet eine Übersicht über die Komponenten des Ortungsframeworks. Die Basis bildet die Bluetoothkommunikation zwischen Beacon und Scannern. Das Beacon im Fahrzeug sendet kontinuierlich Pakete aus. Diese werden von den Scannern und dem Basescanner, welche eine Einheit in Form von einem Bluetooth Mesh bilden, mit

einer bestimmten Signalstärke empfangen. Der Basescanner dient als zentrale Kommunikationsschnittstelle zwischen den Teilnehmern im Mesh und der Basisstation. Er leitet die vom Mesh gesammelten Daten zusammen mit der Information, welches Element des Meshs diese Daten gesammelt hat, an die Basisstation weiter. Diese wendet ein Whitelisting an und leitet nur die Daten jener Beacons an den Concentrator weiter, die tatsächlich Teil des Ortungssystems sind. Der Concentrator ist die zentrale Sammelstelle der Daten aller Basisstationen und damit aller Scanner und Basescanner. Er führt eine bereichsgenaue Verarbeitung der Daten unter Verwendung von Systeminformationen, die er vom BLE Manager abruft, durch. Der BLE Manager ist die zentrale Systemverwaltungsanwendung, welche unter anderem Informationen darüber hat, welche Beacons, Scanner und Basisstationen es gibt und wo genau sich diese befinden. Durch diese erste Verarbeitung ist der Concentrator in der Lage ein grobes Ortungsergebnis in Form von Bereichen an die BHM API weiterzuleiten. Für eine standortgenaue Ortung werden die gesammelten Daten dem Trilateration Manager übergeben. Das Ergebnis wird ebenfalls an die BHM API weitergeleitet, welche für die Visualisierung der Standorte der Fahrzeuge zuständig ist.

Neben den genannten Komponenten, welche direkt für die Ortung wichtig sind, gibt es jedoch auch noch weitere Teile des Ortungsframeworks, die für Systemaufbau und -pflege zuständig sind. Im Folgenden werden die wichtigsten Hard- bzw. Softwarekomponenten genauer beschrieben.

3.2 Ausgewählte Hardwarekomponenten

Die wichtigsten Hardwarekomponenten sind Beacon, Scanner und Basescanner sowie der Provisioner. Sie kommunizieren mithilfe des BLE bzw. BLE Mesh Protokolls miteinander.

Beacon

Das Beacon ist ein Hardwareelement, welches im Fahrzeug montiert wird. Seine Hauptaufgabe ist es periodisch Advertisements auszusenden, welche dann von den Scannern mit einer bestimmten Signalstärke empfangen werden. Zusätzlich dazu gibt es noch die Möglichkeit sich direkt mit dem Beacon zu verbinden und verschiedene Charakteristiken des Bluetooth Profils, welches in Abbildung 3.3 dargestellt ist, auszulesen. Im Folgenden werden die Charakteristiken genauer beschrieben.

Die Charakteristik **Device Name** enthält den Namen des Bluetooth-Gerätes. Sie ist eine Standardcharakteristik aus der Bluetooth-Spezifikation.

Um bestimmte Charakteristiken vor dem Zugriff und der Manipulation durch Dritte zu schützen, wurde die **Passwort** Charakteristik erstellt. Der Ablauf, wie auf eine passwortgeschützte Charakteristik zugegriffen wird, besteht aus mehreren Schritten. Zuerst wird eine Bluetoothverbindung zum Beacon aufgebaut. Dann folgt eine Pairingaufforderung durch das Beacon, welche angenommen werden muss. Da das Beacon keine Input oder Output Möglichkeiten hat, wird die Just-Works Pairing Methode verwendet. Neben dieser gibt es beim Secure Simple Pairing noch die Passkey Entry Methode. Der erste Schritt beim Pairing ist der Austausch von Short Term Keys, um im nächsten Schritt die Long

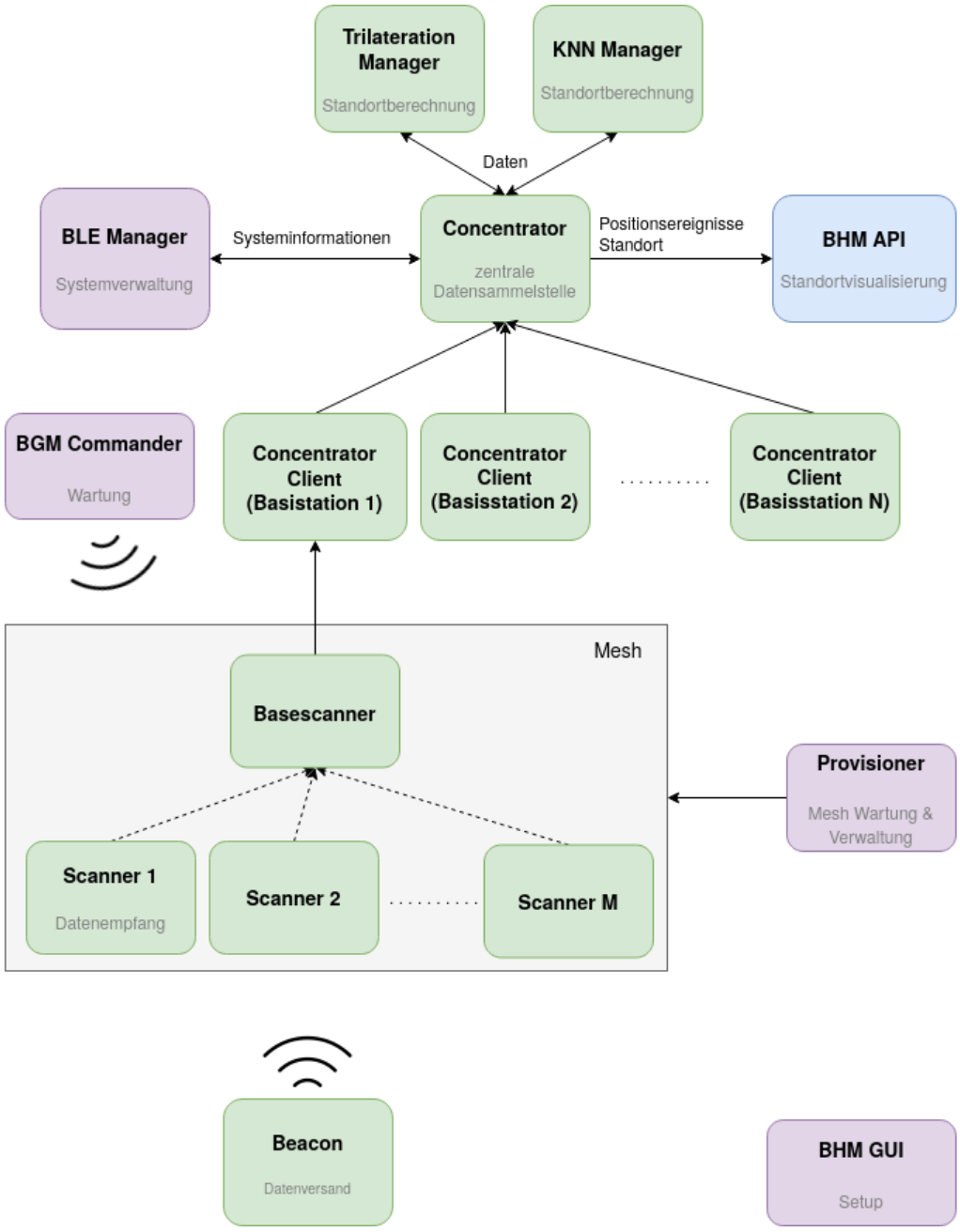


Abbildung 3.2: Das Ortungsframework besteht aus mehreren Komponenten. Diese sind ihren Aufgaben entsprechend eingeteilt: grün für die Datenverarbeitung, violett für die Systemverwaltung und Wartung und blau für die Darstellung der Ergebnisse.

Term Keys verhandeln zu können. Die Short Term Keys bestehen aus einer 128-Bit langen Zufallszahl, welche mit dem Temporary Key kombiniert wird. Im Gegensatz zur Passkey Entry Methode ist bei der Just Works Methode der Temporary Key 0. Ein Beispiel für einen Temorary Key bei der Passkey Entry Methode ist der sechsstellige Pin, der auf einem Gerät angezeigt wird und beim anderen eingegeben werden muss.

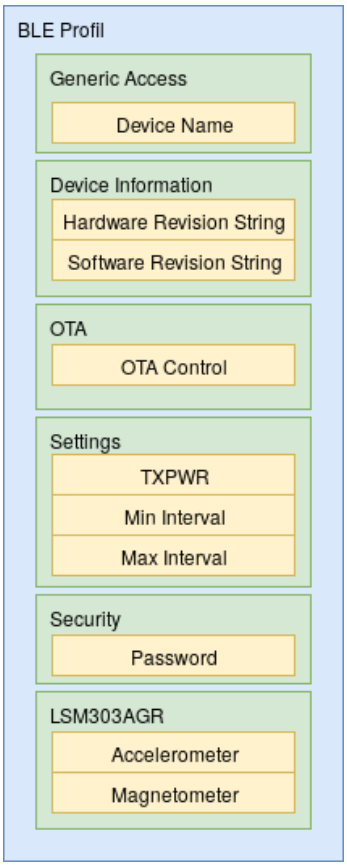


Abbildung 3.3: Das Beacon Bluetooth Profil umfasst mehrere Services mit Informationen zum Gerätezustand, Firmware Update, Advertisement Einstellungen und Sensoren.

Nach erfolgreichem Pairing Aufbau besteht eine begrenzte Zeit lang die Möglichkeit das korrekte Passwort in die Passwortcharakteristik zu schreiben. Ist das Passwort falsch oder wird die Zeitspanne überschritten, so trennt das Beacon die Verbindung. Ist das Passwort korrekt, so kann man die passwortgeschützten Charakteristiken auslesen bzw. schreiben. Im Allgemeinen gilt, dass das Passwort für jedes Gerät und für jede Firmwareversion anders ist, da es sich aus einer Passwortbasis berechnet, welche in jeder Firmwareversion (Major und Minor) geändert wird. Die Information zum aktuellen Passwort jedes Beacons ist über den BLE Manager abrufbar.

Die zwei Charakteristiken **Hardware & Software Revision** enthalten die Information über die aktuelle Hard- bzw. Softwareversion des Beacons. Sie sind besonders wichtig, um zu überprüfen, ob die Firmware am Beacon noch aktuell ist.

Die **Transmit (TX) Power** Charakteristik kann verwendet werden, um die Sendestärke der Beacon Advertisements zu verändern. Sie ist passwortgeschützt, da eine zu geringe Sendestärke dazu führen kann, dass das Beacon nicht mehr geortet wird. Auf der anderen Seite kann ein zu hoher TX Power Wert dazu führen, dass die Advertisements des Beacons auch in anschließenden Hallen oder Stockwerken empfangen werden können. Zu

beachten ist hierbei, dass sobald der TX Power verändert wird, der RSSI in 1 m Abstand sich ebenfalls verändert. Dies hat Auswirkungen auf die RSSI-basierte Trilateration. Eine derartige Veränderung muss vom BLE Manager wahrgenommen und gespeichert werden.

Die Werte der ebenfalls passwortgeschützten **Advertising Min/Max Interval** Charakteristiken bestimmen, mit welcher Häufigkeit das Beacon Advertisements aussendet. Der Wert kann von 10 ms bis 3 s eingestellt werden. Je öfter das Beacon sendet umso mehr Daten werden von den (Base)Scannern empfangen. Jedoch ist auch der Datenverkehr sehr hoch. Dies kann wiederum zur Auslöschung der Advertisements anderer Beacons führen.

Der Wert der **Accelerometer** Charakteristik entspricht dem Wert des integrierten Beschleunigungssensors. Er wird bei der Montage des Beacons einmal eingelesen und danach regelmäßig überprüft und mit dem Initialwert verglichen. Ist die Abweichung zu groß, hat sich die Position des Beacons im Fahrzeug verändert. Eventuell ist das Beacon nicht mehr korrekt montiert und sollte überprüft werden.

Ähnlich wie die vorige Charakteristik wird bei der **Magnetometer** Charakteristik ein Sensorwert ausgeliefert. Mithilfe von diesem Wert kann die Orientierung des Fahrzeugs bestimmt werden.

Die **Over The Air (OTA) Control** Charakteristik wird verwendet, um eine neue Firmware via Bluetooth auf dem Beacon zu installieren.

Eine Übersicht über den Ablauf des OTA Device Firmware Upgrade (DFU) ist in Abbildung 3.4 gegeben. Zuerst wird das Beacon in den Bootloader Modus versetzt. Danach wird die Attribute (ATT) Maximum Transmission Unit (MTU) zwischen Smartphone bzw. Blue Gecko Module (BGM) Commander und Beacon ausgemacht. Je größer die MTU, umso mehr Daten können auf einmal gesendet werden umso kürzer die Dauer des Updates. Anschließend wird der Code für den Apploader (Bluetooth Stack) gesendet. Das Aktualisieren des Apploaders ist bei OTA DFU grundsätzlich optional, wird jedoch im Ortungsframework immer durchgeführt, um auch den Umstieg auf neue Bluetooth Stack Versionen zu ermöglichen. Nach einer Überprüfung der Checksums wird das Beacon neu gestartet. Nachdem es im Flash den alten Stack mit dem neuen überschrieben hat, ist es erneut für Bluetooth Verbindungen bereit. Im nächsten Schritt kann also der Application Code gesendet werden. Sobald die gesamte neue Application Firmware in den Flash des Beacons geschrieben wurde, wird die Verbindung getrennt und das Beacon startet sich neu.

War das Update erfolgreich, so führt das Beacon nun den neuen Application Code aus. Andernfalls wartet es im Bootloader auf eine erneute Übertragung vom Apploader oder von der Application. Gründe für ein Fehlschlagen des Updates können eine fehlerhafte Übertragung oder falsch signiert oder verschlüsselte Binaries sein. Ersteres wird vom Gerät durch Überprüfen der Cyclic Redundancy Check (CRC) Checksums erkannt. Letzteres ist besonders wichtig um zu garantieren, dass keine Software von Dritten auf das Beacon gespielt werden kann. Dafür wird die Firmware signiert. Zusätzlich dazu wird sie noch verschlüsselt, um zu verhindern, dass eine dritte Partei durch Mithören der Übertragung Zugriff auf die Firmware erhält, da der Bootloader nicht Pairing-fähig ist und somit die Übertragung selbst nicht verschlüsselt ist. Die Schlüssel für beide Operationen werden beim

ersten Einrichten eines Gerätes zusammen mit dem Bootloader auf das Gerät geflasht und sind auch nach Updates verfügbar. Auf die gleiche Weise bleiben die Werte von Generic Attribute (GATT) Profil Charakteristiken erhalten, weil diese ebenfalls im persistenten Speicher stehen, der beim Updatevorgang nicht verändert wird.

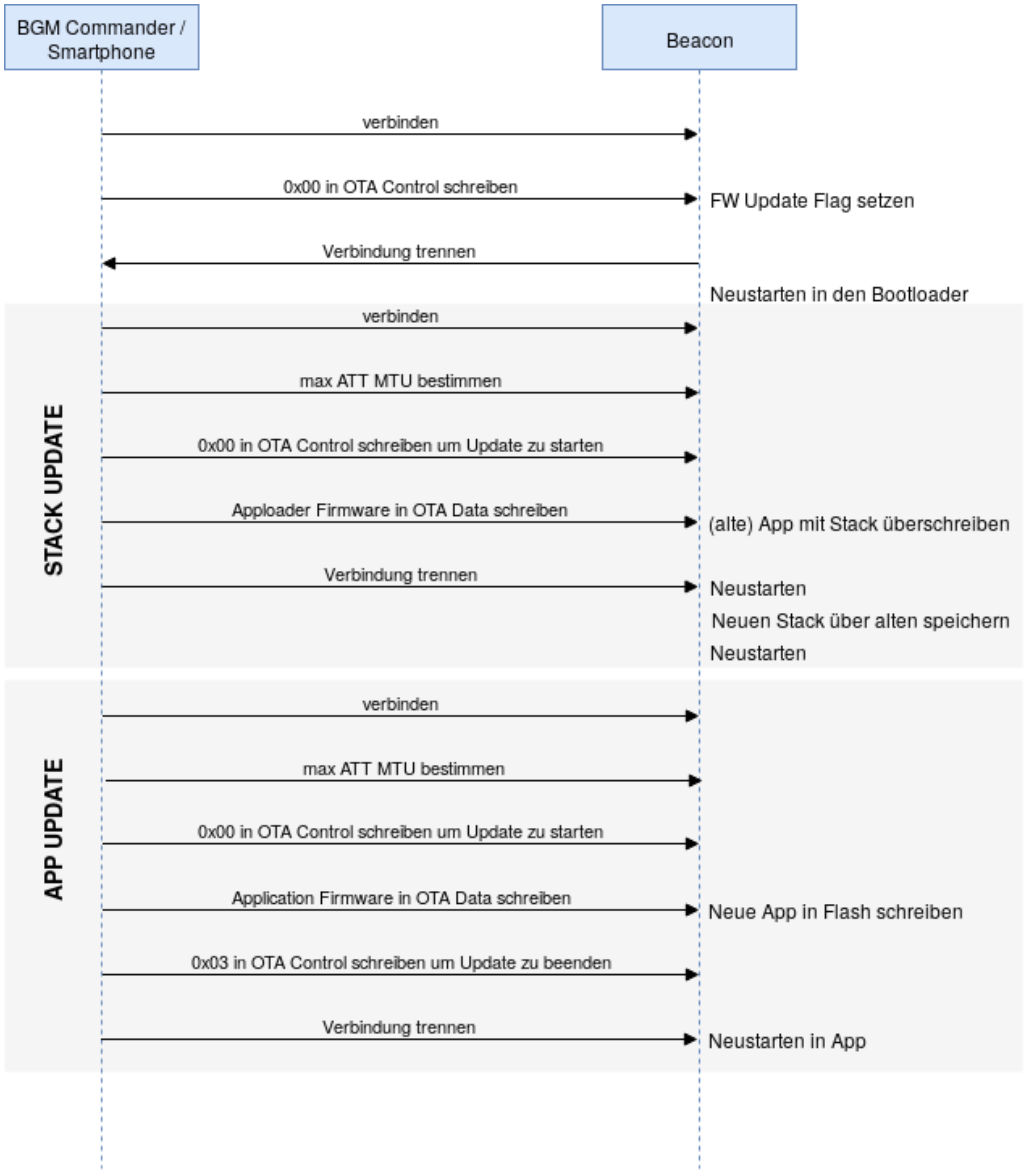


Abbildung 3.4: Die Firmware des Beacons kann drahtlos, Over The Air, in zwei Schritten aktualisiert werden. Zuerst wird der Stack aktualisiert, danach wird die neue Applikationsfirmware auf dem Gerät eingerichtet.

Scanner & Basescanner

Scanner und Basescanner sind sehr ähnlich zueinander. Ihre Hauptaufgabe besteht darin, die Advertisements der Beacons zu scannen und den RSSI Wert, mit dem die Pakete

empfangen wurden, weiterzuleiten. Mehrere Scanner und ein Basescanner bilden ein Bluetoothmesh. Wie in der Bluetooth-Protokollspezifikation vorgesehen, werden im Mesh verschiedene (Vendor-spezifische) Nachrichtentypen versendet. Diese werden vor allem zum Austauschen der Ortungsdaten (RSSI, Beacon Adresse und Scanner Adresse) verwendet.

Der Basescanner ist ein spezieller Typ von Scanner. Er ist das zentrale Element im BLE Mesh, welches die Daten aller Scanner sammelt und auf die gleiche Weise wie die eigenen an die Basisstation weiterleitet.

Das Bluetooth GATT Profil von Scanner und Basescanner besteht aus weniger Charakteristiken als jenes vom Beacon. Sowohl Scanner als auch Basescanner besitzen die Charakteristiken Device Name und Mesh Provisioning Service. Letzteres wird für die Interaktion mit dem Provisioner benötigt. Zusätzlich besitzt der Scanner noch die Hard- bzw. Software Revision und die OTA Control Charakteristik. Das Auslesen der Revisionen und die Aktualisierung der Firmware funktioniert also beim Scanner auf die gleiche Art wie beim Beacon. Im Gegensatz dazu werden diese drei Aufgaben beim Basescanner über die Universal Asynchronous Receiver Transmitter (UART)-Verbindung gelöst.

Mesh Kommunikation

Die aktuelle Kommunikation innerhalb eines Bluetoothmeshs besteht aus drei Operationen: `post_beacon_data`, `start_advertising` und `stop_advertising`. Die erste Operation ist für den RSSI Datenaustausch da während die zwei letzten benötigt werden, um einen Scanner in einen connectable Advertising Modus zu versetzten, damit sich andere Geräte mit ihm verbinden können. Im Zuge der Arbeit wurde die Meshkommunikation um weitere Nachrichtenmodelle und Operationen erweitert. Diese dienen unter anderem der Reduktion der Anzahl der gesendeten Meshnachrichten und des Setzens eines Adressenpräfixes für Beacons, damit schon bei den Scannern keine Daten von Beacons gesammelt werden, dessen Adresse mit einem Präfix anfangen, der dem Ortungsframework unbekannt ist.

Provisioner

Der Provisioner ist das Service, welches für das Erstellen und Konfigurieren von Bluetoothmeshes verantwortlich ist. Jedes Bluetoothmesh benötigt ihn damit Netzwerk- und Anwendungsschlüssel auf die Geräte gespielt werden. Das Setzen der Schlüssel ist besonders wichtig für eine sichere, private Kommunikation innerhalb eines Bluetoothmeshs. Zusätzlich konfiguriert der Provisioner die Mesh-Modelle mit ihren Publication und Subscription Adressen auf den Geräten.

3.3 Softwarearchitektur

Die wichtigsten Softwarekomponenten im Ortungsframework sind Concentrator und dazugehöriger Client, BGM Commander, BLE Manager und der Trilateration Manager. Diese und weitere benötigte Komponenten, sowie die wichtigsten Hintergrundstrukturen und Vorgänge, werden nun genauer beschrieben.

Concentrator Client

Der Concentrator Client ist die erste Komponente, welche nicht nur Daten sammelt, sondern diese auch schon teilweise filtert. Für jedes Datenpaket, welches vom Basescanner empfangen wird und aus RSSI, Beacon- und Scanneradresse besteht, wird überprüft, ob das Beacon Teil des Ortungsframeworks ist. Infolgedessen wird das Datenpaket entweder, im Fall von einem dem System bekannten Beacon, an den Concentrator weitergeleitet oder verworfen. Die Information, ob eine Beaconadresse Teil des Systems ist, kann vom BLE Manager abgefragt werden.

Basisstation

Der Concentrator Client ist eine Anwendung, welche auf einer Basisstation läuft. Pro Bluetooth Mesh gibt es eine Basisstation, welche sich durch eine Identifier (ID), einen eindeutigen Schlüssel und eine Local Area Network (LAN) bzw. WLAN Internetprotokoll (IP) Adresse von den anderen Basisstationen unterscheidet.

BGM Commander

Die Hauptaufgabe des BGM Commanders ist es Bluetooth Charakteristiken von Beacon und Scanner zu lesen bzw. zu schreiben. Die derzeitigen Fähigkeiten des BGM Commanders umfassen das Auslesen der Hard- und Softwarerevisionen und der Beschleunigungssensor Charakteristik sowie die Durchführung eines OTA DFU.

Concentrator

Der Concentrator ist die zentrale Stelle zum Sammeln der Ortungsdaten von allen Basisstationen. Er führt eine erste, bereichsgenaue Auswertung der Daten durch und löst sogenannte Ortungsereignisse aus, welche er an die BHM API weiterleitet. Die Ereignisse enthalten die Information in welchem Bereich sich ein Fahrzeug befindet, wie zum Beispiel Waschstraße oder Tiefgarage. Die Ortungsereignisse werden im BLE Manager konfiguriert und vom Concentrator regelmäßig aktualisiert.

Ein Beispiel dafür wie Ortungsereignisse funktionieren ist in Abbildung 3.5 gegeben. Die Grafik zeigt einen frei erfundenen Plan der Tiefgarage, welche die Ortungsumgebung darstellt. Sie besteht aus den Abschnitten Waschstraße und Tiefgarage OST, welche sechs Stellplätze enthält. Die Tiefgarage ist mit sieben Scannern S1 - S7 ausgerüstet. In dieser Ortungsumgebung gibt es neben dem Beacon-Defaultzustand „*Unbekannt*“, welcher allen Beacons zugewiesen wird die noch zustandslos sind, auch die Zustände „*Linie*“ und „*Betriebshof*“. In diesem Beispiel gilt, sobald ein Fahrzeug die Tiefgarage und damit den Betriebshof verlässt, befindet es sich im Zustand „*Linie*“. Weitere Zustände dieses Beispiels und ihre Hierarchie sind in Abbildung 3.6 ersichtlich. Die Zustände bezeichnen die verschiedenen Abschnitte, in denen sich das Fahrzeug in der Beispielumgebung befinden kann.

Nachdem nun die Zustände und Levels der Umgebung angelegt wurden, können auf die verschiedenen Scanner Trigger gelegt werden. Zum Beispiel kann auf Scanner S1 ein Trigger mit $minRssi = -60$ gelegt werden, welcher einfahrende Fahrzeug erkennt und

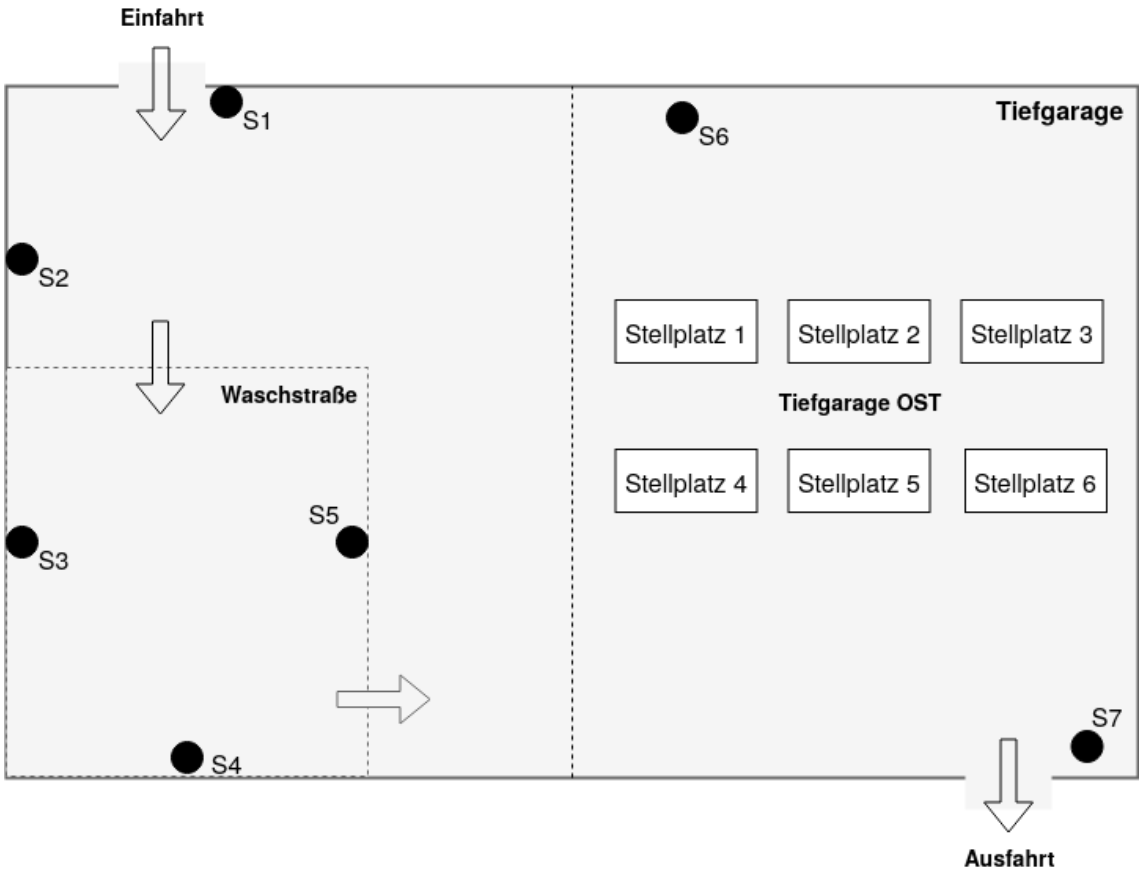


Abbildung 3.5: In einer frei erfundenen Beispielumgebung besteht die Tiefgarage aus den Abschnitten Waschstraße und Tiefgarage OST, welche wiederum Stellplatz 1 - 6 umfasst. Die Tiefgarage hat genau eine Einfahrt und eine Ausfahrt sowie sieben Scanner. Die Pfeile geben die Fahrtrichtung der Fahrzeuge an.

dadurch das Ortungsereignis „*Tiefgarage*“ auslösen kann. Mögliche Vorgängerzustände, die das einfahrende Beacon haben kann, sind „*Unbekannt*“, „*Betriebshof*“ oder „*Linie*“. Der Folgezustand des Beacons ist „*Tiefgarage*“.

Im Allgemeinen gilt, dass ein Ortungsereignis neben den im BLE Manager explizit gesetzten Vorgängerzuständen auch die hierarchisch höheren Zustände als Vorgängerzustände haben kann. Man nehme an, es gibt ein Fahrzeug mit einem Beacon im Zustand „*Betriebshof*“. Zusätzlich existiert ein Ortungsereignis, welches den Übergang von „*Tiefgarage*“ zu „*Waschstraße*“ beschreibt. Wenn nun das Fahrzeug in die Tiefgarage und anschließend in die Waschstraße einfährt und die Scanner S1 und S2 ausgefallen sind, dann könnte das Fahrzeug ohne diese Regelung nie in den Zustand „*Waschstraße*“ versetzt werden, da „*Betriebshof*“ kein explizit gesetzter Vorgängerzustand für Waschstraße ist. Mit dieser Regelung gelten nun auch all jene Zustände als Vorgängerzustände, die generelle sind als die explizit ausgewählten Vorgängerzustände. Nachdem der Betriebshof unter anderem auch die Tiefgarage umfasst, kann das Beacon nun vom Zustand „*Betriebshof*“

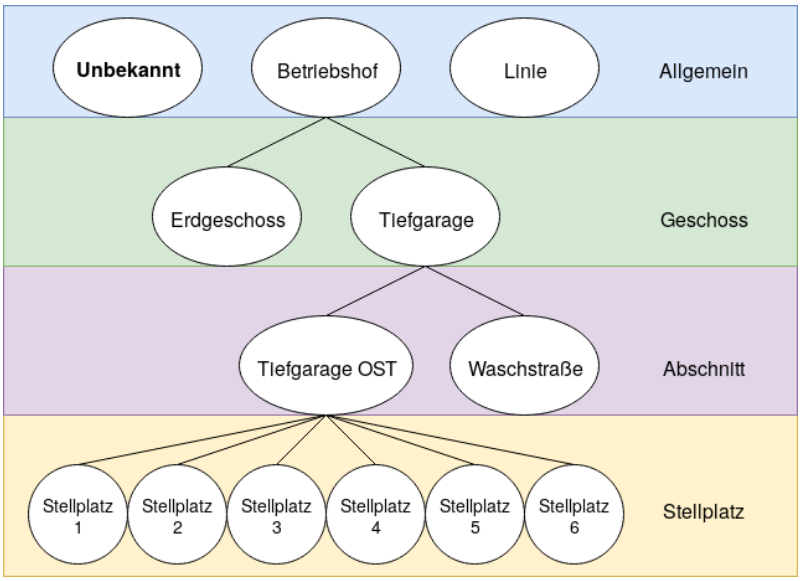


Abbildung 3.6: Die Zustände aus Abbildung 3.5 können vier Ortungslevels zugeteilt werden.

in den Zustand „*Waschstraße*“ wechseln. Zusammengefasst gesagt wurde diese Regelung gewählt, um einen Ausfall einzelner Scanner zu tolerieren.

Neben den Zuständen und Triggern im Ortungsereignis spielt auch das Ortungslevel eine wichtige Rolle. Es bestimmt, welches Ortungsereignis ausgelöst wird, wenn mehrere zur Auswahl stehen. Hängen nun an einem Scanner zum Beispiel zwei Trigger wie jene für die zwei Zustände „*Tiefgarage*“ und „*Tiefgarage OST*“ an S6 so wird das Ortungsereignis, das genauer ist, nämlich Tiefgarage OST ausgelöst.
Neben der bereichsgenauen Auswertung wird auch eine stellplatzgenaue Position benötigt. Die Berechnung der Position ergibt nicht in allen Zuständen Sinn. Manche Zustände umfassen einen zu großen Bereich und würden zu einer ineffizienten Positionsberechnung führen. Ein Beispiel dafür ist der Zustand „*Tiefgarage*“. Nimmt man nun jedoch die Zustände „*Waschstraße*“ oder „*Stellplätze 1-6*“ so muss man nur mit einer kleinen Menge an Scannern bei der Positionsberechnung arbeiten. Im BLE Manager kann man, durch das Setzen einer Flag, festlegen auf welchem Ortungslevel die Position berechnet werden soll. Alle Folgezustände von Ortungsereignissen mit diesem Ortungslevel gelten dann als Trilaterationszustand. Der genaue Zusammenhang zwischen den Klassen `Ortungsereignis`, `Ortungszustand` und `Ortungslevel` ist in Abbildung 3.7 zu sehen. Jedes Ortungsereignis hat neben den Standardfeldern `id` und `name` eine Liste von `vorgängerZuständen`, die das Beacon haben kann, damit es in den `folgeZustand` wechseln kann, sofern einer der zugewiesenen `trigger` ausgelöst wird.

Wie zuvor erwähnt gibt das Ortungslevel die Genauigkeit des Ereignisses an. Daher folgen die Ortungslevel auch einer Baumstruktur wobei ein Level entweder das Toplevel sein kann, also das allgemeinste Level das die Wurzel des Baumes bildet, oder ein Elternlevel haben kann, das über ihm steht. Unter allen Ortungslevels gibt es zwei besondere:

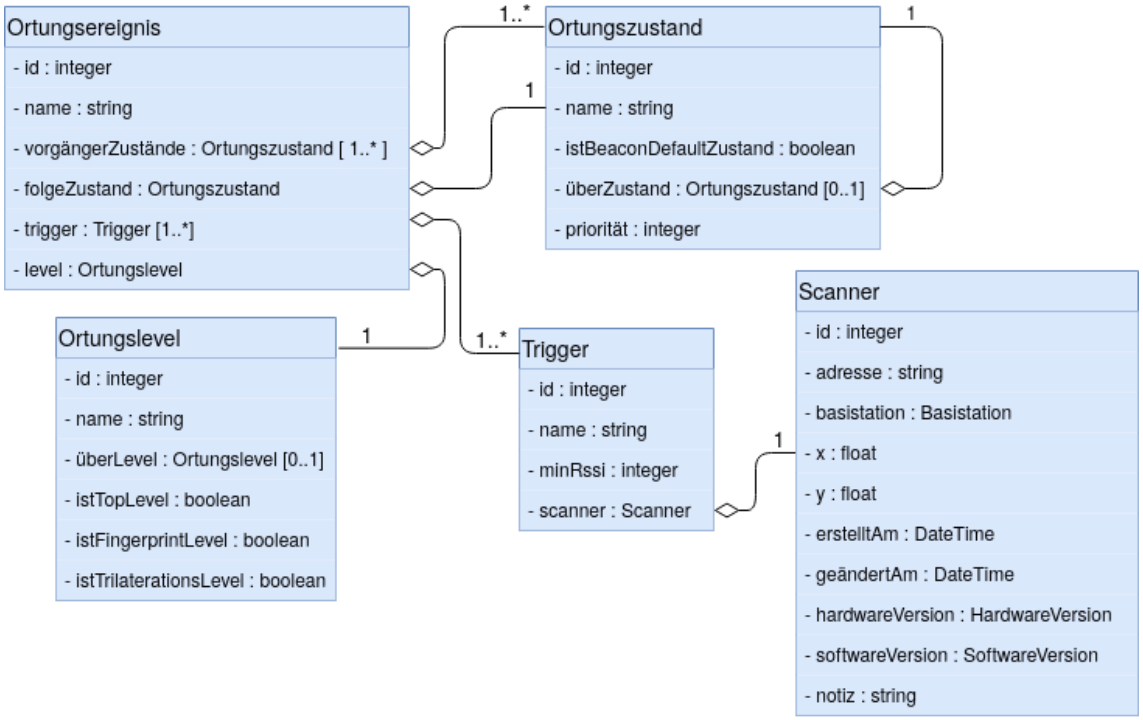


Abbildung 3.7: Das `Ortungsereignis` hat mehrere Ausgangs- oder Vorgängerzustände sowie einen Folgezustand. Es wird durch einen Trigger, also einen RSSI Wert der von einem Scanner empfangen wurde und über dem gesetzten Grenzwert liegt ausgelöst. Das `Ortungslevel` bietet eine Hierarchie der Ortungsereignisse.

das Fingerprintlevel und das Trilaterationslevel. Das Fingerprintlevel ist hierarchisch über dem Trilaterationslevel und löst das Sammeln der Daten zum Generieren der Featurevektoren bzw. der Fingerprints aus. Bei Ortungsereignissen mit Trilaterationslevel oder einem Level unter diesem wird die Positionsberechnung durchgeführt. Die genaue Verwendung von Trilaterations- und Fingerprintlevel wird in Abschnitt 3.4 und Abschnitt 3.6 genauer ausgeführt. Ähnlich wie die Ortungslevel sind auch die Ortungszustände hierarchisch aufgebaut. Ein Zustand umfasst alle Kinderzustände, wie zum Beispiel die „*Tiefgarage*“ im Beispiel Abbildung 3.5 alle „*Stellplätze*“ umfasst.

Die Klasse `Trigger` enthält nur die Information, welche Mindestsignalstärke am gewählten Scanner empfangen werden muss, um das Ortungsereignis auszulösen. Die optimale Konfiguration des RSSI Grenzwertes von einem Trigger ist eine Herausforderung, welche zu zwei Problemen führen kann. Ist der Grenzwert zu hoch gesetzt, so wird der Trigger nicht ausgelöst und das Fahrzeug nicht dem Bereich zugeordnet. Ist er zu niedrig gesetzt so werden an einer Position mehrere Trigger gleichzeitig ausgelöst. Das Beacon wechselt also von „*Zustand A*“ zu „*Zustand B*“ und wieder zurück. Dieser Vorgang wird als Flapping bezeichnet. Um eine gute Abdeckung der Standfläche mit Triggern zu bewirken, also großzügigere RSSI Grenzwerte zu erlauben, wurde das Feld Priorität beim Ortungszustand eingeführt. Mit dieser Eigenschaft kann angegeben werden welchen Zustand des Beacon

im Fall von Flapping behalten soll. Die Priorität der Zustände wird meist so festgelegt, dass sie in Fahrtrichtung zunimmt. Um auch Spezialfälle wie das Fahren entgegen der Fahrtrichtung zu unterstützen ist in Zukunft geplant die Priorität durch das Auslesen der Beschleunigungswerte des Fahrzeugs zu ersetzen.

BLE Manager

Der BLE Manager ist die zentrale Verwaltungseinheit des Systems. Hier werden Beacons, Scanner und Basescanner sowie Ortungsereignisse eingetragen und konfiguriert. Ein Dashboard gibt Auskunft über den (Live-)Zustand des Systems, zum Beispiel wie viele Basisstationen online sind oder wie die Datenrate der empfangenen Ortungsdaten ist.

BHM API

Die BHM API ist die API einer externen Anwendung, welche die Positionen der Fahrzeuge visualisiert, sowohl im Web als auch auf Anzeigetafeln. Sie ist kein direkter Teil des hier entwickelten Ortungsframeworks.

Trilateration Manager

Der Trilateration Manager ist eine Anwendung, die allein im Zuge dieser Arbeit entwickelt wurde. Die Aufgabe des Trilateration Managers ist die Berechnung der standortgenauen Position der Fahrzeuge. Die Funktionalität der Anwendung wird in Abschnitt 3.4 genauer beschrieben.

KNN Manager

Der KNN Manager ist eine Anwendung die, gleich wie der Trilateration Manager, im Zuge dieser Arbeit entwickelt wurde. Sie ist für das Fingerprinting und die Positionsberechnung mithilfe des K-Nearest Neighbors Algorithmus zuständig. Die zwei Hauptaufgaben des KNN Managers bestehen aus dem Generieren von Trainingsdaten und der Positionsberechnung eines Beacons mithilfe des KNN Algorithmus. Ersteres verwendet die vom Concentrator generierten Fingerprints, die Nachteinteilung als Quelle für Groundtruth Information über die Standorte der Fahrzeuge in der Nacht sowie die Fahrzeuginformationen der BHM API. Letzteres benötigt den aktuellen Fingerprints des Beacons und die gesammelten Trainingsdaten. In Abschnitt 3.6 wird auf diese Aufgaben näher eingegangen.

BHM GUI

Die BHM Graphical User Interface (GUI) wird verwendet, um die Bluetooth-Hardwarekomponenten vor dem erstmaligen Deployment aufzusetzen sowie initiale Messungen von Parametern durchzuführen. Zu den Hauptaufgaben zählen das Programmieren der Hardware mit dem Bootloader, Stack und der Application sowie das Flashen von Sign und Encryption Keys. Weitere Aufgaben sind das Registrieren des Gerätes im BLE Manager und das Drucken eines Tags mit Typ, ID und Adresse zur Kennzeichnung des Gerätes. Zusätzlich dazu wird die BHM GUI verwendet, um sowohl den RSSI Wert der Beacons in ein Meter Entfernung als auch den initialen Beschleunigungssensorwert in der Produktionsumgebung jedes einzelnen Beacons zu messen und an den BLE Manager weiterzuleiten.

3.4 Trilateration

Die Positionsbestimmung durch Trilateration wurde in Form einer Processing Pipeline umgesetzt. Im Folgenden werden die Schritte, begonnen beim Sammeln der Daten im Concentrator über die Anbindung des Trilateration Managers an diesen bis hin zum Ablauf der Positionsberechnung und der Verarbeitung des Positionsergebnisses genauer beschrieben.

Vorbereitung Concentrator

Als zentrale Datenverwaltungsstelle ist der Concentrator dafür verantwortlich, die Daten für die Trilateration zu sammeln. Nachdem es nicht sinnvoll ist, die Position eines Fahrzeugs zu berechnen, solange dieses noch in Bewegung ist, versucht der Concentrator zu erkennen, ab wann das Fahrzeug eine stehende Position eingenommen hat. Die Berechnung der Position soll also ressourcenschonend durchgeführt werden, während sie gleichzeitig immer aktuell sein sollte, also auch, wenn ein Fahrzeug, nachdem es geparkt wurde, noch auf einen anderen Standplatz umgestellt wird.

Um zu erkennen, wann ein Fahrzeug in Bewegung ist, werden die in Unterabschnitt 2.2.3 bereits erwähnten Featurevektoren verwendet. Sie sind eine Datenstruktur, welche die Basis der Fingerprints, wie sie bei der Ortungsmethode mit Machine Learning verwendet werden, bilden. Der Unterschied zu den Fingerprints ist, dass die Featurevektoren neben den RSSI Werten noch zusätzlich Informationen enthalten. Welche Daten das sind, wird in Abschnitt 3.6 genauer beschrieben.

Um die Featurevektoren so klein wie möglich zu halten und damit eine gute Performance von Operationen, die mit ihnen durchgeführt werden, zu gewährleisten, werden Daten für sie nur dann gesammelt, wenn das Beacon sich in Zuständen von Ortungsereignissen mit explizitem oder implizitem Fingerprintlevel (FPZ) bzw. in Zuständen von Ortungsereignissen mit explizitem oder implizitem Trilaterationslevel (TLZ) befinden. Alle Level unter Trilaterationslevel zählen implizit ebenfalls als Trilaterationslevel. Ortungslevel zwischen Fingerprint- und Trilaterationslevel gelten als implizite Fingerprintlevel. Wann genau Daten gesammelt und wie die Informationen über das Ortungslevel verwendet werden ist in Abbildung 3.8 anhand eines Beispiels dargestellt.

Wenn ein Beacon zum Beispiel durch Eintreffen des Fahrzeugs im Betriebshof, von einem beliebigen Zustand ein Ortungsereignis auslöst, welches dem Fingerprintlevel entspricht, wird ein neuer Featurevektor für das Beacon erstellt und das Sammeln der Daten beginnt. Es ergeben sich drei Möglichkeiten für den weiteren Verlauf der Featurevektoren. Zum Ersten ist es möglich, dass das Beacon ein Ortungsereignis auslöst, welches ein Level hat, das über dem Fingerprintlevel steht. In diesem Fall wird der Featurevektor verworfen. Zum Zweiten kann entweder kein neues Ortungsereignis oder eines mit folgendem FPZ welcher jedoch kein TLZ ist ausgelöst werden, wie zum Beispiel durch das Eintreffen des Fahrzeugs in der Tiefgarage. Dann werden weiterhin die Daten nur in Form der Featurevektoren gesammelt.

Die dritte Möglichkeit ist, dass ein Ortungsereignis mit Trilaterationlevel oder darunter ausgelöst und das Beacon in einen TLZ versetzt wird. Das passiert beispielsweise, wenn das

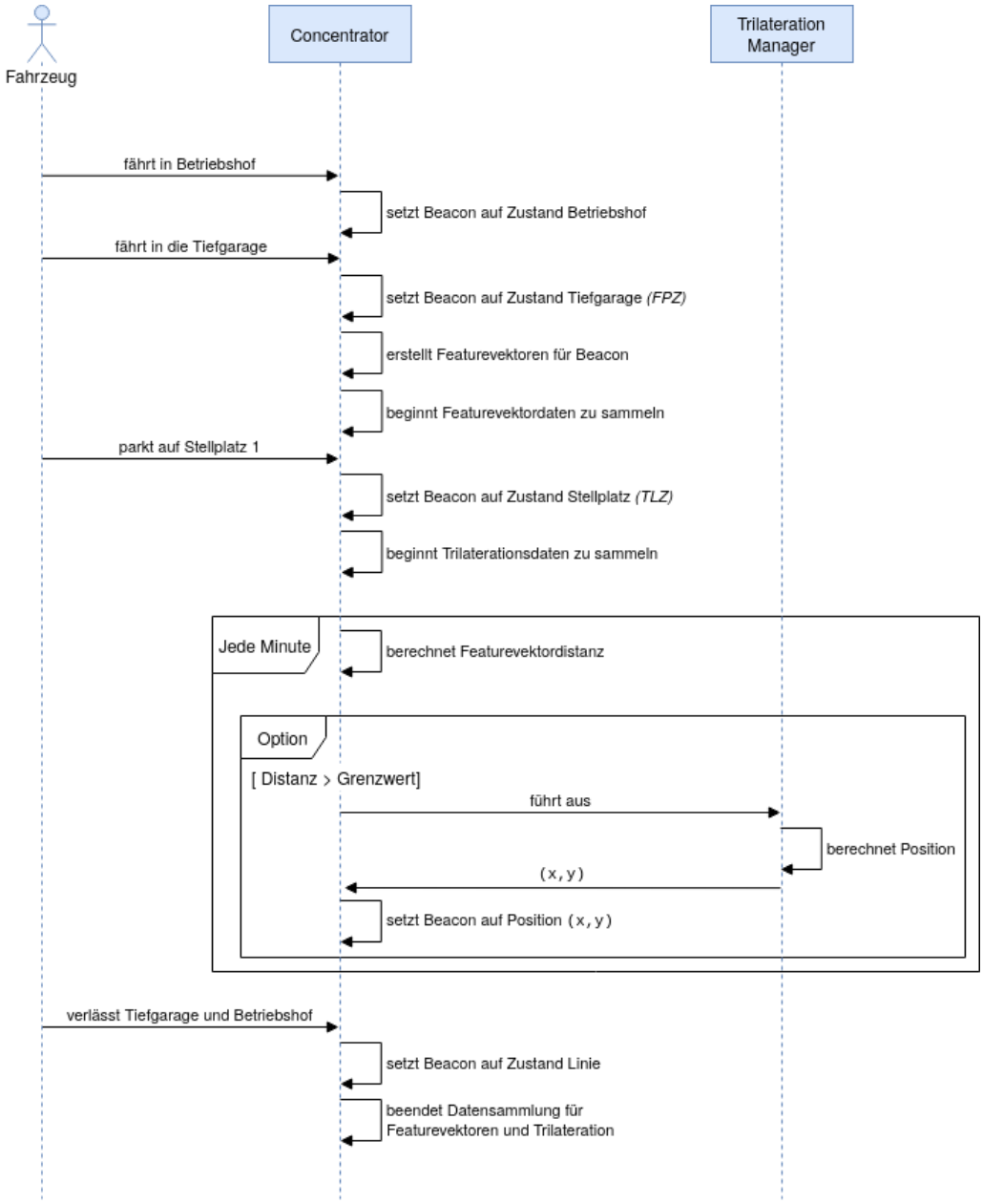


Abbildung 3.8: Sobald ein Fahrzeug in einen FPZ wechselt, beginnt das Sammeln der Daten für die Featurevektoren. Wenn das Fahrzeug einparkt, wechselt es in einen TLZ und die Daten werden an den Trilateration Manager weitergeleitet.

Fahrzeug im Bereich der Stellplätze unterwegs ist. In diesem Fall beginnt das Sammeln der Daten für die Trilateration. Zu diesem Zweck werden alle ankommenden RSSI Werte für dieses Beacon in eine `*.ble` Datei zusammen mit der Adresse des Scanners der den RSSI empfangen hat geschrieben. Für jedes Beacon werden in periodischen Abständen neue Dateien erstellt, damit die Daten die zur genauen Positionsberechnung verwendet werden nie älter als eine festgelegte Zeitspanne sind. Sobald also das Sammeln der Trilaterations-

daten begonnen hat, werden die Featurevektoren verwendet, um zu evaluieren, ob eine Neuberechnung der Position des Fahrzeugs notwendig ist. Wie zuvor erwähnt haben Busse auch nach Abstellen des Motors noch 20 Minuten Strom. Sobald die erste Berechnung des Standorts des stehenden Fahrzeugs durchgeführt wurde, sollte eine Neuberechnung der Position nicht mehr notwendig sein, da sich das Fahrzeug nicht bewegt hat. Um die Änderung zwischen dem aktuellen Standort des Fahrzeugs und dem vorigen zu erkennen, besitzt jedes Beacon zwei Featurevektoren. Einen Aktuellen und einen vorhergehenden. Alle n Minuten wird die euklidische Distanz zwischen diesen Vektoren berechnet, wobei n eine Zahl größer Eins ist, deren Wert über eine Konfigurationsdatei festgelegt wird. Ist die Distanz größer als ein festgelegter Grenzwert, wird angenommen, dass das Fahrzeug sich seit dem Sammeln der Daten für den vorigen Featurevektor bewegt hat, und die Position wird neu berechnet. Andernfalls wird das Beacon bei der periodischen Positionsberechnung ignoriert. Dies führt zu einem weiteren Einsparen der Rechnerressourcen.

Bei der Trilateration dient das Fingerprintlevel primär dazu die Berechnung der Vektordistanz effizient und gleichzeitig aussagekräftig zu halten. Würde man dafür nur die Scanner von Trilaterationslevel verwenden, so wäre die Anzahl der Scanner im Vektor kleiner und der Vektor würde sich häufiger ändern, was wiederum zu einer häufigeren Neuberechnung der Position führen würde. Falls ein Scanner nicht in beiden Featurevektoren enthalten ist, weil er gerade aus dem System entfernt oder zum System hinzugefügt wurde, wird dieser bei der Vektordistanz nicht mit einberechnet.

Positionsberechnung

Die Positionsberechnung wird vom Trilateration Manager durchgeführt und besteht aus drei Hauptschritten, welche ähnlich einer Pipeline durchgeführt werden, wie in Abbildung 3.9 zu sehen. Die Inputdatei vom Concentrator enthält für 1 Beacon N RSSI Werte von M Scannern. Nachdem die intern verwendete Datenstruktur initialisiert wurde und der Trilateration Manager die Position von den beteiligten Scannern vom BLE Manager abgefragt hat, beginnt der erste Schritt der Pipeline, nämlich die Filterung der Daten um Signalreflexionen bzw. -auslöschungen zu entfernen. Von den übrigen Daten wird ein Mittelwert berechnet, sodass nur mehr ein RSSI pro Scanner vorhanden ist. Anschließend werden diese Werte von der Signalstärke in dBm zu einer Distanz in Metern umgerechnet. Danach wird ein Trilaterationsalgorithmus ausgeführt. Da die Daten mehr Beacon-Scanner Abstände enthalten als die Algorithmen benötigen, werden die verbleibenden Distanzen aufsteigend sortiert und danach die drei bzw. k kürzesten zur Positionsberechnung durch Tri- bzw. Multilateration verwendet.

Es wurden verschiedene Filter und Algorithmen wie sie in Abschnitt 2.2 beschrieben sind implementiert. Das Ergebnis wird anschließend auf `stdout` ausgegeben.

Positionsverarbeitung

Der Concentrator verarbeitet die Ausgabe des Trialteration Managers und sendet sie zusammen mit weiteren Informationen über das Beacon an die BHM API.

Abbildung 3.9: Die Positionsberechnung erfolgt in drei Schritten: Zuerst werden die Daten gefiltert, danach werden die Signalstärken zu einer Distanz umgerechnet und anschließend werden sie dem Algorithmus übergeben, welcher dann die Position berechnet.

3.5 Angle of Arrival

Um die Positionsberechnung mittels Trilateration weiter zu verbessern, sollen AoA Daten zur Berechnung hinzugezogen werden. Der hierfür vorgeschlagene Ansatz ist es in jeder Halle an zumindest eine Basisstation ein zusätzliches Gerät, den AoA Detector, anzubringen. Zusätzlich dazu müssen einige Änderungen in mehreren Komponenten umgesetzt werden:

- Beacon Firmware: Diese muss um die Fähigkeit des Sendens von connectionless CTE für die Dauer von 30 s sowie einer Charakteristik zum Aktivieren von CTE erweitert werden.
- AoA Detector: Ein weiteres Hardwareelement muss entwickelt werden, welches mit einem Antennenarray ausgestattet und fähig ist, CTE zu scannen und den Winkel daraus zu berechnen. Der AoA Detector sollte nur auf Anfrage durch die Basisstation die Winkel scannen.
- Basisstation: Sie muss die Kommunikation mit dem AoA Detector bzw. die neuen Nachrichtenarten mit dem Concentrator implementieren.
- BGM Commander: Ein weiterer Command Line Interface (CLI) Befehl muss implementiert werden, um die CTE-Enable-Charakteristik vom Beacon zu setzen.
- Concentrator: Hier muss der Ablauf wie in Abbildung 3.10 gezeigt stattfinden. Nachdem Daten von einem Beacon empfangen wurden, welche ein Ortungsereignis auslösen, infolge dessen das Beacon in einen Trilaterationszustand versetzt wird, wird es zur AoA Queue seines aktuellen Standortes hinzugefügt. Jede Minute wird pro Standort ein Element der AoA Queue bearbeitet. Der erste Schritt dabei besteht darin, dass der Cocentrator mithilfe des BGM Commanders am Beacon das Advertising mit CTE aktiviert. Sobald das Beacon CTE aussendet, weist der Concentrator den Concentrator Client an Daten vom AoA Detector zu sammeln. Hierbei muss beachtet werden, dass jener Concentrator Client verwendet wird, an dem ein AoA Detector angeschlossen ist. Sobald keine AoA Daten mehr empfangen werden, sendet der Concentrator Client die Winkelinformationen an den Concentrator. Der Concentrator speichert die Winkeldaten zum betreffenden Beacon dazu und löscht den AoA Queue Eintrag. Beim nächsten Aufruf des Trilateration Mangers für dieses Beacon werden die Winkelinformationen zusammen mit den bisherigen Input Parametern an ihn übergeben.

- Trilateration Manager: Dieser soll nun beim Aufruf nicht nur die Trilaterationsdaten, sondern auch die Winkelinformationen erhalten und in seine Berechnungen miteinbeziehen (und zuvor filtern). Nur solche Schnittpunkte bzw. Positionen sollten beachtet werden, welche im gescannten Winkelbereich liegen. Wie das zum Beispiel aussehen kann, sieht man in Abbildung 3.11.
- BLE Manager: Hier muss die zusätzliche Information enthalten sein, an welche Basisstation ein AoA Detector angeschlossen ist und wie genau das Antennenarray ausgerichtet ist. Letzteres sollte möglichst genau eingetragen und vom Trilateration Manager abfragbar sein.

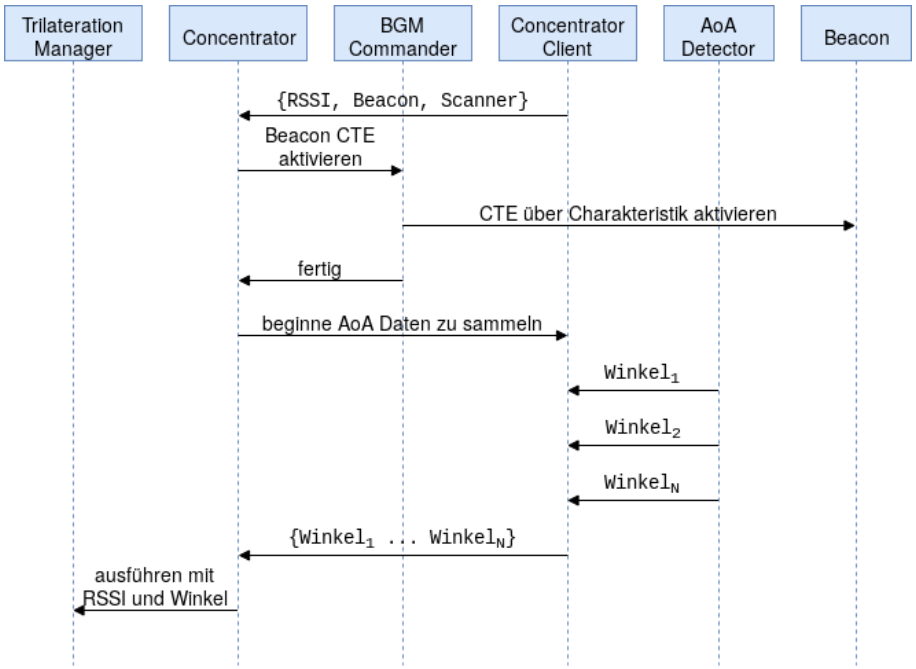


Abbildung 3.10: Um AoA Daten zu sammeln, muss der Concentrator die Aktivierung von CTE beim Beacon und das Sammeln der dadurch entstehenden Daten für alle Ortungsstandorte koordinieren.

3.6 Machine Learning: K-Nearest Neighbors

Die Positionsberechnung mithilfe von KNN besteht wie zuvor erwähnt aus Trainings- und Testphase. Der Fingerprint ist in beiden Phasen die zentrale Dateneinheit. Sein Aufbau ist entscheidend für die Genauigkeit der Ergebnisse des Algorithmus. Um den Ablauf der beiden Phasen zu verstehen, wird nun zuerst der Aufbau und die Gewinnung des Fingerprints und der hierfür benötigten Informationen beschrieben. Anschließend wird der Ablauf von Trainings- und Testphase genauer erklärt.

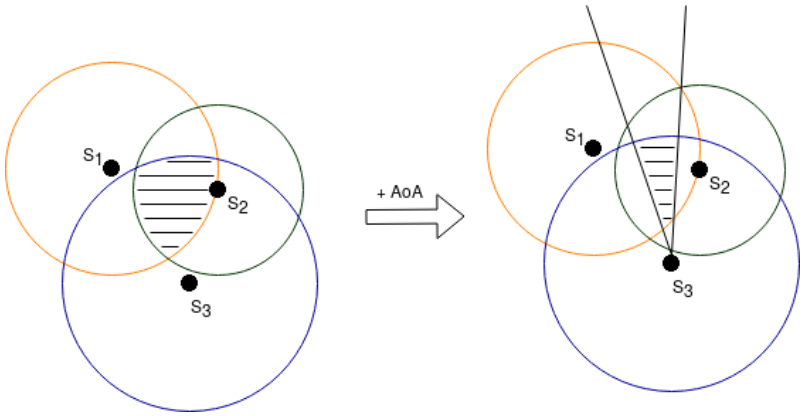


Abbildung 3.11: Die Information über den Empfangswinkel des Signals soll die Ortung mittels Trilateration weiter verbessern, indem eine Bounding Box die Fläche des möglichen Standortes weiter einschränkt.

Fingerprinting

Wie im vorigen Kapitel erwähnt, bezeichnet Fingerprinting den Vorgang bei dem Signalstärken an einem bekannten Ort gesammelt werden und die Information über Signalstärke, dessen Absender bzw. Empfänger und die aktuelle Position des Beacons abgespeichert werden. Im Vergleich zu bisherigen Arbeiten wie [MTP18] unterscheidet sich das hier durchgeführte Fingerprinting. Anstatt dass das Beacon die Signale aller Scanner (APs) empfängt und abspeichert, wird im Zuge dieser Arbeit die Signalstärke des Beacons an den Scannern gemessen und zusammen mit der bekannten Beaconposition gesammelt. Zusätzlich dazu ist es nicht möglich, das Fingerprinting genau so durchzuführen wie in den Vergleichsprojekten vorgeschlagen. Dafür müssten nämlich für alle Beacons in dichten Abständen auf allen verfügbaren Stellplätzen Fingerprints gesammelt werden. Durch die große Stellfläche, den aktiven Verkehrsbetrieb und die große Anzahl an Fahrzeugen ist diese Art von Fingerprinting nicht für den Einsatz im Produktivsystem geeignet. Das bedeutet, dass die Positionsbestimmung mittels Machine Learning möglicherweise nicht die gleiche Genauigkeit wie in Vergleichsprojekten angegeben erreichen kann. Dennoch ist es sinnvoll, eine Evaluierung der Methode und einen Vergleich mit den Trilaterationsergebnissen durchzuführen.

Die Aufgabe die Fingerprints zu erstellen übernimmt der Concentrator. Wie in Abbildung 3.12 zu sehen, hat jedes `Beacon` dafür zwei `Featurevektor`en. Diese beinhalten, wie in Abschnitt 3.4 schon erwähnt, neben der Fingerprintinformationen noch weitere Felder, welche den Erstellungszeitpunkt des Featurevektors angeben (`erstelltAm`) oder für welchen Standort er gültig ist (`erstelltDurch`, `standort`). Die tatsächlichen RSSI Werte finden sich dann in `data`, welches ein Dictionary von Scanneradressen mit `FeaturevektorDaten` ist. `FeaturevektorDaten` ist wiederum eine eigene Klasse, welche die letzten empfangenen Werte enthält. Zusätzlich dazu wird bei jedem hinzugefügten RSSI Wert ein laufender Mittelwert berechnet, welcher primär für die Evaluierung, ob eine Neuberechnung der Position eines Beacons notwendig ist, verwendet wird. Um zu verhindern, dass der Featurevektor eines Beacons alte Daten enthält, wird die Liste der Featurevekto-

ren in periodischen Abständen rotiert: Der aktuelle Featurevektor wird zum alten Featurevektor, während ein neuer Featurevektor erstellt wird. Ist der alte Featurevektor nicht älter als eine gewählte Zeitspanne so werden Daten von ihm übernommen, und zwar die letzten 10 RSSI Werte pro Scanner. Dies ist wichtig, um beim RSSI Mittelwert bzw. Median auch bei einem gerade neu erstellten Featurevektor nicht für Ausreißer anfällig zu sein. Somit sind bei einer Abfrage der Featurevektoren die Werte unabhängig vom Abfragezeitpunkt. Gleichzeitig muss verhindern werden, dass über mehrere Featurevektoren hinweg alte Daten weitergetragen werden, falls keine oder nur wenig neue während dieser Zeitspanne empfangen wurden. Dies wird durch die Zählervariable in `FeaturevektorDaten` garantiert. Sie enthält die Anzahl der übernommenen RSSI Werte. Es werden nun nur so viel Daten übernommen wie neu hinzugekommen sind.

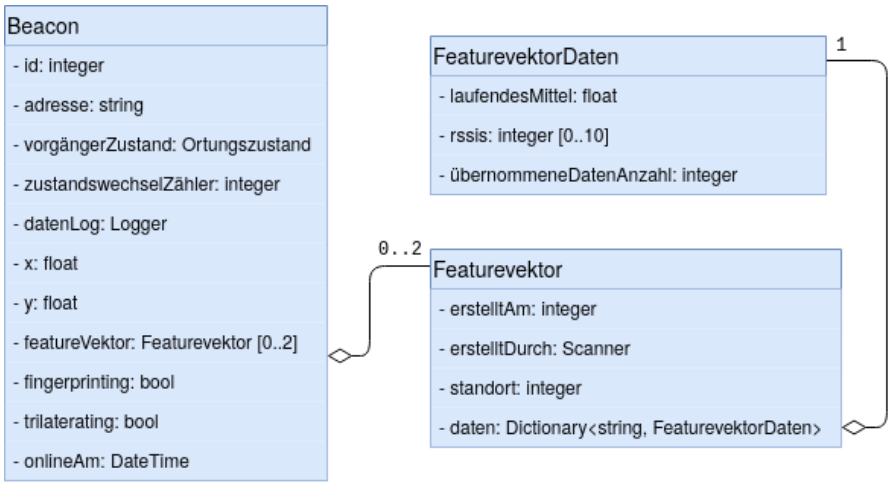


Abbildung 3.12: Neben den Feldern vom BLE Manager hat die Klasse `Beacon` im Concentrator noch das `datenLog`, welches das Logging der Trilaterationsdaten übernimmt. Die `Featurevektor`en werden zum Erstellen der Fingerprints verwendet. Wichtig dabei sind Erstellungszeitpunkt, Standort und die RSSI Daten.

Nachdem nun die Daten von der Concentrator API durch den KNN Manager abgefragt wurden, werden diese im Gegensatz zu anderen Arbeiten wie [MTP18] nicht als Pixelwerte in einer Bilddatei, sondern als Zahlenwert in Tabellenform abgespeichert. Da die Information über den groben Bereich und dadurch über den Standort des Beacons im Livesystem vorhanden ist, werden die Fingerprints noch zusätzlich nach Standort sowie nach Tag gruppiert. Dies führt dazu, dass sowohl die Fingerprints als auch die Anzahl der Datensätze pro Standort kleiner werden. Das wiederum hat positive Auswirkungen auf die Performance des KNN Algorithmus, da die Suche der k nächsten Nachbarn dadurch effizienter wird.

Tabelle 3.1 zeigt ein Beispiel für den Aufbau eines Fingerprints. Er besteht aus einer Kopfzeile mit der Position (x, y) und der Liste von Adressen aller Scanner, die zum gewählten Standort gehören. Welche das genau sind, kann vom BLE Manager abgefragt werden. Alle folgenden Einträge sind die empfangenen rohen RSSI Werte. Die Kopfzeile wird benötigt, um Flexibilität beim Meshaufbau zu gewährleisten, damit im Fall von

x	y	S_1	S_2	...	S_N
x_1	y_1	$rssi_{1,1}$	$rssi_{1,2}$	...	$rssi_{1,N}$
...					
x_M	y_M	$rssi_{M,1}$	$rssi_{M,2}$	...	$rssi_{M,N}$

$+S_{N+1}$

x	y	S_1	S_2	...	S_N	S_{N+1}
x_1	y_1	$rssi_{1,1}$	$rssi_{1,2}$	...	$rssi_{1,N}$	$rssi_{1,N+1}$
...						
x_M	y_M	$rssi_{M,1}$	$rssi_{M,2}$	...	$rssi_{M,N}$	$rssi_{M,N+1}$

$-S_1$

x	y	S_2	...	S_N	S_{N+1}
x_1	y_1	$rssi_{1,2}$	...	$rssi_{1,N}$	$rssi_{1,N+1}$
...					
x_M	y_M	$rssi_{M,2}$	...	$rssi_{M,N}$	$rssi_{M,N+1}$

Tabelle 3.1: Der Fingerprint ist in Tabellenform inklusive Kopfzeile aufgebaut. Die ersten zwei Spalten enthalten die x und y Werte der Position. Die restlichen Werte in der Reihe entsprechen den RSSI Werten welche an den Scannern S_1 bis S_N gemessen wurden.

Hardwareänderungen die bestehenden Fingerprints noch weiter verwendet und gleichzeitig leicht aktualisiert werden können. Es ergeben sich nämlich zwei Fälle. Zum einen ist es möglich, dass ein neuer Scanner an einem Standort montiert wird. In diesem Fall wird beim nächsten Sammeln von Trainings-Fingerprints der Datensatz dieses Tages um den neuen Scanner erweitert. Alle bestehenden Einträge in diesem Datensatz werden auf leer gesetzt. Zum anderen ist es möglich, dass ein bestehender Scanner entfernt wird. Dann wird beim Hinzufügen von einem Fingerprint zu einem bestehenden Datensatz der Scanner auf den Wert leer gesetzt. Sobald der Scanner auch aus dem BLE Manager entfernt wurde, ist er in der Kopfzeile aller folgenden Datensätze ebenfalls nicht mehr vorhanden.

Nachteinteilung

Die Nachteinteilung ist ein Parkplan der Fahrzeuge, welcher von den IVB erstellt wird und die benötigte Groundtruth-Information über den Standort der Fahrzeuge liefert. Der Plan beschreibt, wie die Fahrzeuge bei der Rückkehr in den Betriebshof abgestellt werden. Diese Informationen sind nur in Form von Stellplatzzuweisungen verfügbar und müssen daher in einem Zwischenschritt mithilfe der BHM API zu Positionen umgerechnet werden. Dies passiert, indem von der bekannten Position des ersten Fahrzeugs in einer Parkreihe die Länge aller stehenden Fahrzeuge sowie der Minimalabstand zwischen den Fahrzeugen dazugerechnet wird. Die Information über die Maße der Fahrzeuge ist über die BHM API verfügbar.

Ein Beispiel für einen Nachteinteilungsplan ist in Abbildung 3.13 zu sehen. Er dient dazu die Fahrzeuge so zu parken, dass sie am folgenden Tag bei der Ausfahrt einer be-

stimmten Reihenfolge nach die Halle verlassen können. Nachdem die Parkplätze sehr eng angeordnet sind, ist es nicht möglich, ein Fahrzeug in zweiter Reihe aus der Halle zu fahren, solange das Fahrzeug in erster Reihe noch an seinem Platz steht. Zum Beispiel kann ein Fahrzeug dessen Einsatz um 07:00 Uhr beginnt nicht vor einem Fahrzeug, welches ab 05:00 Uhr in Betrieb genommen werden muss, geparkt sein. Zusätzlich dient die Nachteinteilung auch dazu, dass die Fahrer ihre Fahrzeuge schneller finden. Das Absuchen der Standorte nach einzelnen Fahrzeugen vor Dienstbeginn würde zu lange dauern, da bei den IVB mehr als 200 Fahrzeuge im Einsatz sind. Bereits in der Tiefgarage, welche 16 Spuren aufgeteilt in 3 Hallen mit einer Gesamtfläche von über 10000 m^2 hat, ist das Auffinden eines Fahrzeugs ohne Plan nicht effizient möglich.

Abbildung 3.13: Die Nachteinteilung ist ein Plan welcher bestimmt wie die Fahrzeuge, zum Beispiel in der Tiefgarage, geparkt werden. Hierbei werden den Fahrzeugen Standplätze zugewiesen. Grün markierte Fahrzeuge sind in Planung.

Trainingsphase

Die Trainingsphase dient dazu, Datensätze mit Groundtruth-Information zu sammeln und daraus die Fingerprints zu berechnen. Während die RSSI Werte und der grobe Standortbereich vom Ortungsframework abgefragt werden können, werden die Groundtruth-Informationen mit den Positionen der Fahrzeuge, welche aus dem Nachteinteilungsplan berechnet wurden, von der BHM API abgefragt.

Das Sammeln der Daten für die Trainingsphase findet immer zur selben Tageszeit statt. Diese wurde so gewählt, dass alle Fahrzeuge exakt so geparkt sind, wie auf der Nachteinteilung festgelegt. Nachdem die Beacons in den Bussen, im Gegensatz zu jenen in den Straßenbahnen, nach Abstellen des Fahrzeugs nur begrenzt Strom haben, wird der letzte gültige Fingerprint gespeichert. Dies ist möglich, da die zuvor beschriebene Rotation der Featurevektoren nur dann stattfindet, wenn der aktuelle Featurevektor Daten enthält. Ist

dem nicht so, wird nur der Erstellungszeitpunkt aktualisiert und der vorige Featurevektor bleibt unverändert. Zusätzlich dazu wird bei einer Abfrage der Featurevektoren über die Concentrator API entweder der aktuellere ausgeliefert, falls dieser Daten enthält oder der vorherige. Das bedeutet, dass bei Abholung der Daten um 03:00 Uhr früh sowohl ein Fingerprint von 02:59 Uhr, zum Beispiel von einer Straßenbahn, als auch einer von 20:00 Uhr am Vortag enthalten sein kann. Diese Fingerprints werden dann mit der Information der Nachteinteilung ergänzt und als Trainingsdatensätze gespeichert.

Testphase

In der Testphase werden die Trainingsdaten zur Positionsberechnung verwendet. Die Berechnung der Position wird zum gleichen Zeitpunkt wie bei der Trilateration ausgelöst. Um die Position eines Fahrzeugs zu berechnen, wird zuerst ein aktueller Fingerprint erstellt. Dieser wird dann dem KNN Manager übergeben. Die Berechnung im KNN Manager besteht aus mehreren Schritten, wie in Abbildung 3.14 dargestellt. Zuerst wird der Ordner passend zu dem im Fingerprint angegebenen Standortbereich genommen. Danach werden all jene Scanner, die nicht im Beacon Fingerprint oder nicht in den Trainingsdaten vorkommen, entfernt. Die verbleibenden Spalten werden alphabetisch nach der Scanneradresse sortiert. Das führt dazu, dass die Zuordnung von Spaltenindex zu Scanner beim Beaconfingerprint genau die gleiche wie bei den Trainingsdaten ist. Der KNN Algorithmus erhält also als Input nur zwei eindimensionale Arrays. Aus diesen werden pro Trainingstag die K nächsten Nachbarn berechnet. Die Information über die Position und die Distanz des Fingerprints zum aktuellen Beacon Fingerprint wird dann in einer Tabelle gespeichert. Daraus ergibt sich eine Liste mit M Tagen $* K$ Nachbareinträgen. Diese Liste wird dann aufsteigend nach Distanz sortiert. Anschließend wird erneut der KNN Algorithmus angewandt, um aus den $K * M$ nächsten Nachbarn die gesuchte Position durch Regression als (gewichteter) Mittelwert der Nachbarpositionen zu berechnen. Diese Position wird dann vom KNN Manager als Ergebnis zurückgegeben.

3.7 Zusätzliche Quellen

Als zusätzliche Informationsquelle wird der am Beacon integrierte Beschleunigungssensor verwendet. Nachdem nur die Position von dem in Ruhe stehenden Fahrzeug berechnet werden soll, sollten die RSSI Daten vom bewegten Fahrzeug nicht in die Positionsberechnung mit einfließen. Der Beschleunigungssensor kann nun verwendet werden, um herauszufinden, ob das Fahrzeug fährt oder steht. Diese Information soll in Form eines Movementbytes (MVBs) vom Beacon mit den Advertisementdaten ausgesendet werden. Um das MVB berechnen zu können, müssen zuerst der Wertebereich fixiert und die Sensorwerte vermessen werden. Anschließend kann das MVB als Mittelwert der Sensorwerte der letzten 30 Sekunden berechnet werden. Nach Umrechnung, Normalisierung und ausreichend gutem Gravitationsausgleich des Bewegungsvektors erhält man das MVB. Dieses wird in Form der Manufacturer-Data mit dem Extended Advertising Datenpaket versendet. Die Scanner und Basescanner empfangen das MVB des Beacons und leiten es weiter an den Concentrator Client bzw. Concentrator. Am Concentrator wird über eine Konfigurationsdatei der MVB-Grenzwert eingestellt und alle ankommenden Datenpakete, deren MVB Wert über dem Grenzwert liegen, werden verworfen. Liegt die MVB-Information bei einem Daten-

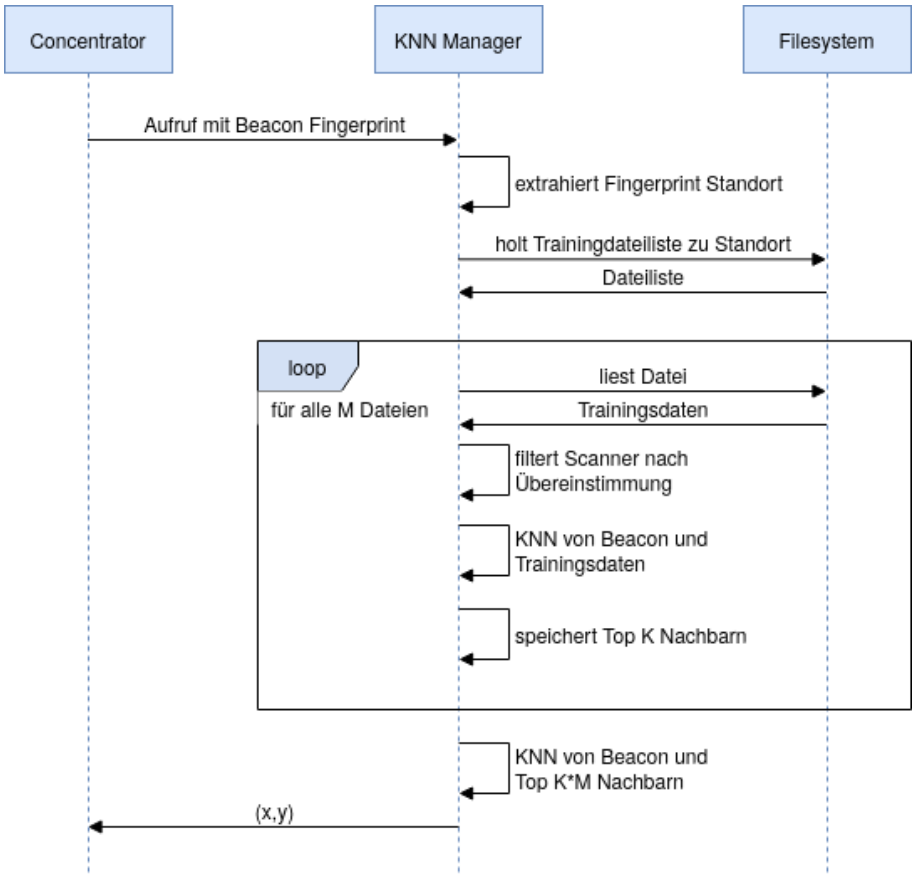


Abbildung 3.14: Für die Positionsberechnung im KNN Manager werden zuerst die Daten zum Standort herausgesucht, dann für jeden Trainingsdatensatz die N nächsten Nachbarn berechnet und aus all diesen anschließend die Position durch Regression berechnet.

paket nicht vor, zum Beispiel weil es noch nicht die aktuelle Firmware enthält, wird der Wert des MVBs als 0 angenommen. Standardmäßig wird also der Zustand jedes Fahrzeug ohne MVB als ruhend angenommen.

Kapitel 4

Implementierung

Das Ortungsframework wurde gebaut, um Fahrzeuge in großen teilweise unterirdischen Hallen zu orten. Die Hallen sind unterschiedlich gebaut und die vorhandenen Stellplätze sind sehr dicht angeordnet, wie in Abbildung 4.1 zu sehen. Neben den unterschiedlichen Hallen sind auch die Fahrzeuge verschieden: Sowohl Busse als auch Straßenbahnen sollen geortet werden. Während die Beacons in den Fahrzeugen montiert werden, werden die Scanner, Basescanner und Basisstationen möglichst hoch oben in den Hallen angebracht. Eine Halle ist mit mehreren Meshes ausgestattet, welche sie in mehrere Bereiche aufteilt. Ein Beispiel dafür ist in Abbildung 4.2 gegeben.

(a) Parkspuren in der Tiefgarage

(b) Fahrzeugordnung

Abbildung 4.1: Die Fahrzeuge werden in bis zu 5 m hohen Hallen abgestellt. Sie werden sehr dicht nebeneinander geparkt.

Die im vorigen Kapitel beschriebenen logischen Komponenten sind, wie in Abbildung 4.3 dargestellt, implementiert. Die für die Hard- bzw. Softwarekomponenten verwendeten Technologien werden im Folgenden genauer beschrieben.

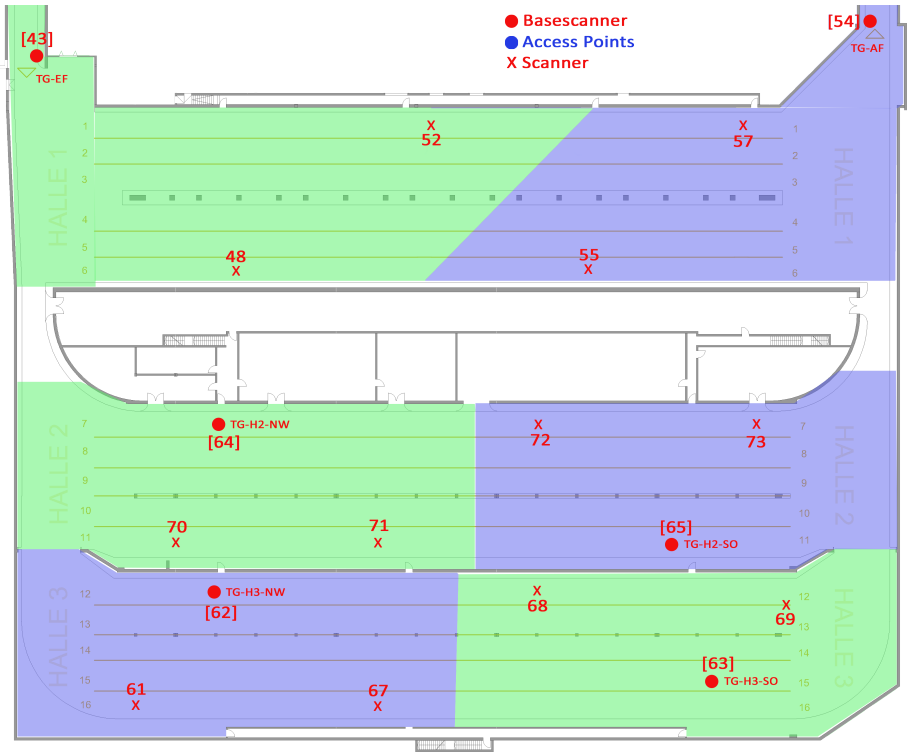


Abbildung 4.2: Der Plan zeigt eine Halle, welche mit sechs verschiedenen Meshes ausgestattet ist. Jedes Mesh besteht in diesem Fall aus einem Basescanner und zwei Scannern. Die Halle hat genau eine Einfahrt und eine Ausfahrt.

4.1 Ausgewählte Hardwarekomponenten

Alle vier Hardwarekomponenten haben ein Silicon Labs [sil] BGM13P22 Bluetooth Modul als zentrale Einheit. Während die Firmware der Beacons, welche aktuell im Produktivsystem läuft, auf der Gecko Software Development Kit (SDK) Suite v2.3.1[1] basiert, verwendet die Firmware der übrigen drei Bluetoothelemente, welche derzeit im Einsatz sind, den Bluetooth Mesh Stack v1.2.0.0[2].

Beacon

Das Beacon ist mit einem LSMAGR303 Beschleunigungssensor und Magnetometer ausgestattet. Wie in Abbildung 4.4 zu sehen wird es im Fahrzeug verbaut und sendet Advertisements mit einer Standard-TX Power von 8 dBm[3] in einem Sendeintervall zwischen 546.25 ms und 1285 ms aus.

Nachdem die ersten Trilaterationsberechnungen in der Testumgebung teilweise deutliche Abweichungen von der tatsächlichen Position ergeben haben, wurde die Beacon Firmware noch angepasst. Zusätzlich wurde Extended Advertising implementiert, um die

[1] ein Update auf die aktuellste Version 2.6 ist in Entwicklung
[2] ein Update auf die aktuellste Version 1.6 ist in Entwicklung
[3] die höchste Sendeleistung ohne Extended Advertising

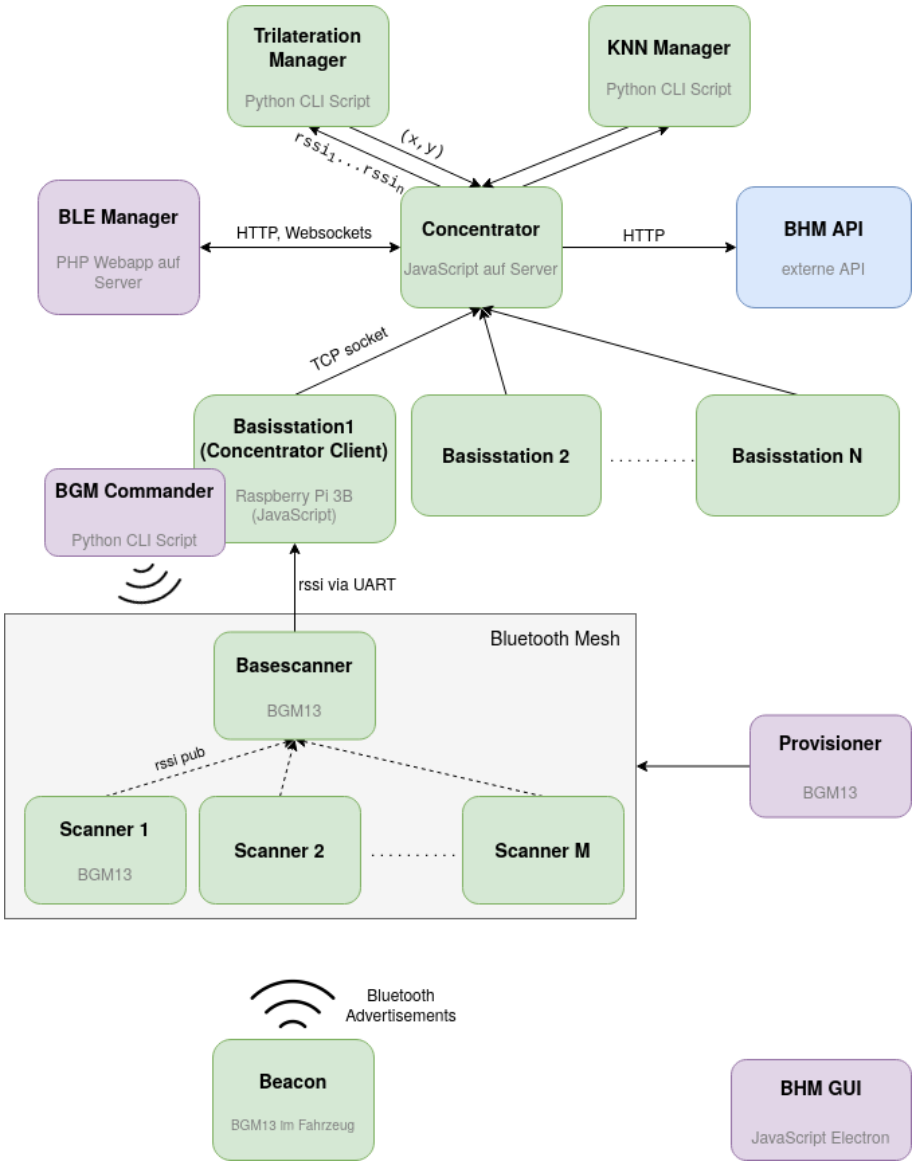
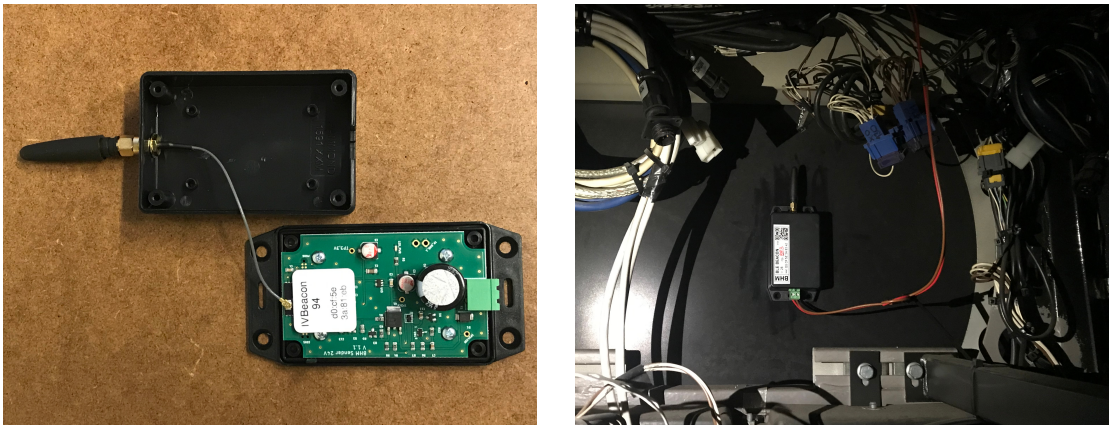


Abbildung 4.3: Die einzelnen Komponenten des Ortungsframeworks sind in verschiedenen Technologien umgesetzt und verwenden Bluetooth, UART, Hypertext Transfer Protocol (HTTP) und Transmission Control Protocol (TCP) bzw. Websockets zur Kommunikation. Die Farben wurden wie in Abbildung 3.1 gewählt: grün für die Datenverarbeitung, violett für die Systemverwaltung und blau für die Visualisierung.

Menge der Signalauslöschungen zu reduzieren. Hierbei werden neben den drei primären Kanälen auch die sekundären Kanäle 0 - 36 zum Versenden von Advertisingpaketen verwendet. Die Scanner und Basescanner Firmware wurde zu dem Zweck auch angepasst. Der (Base)Scanner scannt nun alle Kanäle im Round Robin Modus und signalisiert dem Beacon, welche Kanäle verwendet werden können. Dies ist wichtig, da der (Base)Scanner als Master auch gewisse Channels blockieren kann, falls zu viel Datenverkehr auf ihnen verläuft. Derzeit wurden keine sekundären Kanäle aktiv von der Firmware blockiert.

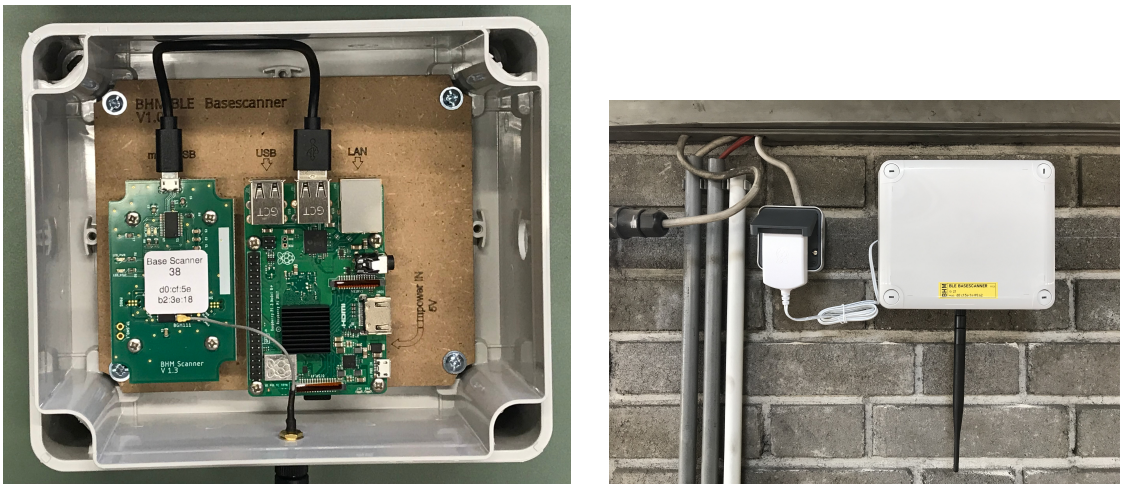


(a) Beacon (b) Beacon im Bus

Abbildung 4.4: Das Beacon wird in einem Gehäuse mit externer Antenne im Fahrzeug verbaut.

Scanner & Basescanner

Scanner und Basescanner kommunizieren innerhalb eines Meshs. Ihr Nachrichtenaustausch folgt dem Publish-Subscribe Pattern. Der Basescanner ist wie in Abbildung 4.5 zu sehen mit dem Raspberry Pi über UART verbunden und wird an die Wand montiert. Der Scanner hat ein ähnliches Gehäuse wie das Beacon und eignet sich auch zum Einsatz im Freibereich, wie in Abbildung 4.6 zu sehen.



(a) Basescanner und Raspberry Pi (b) Montage an der Wand

Abbildung 4.5: Basescanner und Raspberry Pi werden gemeinsam an den Wänden der Tiefgarage montiert.

(a) Scanner im Innenbereich (b) Scanner im Freibereich

Abbildung 4.6: Der Scanner eignet sich auch zur Montage im Freibereich.

Provisioner

Da niemals eine Nachricht an alle Meshes gesendet wird, wurde für jedes Mesh ein eigener Netzwerk- und Anwendungsschlüssel erstellt. Aufgrund des begrenzten Speichers des embedded Provisioners und der großen Anzahl an Scannerdaten besteht ein Mesh aus einem Basescanner und maximal sechs Scannern. Derzeit werden die Meshes vor der Montage eingerichtet. Der Einsatzort des Provisioners ist also das Büro. Nach jedem Mesh wird der Provisioner zurückgesetzt, damit nur ein Provisioner für alle Mesh benötigt wird.[4]

4.2 Softwarearchitektur

Die Softwarekomponenten wurden hauptsächlich in Hypertext Preprocessor (PHP), JavaScript und Python umgesetzt. Der Großteil der Kommunikation zwischen den Komponenten basiert auf HTTP bzw. TCP und Bluetooth.

Basisstation, Concentrator Client und BGM Commander

Die Basisstation ist ein Raspberry Pi 3B Modul mit einem Raspbian Image. Der Concentrator Client läuft auf dem Raspberry Pi Modul mithilfe von `node-pm`[5] dauerhaft. Er wird verwendet, um die Daten vom Basescanner via UART zu empfangen und über einen TCP Socket an den Concentrator weiterzuleiten.

[4] In Zukunft soll die Einrichtung der Meshes über eine neue Python Anwendung, den Mesh Manager passieren. Gesteuert werden soll dieser mithilfe des BLE Managers, welcher auch die aktuellen Schlüssel für jedes Mesh verwalten können soll.

[5] ein Node Package Manager (NPM) Paket das eine Node Anwendung als Service laufen lässt [nod]

Jede Basisstation besitzt eine eindeutige ID mit dazugehörigem Schlüssel und eine IP Addresse. Alle drei werden im BLE Manager festgelegt und sind für die Integration ins System wichtig. Der Schlüssel ist besonders wichtig für die Kommunikation mit dem BLE Manger, wie zum Beispiel der Upload von aktuellen Basisstationszustandsdaten. Er wird gemeinsam mit den IP Addressen über das Ticketsystem im BLE Manager konfiguriert. Der Concentrator Client ist in JavaScript implementiert und verwendet Node.js. Zusätzlich zu ihm ist auf der Basisstation der BGM Commander eingerichtet, welcher in Python 3.6 implementiert ist. Der BGM Commander verwendet die Bluetooth-Bibliotheken Bluepy und Bluez für die Bluetoothkommunikation über die interne Bluetoothantenne des Raspberry Pi.

Da die Basisstation über LAN oder WLAN mit einer über das Ticketsystem konfigurierten Adresse im internen Netz hängt, ist es möglich, eine Secure Shell (SSH) Verbindung mit der Basisstation aufzubauen und so den BGM Commander aufzurufen. Dies ist vor allem wichtig, wenn man regelmäßige Systemstatuschecks durchführen will wie zum Beispiel die Überprüfung der Beschleunigungssensorcharakteristik der Beacons. Hierfür wird reihum auf jeder Basisstation der BGM Commander mit dem Befehl, die Beschleunigungssensorcharakteristik für alle Beacons auszulesen, die auf der Todo Liste stehen, aufgerufen. Diese Liste holt sich der BGM Commander vom BLE Manager ab. Nach jedem erfolgreichen Auslesen werden die Beschleunigungssensordaten an den BLE Manager geschickt. Dadurch enthält die Liste bei der nächsten Abfrage durch die folgende Basisstation nicht mehr bereits überprüfte Beacons.

Concentrator

Der Concentrator ist ebenfalls in JavaScript mit Node.js und Express umgesetzt und am Server eingerichtet. Er ist sowohl mit dem Internet als auch mit dem internen Netz der IVB verbunden. Über mehrere TCP Socketverbindungen werden die Daten von den Concentrator Clients empfangen. Gleichzeitig holt sich der Concentrator Systeminformationen vom BLE Manager und schickt diesem Daten von aktuellen (Ortungs-)Ereignissen für das Dashboard mithilfe einer Websocketverbindung. Der Datenaustausch mit der BHM API läuft ebenfalls über HTTP.

BLE Manager

Der BLE Manager ist eine PHP 7 Webanwendung, welche die zu verwaltenden Systeminformationen in eine Microsoft SQL Server (MSSQL) Datenbank speichert und mittels REST API externen Anwendungen zur Verfügung stellt. Über Asynchronous JavaScript and XML (AJAX) Requests und die Websocketverbindung werden Livedaten des Systems vom Concentrator abgefragt und anschließend am Dashboard dargestellt.

Der aktuelle **Systemstatus** und Umfang kann am Dashboard des BLE Managers betrachtet werde, wie auf dem Screenshot in Abbildung 4.7 zu sehen ist. Das aktuelle Produktivsystem umfasst derzeit 45 Basisstationen, 101 (Base-)Scanner und 252 Beacons. Von diesen werden zu Nichtstoßzeiten in 10 Minuten durchschnittlich 137000 RSSI Werte gemessen.

Um System- und Serverausfälle schnell zu erkennen, wurden beim BLE Manager in Zusammenarbeit mit der BHM API **Health Checks** umgesetzt. Der BLE Manager und die BHM API liegen auf unterschiedlichen Servern. In regelmäßigen Abständen überprüfen beide Anwendungen ob der Server der jeweils anderen Anwendung erreichbar ist und ob ausgewählte APIs die erwarteten Daten liefern. Auch die Load und die Serverlaufzeit wird zu diesem Zweck, ausgelöst durch einen Cronjob, abgefragt. Im Fehlerfall wird je nach Schwere einmalig oder in regelmäßigen Abständen eine E-Mail versendet.

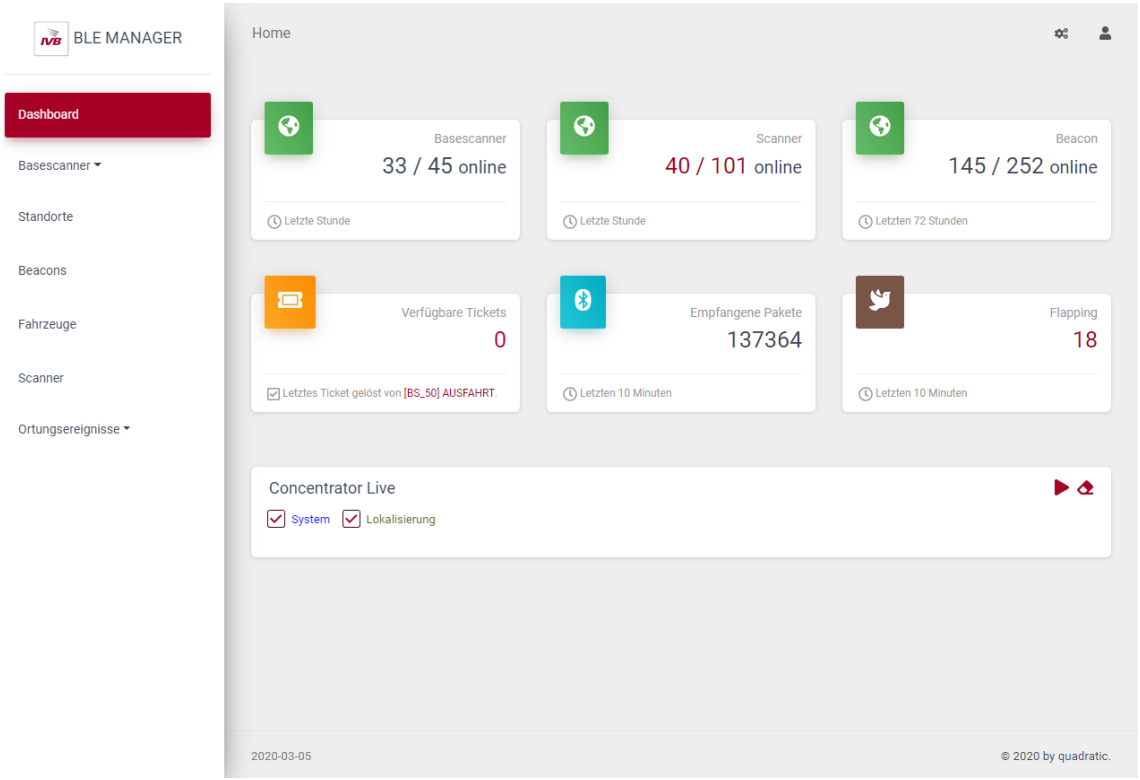


Abbildung 4.7: Das Dashboard des BLE Managers bietet eine Live-Übersicht über den Zustand der Hardwarekomponenten sowie die Zahl der zu verarbeitenden Pakete. Weiters stehen aktuelle (Ortungs-)Ereignisse des Concentrators On Demand zur Verfügung.

BHM API

Die Kommunikation mit der BHM API erfolgt über HTTP.

BHM GUI

Die BHM GUI ist eine Node.js Desktopanwendung, die Electron als Basisframework verwendet und mit dem BLE Manager über HTTP kommuniziert.

4.2.1 Systemupdates

Beim Erweitern oder Aktualisieren des Ortungsframeworks müssen bestimmte Abhängigkeiten zwischen und innerhalb von vier Gruppen von Komponenten beachtet werden. Diese Abhängigkeiten sind in Abbildung 4.8 zu sehen. Generell folgt der Aufbau und das Zusammenspiel der Hard- und Softwarekomponenten einer Art von Wasserfallmodell. Bei diesem gilt, je näher man von reinen Softwarelösungen am Server zur Firmware der Hardwarekomponenten kommt, umso aufwändiger ist das Update dieser Komponente und umso mehr Vorbereitungen, teilweise durch Updates von darüberliegenden Komponenten, wird benötigt.

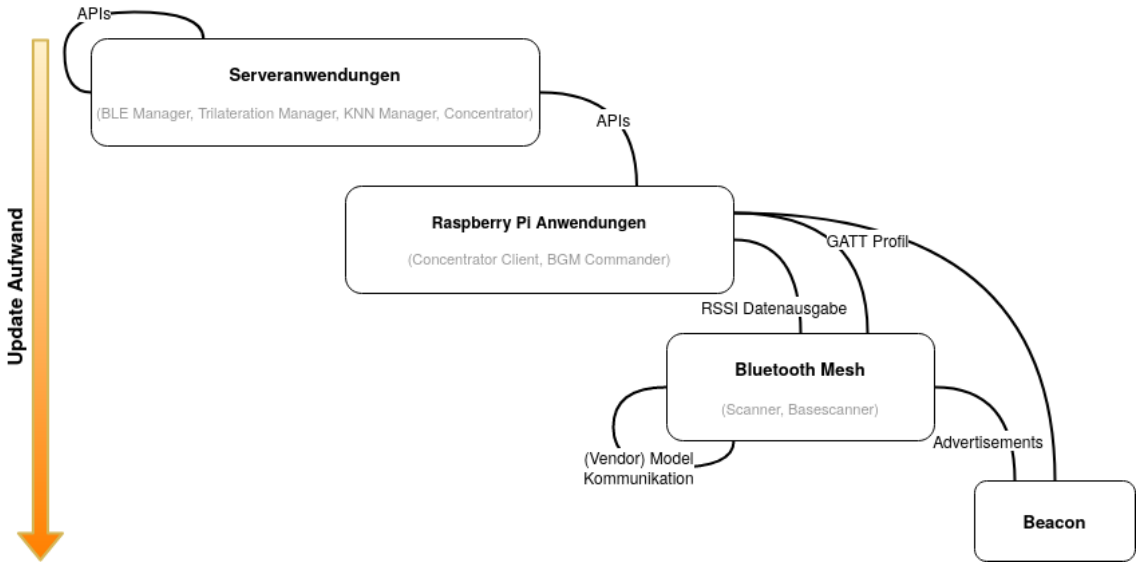


Abbildung 4.8: Abhängig davon welcher Teil des Systems aktualisiert wird ergeben sich unterschiedliche Abhängigkeiten zu anderen Komponenten. Firmware Updates sind aufwendiger als Software Updates.

An oberster Stelle stehen reine Serveranwendungen. Die zwei Schlüsseleigenschaften dieser Gruppe liegen darin, dass zum Ersten nur eine Instanz der Anwendung existiert. Zum Zweiten können Aktualisierungen und Patches mit verhältnismäßig geringem Aufwand eingespielt werden. Dies geschieht normalerweise durch Einloggen auf dem Server gefolgt von dem Download der neuesten Änderungen. Anwendungen dieser Ebene umfassen im aktuellen Ortungsframework den BLE Manager, Trilateration Manager, KNN Manager und den Concentrator. Diese Anwendungen bieten der Ebene darunter APIs, welche nicht verändert, sondern stattdessen erweitern, werden sollen. Dies ist nicht nur wichtig für die Kommunikation zwischen den Anwendungen dieser Ebene, sondern vor allem wichtig für die Anwendungen der nächsten Ebene, die diese APIs konsumieren.

Die nächste Gruppe umfasst alle Anwendungen, die mit der Verarbeitung der Rohdaten zu tun haben und auf den Raspberry Pis laufen. Letzteres bedeutet, dass es mehrere Instanzen derselben Anwendung gibt. Soll nun zum Beispiel ein Patch für den Concentrator Client eingespielt werden, so muss dies auf allen Raspberry Pis gemacht werden.

Ist ein Raspberry Pi nicht erreichbar, zum Beispiel weil die automatische Konfiguration der Ethernetverbindung nicht funktioniert hat, oder gibt es allgemein eine neue Version des Raspberry Pi Images, so muss die Secure Digital Memory Card (SDMC) des Gerätes vor Ort ausgetauscht werden. Da diese im Regelfall an der Decke der Halle montiert sind, bedeutet das, dass sowohl eine Hebebühne organisiert werden muss, um die Hardware zu erreichen, als auch ein Zeitfenster, an dem in diesem Bereich nicht gearbeitet wird und sich kein Fahrzeug befindet. Eine besondere Herausforderung ist dies in Bereichen wie der Waschstraße oder der Werkstatt. Zusätzlich dazu muss nach Austauschen der SDMC erneut über den BLE Manager ein Ticket für diese Basisstation angelegt werden, um ID und Netzwerkkonfiguration zu setzen. Weiters ist besonders beim Concentrator Client darauf zu achten, dass alle Ausgaben der Basescanner verarbeitet werden können. Ist dies nicht der Fall, wäre ein Verlust der Daten des gesamten dazugehörigen Meshs die Folge. Neben dem Concentrator Client gehört auch der BGM Commander zu dieser Ebene.

Noch etwas schwieriger gestaltet sich das Update bei Basescanner und Scanner. Neben dem Datenaustausch mit der oberen Ebene über UART oder Bluetooth GATT ist hier besonders die Kommunikation innerhalb des Meshs wichtig. Diese funktioniert, wie bereits erwähnt über den Austausch von Meshnachrichten. Derzeit werden vor allem Vendor Model Nachrichten ausgetauscht. Es gibt verschiedene Typen von Nachrichten, zu denen unterschiedliche Payloads gehören. Weiters werden je nach Model die Nachrichten an andere Gruppenadressen geschickt. Es ist besonders wichtig darauf zu achten, dass zu jedem Zeitpunkt alle gesendeten Nachrichten vom Empfänger gelesen werden können. Es dürfen also keine Fehler bei der Konfiguration der Gruppenadressen passieren oder die Form der Payloads verändert werden. Beides würde zu einem Verlust der Ortungsdaten führen. Daher werden hier zwei Grundregeln eingehalten. Erstens dürfen Vendor Model Strukturen nie verändert werden. Sollen nun bei einem Payload eines Models zusätzliche Informationen mitgeschickt werden, so muss eine neue Nachricht oder ein neues Model erstellt werden, welches die Daten des bestehenden Models sowie die neuen umfasst. Die zweite Regel sagt aus, dass bei einer Erweiterung der Meshkommunikation immer zuerst der Gerätetyp aktualisiert werden muss, welcher die Nachricht empfängt. Erst danach soll der Sender aktualisiert werden.

Ein Beispiel für ein Vendor Model Update ist in Abbildung 4.9 zu sehen. Hierbei wurde das Sammeln der Advertisements bei den Scannern verändert. Anstatt dass jedes Advertisement sofort an den Basescanner gemeldet wird, werden nun bis zu 40 Advertisements pro Beacon gesammelt. Jede Sekunde werden dann die Daten von einem Beacon in der Liste an den Basescanner gesendet. Anstatt nun das alte Vendor Model zu verändern, wurde ein neues Model erstellt. Dies bedeutet auch, dass falls ein neuer Scanner zu einem alten Mesh hinzugefügt wird, der Basescanner zuvor aktualisiert werden muss.

Neben der Handhabung von Vendor Models ist das Initialization Vector Index (IVI) Update noch notwendig damit die Kommunikation auch mit neu zum Mesh hinzugefügten Nodes funktioniert.

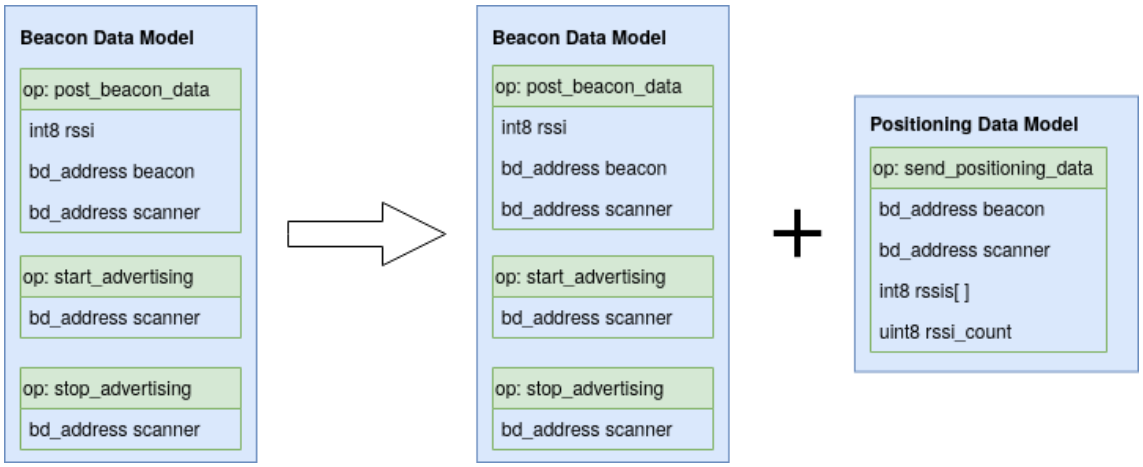


Abbildung 4.9: Will man die Daten die in einer Vendor Model Nachricht mitgesendet werden verändern oder erweitern so soll ein neues Vendor Model auf Basis des zu erweiternden erstellt werden.

IVI Update

Um Replay Attacken beim Datenpaketaustausch innerhalb eines Meshs zu verhindern, werden neben Netzwerk- und Anwendungsschlüssel, auch der IVI und die Sequencenumber verwendet [Blu19a]. Die Sequencenumber ist eine 24-Bit große Zahl die in jedem Paket, das verschickt wird, mitgesendet und anschließend um eins erhöht wird. Jedes Element im Bluetoothmesh hat eine eigene Sequencenumber. Damit auch nach dem Auslaufen der Sequencenumber, nach ca. 16.7 Millionen versendeter Pakete, weitere Datenpakete versendet werden können wird der IVI verwendet. Er ist 32 Bit groß und, im Gegensatz zur Sequencenumber, wird sein Wert innerhalb des Meshs synchronisiert. Der IVI wird erhöht, indem eine Node im Mesh, dessen Sequencenumber auszulaufen droht das IVI Update auslöst. Infolgedessen werden alle Nodes im Mesh in den Zustand IVI Update Ongoing versetzt. Dieser Zustand hält mindestens 96 Stunden an, damit alle Meshteilnehmer ihren IVI an den Mesh IVI anpassen. Eine Meshnode kann bis zu 42 IVIs hinten sein und trotzdem noch den aktuellen IVI mittels IVI Recovery aufholen bevor sie nicht mehr an der Meshkommunikation teilnehmen kann. Das bedeutet, dass beim späteren Hinzufügen von Nodes zu einem Mesh diese mit dem korrekten IVI initialisiert werden müssen.

Unabhängig vom IVI ist das Beacon. Es steht an unterster Stelle im Updateabhängigkeitsdiagramm. Das Aktualisieren der Firmware funktioniert gleich wie bei den Scannern, nämlich OTA. Allerdings sind die Beacons im Gegensatz zu den Scannern nicht immer in Reichweite eines BGM Commanders, welcher das OTA DFU durchführt. Zusätzlich sind sie nach Eintreffen im Betriebshof nur zeitlich begrenzt mit Strom versorgt.

Die größte Herausforderung beim Beacon Update liegt jedoch bei den Advertisements. Werden hier Änderungen gemacht so muss gewährleistet sein, dass die neuen Advertisements genauso von den Scannern und Basescannern empfangen werden wie die bisherigen. Das beste Beispiel hierfür ist die Einführung von Extended Advertising. Bei diesem werden nämlich zusätzlich zu den drei primären Bluetooth Kanälen auch die über 30 sekundären

zum Versenden der Advertisements verwendet. Im Fall von Problemen beim Empfang der Advertisements können ausbleibende Ortungsdaten oder sogar ein Fehlschlagen der Verbindung zum Beacon die Folge sein. Das würde bedeuten, dass auch ein Aktualisieren der Firmware via OTA DFU nicht mehr möglich ist. Das Einspielen des Patches zur Lösung des Problems müsste dann manuell über die Programmierschnittstelle erfolgen. Weniger gravierend, dennoch wichtig zu beachten, sind Änderungen die das GATT Profil betreffen. Hier gilt wie bei den anderen Bluetooth Hardwarekomponenten, dass zuerst der BGM Commander aktualisiert werden sollte.

4.3 Trilateration

Der Trilateration Manager ist eine Python 3.6 CLI Anwendung welche am gleichen Server wie der Concentrator liegt. Er wird vom Concentrator aufgerufen und schreibt das Positionsergebnis auf `stdout`. Es wurden alle Filter und Algorithmen implementiert, welche in Abschnitt 2.2 beschrieben wurden. Sowohl die Anzahl und Reihenfolge der Filter als auch die Wahl des Algorithmus sind flexibel konfigurierbar. Es gibt zwei `main.py` Dateien: die erste, um eine Performance Evaluierung durchzuführen. Dies passiert, indem alle Filter-Algorithmen Kombinationen ausgeführt werden. Das Ergebnis wird hierbei in eine `*.csv` Datei geschrieben. Die zweite `main.py` Datei ist für den Einsatz in der Produktionsumgebung. Hier wird nur ein Filter und ein Algorithmus ausgeführt, und zwar jene, die bei der Evaluierung die besten Ergebnisse gezeigt haben. Der Performancevergleich wird im nächsten Kapitel genauer beschrieben.

4.4 Angle of Arrival

Für die Implementierung des AoA Ansatzes zur Verbesserung der Ortung wird sowohl ein Antennenarray als auch die Unterstützung von CTE durch den Bluetoothstack benötigt. Da Letzteres zum Zeitpunkt der Arbeit noch beim Hersteller in Entwicklung und dadurch nicht für Testzwecke verfügbar war, wurde das AoA Design nicht umgesetzt.

4.5 Machine Learning: K-Nearest Neighbors

Der KNN Manager wurde ebenfalls als Python 3.6 CLI Anwendung umgesetzt. Er hat zwei Einstiegspunkte.

Der erste wird zur Berechnung der Trainingsdaten aus den Fingerprints des Concentrators in Form einer `*.json` Datei und der Nachteinteilung, welche als `*.xlsx` und `*.txt` vorliegt, verwendet. Die Trainingsdaten werden nach Tag und Standort gruppiert und in eine `*.csv` Datei gespeichert. Die Nachteinteilung ist immer bereits einige Stunden vor dem nächsten Tagesbeginn verfügbar.

Um 03:00 früh lädt ein Cronjob zuerst die Featurevektoren des Concentrators und anschließend den Nachteinteilungsplan. Der Zeitpunkt wurde so gewählt, dass möglichst alle Fahrzeuge bereits auf ihren zugewiesenen Standplätzen geparkt und gleichzeitig noch keine für den Frühbetrieb ausgefahren sind. Die Parkordnung sollte zu diesem Zeitpunkt

also am ehesten dem Nachteinteilungsplan entsprechen. Das Berechnen der Trainingsdaten findet daher immer statt, sobald alle Daten verfügbar sind. Es wird ebenfalls über einen Cronjob ausgelöst.

Der zweite Einstiegspunkt bestimmt die Position eines Beacons auf Basis von dessen Fingerprint (ebenfalls als `*.json` Datei verfügbar). Hierfür werden alle Trainingsdaten des Standorts verwendet. Die Position wird auf `stdout` ausgegeben. Über eine Konfigurationsdatei ist es möglich, die Anzahl der Nachbarn und die Art ihrer Gewichtung beim Ausführen des KNN Algorithmus anzugeben. Zur Umsetzung des KNN Algorithmus wurde SciPy verwendet.

4.6 Zusätzliche Quellen

Für die Umsetzung des MVB wurde die Firmware des Beacons angepasst. Jede Sekunde liest das Beacon nun einen 3D Beschleunigungswert vom LSM303AGR Modul aus, verarbeitet ihn und schreibt ihn anschließend in einen Ringbuffer. Es werden bis zu 30 Werte gleichzeitig gespeichert. Aus diesen wird dann das MVB berechnet. Die Schritte, die notwendig sind um von einem 3D Beschleunigungsvektor zu einem `uint8` MVB zu kommen sind in Abbildung 4.10 zu sehen. Zuerst müssen die Achsenwerte auf einen Bereich von 0 g bis 1 g normalisiert werden. Dafür wird die in einem Vorbereitungsschritt gesammelte Information über den Wertebereich, den der Sensor zwischen 0 g und 1 g ausliefert verwendet. Nach der Division der Achsenwerte durch die maximale Beschleunigung von 1 g werden alle Werte die nun über 1.0 liegen, bei welchen also eine Beschleunigung von mehr als 1g vorliegt, auf 1.0 gesetzt, da nicht die genaue Geschwindigkeit des Fahrzeugs, sondern nur die Tatsache, ob es in Bewegung ist, benötigt wird.

Als Nächstes wird die Länge des normalisierten Vektors berechnet. Da bei der Beschleunigung keine Achse vernachlässigt wurde, damit die Berechnung auch noch funktioniert, falls das Gerät nach der Montage seine Position verändert, muss nun noch die Beschleunigung durch die Erdanziehungskraft aus dem Vektor gerechnet, also seine Länge korrigiert werden. Hierfür wird von der Vektorlänge noch 1.0 oder genauer gesagt $\sqrt{0^2 + 0^2 + 1^2}$ abgezogen. Da die Kraft, die in Z Richtung wirkt, durch die Fahrtrichtung beeinflusst werden kann, zum Beispiel beim Einfahren des Fahrzeugs in die Tiefgarage, kann das Subtrahieren des fixen Wertes dazu führen, dass das Fahrzeug öfter als stehend identifiziert wird als es tatsächlich ist. Dies kann jedoch durch eine Anpassung des MVB Grenzwertes gelöst werden. Nach Ausgleich der Erdanziehung wird vom Ergebnis der Betragswert genommen. Dies ist wichtig, falls es zum Beispiel durch eine Bodenwelle eine Beschleunigung entgegen der Erdanziehung gegeben hat, welche die Kraft in Richtung z-Achse verändert. Negative Vektoren würden also auch wieder auf eine Bewegung hinweisen. Nach diesem letzten Verarbeitungsschritt wird das Ergebnis nun als `float` in den Ringbuffer geschrieben.

Nimmt man nun den Durchschnittswert aller Werte im Ringbuffer und bildet diesen auf einen Wertebereich von 0 bis 255 ab, so erhält man, nach einem Rundungszwischenschritt, ein MVB das nun zum Beacon Advertisement hinzugefügt werden kann. Je höher sein Wert, umso mehr hat sich das Fahrzeug in den letzten 30 Sekunden bewegt.

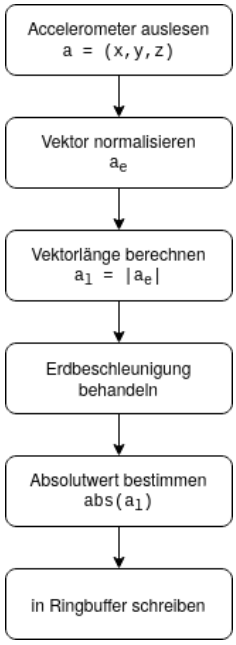


Abbildung 4.10: Die Transformation erfolgt, indem der Beschleunigungsvektor zuerst normalisiert wird. Anschließend wird dessen Länge berechnet und die Erdbeschleunigung rausgerechnet. Der Absolutwert der Länge wird dann gespeichert.

Kapitel 5

Ergebnisse

Zur Evaluierung der Genauigkeit der Ortungsmethoden wurden Daten in der Produktivumgebung gesammelt. In den folgenden Abschnitten wird der Testaufbau genauer beschrieben. Anschließend folgt eine Auswertung der Trilaterationsergebnisse. Wie zuvor erwähnt, war es zum Zeitpunkt dieser Arbeit noch nicht möglich das Design von AoA umzusetzen, da der Hard- und Softwaresupport noch nicht zur Verfügung stand. Auch die anderen beiden Verbesserungsmethoden konnten nicht getestet werden, da hierfür das Sammeln von Daten über einen längeren Zeitraum bzw. die Durchführung von Updates aller Komponenten notwendig ist.

5.1 Testaufbau

Wie zuvor erwähnt wurden die Tests in der Produktivumgebung durchgeführt. Hierfür wurde die Tiefgarage als Testort verwendet. Während die Scanner und Basescanner des Produktivsystems und daher auch alle (Software-)Komponenten darüber verwendet werden konnten, wurde ein extra Beacon zum Testen fix an der Wand der Tiefgarage montiert. Die Position dieses Beacons wird für die folgenden Auswertungen berechnet.

Die Tiefgarage wurde bereits im Zuge der Systemeinrichtung vermessen. Während die Position der Scanner und Basescanner sowohl für die Trilateration als auch für die Machine Learning Methode wichtig ist, wurden noch extra für die Auswertung der Groundtruthdaten beim Machine Learning die Parkreihen und die Position des ersten Parkplatzes pro Reihe vermessen. Die genannten Positionen sind auf dem Plan der Tiefgarage in Abbildung 5.1 markiert.

5.2 Trilateration

Zur Evaluierung der Genauigkeit der Trilateration wurden Daten von einem Testbeacon gesammelt und ausgewertet. Die Daten wurden an einem Werktag zwischen 17:00 und 18:00 Uhr gesammelt. Das Beacon dessen Position bestimmt werden soll befand sich in der Tiefgarage mit den Koordinaten $(x, y) = (58.38, 83.00)$. Der RSSI in 1 m Entfernung betrug -23.1 dBm. Der Datensatz umfasste 290 RSSI Werte empfangen von sieben Scannern innerhalb einer Minute.

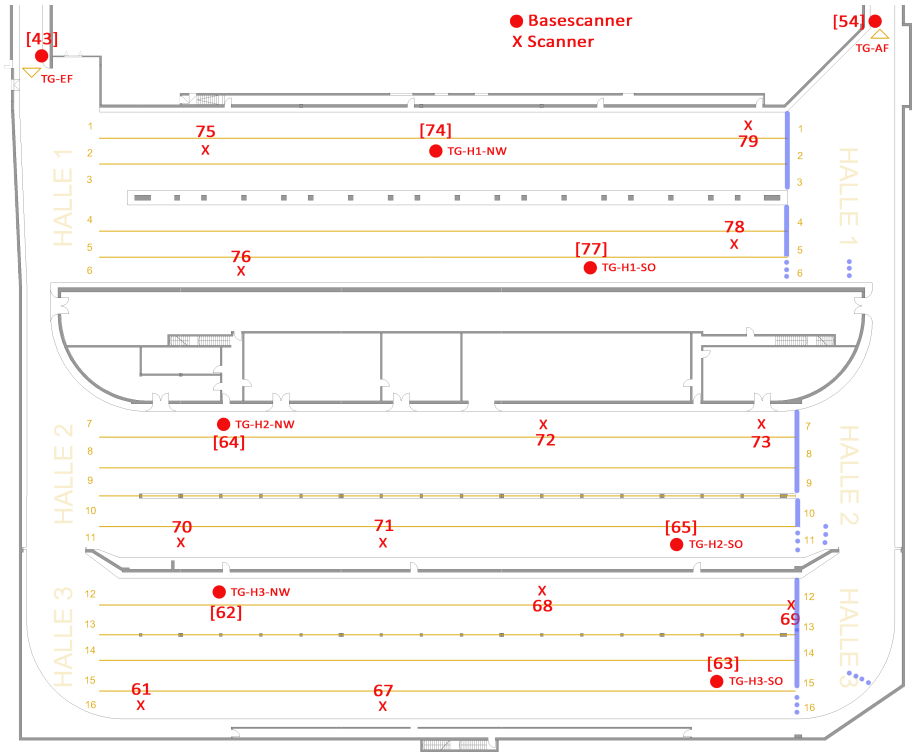


Abbildung 5.1: Die Tiefgarage ist mit mehreren Scannern (x) und Basescannern (o) ausgestattet welche die Advertisements des Testbeacons sammeln. Die blauen Linien und Punkte geben den Beginn des ersten Fahrzeugs in jeder Reihe an.

Im ersten Schritt wurde die Position des Testbeacons von allen Algorithmen mit allen Filtern berechnet. Wie in Abbildung 5.2 zu sehen liefern der ECA und der WCL Ansatz die besten Ergebnisse. Im Gegensatz dazu haben die verschiedenen Filter auf diese beiden Algorithmen keine gravierenden Auswirkungen.

Um den besten Algorithmus zu finden, wurden die Algorithmen mit nur einem Filter, dem Averaging Filter, ausgeführt. Wie in Abbildung 5.3 zu sehen, liefert Trilateration Centroid kein Positionsergebnis aufgrund der fehlenden Überschneidungen der stärksten drei Signale (kleinsten drei Kreise). Die Least Squares Estimation Methoden schneiden auch hier schlechter ab als ECA und WCL. Nachdem ECA nicht immer ein Ergebnis liefert, da es auf eine Anzahl an Überschneidungen angewiesen ist, wurden sowohl für ECA als auch für WCL die Auswirkungen der Filterwahl evaluiert. Die genaue Abweichung der berechneten Positionen sind in Tabelle 5.1 zu sehen.

Nachdem nun die Auswahl der Algorithmen beschränkt wurde, müssen die Filter evaluiert werden. Abbildung 5.4 zeigt die besten drei Filter bei ECA. Diese sind Moving Average, Averaging und Standard Deviation. Bei WCL liegen Median und Gauss vorne, wie in Abbildung 5.5 zu sehen. Die Tabelle 5.2 zeigen die Unterschiede im Mean Squared Error (MSE).

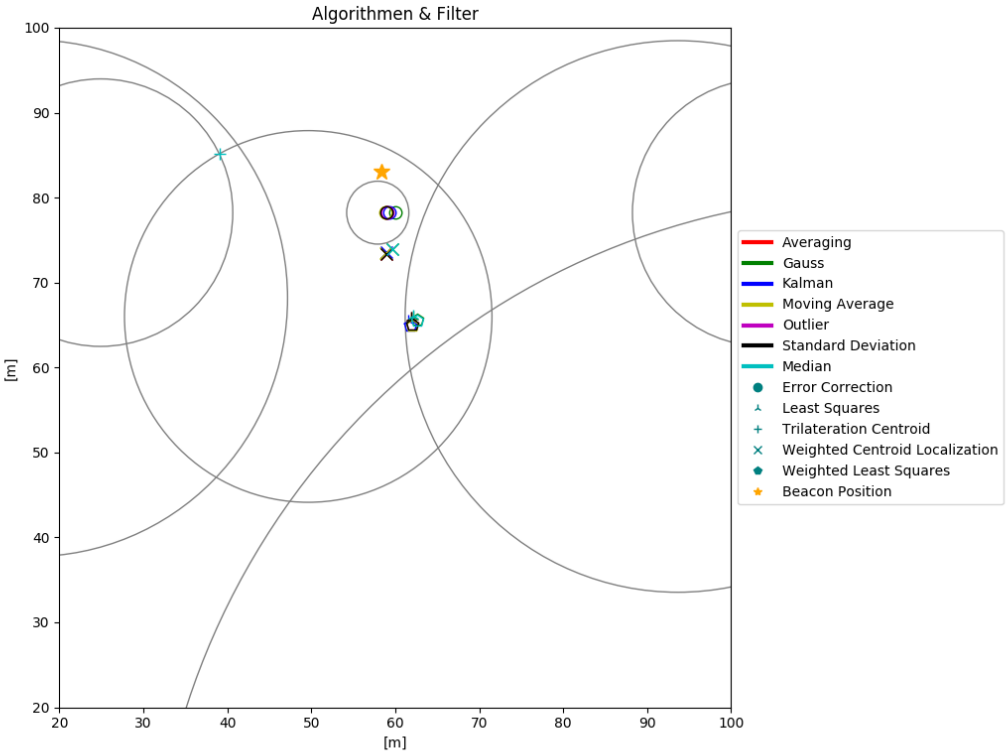


Abbildung 5.2: Während die verschiedenen Algorithmen (kodiert durch Symbole) unterschiedliche Genauigkeiten liefern ist der Unterschied zwischen den Filtern (farbcodiert) geringer.

Algorithmus	x	y	MSE
Error Correction	58.95	78.23	4.80
Least Squares Estimation	61.92	65.83	17.53
Weighted Centroid Localization	58.97	73.41	9.61
Weighted Least Squares Estimation	62.00	65.03	18.33

Tabelle 5.1: Beim direkten Vergleich der Algorithmen wird sichtbar, dass ECA gefolgt von WCL die besten Ergebnisse liefert. Der Trilateration Centroid Algorithmus konnte kein Ergebnis berechnen.

RSSI Grenzwert

Neben der Verwendung verschiedener Filter und Algorithmen wurde auch die Auswirkung von der Anwendung eines RSSI Grenzwertes evaluiert. Es wurden also Signalstärken, die über dem gewählten Grenzwert lagen verworfen. Anschließend wurde die Position unter Verwendung des Averaging Filters und der verschiedenen Algorithmen berechnet und die Distanz zur tatsächlichen Beacon Position mit den verschiedenen Grenzwerten berechnet. Das Ergebnis ist in Abbildung 5.6 zu sehen. Aus der Grafik lässt sich erkennen, dass ein

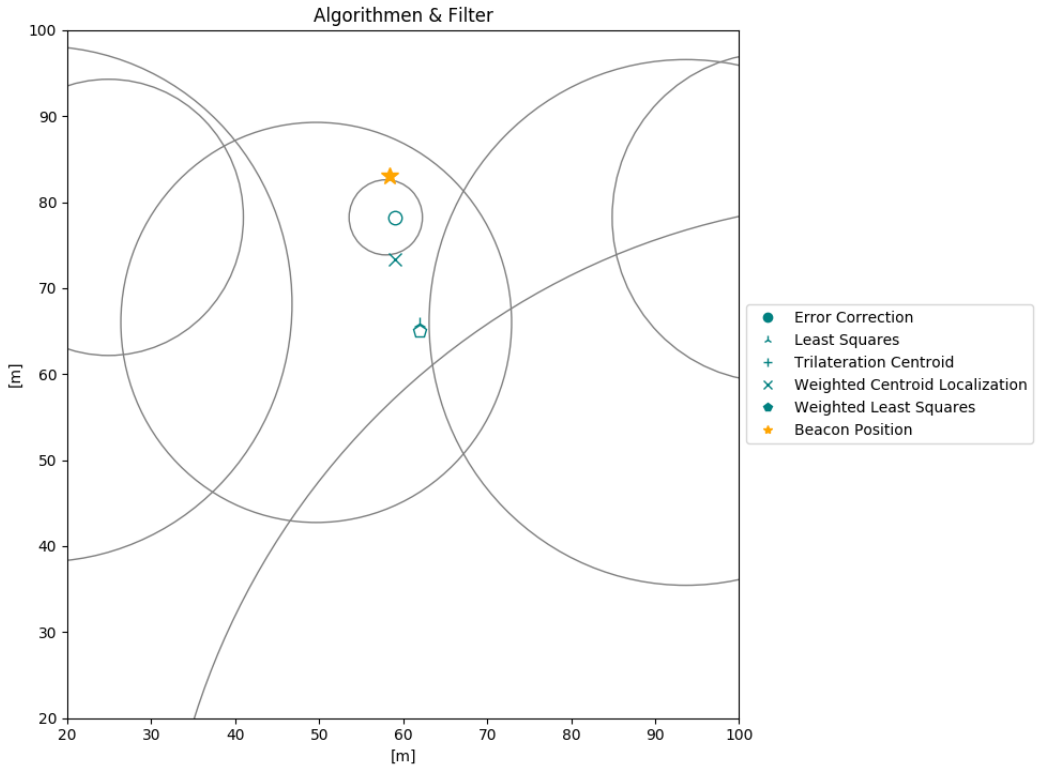


Abbildung 5.3: Der ECA und die WCL Methode liefern die besten Ergebnisse. Die Least Squares Estimation Algorithmen schneiden schlechter ab und der Trilateration Centroid Algorithmus, aufgrund von fehlenden Überscheidungen der drei größten Signalstärken, kann kein Ergebnis berechnen.

Filter (ECA)	x	y	MSE
Averaging	58.95	78.23	4.80
Gauss	59.98	78.23	5.03
Kalman	59.27	78.23	4.85
Moving Average	58.78	78.23	4.79
Outlier	59.02	78.23	4.81
Standard Deviation	58.95	78.23	4.80

Filter (WCL)	x	y	MSE
Averaging	58.97	73.41	9.61
Gauss	59.63	73.93	9.16
Kalman	58.92	73.64	9.38
Moving Average	58.88	73.41	9.60
Standard Deviation	58.97	73.41	9.61
Median	59.63	73.90	9.19

Tabelle 5.2: Der MSE wird sowohl bei ECA als auch bei WCL kaum durch die Anwendung verschiedener Filter verbessert. Die Verwendung von Median Filter bei ECA bzw. Outlier Filter bei WCL führt dazu, dass kein Ergebnis berechnet werden kann.

RSSI Grenzwert zwischen -75 dBm und -80 dBm die Ergebnisse fast aller Algorithmen verbessern kann. Wird der Grenzwert jedoch zu hoch angesetzt, so wird der MSE bei den Least Squares Estimation Algorithmen rasch größer. Zusätzlich zeigt der Plot, dass

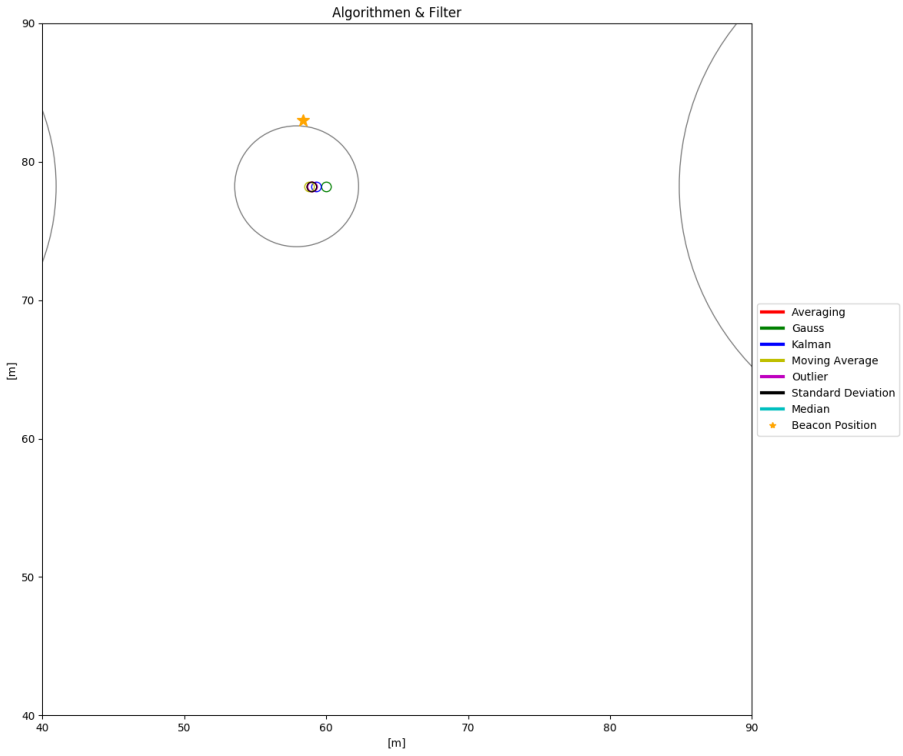


Abbildung 5.4: Die Ergebnisse des ECA können durch den Einsatz von Filtern nur gering verbessert werden. Die größte Verbesserung liefert der Moving Average Filter.

der Algorithmus Trilateration Centroid nur in einem bestimmten Grenzwertbereich ein Ergebnis liefert. Die Grenzwerte in diesem Bereich führen auch zu einer Verbesserung des ECA.

Extended Advertising

Um die Trilaterationsergebnisse noch weiter zu verbessern, wurde Extended Advertising implementiert. Da die neue Firmware ein umfassendes Update des Großteils der Bluetooth Hardware Module benötigt hätte, welches zum aktuellen Zeitpunkt aus den zuvor genannten Gründen nicht möglich war, wurde der Effekt von Extended Advertising im Vergleich zum Advertising ohne sekundäre Kanäle in einer anderen Testumgebung evaluiert. Die Umsetzung von Extended Advertising war so konfiguriert, dass keine sekundären Kanäle aktiv blockiert wurden. Zur Evaluierung wurden Daten von einem Beacon über den Zeitraum von zwei aufeinanderfolgenden Stunden gesammelt. In der ersten Stunde wurde eine Firmware mit dem Advertising ohne sekundäre Kanäle getestet. Für die zweite Stunde wurde das Beacon mit der Firmware mit Extended Advertising programmiert. Aus den Signalstärken, welche von einem Basescanner, der 1 m entfernt zum Beacon positioniert war, empfangen wurden, wurde dann die Verteilung der RSSI Werte berechnet. Die normalisierten Histogramme beider Berechnungen sind in Abbildung 5.7 zu sehen. Die

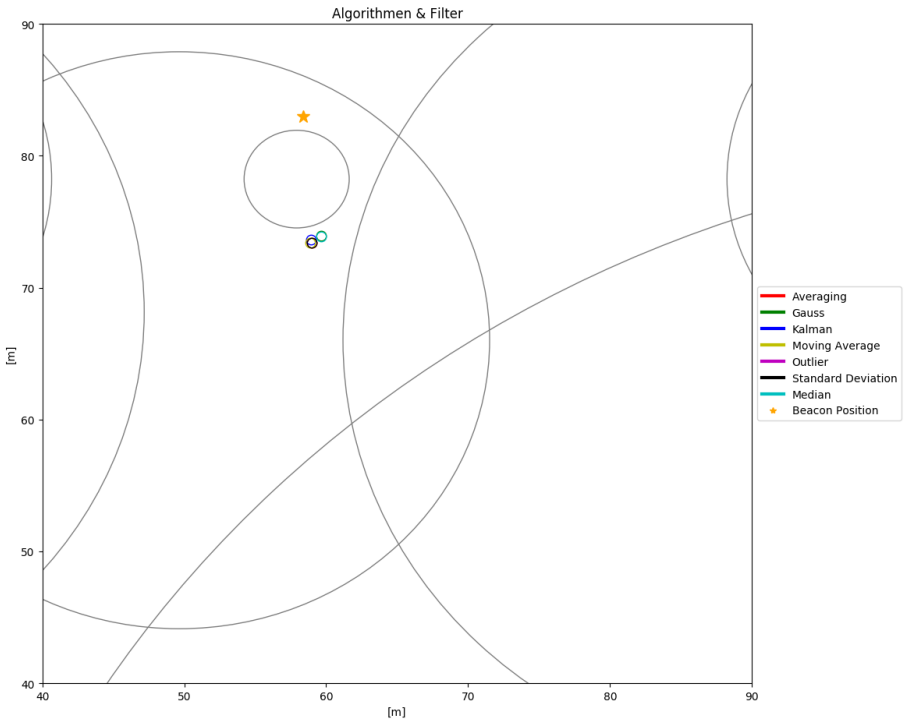


Abbildung 5.5: Die Ergebnisse des WCL können durch den Einsatz von Filtern, ähnlich wie bei ECA, nur gering verbessert werden. Median und Gauß bieten hier die größte Verbesserung.

Grafiken zeigen einen deutlichen Unterschied in der Streuung der RSSI Werte bei normalem und bei Extended Advertising. Zum einen sind mehr Ausreißer beim herkömmlichen Advertising zu sehen. Zum anderen ist der Anteil der stärkeren Signale beim Extended Advertising höher, da durch die Nutzung der sekundären Kanäle weniger Signalreflexionen vorkommen.

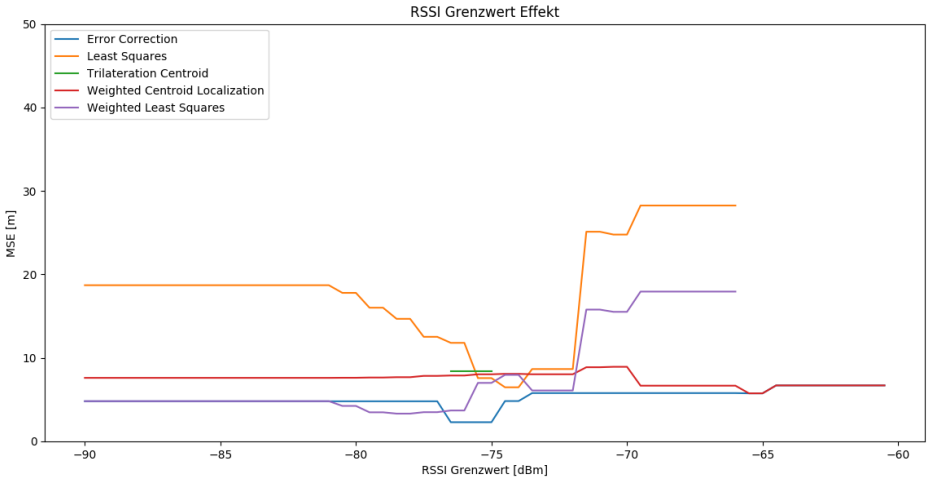
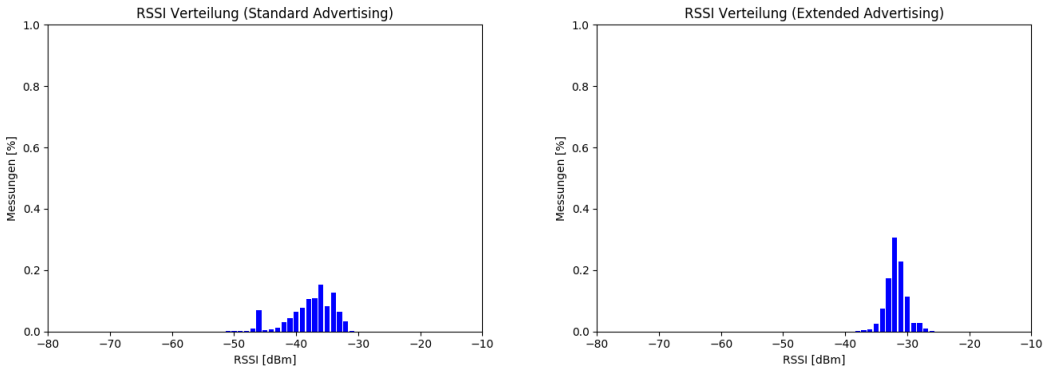


Abbildung 5.6: Der optimale RSSI Grenzwert für die Testumgebung liegt zwischen -75 dBm und -80 dBm. Wird der RSSI Grenzwert zu hoch gewählt kann das Ergebnis auch schlechter werden.



(a) RSSI Verteilung Standard

(b) RSSI Verteilung Extended

Abbildung 5.7: Durch die Verwendung von Extended Advertising entstehen weniger Signalreflexionen und -auslöschungen. Dies führt zu einer gleichmäßigeren Verteilung der RSSI Werte mit kleinerer Varianz.

Kapitel 6

Schlussbemerkung und Ausblick

6.1 Schlussbemerkung

Die Berechnung der Position von unbewegten Fahrzeugen in Innenräumen ist die Motivation dieser Arbeit. Die Herausforderungen sind der Aufbau der Gebäude, in welchen die Standortbestimmung erfolgen soll sowie die große Anzahl und dichte Anordnung der Fahrzeuge. Die Umsetzung basiert auf der Verwendung von Hardwaremodulen, welche mittels RSSI kommunizieren. Zu Beginn dieser Arbeit sind bereits Standorte und Fahrzeuge mit der Hardware ausgestattet und die Verwaltung der Komponenten im Ortungsframework möglich.

Für die Positionsberechnung werden vier Methoden untersucht. Diese umfassen die RSSI-basierte Trilateration, das RTLS AoA, der Machine Learning Algorithmus KNN sowie die Verwendung der Beschleunigungsdaten des Beacons als zusätzliche Informationsquelle für den Bewegungszustand des Fahrzeugs.

Die Trilaterationsergebnisse zeigen, dass eine Positionsgenauigkeit von 2 - 3 m mit den Algorithmen ECA und WCL möglich ist. Ebenfalls wird deutlich, dass die verschiedenen Datenfilter bei annehmbarer Positionsgenauigkeit kaum eine Verbesserung bringen. Im Gegensatz dazu bietet die Wahl eines guten RSSI Grenzwertes eine klare Verbesserung der Genauigkeit.

Aufgrund des fehlenden Hard- bzw. Softwaresupports für den neuen Bluetooth 5.1 Standard AoA ist ein Testen dieser Methode nicht möglich. Obwohl Signalreflexionen und -auslöschungen auch bei AoA vorkommen können, ist großes Potenzial für die Verbesserung der Position durch minimale Erweiterung der derzeit bestehenden Hardwareausstattung der Standorte erkennbar.

Die letzten beiden Verbesserungsmethoden, der Machine Learning Algorithmus KNN und die Verwendung zusätzlicher Informationsquellen, konnten nicht getestet werden, da das Ortungsframework zum Zeitpunkt dieser Arbeit noch nicht über ein automatisiertes Updatesystem aller Komponenten verfügt. Durch die Abhängigkeit der Komponenten von Services darüberliegender Komponenten sind Updates aller Komponenten jedoch ein notwendiger Schritt, um Firmwareänderungen testen zu können.

6.2 Ausblick

Zur weiteren Verbesserung der Positionsgenauigkeit sowie zum Testen der neu entwickelten Methoden werden noch mehrere Schritte benötigt. Der erste Schritt ist eine Verbesserung der Kalibration der Ortungsereignisse und Trigger mit dem Ziel alle Grobbereiche korrekt zu erkennen und dadurch die Qualität der gesammelten Daten weiter zu verbessern.

Als Nächstes wird eine vollständige Implementierung von automatischen Systemupdates, welche dann über den BLE Manager gesteuert werden, benötigt. Die größte Herausforderung hierbei ist das automatische Provisioning und Konfigurieren der Bluetooth Meshes, das in Zukunft nicht mehr über den embedded Provisioner, sondern über eine Python Anwendung auf den Raspberry Pis erfolgen soll. Hierfür wird eine Library verwendet, welche derzeit noch entwickelt wird. Sobald die Implementierung der automatischen Updates abgeschlossen ist, wird die Soft- bzw. Firmware aller Komponenten des Ortungsframeworks aktualisiert werden. Ab diesem Zeitpunkt können sowohl Extended Advertising als auch das MVB getestet werden. Parallel dazu können zusätzlich zur Tiefgarage noch weitere Standorte inklusive Scannerpositionen und Stellplätze vermessen werden.

Weiters kann der Trilateration Manager an das Produktivsystem zur automatischen Standortberechnung angebunden werden. Es wird vorgeschlagen die Auslastung der Fahrzeughallen, z. B. zu Stoßzeiten, auf Basis der Anzahl der ausgelösten Trigger zu verschiedenen Zeitpunkten zu messen und dessen Auswirkung auf die Positionsgenauigkeit zu untersuchen. Daraus kann dann eine Anpassung der Trilaterationsparameter abhängig von der Standortauslastung erfolgen.

Nach einer ausreichend langen Trainingsphase des KNN Managers, bei der die empfohlene Mindestanzahl von 300 Datensätzen erreicht wurde, kann mit den ersten Positionsberechnungen begonnen werden. Sowohl für die Trilateration als auch für KNN kann untersucht werden, welche Auswirkungen unterschiedliche Datensammelzeiträume auf die Positionsgenauigkeit haben.

Sobald Hard- und Softwaresupport für AoA verfügbar sind, kann mit der Umsetzung der Methode begonnen werden. Der erste Einsatzort wird der Vorhof sein, welcher durch seine große Standfläche im Freien gekennzeichnet ist.

Wurden nun alle diese Methoden umgesetzt und einzeln getestet kann die optimale Kombination entwickelt werden.

Obwohl schon zu Beginn dieser Arbeit ein großer Teil des Ortungsframeworks entwickelt war und dieses auch im Zuge der Arbeit um viele neue Funktionen erweitert worden ist, gibt es immer noch zahlreiche Verbesserungen, die es noch umzusetzen gilt. Ist die Positionsbestimmung abgeschlossen, ermöglicht die Bluetooth Low Energie Technologie sogar Erweiterungen in Bereiche außerhalb der Standortbestimmung.

Literaturverzeichnis

[blua] blukii Location-Based Services. `https://www.blukii.com/`. Accessed: 2020-01-02.

[Blub] Bluetooth SIG. *Enhancing Bluetooth Location Services with Direction Finding.*

[Blu19a] Bluetooth SIG. *Bluetooth Core Specification v5.1*, 2019.

[Blu19b] Bluetooth SIG. *Bluetooth Direction Finding - A Technical Overview*, 2019.

[CLZW18] L. Cheng, Y. Li, M. Zhang, and C. Wang. A Fingerprint Localization Method Based on Weighted KNN Algorithm. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, pages 1271–1275, Oct 2018.

[DVG+17] I. Draghici, A. Vasilateanu, N. Goga, B. Pavaloiu, L. Guta, M. N. Mihailescu, and C. Boiangiu. Indoor Positioning System for Location based Healthcare using Trilateration with Corrections. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 169–172, June 2017.

[eli] eliko KIO RTLS - A UWB-based Indoor Positioning System. `https://www.eliko.ee/products/kio-rtls/`. Accessed: 2020-01-02.

[Fri15] Christoph Friedrich. Positionsbestimmung von Fahrzeugen in Tiefgaragen - Entwurf und Implementierung basierend auf IEEE 802.11 Wireless LAN. Master's thesis, Technische Universität Graz, 2015.

[inf] infsoft Smart Connected Locations. `https://www.infsoft.com/`. Accessed: 2020-01-02.

[int] INTRANAV Tracking Industrial Assets in Motion. `https://intranav.com/`. Accessed: 2020-01-02.

[KAP06] R. Kumar K., V. Apte, and Y. A. Powar. Improving the Accuracy of Wireless LAN based Location Determination Systems using Kalman Filter and Multiple Observers. In *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, volume 1, pages 463–468, April 2006.

[lea] leantegra RTLS. `https://leantegra.com/rtls/`. Accessed: 2020-01-02.

[LK16] Matej Liskovec and Alena Kovarova. Beacon Based Localization Refined by Outputs from Mobile Sensors. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, CompSysTech '16, pages 277–284, New York, NY, USA, 2016. ACM.

[LLH17] Z. Liu, X. Luo, and T. He. Indoor Positioning System Based on the Improved W-KNN Algorithm. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 1355–1359, March 2017.

[LQ13] H. Li and H. Qian. A Incremental Localization Based on Feasible Weighted Least Squares. In *2013 3rd International Conference on Consumer Electronics, Communications and Networks*, pages 117–120, Nov 2013.

[MJH99] William S. Murphy, Jr., and Willy Hereman. Determination Of A Position In Three Dimensions Using Trilateration and Approximate Distances. Technical report, 1999.

[MSA10] M. Moradi Zaniani, A. M. Shahar, and I. Abdul Azid. Trilateration Target Estimation Improvement using New Error Correction Algorithm. In *2010 18th Iranian Conference on Electrical Engineering*, pages 489–494, May 2010.

[MTP18] Ayush Mittal, Saideep Tiku, and Sudeep Pasricha. Adapting Convolutional Neural Networks for Indoor Localization with Smart Mobile Devices. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, GLSVLSI 18, page 117122, New York, NY, USA, 2018. Association for Computing Machinery.

[NJNR17] Q. H. Nguyen, P. Johnson, T. T. Nguyen, and M. Randles. Optimized Indoor Positioning for static mode smart devices using BLE. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1–6, Oct 2017.

[nod] node-pm. `https://github.com/sazze/node-pm`. Accessed: 2020-05-15.

[sci] Scikit Learn KNeighborsRegressor. `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html`. Accessed: 2020-02-12.

[sew] sewio Real Time Location System. `https://www.sewio.net/rtls-in-logistics/`. Accessed: 2020-01-02.

[sil] Silicon Labs Bluetooth. `https://www.silabs.com/wireless/bluetooth`. Accessed: 2020-01-02.

[SKS+16] S. Subedi, G. Kwon, Seokjoo Shin, Suk-seung Hwang, and Jae-Young Pyun. Beacon Based Indoor Positioning System Using Weighted Centroid Localization Approach. In *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 1016–1019, July 2016.

[SS18] A. Sunardy and N. Surantha. Performance Evaluation of Indoor Positioning Algorithm using Bluetooth Low Energy. In *2018 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 503–507, Oct 2018.

[Wil15] Roman Wilfinger. Absolute positioning of vehicles in indoor environments using Wireless LAN and Bluetooth LE. Master's thesis, Technische Universität Graz, 2015.