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). Somit sind die wesentlichen Anforderungen an die Versuchsplattform, dass sie eine einfache und effiziente Entwicklung, Simulation und Echtzeitausführung von Regelungsalgorithmen ermöglicht, um die Untersuchung und Validierung von Regelungstechnischen Methoden zu unterstützen.

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

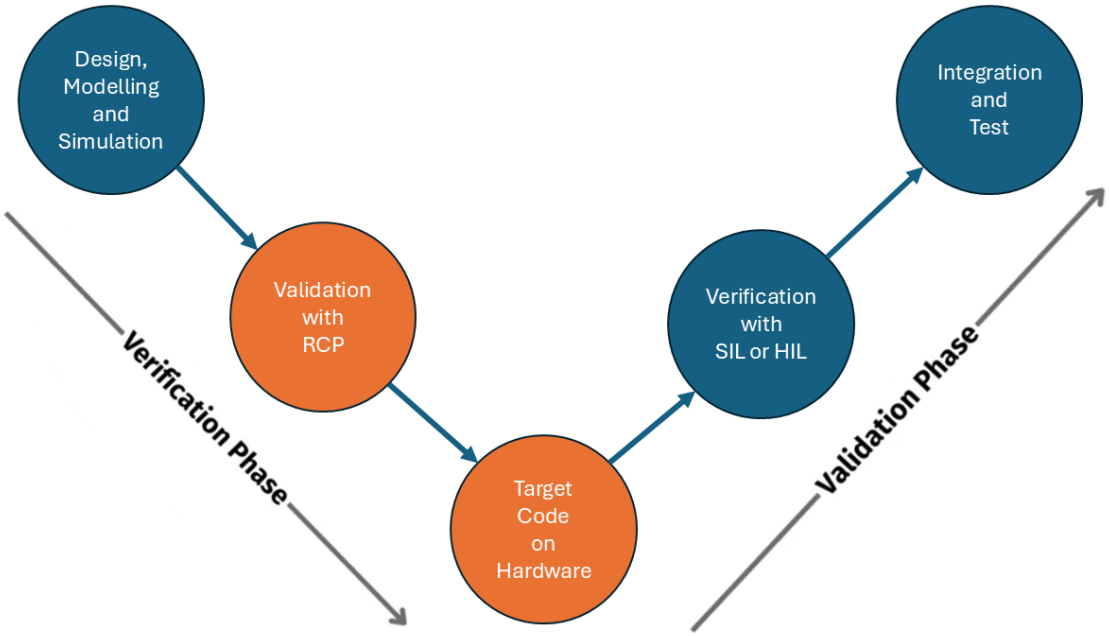

Abb. 1: V-Modell zur Einordnung des Rapid Control Prototyping (Hoyos-Gutiérrez et al. 2023).

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.

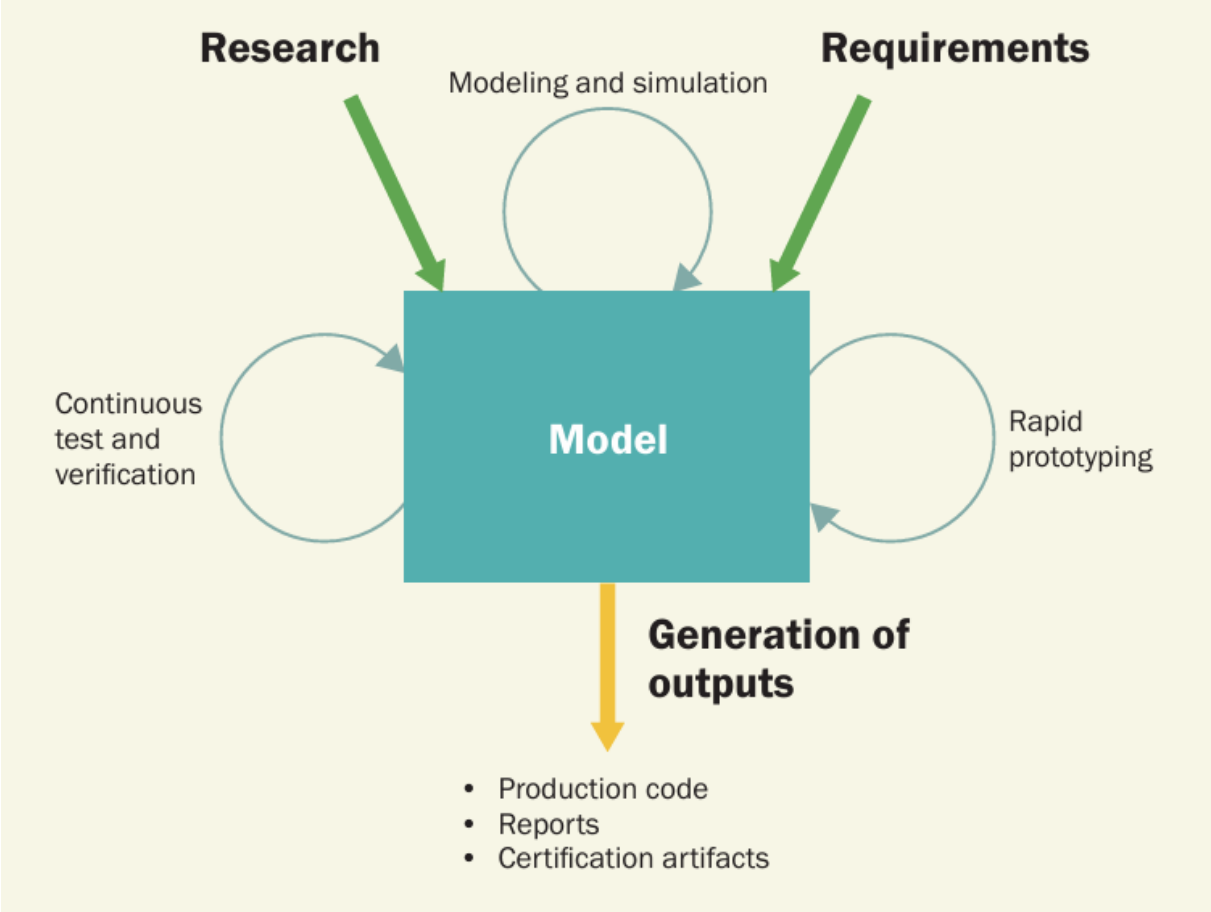

Abb. 2: Zentraler Workflow der modellbasierten Entwicklung(Aarenstrup 2025).

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 Inertialmesseinheit

Die Funktionsweise und die Anwendungsgebiete einer IMU werden im folgenden Abschnitt im Kontext der Lageschätzung als Anwendungsszenario der RCP Versuchsplattform näher erläutert.

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 IMU liefert ihre Messwerte typischerweise als zeitdiskrete Signale. Dazu werden die analogen Sensorausgänge intern aufbereitet, digitalisiert und anschließend über eine Datenschnittstelle ausgegeben. Die Messraten liegen häufig im Bereich mehrerer hundert Hertz und sind damit für dynamische Vorgänge geeignet, bei denen schnelle Änderungen der Bewegung erfasst werden müssen. Für die Interpretation der Signale ist wesentlich, dass ein Beschleunigungssensor nicht die kinematische translatorische Beschleunigung direkt misst, sondern die spezifische Kraft. Diese Größe ist relativ zu einem frei fallenden Bezug definiert und beinhaltet daher die Gravitation als dominanten Anteil, sobald keine starken linearen Beschleunigungen vorliegen. Das Gyroskop misst die Winkelgeschwindigkeit des Sensorsystems um die jeweiligen Achsen und stellt damit die Grundlage für die Beschreibung von Rotationsbewegungen dar (Groves 2015).

In praktischen Anwendungen werden die Rohmessungen einer IMU durch Sensoreigenschaften und Fehlerquellen beeinflusst. Dazu zählen insbesondere konstante oder langsam driftende Offsets, die als Bias beschrieben werden, Skalierungsfehler sowie Achsfehlstellungen und Kreuzkopplungen zwischen den Sensorkanälen. Zusätzlich wirkt stochastisches Rauschen auf die Messwerte, wodurch kurzzeitige Schwankungen entstehen. Diese Fehleranteile sind für die spätere Verarbeitung relevant, weil sich insbesondere Bias und Skalierungsfehler bei einer zeitlichen Integration oder bei länger andauernder Nutzung als Drift bemerkbar machen können. Entsprechend hängt die Qualität der bereitgestellten Bewegungsinformation stark von der Sensorgüte und von der Kalibrierung ab, mit der systematische Abweichungen reduziert werden.

Die Einsatzgebiete von IMUs ergeben sich aus der Fähigkeit, Bewegungen unabhängig von externen Referenzen zu erfassen. Ein zentrales Anwendungsfeld ist die Bestimmung der Orientierung und Bewegungsdynamik in technischen Systemen, insbesondere wenn optische oder satellitengestützte Referenzen nicht verfügbar oder nur eingeschränkt nutzbar sind. Demnach wird eineIMUs in Fahrzeugen, Fluggeräten und mobilen Robotern zur Erfassung von Drehraten, Neigungen und

Beschleunigungen eingesetzt, beispielsweise zur Stabilisierung, zur Regelung und als Eingangssignal für Navigations und Lokalisierungssysteme. Darüber hinaus finden sie Anwendung in der Industrieautomation, etwa zur Zustandsüberwachung bewegter Komponenten und zur Erfassung von Schwingungen oder Lageänderungen in Maschinen und Anlagen. Weitere typische Einsatzbereiche sind die Bewegungserfassung in tragbaren Systemen und Consumer Geräten, bei denen IMUs Gesten, Lageänderungen und Aktivitätsmuster erkennen und als Sensorbasis für Interaktions und Assistenzfunktionen dienen.

Die IMU stellt damit die grundlegenden Messgrößen für nachgelagerte Schätz und Regelalgorithmen bereit. Aus den Drehraten können Orientierungsänderungen abgeleitet werden, während die spezifische Kraft als Referenzinformation genutzt werden kann, wenn die Gravitation als dominanter Anteil vorliegt. Für eine robuste Lageschätzung wird die IMU in der Regel nicht isoliert betrachtet, sondern als Sensormodul, dessen Messwerte mit geeigneten Algorithmen weiterverarbeitet und mit Modellannahmen oder weiteren Sensoren kombiniert werden, um die Auswirkungen von Rauschen und Drift zu begrenzen.

Eine solche IMU soll in dieser Projektarbeit zunächst als Simulationsmodell abgebildet und untersucht werden und, sofern erforderlich, zusätzlich als reale Hardwarekomponente im Rahmen des Rapid Control Prototyping in die Versuchsplattform integriert werden.

2.4 Quaternionen und Euler-Winkel

"Lorem ipsum"

2.5 Lageschätzung mittels Sensorfusion

"Lorem ipsum"

3 Auswahl geeigneter Hardware

Im folgenden Kapitel wird die Auswahl der geeigneten Hardware für das (RCP) beschrieben. Dies gliedert sich in die Boardauswahl in Abschnitt 3.1 als auch die Auswahl der Sensorik in Abschnitt 3.2. Dabei werden verschiedene Kriterien in einer Pugh-Matrix bewertet, um die bestmögliche Lösung bei der Auswahl des Boardsfür die Anforderungen des Projekts zu finden.

3.1 Boardauswahl

Auf Basis der in Abschnitt 1.2 abgeleiteten Anforderungen werden verschiedene Hardwareplattformen für das RCP gegenübergestellt. Die hierfür verwendeten Bewertungskriterien sind in Tabelle 1 zusammengefasst.

Die Auswahlentscheidung erfolgt anschließend mithilfe einer Pugh-Matrix, in der die betrachteten Plattformen anhand dieser Kriterien und der Bewertungsskala aus Tabelle 1 bewertet werden, um die geeignetste Hardwareplattform für die Versuchsplattform zu bestimmen. Nachdem die Bewertungskriterien definiert wurden, erfolgt die Recherche und Auswahl verschiedener Hardwareplattformen, die für das RCP in Frage kommen. Diese Plattformen werden in der Pugh-Matrix in Tabelle 2 anhand der zuvor definierten Kriterien bewertet. Die Gesamtpunktzahl jeder Plattform wird berechnet, indem die Bewertungen mit den jeweiligen Gewichtungen multipliziert und summiert werden. Die Plattform mit der höchsten Gesamtpunktzahl wird als die am besten geeignete Hardwarelösung für die RCP-Versuchsplattform identifiziert. Hierbei ist jeoch wichittig zu beachten, dass neben der reinen Punktzahl auch praktische Aspekte wie Verfügbarkeit, Support und Kosten in die endgültige Entscheidung einfließen sollten, somit muss nicht nur die höchste Punktzahl zur Auswahl herangezogen werden. Die Eckdaten der betrachteten Hardwareplattformen sind in Tabelle 3 zusammengefasst, welche genutzt wurden um die Pugh-Matrix zu erstellen.

Die Evaluierung der letzendlichen Entwicklungsplattform ergibt, dass das *STM32H755ZI-Q* die beste Balance zwischen den Anforderungen und den Bewertungskriterien bietet. Es überzeugt durch eine gute Matlab/Simulink-Anbindung für RCP, ausreichende Rechenleistung und RAM, eine Dual-Core-Architektur zur funktionalen Trennung, sowie umfangreiche I/O-Fähigkeiten und Debugging-Optionen. Zudem ist es im Vergleich zu den professionellen Echtzeitsystemen von dSpace und Speedgoat deutlich kostengünstiger, während es dennoch die wesentlichen Anforderungen für die geplante Versuchsplattform erfüllt. Daher wird das *STM32H755ZI-Q* als die am besten geeignete Hardwareplattform für die Umsetzung der RCP-Versuchsplattform ausgewählt. Dies umfasst die evaluierung weshalb sich das *STM32H755ZI-Q* als Entwicklungsplattform am besten eignet für diese Projektarbeit. Allerdings wird im folgenden Abschnitt beschrieben, welche Gründe gegen die Nutzung der anderen Plattformen sprechen.

Ein weiterer vielversprechender Kandidat ist das *NXP i.MX RT1170*, welches ebenfalls eine Dual-Core-Architektur und gute Performance bietet. Allerdings sind die Simulink-Integrationsmöglichkeiten und der Support im Vergleich zum STM32 etwas eingeschränkt, was die Entwicklungsarbeit erschweren könnte. Die professionellen Echtzeitsysteme von dSpace und Speedgoat bieten zwar eine nahtlose Integration mit Simulink und umfangreiche Debugging-Optionen, sind jedoch aufgrund ihrer hohen Kosten für dieses Projekt nicht praktikabel. Kleinere Mikrocontroller-Plattformen wie der *Teensy 4.1* oder der *Arduino Portenta H7* bieten zwar gute I/O-Fähigkeiten und sind kostengünstig, jedoch fehlt es ihnen an offizieller Simulink-Unterstützung und ausreichender Rechenleistung für komplexe RCP-Anwendungen.

3.2 Sensorauswahl

Tab. 1: Kriterien und Bewertungsskala der Pugh-Matrix zur Boardauswahl

Kriterium	Beschreibung
Matlab/Simulink-Anbindung (RCP)	Unterstützung der modellbasierten Implementierung und automatischen Codegenerierung sowie Möglichkeit zur Parametrierung und Signalbeobachtung im Echtzeitbetrieb.
Performance (Rechenleistung, RAM)	Verfügbare Rechenleistung und Speicherressourcen für Sensorfusion, Filteralgorithmen und Echtzeitausführung.
Kernarchitektur (Single, Dual)	Verfügbarkeit und Nutzbarkeit einer Single-Core- oder Dual-Core-Architektur zur funktionalen Trennung, beispielsweise von Kommunikations- und Rechenaufgaben.
Debugging und Tracing (MIPI20, ETM)	Unterstützung von Debug- und Trace-Schnittstellen zur Laufzeitanalyse, Fehlerdiagnose und Performanzbewertung, beispielsweise über ETM oder MIPI20.
Verfügbarkeit und Support (Community, Hersteller)	Verfügbarkeit der Hardware, Qualität der Dokumentation sowie Community- und Herstellerunterstützung für Integration und Fehlersuche.
Preis und Lizenzkosten	Hardwarekosten sowie mögliche Zusatzkosten durch erforderliche Softwarelizenzen, Toolchains oder kommerzielle Erweiterungen.
I/O-Fähigkeiten (Analog, PWM, SPI, CAN)	Umfang und Leistungsfähigkeit der Ein- und Ausgänge für Sensorik, Aktorik und Bussysteme einschließlich analoger Kanäle und Kommunikationsschnittstellen.
ROS 2-Unterstützung	Möglichkeit zur Integration in ROS 2-Umgebungen.

Bewertungsskala

Symbol	Bedeutung
`++`	deutlich besser als Referenz (technisch führend)
`+`	besser als Referenz
`0`	vergleichbar
`-`	schlechter
`--`	deutlich schlechter

Tab. 2: Pugh-Matrix zur Bewertung von Hardwareplattformen für die RCP-Versuchsplattform

Kriterium	**Gewicht**	Arduino Giga R1	Nano RP2040	Raspberry Pi 4	Teensy 4.1	Portenta H7	STM32 H7S3L8	STM32 F767ZI	STM32 H755ZI-Q	STM32 H753XI	NXP i.MX RT1170	TI F2837D	STM32 MP157	Jetson Nano	dSpace	Speedgoat
Matlab/Simulink-Anbindung (RCP)	5	0	-	-	0	+	-	0	+	0	+	+	0	++	++	++
Performance (Rechenleistung, RAM)	5	0	-	+	+	+	++	0	++	+	++	0	++	++	++	++
Kernarchitektur (Single/Dual)	4	+	-	+	+	+	+	0	++	0	++	+	++	++	++	++
Debugging und Tracing (MIPI20/ETM)	2	0	-	0	+	+	+	0	+	0	+	+	0	++	++	++
Verfügbarkeit, Community, Support	4	+	++	++	++	+	0	++	+	++	+	0	+	+	-	-
Preis und Lizenzkosten	4	+	++	++	++	0	+	++	+	++	0	+	0	-	- -	- -
I/O-Fähigkeiten (Analog, PWM, SPI, CAN)	4	+	-	++	++	+	++	++	++	++	++	+	++	++	++	++
ROS 2-Unterstützung	2	-	-	++	0	0	-	-	-	-	0	-	+	++	++	++
Gesamtpunktzahl		**19**	**8**	**32**	**37**	**24**	**38**	**24**	**41**	**31**	**37**	**17**	**32**	**63**	**75**	**78**

Tab. 3: Eckdaten der betrachteten Hardwareplattformen einschließlich Preisangaben (aus der Pugh-Matrix abgeleitet)

Plattform	CPU und Architektur	Takt	RAM	I/O und Debugging	Hinweise und Preis
Arduino Giga R1	Dual-Core (Cortex-M7 + Cortex-M4)	480 MHz 240 MHz	1 MB	Viele I/O und PWM; kein ETM	Begrenzte Simulink-Unterstützung; kein ROS 2; ∼60 €.
Arduino Nano RP2040	Dual-Core (Cortex-M0+)	133 MHz	264 KB	I/O begrenzt; kein ETM	Keine Matlab-Unterstützung; kein ROS 2; ∼10 €.
Raspberry Pi 4	Quad-Core (Cortex-A72)	1.5 GHz	4–8 GB	GPIO, SPI, I^2C, UART; Linux-Debug	Simulink über ROS/UDP; ROS 2; ∼70 €.
Teensy 4.1	Single-Core (Cortex-M7)	600 MHz	1 MB	Viele I/O inkl. CAN, SPI; JTAG/SWD	Kein offizielles Simulink-Target; microROS möglich; ∼35 €.
Arduino Portenta H7	Dual-Core (Cortex-M7 + Cortex-M4)	480 MHz 240 MHz	8 MB SDRAM	Umfangreiche I/O inkl. Ethernet; ETM möglich	Matlab-Target via STM32; microROS teils; ∼100 €.
STM32 H7S3L8 (neueste)	Single-Core (Cortex-M7)	600 MHz	2 MB	CAN FD, ADC, SPI, PWM; MIPI20	Keine Simulink-Targets; Zugang begrenzt; ∼80 €.
STM32 F767ZI (Vor-Gen.)	Single-Core (Cortex-M7)	216 MHz	512 KB	Viele I/O, Ethernet; SWD	STM32-Packs möglich; kein ROS 2; ∼25 €.
STM32 H755ZI-Q	Dual-Core (Cortex-M7 + Cortex-M4)	480 MHz 240 MHz	n. a.	CAN FD, Ethernet; ETM	STM32-Target programmierbar; ∼80 €.
STM32 H753XI (Klassiker)	Single-Core (Cortex-M7)	400 MHz	2 MB	Viele I/O, FPU; SWD	Simulink via STM32Cube manuell; ∼60 €.
NXP i.MX RT1170	Dual-Core (Cortex-M7 + Cortex-M4)	1 GHz 400 MHz	2 MB	Ethernet, CAN FD; ETM/SWO	MATLAB-Target via SDK; EVK ∼120 €.
TI TMS320 F2837D	Dual-Core (C28x)	n. a.	n. a.	PWM, ADC, CAN, SPI, Ethernet; JTAG/ETM	C2000 in Simulink integriert; LaunchPad ∼80 €.
STM32 MP157 (SoC)	Dual-Core A7 + Cortex-M4	650 MHz	512 MB–1 GB	Viele IF (USB, Ethernet); Linux-Debug	ROS 2 via Linux; Dev-Board ∼100 €.
Nvidia Jetson Nano	Quad-Core ARM + GPU	n. a.	4 GB	Linux; Profiling (Nsight)	ROS 2 Image; Matlab über ROS 2; ∼130 €.
dSpace	Echtzeit-System (Multicore-RT)	1–2 GHz	n. a.	Echtzeit-Tracing; flexible I/O	Voll integriert; proprietär; >5000 €.
Speedgoat	Echtzeit-System (Multicore-RT)	bis 3 GHz	n. a.	Debugging mit Scope-Sync; flexible I/O	Nahtlos Simulink; ROS 2-Target; >8000 €.

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).