# Entwicklung einer Versuchsplattform für die Automatisierung

**Projektarbeit T3100**

des Studienganges Elektrotechnik
Fachrichtung Automation
an der Dualen Hochschule Baden-Württemberg
Standort Stuttgart

**Tudor Stefan & Nils Jakobs**

25.01.2025

| | |
|---|---|
| **Bearbeitungszeitraum** | 29.09.25 - 25.01.26 |
| **Matrikelnummer, Kurs** | 1491114, 4770321, TEL23AT |
| **Betreuer der Dualen Hochschule** | Prof. Dr.-Ing. Frank Bender |

# Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Projektarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____

## Kurzreferat

## Abstract

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| **RCP** | Rapid Control Prototyping, deutsch: schnelles Regelungsprototyping |
| **IMU** | Inertial Measurement Unit, deutsch: Inertialmesseinheit |
| **PC** | Personal Computer, deutsch: Persönlicher Computer |
| **SIL** | Software in the Loop, deutsch: Software im Regelkreis |
| **HIL** | Hardware in the Loop, deutsch: Hardware im Regelkreis |
| **DSP** | Digital Signal Processor, deutsch: Digitaler Signalprozessor |
| **FPGA** | Field Programmable Gate Array, deutsch: Feldprogrammierbare Gatter |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Quellcodeverzeichnis

# 1 Einführung

Dieses Kapitel gibt eine Einführung in die Thematik der Projektarbeit. Es werden die Zielsetzung und die geplante Vorgehensweise beschrieben.

## 1.1 Zielsetzung

Das Ziel dieser Projektarbeit ist es eine Versuchsplattform für die Automatiseirungstechnik zu entwickeln bei der typische Regelungstechnische Methoden experimentelle untersucht werden und anschließend auf die Zielplattform übertragen werden können. Hierfür soll ein Rapid Control Prototyping, deutsch: schnelles Regelungsprototyping (RCP)-System verwendet werden, um die Regelungsalgorithmen in Echtzeit auf der Zielplattform auszuführen, indem bereits vorhandene Hardware und Software Bausteine genutzt werden. Das RCP ist ein Verfahren mit dem zu regelnde Systeme schnell und flexibel entwickelt und getestet werden können. Hierbei ist es nicht notwendig manuelle Implementierung in Programmiersprachen für die Zielhardware zu erstellen, sondern es kann direkt von einer grafischen Simulationsumgebung wie z.B. *Matlab/Simulink* auf die Zielhardware übertragen werden mithilfe von einer automatischen Codegenerierung. Dies ermöglicht eine schnelle Iteration und Anpassung der Regelungsalgorithmen, was besonders in der Entwicklungsphase von Vorteil ist, da somti schnell und kosteneffizient Prototypen erstellt und getestet werden können. Die Versuchsplattform soll einen durchgängigen Prozess von der Modellerstellung über die Simulation bis zur Echtzeitausführung auf der Zielhardware abbilden und dabei die experimentelle Parametrierung, Optimierung sowie die Beobachtung relevanter Signale und Messgrößen ermöglichen, indem eine Kopplung zwischen Entwicklungsrechner und Zielplattform zur Signal und Parameterkommunikation genutzt wird (Hoyos-Gutiérrez et al. 2023).

## 1.2 Vorgehensweise

Die Projekarbeit setzt auf eine Einarbeitung in das Rapid Control Prototyping, um die theoretischen Grundlagen, die verwendeten Begriffe und typische Prozessabläufe einzuordnen und daraus geeignete Vorgehensprinzipien abzuleiten. Aufbauend auf diesem Kenntnisstand werden Anforderungen an die zu entwickelnde Versuchsplattform definiert. Dabei werden funktionale Anforderungen beschrieben, die sich aus den geplanten Experimenten ergeben, sowie nicht funktionale Anforderungen festgelegt, die unter anderem die Umsetzbarkeit, Erweiterbarkeit und Randbedingungen der Nutzung betreffen.

Nachdem die Randbedingungen gesetzt wurden, erfolgt die Auswahl geeigneter Hardware und Software Werkzeuge für die Umsetzung der Versuchsplattform. Ziel ist es, eine Kombination

aus Hardware und Software zu identifizieren, die eine effiziente Entwicklung, Simulation und Echtzeitausführung der Regelungsalgorithmen ermöglicht.

Parallel zum physischen Aufbau wird die Versuchsplattform in Matlab und Simulink modelliert, um ein ausführbares Systemmodell für die Simulation bereitzustellen. In diesem Modell werden Sensorik, Aktorik und die wesentlichen dynamischen Eigenschaften des Prozesses abgebildet, sodass Algorithmen vor der Implementierung auf der realen Plattform unter definierten Bedingungen untersucht werden können. Abschließend wird der Ansatz des Rapid Control Prototyping exemplarisch angewendet, indem ausgewählte Funktionen in Simulink entworfen, simuliert und anschließend mittels automatischer Codegenerierung auf die Zielhardware übertragen werden. Die Implementierung wird getestet und validiert, um die Funktionalität und Leistungsfähigkeit der entwickelten Versuchsplattform zu gewährleisten.

Für die Umsetzung der RCP-Methodik wird in der Projektarbeit ein Anwedungssezenario der "Lageschätzung"genutzt, dies wird im folgendem Abschnitt näher erläutert. Hierbei soll eine Inertial Measurement Unit, deutsch: Inertialmesseinheit (IMU) simuliert werden und gegebenfalls aus der ausgewählten Hardware eine reale IMU ausgelesen werden. Anschließend soll die Lage des Systems in Form von Quaterionen/Eulerwinkeln geschätzt werden (siehe Abschnitt 2.4). Die Lageschätzung soll auf Grundlage einer Sensorfusion funktionieren, indem Beschleunigungs- und Gyroskopdaten kombiniert werden. Die genaue Funktionsweise der IMU wird im Abschnitt Hierfür sollen verschiedene Algorithmen implementiert und getestet werden, um die Genauigkeit und Robustheit der Lageschätzung zu bewerten. Ziel ist es, eine zuverlässige Methode zur Bestimmung der Systemlage zu entwickeln, die in Echtzeit auf der RCP-Plattform ausgeführt werden kann.

# 2 Grundlagen und Stand der Technik

Für die vorliegende Arbeit sind Kenntnisse in den Bereichen Rapid Control Prototyping, modellbasierte Entwicklung mit Matlab und Simulink, Inertialsensorik sowie Lageschätzung mittels Sensorfusion erforderlich, weshalb in den folgenden Kapiteln die hierfür relevanten Grundlagen und der Stand der Technik dargestellt werden.

## 2.1 Rapid Control Prototyping

Rapid Control Prototyping bezeichnet einen Ansatz zur schnellen Entwicklung und Erprobung von Regelungsstrategien an Anwendungen oder Laboraufbauten. Charakteristisch ist, dass der Entwurf nicht primär durch eine manuelle Implementierung in textbasierten Programmiersprachen erfolgt, sondern durch die Modellierung als grafisches Blockdiagramm, das anschließend auf eine Zielplattform übertragen wird. Ziel ist es, den Übergang vom Entwurf zur lauffähigen Implementierung zu verkürzen, indem aus dem grafischen Modell automatisch ausführbarer Code erzeugt und auf der Zielhardware ausgeführt wird. Dadurch werden kurze Iterationszyklen unterstützt, weil Entwurf, Test und Parametrierung in engem Zusammenhang durchgeführt werden können (Hoyos-Gutiérrez et al. 2023).

Für die praktische Umsetzung wird RCP typischerweise als Kombination aus Zielhardware und Entwicklungssoftware verstanden. Die Zielhardware umfasst eine Recheneinheit mit Speicher sowie Ein und Ausgängen und kann je nach Anwendung als Personal Computer, deutsch: Persönlicher Computer (PC), Mikrocontroller oder programmierbare Logik realisiert sein. Die Entwicklungssoftware stellt eine grafische Modellierungsumgebung bereit, in der der Regelalgorithmus als Blockdiagramm beschrieben wird und aus der anschließend ausführbarer Code für die Zielplattform generiert wird (Hoyos-Gutiérrez et al. 2023).

Die Abbildung 1 ordnet den Einsatz von RCP in einen V Modellentwicklungsprozess ein. Die linke Seite beschreibt die Schritte von *Design, Modelling and Simulation* bis zur *Validation with RCP*. In dieser Phase wird der Reglerentwurf im Modell erstellt, simulativ bewertet und für die Ausführung vorbereitet. Der Übergang zur Zielplattform erfolgt im Schritt *Target Code on Hardware*, bei dem der aus dem Modell abgeleitete Code auf der Hardware ausgeführt wird. Die rechte Seite des V Modells umfasst die Schritte *Verification with SIL or HIL* sowie *Integration and Test*. Dabei werden Implementierung und Systemverhalten anhand geeigneter Testumgebungen geprüft, wobei SIL und HIL als etablierte Testformen zur schrittweisen Absicherung dienen. Die in der Abbildung markierte Verifikationsphase adressiert die Frage, ob die Implementierung die spezifizierten Anforderungen korrekt erfüllt. Die Validierungsphase adressiert die Frage, ob die Lösung im Anwendungskontext den beabsichtigten Nutzen liefert und die vorgesehenen Funktionen in der Zielumgebung erfüllt. RCP unterstützt diesen Ablauf, indem Modell, Simulation und

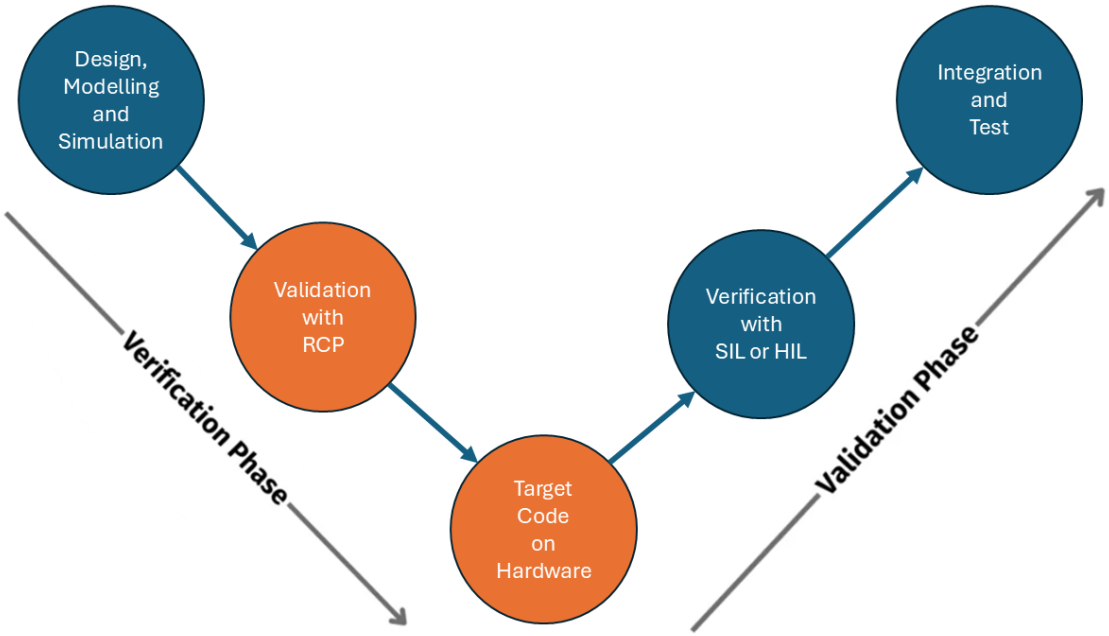


Abb. 1: V-Modell zur Einordnung des Rapid Control Prototyping (Hoyos-Gutiérrez et al. 2023).

Ausführung auf der Zielhardware eng gekoppelt sind und Anpassungen am Entwurf zeitnah am realen System überprüft werden können (Hoyos-Gutiérrez et al. 2023).

In vielen Anwendungen bildet Matlab in Verbindung mit Simulink die zentrale Entwicklungsumgebung, da Reglerentwurf, Simulation und Implementierung über einen konsistenten Modellkern verbunden werden können (Werth et al. 2020; Fang et al. 2009). Die Funktionsweise der modellbasierten Entwicklung mit Matlab und Simulink wird in Abschnitt 2.2 erläutert.

## 2.2 Modellbasierte Entwicklung mit Matlab/Simulink

Bei der modellbasierten Entwicklung in Matlab und Simulink steht ein ausführbares Systemmodell im Mittelpunkt des Engineeringprozesses. Anstatt frühzeitig eine reine Softwareimplementierung zu erstellen, wird das Systemverhalten zunächst durch Modelle beschrieben, die sowohl für Analysen als auch für Simulationen genutzt werden. Matlab wird dabei vor allem für Berechnungen, Auswertungen, Parametrierungen und Visualisierungen eingesetzt. Simulink ergänzt dies durch eine grafische Blockdiagrammumgebung, mit der dynamische Systeme strukturiert aufgebaut und simuliert werden können. Für physikalische Teilbereiche wie mechanische oder elektrische Komponenten lässt sich Simscape verwenden, um domänenübergreifende Modelle konsistent in einem gemeinsamen Rahmen zu formulieren.

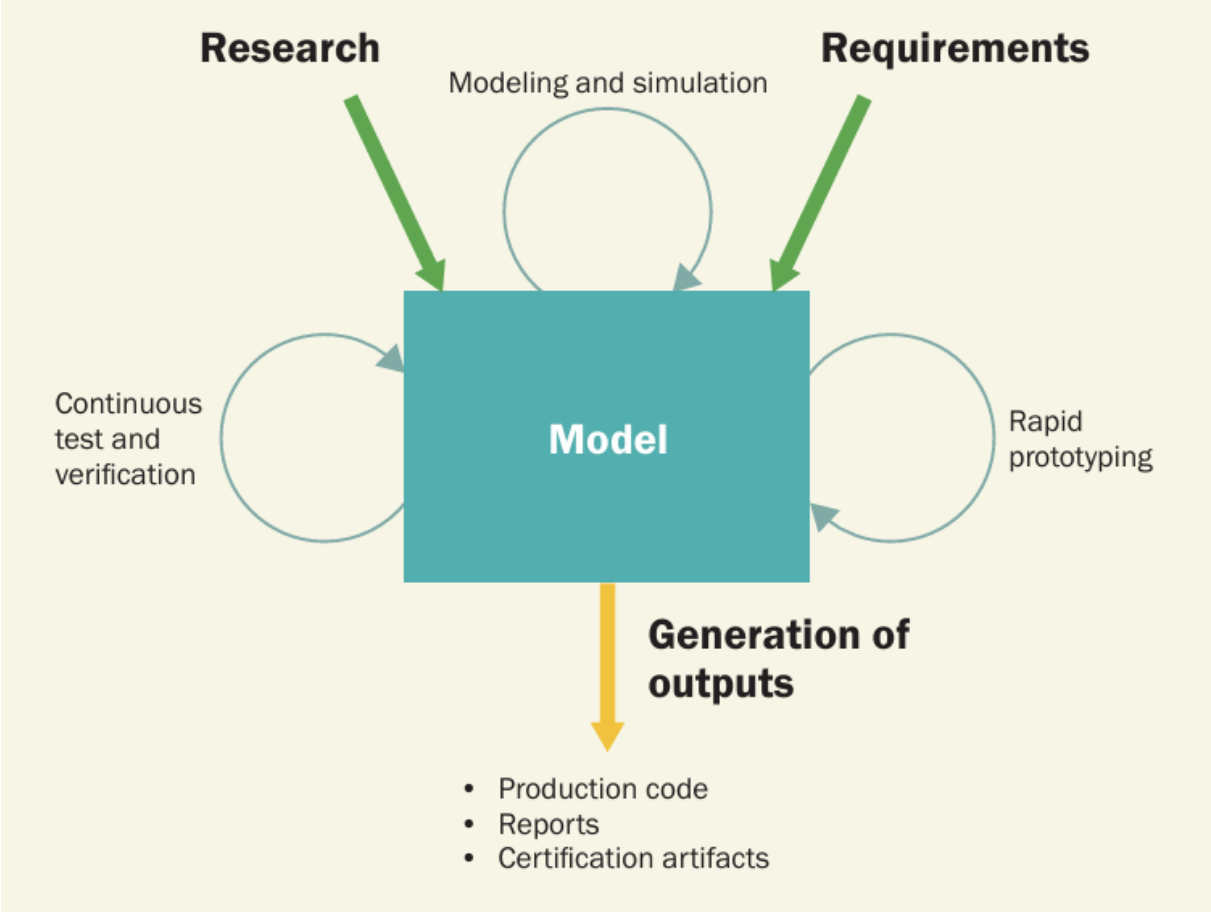


Abb. 2: Zentraler Workflow der modellbasierten Entwicklung(Aarenstrup 2025).

Die Abbildung 2 veranschaulicht den Grundgedanken der modellbasierten Entwicklung, bei der ein ausführbares Modell als zentrales Entwicklungsartefakt den gesamten Engineeringprozess strukturiert (Aarenstrup 2025). Im Mittelpunkt steht das *Model*, in dem sowohl die Anforderungen (*Requirements*) als auch Ergebnisse aus Voruntersuchungen und Domänenwissen (*Research*) zusammengeführt werden. Das Modell dient als gemeinsame Grundlage, aus der sich das Systemverhalten, die Schnittstellen und wichtige Entwurfsentscheidungen ableiten lassen, ohne dass zu

Beginn bereits eine separate Implementierung als Textcode im Fokus steht. Simulink unterstützt dies durch die grafische Modellierung dynamischer Systeme als Blockdiagramm, während Matlab für Berechnungen, Auswertungen, Parametrierungen und Visualisierungen genutzt wird. Für physikalische Teilmodelle kann Simscape eingesetzt werden, um mechanische, elektrische oder andere Domänen im gleichen Modell abzubilden.

Die in der Abbildung umlaufenden Rückkopplungen verdeutlichen, dass die Modellentwicklung nicht linear erfolgt, sondern iterativ organisiert ist. Der Zyklus *Modeling and simulation* steht für das wiederholte Aufbauen und Simulieren des Systems, um das zeitliche Verhalten zu analysieren und zentrale Eigenschaften wie Stabilität und Regelgüte unter variierenden Betriebsbedingungen zu bewerten. Die dabei gewonnenen Ergebnisse fließen in die Modellanpassung und Parametrierung zurück, wodurch die Modellgüte schrittweise erhöht wird. Matlab wird in dieser Phase insbesondere zur systematischen Auswertung von Simulationsergebnissen eingesetzt, etwa für Parameterstudien, Kennwertberechnungen und die grafische Darstellung relevanter Signale.

Der Kreis *Rapid prototyping* beschreibt den frühen Übergang vom Modell zur lauffähigen Ausführung, um Entwurfsentscheidungen unter realitätsnäheren Randbedingungen zu prüfen. Ein zentraler Baustein ist hierbei die Codegenerierung aus dem Simulink Modell, durch die automatisch ausführbarer Code erzeugt wird, der auf Zielhardware wie einem Mikrocontroller, Digital Signal Processor, deutsch: Digitaler Signalprozessor (DSP) oder Field Programmable Gate Array, deutsch: Feldprogrammierbare Gatter (FPGA) eingesetzt werden kann. Die Abbildung verdeutlicht damit, dass Implementierungsergebnisse nicht als unabhängige Parallelentwicklung entstehen, sondern als abgeleitete Artefakte aus dem zentralen Modell, wodurch die Konsistenz zwischen Entwurf und Ausführung erhöht wird.

Der dritte Kreis *Continuous test and verification* betont, dass Verifikation und Tests fortlaufend in den Entwicklungsprozess integriert sind. In diesem Kontext werden gestufte Testmethoden eingesetzt, um die Anforderungen systematisch abzusichern. Bei Software in the Loop, deutsch: Software im Regelkreis (SIL) wird das Regelungssystem gemeinsam mit der Strecke in einer simulierten Umgebung ausgeführt, während bei Hardware in the Loop, deutsch: Hardware im Regelkreis (HIL) eine reale Steuerungshardware mit einem in Echtzeit berechneten Streckenmodell gekoppelt wird. Diese Vorgehensweisen ermöglichen eine frühe Prüfung gegen Anforderungen und unterstützen die Identifikation von Abweichungen, bevor ein vollständiger Versuchsaufbau oder ein finales Gesamtsystem vorliegt.

Der nach unten gerichtete Pfeil *Generation of outputs* fasst zusammen, welche Ergebnisse aus dem Modell abgeleitet werden. Dazu zählen insbesondere Produktionscode, Berichte sowie Nachweisartefakte für eine spätere Dokumentation und gegebenenfalls Zertifizierung. Abbildung 2 macht damit deutlich, dass modellbasierte Entwicklung in Matlab und Simulink nicht nur die Simulation unterstützt, sondern einen durchgängigen Ablauf von der Anforderungsableitung über die iterative Modellentwicklung bis zur Implementierung und Absicherung bereitstellt. Für die vorliegende Arbeit ist dieser Ansatz zentral, da die Versuchsplattform und die Algorithmen in einer

einheitlichen Modellbasis entwickelt, prototypisch auf Zielhardware ausgeführt und anschließend experimentell validiert werden.

## 2.3 Inertialsensorik

Eine Inertialmesseinheit (IMU) ist ein Sensorsystem zur Erfassung von Bewegungen eines Körpers im Raum. Sie kombiniert Beschleunigungssensoren, die die spezifische Kraft messen, mit Drehratensensoren, die die Winkelgeschwindigkeit erfassen. Üblich ist eine Ausführung mit jeweils drei Sensoren pro Messgröße, deren empfindliche Achsen orthogonal angeordnet sind, sodass dreidimensionale Messungen in einem körperfesten Koordinatensystem bereitgestellt werden können (Groves 2015).

Die Messgrößen einer IMU werden typischerweise als zeitdiskrete Signale ausgegeben. Dazu versorgt die IMU die Sensoren, digitalisiert deren Ausgangssignale und überträgt die Messwerte über eine Datenschnittstelle. Die Ausgabe erfolgt häufig mit hohen Datenraten im Bereich von mehreren hundert Hertz, was insbesondere für dynamische Systeme in Regelungs und Navigationsaufgaben relevant ist (Groves 2015). Inhaltlich ist zu beachten, dass Beschleunigungssensoren nicht direkt die translatorische Beschleunigung im Sinne der Kinematik liefern, sondern die spezifische Kraft. Diese entspricht der durch Trägheit verursachten Beschleunigung relativ zu einem frei fallenden Bezug und enthält damit die Wirkung der Gravitation in der Messinterpretation (Groves 2015).

Aus den IMU Messungen kann mittels inertialer Navigation eine Schätzung von Position, Geschwindigkeit und Orientierung abgeleitet werden. Das Grundprinzip besteht darin, die gemessenen Winkelgeschwindigkeiten zur Aktualisierung der Orientierung zu integrieren. Mit der so bestimmten Orientierung lassen sich die gemessenen spezifischen Kräfte in ein erdfestes oder navigationsbezogenes Koordinatensystem transformieren. Nach Berücksichtigung eines Gravitationsmodells wird daraus die translatorische Beschleunigung bestimmt, die wiederum zu Geschwindigkeit und Position integriert wird (Groves 2015). Dieses Vorgehen wird in der Praxis als Strapdown Inertialnavigation umgesetzt, bei der eine IMU mit einem Navigationsprozessor zusammenwirkt, der die Navigationslösung aus spezifischer Kraft und Winkelgeschwindigkeit fortschreibt (Groves 2015).

Ein wesentliches Merkmal inertialer Systeme ist die Fehlerfortpflanzung durch Integration. Bereits kleine konstante Sensorabweichungen, etwa Bias, Skalierungsfehler oder Achsfehlstellungen, führen über die Zeit zu wachsenden Fehlern in Geschwindigkeit, Orientierung und Position. Das Tutorial beschreibt typische Biaswerte für unterschiedliche Qualitätsklassen von IMU Systemen und zeigt, dass insbesondere bei niedrigeren Sensorgüten deutliche Drift auftreten kann (Groves 2015). Für taktische Sensorik wird beispielsweise beschrieben, dass sich Positionsfehler im Bereich mehrerer Kilometer innerhalb etwa einer Stunde aufbauen können, wenn entsprechende Fehleranteile nicht kompensiert werden (Groves 2015). Daraus folgt, dass eine reine inertiale

Lösung für längere Zeiträume meist nur begrenzt geeignet ist, sofern keine zusätzlichen Informationsquellen zur Korrektur verfügbar sind.

Zur Begrenzung der Drift werden IMU basierte Lösungen häufig mit weiteren Sensorsystemen kombiniert. Das Tutorial beschreibt hierfür integrierte Navigationsarchitekturen, bei denen ein Schätzalgorithmus, typischerweise ein Kalmanfilter, die inertiale Lösung mit einem Hilfssystem wie **GNSS!** (**GNSS!**) vergleicht und daraus Korrekturen für Position, Geschwindigkeit und Orientierung ableitet. Die Kopplung ist so ausgelegt, dass auch bei zeitweilig fehlenden Hilfsmessungen weiterhin eine Navigationslösung vorliegt, während bei verfügbaren Messungen eine geschlossene Rückführung zur Fehlerreduktion erfolgt (Groves 2015). Für Anwendungen mit wiederkehrenden Stillstandsphasen oder Bewegungsrestriktionen können zusätzlich sogenannte Nullupdates oder Bewegungskonstraints genutzt werden, um den Zustand zu stabilisieren und Sensorfehler indirekt zu kalibrieren (Groves 2015).

## 2.4 Quaternionen und Euler-Winkel

"Lorem ipsum"

## 2.5 Lageschätzung mittels Sensorfusion

"Lorem ipsum"

# Literaturverzeichnis

**Aarenstrup, Roger (2025)**: Managing Model-Based Design. E-Book PDF, ISBN-10: 1512036137, Copyright 2015–2025 The MathWorks, Inc. Natick, MA: The MathWorks, Inc. ISBN: 978-1512036138. URL: https://www.mathworks.com/content/dam/mathworks/ebook/managing-model-based-design-2025.pdf (Abruf: 13. 01. 2026).

**Fang, Zheng; Qi, Yucheng; Zhang, Qichun; Chai, Tianyou (2009)**: Design and Implementation of a Low Cost DSP Based Rapid Control Prototyping System. In: In den Arbeitsunterlagen als PDF vorhanden. IEEE. DOI: 10.1109/ICIEA.2009.5138516. URL: https://doi.org/10.1109/ICIEA.2009.5138516 (Abruf: 12. 01. 2026).

**Groves, Paul D. (2015)**: Navigation Using Inertial Sensors. In: *IEEE Aerospace and Electronic Systems Magazine*. Tutorial, Part II of II. DOI: 10.1109/MAES.2014.130191. URL: https://doi.org/10.1109/MAES.2014.130191 (Abruf: 13. 01. 2026).

**Hoyos-Gutiérrez, Jose; Cardona-Aristizabal, Jaiber; Muñoz-Gutiérrez, Pablo Andrés; Ramirez-Jimenez, Diego (2023)**: A Systematic Literature Review on Rapid Control Prototyping Applications. In: *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*. IEEE Xplore Document 10056232. DOI: 10.1109/RITA.2023.3250559. URL: https://ieeexplore.ieee.org/document/10056232 (Abruf: 12. 01. 2026).

**Werth, Wolfgang; Faller, L.; Liechtenecker, H.; Ungermanns, Christoph (2020)**: Low Cost Rapid Control Prototyping a useful method in Control Engineering Education. In: In den Arbeitsunterlagen als PDF vorhanden. IEEE. DOI: 10.23919/MIPRO48935.2020.9245122. URL: https://doi.org/10.23919/MIPRO48935.2020.9245122 (Abruf: 12. 01. 2026).