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
KF	Kalmanfilter
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

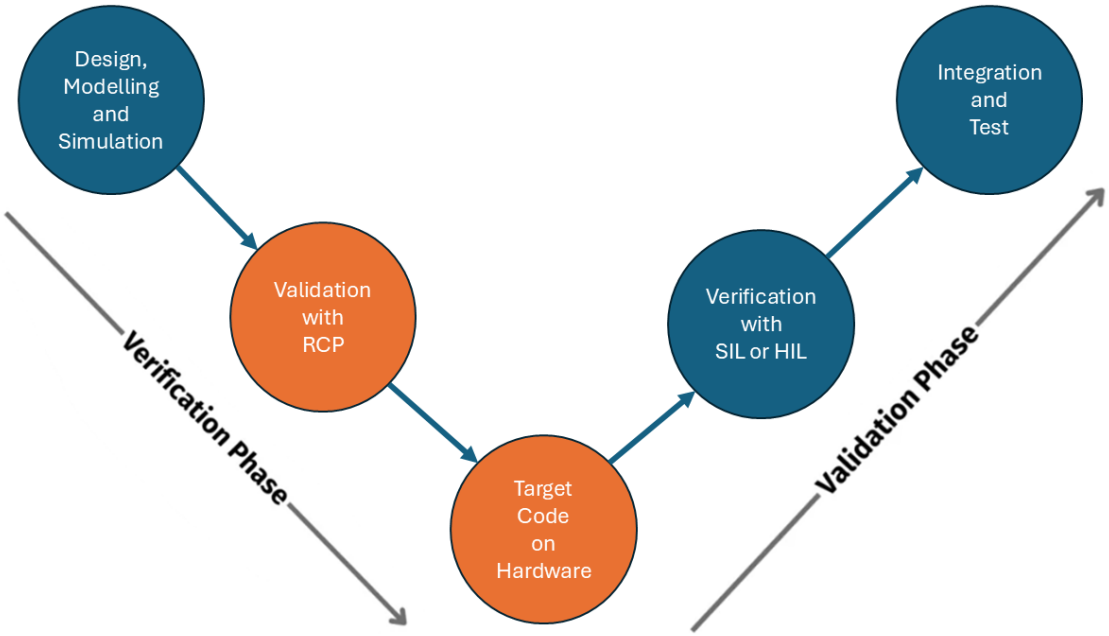

Abb. 1: V-Modell zur Einordnung des Rapid Control Prototyping (Hoyos-Gutiérrez et al. 2023).

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.

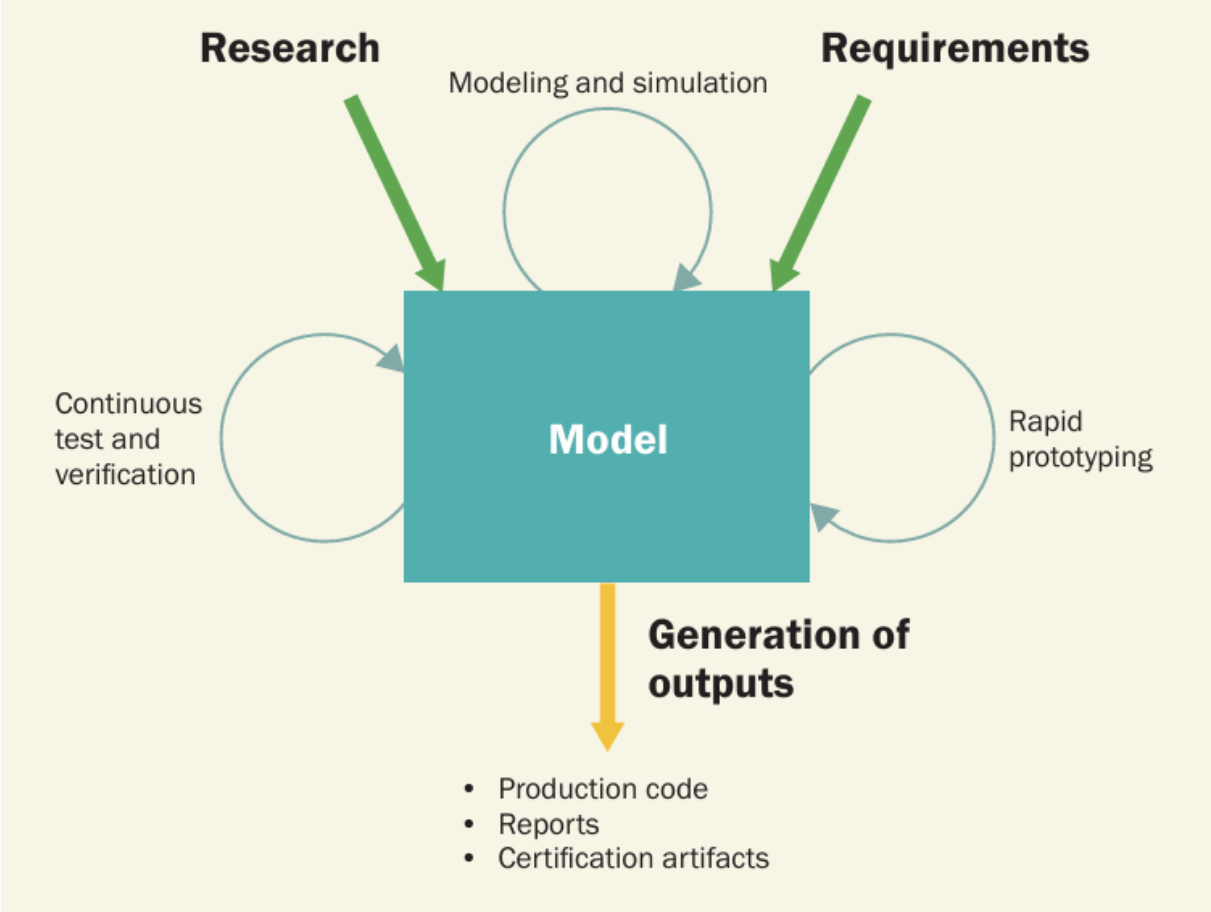

Abb. 2: Zentraler Workflow der modellbasierten Entwicklung(Aarenstrup 2025).

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 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

Quaternionen und Euler-Winkel sind mathematische Darstellungen, um Rotationen im dreidimensionalen Raum zu beschreiben. Solche Rotationsbeschreibungen werden unter anderem in der Lageschätzung von Inertialmesseinheiten (IMU) benötigt, um die Orientierung aus Sensormessgrössen zu modellieren und zu bewerten.
Grundsätzlich kommen hierfür sowohl Quaternionen als auch Euler-Winkel in Frage. Quaternionen eignen sich besonders für die interne Berechnung, da sie eine kompakte und singularitätsfreie Darstellung von Rotationen ermöglichen. Euler-Winkel werden hingegen häufig zur Ausgabe und Visualisierung verwendet, da sie die Orientierung anschaulich über drei aufeinanderfolgende Winkel beschreiben: *Yaw* (Gierwinkel), *Pitch* (Nickwinkel) und *Roll* (Rollwinkel). Abbildung 3 zeigt diese Rotationen um die körperfesten Achsen. *Roll* entspricht einer Rotation um die x-Achse, *Pitch* einer Rotation um die y-Achse und *Yaw* einer Rotation um die z-Achse.
Euler-Winkel besitzen jedoch Singularitäten und sind daher als interne Zustandsdarstellung nur eingeschränkt geeignet; in der Praxis werden sie meist für die Visualisierung der geschätzten Orientierung verwendet.

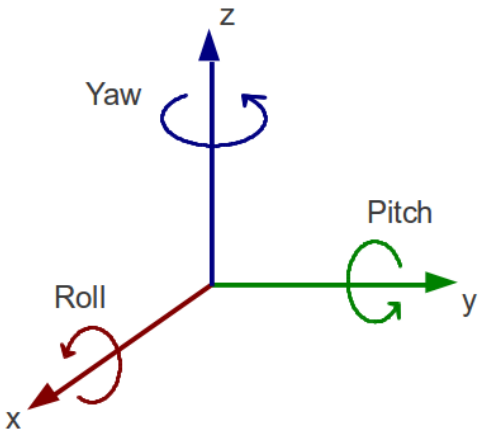

Abb. 3: Schematische Darstellung der Euler-Winkel(**ResearchGate**)

Für die konkrete mathematische Beschreibung ist zu beachten, dass Euler-Winkel stets an eine festgelegte Rotationskonvention gebunden sind. Da räumliche Rotationen nicht vertauschbar sind, führt eine andere Reihenfolge der Einzelrotationen zu einer anderen resultierenden Orientierung. Beispielsweise leifert die Abfolge *Yaw–Pitch–Roll* im Allgemeinen ein anderes Ergebnis als *Pitch–Yaw–Roll*, selbst wenn die gleichen Winkelwerte verwendet werden.
Neben dieser Konventionsabhänigkeit besitzen Euler-Winkel einen wesentlichen Nachteil: Sie weisen Singularitäten auf. Für die Rotationsreihenfolge *Roll–Pitch–Yaw* (XYZ) tritt der kritische Fall in der Nähe von $\theta = \pm 90°$ (Pitch) auf, da zwei Rotationsachsen dabei effektiv zusammenfallen. In diesem Bereich sind Roll und Yaw nicht mehr unabhängig bestimmbar, sodass bereits kleine Änderungen der tatsächlichen Orientierung zu großen oder sprunghaften Änderungen ein-

zelner Euler-Winkel führen können. Dieses Verhalten wird als Gimbal-Lock bezeichnet.(**KIT**)

Um diese Einschränkung zu vermeiden, wird die Orientierung in Filteralgorithmen **Verweis zu 4.2?** intern nicht über Euler-Winkel angegeben, sondern durch Quaternionen beschrieben. Diese Darstellung ist frei von Singularitäten und eignet sich für eine robuste Zustandsführung.
Quaternionen sind eine Erweiterung der komplexen Zahlen und lassen sich allgemein in der Form

$$q = a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k} \tag{2.1}$$

darstellen, wobei $a, b, c, d \in \mathbb{R}$ und $\mathbf{i}, \mathbf{j}, \mathbf{k}$ imaginäre Einheiten sind. Für die Beschreibung von Rotationen im 3-dimensionalen Raum werden in der Praxis Einheitsquaternionen verwendet, die häufig in Vektorform als

$$q = \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^\mathsf{T} = \begin{bmatrix} q_w & \mathbf{q}_v \end{bmatrix}^\mathsf{T} \tag{2.2}$$

geschrieben werden. Dabei bezeichnet q_w den Skalaranteil und

$$\mathbf{q}_v = \begin{bmatrix} q_x & q_y & q_z \end{bmatrix}^\mathsf{T} \tag{2.3}$$

den Vektoranteil. Eine Rotation lässt sich mit Quaternionen besonders anschaulich über das Achse-Winkel-Konzept darstellen: Jede räumliche Drehung kann als Drehung um eine bestimmte Achse mit einem bestimmten Winkel beschrieben werden. Quaternionen verpacken diese Information kompakt, indem der Skalarteil den Winkelanteil und der Vektorteil die Drehachse enthält. Dadurch lassen sich Rotationen effizient speichern.
Ein weiterer praktischer Vorteil ist, dass sich aufeinanderfolgende Rotationen einfach verketten lassen. Wird eine Orientierung nacheinander um mehrere Rotationen verändert, kann die Gesamtrotation durch Kombination der zugehörigen Quaternionen berechnet werden. Die Hintereinanderausführung von Rotationen entspricht dabei der Quaternion-Multiplikation. Dadurch ist es möglich, Orientierungsänderungen schrittweise zu aktualisieren. **Rotationsmatrix einfügen, falls später verwendet!** (Clemens 2015, S.25-28)

Da die interne Zustandsdarstellung als Quaternionen erfolgt, die Ergebnisse jedoch zur Interpretaion als Euler-Winkel angegeben werden, ist eine konsistente Umrechnung zwischen beiden Darstellungen erforderlich. Durch Gleichsetzen der Rotationsbeschreibungen

$$R(\phi, \theta, \psi) = R_x(\phi)\, R_y(\theta)\, R_z(\psi) \tag{2.4}$$

und den anschließenden Vergleich der Matrixelemente, ergeben sich folgenden Umrechnungsformeln von Quaternionen zu Euler-Winkeln in Roll-Pitch-Yaw (XYZ) Reihenfolge. Im Folgenden werden die resultierenden Ausdrücke direkt angegeben; eine vollständige Herleitung findet sich in der Literatur.(Diebel 2006, S.24)

$$\phi = \text{atan2}\big(2(q_w q_x + q_y q_z),\, 1 - 2(q_x^2 + q_y^2)\big) \tag{2.5}$$

$$\theta = \arcsin(2(q_w q_y - q_z q_x)) \tag{2.6}$$

$$\psi = \text{atan2}\big(2(q_w q_z + q_x q_y),\, 1 - 2(q_y^2 + q_z^2)\big) \tag{2.7}$$

2.5 Lageschätzung mittels Sensorfusion

Sensorfusion bezeichnet die Kombination von Messdaten mehrerer Sensoren, um die Qualität der Zustandsinformation zu erhöhen. Durch die Nutzung komplementärer Sensoreigenschaften lassen sich Unsicherheiten reduzieren und konsistente Schätzwerte gewinnen. **In dieser Arbeit steht die Orientierungsschätzung (Attitude) einer IMU im Vordergrund**. Da kein Magnetometer eingesetzt wird, basiert die Schätzung auf Gyroskop- und Beschleunigungssignalen. Dadurch können Roll und Pitch langfristig stabilisiert werden, während der Yaw-Winkel ohne externe Referenz grundsätzlich driftbehaftet bleibt.
Eine robuste Orientierungsbestimmung ist mit einem einzelnen Sensor nicht zuverlässig möglich, da die Messgrößen jeweils systematische Einschränkungen besitzen. Das Gyroskop misst die Winkelgeschwindigkeit und erlaubt eine hochdynamische Fortschreibung der Orientierung. Die Messung ist jedoch durch Rauschen beeinflusst und weist typischerweise einen Bias auf.
Unter Bias wird ein systematischer, meist langsam zeit- und temperaturabhängiger Offset verstanden, der dazu führt, dass auch im Stillstand eine von Null abweichende Winkelgeschwindigkeit gemessen wird. Da die Orientierung aus der Winkelgeschwindigkeit durch Integration berechnet wird, führt ein solcher Offset zu einem über die Zeit anwachsenden Fehler und damit zu Drift.
Der Beschleunigungssensor misst die spezifische Kraft. In quasi-statischen Situationen kann daraus die Richtung der Gravitation als Referenz abgeleitet werden, wodurch insbesondere Roll und Pitch langfristig stabilisiert werden. Bei dynamischen Bewegungen überlagern zusätzliche translatorische Beschleunigungen den Gravitationsanteil, sodass die Gravitation nicht mehr eindeutig aus der Messung bestimmbar ist. Aus diesem Grund wird die beschleunigungsbasierte Korrektur in Fusionsverfahren häufig situationsabhängig gewichtet oder bei erkennbar starker Dynamik unterdrückt, um Fehlkorrekturen zu vermeiden.
Die grundlegende Idee der Sensorfusion aus Gyroskop und Beschleunigungssensor besteht in der Kombination beider Eigenschaften. Die Gyroskopintegration liefert eine kurzzeitig genaue Orientierung, driftet jedoch über die Zeit. Die Beschleunigung liefert eine langfristige Referenz, ist aber bei Dynamik verfälschbar. Alle betrachteten Fusionsverfahren folgen dabei dem Prinzip Prädiktion und Korrektur: In der Prädiktion wird die Orientierung aus der Gyroskopmessung fortgeschrieben. In der Korrektur wird die aus der aktuellen Orientierungsschätzung erwartete Gravitationsrichtung mit der Beschleunigungsmessung verglichen. Die daraus resultierende Abweichung wird genutzt, um den Driftanteil der Gyro-Integration zu reduzieren.
?Auf Basis dieser gemeinsamen Struktur werden im Folgenden unterschiedliche Verfahren zur Sensorfusion vorgestellt. Abschnitt 2.5.1 skizziert die prinzipielle Funktionsweise eines (E)KF zur Orientierungsschätzung, ohne eine vollständige Herleitung vorzunehmen, da ein bestehender EKF verwendet wird. Abschnitt 2.5.2 beschreibt komplementärfilterbasierte Ansätze als recheneffiziente Alternative, bevor in Abschnitt 2.5.3 der Mahony-Filter als spezielle Ausprägung des Komplementärfilters eingeordnet und die Auswahl begründet wird. (von Rosenberg 2006, S.25-27)

2.5.1 Erweitertes Kalmanfilter

Der Kalmanfilter (KF) ist ein rekursiver Zustandsschätzer für dynamische Systeme, der aus verrauschten Messungen einen möglichst guten Schätzwert des Systemzustands bestimmt. Er basiert auf einem Zustandsraummodell mit Prozessgleichung und Messgleichung und kombiniert dabei Modellwissen mit Sensordaten. Der klassische KF setzt ein lineares Systemmodell sowie lineare Messgleichungen voraus. Der Algorithmus besteht aus zwei wiederkehrenden Schritten: In der Prädiktion wird der Zustand mit dem Systemmodell fortgeschrieben und die zugehörige Unsicherheit (Kovarianz) propagiert. In der Korrektur wird die Vorhersage mithilfe der Messung aktualisiert, wobei der Kalman-Gain die Gewichtung zwischen Modell und Messung bestimmt.
Für die IMU-basierte Orientierungsschätzung ist der klassische KF jedoch nur eingeschränkt geeignet, da die zugrunde liegenden Zusammenhänge nichtlinear sind. Die Orientierungskinematik wird über Quaternionen bzw. Rotationsmatrizen beschrieben und ist nicht linear in den Zuständen. Eine direkte Anwendung des linearen KF würde daher die Modellrealität nur unzureichend abbilden oder starke Vereinfachungen erfordern, was zu ungenauen oder instabilen Schätzergebnissen führen kann.
Aus diesem Grund wird in der Praxis ein Erweiterter Kalmanfilter (EKF) eingesetzt. Das EKF überträgt das Prinzip des Kalmanfilters auf nichtlineare Systeme, indem nichtlineare Prozess- und Messmodelle verwendet und in jedem Zeitschritt um den aktuellen Arbeitspunkt linearisiert werden. Die Linearisierung erfolgt über die Jacobi-Matrizen der Prozess- und Messfunktionen. Dadurch bleibt der rekursive Aufbau aus Prädiktion und Korrektur erhalten, während die für die Orientierungsschätzung notwendigen nichtlinearen Rotationsbeziehungen berücksichtigt werden.

ABBILDUNG MA Vgl. Kim(2011) (Kim 2011, S. 146)

Die Abbildung **ref** zeigt den Ablauf des erweiterten Kalmanfilters (EKF) als rekursiven Algorithmus. Das EKF berechnet in jedem Zeitschritt k aus dem vorherigen Schätzwert und der aktuellen Messung einen verbesserten Schätzwert des Zustands. Der Algorithmus lässt sich in in vier Teile aufspalten:

0) **Initialisierung**
Zu Beginn werden ein Startwert $\hat{x}_0$ sowie die zugehörige Fehlerkovarianz P_0 festgelegt. $\hat{x}_0$ stellt die erste Zustandsabschätzung dar, P_0 beschreibt die anfängliche Unsicherheit dieser Schätzung.

I) **Prädiktion von Zustand und Fehlerkovarianz**
Im ersten Schritt wird aus dem vorherigen Schätzwert $\hat{x}_{k-1}$ eine a-priori-Schätzung $\hat{x}_k^-$ gebildet:

$$\hat{x}_k^- = f(\hat{x}_{k-1})\,. \tag{2.8}$$

Gleichzeitig wird die Unsicherheit dieser Vorhersage fortgepflanzt:

$$P_k^- = A\,P_{k-1}\,A^\top + Q. \tag{2.9}$$

Dabei beschreibt A die Linearisierung (Jacobi-Matrix) des Systemmodells um den aktuellen Arbeitspunkt, und Q modelliert Prozessrauschen. P_k^- charakterisiert damit die Unsicherheit der a-priori-Schätzung.

II) **Berechnung des Kalman-Gains**
Im zweiten Schritt wird der Kalman-Gain K_k berechnet:

$$K_k = P_k^- H^\top \left(H P_k^- H^\top + R \right)^{-1}. \tag{2.10}$$

Der Kalman-Gain legt fest, wie stark die Messung im nächsten Schritt zur Korrektur herangezogen wird. H ist die Linearisierung (Jacobi-Matrix) der Messfunktion, R beschreibt das Messrauschen der Sensorik.

III) **Berechnung der korrigierten Schätzung (Update)**
Nun wird die a-priori-Schätzung mit der aktuellen Messung z_k korrigiert:

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - h\big(\hat{x}_k^-\big) \right). \tag{2.11}$$

Der Term $z_k - h\big(\hat{x}_k^-\big)$ wird als **Innovation** bezeichnet. Er beschreibt die Abweichung zwischen der tatsächlichen Messung z_k und der aus der Vorhersage erwarteten Messung $h\big(\hat{x}_k^-\big)$. Diese Abweichung wird mit K_k gewichtet und zur Vorhersage addiert.

IV) **Aktualisierung der Fehlerkovarianz**
Abschließend wird die Unsicherheit nach dem Update aktualisiert:

$$P_k = P_k^- - K_k H P_k^-. \tag{2.12}$$

Damit liegt für den aktuellen Zeitschritt k sowohl der korrigierte Zustandsvektor $\hat{x}_k$ als auch die zugehörige Fehlerkovarianz P_k vor. Diese Größen dienen im nächsten Zeitschritt wieder als Ausgangspunkt, wodurch der EKF kontinuierlich in einer Schleife arbeitet. (Michaelsen 2018)

2.5.2 Komplementärfilter

Der Komplementärfilter ist ein einfaches Verfahren der Sensorfusion, das zwei Messgrößen mit komplementären Fehler- und Frequenzeigenschaften kombiniert, um eine robuste Gesamtschätzung zu erhalten. Aufgrund der geringen algorithmischen Komplexität wird er häufig in Embedded-Systemen eingesetzt, wenn eine Echtzeitschätzung bei begrenzten Rechenressourcen erforderlich ist.

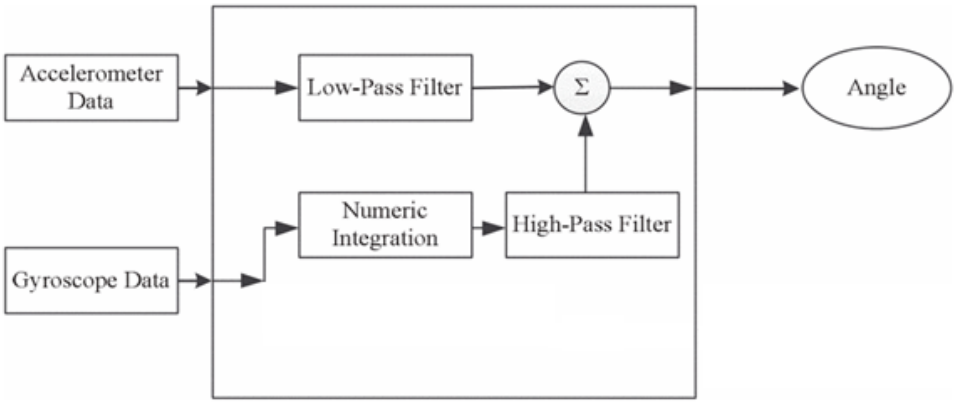

Abb. 4: Blockdiagramm zur Veranschaulichung der Funktionsweise eines Komplementärfilter(Gui, Tang, Mukhopadhyay 2015)

Anhand von Abbildung 4 wird im Folgenden das Funktionsprinzip des Komplementärfilters erläutert. Der Komplementärfilter fusioniert die Messinformationen von Beschleunigungssensor und Gyroskop, indem beide Signalpfade gezielt in unterschiedlichen Frequenzbereichen genutzt werden. Wie in der Abbildung dargestellt, wird aus den Beschleunigungsdaten zunächst eine Neigungsinformation abgeleitet, die anschließend durch einen Tiefpass gefiltert wird. Dieser Pfad trägt damit vor allem die niederfrequenten Anteile der Orientierungsschätzung und liefert insbesondere eine langfristig stabile Referenz über die Schwerkraftrichtung.

Parallel dazu wird die Winkelgeschwindigkeit des Gyroskops numerisch integriert, um eine Winkelschätzung zu erhalten. Diese Schätzung bildet hochfrequente, dynamische Orientierungsänderungen zuverlässig ab, ist jedoch ohne Korrektur driftbehaftet. Durch Hochpassfilterung werden die niederfrequenten Driftanteile unterdrückt, während die hochfrequenten Anteile erhalten bleiben.

Beide Anteile werden im Summationsblock zusammengeführt. Durch die komplementäre Aufteilung in Tiefpassanteil (Beschleunigung) und Hochpassanteil (Gyroskop) entsteht eine Winkelschätzung, die kurzfristig dynamikfähig ist und gleichzeitig langfristig stabilisiert wird. In diskreter Form lässt sich dieses Prinzip exemplarisch durch

$$\theta_k = \alpha \left(\hat{\theta}_{k-1} + \omega_k \Delta t \right) + (1 - \alpha)\, \theta_{\mathrm{acc},k} \tag{2.13}$$

beschreiben. Dabei bezeichnet $\hat{\theta}_k$ den geschätzten Winkel zum Zeitpunkt k, ω_k die gemessene Winkelgeschwindigkeit, Δt die Abtastzeit und $\theta_{\mathrm{acc},k}$ den aus dem Beschleunigungssignal abgeleiteten Winkel. Der Parameter $\alpha \in (0, 1)$ bestimmt die Aufteilung zwischen beiden Informationsquellen. Große α-Werte gewichten die Gyro-Integration stärker, was eine gute Dynamik ermöglicht, jedoch die Driftkorrektur reduziert. Kleinere α-Werte erhöhen die niederfrequente Stabilisierung durch den Beschleunigungssensor. Dadurch wird die Drift besser unterdrückt, gleichzeitig steigt jedoch die Empfindlichkeit gegenüber dynamischen Zusatzbeschleunigungen. (Gui, Tang, Mukhopadhyay 2015)

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).

Clemens, Joachim (2015): Entwicklung eines Subsystems „Sensorfusion“ für Navigation und Operatorsupport. In: URL: https://edocs.tib.eu/files/e01fb16/87093077X.pdf (Abruf: 12. 01. 2026).

Diebel, James (2006): Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors. In: URL: https://pasta.place/Informatik/Robotik_1_%5BHIS%5D/Folien/WS_17-18/01-Mathematische-Grundlagen.pdf (Abruf: 12. 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).

Gui, Pengfei; Tang, Liqiong; Mukhopadhyay, Subhas C. (2015): MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion. In: *Proceedings of the 2015 10th IEEE Conference on Industrial Electronics and Applications (ICIEA 2015)*. IEEE, S. 2004–2009. DOI: 10.1109/ICIEA.2015.7334442. URL: https://ieeexplore.ieee.org/document/7334442 (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).

Kim, Phil (2011): Kalman Filter for Beginners: With MATLAB Examples. In: *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*. (Abruf: 13. 01. 2026).

Michaelsen, Tobias (2018): Lagebestimmung durch Sensorfusion mittels Kalmanfilter. Masterarbeit. Hamburg, Deutschland: Universität Hamburg. URL: http://hdl.handle.net/20.500.12738/8460 (Abruf: 13. 01. 2026).

Von Rosenberg, Harald (2006): Sensorfusion zur Navigation eines Fahrzeugs mit low-cost Inertialsensorik. Public version (PDF). Diplomarbeit. Stuttgart, Deutschland: Universität Stuttgart. URL: https://www.vrosenberg.de/download/Diplomarbeit%20Harald%20von%20Rosenberg%20%28public%20version%29.pdf (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).