HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

# BACHELORARBEIT

**Herr**
**Trushal Babubhai Paghadar**

## Entwicklung einer Datenerfassung in ROS für die Markierung von betrieblich relevanten Ereignissen im Bahnbereich

**Mittweida,2023**

Fakultät: Ingenieurwissenschaften

# BACHELORARBEIT

## Entwicklung einer Datenerfassung in ROS für die Markierung von betrieblich relevanten Ereignissen im Bahnbereich


Autor:
**Herr Trushal Babubhai Paghadar**

Studiengang:
**Elektrotechnik - Automation**

Seminargruppe:
**EA19wA-B**

Erstprüfer:
**Prof. Dr.-Ing. Jan Thomanek**

Zweitprüfer:
**Dr.-Ing. Henrik von der Haar**



Einreichung:
**Mittweida, 30.04.2023**

Verteidigung/Bewertung:
**Mittweida,2023**

Faculty: Ingenieurwissenschaften

# BACHELORTHESIS

# Development of a data recording in ROS for the marking of relevant events in the railway sector

Autor:
**Mr. Trushal Babubhai Paghadar**

course of studies:
**Elektrotechnik - Automation**

seminar group:
**EA19wA-B**

first examiner:
**Prof. Dr.-Ing. Jan Thomanek**

second examiner:
**Dr.-Ing. Henrik von der Haar**

submission:
**Mittweida, 30.04.2023**

defence/ evaluation:
**Mittweida,2023**

# Bibliografische Angaben

Paghadar, Trushal Babubhai:

Entwicklung einer Datenerfassung in ROS für die Markierung von betrieblich relevanten Ereignissen im Bahnbereich.

Development of a data recording in ROS for the marking of relevant events in the railway sector.

- 51 S. Mittweida, Hochschule Mittweida, University of Applied Sciences, Fakultät Ingenieurwissenschaften,

Bachelorarbeit, 2023

# Referat:

Die Bachelorarbeit beschäftigt sich mit der Entwicklung einer Datenerfassung in ROS zur Markierung von betriebsrelevanten Ereignissen im Bahnbereich. Dazu wurden zunächst einige mögliche Implementierungen im Perzeptionssystem näher betrachtet und anschließend einige praktische Tests durchgeführt. Nach der Anwendung dieser Implementierung im Perzeptionssystem wurde sie im Perzeptionslabor getestet, und die Ergebnisse sind hier dokumentiert.

# Vorwort

Ich freue mich, Ihnen diese Bachelorarbeit vorstellen zu dürfen, die zwischen Februar 2023 und April 2023 bei der DB Systemtechnik GmbH in Minden entstanden ist. Diese Arbeit wurde unter der Betreuung von Prof. Dr.-Ing. Jan Thomanek von der Hochschule Mittweida und Dr.-Ing. Henrik von der Haar von der DB Systemtechnik durchgeführt.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Jan Thomanek, der mich während meines Praktikums und meiner Bachelorarbeit unterstützt und betreut hat. Besonderer Dank gebührt auch Dr.-Ing. Henrik von der Haar für seine technische und organisatorische Unterstützung sowie für seine wertvollen Anregungen und Ratschläge.

Ich danke auch meinen Kolleginnen und Kollegen bei der DB Systemtechnik, die mir hervorragende Empfehlungen gaben und meine Arbeit durch ein freundliches und unterstützendes Arbeitsumfeld erleichterten.

Schließlich möchte ich mich bei meiner Familie bedanken, insbesondere bei meinem Vater und meiner Mutter, Herrn Babubhai Paghadar und Frau Narmadaben Babubhai Paghadar, die mir das Studium ermöglicht und mich auf diesem Weg stets unterstützt haben. Ich bin auch meinen Freunden für ihre Ermutigung und Unterstützung dankbar.

Ich hoffe, dass diese Bachelorarbeit einen Beitrag zum Bereich der Automatisierungstechnik im Eisenbahnsektor leistet und als wertvolle Ressource für zukünftige Forschungen dienen wird.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# 1 Einleitung

## 1.1 Firmenportrait Deutsche Bahn und DB Systemtechnik

Die Deutsche Bahn (DB) ist ein deutsches Eisenbahnunternehmen, das 1994 gegründet wurde. Sie ist einer der größten Bahnbetreiber in Europa, der sowohl im Güter- und Personenverkehr als auch in der Logistik und im Infrastrukturmanagement tätig ist. Das Unternehmen ist in 130 Ländern tätig und hat mehr als 300.000 Mitarbeiter. Neben dem Bahnbetrieb bietet die DB auch Bus- und Luftverkehrsdienste an [1].

Die DB Systemtechnik ist eine Tochtergesellschaft der Deutschen Bahn, die technische und ingenieurtechnische Dienstleistungen für Schienenverkehrssysteme anbietet. Sie wurde 1999 gegründet und hat ihren Sitz in Minden, Deutschland (Abbildung 1). Das Unternehmen beschäftigt über 1.000 Mitarbeiter und ist an mehreren Standorten in Deutschland und Europa tätig. Das Leistungsspektrum der DB Systemtechnik umfasst die Prüfung und Zertifizierung von Schienenfahrzeugen und -komponenten, die Entwicklung und Realisierung von Schienenverkehrssystemen sowie Beratungsleistungen für Schieneninfrastruktur und Schienenfahrzeuge [2].

Abbildung 1: DB Systemtechnik, (Minden) [3]

Darüber hinaus ist die DB Systemtechnik in der Forschung und Entwicklung im Bereich der Bahntechnik tätig. In Zusammenarbeit mit Hochschulen, Forschungseinrichtungen und anderen Unternehmen entwickelt das Unternehmen

innovative Lösungen für die Schienenverkehrsindustrie. Die DB Systemtechnik setzt sich für nachhaltige und effiziente Schienenverkehrssysteme ein [3].

DB Systemtechnik ist an insgesamt neun Standorten tätig. Die größten Standorte befinden sich in Minden und München, wo insgesamt jeweils knapp 300 Mitarbeiter beschäftigt sind. DB Systemtechnik Standort in Minden ein wichtiges Kompetenzzentrum für die deutsche Bahnindustrie und bietet Kunden im In- und Ausland ein breites Spektrum an hochwertigen Test-, Validierungs- und Engineering-Dienstleistungen. Darüber hinaus unterhält die DB Systemtechnik über ihre Tochterunternehmen ESG und RAL auch Niederlassungen in Derby (Großbritannien) und in Paris (Frankreich).

## 1.2 Das Perzeptionslabor

Das Perzeptionslabor in Minden ist eine unabhängige Einrichtung mit Kompetenzen in den Bereichen technische Wahrnehmung und Lokalisierung sowie automatisiertes Testmanagement für die sichere Integration von Systemen und Komponenten in das digitale und modulare Schienenverkehrssystem. Dies geschieht im funktional-technischen Zusammenspiel mit Automatic Train Operation (ATO), Automatic Train Protection (ATP) oder Automatic/National Train Control (ATC/NTC) Systemen. Das Perzeptionsslabor ist im Aufbau, um die Systementwicklung und Sicherheitsüberprüfung sowie die Produktzertifizierung zu unterstützen.

## 1.3 Problemstellung

Die Automatisierungstechnik hat die Bahnindustrie revolutioniert und zu mehr Sicherheit, kürzeren Fahrzeiten und höherer Energieeffizienz geführt. Fortschrittliche Sensoren, Kommunikationssysteme und Steuerungsalgorithmen haben eine präzise Steuerung der Zugbewegung ermöglicht und zur Entwicklung hochautomatisierter Züge geführt. Die Integration dieser Technologien hat das Bahnfahren für Fahrgäste und Bahnunternehmen komfortabler, bequemer und kostengünstiger gemacht. Die Zukunft des Zugverkehrs wird weiterhin von der Automatisierungstechnik abhängen, da die Bahnunternehmen bestrebt sind, die Sicherheit und Effizienz zu verbessern und gleichzeitig die Kosten zu senken.

Im Perzeptionslabor der DB Systemtechnik wird derzeit ein mobiles Perzeptionssystem entwickelt, um Trainingsdaten für die automatisierte Schadenserkennung oder für die Entwicklung von Fahrerassistenzsystemen für das automatisierte Fahren während der Zugfahrt zu sammeln. Das System besteht aus hochauflösenden Kameras, LiDARen, Sensoren und GNSS-Empfängern sowie weiterer Hardware, um diese Daten in geeigneter Weise zu verwalten.

In naher Zukunft wird es Züge geben, die mit Perzeptionssensoren ausgestattet sind, um während der gesamten Zugfahrt Trainingsdaten für das automatisierte Fahren zu sammeln (z.B. Daten von Kamera, LiDAR). Aber ist es wichtig, nicht alle Daten aufzuzeichnen und zu speichern, da die Datenmenge sehr groß ist und alle Daten für die nachfolgende Verarbeitung relevant sind. Es kann daher sinnvoll und praktisch sein, wenn nur die betrieblich relevanten Daten erfasst und gespeichert werden (z.B. Großtiere auf der Strecke). So wird es sehr einfach, nur die relevanten Daten zu verwalten und zu bearbeiten.

## 1.4 Zielsetzung und Herangehensweise

Zielsetzung: Ziel dieser Bachelorarbeit ist es, die Erfassung von Trainingsdaten für das automatisierte Fahren im Bahnbereich zu verbessern, indem ein effizienterer und praktikablerer Ansatz für die Erfassung und Speicherung relevanter Daten entwickelt wird. Der Schwerpunkt liegt auf der Entwicklung einen Taster-Mechanismus, der es den Triebfahrzeugführern ermöglicht, betriebsrelevante Ereignisse, wie z. B. Personen oder Tiere im Gleis oder Schäden an der Bahninfrastruktur zu markieren und aufzuzeichnen. Weiter soll ein Ringspeicherkonzept implementiert werden. Ziel ist es, die Erfassung und Verwaltung von Trainingsdaten für Eisenbahnunternehmen praktikabler, effizienter und automatischer zu gestalten.

Der Ansatz: Der erste Schritt zur Erreichung des Ziels besteht darin, einen Taster-Mechanismus in das Perzeptionssystems zu implementieren, der vom Triebfahrzeugführer während der Fahrt per Knopfdruck ausgelöst werden kann. Dieser Mechanismus ermöglicht es dem Triebfahrzeugführer relevante Ereignisse, die während der Fahrt auftreten, zu markieren und aufzuzeichnen. Sobald ein Ereignis markiert ist, speichert das Perzeptionssystems die relevanten Daten aus einem Ringspeicher in einen anderen Ordner.

Der zweite Schritt ist die Implementierung eines Ringspeicherkonzepts für die Speicherung der aufgezeichneten Ereignisse. Der Ringspeicher ermöglicht es dem System, nur eine bestimmte Menge an Daten zu speichern, und neue Daten überschreiben die ältesten Daten, wenn der Speicher voll ist. Die Umsetzung dieses Konzepts hilft, die Menge der aufgezeichneten Daten zu verwalten und den Verlust relevanter Daten zu verhindern.

Schließlich sollte der Beginn und das Ende der Datenaufzeichnung im Ringspeicher automatisch und in Abhängigkeit von der Fahrtrichtung erfolgen. Dies kann durch die Implementierung eines Encoders erreicht werden, der die Richtung des Zuges erkennt und den Beginn und das Ende der Datenaufzeichnung entsprechend auslöst.

Insgesamt wird dieser Ansatz dazu beitragen, die Effizienz und Praktikabilität der Erfassung und Verwaltung relevanter Trainingsdaten für das automatisierte Fahren im Bahnsektor zu verbessern.

## 1.5 Aufbau der Arbeit

Der Aufbau der Arbeit beginnt mit dem Einleitung Kapitel. In der Einleitung der Bachelorarbeit werden das Unternehmen Deutsche bahn und die DB Systemtechnik GmbH und Perzeptionslabor vorgestellt, bei denen die Arbeit entstanden ist. Außerdem werden die Problemstellung und das Ziel der Arbeit erläutert, die Vorgehensweise zur Erreichung des Ziels beschrieben und ein kurzer Überblick über den Aufbau der Arbeit gegeben.

Im zweiten Kapitel, dem Automatisierung und Datenerfassung im Bahnbereich, wird ein Überblick über die Automatisierungstechnik in der Bahnindustrie gegeben, die derzeit für das automatisierte Fahren im Bahnsektor verwendeten Perzeptionssysteme untersucht und die bestehenden Methoden zur Erfassung und Speicherung von Trainingsdaten bewertet. Kapitel 3 beschreibt das mobile Perzeptionssystem, das für die Datenerfassung verwendet wird, erläutert das ROS-Framework für die Datenaufzeichnung und -verarbeitung und liefert eine detaillierte Beschreibung der Trigger-, Ringspeicher und Drehgeberkomponenten des vorgeschlagenen Ansatzes.


Abbildung 2: Aufbau der Arbeit

Kapitel 4 enthält Informationen über die Implementierung des vorgeschlagenen Ansatzes auf dem mobilen Perzeptionssystem, einschließlich technischer Details zu den Komponenten Trigger, Ringspeicher und Drehgeber. Außerdem wird die Prüfung und Validierung des Ansatzes beschrieben. Kapitel 5 vergleicht den vorgeschlagenen Ansatz mit bestehenden Methoden, diskutiert die Vorteile und Grenzen des vorgeschlagenen Ansatzes und liefert eine detaillierte Analyse der erzielten Ergebnisse. Das letzte Kapitel, Kapitel 6, fasst die Schlussfolgerungen aus den Untersuchungen und der Diskussion zusammen und gibt Empfehlungen für künftige Forschungen zur weiteren Verbesserung des vorgeschlagenen Ansatzes.

# 2 Automatisierung und Datenerfassung im Bahnbereich

## 2.1 Überblick über die Automatisierungstechnik in der Bahnindustrie

Die Eisenbahnindustrie hat in den letzten Jahren dank der Integration der Automatisierungstechnik einen technologischen Wandel durchlaufen. Diese Technologie hat das Bahnfahren nicht nur komfortabler und kostengünstiger gemacht, sondern auch die Entwicklung autonomer Züge ermöglicht. Fortschrittliche Sensoren, Kommunikationssysteme und Steuerungsalgorithmen haben den Zugverkehr revolutioniert und für mehr Präzision und Kontrolle gesorgt. Die Einführung von Automatisierungstechnik in der Bahnindustrie hat mehrere Vorteile, darunter verbesserte Sicherheit, kürzere Reisezeiten und höhere Energieeffizienz.

Die Einführung der Automatisierungstechnik in der Bahnindustrie ist jedoch auch mit einigen Herausforderungen verbunden. Eine der größten Herausforderungen ist die Notwendigkeit erheblicher Investitionen in Infrastruktur und Technologie. Die Implementierung der Automatisierungstechnik erfordert hochentwickelte Sensoren, Kommunikationssysteme und Algorithmen, die in die bestehende Eisenbahninfrastruktur integriert werden müssen. Dies erfordert erhebliche Investitionen in Zeit und Geld, was für einige Bahnunternehmen ein Hindernis darstellen kann.

Eine weitere Herausforderung im Zusammenhang mit der Einführung von Automatisierungstechnik in der Bahnindustrie ist der mögliche Verlust von Arbeitsplätzen. Die Automatisierung verschiedener Prozesse, einschließlich der Zugbewegung und -steuerung, kann dazu führen, dass menschliche Arbeitskräfte durch Maschinen ersetzt werden. Dies kann zum Verlust von Arbeitsplätzen führen, insbesondere für ungelernte Arbeitskräfte oder solche, die Routinetätigkeiten ausüben.

Darüber hinaus stellen Wartung und Reparatur der automatisierten Systeme eine weitere Herausforderung dar. Die automatisierten Systeme sind komplex und

müssen regelmäßig gewartet werden, um ihr einwandfreies Funktionieren zu gewährleisten. Die mit der Wartung und Reparatur dieser Systeme verbundenen Kosten können hoch sein, insbesondere für kleine oder mittlere Eisenbahnunternehmen.

Sicherheitsbedenken sind ebenfalls eine große Herausforderung im Zusammenhang mit der Einführung von Automatisierungstechnik in der Bahnindustrie. Es besteht die Gefahr von Cyberangriffen, die die Sicherheit der automatisierten Systeme gefährden könnten. Außerdem könnten Systemausfälle zu Unfällen führen und die Sicherheit der Fahrgäste gefährden.

Daher ist es von entscheidender Bedeutung, die Vorteile und Herausforderungen im Zusammenhang mit der Einführung der Automatisierungstechnologie in der Bahnindustrie sorgfältig abzuwägen. Sie bietet zwar mehrere Vorteile wie verbesserte Sicherheit und Energieeffizienz, birgt aber auch erhebliche Herausforderungen wie hohe Kosten, potenzielle Arbeitsplatzverluste und Sicherheitsbedenken. Die Eisenbahnunternehmen müssen diese Faktoren abwägen, bevor sie sich für die Einführung der Automatisierungstechnik entscheiden, und Strategien zur Bewältigung der damit verbundenen Herausforderungen entwickeln [4].

## 2.2 Überprüfung von Perzeptionssystemen für das automatisierte Fahren im Eisenbahnsektor

Perzeptionssysteme sind entscheidend für die Entwicklung des automatisierten Fahrens im Eisenbahnsektor. Diese Systeme stützen sich auf verschiedene Sensoren, darunter Kameras, LiDAR, Radar und GNSS-Empfänger, um Daten über die Umgebung und die Bewegung des Zuges zu sammeln. Es werden verschiedene Perzeptionssysteme entwickelt, die im Eisenbahnsektor zur Unterstützung des automatisierten Fahrens eingesetzt werden sollen. Diese Systeme stützen sich auf Sensoren und Algorithmen, um die Umgebung wahrzunehmen und Entscheidungen zu treffen [5].

Einige der im Eisenbahnsektor häufig verwendeten Sensoren sind:

1. Kameras: Hochauflösende Kameras werden eingesetzt, um Bilder der Umgebung zu erfassen und Hindernisse oder Veränderungen im Gleis zu erkennen. Sie können auch zur Objekterkennung und -klassifizierung eingesetzt werden.

2. LiDAR: LiDAR verwenden Laserimpulse, um Entfernungen zu messen und 3D-Karten der Umgebung zu erstellen. Sie sind nützlich für die Erkennung von Hindernissen und können auch bei schlechten Lichtverhältnissen eingesetzt werden.

3. Radar: Es ist eine weitere Sensortechnologie, die in Perzeptionssystemen für Eisenbahnen eingesetzt wird. Es nutzt Funkwellen, um Objekte in der Umgebung zu erkennen und zu lokalisieren, und kann zur Messung von Entfernung und Geschwindigkeit eingesetzt werden. Radargeräte sind besonders nützlich in Situationen mit schlechten Sichtverhältnissen wie Nebel, Regen oder Schnee und können auch Hindernisse erkennen, die für das menschliche Auge nicht sichtbar sind. Sie sind ein wichtiger Bestandteil von Eisenbahnsicherheitssystemen und tragen dazu bei, Kollisionen zu vermeiden und die Gesamtleistung der Züge zu verbessern.

4. GNSS-Empfänger: Diese Empfänger nutzen Satellitensignale, um die Position und Geschwindigkeit des Zuges zu bestimmen. Sie werden zur Gleisüberwachung und zur Erkennung möglicher Kollisionen eingesetzt.

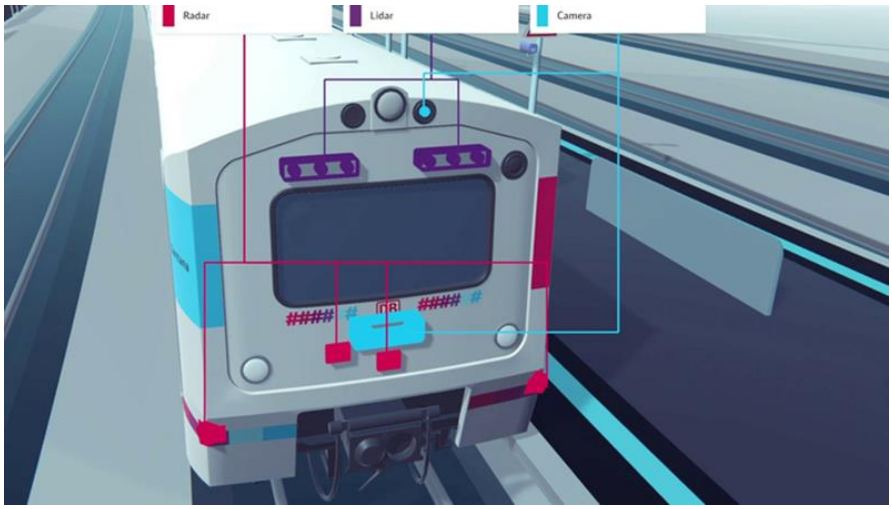


Abbildung 3: Perzeptionssystemen im Eisenbahnsektor [5]

Die Leistung dieser Sensoren kann je nach der Umgebung, in der sie eingesetzt werden, variieren. So können Kameras bei schlechten Lichtverhältnissen oder bei starkem Regen oder Nebel Probleme haben. LiDARs können in staubigen Umgebungen oder bei starkem Regen oder Schnee Probleme haben. GNSS-Empfänger haben Probleme in Tunneln, unter Brücken und in städtischen Umgebungen, wo hohe Gebäude die Satellitensignale blockieren.

Daher müssen stets verschiedene Sensoren gleichzeitig verwendet werden, die sich bei unterschiedlichen Umweltbedingungen gegenseitig ergänzen. Ein robustes und zuverlässiges Erkennungssystem sollte in der Lage sein, in verschiedenen Umgebungen gute Leistungen zu erbringen und die Herausforderungen zu meistern, die jede Umgebung mit sich bringt.

Die vom mobilen Perzeptionssystems gesammelten Daten werden normalerweise an Bord des Schienenfahrzeugs und zur Verbesserung der Gesamtleistung des Bahnsystems verwendet werden können.

## 2.3 Bestehende Methoden zur Erfassung und Speicherung von Trainingsdaten

Die Erfassung und Speicherung von Trainingsdaten ist für die Entwicklung und Erprobung von Perzeptionssystemen für das automatisierte Fahren im Eisenbahnsektor von entscheidender Bedeutung. Es gibt mehrere Methoden zur Sammlung und Speicherung von Trainingsdaten für Perzeptionssysteme im Eisenbahnsektor. Eine gängige Methode ist die Ausrüstung der Züge mit Sensoren und die Verwendung von Simulationen zur Datengenerierung. Diese Methoden haben ihre eigenen Vorteile, die bei der Entwicklung und dem Testen von Wahrnehmungsalgorithmen für das automatisierte Fahren von Nutzen sein können.

Das Sammeln von Daten vor Ort durch Sensoren liefert Daten aus der realen Welt, die für die Umgebung und die Bewegung des Zuges repräsentativ sind. Diese Methode ist vorteilhaft, da die Daten unter den tatsächlichen Betriebsbedingungen des Zuges erfasst werden, was eine genauere Entwicklung und Prüfung von Algorithmen ermöglicht. Allerdings kann diese Methode teuer und zeitaufwändig sein,

und es ist möglicherweise nicht möglich, alle für das Training und Testen von Wahrnehmungsalgorithmen erforderlichen Daten zu sammeln [5].

Simulationen hingegen können schnell und kostengünstig große Datenmengen erzeugen. Diese Methode ist vorteilhaft, da sie eine große Menge an Daten liefert, die zum Trainieren von Algorithmen unter verschiedenen Bedingungen verwendet werden können. Allerdings spiegeln Simulationen die realen Bedingungen möglicherweise nicht genau wider, so dass Algorithmen nicht so effektiv trainiert werden können, dass sie in allen Situationen funktionieren [6].

Für die Speicherung von Daten ist ein Ringspeicher eine gängige Methode, die sicherstellt, dass immer die neuesten Daten verfügbar sind. Der Ringspeicher speichert Daten in einem Ringspeicher und überschreibt alte Daten, wenn der Speicher voll ist. Dieser Ansatz ist vorteilhaft, weil er sicherstellt, dass die aktuellen Daten für die Analyse zur Verfügung stehen. Es besteht jedoch die Gefahr, dass wichtige Daten verloren gehen, wenn sie vor dem Überschreiben nicht gesichert werden.

Die Speicherung von Daten in einer Datenbank bietet mehr Flexibilität beim Zugriff und bei der Analyse der Daten, erfordert jedoch mehr Speicherkapazität und kann langsamer sein als ein Ringspeicher. Diese Methode ist vorteilhaft, weil sie eine anspruchsvollere Datenanalyse und die Speicherung eines größeren Datenvolumens ermöglicht [5].

Zusammenfassend lässt sich sagen, dass die Wahl der Methode zur Erfassung und Speicherung von Trainingsdaten im Eisenbahnsektor von den spezifischen Anforderungen des Projekts abhängt. Die Erfassung von Daten im Feld durch Sensoren liefert reale Daten, die für die Umgebung und die Bewegung des Zuges repräsentativ sind, während Simulationen schnell und kostengünstig große Datenmengen erzeugen können. Die Speicherung von Daten in einem Ringspeicher stellt sicher, dass immer die neuesten Daten verfügbar sind, während die Speicherung von Daten in einer Datenbank mehr Flexibilität beim Zugriff auf die Daten und deren Analyse bietet.

## 2.4 Bewertung der Grenzen der Bestehende Methoden

Die derzeitigen Ansätze zur Erfassung und Speicherung von Trainingsdaten haben zwar ihre Vorteile, weisen aber auch einige Einschränkungen auf, die berücksichtigt werden müssen. Das Verständnis dieser Einschränkungen ist entscheidend für die Gewährleistung der Genauigkeit und Zuverlässigkeit von Perzeptionssystemen für das automatisierte Fahren im Eisenbahnsektor.

Eine Einschränkung bei der Datenerfassung im Feld ist, dass sie zeitaufwändig und teuer sein kann. Die Ausrüstung von Zügen mit Sensoren wie Kameras, LiDAR, RADAR und GNSS-Empfängern kann kostspielig sein, und das Sammeln von Daten über ein breites Spektrum von Bedingungen und Umgebungen kann sehr viel Zeit in Anspruch nehmen. Dies kann die Menge der gesammelten Daten begrenzen, die für das Training und Testen von Wahrnehmungsalgorithmen unzureichend sein kann. Darüber hinaus kann die Datenerfassung im Feld durch die Verfügbarkeit geeigneter Umgebungen und Bedingungen für die Tests eingeschränkt sein, was den Datenerfassungsprozess weiter einschränkt.

Eine weitere Einschränkung bei der Datenerfassung im Feld besteht darin, dass die Daten möglicherweise nicht für alle möglichen Szenarien repräsentativ sind, denen ein Perzeptionssystem begegnen kann. Die Umgebung und die Bewegung des Zuges können stark variieren, und die Erfassung von Daten, die diese Variabilität genau widerspiegeln, kann schwierig oder unmöglich sein. Dies kann die Fähigkeit des Perzeptionssystems einschränken, neue und unvorhergesehene Situationen zu erkennen und darauf zu reagieren.

Mit Hilfe von Simulationen lassen sich schnell und kostengünstig große Datenmengen erzeugen, aber auch sie haben ihre Grenzen. Eine Einschränkung besteht darin, dass die erzeugten Daten möglicherweise nicht genau die realen Bedingungen widerspiegeln. Simulationen können durch die Genauigkeit der Modelle, die zur Simulation der Umgebung und der Bewegung des Zuges verwendet werden, begrenzt sein, was zu Fehlern in den erzeugten Daten führen kann. Darüber hinaus können die Szenarien, die simuliert werden können, durch die verfügbaren Rechenressourcen begrenzt sein, was den Umfang und die Vielfalt der Daten, die generiert werden können, einschränken kann [7].

Die Grenzen der derzeitigen Methoden zur Speicherung von Trainingsdaten müssen ebenfalls berücksichtigt werden. Die Verwendung eines Ringspeichers kann zum Verlust wichtiger Daten führen, wenn diese vor dem Überschreiben nicht gesichert werden, was die Fähigkeit zur Analyse und zum Lernen aus den Daten einschränken kann. Darüber hinaus kann die Verwendung eines Ringspeichers auch die Menge der speicherbaren Daten begrenzen, was die Fähigkeit des Perzeptionssystems, zu lernen und sich im Laufe der Zeit zu verbessern, weiter einschränken kann.

Die Speicherung von Daten in einer Datenbank bietet mehr Flexibilität beim Zugriff und bei der Analyse der Daten, bringt aber auch Einschränkungen mit sich. Eine Einschränkung ist die für die Speicherung großer Datenmengen erforderliche Speicherkapazität, die bei Projekten mit umfangreichen Datenanforderungen erhebliche Kosten verursachen kann. Eine weitere Einschränkung ist die Zeit, die für den Zugriff auf die Daten und deren Analyse benötigt wird, die langsamer sein kann als bei der Verwendung eines Ringspeichers. Dies kann die Fähigkeit des Perzeptionssystems einschränken, Daten in Echtzeit zu verarbeiten und darauf zu reagieren [8], [9].

Zusammenfassend lässt sich sagen, dass die derzeitigen Ansätze zur Erfassung und Speicherung von Trainingsdaten ihre Grenzen haben, die bei der Entwicklung und Erprobung von Perzeptionssystemen für das automatisierte Fahren im Bahnsektor berücksichtigt werden müssen. Um die Genauigkeit und Zuverlässigkeit dieser Systeme zu gewährleisten, ist es von entscheidender Bedeutung, die Vorteile und Grenzen jedes Ansatzes abzuwägen und die am besten geeignete Methode auf der Grundlage der spezifischen Bedürfnisse und Einschränkungen des Projekts auszuwählen. Letztendlich wird die Überwindung der Grenzen der derzeitigen Ansätze eine kontinuierliche Forschung und Entwicklung erfordern, um die Genauigkeit und Effizienz der Methoden zur Datenerfassung und -speicherung zu verbessern.

# 3 Konzept zur Erstellung eines Datenerfassungssystems mit ROS-Framework

## 3.1 Beschreibung des für die Datenerfassung verwendeten mobilen Perzeptionssystems

Während des Praktikums war es meine Aufgabe, den Aufbau eines mobilen Perzeptionssystems für die DB Systemtechnik im Perzeptionslabor zu unterstützen. Das mobile Perzeptionssystem ist ein komplexes Datenerfassungssystem. Das Ziel des Perzeptionssystems, das von DB Systemtechnik im Perzeptionslabor in Minden entwickelt wird, ist es, genaue Echtzeitdaten über die Umgebung des Zuges, einschließlich seiner Geschwindigkeit, Position und eventueller Hindernisse auf seinem Weg für die automatische Schadenserkennung oder die Entwicklung von Fahrerassistenzsystemen für das automatisierte Fahren zu liefern. Es verwendet mehrere Hardware-Geräte, darunter LiDAR, Kameras, Ethernet-Switches, eine Grandmaster-Uhr, einen Trigger-Timing-Controller, einen GNSS-Empfänger, einen LTE-Router, einen Feldrechner, eine LCD-Konsole, eine unterbrechungsfreie Stromversorgung und eine Festplattenstation, wie in Abbildung 4 dargestellt.

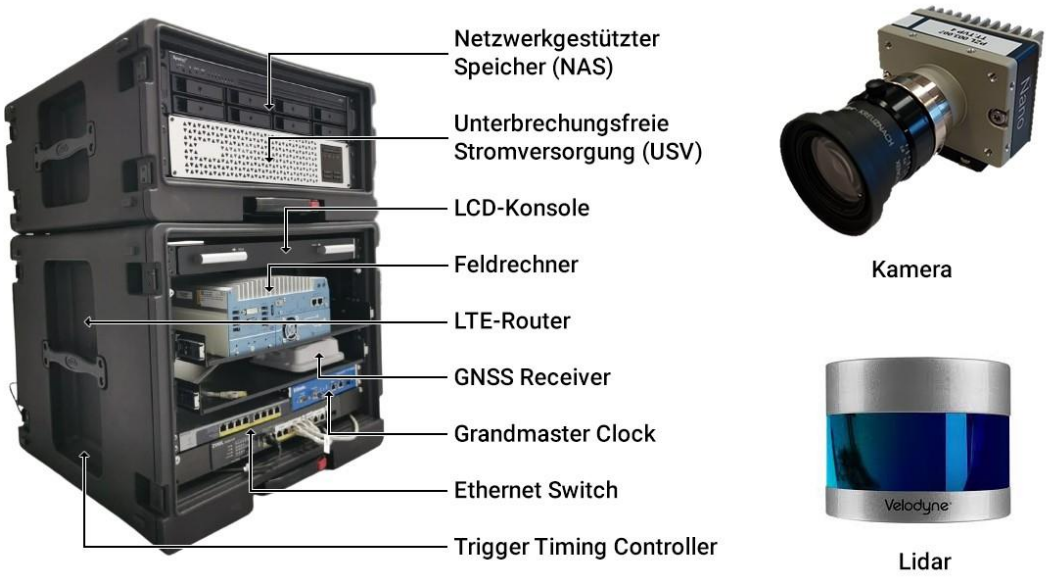


Abbildung 4: Mobiles Perzeption-System mit Liste der Komponenten

Die detaillierte Beschreibung jeder Komponente und ihrer Verwendung im Perzeptionssystem wurde im Praktikumsbericht beschrieben [10].

Insgesamt ist das mobile Perzeptionssystems ein komplexes Netzwerk von Geräten, die zusammenarbeiten, um Daten von mehreren Sensoren zu sammeln und zu verarbeiten. Diese Daten werden dann verwendet, um eine 3D-Karte der Umgebung zu erstellen, die für verschiedene Anwendungen wie autonomes Fahren, Kartierung und Überwachung genutzt werden kann.

Das in diesem Projekt eingesetzte mobile Perzeptionssystem ist einzigartig in seiner Kombination von Hardwarekomponenten und seinem spezifischen Anwendungsfall. Das System nutzt eine Vielzahl von LiDARs und Kameras für die Datenerfassung, die eine 360-Grad-Abdeckung ermöglichen und sowohl statische als auch dynamische Objekte in der Umgebung erkennen können. Das System umfasst auch eine Grandmaster-Uhr zur präzisen Zeitsynchronisation und einen Trigger-Timing-Controller für eine genaue Datenerfassung.

Abbildung 5: Perzeptionssystem im Feld [DB ST]

Darüber hinaus ist das System, wie in der Abbildung 5 zu sehen ist, äußerst mobil und kann problemlos in einen Zug eingebaut werden, um unterwegs Daten zu sammeln. Der Einsatz eines LTE-Routers ermöglicht den Fernzugriff auf das System, so dass die Daten in Echtzeit von einem zentralen Standort aus erfasst und analysiert werden können.

Insgesamt macht die Kombination aus Hardware-Komponenten, Mobilität und Echtzeit-Datenerfassung das System zu einem einzigartigen und leistungsstarken Werkzeug für eine Vielzahl von Anwendungen, wie z. B. die Entwicklung autonomer Fahrzeuge, Umweltüberwachung und vieles mehr.

## 3.2 Erläuterung des ROS-Frameworks zur Datenerfassung und -verarbeitung

Das Robot Operating System (ROS) ist ein Open-Source-Framework, das eine Sammlung von Softwarebibliotheken und Werkzeugen für den Bau und Betrieb von Robotersystemen bietet. Es sind mehrere ROS-Distributionen verfügbar, jede mit ihren eigenen Funktionen und Möglichkeiten. In dieser Arbeit wird das ROS-Framework (Noetic Distribution) für die Datenaufzeichnung und -verarbeitung verwendet, da ROS Noetic die neueste stabile Version von ROS ist, die im Mai 2020 veröffentlicht wurde[11]. Sie ist so konzipiert, dass sie mit Ubuntu 20.04, der neuesten Long-Term-Support-Version (LTS) von Ubuntu, kompatibel ist. Noetic unterstützt Python 3, was bedeutet, dass Benutzer ROS-Knoten mit Python 3 anstelle von Python 2 schreiben können. Noetic bietet außerdem mehrere neue Funktionen und Verbesserungen, wie die Unterstützung von ROS2-Schnittstellen, verbesserte Sicherheit und eine bessere Dokumentation [12].

ROS ist mehr als nur Middleware und die Verfügbarkeit von vielen Lösungen und Paketen für Roboternavigation, -perzeption, -steuerung, -bewegungsplanung, -simulation und mehr macht ROS unverzichtbar (Abbildung 6).

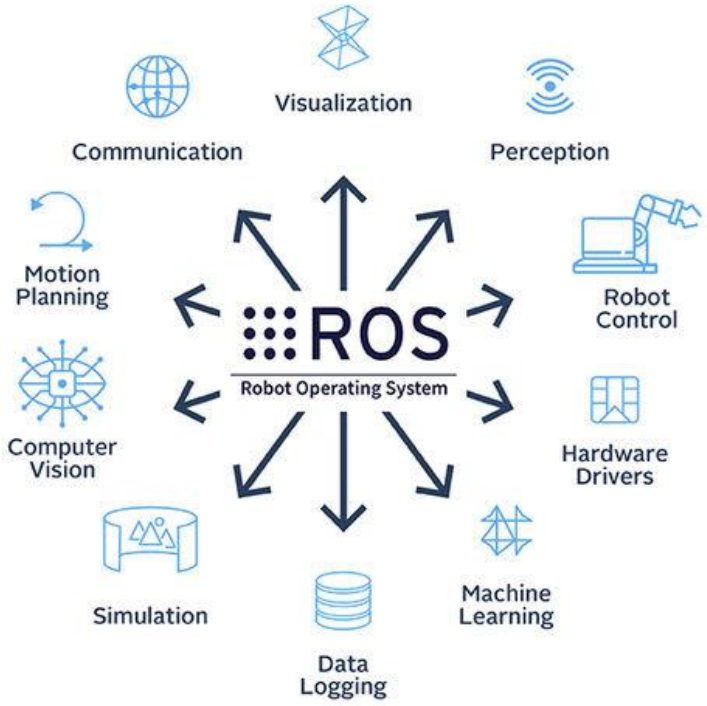


Abbildung 6: ROS-Leistungen

ROS stellt ein System von Knotenpunkten bereit, das den Austausch von Prozessen innerhalb der intelligenten Plattform des Ziels ermöglicht. Diese Interprozesse ermöglichen die gemeinsame Nutzung von funktionalen Nachrichten innerhalb einer Roboterarchitektur. Die Architektur eines ROS-Systems, wie die Abbildung 7 zeigt, besteht aus fünf Komponenten: einem „ROS-Master, Node, Publishers, Subscribers und Topics [13].

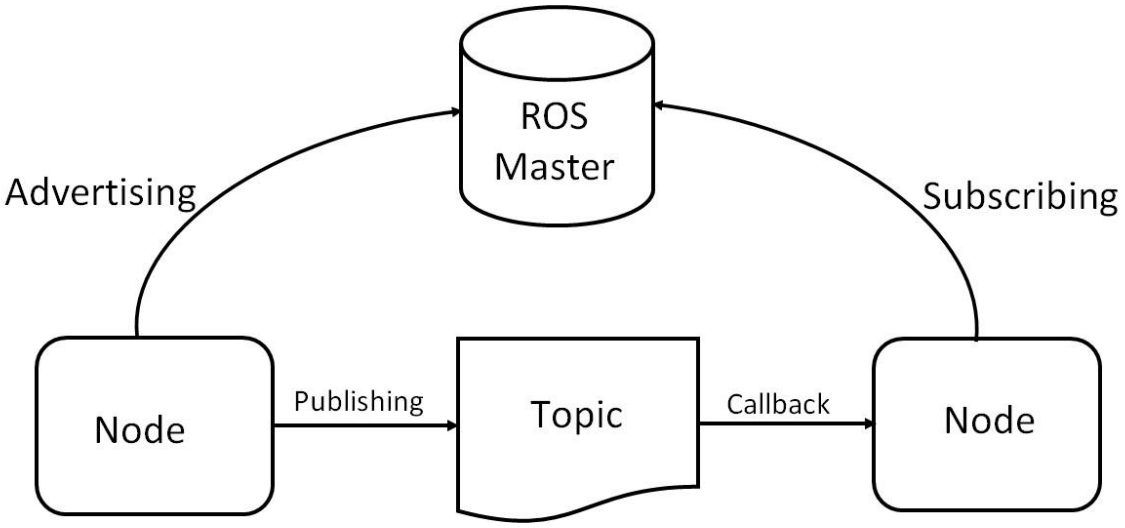


Abbildung 7: Komponenten des ROS-Systems [13]

Ein typisches ROS-Modell: Systemkomponenten.

ROS-Master: Der ROS-Master ist für die Verwaltung von Namen und Registrierungsdiensten für die Knoten innerhalb eines ROS-Systems verantwortlich. Verleger und Abonnenten werden vom ROS-Master überwacht, um sicherzustellen, dass die zugehörigen Themen sowie die Dienste innerhalb des Robotersystems bereitgestellt werden. Der ROS-Master ermöglicht auch die Lokalisierung und Kommunikation zwischen Knoten innerhalb des Robotersystems. Schließlich initiiert der ROS-Master in der Regel die Knotenkommunikationsfunktion mit dem Befehl Roscore. Der Roscore-Befehl wird verwendet, um den ROS-Master mit den wesentlichen Softwarekomponenten zu laden, die die Kommunikation zwischen den Knoten ermöglichen.

Node: Eine ausführbare Datei innerhalb des ROS-Systems, die die Kommunikation zwischen anderen Node ermöglicht.

Publisher: Eine Nachricht, die von einem Knoten oder Topic innerhalb eines ROS-Systems übertragen wird, wird als Publisher bezeichnet.

Subscriber: Eine Nachricht, die von einem Knoten oder einem Topic innerhalb eines ROS-Systems empfangen wird, wird als Subscriber bezeichnet.

Topics: Das Veröffentlichen und Abonnieren einer Nachricht eines bestimmten Namenstyps wird als Topic bezeichnet.

Jede dieser Softwarekomponenten ermöglicht es einem Robotersystem, eine Vielzahl von Signal- und Bilddaten zu bewegen, zu erfassen, zu überwachen und zu verarbeiten.

Das ROS-Framework bietet eine flexible und modulare Architektur, die die Integration verschiedener Sensoren, Algorithmen und Hardware-Geräte ermöglicht. Die ROS-Knoten, die die grundlegenden Bausteine von ROS-Anwendungen sind, sind für die Erfassung von Daten von den Sensoren und deren Veröffentlichung im ROS-Netzwerk verantwortlich. Die ROS-Themen, die Kommunikationskanäle im ROS-Netzwerk, ermöglichen den verschiedenen Knoten den Datenaustausch.

Für die Datenaufzeichnung wird das ROS-Bag-Format verwendet. Ein ROS-Bag ist ein Dateiformat, das die Speicherung und Wiedergabe von ROS-Nachrichten ermöglicht. Das ROS-Bag-Format bietet eine praktische Möglichkeit, Daten von mehreren Sensoren aufzuzeichnen und in einer einzigen Datei zu speichern. Das ROS-Bag-Format ermöglicht auch die selektive Wiedergabe bestimmter Themen, wodurch die Analyse und Verarbeitung der Daten erleichterten wird. Es handelt sich im Wesentlichen um ein Binärformat zur Speicherung von ROS-Meldungsdaten und Metadaten wie Meldungstyp, Themenname und Zeitstempel [14].

Einige der wichtigsten Merkmale von ROS-Bag-Dateien sind:

1. Flexibilität: ROS-Bag-Dateien können Nachrichten von mehreren Themen und sogar Nachrichten von mehreren Knoten gleichzeitig aufzeichnen. Diese Flexibilität ermöglicht komplexe Datenerfassungsszenarien.

2. Zeitstempel: ROS-Bag-Dateien zeichnen Zeitstempel für jede Nachricht auf, was für die Synchronisierung und Wiedergabe der aufgezeichneten Daten wichtig ist.

3. Komprimierung: ROS-Bag-Dateien können komprimiert werden, um den Speicherbedarf zu verringern. Dies ist besonders nützlich für die Speicherung großer Datenmengen.

4. Zufälliger Zugriff: ROS-Bag-Dateien unterstützen den wahlfreien Zugriff, d. h. bestimmte Nachrichten oder Abschnitte der Daten können leicht extrahiert und analysiert werden.

5. Portabilität: ROS-Bag-Dateien können leicht zwischen verschiedenen ROS-Installationen und verschiedenen Maschinen ausgetauscht werden, was sie zu einem praktischen Format für die Zusammenarbeit und den Datenaustausch macht.

Einige der wichtigsten Vorteile der Verwendung von ROS-Bag-Dateien im Zusammenhang mit der Datenerfassung und -verarbeitung sind:

1. Reproduzierbarkeit: Durch die Aufzeichnung von Daten in einer ROS-Bag-Datei können Forscher und Ingenieure Experimente oder Szenarien leicht reproduzieren, selbst Monate oder Jahre nach der Aufzeichnung der ursprünglichen Daten.

2. Analyse: Die Zufallszugriffs- und Komprimierungsfunktionen von ROS-Bag-Dateien erleichtern das Extrahieren und Analysieren bestimmter Teilmengen der Daten.

3. Fehlersuche: ROS-Bag-Dateien können zum Debuggen von Knoten oder Systemen verwendet werden, indem die aufgezeichneten Daten abgespielt und das Verhalten des Systems beobachtet wird.

4. Gemeinsame Nutzung: ROS-Bag-Dateien können leicht zwischen verschiedenen Forschern und Ingenieuren ausgetauscht werden, was die Zusammenarbeit und den Wissensaustausch erleichtern kann.

Für die Datenverarbeitung bietet das ROS-Framework eine Reihe von Werkzeugen und Bibliotheken, darunter das rosbag-Kommandozeilenwerkzeug und die rosbag-API. Diese Werkzeuge und Bibliotheken ermöglichen die Manipulation und Analyse von rosag-Dateien. Das rosbag-Kommandozeilen-Tool kann verwendet werden, um bestimmte Themen aus einer ROS-Bag-Datei zu extrahieren und in andere Formate, wie z. B. CSV (comma-separated values), zu konvertieren. Die rosbag-API ermöglicht die Erstellung von benutzerdefinierten Werkzeugen zur Verarbeitung und Analyse von ROS-Bag-Dateien.

In dieser Arbeit werden Rosbag-Daten als primäre Trainingsdatenquelle für das mobile Perzeptionssystem verwendet. Rosbag bietet eine bequeme Möglichkeit, Daten von mehreren Sensoren und anderen Quellen zu erfassen und diese Daten für Tests und Analysen wiederzugeben.

Das in dieser Arbeit verwendete mobile Perzeptionssystem besteht aus mehreren Sensoren, einschließlich LiDAR und Kamera, die synchronisiert und im Rosbag-Format aufgezeichnet werden. Die aufgezeichneten Daten werden dann verarbeitet und zum Trainieren eines maschinellen Lernalgorithmus verwendet, damit das System Objekte in der Umgebung erkennen und klassifizieren kann.

Zusammenfassend lässt sich sagen, dass das ROS-Framework eine flexible und modulare Architektur für die Datenerfassung und -verarbeitung bietet. Die ROS-Knoten und -Themen ermöglichen die Integration verschiedener Sensoren und Hardwaregeräte, während das ROS-Bag-Format eine bequeme Möglichkeit zur

Speicherung und Wiedergabe der Daten bietet. Die ROS-Tools und -Bibliotheken ermöglichen die Manipulation und Analyse der Daten und erleichtern so die Gewinnung von Erkenntnissen und die Entwicklung von Algorithmen für das automatisierte Fahren im Eisenbahnsektor.

## 3.3 Detaillierte Beschreibung des vorgeschlagenen Ansatzes

Das Versuch besteht aus mehreren Komponenten, wie im Abbildung 8 dargestellt. Ein Arduino Uno und ein Ethernet Shield sind mit einer LED, einem Encoder und einem Knopf verbunden. Der Arduino Uno und das Ethernet-Shield sind dann mit einem PC verbunden, auf dem auch der ROS- und Ringspeicher-Code läuft. Alle diese Komponenten sind so integriert und konfiguriert, dass sie die Anforderungen des vorgeschlagenen Ansatzes zur Datenerfassung unter ROS erfüllen.

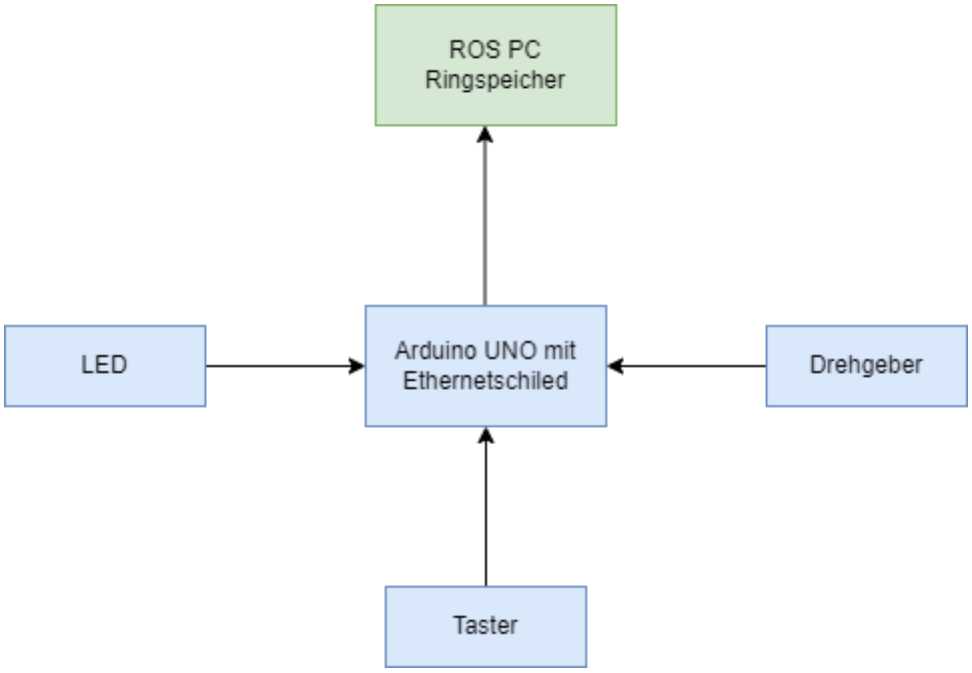


Abbildung 8: Blockdiagramm des Versuchsaufbaus mit allen Komponenten

### 3.3.1 Taster

Der erste Ansatz zur Erreichung des Ziels besteht darin, einen Taster-Mechanismus in das Perzeptionssystems zu implementieren, der vom Triebfahrzeugführer während der Fahrt gesteuert werden kann. Dieser Mechanismus ermöglicht es dem Triebfahrzeugführer, relevante Ereignisse (Trainingsdaten für die automatische

Schadenserkennung oder für die Entwicklung von Fahrerassistenzsystemen für das automatisierte Fahren, z.B. große Tiere oder Personen auf der Gleise), die während der Fahrt auftreten, zu markieren und aufzuzeichnen. Dieser Mechanismus besteht aus einem kompakten (Abbildung 9) Controller mit einem Taster und einer LED-Lampe, der mit ROS kommunizieren kann, eine Reihe von Eingangs- und Ausgangspins hat, eine niedrige Stromversorgung, einen Ethernet-Anschluss und wenn möglich einen POE benötigt. Die Lampe wird verwendet, um anzuzeigen, dass der Fahrer die Taste richtig gedrückt hat und das entsprechende Ereignis zu markieren. Sobald ein Ereignis markiert ist, speichert das Perzeptionssystems die relevanten Daten (z.B. 3 relevante Rosbag-Dateien, eine vor dem Ereignis, eine während des Ereignisses und eine nach dem Ereignis) in einem permanenten Speicher aus einem Ringspeicherspeicher.

Abbildung 9: Taster

### 3.3.2 Ringspeicher

Der nächste Schritt ist die Implementierung eines Ringspeicherkonzepts für die Speicherung der aufgezeichneten Ereignisse. Ein Ringspeicher ist eine Datenstruktur, die für die effiziente Speicherung und den Zugriff auf Daten in einem mobilen Perzeptionssystem verwendet wird. Ein Ringspeicher ist ein Speicher fester Größe, der als zirkuläre Datenstruktur behandelt wird, so dass das System nur eine feste Datenmenge speichern kann und neue Daten die ältesten Daten überschreiben, wenn der Speicher voll ist [15].

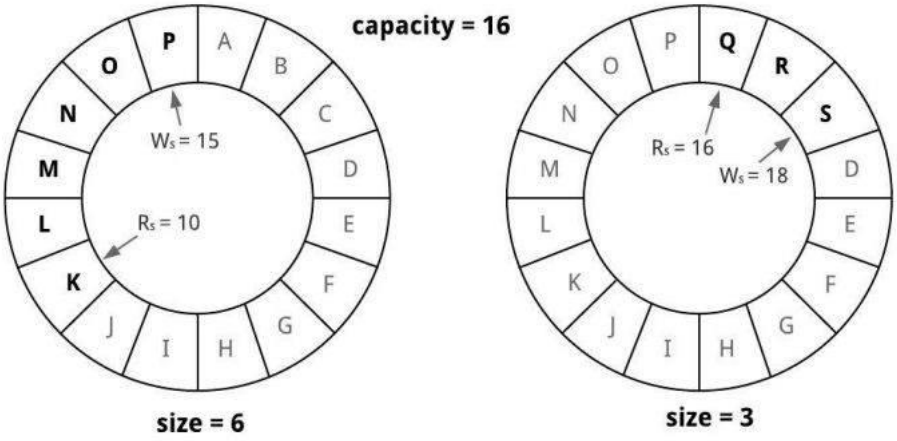


Abbildung 10: Ringspeicher [16]

Der Ringspeicher ist in zwei Teile unterteilt: den Head und den Tail, wie in Abbildung 10 gezeigt. Der Head steht für den Index des nächsten Elements, das dem Speicher hinzugefügt wird, während der Tail den Index des ältesten Elements im Speicher darstellt.

Der Vorteil der Verwendung eines Ringspeichers im mobilen Perzeptionssystems besteht darin, dass er eine effiziente und kontinuierliche Datenaufzeichnung ermöglicht. Die Daten werden kontinuierlich zirkulär in den Speicher geschrieben, was bedeutet, dass der Speicher nie in seiner Größe verändert oder neu zugewiesen werden muss. Dadurch wird sichergestellt, dass das System Daten ohne Unterbrechungen oder Verzögerungen aufzeichnen kann. Außerdem ermöglicht der Ringspeicher einen einfachen und schnellen Zugriff auf die Daten, da sie in einem zusammenhängenden Speicherblock gespeichert sind. Dadurch wird die Verarbeitung und Analyse der Daten erleichtert. Insgesamt trägt die Verwendung eines Ringspeichers im mobilen Perzeptionssystems dazu bei, dass die Daten effizient und zuverlässig aufgezeichnet werden und für die weitere Analyse einfach abgerufen und verarbeitet werden können.

Hier wird der Ringspeicher so implementiert, dass, wenn der Triebfahrzeugführer die Taster drückt, um das entsprechende Ereignis zu markieren und zu speichern, die spezifische relevante Aufzeichnung ROS-Bag-Datei aus dem Ringspeicher in einem permanenten Speicher abgelegt wird. Für ein relevantes Ereignis ist es auch wichtig,

die vorherige und zukünftige Aufzeichnung des Ereignisses zu speichern. Bei dieser Implementierung werden beim Drücken der Taste drei relevante ROS-Bag-Datei gespeichert (eine vor dem Ereignis, eine während des Ereignisses und eine nach dem Ereignis). Die Umsetzung dieses Konzepts wird helfen, die Menge der aufgezeichneten Daten zu verwalten und den Verlust relevanter Daten zu verhindern.

### 3.3.3 Drehgeber

Schließlich sollen Beginn und Ende der Datenaufzeichnung im Ringspeicher automatisch und fahrtrichtungsabhängig erfolgen. Dies kann durch den Einsatz eines Drehgebers erreicht werden. Ein Drehgeber (Abbildung 11) ist ein Gerät, das zur Messung der Drehposition und der Winkelverschiebung einer Welle verwendet wird. Er wandelt die Winkelposition der Welle in ein elektrisches Signal um, das von einer Steuerung oder einem Prozessor verwendet werden kann.

Der gebräuchlichste Typ von Inkrementalgebern ist der Quadraturgeber, der aus zwei Kanälen besteht, die elektrische Signale erzeugen, wenn sich die Welle dreht. Die Signale dieser Kanäle sind um 90 Grad verschoben, so dass sie zueinander phasenverschoben sind. Durch die Überwachung des Timings und der Sequenz dieser Signale kann die Steuerung die Richtung und Größe der Wellendrehung bestimmen [17].


Abbildung 11: Drehgeber [18]

Neben der Messung der Wellenposition können Drehgeber auch zur Messung von Geschwindigkeit und Beschleunigung verwendet und mit anderen Sensoren und Steuersystemen integriert werden, um eine präzise Rückmeldung und Steuerung in einer Vielzahl von Anwendungen, wie z. B. in der Robotik und Automatisierung, zu ermöglichen.

Hier ist der Drehgeber so aufgebaut, dass er Richtung eines Zuges bestimmen kann, indem er die Drehung eines Rades oder einer Achse überwacht.

Um das Signal des Drehgebers in das ROS-Framework zu integrieren, wird das Signal zunächst von dem an den Drehgeber angeschlossenen Mikrocontroller Arduino Uno verarbeitet. Der Arduino uno liest das vom Drehgeber erzeugte elektrische Signal und wandelt es in ein digitales Signal um, das vom ROS-Framework verstanden werden kann. Dies geschieht mit Hilfe der ROS-Bibliotheken, die für die Arduino uno-Plattform verfügbar sind, wie z. B. die rosserial_arduino-Bibliothek.

Sobald das Signal in ein digitales Format übersetzt wurde, kann es in einem ROS-Topic veröffentlicht werden. Dieses Topic kann von anderen Knoten im ROS-Netzwerk abonniert werden, so dass sie das Encoder-Signal empfangen und verarbeiten können. Später könnte das Encoder-Signal verwendet werden, um den Start und das Ende der Datenaufzeichnung im Ringspeicher auszulösen.

Der Drehgeber wurde so eingestellt, dass er die Bewegungsrichtung des Zuges erkennt, indem er die Drehung seines Rades oder seiner Achse überwacht. Wenn sich der Zug vorwärtsbewegt, sendet der Drehgeber ein Vorwärtssignal an den Arduino, der die Informationen an das ROS-Framework weiterleitet. Dies löst den Start der Rosbag-Aufzeichnung im Ringspeicher aus. Falls der Encoder eine Rückwärtsbewegung erkennt, liest der Arduino ein Rückwärtssignal und stoppt die Aufzeichnung. Bleibt der Zug für eine bestimmte Dauer (z. B. 10 Minuten) stehen, liest der Arduino ein Stoppsignal vom Encoder, wodurch die Aufzeichnung des Rosbags im Ringspeicher ebenfalls gestoppt wird.

# 4 Implementierung des Datenerfassungssystems mit dem ROS-Framework

## 4.1 Überblick des vorgeschlagenen Ansatzes auf dem mobilen Perzeptionssystem

Die Umsetzung des vorgeschlagenen Ansatzes auf dem mobilen Perzeptionssystem erfolgte in mehreren Schritten. Zunächst wurden die erforderlichen Hardwarekomponenten zusammengestellt, darunter Sensoren (wie LIDAR und Kamera) und eine Recheneinheit (wie ein Industriecomputer). Die Sensoren wurden auf dem mobilen Perzeptionssystem montiert, und die Recheneinheit wurde mit den Sensoren und der Plattform verbunden.

Anschließend wurde das Robot Operating System (ROS) auf dem Computer installiert. Der vorgeschlagene Ansatz wurde als ROS-Paket implementiert, das Knotenpunkte für die Datenerfassung, -verarbeitung und -speicherung enthält.

Der Auslöser für die Datenerfassung wurde über einen Taster implementiert, die mit dem Perzeptionssystem verbunden waren. Wenn der Taster gedrückt, begann der Datenerfassungsknoten, Daten von den Sensoren zu sammeln. Die Daten wurden vom Perzeptionsknoten verarbeitet, der verschiedene Algorithmen anwendete, um relevante Merkmale aus den Sensordaten zu extrahieren. Der Ringspeicher wurde verwendet, um die relevanten Daten für einen bestimmten Zeitraum zu speichern, der je nach den Anforderungen der Anwendung angepasst werden konnte.

Der Encoder wurde zum Starten und Beenden der ROS-Bag-Aufzeichnung im Ringspeicher verwendet. Dazu wurde ein Signal vom Encoder an den Datenerfassungsknoten gesendet, wenn die Aufzeichnung gestartet und gestoppt wurde. Die Daten zwischen dem Start- und dem Stoppsignal wurden dann im Ringspeicher gespeichert.

## 4.2 Technische Details zu den Komponenten

### 4.2.1 Der Taster

Um im Perzeptionssystems einen Mechanismus zur Markierung relevanter Daten zu implementieren, muss ein Taster mit einem Controller implementiert werden. Zu diesem Zweck muss ein Controller einige spezifische Anforderungen erfüllen.

kompakt: Der Controller sollte einen kleinen Formfaktor haben, um in enge Räume zu passen, was ihn ideal für den Einsatz in einem Perzeptionssystems macht. so dass er die Tasterbox sehr flexibel und einfach zu handhaben macht.

Ethernet-Anschluss: Der Controller sollte über einen Ethernet-Anschluss verfügen, um die Kommunikation mit anderen Geräten im Perzeptionssystem zu ermöglichen (z.B. um die Tasternachricht an einen hinterlegten Computer im Perzeptionssystem zu senden).

Stromsparende Anwendung: Der Controller sollte einen geringen Stromverbrauch haben, um einen effizienten Betrieb zu gewährleisten und Überhitzung oder strombezogene Probleme zu vermeiden.

Stromversorgung vorzugsweise über POE: Der Controller sollte die Möglichkeit haben, über Power-over-Ethernet (POE) mit Strom versorgt zu werden, um die Einrichtung der Stromversorgung zu vereinfachen und die Notwendigkeit einer zusätzlichen Stromquelle für schwer zugängliche Orte mit begrenzten Stromquellen zu eliminieren.

Kommunikation mit ROS: Die Steuerung sollte mit dem Roboterbetriebssystem (ROS) kompatibel sein.

Anzahl der Eingangs-/Ausgangsstifte: Die Steuerung sollte über eine ausreichende Anzahl von Eingangs-/Ausgangsstiften verfügen, um alle erforderlichen Geräte des Perzeptionssystems anschließen zu können.

Je nach den Anforderungen an den Controller gab es 2 Optionen, ESP32-PoE-ISO und Arduino UNO mit Ethernet-Shield.

Die erste Wahl für die Einrichtung war der ESP32-PoE-ISO (Abbildung 12). Der ESP32-PoE-ISO ist ein kompakter Controller mit einem ESP32-WROOM-32-Modul, Ethernet-Konnektivität und Power-over-Ethernet (PoE) Unterstützung. Er wurde entwickelt, um Low-Power- und High-Performance-Computing-Funktionen für Embedded-Systeme bereitzustellen. Der Controller hat eine integrierte 10/100 Mbps Ethernet-Schnittstelle, die die Kommunikation mit dem ROS-Framework über ein kabelgebundenes Netzwerk ermöglicht. Er verfügt über mehrere Eingangs- und Ausgangspins, die als Schnittstelle zu verschiedenen Sensoren und Geräten verwendet werden können. [19]


Abbildung 12: ESP32-PoE-ISO [19]

Es gibt jedoch mehrere Gründe, warum der ESP32-PoE-ISO nicht mit ROS kompatibel ist, da er die Kommunikation mit ROS während des Betriebs verliert. [20]Einige mögliche Gründe sind:

1. Mangel an Dokumentation und Ressourcen: ESP32-PoE-ISO ist eine relativ neue Plattform und es gibt nur eine begrenzte Menge an ROS-bezogener Dokumentation und Ressourcen für sie. Aufgrund des Mangels an Ressourcen muss der Code für die Kopplung von ESP32-PoE-ISO mit ROS möglicherweise von Grund auf neu entwickelt werden, was für diejenigen, die neu in ROS sind, zeitaufwändig und schwierig sein kann.

2. Inkompatibilität mit ROS: Obwohl der ESP32-PoE-ISO ein leistungsfähiger Mikrocontroller mit integrierter Ethernet- und PoE-Unterstützung ist, ist er möglicherweise nicht vollständig mit ROS kompatibel. ROS hat spezifische Anforderungen für Kommunikationsprotokolle und Datenformate, und es ist möglich, dass der ESP32-PoE-ISO nicht vollständig mit diesen Anforderungen kompatibel ist, was zu Kommunikationsproblemen führt.

3. Probleme mit der Stromversorgung: Der ESP32-PoE-ISO benötigt PoE-Unterstützung für die Stromversorgung, was bedeutet, dass er über das Ethernet-Kabel mit Strom versorgt wird. Wenn die Stromversorgung jedoch nicht stabil ist oder Schwankungen in der Stromversorgung auftreten, kann dies zu Kommunikationsproblemen und sogar zu Datenverlusten führen.

4. Hardware-Einschränkungen: Obwohl der ESP32-PoE-ISO ein leistungsfähiger Mikrocontroller ist, verfügt er möglicherweise nicht über die notwendigen Hardware-Ressourcen, um die Anforderungen des Perzeptionssystems zu erfüllen. Wenn das Perzeptionssystem beispielsweise eine große Datenmenge erzeugt, kann der ESP32-PoE-ISO diese möglicherweise nicht effizient verarbeiten, was zu Kommunikationsproblemen und Datenverlusten führt.

Ausgehend von diesen möglichen Gründen ist es möglich, dass der ESP32-PoE-ISO die Kommunikation mit ROS aufgrund einer Kombination aus Inkompatibilitätsproblemen, Problemen mit der Stromversorgung und Hardwarebeschränkungen verloren hat.

Abbildung 13: Arduino uno mit Ethernet Shield [21]

Die nächste bevorzugte Hardware ist der Arduino uno mit Ethernet Shield (Abbildung 13). Der Arduino UNO mit Ethernet Shield ist ein Mikrocontroller-Board, das auf dem Mikrocontroller ATmega328P basiert. Das Ethernet Shield fügt dem Board Ethernet-

Konnektivität hinzu, so dass es mit dem ROS-Framework über ein kabelgebundenes Netzwerk kommunizieren kann [22]. Es hat ein kompaktes Design und verfügt über mehrere Eingangs- und Ausgangspins, die für den Anschluss verschiedener Sensoren und Geräte verwendet werden können. Darüber hinaus unterstützt es einen niedrigen Stromverbrauch und ist mit dem ROS-Framework kompatibel, was es zu einer idealen Wahl für die Implementierung eines tastenbasierten Mechanismus zur Markierung relevanter Daten während der Datenerfassung macht.

Im Hinblick auf die Kompatibilität mit ROS und die Zuverlässigkeit wurde der Arduino UNO mit Ethernet Shield dem ESP32-PoE-ISO für das Perzeptionssystem vorgezogen, da der Arduino uno mit Ethernet Shield aus mehreren Gründen besser mit ROS kompatibel ist:

Unterstützung für ROS-Bibliotheken: Die Arduino-Plattform verfügt über eine solide Unterstützung für ROS-Bibliotheken, die eine einfachere Integration des Arduino in ROS ermöglichen. Es gibt zahlreiche Bibliotheken für die Arduino-Plattform, die die Kommunikation mit ROS ermöglichen und die Verwendung des Arduino in ROS-Anwendungen erleichtern. Einige der am häufigsten verwendeten Bibliotheken sind:

rosserial_arduino: Diese Bibliothek wird für die Kommunikation zwischen einem Arduino-Board und einem ROS-System über die serielle Schnittstelle verwendet. Sie ermöglicht es dem Arduino-Board, als ROS-Knoten zu agieren und Nachrichten vom ROS-System zu senden und zu empfangen [22].

ros_lib: Diese Bibliothek enthält Nachrichten-Definitionen für verschiedene ROS-Nachrichten, die für die Kommunikation zwischen dem Arduino-Board und dem ROS-System verwendet werden können [22].

Arduino-ROS: Diese Bibliothek ist eine Implementierung von ROS für die Arduino-Plattform, die es dem Arduino-Board ermöglicht, mit anderen ROS-Knoten über ein Netzwerk zu kommunizieren.

ros_arduino_bridge: Diese Bibliothek stellt eine Brücke zwischen ROS und der Arduino-Plattform dar, so dass das Arduino-Board als ROS-Knoten fungieren und mit anderen ROS-Knoten kommunizieren kann [23].

Diese Bibliothek erleichtert die Integration der Arduino-Plattform mit ROS und bietet eine effiziente Möglichkeit zur Kommunikation zwischen den beiden Systemen. In dem vorliegenden Entwicklungsprojekt wurden die Bibliotheken Ros_lib und Rosserial_arduino für die Kommunikation zwischen dem Arduino UNO mit Ethernet Shield und dem ROS-System verwendet.

Ros_lib und Rosserial_Arduino sind zwei beliebte Bibliotheken für Arduino, die die Kommunikation mit ROS (Robot Operating System) ermöglichen. Ros_lib ist eine Bibliothek, die Code zum Erstellen von ROS-Nachrichten und -Diensten enthält. Sie bietet eine einfache Möglichkeit, Nachrichtenheader und Quelldateien für benutzerdefinierte Nachrichten zu generieren, was die Erstellung benutzerdefinierter Nachrichtentypen, die mit ROS verwendet werden können, erleichtert. Mit Ros_lib können Sie benutzerdefinierte Nachrichtentypen für bestimmte Sensoren oder Aktoren erstellen und diese in Ihrer ROS-Anwendung verwenden.

Rosserial_Arduino ist eine Bibliothek, die die serielle Kommunikation zwischen einem Arduino-Board und einem Computer mit ROS ermöglicht. Sie bietet eine einfache Möglichkeit, eine serielle Verbindung zwischen den beiden Geräten herzustellen und ROS-Nachrichten über diese Verbindung zu senden. Mit Rosserial_Arduino können Sie Code auf einem Arduino-Board schreiben, um Nachrichten in einem ROS-Netzwerk zu empfangen und zu senden. Diese Bibliothek macht es einfacher, Arduino-basierte Geräte in ein ROS-Ökosystem zu integrieren.

Beide Bibliotheken sind in der ROS-Community weit verbreitet und bieten eine einfache Möglichkeit, Arduino-basierte Geräte mit ROS zu verbinden. Mit Ros_lib können Sie eigene Nachrichtentypen definieren und Rosserial_Arduino ermöglicht die serielle Kommunikation zwischen Arduino-Boards und ROS.

Kompatibilität mit dem ROS-Netzwerk: Der Arduino mit Ethernet Shield ist besser mit dem ROS-Netzwerk kompatibel. Es kann direkt mit dem Ethernet-Port des Computers oder des Routers verbunden werden und bietet eine zuverlässige und stabile Verbindung. Dies ermöglicht eine bessere Kommunikation mit ROS-Knoten und verringert die Wahrscheinlichkeit von Kommunikationsproblemen.

Open-Source-Plattform: Arduino ist eine Open-Source-Plattform, und der Code ist für Änderungen und die Integration mit ROS leicht verfügbar. So können Entwickler den

Code anpassen und verändern, um die spezifischen Anforderungen der ROS-Anwendung zu erfüllen.

Verfügbarkeit von Community-Unterstützung: Arduino hat eine große und aktive Gemeinschaft von Entwicklern, die Unterstützung und Anleitung für die ROS-Integration bieten können. Dies kann bei der Behebung von Problemen und der Entwicklung neuer Anwendungen hilfreich sein.

Insgesamt ist die Arduino-Plattform mit Ethernet Shield aufgrund ihrer Benutzerfreundlichkeit, Kompatibilität und Verfügbarkeit von Support eine beliebte und zuverlässige Wahl für ROS-Anwendungen.

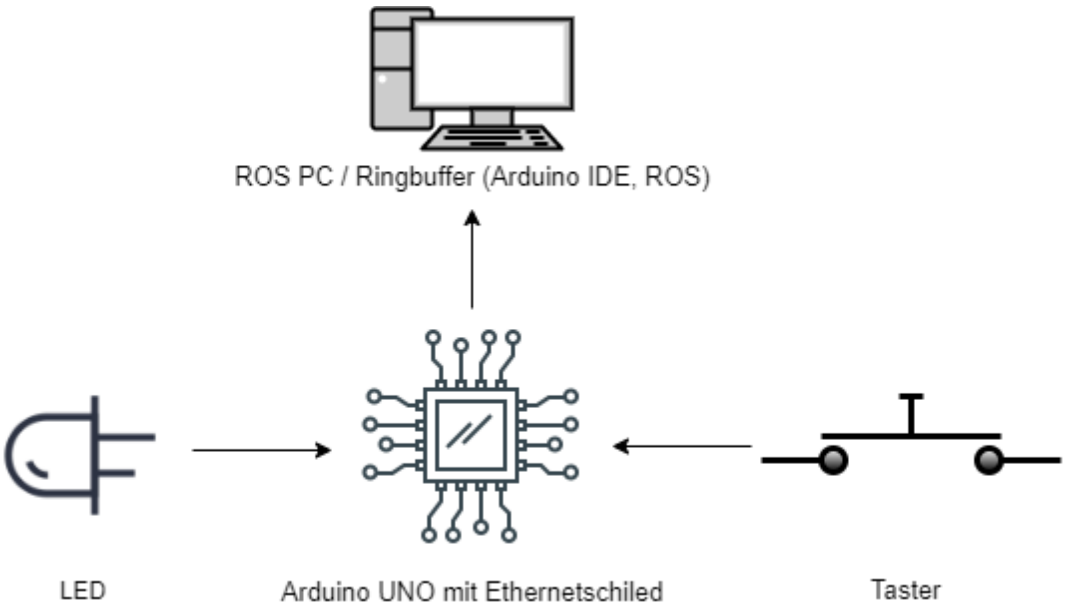


Abbildung 14: Bestandteile und Software für das Experiment

Nach der Auswahl eines Arduino uno mit Ethernet Shield als Controller für den Auslösemechanismus, der im mobilen Perzeptionssystem verwendet wird, wurden ein Taster und eine LED wie in der Abbildung 15 gezeigt mit einer einfachen Schaltung an den Arduino uno angeschlossen. Wie die Abbildung 14 zeigt, läuft das ROS im Computer und der Arduino uno, der mit dem Ethernet Shield, der LED und dem Button verbunden ist, ist mit dem ROS über Ethernet verbunden. Der Code für den Arduino UNO wurde in C++ geschrieben (Arduino IDE) und ein ROS-Knoten wurde in Python geschrieben (Visual Studio Code).

Dieser Code ist in C++ geschrieben und wird verwendet, um eine LED mit einer Taste zu bedienen und den Zustand der Taste über eine Ethernet-Verbindung in einem ROS-Netzwerk zu veröffentlichen.

Der Code beginnt mit der Einbindung der erforderlichen Bibliotheken und Nachrichtentypen für die ROS-Kommunikation sowie der Definition der Anschlüsse für den Taster und die LED. Die MAC-Adresse des Ethernet-Shields wird ebenso definiert wie die IP-Adresse des ROS-Masterknotens.

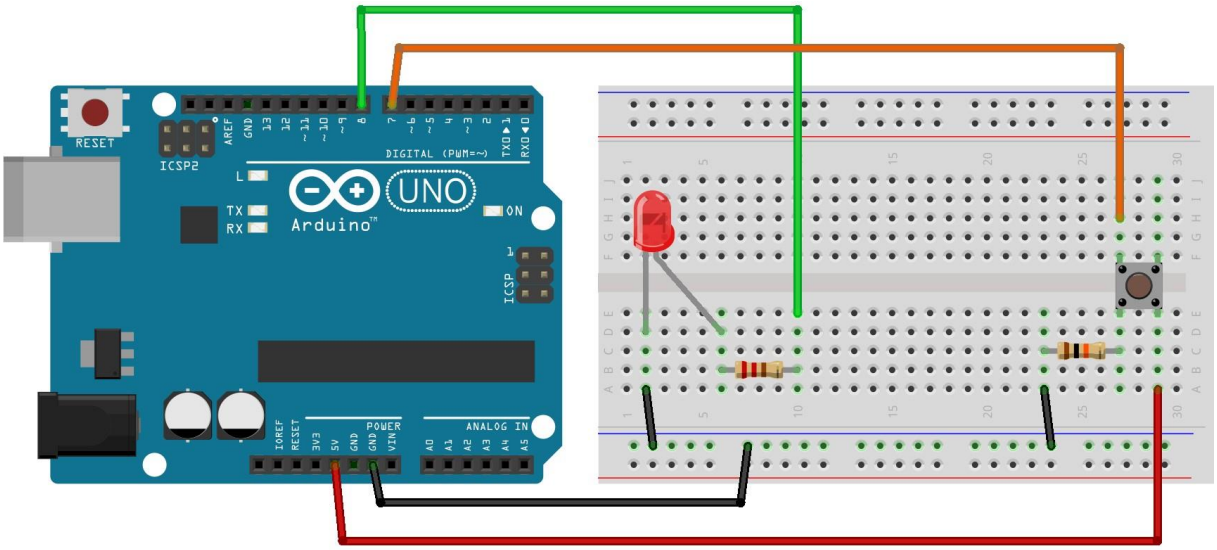


Abbildung 15: Arduino Uno mit Taster und LED

Ein ‚`NodeHandle`‘ wird initialisiert, um die Kommunikation mit dem ROS-Netzwerk zu ermöglichen. Ein ‚`Publisher`‘ und ein ‚`Subscriber`‘ werden für den Taster bzw. die LED angelegt. Der ‚`Subscriber`‘ erhält eine ‚`Callback`‘-Funktion, die immer dann ausgeführt wird, wenn eine Nachricht auf dem Topic ‚`toggle_led`‘ eingeht.

Die ‚`Setup`‘-Funktion initialisiert den ‚`pinMode`‘ für die LED und den Taster und setzt den letzten Zustand des Tasters auf 0. Das Node-Handle registriert den ‚`Publisher`‘ und den ‚`Subscriber`‘. Die Variable ‚`statechange`‘ wird auf 1 initialisiert, um sicherzustellen, dass die erste Nachricht veröffentlicht wird.

In der Schleifenfunktion prüft der Code, ob das Ethernet-Shield mit dem ROS-Masterknoten verbunden ist. Wenn dies der Fall ist, wird das Knotenhandle aufgerufen, um Nachrichten zu empfangen. Wenn die Taste gedrückt wird und der letzte Zustand nicht gedrückt war, leuchtet die LED auf, der letzte Zustand wird

aktualisiert und eine "`True`"-Nachricht wird gesendet. Wenn die Taste losgelassen wird und der letzte Zustand gedrückt war, schaltet sich die LED aus, der letzte Zustand wird aktualisiert, und es wird eine "`False`"-Meldung zurückgegeben. Wenn sich der Zustand ändert, wird die Tastennachricht veröffentlicht und der Knotenhandle aktualisiert.

Wenn die ‚`statechange`'-Variable gleich 1 ist, wird die Tastennachricht veröffentlicht und das Knotenhandle aufgerufen, um Nachrichten an den ROS-Masterknoten zu senden. Schließlich wird die Schleifenfunktion um 100 Millisekunden verzögert, bevor sie erneut beginnt.

### 4.2.2 Ringspeicher

Nach der Markierung relevanter Daten mit einem Taster im Perzeptionssystem wird hier der Ringspeicher implementiert, um diese markierten relevanten Daten in einem Speicher aus Ringspeicher dauerhaft zu speichern.

Zur Implementierung des Ringspeichers wurde ein Code geschrieben. Dieser Code ist ein Python-Skript, das Teil eines ROS-Projekts (Robot Operating System) ist. Es enthält mehrere Bibliotheken, die für die Interaktion mit dem Betriebssystem, primitive ROS-Typen, Datums- und Zeitmanipulation, Kopieren und Entfernen von Dateien, Threading und ROS-API verwendet werden.

Der Hauptzweck des Codes ist es, einen Ringspeicher von vier ROS-Bag-Dateien mit einer Dauer von (bspw. fünf Sekunden) zu erstellen (die Anzahl der ROS-Bag-Dateien und die Dauer des Ringspeichers können je nach Bedarf geändert werden) und sie in einem bestimmten Ordner zu speichern. Außerdem lauscht der Code auf ein ROS-Topic namens "`button_press`" und wenn eine Nachricht mit dem Datum "`True`" empfangen wird, veröffentlicht der Code die aktuelle Zeit in einem anderen ROS-Topic namens "`datetime_now`" und schreibt das aktuelle Datum und die Zeit des betreffenden Ereignisses in eine "`.txt`"-Datei. Das Ablaufdiagramm des Ringspeicherkonzepts ist in der Abbildung 16 dargestellt.

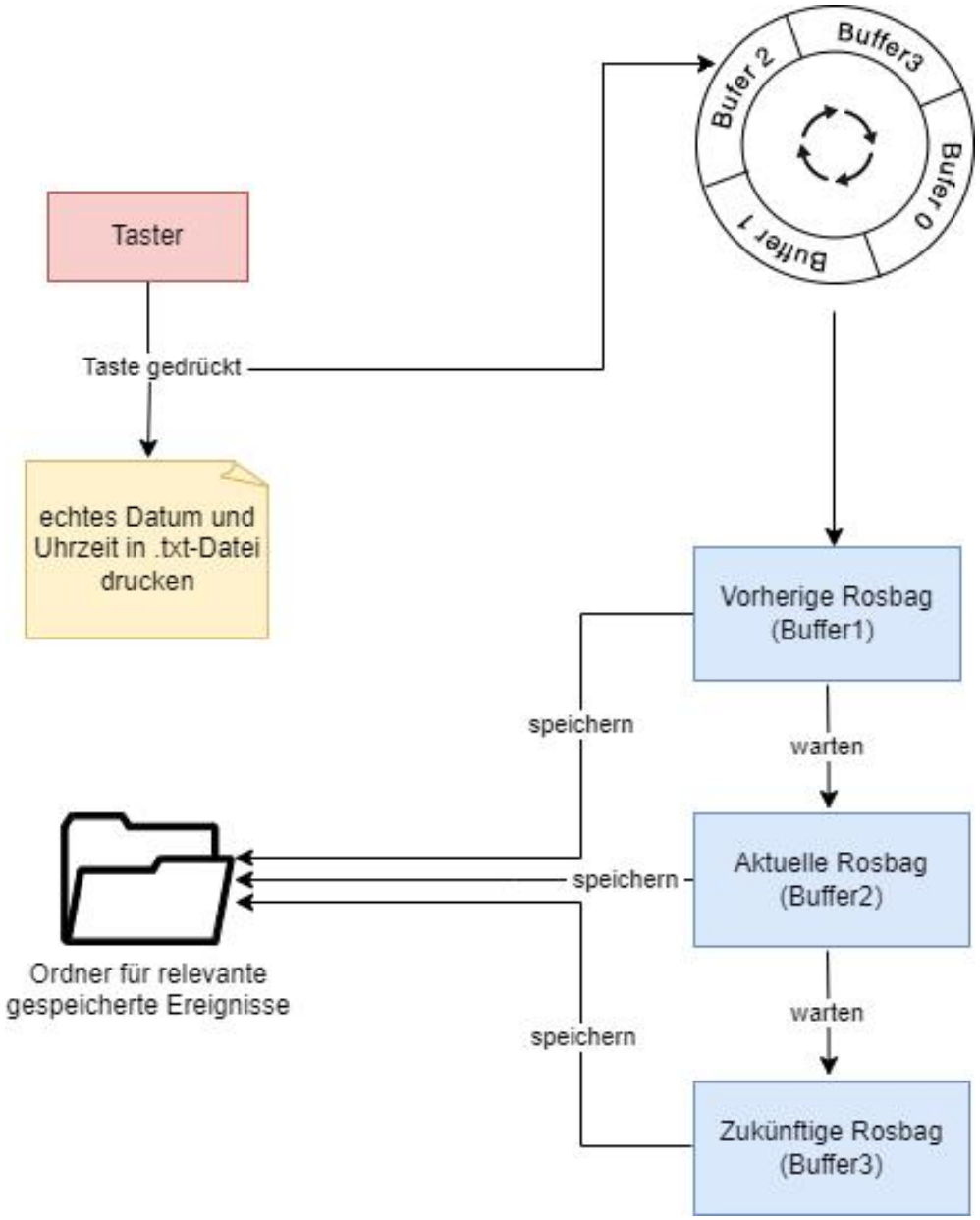


Abbildung 16: Ablaufdiagramm des Ringspeicherkonzepts

Wenn die Taste gedrückt wird, druckt der Code die aktuelle, die vorherige und die zukünftige Zeit auf dem Terminal aus, und wenn die vorherige Datei existiert, kopiert er die Datei aus dem Ringspeicher-Ordner in einen anderen Ordner, der zum Speichern der entsprechenden ROS-Bag-Dateien erstellt wurde. Die aktuelle Datei wird mit einer Verzögerung von z.B. fünf Sekunden und die zukünftige Datei mit einer Verzögerung von z.B. zehn Sekunden kopiert, wie in der Abbildung 16 gezeigt (die Verzögerung ist abhängig von der Zeit der ROS-Bag-Dateien). Wenn die vorherige Datei nicht existiert, gibt der Code "`Wait for a while`" auf dem Terminal aus.

Schließlich verwendet der Code Threading, um die beiden Hauptfunktionen des Skripts gleichzeitig auszuführen: die "`ringspeicher`"-Funktion, die den Ringspeicher erstellt, und die "`listener`"-Funktion, die auf das Thema "`button_press`" hört.

Mit diesem implementierten Ringspeicher-Code werden die drei relevanten Dateien (vergangene, aktuelle und zukünftige Dateien) aus dem Ringspeicher im ROSbag-Format in einem permanenten Speicher gespeichert, wenn die Taste gedrückt wird.

### 4.2.3 Drehgeber

Die Datenerfassung im Ringspeicher sollte automatisch in Abhängigkeit von der Zugposition erfolgen. Der Encoder ist mit dem Arduino Uno verbunden, der später mit ROS über Ethernet verbunden wird, wie in Abbildung 17 gezeigt. Der Code für diese Implementierung wurde in Arduino IDE geschrieben. Dieser Code ist für ein System, das Encoder Impulse von zwei Sensoren zählt, die an Pins auf einem Arduino Uno Board angeschlossen sind. Er verwendet die Arduino-Programmiersprache und die ROS-Bibliothek (Robot Operating System).

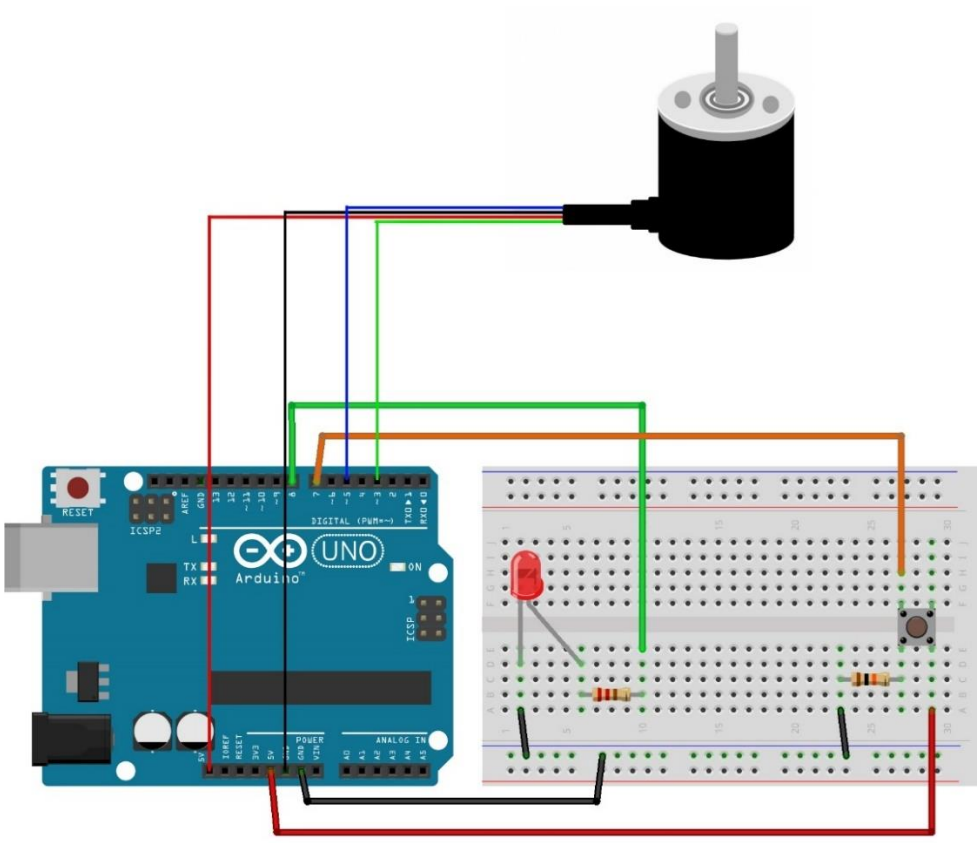


Abbildung 17: Arduino Uno mit Drehgeber

Der Code beginnt mit der Einbindung der erforderlichen Bibliotheken, der Definition einiger Konstanten und der Erstellung von ROS-bezogenen Variablen, wie z. B. einem ‚NodeHandle' und einem ‚Publisher'.

Dann werden zwei Interrupt-Service-Routinen (ISR) definiert. Dabei handelt es sich um Funktionen, die von der Mikrocontrollerplatine aufgerufen werden, sobald ein Interrupt von den Encoder Sensoren ausgelöst wird. In diesem Fall gibt es zwei ISR-Funktionen, eine für jeden Encoder Sensor. Diese Funktionen inkrementieren oder dekrementieren eine Zählervariable (encoder_count) in Abhängigkeit von der Richtung des Encoder Impulses.

In der Funktion ‚setup()' wird der ‚NodeHandle' initialisiert und der Publisher bekannt gemacht. Die Pins, an die die Encoder Sensoren angeschlossen sind, werden als Eingang gesetzt, und die ISRs werden mit diesen Pins verbunden.

In der Funktion ‚loop()' wird der Wert der Zählervariable mit Hilfe des zuvor erstellten Publishers im ROS-Topic ‚encoder_count' veröffentlicht. Das ROS-System wird dann mithilfe von ‚nh.spinOnce()' mit allen neuen Informationen aktualisiert, und es wird eine Verzögerung von 10 Millisekunden hinzugefügt, um zu verhindern, dass die Schleife zu schnell läuft.

Zum Aufbau der Kommunikation zwischen Ros und Arduino UNO wurde ein Python-Skript für einen ROS-Knoten bearbeitet, das für den Ringspeicher aufgeschrieben ist. Ein Python-Skript für einen ROS-Knoten, das die Themen ‚button_press' und ‚encoder_count' abonniert und die aktuelle Datumszeit im Thema ‚datetime_now' veröffentlicht.

Das Skript nutzt außerdem Threading, um die Funktion ‚ringspeicher' in einem separaten Thread auszuführen, der die ROS-Dateien im Ringspeicher aufzeichnet. Die Funktion ‚righ_ticks_callback' wird aufgerufen, wenn das Thema ‚encoder_count' eine Nachricht empfängt, die den Wert der Variablen ‚forward_count' aktualisiert und prüft, ob das ‚running_flag' True oder False ist. Wenn der Wert der Variablen ‚forward_count' größer oder gleich zehn ist, wird das ‚running_flag' auf False gesetzt und die Nachricht ‚stop' in die Warteschlange gestellt.

Mit diesem Code wird die Aufzeichnung der ROS-Bag-Dateien im Ringspeicher automatisch entsprechend der Position des Encoders erfolgen.

## 4.3 Test und Validierung des Ansatzes

Um die Funktionalität der vorgeschlagenen Implementierungen sicherzustellen, haben wir Tests im Perzeptionslabor durchgeführt.

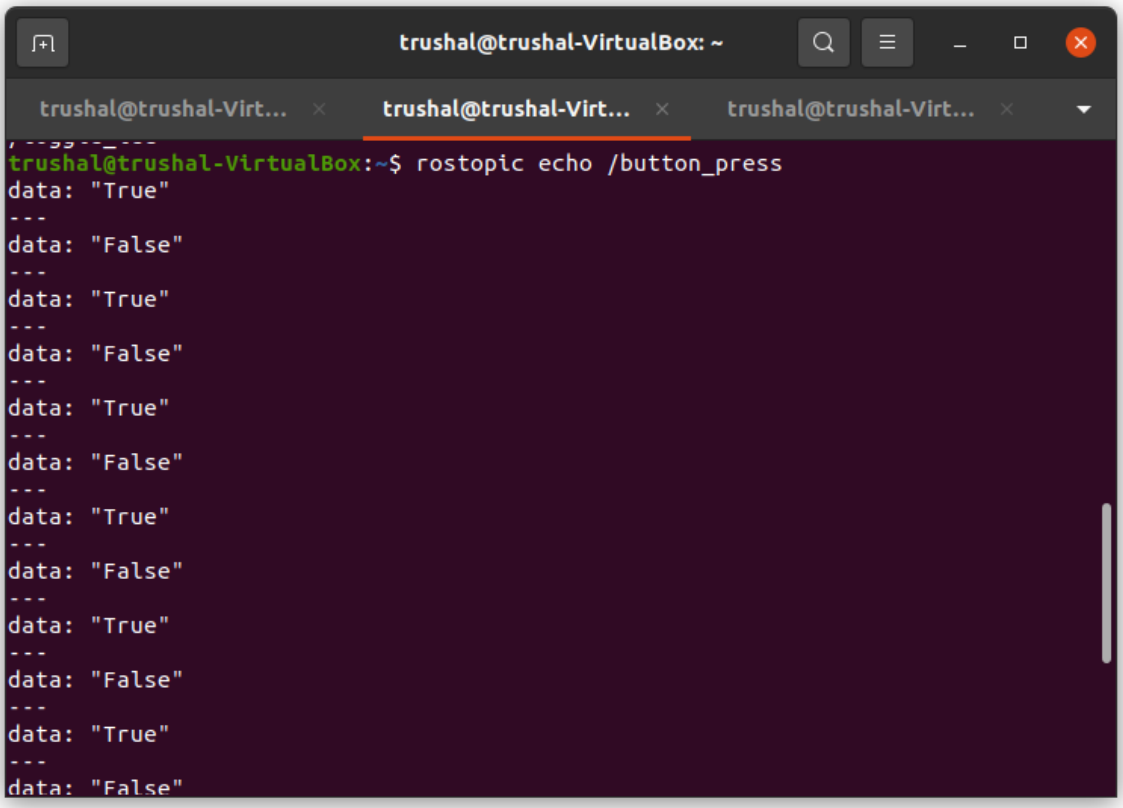


Abbildung 18: True und False auf dem Terminal

Im Labor haben wir den Taster an das Perzeptionssystem angeschlossen und die Ausgabe auf dem Terminal überwacht. Während des Tests drückten wir den Taster mehrmals, und jedes Mal gab das System "True" an das Terminal aus, und die LED auf dem Arduino-Board blinkte, was anzeigte, dass der Tastendruck erfolgreich erkannt wurde.

Wie in der Abbildung 18 gezeigt, gab das System "True" an das Terminal aus, wenn der Taster gedrückt wurde, und sobald der Taster losgelassen wurde, gab das System "False" an das Terminal aus, bis der Taster das nächste Mal gedrückt wurde. Wie in der Abbildung 19 gezeigt ist, wenn die Taste zur gleichen Zeit unter dem anderen Rostopic für den Zeitstempel gedrückt wird, können wir sehen, dass das System "Datum und Uhrzeit" an das Terminal und in die andere ".txt"-Datei ausgibt (Abbildung 20).

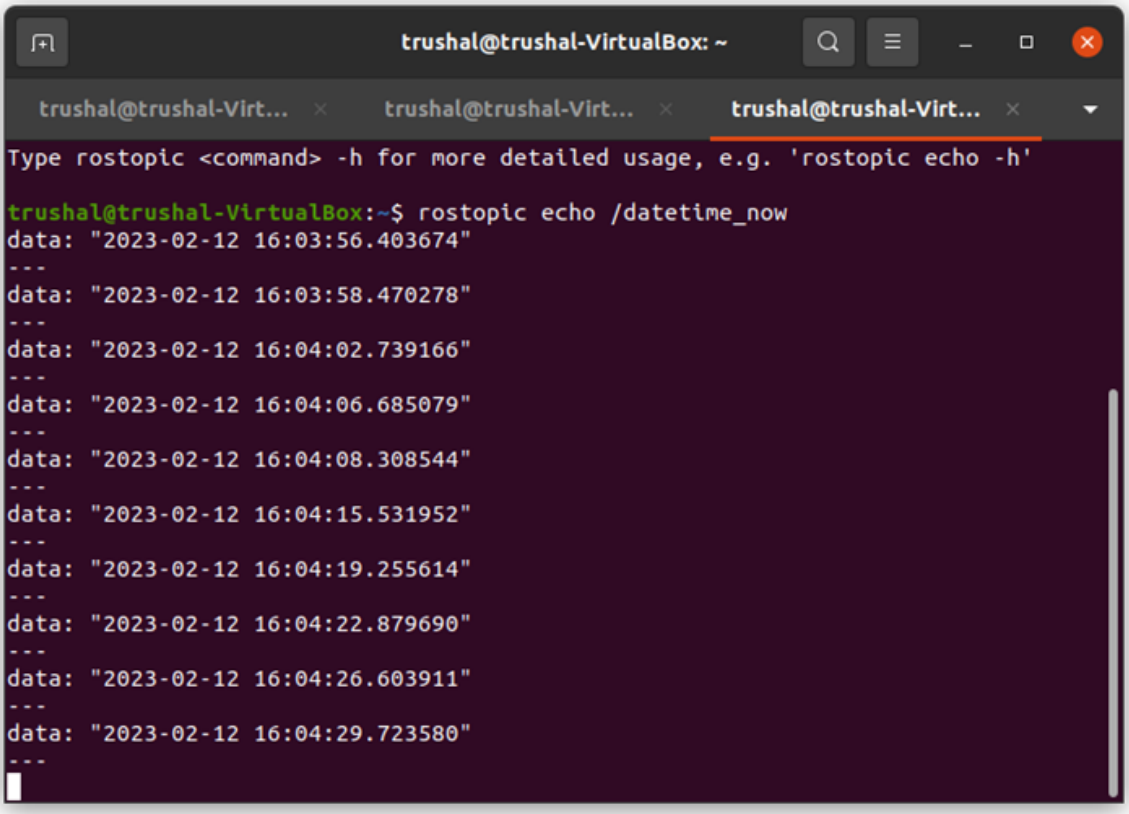


Abbildung 19: Datum und Uhrzeit des Tastendrucks

Um die Genauigkeit der Tastenimplementierung zu überprüfen, haben wir auch Screenshots der Terminalausgabe während der Labortestfahrt gemacht. Diese Screenshots bestätigen, dass das System "True" ausgab, wenn die Taste gedrückt wurde, und ansonsten "False".

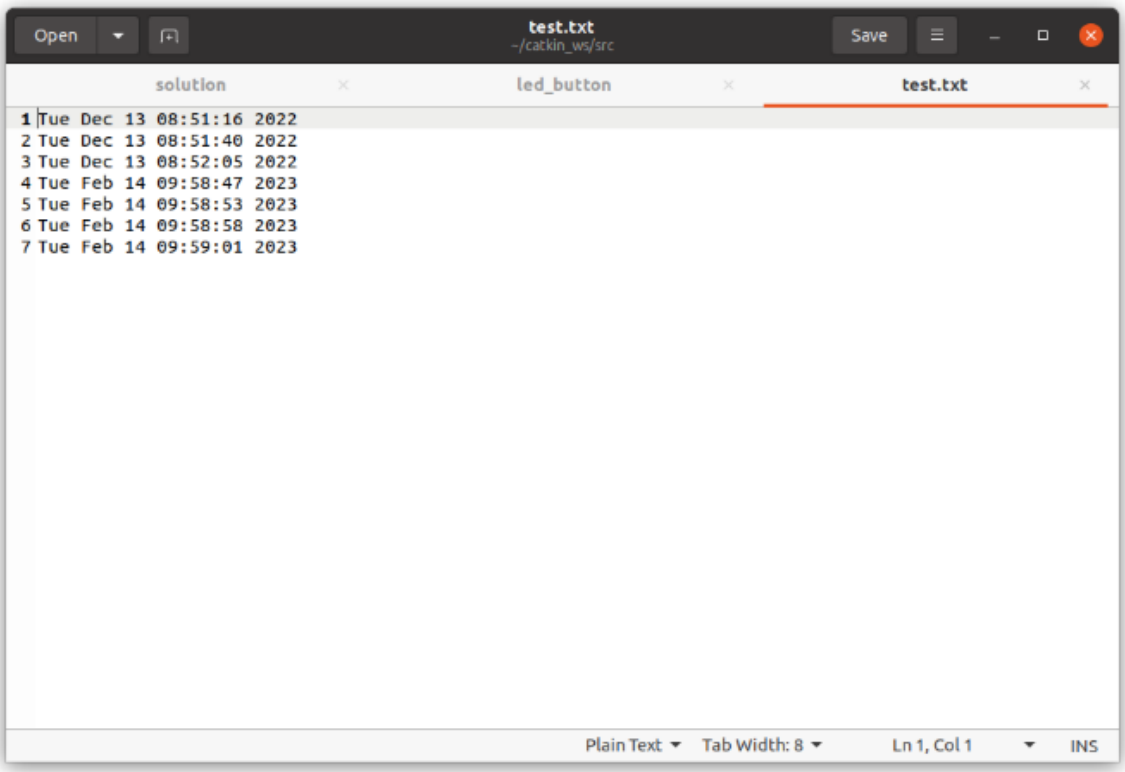


Abbildung 20: Datum und Uhrzeit in .txt- Datei

Um die Funktionalität der Taster zur Speicherung der relevanten Daten zu testen, haben wir die Taster während eines Testlaufs des Systems manuell ausgelöst. Wir stellten fest, dass die Taste die relevanten Daten erfolgreich in dem angegebenen Zielordner mit dem Zeitstempel des Tastendrucks (Abbildung 22) aus dem Ringspeicher (Abbildung 21) speicherte. Wir bestätigten auch, dass der Ringspeicher nach dem Drücken der Taste wie erwartet weiterhin Daten aufzeichnete.

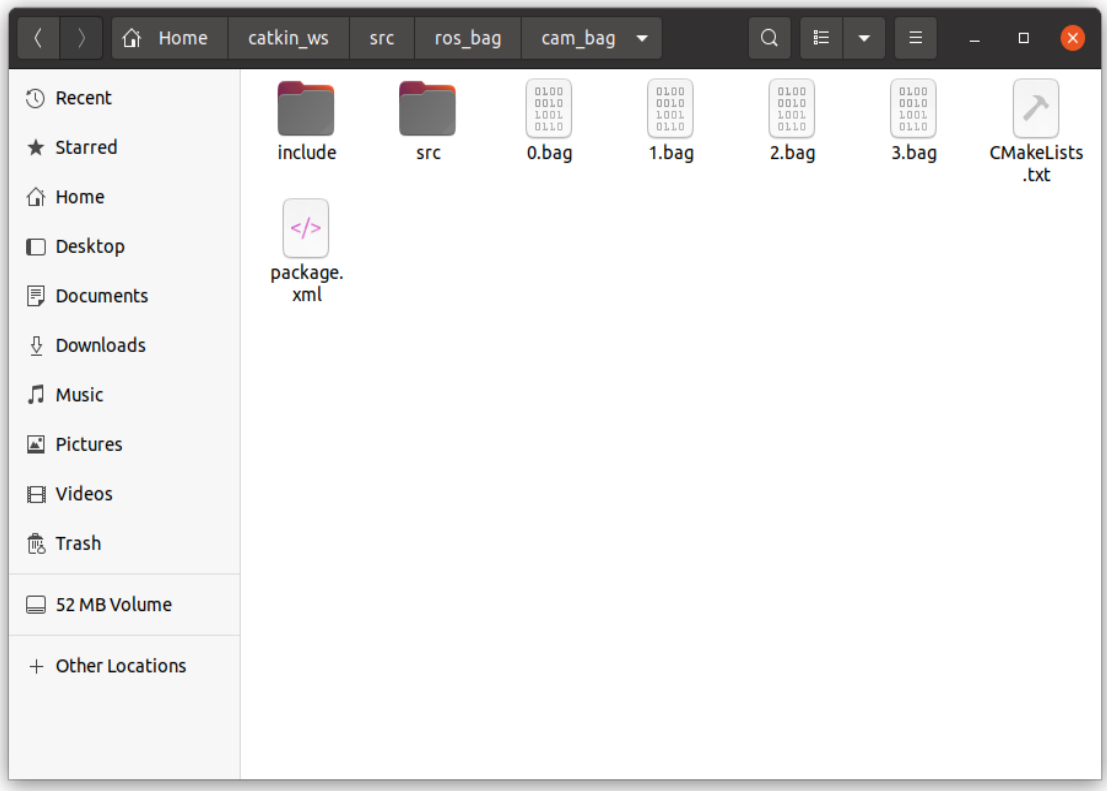


Abbildung 21: Ringspeicher im Ordner

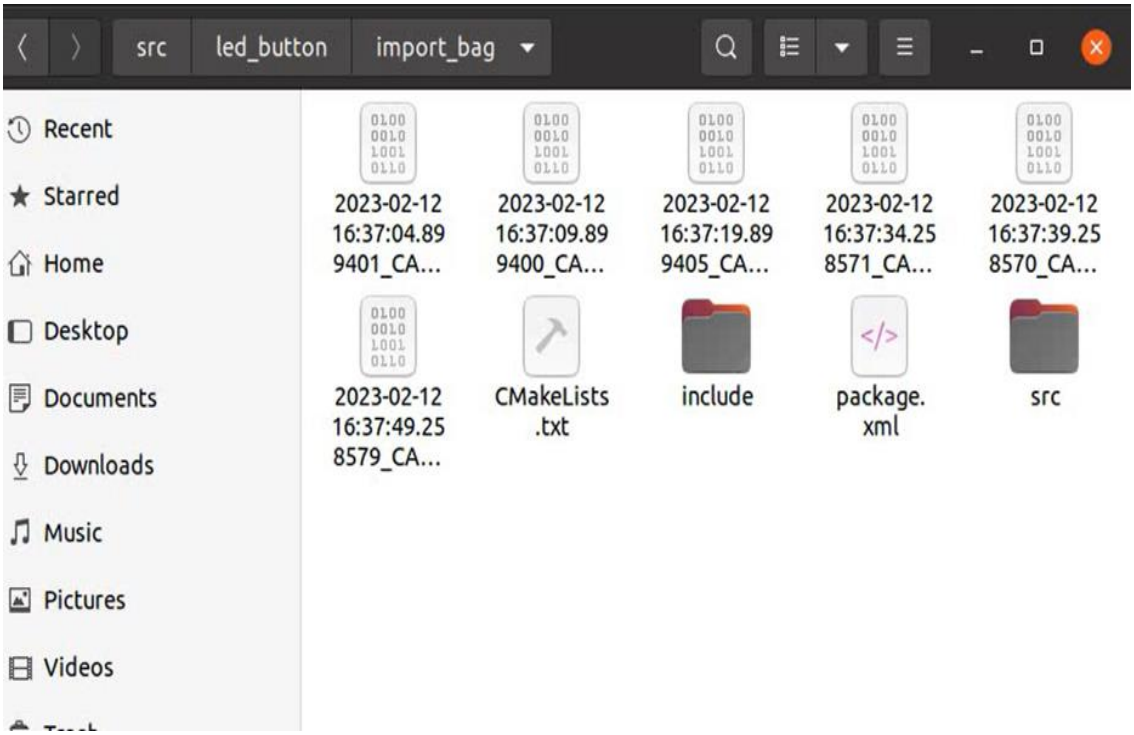


Abbildung 22: relevanten Daten im Zielordner

Um sicherzustellen, dass die gespeicherten Daten korrekt waren, verglichen wir die gespeicherten Daten mit den im gleichen Zeitraum im Ringspeicher aufgezeichneten Daten. Wir konnten feststellen, dass die gespeicherten Daten mit dem entsprechenden Abschnitt der Ringspeicherdaten identisch waren.

Um schließlich die automatische Datenaufzeichnung auf der Grundlage der Zugposition zu testen, implementierten wir einen Drehgeber und überprüften, dass die Sensordaten nur aufgezeichnet wurden, wenn sich der Drehgeber vorwärtsbewegte. Wir bestätigten auch, dass die Aufzeichnung gestoppt wurde, wenn sich der Drehgeber zurückbewegte, anhielt oder wenn die Fahrt endete.

Insgesamt zeigen diese Versuche, dass der vorgeschlagene Ansatz, einschließlich der Ringspeicher-Komponente, der Taste zum Speichern relevanter Daten und der automatischen Datenaufzeichnung auf der Grundlage der Zugposition, korrekt funktioniert und tatsächlich nur die relevanten Daten speichert.

Insgesamt bestätigen diese Tests und Validierungsmethoden, dass die Implementierung des Knopfes relevante Ereignisse korrekt erkennt und für den Einsatz im Eisenbahnsektor geeignet ist.

# 5 Ergebnisse und Diskussion

## 5.1 Vergleich des vorgeschlagenen Ansatzes mit bestehenden Methoden

Der vorgeschlagene Ansatz zur Entwicklung der Datenerfassung in ROS zur Markierung von betriebsrelevanten Ereignissen im Bahnbereich zur Verwaltung und Speicherung der im mobilen Perzeption System gesammelten Daten im Bahnbereich beinhaltet die Implementierung eines Tasters zur Markierung relevanter Ereignisse, eines Ringspeichers zur Speicherung nur relevanter Daten und eines Encoders zur automatischen Datenerfassung anhand der Zugposition.

Im Vergleich zu bestehenden Methoden bietet der vorgeschlagene Ansatz mehrere Vorteile. Erstens ermöglicht die Verwendung einer Taste zur Markierung relevanter Ereignisse eine einfache Identifizierung wichtiger Ereignisse in der Datenspeicherung. Dies ist eine erhebliche Verbesserung gegenüber bestehenden Methoden, bei denen es schwierig sein kann, wichtige Ereignisse in großen Datensätzen zu identifizieren.

Zweitens wird durch die Verwendung eines Ringspeichers, in dem nur relevante Daten gespeichert werden, der für die Datenspeicherung erforderliche Speicherplatz reduziert. Dies ist ein entscheidender Vorteil im Eisenbahnsektor, wo der Speicherplatz für Datenaufzeichnungen begrenzt sein kann. Darüber hinaus spart es Zeit und Arbeit, da die Notwendigkeit entfällt, große Datensätze manuell zu durchsuchen, um relevante Informationen zu finden.

Drittens entfällt durch den Einsatz eines Encoders für die automatische Datenaufzeichnung auf der Grundlage der Zugposition die Notwendigkeit manueller Eingriffe, und es wird sichergestellt, dass die relevanten Daten automatisch aufgezeichnet werden. Dieser Ansatz ist wesentlich effizienter als bestehende Methoden, bei denen die Datenaufzeichnung entweder manuell erfolgt oder kontinuierlich aufgezeichnet wird, unabhängig davon, ob sie relevant sind oder nicht.

Insgesamt bietet der vorgeschlagene Ansatz eine effiziente und effektive Lösung für die Verwaltung und Speicherung von Daten, die im Rahmen des mobilen Perzeptionssystems im Eisenbahnsektor erfasst werden. Er bietet erhebliche Vorteile

gegenüber bestehenden Methoden und dürfte die Genauigkeit und Effizienz der Datenverwaltung und -analyse im Eisenbahnsektor verbessern.

## 5.2 Diskussion der Vorteile und Grenzen des vorgeschlagenen Ansatzes

Der vorgeschlagene Ansatz, eine Taste zur Markierung relevanter Ereignisse zu implementieren, einen Ringspeicher zu verwenden, um nur die relevanten Daten zu speichern, und einen Encoder für die automatische Datenerfassung im Eisenbahnsektor einzubauen, hat mehrere Vorteile und Einschränkungen.

**Vorteile:**

1. Effiziente Datenspeicherung: Mit dem vorgeschlagenen Ansatz werden nur die relevanten Daten im Zugdatenspeicher gespeichert, was die Datenmenge und den Speicherbedarf reduziert. So wird sichergestellt, dass die wichtigen Informationen erhalten bleiben und gleichzeitig der Speicheraufwand reduziert wird.

2. Einfacher Zugang zu relevanten Ereignissen: Durch die Implementierung von Tasten lassen sich relevante Ereignisse in der gesamten Datenaufzeichnung leicht identifizieren. Der Fahrer kann die relevanten Ereignisse während der Fahrt markieren, und später werden nur die markierten Daten gespeichert, auf die einfach und schnell zugegriffen werden kann.

3. Automatische Datenaufzeichnung: Die Kodierer basierte automatische Datenaufzeichnung sorgt dafür, dass die Datenaufzeichnung in Abhängigkeit von der Zugbewegung automatisch startet und stoppt. Dies macht manuelle Eingriffe überflüssig und verringert die Möglichkeit von Datenverlusten oder Fehlern.

**Einschränkung:**

1. Abhängigkeit von manuellen Eingriffen: Der vorgeschlagene Ansatz erfordert ein manuelles Eingreifen bei der Markierung relevanter Ereignisse über eine Taste. Dies kann zu menschlichen Fehlern führen, und es ist dem Fahrer nicht immer möglich, die Taste zum richtigen Zeitpunkt zu drücken, was zu verpassten Ereignissen führen kann.

2. Begrenzte Kapazität des Ringspeichers: Die Kapazität des Ringspeichers, in dem die relevanten Daten gespeichert werden, ist begrenzt, d. h. wenn die relevanten Ereignisse zu häufig auftreten oder das Datenvolumen zu groß ist, können einige relevante Daten verloren gehen.

Zusammenfassend lässt sich sagen, dass der vorgeschlagene Ansatz mehrere Vorteile hat, z. B. eine effiziente Datenspeicherung, einen einfachen Zugriff auf relevante Ereignisse und eine automatische Datenaufzeichnung. Er hat jedoch auch einige Einschränkungen, wie die Abhängigkeit von manuellen Eingriffen und die begrenzte Kapazität des Ringspeichers. Bei der Umsetzung des vorgeschlagenen Konzepts ist es daher wichtig, diese Vorteile und Einschränkungen zu berücksichtigen, um seine Effektivität und Effizienz zu gewährleisten.

# 6 Schlussfolgerung und zukünftige Arbeit

## 6.1 Aus den Ergebnissen und der Diskussion gezogene Schlussfolgerungen

Auf der Grundlage der in dieser Arbeit vorgestellten Ergebnisse und Diskussionen können mehrere Schlussfolgerungen gezogen werden.

Erstens hat sich die Implementierung Eine Taste zur Markierung relevanter Ereignisse während einer Zugfahrt als effektive Lösung für die Verwaltung großer Datenmengen erwiesen, die von einem mobilen Perzeptionssystem im Eisenbahnsektor erfasst werden. Der Tester ermöglicht es dem Zugführer, auf einfache Weise wichtige Ereignisse zu markieren, die dann leicht aus dem gesamten Datenaufzeichnungsspeicher abgerufen werden können.

Zweitens hat sich die Implementierung eines Ringspeichers zur Speicherung relevanter Daten als wirksame Lösung erwiesen, um den gesamten Datenspeicherbedarf für die Zugfahrt zu reduzieren. Indem nur relevante Daten gespeichert werden, wird der Speicherbedarf reduziert, und die Daten können bei Bedarf leicht abgerufen werden. Darüber hinaus hat die automatische Aufzeichnung von Daten im Ringspeicher entsprechend der Zugposition mit Hilfe eines Encoders den Datenerfassungsprozess weiter verbessert, da keine manuellen Eingriffe mehr erforderlich sind.

Schließlich haben die in dieser Arbeit durchgeführten Tests und Validierungen die Wirksamkeit des vorgeschlagenen Ansatzes bestätigt. Die Ergebnisse zeigen, dass der Taster relevante Ereignisse effektiv markiert und der Ringspeicher relevante Daten in einem bestimmten Ordner speichert, während alte Daten überschrieben werden, wenn der Speicher voll ist. Diese Ergebnisse belegen, dass der vorgeschlagene Ansatz große Datenmengen im Eisenbahnsektor effektiv verwalten kann.

Insgesamt hat diese Arbeit gezeigt, dass der vorgeschlagene Ansatz der Implementierung eines Tasters und eines Ringspeichers große Datenmengen im Eisenbahnsektor effektiv verwalten kann und das Potenzial hat, den Datenerfassungsprozess in diesem Bereich zu verbessern.

## 6.2 Empfehlungen für zukünftige Forschung zur Verbesserung des Ansatzes

Auf der Grundlage der Ergebnisse und Einschränkungen dieser Studie gibt es mehrere Empfehlungen für zukünftige Forschung zur Verbesserung des vorgeschlagenen Ansatzes für das mobile Perzeption System im Eisenbahnsektor:

1. Erforschung der Verwendung von fortschrittlicherer Hardware: Während der ESP32-POE in der Lage war, den vorgeschlagenen Ansatz zu handhaben, hat er einige Einschränkungen, wenn es um die Ausführung von ROS geht. Weitere Forschungen können die Verwendung von fortschrittlicherer Hardware mit höherer Verarbeitungsleistung und Speicherkapazität untersuchen, wie z.B. NVIDIA Jetson oder Raspberry Pi 4.

2. Bibliotheken und Code aktualisieren: Im Zuge des technologischen Fortschritts werden neue Bibliotheken und Codes entwickelt, um bestehende zu verbessern. Zukünftige Forschungen können die Aktualisierung der in dieser Studie verwendeten Bibliotheken und Codes untersuchen, um von neuen technologischen Fortschritten zu profitieren.

3. Implementierung von Algorithmen für maschinelles Lernen: Algorithmen des maschinellen Lernens können verwendet werden, um die vom mobilen Perzeptionssystem gesammelten Daten zu analysieren und Anomalien oder Ereignisse zu erkennen, die von Interesse sind. Zukünftige Forschungen können die Verwendung von Algorithmen des maschinellen Lernens in dem vorgeschlagenen Ansatz untersuchen, um die Genauigkeit und Effizienz der Datenanalyse zu verbessern.

4. Integration mit anderen Systemen: Der vorgeschlagene Ansatz kann mit anderen bestehenden Eisenbahnsystemen integriert werden, um eine umfassendere Lösung zu bieten. Zukünftige Forschungen können die Integration des mobilen Perzeptionssystems mit anderen Systemen wie der automatischen Zugsteuerung oder Zugüberwachungssystemen untersuchen.

Insgesamt hat der vorgeschlagene Ansatz vielversprechende Ergebnisse bei der Bewältigung der Herausforderungen der Datenerfassung und -speicherung im Eisenbahnsektor gezeigt. Es sind jedoch weitere Forschungsarbeiten erforderlich, um den Ansatz zu verbessern und neue technologische Entwicklungen zu erforschen, um die Möglichkeiten des mobilen Perzeptionssystems zu erweitern.

# Literaturverzeichnis

[1] „Deutsche Bahn“, *Wikipedia*. 4. April 2023. Zugegriffen: 10. April 2023. [Online]. Verfügbar unter: https://de.wikipedia.org/w/index.php?title=Deutsche_Bahn&oldid=232489507

[2] „DB Systemtechnik“, *Wikipedia*. 20. Januar 2022. Zugegriffen: 10. April 2023. [Online]. Verfügbar unter: https://de.wikipedia.org/w/index.php?title=DB_Systemtechnik&oldid=219362820

[3] „DB Systemtechnik“, *Stadt Minden – Die Stadt mit dem Plus*. https://www.minden.de/wirtschaft-mobilitaet-wohnen/standort-minden/arbeiten-in-minden/db-systemtechnik/ (zugegriffen 2. Februar 2023).

[4] „Trends and developments in the automation of heavy rail operations“, *Global Railway Review*. https://www.globalrailwayreview.com/article/97734/trends-developments-automation-heavy-rail/ (zugegriffen 3. Februar 2023).

[5] „Sensors4Rail bringt mehr Kapazität auf die Schiene“. http://digitale-schiene-deutschland.de/en/Sensors4Rail (zugegriffen 10. April 2023).

[6] C. Vavra, „Researchers use simulated environments to train AI“, *Control Engineering*, 27. November 2021. https://www.controleng.com/articles/researchers-use-simulated-environments-to-train-ai/ (zugegriffen 7. Februar 2023).

[7] T. Team, „The Advantages of Real Data Over Synthetic Data“, *Tasq.ai*, 27. Dezember 2022. https://www.tasq.ai/blog/the-advantages-of-real-data-over-synthetic-data/ (zugegriffen 7. Februar 2023).

[8] „Advantages and Disadvantages of DBMS - The Study Genius“, 20. Juni 2019. https://www.thestudygenius.com/advantages-and-disadvantages-of-database/ (zugegriffen 10. Februar 2023).

[9] P. Mcmahon, T. Zhang, und R. Dwight, „Requirements for Big Data Adoption for Railway Asset Management“, *IEEE Access*, Bd. 8, S. 15543–15564, 2020, doi: 10.1109/ACCESS.2020.2967436.

[10] „Praktikumsbericht Trushal 1.pdf“, *Google Docs*. https://drive.google.com/file/d/1fJ1y8aRcUYJjbO0rebXD8iLKWdiI5n4C/view?usp=embed_facebook (zugegriffen 29. April 2023).

[11] „noetic - ROS Wiki“. http://wiki.ros.org/noetic (zugegriffen 3. Februar 2023).

[12] „ROS: Home“. https://www.ros.org/ (zugegriffen 2. Februar 2023).

[13] „ROS 101: An Intro to the Robot Operating System“, *designnews.com*, 28. Juni 2019. https://www.designnews.com/gadget-freak/ros-101-intro-robot-operating-system (zugegriffen 29. April 2023).

[14] „rosbag - ROS Wiki“. http://wiki.ros.org/rosbag (zugegriffen 25. Februar 2023).

[15] „Circular Buffer - an overview | ScienceDirect Topics“. https://www.sciencedirect.com/topics/computer-science/circular-buffer (zugegriffen 25. Februar 2023).

[16] P. Srivastava, „Implementing a Ring Buffer in Java | Baeldung“, 27. Juni 2020. https://www.baeldung.com/java-ring-buffer (zugegriffen 2. März 2023).

[17] „Quadrature Fundamentals“. https://deltamotion.com/support/webhelp/rmctools/Controller_Features/Transducer_Basics/Quadrature_Fundamentals.htm#:~:text=Quadrature%20encoders%20use%20two%20output,rotating%20in%20a%20clockwise%20direction (zugegriffen 1. März 2023).

[18] „600P/R Drehgeber, inkrementelles Drehgebermodul, DC 5-24v 2 Phase, 600P / R photoelektrischer inkrementaler Drehgeber 5V-24V AB 2 phasige Welle 6mm : Amazon.de: Gewerbe, Industrie & Wissenschaft“. https://www.amazon.de/-/en/rotary-encoder-incremental-photoelectric-5V-24V/dp/B07R8VSMQH (zugegriffen 15. Februar 2023).

[19] „ESP32-POE-ISO - Open Source Hardware Board“, *Olimex*. https://www.olimex.com/Products/IoT/ESP32/ESP32-POE-ISO/open-source-hardware (zugegriffen 15. Februar 2023).

[20] „Answer to ‚Mismatched protocol version in packet Error: lost sync or rosserial_python is from different ros release than the rosserial client‘“, *Stack*

*Overflow*, 22. Oktober 2021. https://stackoverflow.com/a/69678199 (zugegriffen 15. Februar 2023).

[21] „Figure 1. Arduino Uno extended with Arduino Ethernet Shield. [ntpro.nl]“, *ResearchGate*. https://www.researchgate.net/figure/Arduino-Uno-extended-with-Arduino-Ethernet-Shield-ntpronl_fig1_313252037 (zugegriffen 25. Februar 2023).

[22] „rosserial_arduino/Tutorials/Arduino IDE Setup - ROS Wiki“. http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup (zugegriffen 7. Februar 2023).

[23] „ros_arduino_bridge - ROS Wiki“. http://wiki.ros.org/ros_arduino_bridge (zugegriffen 29. April 2023).

# Anlagen

Die Zip-Datei für den Programmcode ist der Bachelorarbeit beigefügt.

# Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

---

Mittweida, Trushal Babubhai, Paghadar