Verbesserung eines Bluetooth-basierten Warnsystems für den Straßenverkehr durch Datenlogging und innovative Visualisierungstechniken

Studienarbeit T3200

Studiengang Elektrotechnik

Studienrichtung Fahrzeugelektronik

Duale Hochschule Baden-Württemberg Ravensburg, Campus Friedrichshafen

von

Luka Tadic

Abgabedatum:	14.07.2025
Bearbeitungszeitraum:	07.04.2025 - 14.07.2025
Matrikelnummer:	5726700
Kurs:	TFE22-1
Dualer Partner:	
Betreuerin / Betreuer:	Prof. Dr. Ing. Tobias Frank
Gutachterin / Gutachter:	Prof. Dr. Ing. Tobias Frank

Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit T3200 mit dem Thema:

> *Verbesserung eines Bluetooth-basierten Warnsystems für den Straßenverkehr durch Datenlogging und innovative Visualisierungstechniken*

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Friedrichshafen, den 13. Juli 2025

Luka Tadic

Kurzfassung

Diese Studienarbeit verbessert ein Bluetooth-basierendes Warnsystem, das Abbiegeunfälle zwischen Lastkraftwagen (LKW) und ungeschützten Verkehrsteilnehmenden vermeiden soll. Zentrale Beiträge sind

Datenlogger: Ein Python-Modul zeichnet alle AoA-Messdaten im zeilenbasierten JavaScript Object Notation Lines (JSONL)-Format auf und erlaubt deren hardwareunabhängiges Replay. So können Entwickler Algorithmen offline testen und parametrisieren.

Visualisierung: Die bisher punktförmige Darstellung des Tags wurde durch eine dynamische Stickman-Figur ersetzt; ein LKW-Bild, farbige Warnstufen sowie ein optionales Koordinatensystem erhöhen die Anschaulichkeit. Sämtliche Daten werden per TCP in quasi-Echtzeit in die Oberfläche gestreamt.

Evaluation: Funktional-, Integrations- und Systemtests auf Windows 11 bestätigten eine stabile Datenaufzeichnung, fehlerfreie Transmission Control Protocol (TCP)-Kommunikation und eine insgesamt benutzerfreundliche UI. Schwächen zeigten sich bei ruckartiger Tag-Bewegung und reduzierter Ortungsgenauigkeit in Randlagen.

Ausblick: Vorgeschlagen werden adaptive Glättungsverfahren (z. B. Kalman-Filter), ein drittes Antennen-Board zur Genauigkeitssteigerung, interaktive 3D-Darstellungen (Unity/WebGL) sowie eine stärkere Modularisierung der Software.

Das Ergebnis ist ein robuster Prototyp, der Entwicklungs- und Testprozesse beschleunigt und eine fundierte Basis für weiterführende Fahrerassistenzsysteme bildet.

Abstract

This study enhances a Bluetooth AoA based warning system designed to prevent turning collisions between trucks and vulnerable road users. Key contributions are

Data Logger – A Python module records all Angle-of-Arrival (AoA) measurements in a line-wise JSONL format and provides hardware-independent replay, enabling offline algorithm tuning and analysis.

Visualization – The original single-point display was replaced by a dynamic "stickman", complemented by a truck image, colour-coded risk levels and a toggleable coordinate grid. Live or replayed data are streamed to the UI via TCP in near real time.

Evaluation – Functional, integration and system tests on Windows 11 proved reliable data recording, loss-free TCP communication and an overall user-friendly interface. Limitations were observed in slightly jerky tag motion and decreased positioning accuracy at the edges of the coverage area.

Future Work – Proposed improvements include adaptive smoothing (e.g., Kalman filtering), adding a third antenna board for higher accuracy, interactive 3-D visualizations (Unity/WebGL) and deeper modularisation of the software stack.

The resulting prototype shortens development cycles, supports thorough testing and lays a solid foundation for advanced collision-avoidance driver-assistance systems.

Hilfsmittel

Für die Erstellung dieser Arbeit wurden folgende Werkzeuge genutzt:

- **LLMs (ChatGPT):** Ideen- und Code-Gerüst, finale Algorithmen und Texte wurden manuell überprüft und angepasst.
- **Entwicklungsumgebung:** Visual Studio Code + Python 3.12 (tkinter, pandas, socket) sowie g++ für bestehende C++-Module.
- **Versionsverwaltung:** Git (GitHub-Repository).
- **Hardware uns Softwaretests:** u-blox XPLR-AOA-2-Kit und lokales TCP-Loopback unter Windows 11.

Abbildungsverzeichnis

Abkürzungsverzeichnis

AoA Angle-of-Arrival

API Application Programming Interface

BLE Bluetooth Low Energy

LKW Lastkraftwagen

SDK Software Development Kit

CSV Comma-seperated values

JSON JavaScript Object Notation

UI User Interface

JSONL JavaScript Object Notation Lines

BAT batch

TCP Transmission Control Protocol

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Zahl der tödlichen Verkehrsunfälle ist in den vergangenen Jahren erneut gestiegen. Ursachen sind unter anderem eine wachsende Verkehrsdichte sowie nachlassende Aufmerksamkeit und Sorgfalt vieler Verkehrsteilnehmender. Um dieser Entwicklung entgegenzuwirken, wurden sowohl gesetzliche Vorgaben verschärft als auch technische Innovationen vorangetrieben. Zu Letzteren zählen Fahrzeugkameras, Sensorik und moderne Fahrerassistenzsysteme, die sich als zentrale Instrumente der Unfallprävention erwiesen haben.

Abbildung 1: Rechtsabbiegesituation mit Abbiegeassistent [1]

Eine besonders wichtige Neuerung im Bereich der Lastkraftwagen (LKW)-Sicherheit sind Abbiegeassistenten. Sie reduzieren den toten Winkel und tragen so wesentlich dazu bei, Unfälle – insbesondere beim Rechtsabbiegen – zu vermeiden. Obwohl erste Systeme bereits im Einsatz sind, besteht weiterhin erhebliches Potenzial für Verbesserungen. Vertiefte Forschung und Entwicklung im Bereich der Fahrerassistenz kann das allgemeine Verkehrsrisiko weiter senken und die Verkehrssicherheit signifikant erhöhen [2].

1.2 Zielsetzung

Diese Studienarbeit verfolgt zwei Hauptziele: Erstens soll ein leistungsfähiger Datenlogger entwickelt werden, der Messwerte des Fahrerassistenzsystems systematisch und zuverlässig erfasst. Dadurch können mehrere Teammitglieder parallel mit authentischen Messdaten arbeiten, ohne dauerhaft auf die Hardware zugreifen zu müssen. Zweitens wird die bestehende Visualisierung überarbeitet, um Messergebnisse klarer und übersichtlicher darzustellen. Die Kombination aus verbesserter Datenerfassung und optimierter Darstellung erleichtert zukünftige Analysen und Tests, erhöht die Qualität der Positionsbestimmung und steigert die Effizienz des gesamten Entwicklungsprozesses – letztlich mit dem Ziel, die Verkehrssicherheit weiter zu verbessern.

2 Rückblick auf das mobile Warnsystem

2.1 Konzept und Zielsetzung

Die vorangegangene Studienarbeit hatte das Ziel, ein mobiles Warnsystem zu entwickeln, das Abbiegeunfälle zwischen LKW und ungeschützten Verkehrsteilnehmenden – insbesondere Fußgänger*innen und Radfahrer*innen – minimiert. Kern des Ansatzes war eine Smartphone-Applikation, die als aktiver Bluetooth-Sender (Tag) fungiert. In Verbindung mit einem am LKW montierten Empfängerboard (u-blox XPLR-AOA-1) sollte mithilfe der Angle-of-Arrival (AoA)-Technologie die Position des Smartphones bestimmt und bei kritischen Situationen eine Warnung ausgegeben werden. Durch den Verzicht auf spezielle Hardware für die Verkehrsteilnehmenden versprach das System einen kostengünstigen und leicht zugänglichen Beitrag zur Verkehrssicherheit im urbanen Raum.

2.2 Bluetooth-AoA – technische Grundlagen

AoA ist Bestandteil des Bluetooth-5.1-Standards. Mehrere Antennen auf dem Empfängerboard messen den Einfallswinkel des Funksignals; aus mindestens zwei Winkeln lässt sich per Triangulation die Position des Senders bestimmen [3], [4]. Voraussetzungen sind:

- Zugriff auf die Sendeparameter des Bluetooth-Tags (z. B. Werbekanäle und Signalform),
- eine Antennenkonfiguration mit bekannten geometrischen Abständen,
- zeitlich hochaufgelöste Phasen- bzw. I/Q-Messungen im Empfänger.

Das u-blox XPLR-AOA-1 Explorer Kit erfüllt diese Bedingungen, da es sowohl ein AoA-fähiges Empfängerboard als auch einen vorkonfigurierten Tag mitbringt. In der vorangegangenen Arbeit sollte das Smartphone diese Tag-Funktion übernehmen.

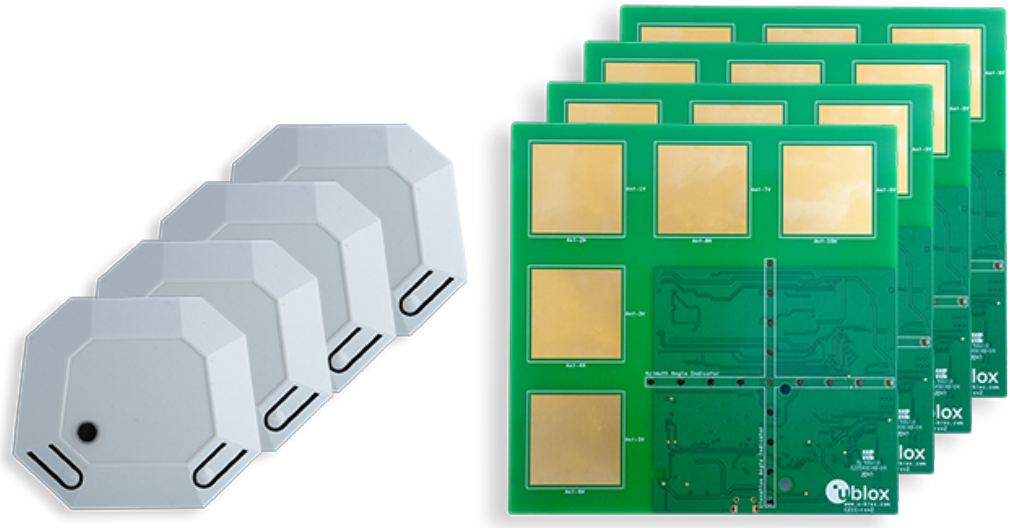

Abbildung 2: u-blox XPLR-AOA-2 Explorer Kit [5]

2.3 Herausforderungen und Gründe für die Projektpause

Im praktischen Versuch zeigten sich mehrere technische Hürden, die eine Fortführung des Projekts verhinderten:

- **Eingeschränkter Bluetooth-Stack auf Smartphones:** Aktuelle Android-Firmware erlaubt keinen vollständigen Zugriff auf Low-Level-Sendeparameter. Ein gezieltes Aussenden der für AoA nötigen Bluetooth Low Energy (BLE)-Werbepakete ist mit dem Standard-Software Development Kit (SDK) nicht möglich [6].
- **Begrenzte Verfügbarkeit von Bluetooth 5.1:** Nur wenige Endgeräte unterstützen bislang den vollständigen 5.1-Funktionsumfang. Dies reduziert Reichweite und Genauigkeit der Ortung.

- **Fehlende Systemrechte:** Die Emulation des u-blox-Tags scheiterte an fehlendem Root-Zugriff sowie an nicht freigegebenen Low-Level-Application Programming Interface (API)s.

Aufgrund dieser Limitierungen konnte das Projekt nicht wie geplant abgeschlossen werden. Die zugrunde liegende Idee bleibt jedoch vielversprechend und kann erneut aufgegriffen werden, sobald die Geräteunterstützung für Bluetooth 5.1 breiter verfügbar ist und Betriebssysteme einen tieferen Zugriff auf den Bluetooth-Stack erlauben.

3 Grundlagen Datenlogger

3.1 Zweck und Funktionalität eines Datenloggers

Ein Datenlogger ist ein autonom arbeitendes Gerät bzw. Software-Modul, das physikalische oder digitale Messwerte über einen definierten Zeitraum aufzeichnet. Ziel ist es, Veränderungen systematisch und zuverlässig zu erfassen, um sie später analysieren, auswerten oder dokumentieren zu können [7], [8]. Typische Messgrößen sind beispielsweise Temperatur, Spannung, Zeitstempel sowie Lage- oder Positionsdaten.

Abbildung 3: Datenlogger Cube (Avisaro 2.0) zur Speicherung von Sensor- und Prozessdaten [9]

Im Unterschied zu interaktiven Messsystemen arbeitet ein Datenlogger in der Regel ohne permanente Verbindung zu einem Steuergerät. Damit eignet er sich besonders für mobile oder schwer zugängliche Anwendungen. Die kontinuierliche Aufzeichnung ermöglicht eine lückenlose Nachverfolgbarkeit und ist vor allem während Entwicklung, Fehlersuche und Validierung technischer Systeme von großer Bedeutung [10].

Wesentliche Eigenschaften eines Datenloggers sind eine ausreichende Speicherkapazität, hohe Energieeffizienz sowie die Kompatibilität zu standardisierten Datenformaten wie Comma-seperated values (CSV) oder JavaScript Object Notation (JSON). Moderne Geräte bieten darüber hinaus Schnittstellen für drahtlose Datenübertragung oder eine automatisierte Synchronisation

mit Analysetools [11].

3.2 Anwendungsbereiche im Kontext dieser Studienarbeit

In dieser Arbeit wird der Datenlogger eingesetzt, um die Entwicklung des Bluetooth-AoA-Lokalisierungssystems effizienter und flexibler zu gestalten. Während Test- und Optimierungsphasen soll die Abhängigkeit von einer Live-Verbindung zur Hardware minimiert werden [7]. Entwickelnde können somit reale Messdaten analysieren, ohne direkt auf die u-blox-Hardware zugreifen zu müssen.

Der Logger speichert originalgetreue Messwerte, die anschließend in externe Werkzeuge importiert werden können, um Berechnungen, Analysen oder grafische Darstellungen vorzunehmen. Parameter- oder Algorithmusänderungen lassen sich so anhand aufgezeichneter Daten überprüfen, ohne dass dafür eine erneute Datenerfassung erforderlich ist [8].

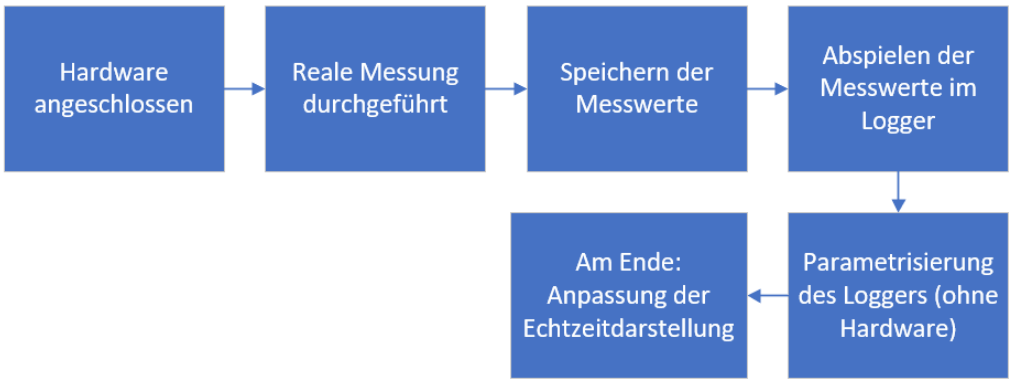

Abbildung 4: Einbindung des Datenloggers in den Entwicklungsprozess
Quelle: Eigene Darstellung

Darüber hinaus ermöglicht der Datenlogger eine *Offline-Simulation* beziehungsweise ein Daten-Replay mit realen Werten. Optimierungen können zunächst im Testsystem bewertet und erst bei zufriedenstellendem Ergebnis auf das reale System übertragen werden. Der Logger trägt somit wesentlich zur Entkopplung von Entwicklung und Hardwareverfügbarkeit bei.

3.3 Herausforderungen und Lösungsansätze

Bei der Entwicklung eines zuverlässigen Datenloggers für das Bluetooth-AoA-System treten mehrere Herausforderungen auf:

- **Zeitliche Synchronisation:** Für eine konsistente Auswertung müssen Zeitstempel, Empfangswinkel und weitere Messgrößen exakt zueinander passen [11].
- **Speichereffizienz:** Große Datenmengen sollen verlustfrei, aber kompakt abgelegt werden. Ein strukturiertes Format wie CSV oder JSON erleichtert die spätere Analyse in Python oder MATLAB [10].
- **Integration in bestehende Prozesse:** Der Logger muss modular sein und sich ohne großen Aufwand in unterschiedliche Testumgebungen einfügen lassen. Ein niedriger Energieverbrauch ist dabei essenziell, um auch längere mobile Einsätze zu unterstützen [8].

Als Lösungsansätze dienen

- die Verwendung eines einfachen, standardisierten Speicherformats,
- ein modular aufgebautes Software-Design mit klar dokumentierten Schnittstellen,
- sowie grundlegende Mechanismen zur Fehlererkennung und Datensynchronisation.

Diese Maßnahmen gewährleisten, dass der Datenlogger zuverlässig, energieeffizient und benutzerfreundlich eingesetzt werden kann [7].

4 Entwicklung des Datenloggers

4.1 Anforderungen an die Software

Die zu entwickelnde Datenlogger-Software muss sich nahtlos in die bestehende Systemarchitektur integrieren lassen und dabei drei Kernanforderungen erfüllen:

- **Robuste Datenspeicherung** aller relevanten Messwerte,
- **flexible Wiederverwendung** der gespeicherten Daten für Tests und Visualisierungen,
- **einfache Wart- und Erweiterbarkeit** des Quellcodes.

Ein zentrales Ziel ist die *Entkopplung* der Datenerfassung von der physischen Hardware. Durch die Offline-Nutzung gespeicherter Messreihen können Entwicklerinnen und Entwickler auch ohne Zugang zur u-blox-Hardware Analysen durchführen und Visualisierungen testen.

Darüber hinaus wurde eine klar strukturierte, maschinenlesbare Datenhaltung gefordert, damit die aufgezeichneten Werte in gängigen Werkzeugen (z. B. Python, MATLAB) unmittelbar weiterverarbeitet werden können. Mehrbenutzerfähigkeit und Plattformunabhängigkeit gelten als wünschenswerte Eigenschaften.

4.2 Auswahl geeigneter Technologien und Tools

Die vorhandene Kommunikation mit der u-blox-Hardware basiert auf einer performanten C++-Schicht. Um diese beizubehalten und dennoch agile Erweiterungen zu ermöglichen, wurde eine zweistufige Architektur gewählt:

- **C++**: bestehende, zeitkritische Kommunikation mit den Empfängerboards,
- **Python**: neuer Datenlogger, Replay-Funktion und Visualisierung.

Diese Kombination vereint die Rechenleistung von C++ mit der Entwicklungsproduktivität von Python [12].

Als Speicherformat kommt JSON zum Einsatz, genauer das JavaScript Object Notation Lines (JSONL)-Format. Es gestattet, komplexe Datensätze – Zeitstempel, AoA-Winkel, Signalstärken und Metadaten – zeilenweise abzulegen, maschinenlesbar zu versionieren und später sequentiell einzulesen [13].

Die in Abschnitt 4.1 genannten Anforderungen werden damit erfüllt: Die Lösung ist plattformunabhängig, modular und teamfreundlich.

4.3 Implementierung und Integration in das bestehende System

Das Lokalisierungssystem basiert auf der Bluetooth-AoA-Technologie. Der Datenfluss wurde so gestaltet, dass Messung, Speicherung und Analyse wahlweise live oder offline erfolgen können.

Messung – Entstehung der Rohdaten

- **Tag**: sendet kontinuierlich Bluetooth-Signale.
- **Anker** (mind. zwei): empfangen die Signale mit Mehrantennen-Arrays und bestimmen die Einfallswinkel (AoA).

Datenerfassung und -aufbereitung Ein Python-Skript (`getAnchorData.py` [14]) liest die Rohdaten über eine serielle Schnittstelle bzw. BLE aus. Pro Messzyklus werden unter anderem

- `anchors` (Koordinaten der Anker),
- `angles` (gemessene Winkel) sowie
- `sensor_values` (weitere Metadaten)

in eine `.jsonl`-Datei geschrieben. Jede Zeile enthält einen ISO-Zeitstempel und das zugehörige Messobjekt.

Berechnung der Tag-Position – Triangulation Aus zwei Winkelmessungen und den bekannten Ankerpositionen lässt sich die Tag-Position über Triangulation bestimmen. Für jeden Anker i gilt mit Winkel θ_i der Richtungsvektor

$$\vec{d_i} = (\sin\theta_i,\ \cos\theta_i).$$

Schneidet man die durch (x_i, y_i) mit Richtung $\vec{d_i}$ definierten Geraden, erhält man

$$t_1 = \frac{(x_2 - x_1)\, d_{y,2} - (y_2 - y_1)\, d_{x,2}}{d_{x,1}\, d_{y,2} - d_{y,1}\, d_{x,2}}, \qquad \vec{p} = (x_1, y_1) + t_1\, \vec{d_1}.$$

Die Berechnung erfolgt wahlweise *live* oder später im Replay-Modus.

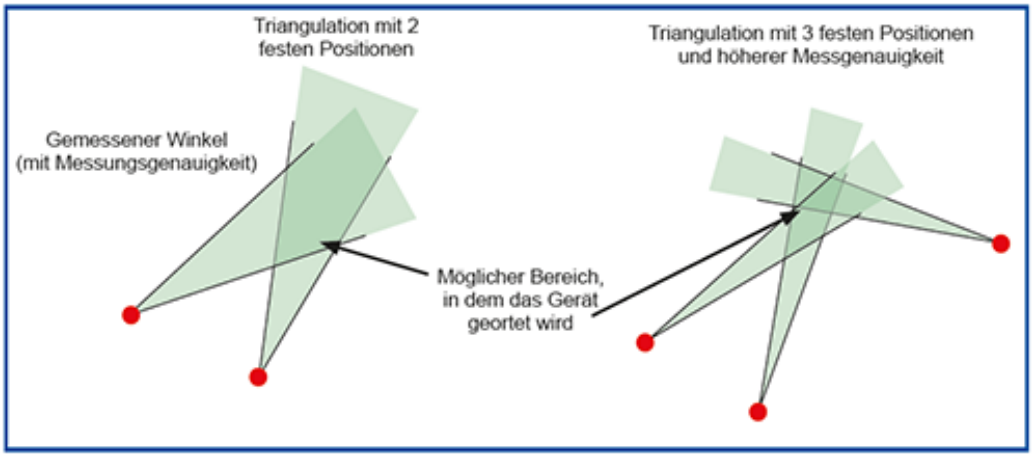

Abbildung 5: Beispielhafte Triangulation [15]

Speicherung und Wiederverwendung Alle berechneten Werte werden chronologisch gespeichert. Das Skript `replayLogger.py [14]` liest die Logdatei ein, berechnet bei Bedarf die Position neu und stellt die Daten über eine Transmission Control Protocol (TCP)-Schnittstelle der Visualisierung (`ui.py` [14]) bereit.

Abbildung 6: Gesamter Datenfluss vom Messvorgang bis zur Visualisierung
Quelle: Eigene Darstellung

Nachträgliche Optimierung Frühe Versionen spielten im Replay nur die gespeicherten Koordinaten ab. Das Modul wurde daher erweitert, sodass beim Einlesen stets aus den Rohwinkeln *neu* trianguliert wird. Somit lassen sich Algorithmen offline verifizieren, bevor sie in das Echtzeitsystem übernommen werden.

Visualisierung Die TCP-Schnittstelle sendet jede Messung als JSON-Objekt an die User Interface (UI). Dort werden Tag-Position, Anker, Bewegungsrichtung und Risikostufe live angezeigt.

Zusammenfassung des Datenflusses

1. **Messung** – Anker erfassen Winkel und Sensordaten.
2. **Logging** – Python-Skript speichert Messwerte Zeile für Zeile in `.jsonl`.
3. **Replay** – `replayLogger.py [14]` liest Daten, trianguliert erneut und streamt sie.
4. **Visualisierung** – `ui.py [14]` stellt Position und Zusatzinfos dar.

4.4 Ergebnisse und Evaluation

Der Datenlogger ließ sich mit wenigen Code-Ergänzungen in die bestehende Software integrieren. Die Lösung

- erfasst Messdaten zuverlässig,
- speichert sie zeilenweise im `.jsonl`-Format,
- ermöglicht ein hardwareunabhängiges Replay,
- und überträgt die Daten stabil per TCP an die UI.

Zur komfortablen Bedienung wurde eine `.bat`-Datei bereitgestellt, die den Logger startet und aufgezeichnete Datensätze zur Auswahl stellt. Mehrere Teammitglieder können so parallel mit denselben Messdaten arbeiten – unabhängig von der Hardwareverfügbarkeit.

Erste Tests mit kurzen bis mittellangen Aufzeichnungen (mehrere Minuten) verliefen problemlos. Für den Produktiveinsatz sind Langzeittests vorgesehen, um Speicherstabilität und Performance bei großen Datenmengen zu validieren.

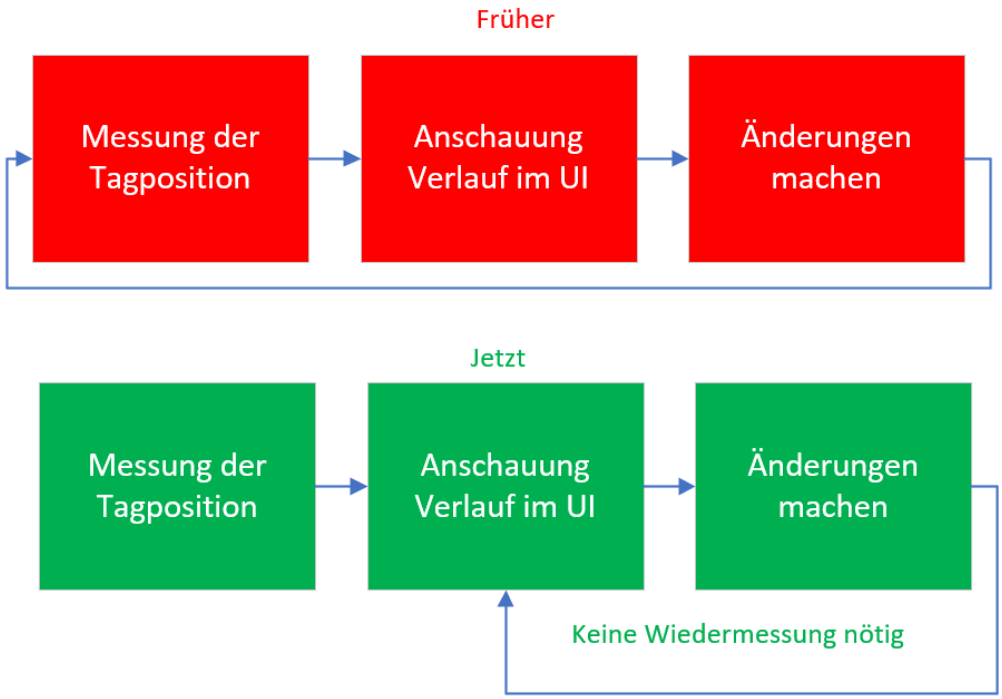

Abbildung 7: Parametrierung ohne (links) und mit (rechts) Datenlogger-Workflow
Quelle: Eigene Darstellung

Der entwickelte Logger bildet somit eine stabile Grundlage für künftige Entwicklungs- und Testarbeiten und trägt wesentlich zu effizienteren, hardwareunabhängigen Validierungsprozessen bei.

5 Grundlagen der Visualisierung

5.1 Anforderungen an eine effektive Visualisierung

Die visuelle Aufbereitung von Mess- und Sensordaten ist ein zentrales Element der Systementwicklung. In dieser Studienarbeit dient die Visualisierung nicht allein als Ausgabeinstrument, sondern als aktives *Analysewerkzeug*. Daraus ergeben sich spezifische Anforderungen an Struktur, Funktionalität und Performance:

Klarheit und Verständlichkeit
Tag-Position, Winkelmessungen sowie Zusatzinformationen (z. B. Risikostufe) müssen intuitiv erfassbar sein und dürfen keine unnötige Komplexität erzeugen.

Reaktionsfähigkeit
Insbesondere beim Replay ist eine quasiechtzeitfähige Darstellung erforderlich, damit zeitkritische Situationen nachvollziehbar bleiben.

Strukturierte, modulare Ebenen
Ankerpositionen, Trajektorien, Messwinkel und Risikoindikatoren werden auf separaten Layern visualisiert; eine Farb- oder Symbolcodierung erleichtert das Erkennen von Korrelationen.

Konfigurierbarkeit
Parameter und Datensätze müssen sich flexibel umschalten lassen, um unterschiedliche Szenarien vergleichen und analysieren zu können.

Stabilität & Performance
Auch bei großen Datenmengen darf die Oberfläche weder verzögert reagieren noch hohe Systemressourcen beanspruchen.

Schnittstellenkompatibilität
Die UI empfängt Daten kontinuierlich per TCP in JSON-Struktur; eingehende Pakete sind verlustfrei zu dekodieren und anzuzeigen.

Benutzerfreundliche Interaktion
Eine klare Bedienlogik ermöglicht auch neuen Teammitgliedern oder externen Testern einen schnellen Einstieg.

5.2 Darstellungsformen und Technologien

2D-Darstellung Ein kartesisches Koordinatensystem bildet die Basis: Anker werden als feste Punkte eingezeichnet, die aktuelle Tag-Position als Marker (ggf. farblich kodiert: Grün = sicher, Gelb = Warnung, Rot = Kollisionsgefahr). Sensorlinien oder Vektorpfeile verdeutlichen gemessene Winkel und Gefahrenzonen. Ein Replay-Modus zeigt frühere Positionen sequenziell an und gestattet so die Analyse von Bewegungsmustern.

Technische Umsetzung

- **Python / Tkinter** – leichtgewichtige UI, Canvas-Zeichenfläche für dynamische Objekte [16].
- **TCP** – kontinuierlicher Datenstrom vom Logger zur UI (Modul `ui.py`).
- **JSON-Parsing** – Dekodierung der Log- und Live-Pakete.

Die modulare Struktur erlaubt spätere Migration zu `PyQt`, `Plotly Dash` oder 3D-Frameworks (OpenGL, Unity), ohne die Backend-Logik anzutasten.

Schnittstellen & Erweiterbarkeit Neben Live-Daten unterstützt die Oberfläche einen reinen Replay-Betrieb. Ein Startskript (Batch oder CLI-Parameter) wählt die gewünschte Logdatei. Backend (Logger / Replay) und Frontend (UI) sind klar getrennt; neue Sensorquellen lassen sich per standardisiertem JSON-Schema einbinden.

Empfohlene Übersichtsgrafik (optional) Eine schematische Abbildung kann den vollständigen Visualisierungspfad veranschaulichen:

- Anker-Positionen als fixe Punkte
- Messwinkel als Richtungslinien
- Tag-Marker mit Zeitstempel
- Farbring für Risikostufe
- Bewegungspfad (gestrichelte Linie)

Solch eine Grafik erleichtert auch fachfremden Stakeholdern das Verständnis des Systems.

Fazit Die gewählten Technologien liefern eine performante, verständliche und erweiterbare Visualisierung. Sie bilden damit eine solide Grundlage für die weiterführende Analyse, Parametrierung und Validierung des Fahrerassistenzsystems.

6 Entwicklung der neuen Visualisierung

6.1 Analyse der bestehenden Visualisierung

Die bisherige grafische Benutzeroberfläche (UI) wurde in Python entwickelt und diente der Echtzeitdarstellung der aus dem Bluetooth-AoA-System ermittelten Tag-Position. Als zentrales Element wurde ein zweidimensionales kartesisches Koordinatensystem verwendet, innerhalb dessen ein einzelner Punkt die Bewegung des Tags repräsentierte. Die Position dieses Punktes wurde entweder durch eine direkte Live-Messung mit der Hardware oder durch eine vereinfachte Simulation erzeugt – eine Möglichkeit zur Wiederverwendung gespeicherter Daten durch einen Datenlogger war zu diesem Zeitpunkt noch nicht implementiert [16].

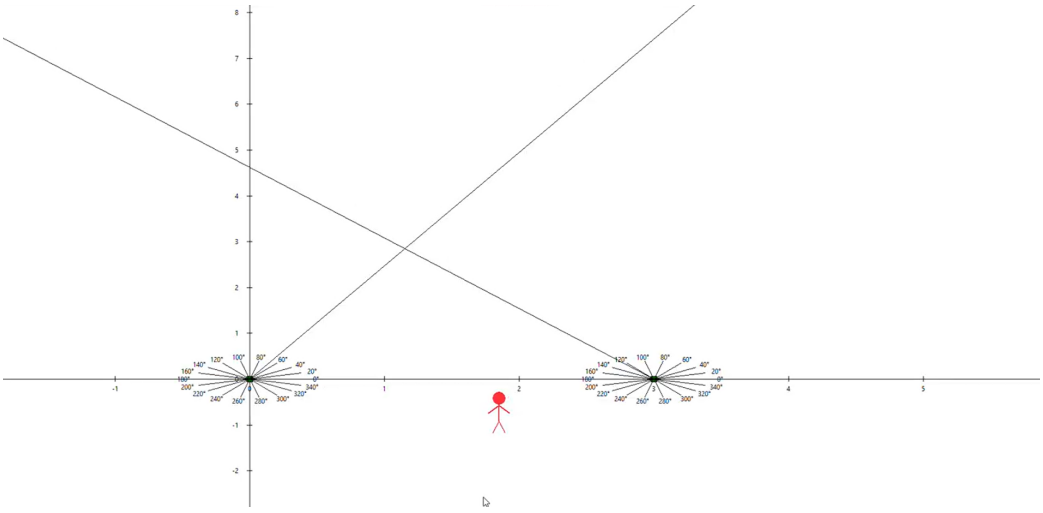

Abbildung 8: Alte Visualisierung [14]

Zur besseren Nachvollziehbarkeit der geometrischen Zusammenhänge wurden auch die fest installierten Empfangseinheiten (Anker) im Koordinatensystem visualisiert. Von deren Positionen ausgehend wurden Linien gezeichnet, welche die gemessenen Einfallswinkel (AoA) darstellten. Die Idee dahinter war, die Triangulation – also die Schnittpunktberechnung zweier Richtungslinien – visuell darzustellen und die damit berechnete Tag-Position zu validieren.

Allerdings wies diese Darstellung mehrere funktionale und darstellerische Einschränkungen auf. Eine grundlegende Schwäche bestand darin, dass der dargestellte Schnittpunkt der Winkel-Linien oft nicht exakt mit dem ange-

zeigten Punkt des Tags übereinstimmte. Dies führte zu Inkonsistenzen, die insbesondere bei der Bewertung der Lokalisierungsgenauigkeit zu Verwirrung führen konnten.

Darüber hinaus fehlten wichtige Interaktionsmöglichkeiten und Darstellungsmodi: Sensorwerte oder Risikobewertungen konnten nicht separat angezeigt oder analysiert werden. Eine Zeitschrittsteuerung oder ein Replay-Modus waren nicht vorhanden. Da der Datenlogger zum damaligen Zeitpunkt noch nicht existierte, beschränkte sich die Funktionalität der Oberfläche auf Live-Datenströme oder manuell eingespeiste Simulationsdaten.

Zusammenfassend war die ursprüngliche Visualisierung funktional auf das Wesentliche reduziert, ermöglichte aber weder eine tiefere Analyse noch eine Nachbearbeitung historischer Messdaten. Die Einführung des Datenloggers und eine verbesserte Visualisierung sollen diese Defizite beheben und eine fundierte, nachvollziehbare sowie interaktive Darstellung aller relevanten Systemparameter ermöglichen.

6.2 Konzept einer verbesserten Visualisierung

Basierend auf der Analyse der bisherigen Darstellung ergeben sich mehrere Ansatzpunkte für eine funktional und ästhetisch verbesserte Visualisierung der Tag-Ortung. Ziel ist es, die Ausgabe so zu gestalten, dass auch technisch weniger versierte Personen – beispielsweise Besucher auf Messen oder Mitglieder in interdisziplinären Projektteams – intuitiv die Funktionsweise und Aussagekraft des Fahrerassistenzsystems nachvollziehen können.

Ein zentrales Element der neuen Visualisierung ist die verbesserte Darstellung des Tags. Bisher wurde dieser lediglich als bewegender Punkt im Koordinatensystem dargestellt, was der realen Anwendung – z. B. einem Fußgänger oder Fahrradfahrer im Straßenverkehr – kaum gerecht wurde. Zukünftig soll der Tag durch ein erkennbares Modell wie etwa eine stilisierte menschliche Figur (z. B. ein Stickman) ersetzt werden. Zusätzlich ist vorgesehen, visuelle Elemente wie ein LKW-Modell in die Darstellung zu integrieren, um die reale Gefahrenlage bei Abbiegesituationen plausibler zu simulieren.

Zur Verbesserung der Verständlichkeit soll die Bewegung des Tags flüssiger und zeitlich realistischer erfolgen. In der bisherigen Version war die Bewegung teils sprunghaft oder verzögert, was eine genaue Nachverfolgung erschwerte. Durch optimierte Synchronisation mit den übermittelten Messwerten soll die visuelle Bewegung stärker der tatsächlichen Bewegung des realen Objekts entsprechen.

Darüber hinaus ist geplant, ein interaktives Interface mit zusätzlichen Informationen bereitzustellen. So sollen Parameter wie Winkel, Positionskoordinaten oder Risikoindikatoren dynamisch eingeblendet und aktualisiert

werden können – ein wesentliches Werkzeug für Entwickler bei der Fehleranalyse und Parametrierung.

Ein weiterer Aspekt der verbesserten Darstellung betrifft das bisher fest eingeblendete Koordinatensystem. Dieses war zwar funktional nützlich, beeinträchtigte jedoch stellenweise die Lesbarkeit und ästhetische Wirkung der grafischen Oberfläche – insbesondere bei öffentlichkeitswirksamen Präsentationen oder auf begrenzten Anzeigeflächen. Künftig soll daher eine Option implementiert werden, das Koordinatensystem bei Bedarf auszublenden oder gezielt ein- und auszublenden (Toggle-Funktion). Dies erlaubt eine flexiblere Darstellung und steigert die Übersichtlichkeit – insbesondere wenn die rein visuelle Wahrnehmung im Vordergrund stehen soll.

Insgesamt verfolgt das neue Visualisierungskonzept das Ziel, eine Brücke zwischen technischer Präzision und intuitiver Darstellbarkeit zu schlagen. Es soll nicht nur die Qualität der Entwicklung verbessern, sondern auch die Kommunikation der Systemfunktionalität nach außen erleichtern – etwa in Präsentationen, bei Tests oder in zukünftigen Messen.

6.3 Implementierung der verbesserten Visualisierung

Zur besseren Nachvollziehbarkeit der gemessenen Bewegungsdaten und zur verständlichen Darstellung des Systems für technische wie nicht-technische Betrachter wurde die Benutzeroberfläche (UI) umfassend überarbeitet. Die Implementierung erfolgte in Python unter Verwendung des `tkinter`-Frameworks, wodurch eine flexible und plattformunabhängige grafische Visualisierung ermöglicht wurde [16].

Grundstruktur und Aufbau Die Benutzeroberfläche besteht aus einem zentralen Zeichenbereich (Canvas), auf dem die Bewegung des zu lokalisierenden Objekts (Tag) visualisiert wird. Ergänzt wird dieser durch eine Steuerleiste am unteren Rand zur Interaktion (z. B. Start/Stop der Visualisierung oder Ein-/Ausblenden von Antennendaten) sowie einer Informationsleiste zur Anzeige der aktuellen Position und Zusatzdaten.

Die Visualisierung ist in mehreren Schichten aufgebaut:

- Darstellung des Tags als dynamisch skalierbare Figur (z. B. Stickman oder Fahrzeugabbildung)
- Hintergrundkoordinatensystem mit optional einblendbaren Achsen und Skalen
- Einblendung der Position und Orientierung der Antennenarrays sowie der zugehörigen Messrichtungen (AoA-Linien)

Verarbeitung eingehender Daten Die Visualisierung kommuniziert über eine TCP-Verbindung mit dem Datenlogger, der kontinuierlich strukturierte JSON-Datenpakete sendet. Diese enthalten Informationen zur geschätzten Position des Tags, Sensordaten sowie Metainformationen. Eine dedizierte Serverkomponente in der UI nimmt die Daten entgegen, extrahiert relevante Parameter und aktualisiert die grafische Darstellung entsprechend.

Zur Verbesserung der Darstellung wurde ein Glättungsverfahren mit exponentiellem Filter implementiert, um Positionssprünge durch Rauschen oder Ausreißer zu reduzieren. Gleichzeitig werden unrealistische Sprünge oder abrupte Änderungen durch Grenzwerte abgefangen.

Darstellungskonzept des Tags Anstelle eines einfachen Punkts wird das Tag nun als abstrahierte Figur (Stickman) dargestellt. Diese passt sich dynamisch in Größe und Farbe an den Abstand zu den Antennen an, um Nähe, Risiko oder Interaktionszonen visuell hervorzuheben. Alternativ kann ein repräsentatives Fahrzeugmodell (ein LKW-Bild) eingebunden werden, um reale Szenarien wie Fahrassistenzsysteme intuitiver zu vermitteln.

Abbildung 9: Grün - keine Gefahr noch [14]

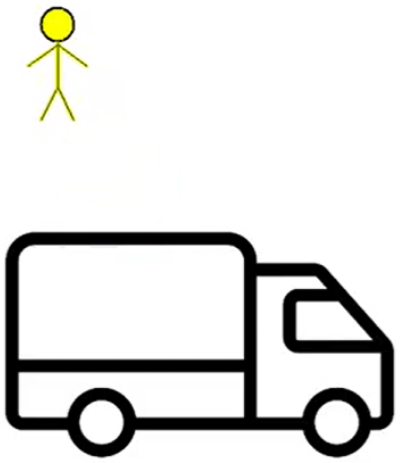

Abbildung 10: Gelb - Warnung, Gefahr möglich [14]

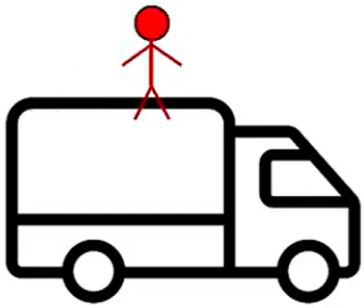

Abbildung 11: Rot - Achtung, Kollisionsgefahr [14]

Visualisierung der Antennen und Sensorlinien Die Position der Antennenarrays wird grafisch dargestellt und durch Linien ergänzt, welche die gemessenen Einfallswinkel (AoA) andeuten. Diese Linien zeigen die theoretische Messrichtung, was ein besseres Verständnis der Triangulation erlaubt.

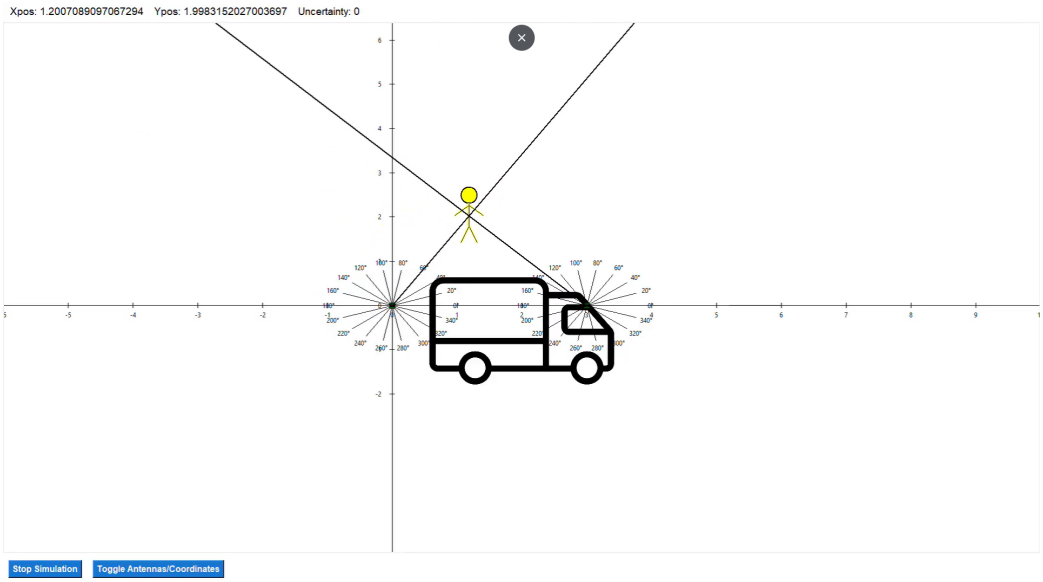

Abbildung 12: Visualisierung der Antennen und Sensorlinien [14]

Interaktive Steuerung und Optimierungen Ein zentrales Element ist die Möglichkeit, zwischen verschiedenen Visualisierungsmodi umzuschalten. Über ein Toggle-Button kann die Darstellung der Koordinatenachsen und Sensorlinien aktiviert oder ausgeblendet werden, um je nach Anwendungsfall zwischen technischer Analyse und Präsentation zu wechseln. Zusätzlich kann die Darstellung bei Bedarf in den Vollbildmodus gesetzt werden.

Zur Steuerung und flexiblen Nutzung wurde außerdem eine ausführbare Batch-Datei (`.bat`) bereitgestellt, mit der definierte Logdateien gezielt abgespielt und getestet werden können. Diese Struktur erlaubt es, mehrere gespeicherte Datensätze individuell zu benennen, abzurufen und ohne Hardwarebezug zu analysieren.

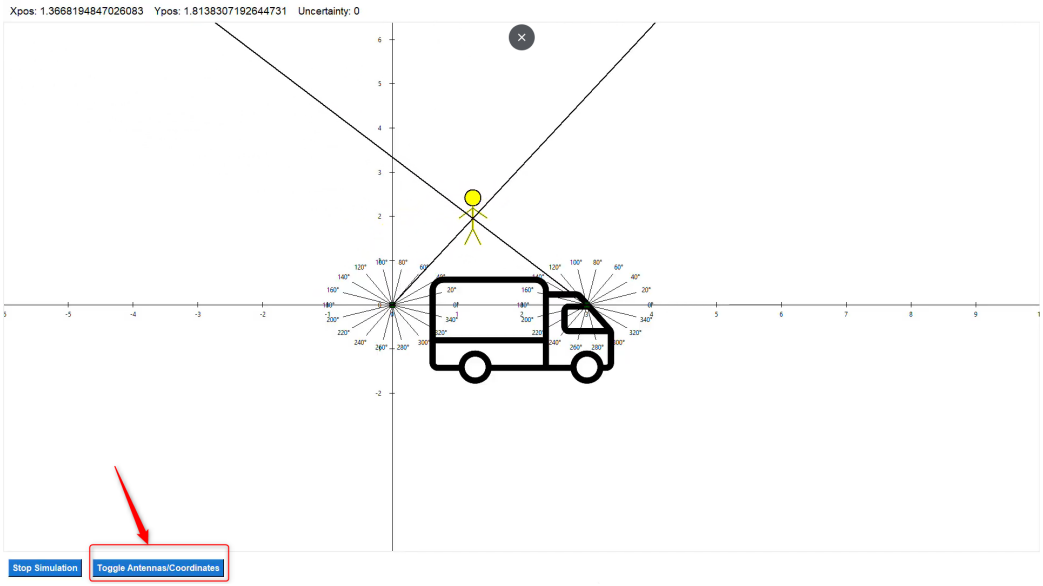

Abbildung 13: Volle Darstellung Benutzeroberfläche [14]

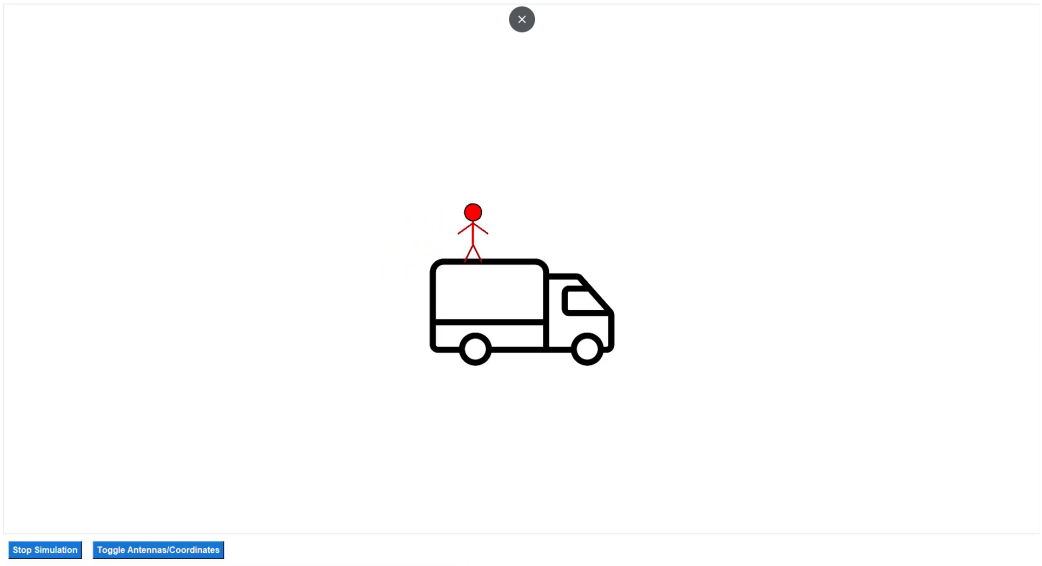

Abbildung 14: Minimale Darstellung Benutzeroberfläche [14]

Zusammenfassung Die überarbeitete Visualisierung stellt eine signifikante Verbesserung gegenüber der früheren Version dar. Sie bietet nicht nur eine realitätsnähere und dynamischere Darstellung der Position des Tags, sondern erleichtert durch optionale Darstellungselemente und glattere Bewegung auch die Analyse und Präsentation für unterschiedliche Zielgruppen. Gleichzeitig wurde Wert auf eine modulare und wartbare Struktur gelegt, sodass zukünftige Erweiterungen – etwa die Integration weiterer Visualisierungselemente oder zusätzlicher Sensoren – problemlos möglich sind.

6.4 Ergebnisse und Benutzerfreundlichkeit

Die Implementierung der verbesserten Visualisierung zeigt insgesamt funktionale Fortschritte und eine gesteigerte Benutzerfreundlichkeit. Die Anbindung über TCP funktioniert stabil sowohl im Echtzeitbetrieb mit Live-Messdaten als auch im Offline-Modus über den Datenlogger. Dies ermöglicht eine flexible Analyse und Bewertung von Positionsdaten ohne permanenten Zugriff auf die Hardware. Die Integration der Replay-Funktion in die Benutzeroberfläche verläuft nahtlos – gespeicherte Daten werden zuverlässig geladen, verarbeitet und dargestellt. Dies bietet insbesondere im Entwicklungs- und Testkontext deutliche Vorteile, da Anpassungen und neue Algorithmen unabhängig von einer aktiven Messumgebung evaluiert werden können.

Im Bereich der Darstellung konnte mit der Ersetzung des bisherigen Tag-Punktes durch eine abstrahierte Stickman-Figur ein deutlicher visueller Fortschritt erzielt werden. Zusätzlich wurden Warnstufen eingeführt, welche die Distanz zu einem virtuellen LKW durch Farbwechsel (grün, gelb, rot) kodieren. Auch die Antennenpositionen und deren Richtungsinformation werden grafisch dargestellt, wenngleich deren Schnittpunkt noch nicht exakt mit der berechneten Tag-Position übereinstimmt. Insgesamt ist die Visualisierung deutlich intuitiver und eignet sich besser zur Präsentation auf Fachmessen oder vor nicht-technischem Publikum. Ein optional zuschaltbares Koordinatensystem erlaubt zudem eine differenzierte technische Auswertung.

Die Bewegung des Tags erscheint derzeit noch leicht ruckartig und verzögert. Es wurden bereits Maßnahmen zur Glättung durch exponentielles Smoothing implementiert, allerdings zeigen sich in der Praxis weiterhin Limitierungen, die in zukünftigen Iterationen adressiert werden müssen.

Die Bedienbarkeit der Oberfläche ist insgesamt positiv zu bewerten. Die grafische Benutzeroberfläche zeigt beim Start die relevanten Schaltflächen gut sichtbar an. Durch den Toggle-Modus können Darstellungsmodi flexibel gewechselt werden – beispielsweise die Ein- und Ausblendung technischer Hilfslinien. Die Benutzerführung ist intuitiv und ermöglicht auch technisch weniger versierten Anwendern eine effiziente Nutzung.

Zusammenfassend lässt sich festhalten, dass die entwickelte Benutzeroberfläche eine stabile Basis für zukünftige Erweiterungen bietet. Neue Features können mit überschaubarem Aufwand ergänzt werden, und die vorhandene Architektur erlaubt eine Trennung von Datenaufnahme und Visualisierung. Damit stellt das System ein praktikables Werkzeug für die Weiterentwicklung, Testung und Demonstration von Assistenzsystemen dar.

7 Test und Validierung

7.1 Testmethodik und -umgebung

Ziel der Tests war es, die Funktionsweise der entwickelten Komponenten – insbesondere des Datenloggers und der neuen Visualisierung – in einer realitätsnahen, aber bewusst reduzierten Umgebung zu validieren. Es handelte sich dabei nicht um umfangreiche Praxistests in verschiedensten Anwendungsszenarien, sondern um gezielte Funktionstests, die Fehlerquellen aufdecken und Verbesserungspotenziale aufzeigen sollten [17].

Testziele Im Mittelpunkt stand die Überprüfung folgender Aspekte:

- Funktion der Datenerfassung, Speicherung und Wiedergabe im Replay-Modus
- Konsistenz der TCP-Kommunikation zwischen Logger und UI
- korrekte Verarbeitung gespeicherter JSONL-Daten
- visuelle Rückmeldung in der Benutzeroberfläche (z. B. Darstellung des Tags, Antennen und Warnfarben)
- allgemeine Bedienbarkeit und Reaktion der UI-Komponenten [18]

Testarten Es wurden verschiedene Testarten angewandt:

- **Funktionstests**: Einzelne Module wurden separat geprüft (z. B. Speicherung, Wiedergabe, Visualisierung)
- **Integrationstests**: Zusammenspiel von Datenlogger, Netzwerkverbindung und Visualisierung
- **Systemtests**: Gesamtprüfung mit Live- oder gespeicherten Datenströmen

Testumgebung Die Tests wurden auf einem Standard-PC unter Windows 11 mit Python 3.12 durchgeführt. Die Kommunikation lief lokal über eine TCP-Verbindung (`localhost`), was eine stabile Übertragung ohne externe Netzwerkeinflüsse sicherstellte. Als Datenbasis kamen sowohl Live-Messungen mit angeschlossener Hardware als auch gespeicherte JSONL-Dateien aus früheren Sessions zum Einsatz. Die grafische Benutzeroberfläche wurde dabei im Vollbildmodus auf einem 17-Zoll-Bildschirm (Laptop) betrieben.

Ablauf Die Tests erfolgten manuell. Zunächst wurde die Verbindung zwischen Logger und UI geprüft. Anschließend wurden verschiedene Datensätze – sowohl reale Messungen als auch synthetische Beispiele – über die `replayLogger.py [14]` wiedergegeben. Dabei wurde beobachtet, wie die UI auf Positionsdaten, Sensordaten und Änderungen in der Darstellung reagiert. Auch die neue Funktion zur Darstellung eines Stickmans und des LKW-Bildes wurde systematisch getestet. Der Wechsel des Darstellungsmodus (z. B. Ein-/Ausblenden der Achsen) wurde ebenfalls geprüft.

Ergänzend dazu wurde gezielt versucht, typische Grenzsituationen und Extremszenarien zu simulieren. Dazu gehörten beispielsweise sehr schnelle oder sehr langsame Bewegungen des Tags sowie Positionen am äußersten linken oder rechten Rand des Koordinatensystems. Ziel war es zu prüfen, ob die Positionsberechnung, Visualisierung und Stabilität auch in diesen Randbereichen erwartungsgemäß funktionieren.

Einschränkungen Aufgrund zeitlicher und infrastruktureller Begrenzungen wurde auf umfangreiche Feldversuche oder Szenarien im Realbetrieb verzichtet. Ziel war primär eine funktionale Prüfung der neu entwickelten Komponenten. Eine tiefgreifende Evaluierung im produktiven Umfeld ist für spätere Projektphasen vorgesehen.

7.2 Ergebnisse der Tests

Die durchgeführten Tests hatten zum Ziel, die grundsätzliche Funktionalität des entwickelten Datenloggers sowie der erweiterten Visualisierung zu überprüfen. Dabei standen insbesondere Aspekte wie Datenintegrität, Kommunikation, Benutzerfreundlichkeit und visuelle Darstellung im Vordergrund.

Funktionalität und Kommunikation Die Tests zeigten, dass die Speicherung der erfassten Sensordaten im JSONL-Format zuverlässig funktioniert. Auch die Wiedergabe der gespeicherten Daten im Replay-Modus über `replayLogger.py [14]` erwies sich als stabil. Die TCP-Kommunikation zwischen Datenlogger und grafischer Benutzeroberfläche konnte ohne Verbindungsabbrüche oder Datenverluste durchgeführt werden. Es war möglich, sowohl reale Messdaten als auch gespeicherte Daten aus dem Logger erfolgreich in die Benutzeroberfläche zu laden.

Darstellung des Tags Die Bewegung des Tags innerhalb der grafischen Oberfläche erwies sich noch als verbesserungsbedürftig. Obwohl bereits ein Glättungsalgorithmus implementiert wurde, wirkt die Bewegung weiterhin

leicht verzögert und ruckartig. Besonders bei Extrempositionen – also wenn sich der Tag weit rechts oder links außerhalb des zentralen Bereichs bewegt – konnte beobachtet werden, dass die Ortung deutlich an Genauigkeit verliert. Dies liegt am zugrunde liegenden Triangulationsverfahren: Wenn sich beide gemessenen Winkel stark überlagern, ist eine exakte Positionsbestimmung kaum noch möglich [19].

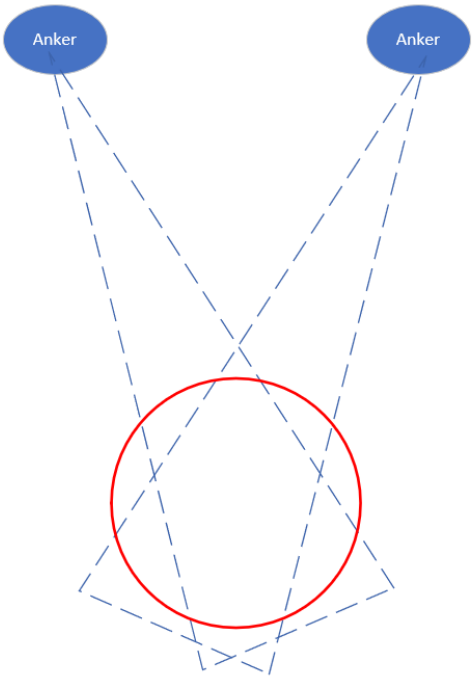

Abbildung 15: Gute Triangulation
Quelle:Eigene Darstellung

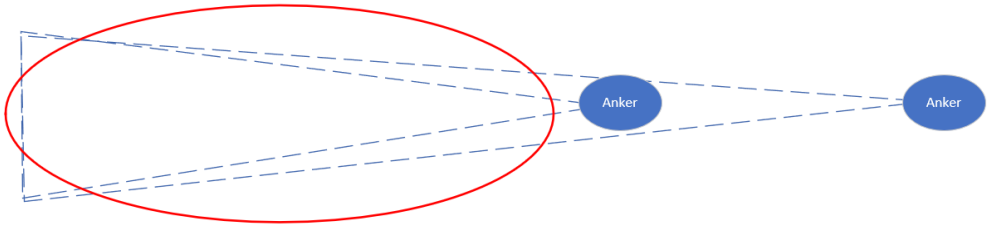

Abbildung 16: Schlechte Triangulation
Quelle:Eigene Darstellung

Visualisierung und Layout Die Darstellung wurde im Vergleich zum vorherigen Zustand deutlich verbessert. Anstelle eines einfachen Punktes wird der Tag nun als stilisierter Stickman visualisiert. Darüber hinaus wurde ein LKW-Bild zur Verdeutlichung der Anwendungssituation ergänzt. Die Antennenpositionen werden grafisch dargestellt und mit Richtungslinien versehen, welche sich in den meisten Fällen korrekt im Bereich der ermittelten Position schneiden – ein klares visuelles Feedback über die Funktionsweise des Systems. Dennoch bleibt die Darstellung grundsätzlich einfach gehalten. Zusätzliche Visualisierungselemente wie eine Geschwindigkeitsanzeige, eine dynamische Skalierung des Bewegungsraums oder realistischere Proportionen der dargestellten Objekte könnten künftig integriert werden.

Benutzerfreundlichkeit Die Benutzeroberfläche erwies sich in der Praxis als einfach zu bedienen. Beim Starten des Programms werden klar beschriftete Schaltflächen angezeigt, die ohne Vorkenntnisse verständlich sind. Funktionen wie das Starten und Stoppen der Simulation oder das Umschalten von Koordinaten- und Antennenanzeige sind intuitiv zugänglich. Die Implementierung der batch (BAT)-Datei zum Start des Datenloggers erleichtert zusätzlich die Bedienung, insbesondere für unerfahrene Nutzer [18]. Zudem kann zwischen verschiedenen gespeicherten Messreihen gewählt, diese umbenannt und erneut abgespielt werden.

Zusammenfassende Bewertung Insgesamt wurde mit dem entwickelten System ein funktionaler Prototyp geschaffen, der die gewünschten Kernfunktionen erfüllt. Die Integration des Datenloggers in die bestehende Softwareumgebung ist gelungen und erlaubt eine flexible Nutzung, sowohl online als auch offline. Die Visualisierung ist verständlicher als zuvor, besonders im

Hinblick auf Präsentationen und Messen. Dennoch bestehen in mehreren Bereichen Potenziale zur weiteren Optimierung, insbesondere in der flüssigen Bewegung des Tags, in der Genauigkeit der Darstellung sowie in der Erweiterung des visuellen Feedbacks für komplexere Anwendungsszenarien.

7.3 Diskussion der Testergebnisse und Optimierungspotentiale

Die durchgeführten Funktionstests zeigen, dass die wesentlichen Kernziele – insbesondere die prototypische Realisierung des Datenloggers und dessen Anbindung an die Visualisierung – erfüllt wurden. Dennoch offenbaren die Testergebnisse mehrere Schwachstellen und Hinweise auf künftigen Verbesserungsbedarf, die im Folgenden diskutiert werden.

Limitierungen in der Positionsdarstellung Trotz implementierter Glättungsmechanismen bleibt die Bewegung des Tags auf der Benutzeroberfläche merklich ruckartig. Insbesondere bei schnellen Richtungswechseln oder Bewegungen am Rand des Koordinatensystems treten Verzögerungen auf. Dies kann auf die gewählten Smoothing-Parameter oder das einfache Exponentialfilter-Verfahren zurückgeführt werden, das bei dynamischen Bewegungen an seine Grenzen stößt. Eine adaptive Glättung, die sich an der Bewegungsgeschwindigkeit orientiert, könnte hier künftig Abhilfe schaffen.

Ein weiterer kritischer Punkt ist die eingeschränkte Genauigkeit der Positionsbestimmung bei extremen Positionen. Wenn sich der Tag sehr weit links oder rechts außerhalb der Hauptmessachse befindet, versagt die Triangulation teilweise. Der Grund liegt in der Geometrie der AoA-Messung: Bei annähernd parallelen Winkelmessungen zweier Anker wird der Schnittpunkt der Geraden numerisch instabil, was eine präzise Lokalisierung erschwert [19].

Darstellung und Interpretation der Ergebnisse Die Visualisierung wurde gegenüber der ursprünglichen Version deutlich verbessert. Die Einführung eines symbolischen Stickman-Modells sowie eines schematischen LKW-Bildes erhöht die Verständlichkeit, besonders für externe Zielgruppen oder Präsentationen. Gleichzeitig bleibt die grafische Ausgestaltung noch rudimentär. Elemente wie Maßstabsanzeige, Bewegungsverläufe, Risikozonen oder Geschwindigkeitspfeile fehlen bislang, könnten aber die Interpretation der Szenarien erheblich erleichtern.

Darüber hinaus könnte auch die Proportionalität zwischen realer Bewegung und Koordinatensystem verbessert werden. Aktuell ist die Zuordnung zwischen physischer Verschiebung und grafischer Darstellung nicht intuitiv

nachvollziehbar, was insbesondere bei der Validierung der Messergebnisse hinderlich sein kann.

Bedienbarkeit und Nutzbarkeit Die Bedienung der Visualisierungsoberfläche gestaltet sich insgesamt benutzerfreundlich. Die eingefügten Steuerungselemente (z. B. Toggle-Schalter für die Darstellung der Koordinatenachsen) sowie die Möglichkeit, das Programm per `.bat`-Datei zu starten, erleichtern die Nutzung auch für weniger erfahrene Anwender. Die einfache Darstellung ist für erste Tests und Demonstrationen durchaus ausreichend, lässt sich jedoch hinsichtlich Detailtiefe und Flexibilität weiterentwickeln.

Potenziale zur Erweiterung Für die Weiterentwicklung ergeben sich mehrere sinnvolle Ansätze. Neben einer grafischen Verfeinerung könnten folgende Maßnahmen zur Optimierung beitragen:

- Einführung eines dynamischen Zoom- oder Perspektivmodus zur besseren Verfolgung der Tag-Bewegung
- Erweiterung der Darstellung um Geschwindigkeit, Bewegungsrichtung und Unsicherheitsbereiche
- Modularisierung der Visualisierung zur einfacheren Einbindung weiterer Objekte oder Assistenzsysteme
- Verbesserung der Glättungslogik mit adaptiven oder maschinellen Verfahren
- Vorbereitung auf künftige Erweiterungen wie eine 3D-Ansicht

Fazit Die bisherigen Testergebnisse zeigen eine solide Basis, auf der sich weiter aufbauen lässt. Sowohl die technische Funktionalität als auch die visuelle Vermittlung der Daten sind grundsätzlich gegeben, müssen jedoch für eine belastbare Systembewertung und breitere Nutzbarkeit noch weiterentwickelt und verfeinert werden.

8 Kritische Bewertung und Ausblick

8.1 Reflexion der erreichten Ergebnisse

Im Rahmen dieser Arbeit wurde das Ziel verfolgt, ein robustes Datenloggersystem zu entwickeln sowie die bestehende Visualisierung einer Bluetooth-AoA-basierten Lokalisierungslösung gezielt zu erweitern. Rückblickend lässt sich feststellen, dass wesentliche Kernziele erfolgreich umgesetzt wurden.

Der entwickelte Datenlogger ermöglicht eine strukturierte und flexible Erfassung sowie Speicherung von Messdaten im JSONL-Format. Er erlaubt eine spätere Wiedergabe der Daten ohne Hardwareeinsatz und schafft somit eine effektive Grundlage für Tests, Parametrierung und Algorithmenentwicklung. Die Integration des Loggers in die bestehende Softwareumgebung erfolgte reibungslos durch ein separates Python-Skript mit minimalen Änderungen am Ursprungscode. Auch die Konnektivität zur Benutzeroberfläche über eine TCP-Verbindung funktionierte stabil, sowohl im Live-Betrieb als auch im Replay-Modus.

Auf Seiten der Visualisierung wurde die bisher sehr reduzierte Darstellung um mehrere Elemente erweitert. Die Verwendung eines stilisierten Stickman-Modells zur Repräsentation des Tags, die Einbindung eines realitätsnahen LKW-Bildes sowie die farbliche Warnstufendarstellung führten zu einer deutlich verständlicheren und praxistauglicheren Repräsentation. Dies ist insbesondere im Hinblick auf externe Präsentationen, z. B. auf Messen oder Präsentationen, ein entscheidender Vorteil.

Trotz dieser Fortschritte zeigen sich auch einige Grenzen. Die Bewegung des Tags ist nach wie vor leicht verzögert und nicht vollständig flüssig, obwohl bereits erste Maßnahmen zur Glättung implementiert wurden. In Randbereichen des Messfeldes stößt die Triangulation durch sich überlappende Winkelpaare an ihre mathematischen Grenzen, was zu ungenauen oder nicht vorhandenen Positionsschätzungen führt. Auch die visuelle Darstellung – etwa in Bezug auf Maßstäblichkeit oder ergänzende Informationen wie Geschwindigkeit – bietet weiteres Verbesserungspotenzial.

Nichtsdestotrotz konnte durch die modulare Struktur, den Einsatz etablierter Technologien (Python, JSON, TCP) sowie den pragmatischen Ansatz ein funktionierendes System geschaffen werden, das die Grundlage für weiterführende Entwicklungen bietet. Die Arbeit zeigt damit deutlich, dass durch gezielte Ergänzungen und strukturiertes Vorgehen auch in einer begrenzten Projektlaufzeit signifikante Fortschritte in Funktionalität und Benutzerfreundlichkeit erzielt werden können.

8.2 Grenzen der aktuellen Umsetzung

Trotz der erfolgreichen Umsetzung eines funktionalen Datenloggers und einer verbesserten Visualisierung bestehen derzeit noch mehrere Einschränkungen, die das System in seiner Einsatzbreite und Qualität begrenzen.

Begrenzte Genauigkeit der Positionsbestimmung Die Positionsschätzung des Tags basiert auf Triangulation aus den gemessenen AoA-Winkeln zweier Anker. Dieses Verfahren liefert nur dann zuverlässige Ergebnisse, wenn die geometrische Anordnung der Anker günstig gewählt ist. Besonders bei Messungen an den Rändern des definierten Messfelds (z. B. weit links oder rechts) kann es aufgrund der Überlagerung oder Parallelität der Winkellinien zu ungenauen oder gar nicht berechenbaren Positionen kommen.

Nicht vollständig realitätsgetreue Bewegungsdarstellung Obwohl Maßnahmen zur Glättung der Bewegung (z. B. mittels Pufferung und Smoothing) umgesetzt wurden, wirkt die Visualisierung des bewegten Tags weiterhin leicht ruckartig und reagiert nicht in Echtzeit. Zudem fehlt derzeit eine maßstabsgetreue Übertragung der tatsächlichen Bewegung auf die Bildschirmanzeige.

Einfaches visuelles Layout Die Visualisierung nutzt bisher einfache grafische Elemente wie einen stilisierten Stickman und ein eingefügtes Bild eines LKWs. Die Darstellung unterstützt die Verständlichkeit, lässt jedoch hinsichtlich Detailtiefe, Ästhetik und Interaktivität noch deutlichen Spielraum für Verbesserungen. Eine Möglichkeit zur perspektivischen Darstellung, Animation komplexerer Objekte oder eine interaktive Benutzersteuerung ist bislang nicht vorgesehen.

Begrenzte Testtiefe Die entwickelte Lösung wurde im Rahmen kleinerer Testszenarien mit kurzer Laufzeit erprobt. Umfangreiche Testreihen unter realitätsnahen Bedingungen – etwa mit komplexer Bewegung, Störeinflüssen oder Mehrpersonen-Szenarien – konnten zeitbedingt nicht durchgeführt werden.

Fehlende erweiterte Analyse- und Mehrbenutzerfunktionen Die aktuelle Implementierung erlaubt eine visuelle Einzelnutzung mit Echtzeitdarstellung bzw. Offline-Replay. Weiterführende Funktionalitäten wie gleichzeitiger Mehrbenutzerzugriff, interaktive Analysewerkzeuge, Exportfunktio-

nen oder statistische Auswertungen (z. B. Heatmaps) sind derzeit nicht enthalten.

Technische Einschränkungen der Visualisierung Die Umsetzung der Benutzeroberfläche auf Basis von Python und Tkinter bringt den Vorteil einer schnellen Realisierbarkeit, schränkt jedoch die grafischen und performanten Möglichkeiten ein. Zudem erfordert der Datenlogger eine strukturierte Ablage der JSONL-Dateien und Pfade, was die Portabilität des Systems derzeit noch begrenzt.

Diese Punkte zeigen klar auf, in welchen Bereichen gezielte Weiterentwicklungen erforderlich sind, um das System robuster, skalierbarer und praxistauglicher zu gestalten.

8.3 Vorschläge für zukünftige Weiterentwicklungen

Basierend auf den Ergebnissen und Erkenntnissen der bisherigen Arbeit ergeben sich verschiedene Ansatzpunkte für eine sinnvolle Weiterentwicklung des entwickelten Systems.

Technische Optimierung der Bewegungsglättung Die Bewegung des Tags ist in der aktuellen Visualisierung noch nicht ausreichend flüssig. Obwohl bereits ein Kalman-basiertes Glättungsverfahren implementiert wurde, treten weiterhin ruckartige Darstellungen und spürbare Verzögerungen auf. Eine Verbesserung kann erzielt werden, indem die Kalman-Parameter – etwa Prozess- und Messrauschen – adaptiv angepasst und feiner auf die Dynamik des Systems abgestimmt werden, um eine realitätsnähere Bewegung des Tags zu erreichen [20].

Verbesserung der Triangulation bei Randlagen Ein beobachtetes Problem betrifft die Genauigkeit der Positionsbestimmung, insbesondere wenn sich das Tag am Rand des Erfassungsbereichs befindet. Hierbei treten geometrische Probleme auf, da die beiden Winkelmessstrahlen nahezu parallel verlaufen und die Schnittpunktbestimmung ungenau wird. Eine Möglichkeit zur Verbesserung besteht in der Integration eines dritten Antennen-Boards. Durch die zusätzliche Winkelmessung kann die geometrische Ambiguität reduziert und die Lokalisierungsgenauigkeit signifikant gesteigert werden.

Erweiterung der Replay-Funktionalität Der Datenlogger erlaubt bereits das Abspielen aufgezeichneter Messdaten. In zukünftigen Versionen

könnte diese Funktion erweitert werden, um Parameter direkt über eine grafische Benutzeroberfläche anpassen zu können. Eine visuelle Konfigurationsmaske mit Live-Feedback würde die Entwicklungs- und Testphase erheblich vereinfachen.

Erweiterung der Visualisierungsmöglichkeiten Die Visualisierung des Tags wurde bereits durch einen stilisierten Stickman sowie ein LKW-Modell ergänzt. Um jedoch noch realitätsnähere und professionellere Darstellungen zu ermöglichen – insbesondere bei öffentlichen Vorführungen oder Messen – könnten 3D-Modelle eingesetzt werden. Auch zusätzliche visuelle Indikatoren wie Bewegungstrails, Heatmaps oder Geschwindigkeitssymbole wären denkbar [21].

Zudem könnte das Koordinatensystem um interaktive Funktionen wie Zoom, Skalierung oder Ein-/Ausblendung erweitert werden. Mittelfristig wäre auch der Umstieg auf eine leistungsfähigere Visualisierungsplattform, etwa auf Basis von Unity3D oder WebGL, zu evaluieren [22].

Verbesserung der Benutzeroberfläche Die aktuelle UI bietet bereits eine grundlegend benutzerfreundliche Bedienung. Zukünftig wäre jedoch eine webbasierte Variante wünschenswert, die plattformunabhängig funktioniert und auch über mobile Endgeräte steuerbar ist. Eine intuitive Konfiguration von Replay-, Logging- und Visualisierungsparametern würde die Nutzerfreundlichkeit zusätzlich steigern [23].

Tests und Validierung Die durchgeführten Tests waren in ihrem Umfang noch begrenzt. Für zukünftige Weiterentwicklungen sollte ein strukturierter Testkatalog entwickelt werden, der typische Anwendungsszenarien, Grenzfälle und Langzeittests umfasst. Dadurch ließe sich die Robustheit und Zuverlässigkeit der Lösung weiter erhöhen.

Systemintegration und Modularisierung Langfristig wäre eine stärkere Modularisierung der Softwarestruktur anzustreben. Durch eine saubere Trennung von Datenaufnahme, Auswertung, Visualisierung und Replay-Modulen ließe sich die Wartbarkeit verbessern. Auch die Integration externer Sensoren oder Cloud-Plattformen zur weitergehenden Analyse könnte über standardisierte Schnittstellen ermöglicht werden [24].

Zusammenfassung Insgesamt existieren vielfältige Ansatzpunkte für die Weiterentwicklung der Lösung. Diese reichen von technischen Verbesserungen der Positionsbestimmung und Visualisierung über benutzerfreundliche

UI-Konzepte bis hin zur Erweiterung des Hardware-Setups durch zusätzliche Anker. Gerade die Kombination dieser Maßnahmen verspricht eine deutliche Steigerung der Systemqualität und Anwendbarkeit in praxisnahen Einsatzszenarien.

Literatur

[1] Abbiegeassistent.de. „Förderung von Abbiegeassistenzsystemen.“ Bildquelle zu Abb. 1, besucht am 13. Juli 2025. Adresse: `https://www.abbiegeassistent.de/foerderung/`.

[2] „Slight Increase in the Number of Road Fatalities in 2023.“ Springer Professional; Unfallstatistik 2023, besucht am 13. Juli 2025. Adresse: `https://www.springerprofessional.de/road-safety/companies---institutions/slight-increase-in-the-number-of-road-fatalities-in-2023/26824974`.

[3] K. Townsend, C. Cufí, Akiba und R. Davidson, *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. O'Reilly Media, 2021.

[4] u-blox AG. „Using Bluetooth AoA for High-Precision Indoor Positioning.“ Whitepaper zur AoA-Funktionsweise, besucht am 13. Juli 2025. Adresse: `https://www.u-blox.com/en/docs/UBX-19055559`.

[5] u-blox AG. „XPLR-AOA-2 Kit: Bluetooth AoA/AoD Explorer Kit.“ Bildquelle zu Abb. 2, besucht am 13. Juli 2025. Adresse: `https://www.u-blox.com/en/product/xplr-aoa-2-kit`.

[6] Android Developers. „Bluetooth Low Energy — Android Developers Guide.“ Zugriffsbeschränkungen auf BLE-Sendeparameter unter Android 12+, besucht am 13. Juli 2025. Adresse: `https://developer.android.com/guide/topics/connectivity/bluetooth-le`.

[7] H. Austerlitz, *Data Acquisition Techniques Using PCs*, 2. Aufl. San Diego: Academic Press, 2002, Standardwerk zu PC-basierten Datenaufnahmesystemen.

[8] J. Fraden, *Handbook of Modern Sensors: Physics, Designs, and Applications*, 5. Aufl. Cham: Springer, 2016, Grundlagenwerk zu Sensorprinzipien u. Datenerfassung.

[9] Wikipedia contributors. „Data Logger — Wikipedia, the free encyclopedia.“ Bildquelle zu Abb. 3, besucht am 13. Juli 2025. Adresse: `https://en.wikipedia.org/wiki/Data_logger`.

[10] Dewesoft. „What Is a Data Logger (Datalogger) – The Ultimate Guide,“ besucht am 13. Juli 2025. Adresse: `https://dewesoft.com/blog/what-is-data-logger`.

[11] RS Components. „A Complete Guide to Data Loggers,“ besucht am 13. Juli 2025. Adresse: `https://uk.rs-online.com/web/content/discovery/ideas-and-advice/data-loggers-guide`.

[12] H. P. Langtangen, „Combining Python with Fortran, C, and C++,“ in *Python Scripting for Computational Science*, Springer, 2004, S. 169–204.

[13] A. Isaiah. „A Beginner's Guide to JSON Logging,“ besucht am 13. Juli 2025. Adresse: `https://betterstack.com/community/guides/logging/json-logging`.

[14] L. Tadic. „Studienarbeit – GitHub Repository (Branch “ui”).“ Quellcodebasis zur Visualisierungsoberfläche, besucht am 13. Juli 2025. Adresse: `https://github.com/Lukiano12/Studienarbeit/tree/ui`.

[15] ComConsult GmbH. „Ortung via WLAN – Funktionsweise und Einsatzgebiete.“ Abbildung zur Triangulation (Abb. 4), besucht am 13. Juli 2025. Adresse: `https://www.comconsult.com/ortung-wlan/`.

[16] A. D. Moore, *Python GUI Programming with Tkinter*, 2. Aufl. Packt Publishing, 2021.

[17] G. J. Myers, T. Sandler und C. Badgett, *The Art of Software Testing*, 4. Aufl. Wiley, 2020.

[18] J. Nielsen. „Usability 101: Introduction to Usability,“ besucht am 13. Juli 2025. Adresse: `https://www.nngroup.com/articles/usability-101-introduction-to-usability/`.

[19] V. A. Thurmond, „The point of triangulation,“ *Journal of nursing scholarship*, Jg. 33, Nr. 3, S. 253–258, 2001.

[20] D. Simon, *Optimal State Estimation: Kalman, H-Infinity, and Nonlinear Approaches*. Wiley, 2006.

[21] T. Parisi, *WebGL: Up and Running*. O'Reilly Media, 2012.

[22] I. Buyuksalih, S. Bayburt, G. Buyuksalih, A. Baskaraca, H. Karim und A. A. Rahman, „3D modelling and visualization based on the unity game engine–advantages and challenges,“ *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Jg. 4, S. 161–166, 2017.

[23] D. Norman, *The Design of Everyday Things*, Revised. MIT Press, 2013.

[24] L. Bass, P. Clements und R. Kazman, *Software Architecture in Practice*, 4. Aufl. Addison-Wesley, 2021.