



hochschule mannheim

**Konstruktion, Aufbau und Inbetriebnahme
einer sechsbeinigen Laufmaschine unter
Verwendung inverser Kinematik**

Wilen Askerow

Bachelor-Thesis
zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.)
Studiengang Mechatronik

Fakultät für Elektrotechnik
Hochschule Mannheim

10.08.2014

Betreuer

Prof. Dr. Thomas Ihme, Hochschule Mannheim
Prof. Dr. Jörn Fischer, Hochschule Mannheim

Askerow, Wilen:

Konstruktion, Aufbau und Inbetriebnahme einer sechsbeinigen Laufmaschine unter Verwendung inverser Kinematik / Wilen Askerow. –

Bachelor-Thesis, Mannheim : Hochschule Mannheim, 2014. 64 Seiten.

Askerow, Wilen:

Design, mechanical structure, commissioning of six-legged mobile robot using inverse kinematics / Wilen Askerow. –

Bachelor-Thesis, Mannheim : University of Applied Sciences Mannheim, 2014. 64 pages.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mannheim, 10.08.2014

Wilen Askerow

Abstract

Konstruktion, Aufbau und Inbetriebnahme einer sechsbeinigen Laufmaschine unter Verwendung inverser Kinematik

In letzter Zeit verstärken sich die Forschungsinteressen vieler Wissenschaftler im Bereich mobiler Robotern. Durch die neuesten Entwicklungen auf den Gebieten der Antriebstechnik und Mikroelektronik entstehen neue und verbesserte Konzepte von Laufmaschinen. Die Erforschung fokussiert sich dabei meistens auf die Untersuchung der grundlegenden Lokomotionsprozesse.

Die Ziele der vorliegenden Arbeit sind die nun folgenden: Konstruktionsoptimierung, mechanischer Aufbau und die Inbetriebnahme des Laufroboters. Die Arbeit basiert auf einem existierenden Konstruktionskonzept und bereits ausgewählter Komponenten aus vorhergehenden Arbeiten für die sechsbeinige Laufmaschine. Hierbei wird auf die Entwicklung und Implementierung des Steuerungsalgorithmus und der Methoden zur Generierung der ballistischen Bewegungstrajektorien unter Verwendung inverser Kinematik eingegangen. Um ein deterministisches Verhalten der Steuerung zu ermitteln, müssen die analytisch beschriebenen und implementierten Roboterbewegungen untersucht werden. Die Umsetzung der Steuerung des Laufroboters erfolgt mit Hilfe des ROS Frameworks auf einem "Raspberry Pi" Einplatinencomputer.

Der Laufroboter soll als Experimentierplattform dafür dienen, um ferner Experimente auf den Gebieten der Künstlichen Intelligenz und Softwaretechnik zu ermöglichen.

Design, mechanical structure, commissioning of six-legged mobile robot using inverse kinematics

In recent times, interest in research in the area of mobile robots has increased significantly. Improved concepts for mobile robots arise through new developments in the areas of drive technology and microelectronics. The research mostly focuses on the study of fundamental processes in locomotive systems.

The thesis is based on an existing design concepts and selected components from previous student research project about six-legged mobile robot, its design optimization, mechanical structure and its commissioning. Additionally, the thesis addresses the issues of development and implementation of control algorithm and methods for generating ballistic movement trajectories using inverse kinematics.

In order to identify a deterministic behaviour of the control system, the analysis of the described and implemented movement of the robot must be examined. The implementation of the control system of the mobile robot takes place by means the ROS framework of a “Raspberry Pi” single board computer.

Consequentially, the mobile robot should serve as an experimental platform for further experiments in the areas of artificial intelligence, software engineering and design of secured algorithms.

Vorwort

Die vorliegende Bachelorarbeit stellt den Abschluss meines Studiums der Mechatronik an der Hochschule Mannheim dar. Sie wurde im Institut für Robotik der Fakultät Informatik angefertigt.

Mein besonderer Dank gilt dem Leiter des Instituts für Robotik der Hochschule Mannheim Herrn Prof. Dr. Thomas Ihme für die fachliche Kompetenz, das Engagement in der Materialbeschaffung und der Unterstützung in allen Angelegenheiten während der Anfertigung dieser Arbeit.

Ein weiterer Dank geht an Herrn Prof. Dr. Fischer für seine Bereitschaft zur Betreuung dieser wissenschaftlichen Arbeit.

Ebenso gilt mein Dank dem Team des Instituts für Robotik für die gute Atmosphäre innerhalb des Teams.

Inhaltsverzeichnis

Vorwort	iv
1 Einleitung	1
1.1 Motivation	2
1.2 Ziel und Aufbau der Arbeit	3
1.3 Stand der Technik	4
2 Grundlagen	6
2.1 Planung von Laufmustern	6
2.2 Kinematische Ketten	8
2.2.1 Denavit-Hartenberg Verfahren	9
2.2.2 Direkte Kinematik	11
2.2.3 Inverse Kinematik	13
2.3 Robot Operating System	15
2.3.1 Grundkonzept	16
2.3.2 Nachrichtenaustausch	17
2.3.3 Kommunikationsaufbau	19
2.3.4 Organisation	20
3 Grundlegender Aufbau des Laufroboters	22
3.1 Mechanischer Aufbau	23
3.1.1 Körper des Laufroboters	24
3.1.2 Beine des Laufroboters	25
3.2 Steuerrechner	27
3.3 Sensoren	28
3.4 Motoren	29
4 Entwicklung der Steuerungsalgorithmen	31
4.1 Beinprojektion	32
4.2 Koordinatensysteme am Laufroboter	35
4.2.1 Definition der Koordinatensysteme	35
4.2.2 Beziehungen der Koordinatensysteme zueinander	37
4.2.3 Kinematik des Roboterkörpers	38
4.2.4 Roboterbeine	39
4.3 Nummerierung der Motoren	45

5 Implementierung des Steueralgorithmus	46
5.1 Akrobat Paket	47
5.2 Steuerungskonzept	52
6 Erprobung der Laufmaschine	58
6.1 Erprobung des Laufens	60
6.2 Erprobung der Körperbewegungen	61
7 Zusammenfassung	63
Abkürzungsverzeichnis	vii
Tabellenverzeichnis	viii
Abbildungsverzeichnis	ix
Literaturverzeichnis	xi

Kapitel 1

Einleitung

Die faszinierenden Prozesse der Natur erregten schon immer das Interesse der Menschen. Durch Beobachtung und Erkundung der Natur, insbesondere der Insektenwelt, konnten Naturwissenschaftler, Ingenieure und Architekten in den letzten Jahrhunderten viele innovative Ideen gewinnen, die zu vielen Technologievorsprüngen beitrugen. So entstand das interdisziplinäre Forschungsfeld Bionik [21], in dem die Designs und Prozesse der Natur studiert werden, um von diesen zu lernen und diese Prozesse zu emulieren. Die Forschungserkenntnisse ermöglichen auf Grundlage der Anatomie und der biomechanischen Merkmale der Insekten, neue und verbesserte Konzepte für Konstruktion und Lauftechnologien für mehrbeinige Laufmaschinen zu entwickeln [3]. Darüber hinaus versprechen die Vorteile der natürlichen Vorbilder eine Leistungsverbesserung und höhere Flexibilität der Bewegung für die neuen Fortbewegungstechnologien [24].

Dabei ist das Ziel eine vollkommen autonome Laufmaschine zu entwickeln, die in der Lage ist selbstständig Entscheidungen zu treffen und komplexere Aufgaben zu lösen. In vielen Bereichen nimmt der Einsatz von Maschinen zu. Vom Menschen nicht oder nur beschränkt ausführbare Tätigkeiten, können so überhaupt ausgeführt werden: schneller und präziser. Das betrifft insbesondere das Erkunden bisher unerschlossener Gebiete sowohl auf unserem als auch auf anderen Planeten mit ungünstigen klimatischen Bedingungen.

1.1 Motivation

Neben dem Einsatz der Industrieroboter in der Produktionstechnik, die kaum mehr wegzudenken ist, verstärken sich in letzter Zeit die Forschungsinteressen vieler Wissenschaftler an mobilen Robotern. Die neuesten Entwicklungen auf dem Gebiet der Mikroelektronik und der Antriebstechnik erweitern die Konzeptmöglichkeiten der mobilen Roboter. Diese unterscheiden sich bezüglich der Fortbewegung in die Gruppe der rollenden (Räder, Ketten, Walzen etc.) und schreitenden (beinartige mechanische Strukturen) Roboter [25].

Die Fortbewegung auf Beinen oder auch Lokomotion genannt, ist seit Millionen von Jahren bei einer Vielzahl von Lebensformen erprobt. Diese bleibt die natürlichste aller Bewegungsformen und ist die einzige akzeptable Fortbewegung, die den Bodenbeschaffenheiten in der Natur gerecht wird. Im Vergleich zur radgetriebenen Roboter sind die Anforderungen an das Bodenrelief und die physikalischen Eigenschaften des Bodens bei schreitenden Maschinen wesentlich geringer [11, 12]. Die Laufmaschinen besitzen besondere Vorteile. Die hohe Mobilität, Beweglichkeit und Manövriertfähigkeit erlauben das Überwinden von Hindernissen bis etwa zur Beinhöhe. Die Fähigkeit zum Treppensteigen und Durchqueren enger Durchbrüche ist vorhanden. Die Adaptionsfähigkeit an Untergründe im zerklüfteten Gelände mit unbekannter und wechselnder Tragfähigkeit ist durch das flexible Setzen der Stützstellen realisierbar. Die Nutzung einzelner Beine als Manipulatoren ermöglicht diese als Träger für zusätzliche Werkzeuge einzusetzen [25].

Diese Vorteile führen jedoch zu erheblich höherer Komplexität des Gesamtsystems. Die mobilen Laufmaschinen verfügen über eine komplexe mechanische Struktur mit mehreren Freiheitsgraden, die ihren biologischen Vorbildern entsprechen. Die Abstimmung der elektronischen und steuerungstechnischen Mittel aufeinander, um das mechanische Verhalten der Laufmaschine manipulieren zu können, erfordert das Verständnis des mechatronischen Systems. Darunter entsteht die Problematik des Steuerungssystems mit Verarbeitung umfangreicher Sensorsignalen, die rückwirkend auf die implementierten Steuerfunktionen ausgewertet werden müssen, die bei koordinierten Bewegungen der Roboterbeine und des Roboterkörpers entstehen.

Die Forschungen auf dem Gebiet der schreitenden Laufmaschinen und der zugehörigen technischen Anwendungen beziehen sich meistens auf die Transportfähigkeit. Die Förderung der technischen Anwendungen für Operationen und Ausführung be-

stimmter Arbeiten sind ebenso wichtig. So können die Laufmaschinen z.B. als eine universelle Arbeitsplattform für Sicherungsarbeiten, Manipulationen und Positionierungen der Instrumente bei Montage- und Reparaturarbeiten eingesetzt werden [25].

1.2 Ziel und Aufbau der Arbeit

Das Institut für Robotik der Hochschule Mannheim (ROB Mannheim) umfasst neben einem hochmodernen Computerlabor und einigen Experimentiersystemen, einen sechsbeinigen Laufroboter vom Typ Lauron IVb [10], der zu Bewegungsstudien dient. Neben diesem wird eine weitere mobile sechsbeinige Laufmaschine entwickelt. Im Rahmen einer Studienarbeit [2] wurde bereits das erste Konzept der Laufmaschine ausgearbeitet.

Das Ziel dieser Arbeit ist es, basierend auf dem bereits erstellten Konzept, gegebenenfalls die Analyse und daraus resultierende Optimierung der Konstruktion vorzunehmen. Des Weiteren erfolgt der mechanische Aufbau sowie die Entwicklung und Implementierung eines Steuerungsalgorithmus basierend auf dem Robot Operating System (ROS) und der Methoden zur Generierung der ballistischen Bewegungsstrajektorien unter Verwendung inverser Kinematik¹[30]. Die analytisch beschriebenen und implementierten Roboterbewegungen sollen getestet werden, sodass sich beim Steuern ein deterministisches Verhalten ergibt und somit dieses vorhersagbar ist [12]. Der Laufroboter soll für Studierende als eine funktionstüchtige Plattform dienen und Experimente in Informatik -Teilgebieten, wie der Künstlichen Intelligenz, Softwaretechnik und den Entwurf sicherer Algorithmen ermöglichen.

Die vorliegende Thesis ist insgesamt in 7 Kapitel untergliedert. In diesem Kapitel wurde bereits die Motivation erläutert und ein Einblick in den Stand der Technik gegeben.

In Kapitel 2 werden die Grundlagen zur Entwicklung von Steuerungsalgorithmen dargelegt. Dieses Kapitel informiert über die Gangarten, die kinematischen Ketten und über die Lösung dieser, sowie über das zur Umsetzung der Steuerung verwendete Betriebssystem.

¹Die Lehre der Bewegung von starren Körpern im Raum, befasst mit der geometrischen Beschreibung von Bewegungsverhältnissen.

In Kapitel 3 wird die Laufmaschine und deren Eigenschaften präsentiert. Es wird auf Konstruktion, Steuerungsrechner, Sensoren und Motoren des Laufroboter eingegangen.

Kapitel 4 gibt einen Überblick über die Entwicklung der Steuerungsalgorithmen. Des Weiteren werden die mathematischen Grundlagen und die Herleitung der Gleichungen zur Kinematik der Laufmaschinen, sowie die Definitionen der Koordinatensysteme erläutert.

Die Implementierung der vorgestellten Lösungen wird in Kapitel 5 beschrieben.

Die Erprobung dieser implementierten Lösungen folgt in Kapitel 6.

Kapitel 7 enthält eine Zusammenfassung der Thesis.

1.3 Stand der Technik

International befassen sich zurzeit viele Forschungsprojekte mit mehrbeinigen Laufmaschinen unterschiedlicher Strukturen und deren technischen Anwendungen. Die Konzepte unterscheiden sich im Wesentlichen durch Design- und Kontrollansätze zwischen technisch orientierten und biologisch inspirierten [12]. Bei letzteren stehen die biologischen Vorbilder im Vordergrund. Die Steuerungsansätze werden beispielsweise mittels neuronaler Netze umgesetzt, die eine reflexive und musterbasierte Steuerung ermöglichen [6, 17]. Die technisch orientierten Steueransätze basieren größtenteils auf analytischen Lösungen, beispielweise auf der Interaktion der generierten Laufmuster mit dem Untergrund [12, 29].

Dabei fokussiert die Erforschung die Untersuchung der grundlegenden Lokomotionsfunktionen [16]. Im Vordergrund ist die Entwicklung neuer Steuermethoden zur Annäherung der Qualität des Gangverhaltens von schreitenden Robotern an biologische Vorbilder.

Die Laufmaschine LAURON² wurde im Forschungszentrum Informatik Karlsruhe (FZI Karlsruhe) nach einem biologischen Vorbild einer Stabheuschrecke (*Carausius morosus*) entwickelt. Die mechanische Struktur, die Anordnung der Gelenke und die Beinkonstruktion sind ähnlich wie bei einem Insekt. Die Steuerung der Maschine ist mittels neuronaler Netze realisiert. Im Vordergrund steht die Untersu-

²LAUfROboter Neuronal gesteuert.

chung des stabilen Laufens auf einem unebenen Gelände. Dabei wird das Lernen der Laufbewegungen mittels einer der Methoden des maschinellen Lernens: Reinforcement Learning³ umgesetzt [10].

Ebenso wird diese Methode bei einer vierbeinigen, säugetierartigen Laufmaschine BISAM⁴ eingesetzt, um eine flexible und adaptive Steuerungsarchitektur, die Verhaltenssteuerung zu entwickeln. Für reflexive Steuerung werden neuronale Netze und Fuzzy-Controller⁵ genutzt [1].

In einem technisch orientierten Projekt von der Universität Magdeburg in Kooperation mit dem Fraunhofer Institut in Magdeburg, wurde die sechsbeinige Laufmaschine Katharina entwickelt. Dieses Projekt basiert auf einer rein analytischen Steuerung der Bewegungen. Nur die Konstruktion der Beine wurde nach biologischen Konzepten entwickelt. Das Ziel war die Untersuchung der adaptiven Schreitalgorithmen und der Sensorsysteme [12].

Ein weiteres technisch orientiertes Projekt ist ein Hybrid-Roboter HyTro-I. Der Roboter stellt eine neue, mechanisch entkoppelte Bein- und Rad-Hybrid transformierbare Plattform dar. Das Ziel ist dabei, die Mobilitätsvorteile rollender Fortbewegungen auf einem flachen Untergrund und schreitender Fortbewegung auf einem unebenen Gelände auszunutzen. Dabei wird speziell auf die Transformation der beiden Modi eingegangen [5].

Neben den mit Elektromotoren angetriebenen Robotern, existieren bereits neue Antriebskonzepte. Bei der Laufmaschine AirBug werden ihre Gelenke mit künstlichen pneumatischen Muskeln bewegt. Mittels antagonistischer Muskelpaare an den Gelenken, kann dieses die Position, das Drehmoment und die Steifigkeit der Beine steuern. Die Imitierung des Verhaltens natürlicher Muskeln, ermöglicht eine hohe Stellkraft sowie Stellgeschwindigkeit, ein Leistungsgewicht⁶ und einen Aufbau eines integrierten Feder-Dämpfungssystems. Die sechsbeinige Laufmaschine AirBug wurde vom FZI Karlsruhe in Kooperation mit dem Pneumatik Spezialist FESTO entwickelt [30].

³Reinforcement Learning (deut. bestärkendes Lernen): Durch die Bestrafung und Belohnung lernt ein Algorithmus eines Systems, wie in potenziell auftretenden Situationen zu handeln ist, um den Nutzen des Systems zu maximieren.

⁴Biologically InSpired wALKing Machine.

⁵Lofti Askar-Zadeh entwickelte 1965 die Fuzzy-Set-Theorie an der University of California, Berkeley. Basierend darauf wird beim einem Fuzzy-Controller ein Expertenwissen mit linguistischen Begriffen eingebunden, mittels dessen der Fuzzy-Controller ohne der Erkenntnisse des mathematischen Models des Prozesses, an einem nichtlinearen Prozess mit mehreren Ein- und Ausgangsparametern modelliert wird.

⁶Das Leistungsgewicht stellt das Verhältnis der Masse und Leistung einer Maschine dar.

Kapitel 2

Grundlagen

Zur Realisierung der Aufgabenstellung werden in diesem Kapitel einige grundsätzliche Erläuterungen folgen. Diese werden nicht im vollen Detail ausgeführt, stattdessen nur jene Bereiche erläutert, die für den weiteren Verlauf der vorliegenden Arbeit als wichtig und relevant erscheinen.

2.1 Planung von Laufmustern

Mehrheitlich werden in der Natur die regulären Gangarten mit periodischen Bewegungsmustern bzw. Laufmustern (*engl.* gait patterns) beobachtet, die sich nach Reihenfolge der Beinbewegungen, Anzahl der Füße mit gleichzeitigem Bodenkontakt und der Dauer des Bodenkontakts eines einzelnen Fußes etc. unterscheiden. Die statisch stabilen Gangarten findet man ausschließlich bei sechs- und mehrbeinigen Tieren. Das statisch stabile Laufen bei der sechsbeinigen Fortbewegung kennzeichnet sich dadurch, dass mindestens drei Füße festen Bodenkontakt haben und die Projektion des Körperschwerpunktes sich entlang der Richtung des Gravitationsvektors auf den Boden innerhalb der durch drei oder mehr Ecken gebildeten konvexen Hülle der Stützstellen befindet. Bei dieser Bedingung sind Gleichgewichtsverlust und Umdrehen unmöglich [26]. Die Literaturen [23] [15] [28] [26] liefern detaillierte Beschreibungen mit notwendigen Simulationen und Resultaten der Experimente. Ebenfalls in [8] werden die robusten und adaptiven Fortbewegungen von autonomen sechsbeinigen Robotern und die folgenden Gangarten erläutert.

Einer der Gangarten, die zu den statisch stabilen Laufmustern gehört, ist der Dreifußgang (*engl.* tripod gait). Beim Dreifußgang wechseln sich beim Anheben, Stüt-

zen und Bewegen jeweils zwei Beingruppen mit je drei Beinen ab. Dabei ist die Anordnung der Füße wie folgt: Das vordere und hintere Bein auf einer Seite und das mittlere Bein auf der gegenüberliegenden Roboterseite. Diese Kombination der Beine erlaubt dem Laufroboter zu jedem Zeitpunkt durch das aufspannen eines großen Stützpolygons ein statisch stabiles Stehen. Die Abbildung 2.1 visualisiert das Bewegungsschema des Dreifußgangs. Die eingeklammerten Zahlen neben den Beinen stellen jeweils eine der Fußphasen (z.B. Stemmpphase, Schwingphase) für einen Bewegungszyklus dar.

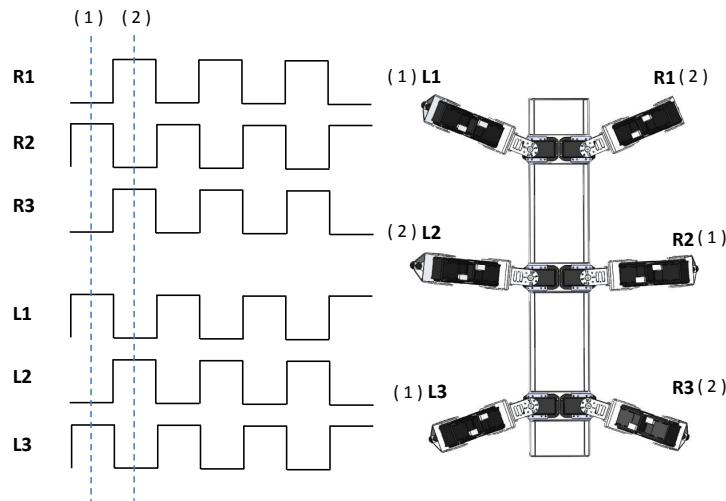


Abbildung 2.1: Dreifußgang

Eine weitere Gangart ist der Wellengang (*engl. wave gait*), dabei werden die Beine einer der Roboterseite, angefangen beim letzten, der Reihe nach abwechselnd bewegt. Dies wird dann mit der Beingruppe auf der gegenüberliegenden Seite wiederholt (Abb. 2.2). Diese Gangart ermöglicht sehr hohe Stabilität des Roboters, da immer nur ein Fuß sich in der Schwingphase befindet, während die restlichen fünf in der Stemmpphase sind. Bei einer Beschleunigung der wellenartigen Bewegung kann es jedoch zur Instabilität der Laufmuster führen, wenn man versucht die Schwingphase der Füße zu verkürzen. Dadurch werden die Schritte kürzer und die Beinbewegungen rasanter. Die Überlappungen der Schwingphase der Füße auf beiden Roboterseiten führen ebenfalls zu einem instabilen Laufmuster. Die Literatur [29] befasst sich ausführlicher mit dieser Gangart.

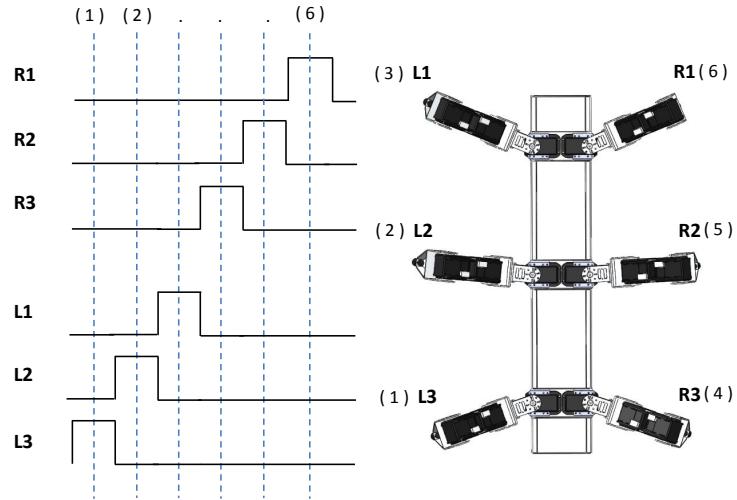


Abbildung 2.2: Wellengang

2.2 Kinematische Ketten

Bekanntlich besteht der Laufroboter aus einer Zusammensetzung von mehreren Gliedern. Um eine Fortbewegung einer Laufmaschine zu realisieren, müssen die einzelnen Glieder in eine bestimmte Position gebracht werden. Dies kann mittels der Kinematik erreicht werden. Übergeordnet befasst sich die Kinematik mit der geometrischen Beschreibung von Bewegungsverhältnissen.

Dieser Abschnitt in der Anlehnung an [7] und [12] beschreibt die Methoden der Kinematik, im genaueren die Zusammenhänge der Koordinatensysteme einzelner Segmente in der kinematischen Kette und die Transformation dieser bis zum letzten Segment, dem sogenannten Endeffektor. Den Endeffektor stellt der Fuß bzw. die Fußspitze eines Beins einer Laufmaschine dar. Die Berechnung der resultierenden Position des Endeffektors im kartesischen Raum bei vorgegebenen Gelenkwinkeln wird mittels *direkter Kinematik* erreicht. Dagegen können die Gelenkwinkel in einer kinematischen Kette durch die vorgegebene Position des Endeffektors und der *inversen Kinematik* bestimmt werden. Die Literatur [27] beschreibt die Methoden der Kinematik ausführlich.

Eine kinematische Kette ist eine Aneinanderreihung von starren Körpern bzw. Segmenten, die jeweils über eine kinematische Komponente, z.B ein Gelenk, miteinander verbunden sind.

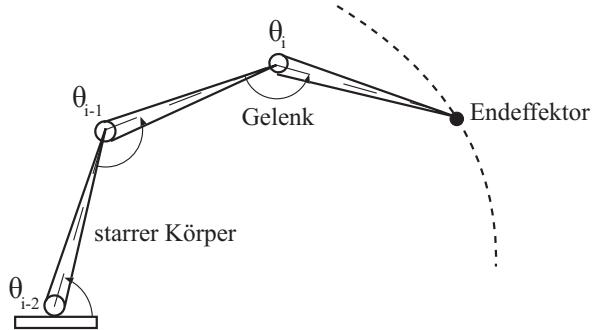


Abbildung 2.3: Kinematische Kette [7]

Jedes Gelenk hat nur einen Freiheitsgrad, der entweder rotatorische oder translatorische Bewegungen ausführt.

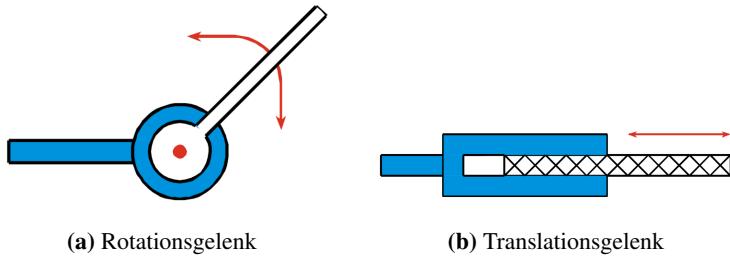


Abbildung 2.4: Zwei Gelenktypen [13]

Die Freiheitsgrade, die eine kinematische Kette bildet, werden durch die Anzahl der Gelenke in dieser Kette bestimmt. So wird jedes einzelne Bein einer Laufmaschine als eine für sich allein bildende kinematische Kette betrachtet.

2.2.1 Denavit-Hartenberg Verfahren

Dies ist ein Verfahren zur Beschreibung kinematischer Ketten, welches von Denavit und Hartenberg entwickelt wurde [4]. Bei diesem Verfahren werden jedem Segment einer kinematischen Kette vier Parameter zugeordnet. Damit kann die Lage benachbarter Segmente zueinander ausreichend beschrieben und eine beliebig große, kinematische Segmentkette beschrieben werden. Dabei sind drei dieser Größen jeweils Geometrieparameter und die vierte Größe ist die Achsenvariable. Diese Parameter werden als DH Parameter bezeichnet.

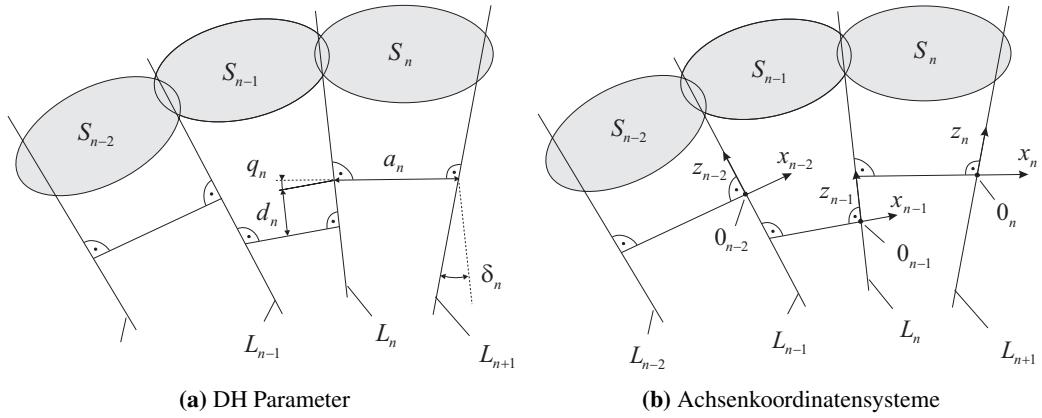


Abbildung 2.5: Das Denavit-Hartenberg Verfahren [12]

DH Parameter

Die in der Abbildung 2.5a dargestellte kinematische Kette besteht aus drei Elementen S_{n-2} bis S_n . Die Parameter für die S_n werden wie folgt bestimmt: Die Achsen L_n und L_{n+1} bilden eine gemeinsame Normale, die Länge dieser Normalen entspricht dem ersten Parameter a_n . Der Versatz zwischen dieser Normalen und der Normalen des Segments S_{n-1} ergibt den zweiten Parameter d_n . Die Verdrehung der Achsen L_n und L_{n+1} zueinander entspricht dem dritten Parameter, dem Drehwinkel δ_n . Der vierte Parameter q_n beschreibt die Drehung um die Rotationsachse L_n . Die Tabelle 2.1 stellt die DH Parameter noch einmal zusammenfassend dar. In der Regel bleiben drei der vier Parameter in einem Segment gleich. Ein Parameter ist variabel und hängt vom Gelenktyp ab. Bei einem Rotationsgelenk ist es der Parameter q_n , beim Translationsgelenk der Parameter a_n .

Tabelle 2.1: Denavit - Hartenberg Parameter

DH Parameter	Beschreibung
a_n	Abstand zwischen den Achsen L_n und L_{n+1}
d_n	Abstand zwischen den Achsen L_n und L_{n+1} entlang z_{n-1}
δ_n	Winkel zwischen den Achsen z_n und z_{n-1}
q_n	Winkel zwischen den Achsen x_n und x_{n-1}

Koordinatensysteme

Neben den DH Parametern werden die körperfesten Koordinatensysteme der Roboterachsen, auch Achsenkoordinatensysteme genannt, in jedem Gelenk festgelegt. Diese werden zur Aufstellung und Berechnung der Transformationsgleichungen benötigt. Die Abbildung 2.5b stellt die Positionierung der körperfesten Koordinaten-

systeme der Roboterachsen dar. Die Festlegung der Lage des Ursprungs des Achsenkoordinatensystems und der Koordinatenachsen ist wie folgt: Die Koordinatenachse x_n ist kollinear zur gemeinsamen Normalen a_n . Die Achse z_n zeigt in Richtung der Achse L_{n+1} . Durch die Vervollständigung des Koordinatensystems um die Achse y_n , entsteht ein rechtsorientiertes, kartesisches Koordinatensystem [27].

2.2.2 Direkte Kinematik

Die direkte Kinematik ermöglicht die Berechnung des resultierenden Endeffektors (Abb. 2.6) bei gegebenen Stellungen der Gelenke.

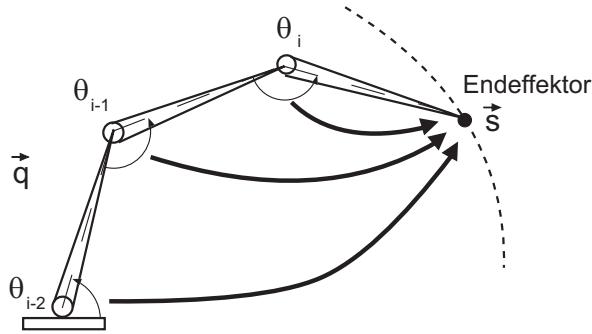


Abbildung 2.6: Direkte Kinematik [7]

Die Vorgehensweise zur Berechnung der direkten Kinematik für den Endeffektor wird formal mit der Formel (2.1) ausgedrückt. Die Funktion f ist nicht linear, d.h es besteht kein linearer Zusammenhang zwischen Position des Endeffektors und der Gelenkvariablen. Der Positionsvektor $\vec{s} = (x, y, z, \alpha, \beta, \gamma)^T$ im kartesischen Raum entspricht dabei der Position und Orientierung des resultierenden Endeffektors, während der Positionsvektor $\vec{q} = (q_1, \dots, q_n)^T$ in Gelenkkoordinaten, mit $n =$ Anzahl der Gelenke, die Stellungen der einzelnen Gelenke darstellt.

$$\vec{s} = f(\vec{q}) \quad (2.1)$$

Mittels der vier Elementartransformationen lässt sich der Übergang von einem Achsenkoordinatensystem (Abb. 2.5b) zu einem in der kinematischen Kette folgenden Achsenkoordinatensysteme ausdrücken. Zum Aufstellen der Transformationen werden die im vorhergehenden Kapitel vorgestellten DH Parameter (Tabelle 2.1) eingesetzt.

$${}_{n-1}^n T = \text{Rot}(z_{n-1}, q_n) \cdot \text{Trans}(0, 0, d_n) \cdot \text{Trans}(a_n, 0, 0) \cdot \text{Rot}(x_n, \delta_n) \quad (2.2)$$

Die Gleichung (2.2) besteht aus folgenden vier Einzeltransformationen:

1. Rotation um die Achse z_{n-1} mit dem Winkel q_n .
2. Translation entlang der Achse z_{n-1} um den Weg d_n .
3. Translation entlang der Achse x_n um den Weg a_n .
4. Rotation um die Achse x_n mit dem Winkel δ_n .

Beim Aufstellen der Transformationsgleichung ist das festgelegte Achsenkoordinatensystem zu beachten, da dieses definiert um welche Achse rotiert oder entlang welcher verschoben werden muss. Hierbei muss je eine Achse auf der Normalen und eine auf der Gelenkachse liegen, damit die Transformationsgleichung angewendet werden kann. Die Gelenkachse wird für die ersten beiden Transformationen verwendet, während die anderen zwei Einzeltransformationen für die Achse, die auf der Normalen liegen angewendet werden.

Mittels der entsprechenden homogenen Transformationsmatrix und der Gleichung (2.2) kann durch das Ausmultiplizieren der Einzeltransformationen die Transformationsmatrix (2.3) aufgestellt werden.

$$\begin{aligned} {}_{n-1}^n T &= \begin{pmatrix} \cos q_n & -\sin q_n & 0 & 0 \\ \sin q_n & \cos q_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \delta_n & -\sin \delta_n & 0 \\ 0 & \sin \delta_n & \cos \delta_n & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ {}_{n-1}^n T &= \begin{pmatrix} \cos q_n & -\sin q_n \cos \delta_n & \sin q_n \sin \delta_n & a_n \cos q_n \\ \sin q_n & \cos q_n \cos \delta_n & -\cos q_n \sin \delta_n & a_n \sin q_n \\ 0 & \sin \delta_n & \cos \delta_n & d_n \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \quad (2.3)$$

Die Gleichung (2.3) beschreibt den Übergang vom Achsenkoordinatensystem des Gelenks $n - 1$ in das Achsenkoordinatensystem des Gelenks n in Abhängigkeit der DH Parameter. Durch Verwendung der homogenen Koordinaten können die

Transformationen für die gesamte kinematische Kette bis zum Endeffektor durch die Multiplikation der Transformationsmatrizen miteinander verknüpft werden.

$${}^0_m T = {}^0_1 T {}^1_2 T \dots {}^{m-1}_m T \quad (2.4)$$

Die resultierende homogene Transformationsmatrix Gleichung (2.3) für das gesamte Bein enthält Teilmatrizen für die Rotation und Translation. Damit ist es möglich die Orientierung und die Position des Endeffektors relativ zum ersten Achsenkoordinatensystem der Kette zu extrahieren.

$${}^0_m T = \begin{pmatrix} {}^0_m R & {}^0 \vec{r} \\ 0 & 1 \end{pmatrix} \quad (2.5)$$

Die 3×3 Rotationsmatrix ${}^n_m R$ beschreibt die Orientierung der beiden Achsenkoordinatensysteme zu einander. Die Position des Koordinatenursprungs und somit die Position des Endeffektors relativ zum ersten Achsenkoordinatensystem in der Kette wird mit dem Ortsvektor ${}^0 \vec{r}$ angegeben.

2.2.3 Inverse Kinematik

Mittels inverser Kinematik (Abb. 2.7) werden die möglichen Stellungen der Gelenke bei gegebener Position des Endeffektors im kartesischen Raum bestimmt.

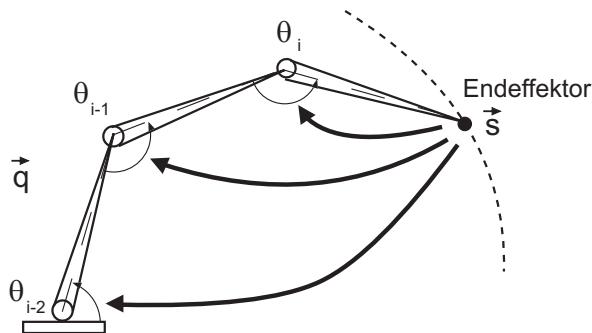


Abbildung 2.7: Inverse Kinematik [7]

Zur Lösung des Problems der inversen Kinematik wird die inverse Funktion (2.6) der Gleichung (2.1) gesucht, die die Berechnung der direkten Kinematik formal beschreibt.

$$\vec{q} = f^{-1}(\vec{s}) \quad (2.6)$$

So muss die Gleichung (2.1) nach q_i aufgelöst werden. Hierfür gibt es jedoch keine direkte, einfach zu erzeugende inverse Funktion, da die zu lösenden Gleichungen nichtlinear sind. Daher werden für die Lösung zwei verschiedene Vorgehensweisen eingesetzt. Bei einer der beiden wird versucht eine analytische Lösung zu finden, während bei der anderen die numerischen Methoden zu Grunde liegen [12]. Im folgenden werden die Gründe, die die Lösungsfindung der inversen Kinematik wesentlich komplizierter gestalten als die Lösung der direkten Kinematik erläutert [12]. Die inverse Kinematik besitzt in den meisten Fällen mehrere Lösungen (Abb. 2.8), dies wird durch die Variation der unterschiedlichen Gelenkstellungen beim Anfahren der Endposition des Endeffektors verdeutlicht. Einige der Lösungen sind durch die mechanischen Randbedingungen nicht zulässig. Genauso gibt es eine große Menge der Positionen für die es überhaupt keine Lösungen der inversen Kinematik gibt. Diese Positionen befinden sich außerhalb des definierten Arbeitsraumes (Kapitel 3.1.2) für eine kinematische Kette.

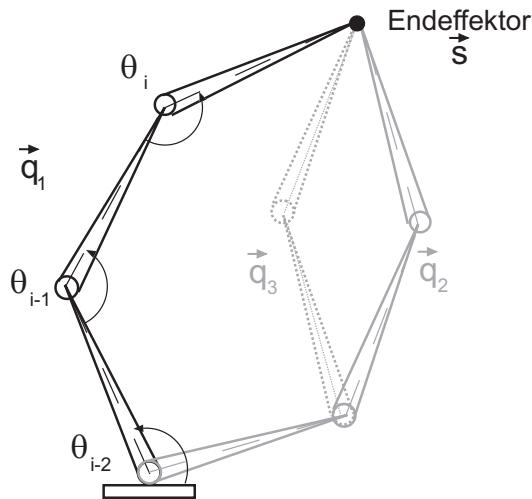


Abbildung 2.8: Lösungen der inversen Kinematik [7]

Analytische Lösung

Bei einer analytischen Lösung der inversen Kinematik gibt es keine allgemeingültige Vorgehensweise. Die Formeln zur Beschreibung der Lösung sind auf Basis geometrischer Eigenschaften der kinematischen Kette herzuleiten. Die Komplexität des Herleitens ist proportional zu der Anzahl von Freiheitsgraden einer Kette. Der Vorteil der analytischen Lösung liegt in der Möglichkeit alle möglichen Gelenkstellungen bei gegebener Position zu berechnen und darin, dass die Berechnung der inversen Kinematik effizienter ist.

2.3 Robot Operating System

Das Ziel verschiedenster Forschungsarbeiten auf dem Gebiet der Robotik sind die vollkommen autonome Roboter, die selbstständig die Entscheidungen treffen und komplexe Aufgaben lösen können. Dazu zählen beispielsweise die Fähigkeiten der autonomen Navigation und Fortbewegung, die Objekterkennung und gezieltes Greifen. Hierfür sind die effizienten Ansätze, die es ermöglichen mit geringem Einstiegsaufwand und großer Flexibilität und Wiederverwendbarkeit, die Roboter-Algorithmen und Treiber zu entwickeln und zu nutzen. Für diese Anforderungen wurde das Open Source: ROS entwickelt. ROS ist ein Framework, das beim Entwickeln, bei der Umsetzung der Algorithmen für mobile Roboter und Verwendung bereits implementierter Algorithmen eingesetzt wird. Die folgenden Erläuterungen sind an die Literatur [14] angelehnt, die sich mit den Funktionalitäten des ROS Systems ausführlicher befasst. Ebenso können die ausführlichen Beschreibungen zur Nutzung der ROS Stacks und einige Implementierungsbeispiele in den Literaturen [20] [9] gefunden werden.

ROS wurde zunächst unter dem Namen „Switchyard“ im Rahmen des STAIR-Projektes¹ am Standford Artificial Intelligence Labarotory im Jahr 2007 entwickelt. Erst 2008 wurde vom Robotikinstitut Willow Garage die Hauptentwicklung unter dem Namen ROS weitergeführt. Seitdem ist dieses bereits in mehreren Distributionen², wie beispielsweise Box turtle, C turtle, Electric, Fuerte, Groovy, Hydro, Indigo Igloo erschienen, die jeweils einen in sich stabilen und gesicherten Entwicklungsstand repräsentieren. Weltweit werden von vielen Instituten und Firmen hunderte weitere Pakete für ROS entwickelt, die in diversen Repositoryn gepflegt werden. Umfangreiches Material, teilweise mit detaillierten Einblicken in das Basis-System und eine Vielzahl an Tutorials für die ersten Erfahrungen im praktischen Umgang mit ROS und seinen Werkzeugen bietet, die ausführliche ROS - Wiki. Die Abbildung³ 2.9 zeigt Roboter, für die bereits die Steuerung mittels ROS realisiert wurde.

¹Standford AI Robot, ein Projekt zur Entwicklung eines Service-Roboters[22].

²<http://wiki.ros.org/Distributions>, Stand: Juli 2014

³<http://www.ros.org/>, Stand: Juli 2014

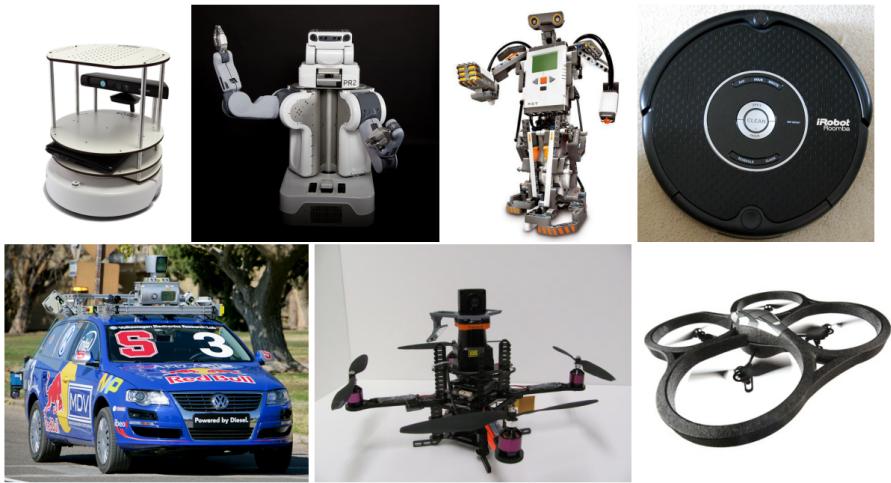


Abbildung 2.9: Oben (v.l.n.r): Turtlebot, PR2, NXT, Roomba
Unten (v.l.n.r.): Stanford Junior, CityFlyer, Parrot AR.Drone [14]

2.3.1 Grundkonzept

Das Framework, das mehrsprachig angelegt ist, ermöglicht auf ROS basierende Systeme möglichst modular aufzubauen, indem überschaubare und abgegrenzte Funktionseinheiten eingesetzt werden. Diese gewährleisten ein flexibles Entwickeln, Programmtests und die Reduzierung der Ausfallwahrscheinlichkeit des Gesamtsystems. So können bereits implementierte Algorithmen, Treiber und Programme anderer ROS-Programmierer beispielweise als ROS-freie Bibliotheken und Programme eingebunden werden. Das ROS-basierte Roboter-Steuerungssystem, zeichnet sich im Wesentlichen durch eine Mehrzahl unabhängiger und ggf. verteilter Prozesse, sogenannter Nodes aus. Diese übernehmen die Teilaufgaben, wie z.B. die Steuerung eines Servomotors oder eines Geifers, bzw. das Auslesen und Auswerten der Sensordaten. Die Interaktion der Nodes wird über den Austausch der Messages realisiert. Dies passiert entweder asynchron über die Topics oder synchron über die Services. Hierfür müssen sowohl die Nodes als auch die Messages, Topics und Services eindeutig identifizierbar sein. Da die Names bzw. Namen ein grundlegendes Element zur Strukturierung und Repräsentation der Komponenten eines ROS-Systems bilden.

Nodes

Die Nodes bilden im Prinzip bei jeder ROS-Anwendung durch das indirekte Interagieren ein Gesamtsystem. Der Informations- bzw. Nachrichtenaustausch geschieht über die festen Kanalnamen. Dabei können zur Laufzeit über eine globale Parameter-Schnittstelle Konfigurationsdaten an diese gesendet werden. Ein Node ist im Wesentlichen ein komplett eigenständiger Prozess, der zur Laufzeit des Systems gestartet und beliebig gestoppt werden kann. Dies führt durch strikte Kapselung und Aufspaltung der Gesamtaufgabe in relativ kleine Funktionseinheiten zur Übersichtlichkeit, Fehlerrobustheit und einer effizienten Form des Debuggings.

2.3.2 Nachrichtenaustausch

Für den Nachrichtenaustausch zwischen Nodes bietet ROS zwei Grundkonzepte, Topics und Services. Diese beiden Konzepte bilden die Basis für die Kommunikation in einem ROS-basierten System.

Topic werden für asynchrone Kommunikation zwischen den Nodes genutzt. Die Nodes tauschen dabei anonym Daten über einen, durch Namen und einen Nachrichtentyp (Message) eindeutig definierten, gemeinsam verwendeten Bus aus. Dabei kann ein Node die Informationsdaten in Form einer Nachricht (Message) zu einem Topic veröffentlichen (publish) oder die Nachrichten eines Topics mit definiertem Nachrichtentyp abonnieren (subscribe) (Abb. 2.10).

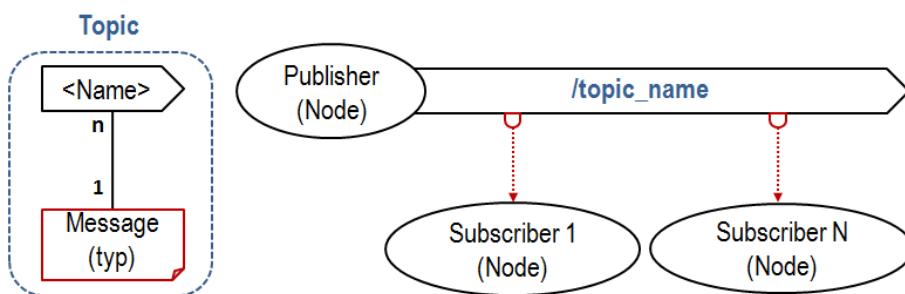


Abbildung 2.10: ROS Topic [14]

Service besteht grundsätzlich aus einem eindeutigen Namen, einer definierten Request-Nachricht und einer definierten Response-Nachricht. Zum Nutzen des Services sendet ein Node eine Request-Nachricht, zusätzlich den eindeutigen Namen des Services und wartet auf die Response-Nachricht. Wenn ein Node einen Service unter diesem Namen zur Verfügung stellt, wird die empfangene Request-

Nachricht verarbeitet, beispielweise werden dem Service zugewiesenen Berechnungen oder Auswertungen durchgeführt und die Response-Nachricht zurückgesendet (Abb. 2.11).

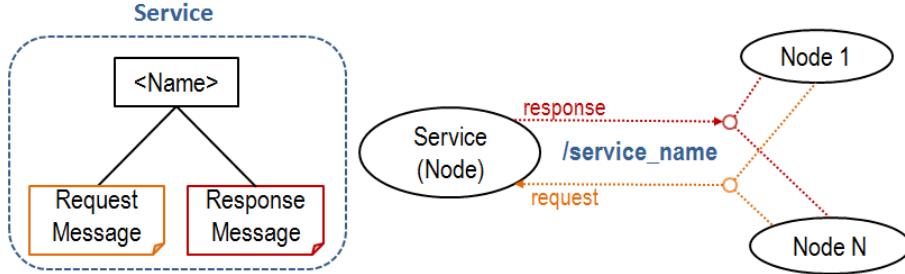


Abbildung 2.11: ROS Service [14]

Ein Service kann von mehreren Nodes genutzt werden, jedoch sind bei diesem keine konkurrierenden Service-Provider erlaubt. Hierfür gibt es die Möglichkeit persistente Verbindungen⁴ aufzubauen. Dies führt zur Performance-Vorteil für sich häufig wiederholende Service-Aufrufe und vermindert aber die Flexibilität des Systems, weil beim Austauschen oder Entfernen der beteiligten Nodes Komplikationen auftreten können.

Messages werden über die Topics und Services verschickt und diesen fest zugeordnet. Ein Nachrichtentyp stellt eine fest definierte Datenstruktur dar. Diese können aus verschiedenen Datentypen, wie beispielweise Integer, Fließkommazahlen, Strings, Arrays aller Typen oder aus anderen Nachrichten aufgebaut werden. Für weitere Basisinformationen können Header mit Sequenznummern und Zeitstempeln eingesetzt werden. Die Definition einer Message erfolgt mit Hilfe der Interface Definition Language (IDL) in einer .msg-Datei. Ein Beispiel für die Nachrichtenbeschreibung ist in der Abbildung 2.12 dargestellt.

⁴Bei einer persistenten Verbindung wird nicht für jeden Service-Aufruf eine neue Verbindung aufgebaut, sondern diese wird einmalig aufgebaut und danach offen gehalten. Es führt insbesondere bei häufig genutzten Verbindungen deutliche Performance-Vorteile, da der Kommunikationsoverhead reduziert wird.

```

Header header
  uint32 seq
  time stamp
  string frame_id
  uint32 height
  uint32 width
  string encoding
  uint8 is_bigendian
  uint32 step
  uint8[] data

```

Abbildung 2.12: Message-Definition: Sensor/Image [14]

2.3.3 Kommunikationsaufbau

Das P2P-Netzwerk verbindet die einzelnen Nodes und führt sämtliche Kommunikationen durch. Das verbindungsorientierte TCP/IP-Protokoll wickelt primär den Austausch der Nachrichten ab. Für eine P2P-Architektur wird zusätzlicher Namensservice eingesetzt, mittels dessen die Peers sich finden und der Aufbau der benötigten Verbindungen gesteuert werden kann (Abb. 2.13).

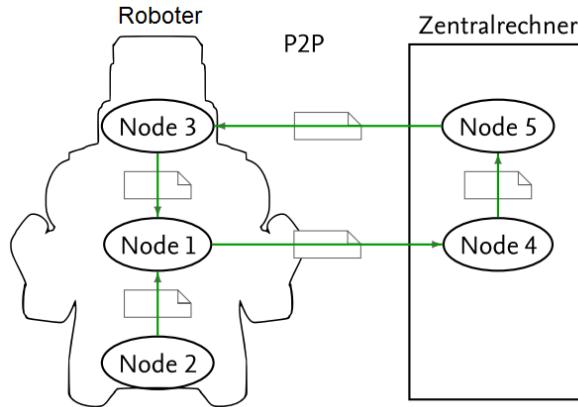


Abbildung 2.13: P2P-Topologie [14]

Die Funktionalität des Namensservices übernimmt im ROS der **Master**. Als Initiator koordiniert er den Nachrichtenaustausch zwischen den einzelnen Peers. Die Topic- oder Service-Messages werden über diesen nicht verschickt. Das Prinzip des Verbindungsaufbaus über den Master anhand eines Beispiels in Abbildung 2.14 dargestellt. Der Master wird über Veröffentlichung der Informationen des Camera-Nodes unter dem Topic /image informiert (a). Zunächst kommt keine Verbindung zustande, da es zu dem Topic /image noch keine Abonnenten gibt. Wenn ein Node, beispielsweise viewer, den Topic /image abonnieren möchte, teilt dieser es

dem Master mit (b). Erst wenn der Master erkennt, dass sowohl ein Veröffentlichter (camera) als auch einen Abonnenten (viewer) existieren, informiert dieser den Abonnenten-Node über den neuen Veröffentlichen. Zur Initiierung einer Verbindung, kontaktiert der Abonnenten-Node den Veröffentlichen, während dieser die Einstellungen für das ausgewählte Transferprotokoll übermittelt, um eine P2P-Verbindung aufzubauen. So können die Nodes unabhängig vom Master die Informationen unter sich austauschen. Der Kommunikationsaufbau ist mittels XML-RPC⁵ umgesetzt.

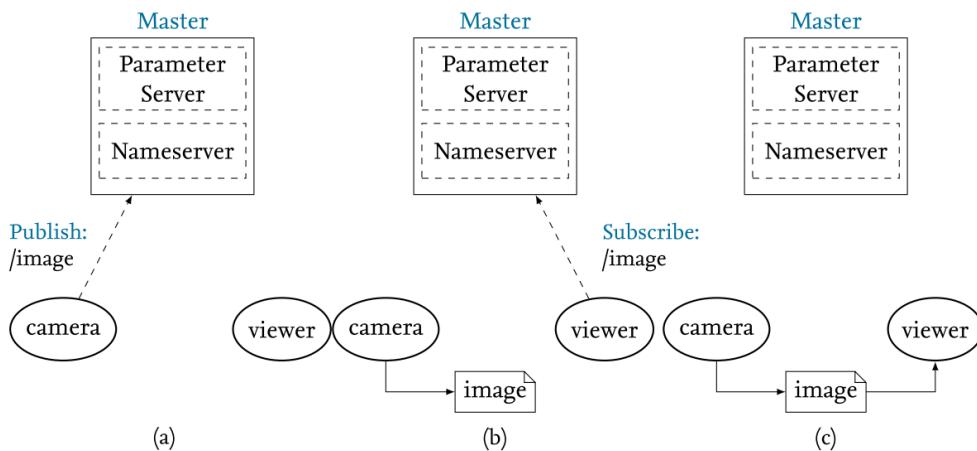


Abbildung 2.14: Topic-Verbindungsauflaufbau über Namensserver [14]

2.3.4 Organisation

Der Aufbau und die Organisation von ROS auf Datei-Ebene und das damit verbundene Konzept zur Wiederverwendbarkeit von Quellcode.

Package (Paket) ist eine Sammlung von Inhalten, die zusammen eine bestimmte Funktionalität zur Verfügung stellen und die Strukturierung von Programmen, Code, Bibliotheken und sonstigen Elementen auf Datei-Ebene ermöglichen. Ein ROS-Paket stellt im Prinzip eine Verzeichniss hierarchie dar und kann beispielsweise, sowohl mehrere Nodes, jeweils mit spezieller Funktionalität, als auch spezielle Bibliotheken und Definitionen für Nachrichtentypen enthalten. Jedes Paket hat eine XML-Datei⁶ `manifest.xml`, die die wichtigsten Informationen über das Paket liefert, wie z.B. die Auflistung der Abhängigkeit bzw. der Pakete die für die Funktionalität

⁵Extensible Markup Language Remote Procedure Call

⁶Extensible Markup Language (Auszeichnungssprache) wird zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien eingesetzt.

dieses Pakets benötigt werden. Das wesentliche Ziel des Strukturierungselements (Paket) ist die Vereinfachung der Wiederverwendbarkeit.

Stack stellt eine Sammlungen von Paketen dar und ist somit ein weiteres strukturelles Element von ROS, welches eine Hierarchieebene höher liegt. Alle Paket-Verzeichnisse unterhalb eines Stack-Verzeichnisses, gehören automatisch zu diesem Stack (Abb. 2.15). Stack-Verzeichnisse werden zum einfachen Verteilen von Code genutzt und enthalten neben einer Versionsnummer die XML-Beschreibungsdatei, die sogenannte Stack Manifest (stack.xml).

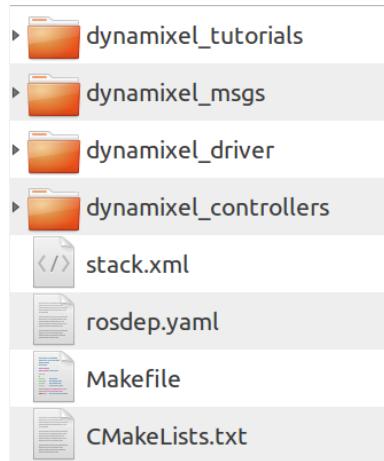


Abbildung 2.15: Die Verzeichnisstruktur des Dynamixel_motor-Stacks

Manifest liefert Informationen zum jeweiligen Paket oder Stack. Dieses enthält beispielsweise den Namen des jeweiligen Autors, den Lizenz-Typ, Verweise zur Projektseite, die Auflistung der Anhängigkeiten von anderen Paketen oder Stacks und Build-Anweisungen. Die Abhängigkeiten werden zum Erleichtern der Installation und zur Verwendung eines Stacks genutzt. Die Build-Anweisungen erlauben, dass ein Stack als Ganzes kompiliert werden kann.

Kapitel 3

Grundlegender Aufbau des Laufroboters

Dieses Kapitel beschreibt die sechsbeinige Laufmaschine (Abb. 3.1), für die ein Steuerungsalgorithmus entwickelt wurde. Der Laufroboter soll für Studierende als eine Plattform dienen und die Möglichkeit für die Experimente in den Informatik-Teilgebieten: Künstliche Intelligenz, Softwaretechnik und den Entwurf sicherer Algorithmen bieten. Ferner sollen Experimente auf den obengenannten Teilgebieten der Informatik beispielsweise, die Fortbewegung auf einem unbekannten Untergrund ermöglicht werden. Hierfür eignet sich die sichere Fortbewegung und die Nutzung der statisch stabilen Laufmustern wie z.B. eines Dreifußgangs, bei dem mindestens drei Beine zu jedem Zeitpunkt auf dem Boden sind und ein großes Stützpolygon bilden. Dies ist eines der Argumente, um einen Laufroboter mit sechs Beinen zu bauen.



Abbildung 3.1: Die Laufmaschine

Die Studienarbeit[2] befasste sich bereits mit der Suche der Komponenten für die Laufmaschine. Im Rahmen dieser wurden die Servomotoren Dynamixel RX64, die Halterungen für die Zusammensetzung einzelner Beine, der Roboterkorpus bzw. das Gehäuses und die Sensoren ausgesucht. Für die Visualisierung des Zusammenspiels der Mechanik bzw. ausgesuchter Komponenten wurde im Umfang der Arbeit das erste Konstruktionskonzept mit dem computer-aided design (CAD) Programm SolidWorks ausgearbeitet, das bereits die nötigen Roboter-Einzelteile und die dazugehörenden detaillierten CAD Zeichnungen enthielt.

Die Anforderungen an die Laufmaschine sind, eine möglichst sichere sechsbeinige Fortbewegung, durch definierte Lauf-und Bewegungsmuster und ein autonomer Betrieb des Roboters. Die Geometrie der Beine und des Grundkörpers wurden aufgrund der länglichen Form einer Stabheuschrecke nachempfunden.

3.1 Mechanischer Aufbau

Entsprechend der aufgestellten Anforderungen an die zu entwickelnde Laufmaschine (Abb. 3.1) ist der mechanische Aufbau umgesetzt. An dem Körper sind sechs Beine mit jeweils drei Freiheitsgraden angebracht. Ferner wird die Laufmaschine um eine Kamera ergänzt, die über zwei Freiheitsgrade verfügt. In der nachfolgenden Tabelle 3.1 ist der Gesamtüberblick über die mechanischen Parameter der entwickelten Laufmaschine abgebildet.

Tabelle 3.1: Mechanische Parameter des Laufroboters

Grundlegende Parameter	
Anzahl der Beine	6
Aktive Freiheitsgrade (gesamt)	18
Gewicht	8 kg
Körper	
Grundkörperabmessungen	Punkt-zu-Punkt
Höhe ca.	56 mm
Seitenlänge ca.	62mm
Breite ca.	102 mm
Beine	
Aktive Freiheitsgrade	3
Arbeitsbereich der Gelenke	
α -Gelenk	von -50° bis 50°
β -Gelenke	von -106° bis 106°
γ -Gelenk	von -135° bis 135°
Antriebe	Gleichstrommotoren
Abmessungen eines Beines	
Gesamt	332 mm
A-Segment	72 mm
B-Segment	97 mm
C-Segment	163 mm
Masse eines Beines ca.	0,8 kg

3.1.1 Körper des Laufroboters

Die Grundform des Laufroboters ähnelt dem Körper einer Stabheuschrecke. Dabei bildet der Grundkörper eine viereckige prismatische Form, die aus vier einzelnen Aluminiumgehäusen besteht (Abb. 3.2). Im vorderen Gehäuse werden der Ultraschallsensor und ferner die erwähnte Kamera montiert. Das erste mittlere Gehäuse verfügt über einen ausreichenden Freiraum für Batterien. Im hinteren Gehäuse befindet sich ein Steuerrechner und übrige Elektronik der Laufmaschine. Die benötigten Schnittstellen, wie z.B. USB und weitere Anschlussmöglichkeiten des Steuerrechners sind ins hintere Gehäuse hinaus geführt. Zwischen den vier beschriebenen Aluminiumgehäusen sind jeweils ein Beinpaar und insgesamt sechs Beine befestigt.

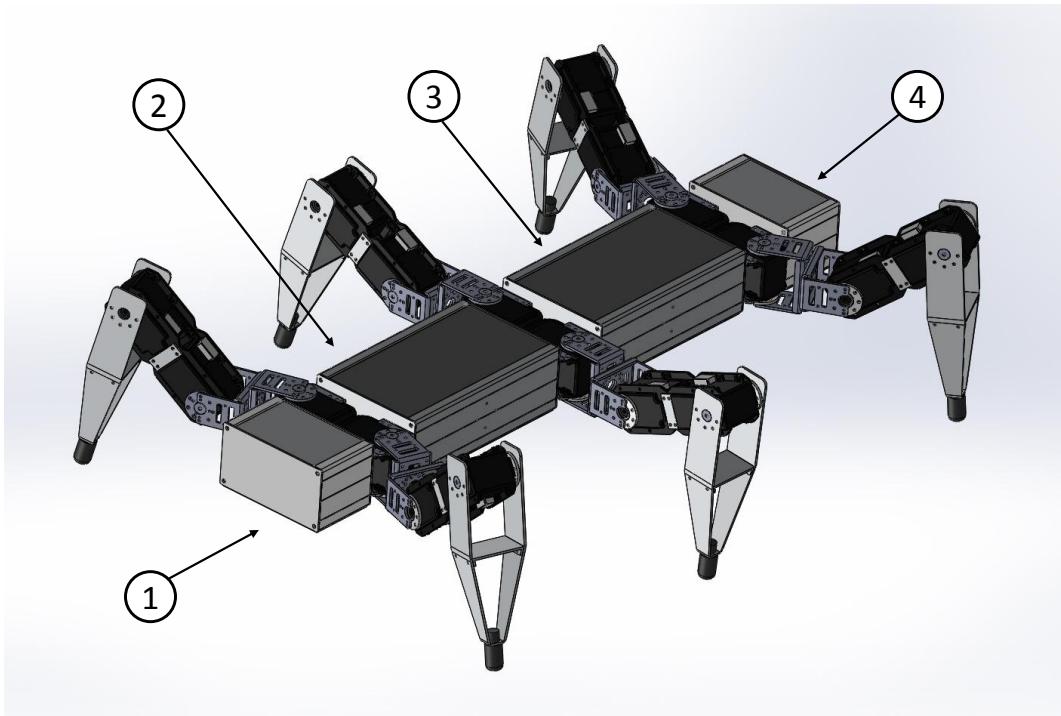


Abbildung 3.2: Ein CAD-Modell des Roboters mit vier Gehäusen

3.1.2 Beine des Laufroboters

Zur Erzeugung der räumlichen Trajektorie besitzt jedes Bein drei Freiheitsgrade. Somit wird die beliebige Positionierung des Fußes im Arbeitsraum des Beins möglich. Die hier definierten Freiheitsgrade werden als Rotationsfreiheitsgrade bezeichnet [27]. Die Verkettung dieser drei rotatorischen Freiheitsgrade erfolgt in jedem Bein hintereinander. Dabei ist das erste Gelenk senkrecht zur Grundfläche des Roboterkörpers, das erste und zweite stehen senkrecht und das zweite und dritte Gelenk parallel zueinander (Abb. 3.3).

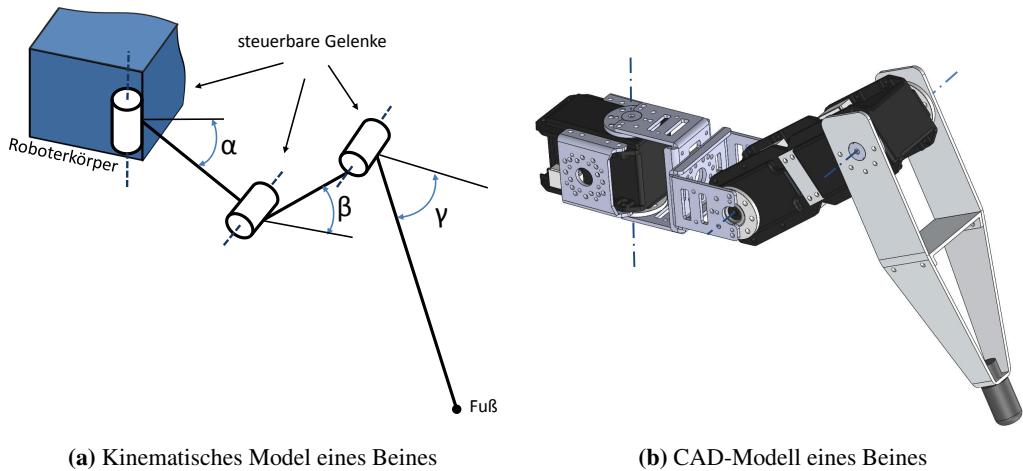


Abbildung 3.3: Kinematisches Model eines Beines

Durch das Anbringen der Gummifüße am unteren Ende des Fußes wird eine punktförmige Auflagefläche realisiert. Nachteilig dabei ist, dass die räumliche Ausrichtung der Fußspitze nicht bestimmbar ist. Die Veränderung der räumlichen Orientierung des Fußes in der Stempphase, zum einen bei den entstehenden Querkräften während der Beinbewegung am weichen Untergrund, zum anderen wegen der geringen Reibung der punktförmigen Auflagefläche auf einem festen Untergrund können nicht erfasst werden.

Der Arbeitsbereich eines Beines ist in der Abbildung 3.4 dargestellt. Die Abbildung 3.4a zeigt den Arbeitsbereich eines Beines in der Ebene der Alphagelenkachse. Der Arbeitsbereich wird durch einen Kreisbogen begrenzt, der durch die Rotationsgelenke und minimale und maximale möglichen Gelenkwinkeln entsteht. Die Abbildung 3.4b zeigt eine Projektion des Arbeitsbereichs auf eine Ebene senkrecht zur Alphagelenkachse.

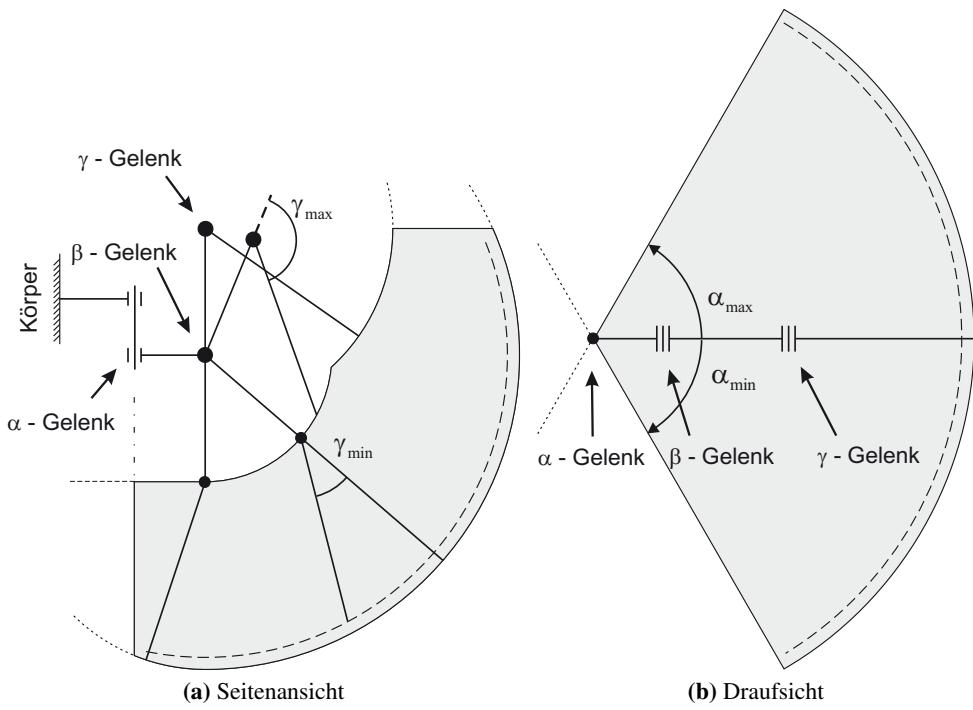


Abbildung 3.4: Arbeitsbereich eines Beines [12]

Zur Vermeidung der Beschädigungen an der Konstruktion, sind bei der Ansteuerung der Motoren, die mechanischen Beschränkungen nicht zu überschreiten. Die Tabelle 3.1 zeigt die maximalen Ausschläge der Gelenke bzw. den Arbeitsbereich für die einzelnen Beine.

3.2 Steuerrechner

Zur Steuerung der Laufmaschine wird ein überarbeitetes Einplatinencomputer Raspberry Pi Modell B+ (Abb. 3.5)¹ auf Basis eines 700 MHz ARM 11 Prozessors genutzt. Raspberry Pi bietet einen Arbeitsspeicher von 512 MB, eine große Anzahl an unterschiedlichen Schnittstellen: Wireless Local Area Network (WLAN), vier Universal Serial Bus (USB)-Anschlüsse, eine Ethernet-Schnittstelle und eine serienmäßige High Definition Multimedia Interface (HDMI)-Schnittstelle zur Visualisierung. Über WLAN kann eine drahtlose Steuerung der Laufmaschine per Gamepad F710 vom Hersteller Logitech erfolgen. Mittels einer SD-Speicherkarte als Bootmedium kann beispielsweise ein ARM-Architektur unterstützendes Linux-

¹<http://www.raspberrypi.org/>

Betriebssystem installiert und genutzt werden. Raspberry Pi bietet eine Installations- und Nutzungsmöglichkeit des ROS. Die Leistungsaufnahme des Mini-Rechners liegt zwischen 2,5 und 3 Watt.

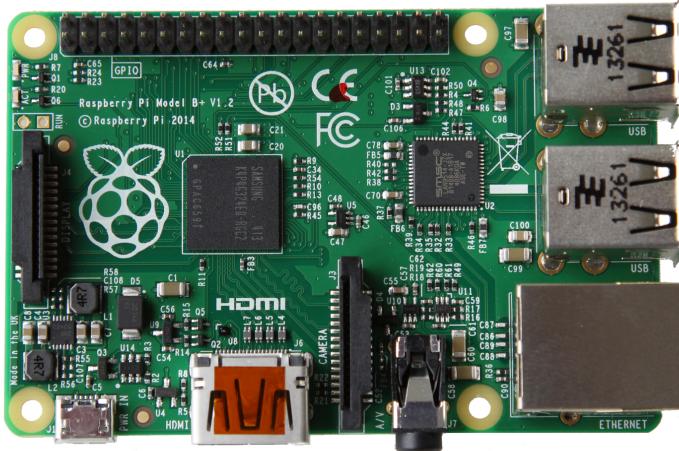


Abbildung 3.5: Das neue Raspberry Pi Modell B+

Dies sind die Gründe, warum Raspberry Pi bei der verwendeten Laufmaschine als Steuerrechner genutzt wird. Neben einem günstigen Beschaffungspreis war die kleine Bauform in einer Kreditkartengröße des Raspberry Pi von Vorteil, da diese platzsparend in den Korpus der Laufmaschine integriert werden kann. Das geringe Gewicht, die Anbindungsmöglichkeit des Raspberry Pi Kamera Moduls, dass die komplexen Bildverarbeitungsverfahren erlaubt und die ausreichende Anzahl an Schnittstellen für die Ansteuerung der Motoren, der Anschlussmöglichkeiten der Sensoren über die 40 zu Verfügung stehenden General-purpose input/output (GPIO)-Pins Inter-Integrated Circuit (I2C) und Universal Asynchronous Receiver Transmitter (UART) Bussystemen zählen zu Vorteilen für diesen Entschluss.

3.3 Sensoren

Im Rahmen der Studienarbeit[2] wurden vorerst die Force Sensing Resistor (FSR) Kraftsensoren für jedes einzelne Bein und ein SRF05 Ultraschallsensor eingeplant. Die Kraftsensoren sollen neben der Erfassung des Bodenkontakts, die Messung der einwirkenden Kräfte auf die einzelnen Füße der Laufmaschine ermöglichen. Diese wurden in Form eines Foliensensors ausgewählt. Die Foliensensoren haben einen kräfteabhängigen Widerstand, sind in kleiner Baugröße mit einem analogen Aus-

gangssignal erhältlich und sind mittels einer selbstklebenden Oberfläche einfach zu befestigen. Der Ultraschallsensor SRF05 hat eine kleine Bauform und eine Reichweite von 4m, dies ermöglicht eine frühzeitige Erkennung von Hindernissen. Das Ausgangssignal des Sensors ist zeitbasiert und kann mit dem Raspberry Pi ausgewertet werden. Alternativ kann das Raspberry Pi Kamera Module mit einem 5 Megapixel-Sensor zur Bildverarbeitung und zur Hinderniserfassung eingesetzt werden. Außerdem sind weite Sensoren zum Einbau vorgesehen: Positions- und Neigungssensor, Beschleunigungssensor.

3.4 Motoren

Für die Laufmaschine wurden die Dynamixel Motoren der RX-Serie[19] ausgewählt. Die Antriebe weisen für das Projekt die passenden Eigenschaften auf. Neben dem ausreichenden Drehmoment, um den Laufroboter anzutreiben und der Positionsregelung, kann die Vielzahl an RX-64 Motoren der Firma Robotis über einen RS485 Bus angesteuert werden. Hierfür wird nur ein Controller benötigt, z.B. Raspberry Pi, der über eine USB-Schnittstelle und einen entsprechenden USB-RS485-Konverter (Abb. 3.7) Protokolle an die integrierte Regelektronik der RX-64 Motoren senden kann. Die Verwendung eines Bussystems reduziert den Verkabelungsaufwand und vereinfacht die Verkabelung des Laufroboters(Abb. 3.6). Die RX-64 Motoren können in zwei Modi angetrieben werden: Wheel-Modus (Rad-Modus) bei dem eine endlose Drehbewegung von 360° möglich ist und der Joint-Modus (Gelenk-Modus), welcher ausgehend von der Mittelstellung jeweils um 150° in beide Richtungen drehbar ist und somit die Gelenke über ein Drehbereich von 300° verfügt (Abb. 3.6). Unter anderem können weitere Informationen zu aktuellen Winkelpositionen, eingestelltem Drehmoment, Geschwindigkeit, Betriebstemperatur abgefragt und überwacht werden. Die Abbildung 3.6 veranschaulicht schematisch die Verbindung der Motoren über einen RS485 Bus.

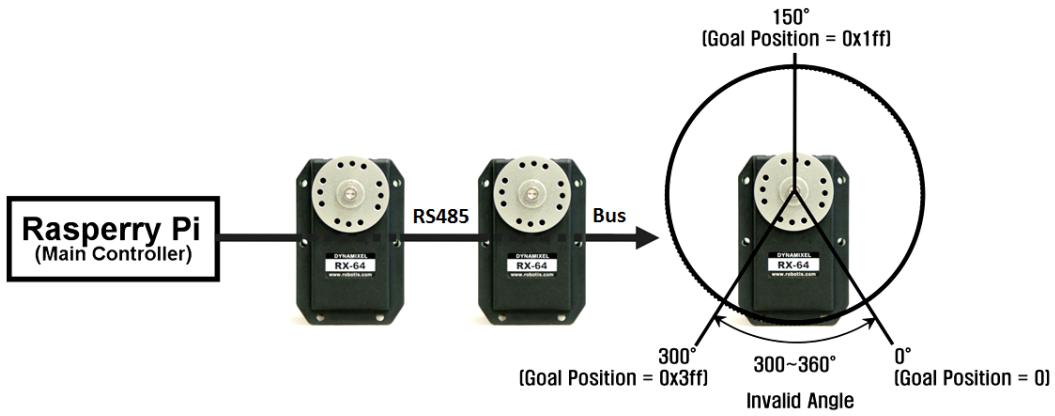


Abbildung 3.6: Dynamixel RX-64 Motoren [19]

Zur Kommunikation mit den RX-64 Motoren und der Ansteuerung dieser wird ein USB-RS485-Konverter Kabel 3.7 von der Firma FDTI verwendet [18]. Dieser Konverter enthält auf der USB-Stecker-Seite einen integrierten Schaltkreis, der eine Signalwandlung der RS-485 Datenleitungen auf die USB-Schnittstelle abwickelt. Die RX-Motoren benötigen eine Versorgungsspannung zwischen 12V und 21V, die über eine externe Spannungsquelle erfolgt, da die Versorgungsspannung der USB-Schnittstelle von 5V nicht ausreichend ist. Dabei müssen die Spannungsquelle und der USB-RS485 Konverter eine gemeinsame Masse haben. Die entsprechend detaillierte Beschreibung der Verkabelung des USB-RS485 Konverters und der Pins des Motors sind im Datenblatt[18] dargestellt.



Abbildung 3.7: USB-RS485-Konverter der Firma FDTI [18]

Kapitel 4

Entwicklung der Steuerungsalgorithmen

Die Steuerungsalgorithmen dienen der Erzeugung und Realisierung von Lauf- und Körperbewegungen von Laufrobotern [12]. Für die im Kapitel 3 vorgestellten Laufmaschine musste ein Steuerungsalgorithmus entwickelt werden. Sofern die verschiedenen Typen sechsbeiniger Maschinen einen starren Körper und Beine mit entsprechend ähnlicher Kinematik besitzen, können die in diesem Kapitel beschriebenen Algorithmen dafür angewendet werden. Dieses Kapitel ist an die Dissertation [12] angelehnt und dieser können die ausführlichen Erläuterungen der in diesem Kapitel vorgestellten Thematiken entnommen werden.

Zunächst wird die Definition der Trajektorien für alle Beine dargelegt und veranschaulicht. Daraufhin werden die verwendeten Koordinatensysteme des Laufroberts für die Koordinierung der Bewegungen des Roboterkörpers und die Beine genauer erläutert. Im nächsten Schritt musste eine kinematische Analyse des Laufroberts, insbesondere der kinematischen Kette der Beine durchgeführt werden. Hierfür wird die Herleitung kinematischer Transformationsgleichungen, sowohl direkter als auch inverser Kinematik beschrieben. Anschließend wird auf die Definition der Gelenke zur Identifikation bei der Implementierung der Algorithmen eingegangen.

4.1 Beintrajektorie

Neben den statisch stabilen Laufmustern für die Fortbewegung müssen die Trajektorien für alle Beine definiert werden, die das Voranschreiten der Laufmaschine realisieren. Die Abbildung 4.1 zeigt die Trajektorie eines Endeffektors im kartesischen Raum. Der Ansatz zur Erzeugung der Bahntrajektorien wurde aus dem Projekt B.E.T.H¹ entnommen.

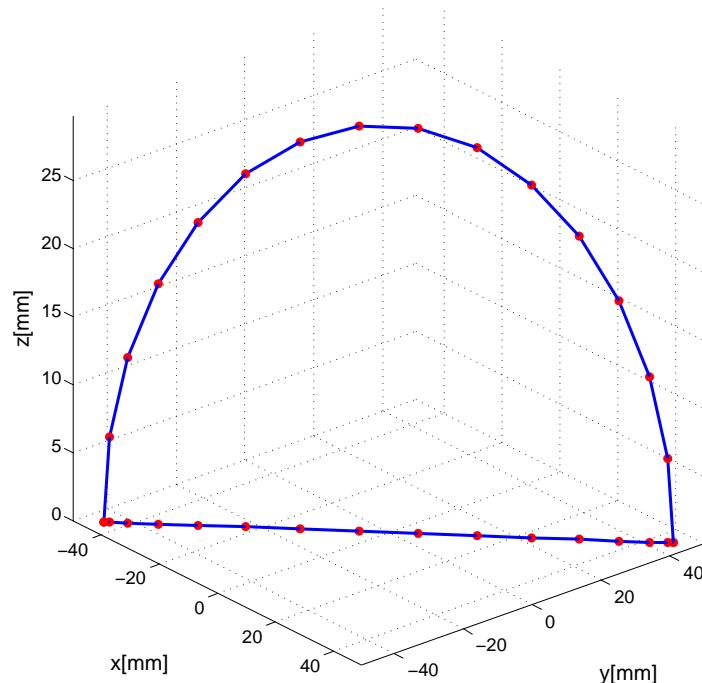


Abbildung 4.1: Die Trajektorie für die Beine

Die einzelne Punkte bzw. Positionen des Endeffektors im kartesischen Raum kann für die Schwingphase ($A_{x_i} \neq 0, A_{y_i} \neq 0, A_{z_i} \neq 0$) mit

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} A_{x_i} \cos(\pi \frac{i}{n}) \\ A_{y_i} \cos(\pi \frac{i}{n}) \\ A_{z_i} \sin(\pi \frac{i}{n}) \end{pmatrix} \quad (4.1)$$

für $i = 0, \dots, n$, mit n = Anzahl der Punkte dargestellt werden. Während die Gleichung (4.2) die Stützphase ($A_{x_i} \neq 0, A_{y_i} \neq 0, A_{z_i} = 0$) eines Schrittes beschreibt. Mittels der Amplituden ($A_{x_i}, A_{y_i}, A_{z_i}$) kann die Ausrichtung der Trajektorie beeinflusst werden. So kann beispielsweise der X-Anteil mit $A_{x_i} = 0$ zu Null gesetzt

¹<http://www.projectsofdan.com/?tag=phantomx>, Stand: Juli 2014

werden (Abb. 4.2), um die Trajektorie in die Richtung der X-Koordinatenachse auszurichten.

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} A_{x_i} \cos(\pi \frac{i}{n}) \\ A_{y_i} \cos(\pi \frac{i}{n}) \\ 0 \end{pmatrix} \quad (4.2)$$

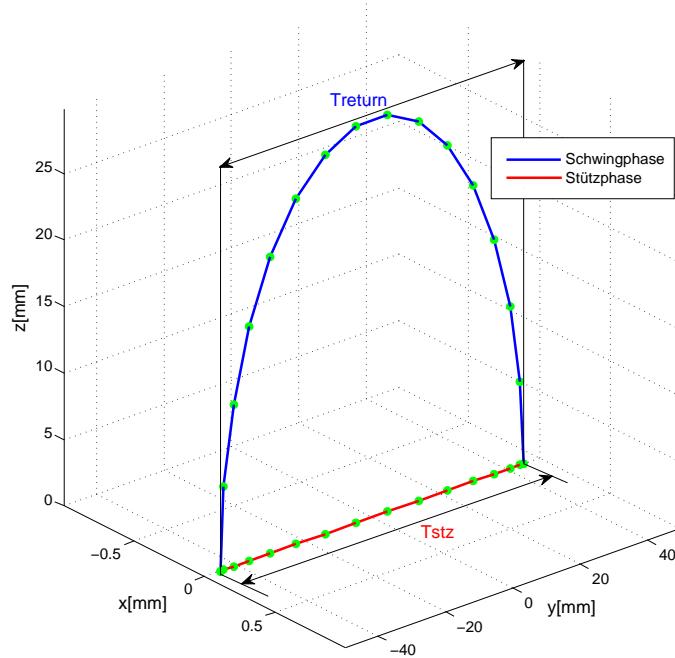


Abbildung 4.2: Die Phasen der Trajektorie

Hierfür müssen bestimmte Randbedingungen berücksichtigt werden, die durch das Laufmuster und der beabsichtigten Form der Trajektorie vorgegeben werden. Die Trajektorie wird innerhalb des im Kapitel 3.1.2 vorgestellten Arbeitsbereichs gelegt. Die notwendigen Schritte bei der Fortbewegung werden in zwei Phasen unterteilt: einer Stütz- und Returnphase. Diese Phasen unterliegen zeitlichen Bedingungen. So setzt sich die benötigte Zeit T (Gleichung (4.3)) eines Laufzyklus aus der Zeit für die Stützphase T_{stz} und die Returnphase T_{return} zusammen.

$$T = T_{stz} + T_{return} \quad (4.3)$$

Das Verhältnis von Stützzeit zur Zykluszeit wird als Stützverhältnis β (*duty factor*) bezeichnet (Gleichung 4.4).

$$\beta = \frac{T_{stz}}{T} \quad (4.4)$$

Bei einem Stützverhältnis $\beta \leq 0,5$, sind nicht immer genügend Füße in Bodenkontakt. Eine statisch stabile Fortbewegung kann bei $\beta \geq 0,5$ erzielt werden, damit mindestens drei Füße zu jeder Zeit auf dem Boden sind. Die implizite Beschreibung der vorgestellten Bedingungen erfolgt durch die Laufmuster (Kapitel 2.1). Bei diesen wird festgehalten, wie und in welcher Reihenfolge die Beine zur Fortbewegung eingesetzt werden.

4.2 Koordinatensysteme am Laufroboter

Zur Steuerung der Laufmaschine werden verschiedene Koordinatensysteme als Bezugssysteme verwendet. Dabei sind es kartesische Koordinatensysteme, deren Koordinatenachsen eine orthogonale Basis bilden, d.h. die Koordinatenachsen stehen paarweise senkrecht aufeinander und sind somit Basisvektoren für sämtliche Einheitsvektoren. Auf diese bezieht sich die Koordininierung der Bewegungen unter kinematischen Bedingungen.

4.2.1 Definition der Koordinatensysteme

Im Folgenden werden die Koordinatensysteme der Laufmaschine genauer erläutert. In der Abbildung 4.3 ist die schematische Darstellung des Körpers mit definierten Koordinatensystemen zu sehen.

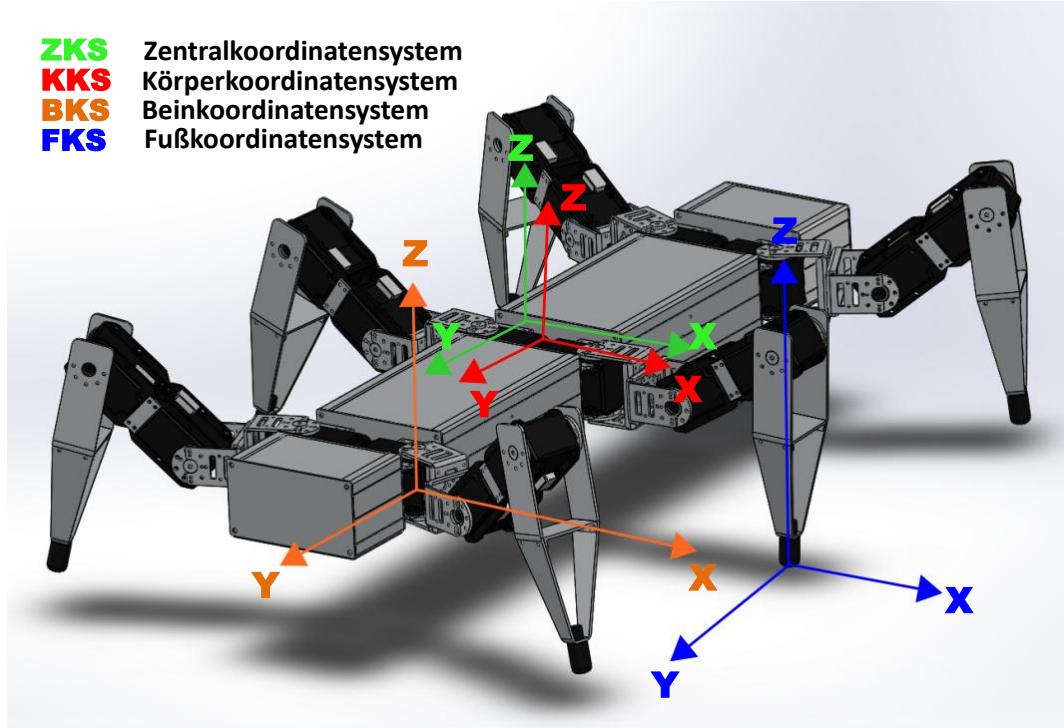


Abbildung 4.3: Definierte Koordinatensysteme am Roboter

Das Körperkoordinatensystem {KKS; X, Y, Z}

Dieses Koordinatensystem dient zur Beschreibung der Geometrie des Roboterkörpers und daran befestigten Objekte. Es ist ein mit dem Roboter verbundenes kar-

tesisches Koordinatensystem, das bezogen auf den Roboterkörper ortsfest ist. Der Ursprung liegt dabei in der Mitte des Körpers. Die durch Koordinatenachsen X und Y bildende Koordinatenebene ist parallel zur Grundfläche des Roboterkörpers.

Das Zentralkoordinatensystem $\{ZKS; X, Y, Z\}$

Das Zentralkoordinatensystem (ZKS) dient sowohl als ein Bezugskoordinatensystem für die Bein- und Körperbewegungen als auch zur Lokalisierung des Roboters. Es ist ein kartesisches Koordinatensystem und ist dem Köperkoordinatensystem deckungsgleich, sofern der Körper sich in der Ausgangsposition befindet.

Das Beinkoordinatensystem $\{BKS^{(i)}; X^{(i)}, Y^{(i)}, Z^{(i)}\}$

Die kartesische Beinkoordinatensysteme (BKS) $\{ BKS^{(i)}; X^{(i)}, Y^{(i)}, Z^{(i)} \}$ mit $i = (1, \dots, 6)$ werden jeweils mit Indizes versehen. Der Ursprung des $BKS^{(i)}$ -Koordinatensystems liegt dabei jeweils am Beinbefestigungspunkt. Die Achse $X^{(i)}$ ist parallel zum Normalenvektor der Seitenfläche des Roboters. Die Seitenfläche liegt in der durch die Koordinatenachsen $Y^{(i)}$ und $Z^{(i)}$ aufgespannten Ebene. Die Koordinatenachse $Z^{(i)}$ ist parallel zur Koordinatenachse Z des Köperkoordinatensystems.

Das Gelenkkoordinatensystem $\{ q^{(i)}; \alpha^{(i)}, \beta^{(i)}, \gamma^{(i)} \}$

Jedes Bein besteht aus drei aktiven, rotatorischen Freiheitsgraden, die mittels Gelenkkoordinaten beschrieben werden. Die drei Freiheitsgrade werden in dem Vektor q zusammengefasst. Dabei werden die Freiheitsgrade ausgehend vom Körper mit $\alpha^{(i)}$ für das erste Gelenk, $\beta^{(i)}$ für das zweite und $\gamma^{(i)}$ für das dritte Gelenk bezeichnet [27].

Das Fußkoordinatensystem $\{FKS^{(i)}; X^{(i)}, Y^{(i)}, Z^{(i)}\}$

Die kartesischen Beinkoordinatensysteme (FKS) $\{ FKS^{(i)}; X^{(i)}, Y^{(i)}, Z^{(i)} \}$ mit $i = (1, \dots, 6)$ haben ihren Ursprung im Endpunkt der Beinspitze. Die Koordinatenachse $Z^{(i)}$ ist senkrecht zur dritten Gelenkachse, während die Koordinatenachse $Y^{(i)}$ parallel zur dritten Gelenkachse ist.

4.2.2 Beziehungen der Koordinatensysteme zueinander

Die oben vorgestellten Koordinatensysteme werden zur Beschreibung bestimmter Bewegungen bzw. Vorgänge sowohl für das gesamte Robotersystem als auch für die Teilsysteme benötigt. Um Bezug zwischen den einzelnen Koordinatensystemen des Roboters herzustellen, lassen sich die Transformation ausführen.

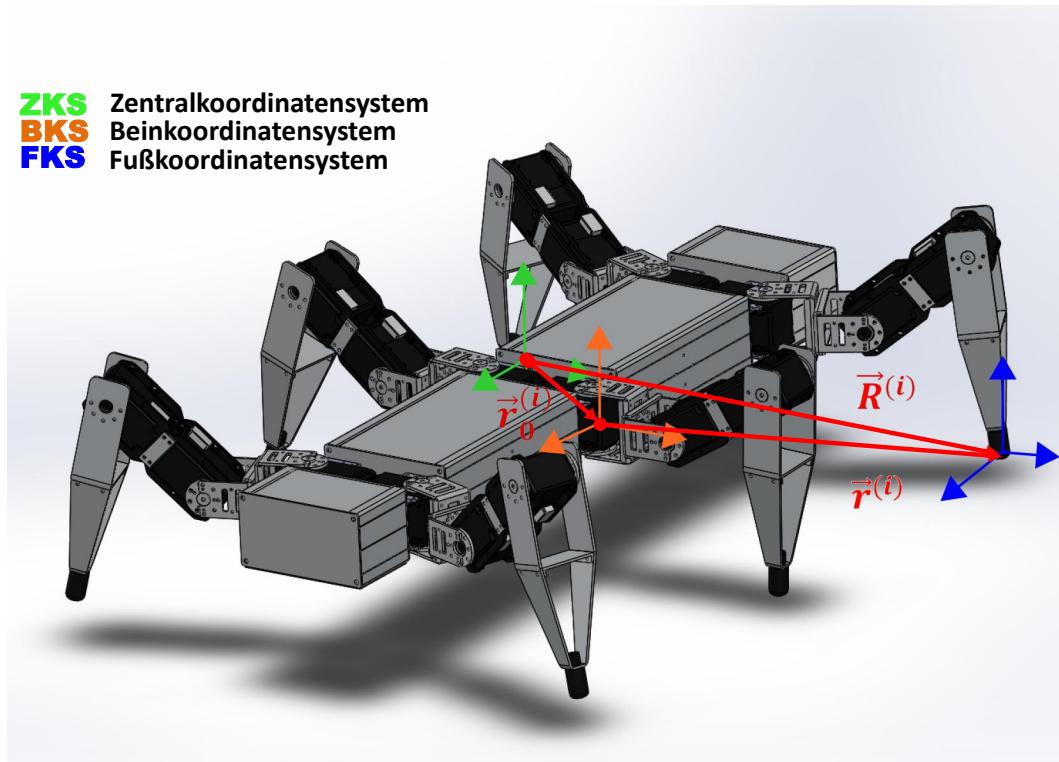


Abbildung 4.4: Beziehungen zwischen den Koordinatensystemen am Laufroboter

Zur Beschreibung der Bewegung und Positionierung des Roboters dient das Zentralkoordinatensystem. Dieses enthält Angaben zur den Orten der Beinbefestigungspunkte und der Fußpunkte. Mit dem Vektor $\vec{R}^{(i)}$ lässt sich die Verbindung des Ursprungs des Zentralkoordinatensystems mit dem Koordinatenursprung des Fußkoordinatensystems, der als Fußspitze bzw. Endeffektor angenommen wird, ausdrücken. Die Beinbefestigungspunkte sind durch die Vektoren $\vec{r}_0^{(i)}$ mit dem Zentralkoordinatensystem verbunden. Die Beinbefestigungspunkte, die im Ursprung der Beinkoordinatensysteme liegen, werden durch die Vektoren $\vec{r}^{(i)}$ mit den Fußkoordinatensystemen verbunden. Folgende Gleichung (4.5) beschreibt den Zusammenhang zwischen den oben beschriebenen Vektoren.

$$\vec{R}^{(i)} = \vec{r}_0^{(i)} + \vec{r}^{(i)} \quad (4.5)$$

Die Position der Fußpunkte beschreibt das kartesische Beinkoordinatensystem, das über die inverse und direkte Kinematikbeziehung mit den Gelenkkoordinaten verbunden ist.

Das kartesische Körperkoordinatensystem steht über zwei Rotations- und Translationsoperationen in Verbindung mit dem Zentralkoordinatensystem. Diese Operationen beschreiben die Lage des Roboterkörpers in Bezug zum Zentralkoordinatensystem.

4.2.3 Kinematik des Roboterkörpers

Die Körperbewegungen ermöglichen ein gezieltes Ausrichten und Orientieren des Körpers. Diese Operationen werden mittels existierender Freiheitsgrade des Körpers erreicht. Hierfür müssen Vektoren ermittelt werden, die die Orientierung und die Lage bei gegebenen Fußpunkten bestimmen. Bei der sechsbeinigen Laufmaschine werden die Beine zur Positionierung des Roboterkörpers genutzt. Aus diesem Grund müssen zur Beschreibung der Körperposition und der zur späteren Ableitung der Beinposition, die Vektoren der Beinfestigungspunkte bestimmt werden. Dabei dient das Zentralkoordinatensystem als Referenzkoordinatensystem. Die Körperposition wird in zwei Schritten ermittelt:

1. Translation in Relation zu den Körperachsen.
2. Rotation um die Zentralkoordinatenachsen.

Die Befestigungspunkte ${}^{KKS}\vec{r}_{0,0}^{(i)}$, die über feste geometrische Beziehungen miteinander verbunden sind, stellen die Ausgangspunkte dar. Die Zentral- und Körperkoordinatensysteme können in der Ausgangsphase deckungsgleich angenommen werden. Demzufolge sind die Koordinaten der Beinfestigungspunkte ${}^{ZKS}\vec{r}_{0,0}^{(i)}$ bekannt. Im ersten Schritt wird die Translation der Beinfestigungspunkte entlang der Achsen des Zentralkoordinatensystems verschoben.

$${}^{ZKS}\vec{r}_{0,0}^{(i)} = Trans(ZKS, \Delta \vec{X}_{(1,KKS)}) \cdot {}^{ZKS}\vec{r}_{0,0}^{(i)} = {}^{ZKS}\vec{r}_{0,0}^{(i)} + \Delta \vec{X}_{(1,KKS)} \quad (4.6)$$

Die Gleichung (4.6) zeigt eine Verschiebung, die bezogen auf den Roboterkörper erzeugt werden kann. Im zweiten Schritt wird die Rotation um die Koordinatenachsen des Zentralkoordinatensystems durchgeführt.

$${}^{ZKS}\vec{r}_{0,1}^{(i)} = \text{Rot}(ZKS_z, \alpha_{KKS,z}) \cdot \text{Rot}(ZKS_y, \alpha_{KKS,y}) \cdot \text{Rot}(ZKS_x, \alpha_{KKS,x}) \quad (4.7)$$

Die Gleichung (4.7) beschreibt die Neigung und Drehung des Roboterkörpers. Die Rotation kann um alle drei Koordinatenachsen gleichzeitig umgesetzt werden. Die Abbildung 4.5 veranschaulicht die oben beschriebenen Schritte.

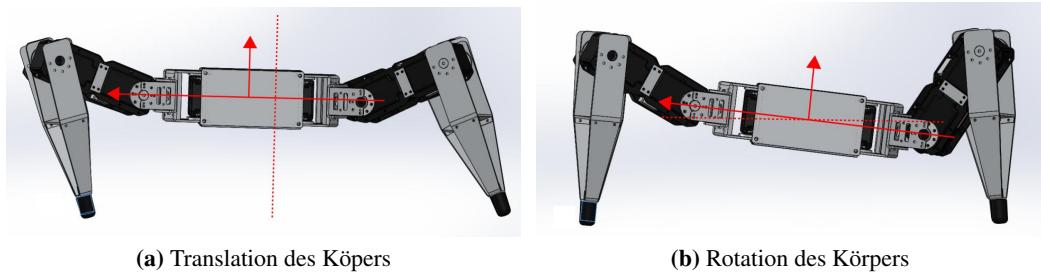


Abbildung 4.5: Kinematik des Laufroboters

4.2.4 Roboterbeine

Zur Erzeugung einer Laufbewegung und einer Körperbewegung, müssen die Beinbewegungen umgesetzt werden. Um dies zu ermöglichen muss die Information zur Beinposition gewonnen werden. Die Beinposition lässt sich aus folgenden Parametern ableiten. Die Positionsfolge der Fußpunkte beschreibt die Laufbewegung, während die Körperbewegung durch die Positionsfolge der Beinbefestigungspunkte bestimmt wird. Dieser Zusammenhang verdeutlicht die graphische Darstellung in der Abbildung 4.4. So kann der Vektor der Beinposition aus der folgenden Vektorgleichung ermittelt werden.

$${}^{ZKS}\vec{R}^{(i)} = {}^{ZKS}\vec{r}_0^{(i)} + {}^{ZKS}\vec{r}^{(i)} \quad (4.8)$$

Durch die Umstellung der Gleichung (4.8) nach der gesuchten Größe (Gleichung (4.9)) ergibt sich der Vektor der Beinposition.

$${}^{ZKS}\vec{r}^{(i)} = {}^{ZKS}\vec{R}^{(i)} - {}^{ZKS}\vec{r}_0^{(i)} \quad (4.9)$$

Zu beachten ist, dass die Vektoren gebundene Vektoren im Zentralkoordinaten darstellen und deren Anfangspunkte mit den Beinbefestigungspunkten zusammenfallen. Aus diesem Grund müssen die gebundenen Vektoren in die lokalen Beinkoordinatensysteme transformiert werden (Gleichungen (4.10)). Für die Darstellung der Beinkoordinaten im Beinkoordinatensystem muss eine Rotation der Koordinaten vorgenommen werden.

$${}^{BKS}\vec{r}^{(i)} = {}_{ZKS}^{BKS} T \cdot {}^{ZKS}\vec{r}^{(i)} \quad (4.10)$$

Aus diesen transformierten Vektoren werden mittels inverser Kinematik die Gelenkstellungen ermittelt und die Beine entsprechend positioniert.

Direkte Kinematik der Beine

Zur kinematischen Beschreibung der Beine wurden die Formeln nach dem Denavit-Hartenberg Verfahren hergeleitet. Hierzu werden zunächst die DH Parameter ermittelt. Die Abbildung 4.6 stellt den konstruktiven Aufbau eines Beines mit den hierfür notwendigen geometrischen Parametern dar. Die zugehörigen Werte sind in der Tabelle 4.1 zusammenfassend dargestellt. Im nachfolgenden ist die Vorgehensweise zur Bestimmung der Parameter beschrieben. Die grundlegende Beschreibung zum Denavit-Hartenberg Verfahren ist im Kapitel 2.2.1 erläutert.

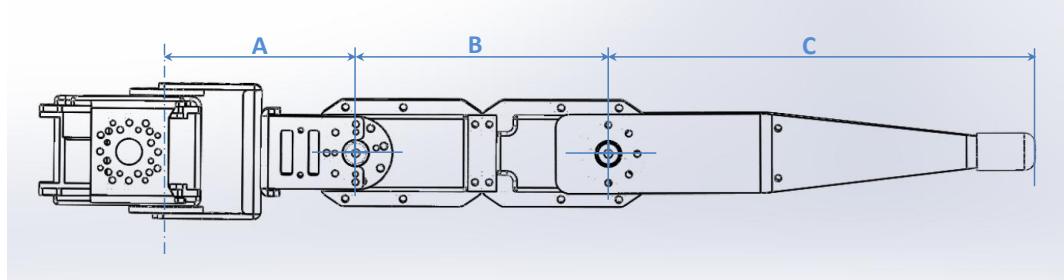
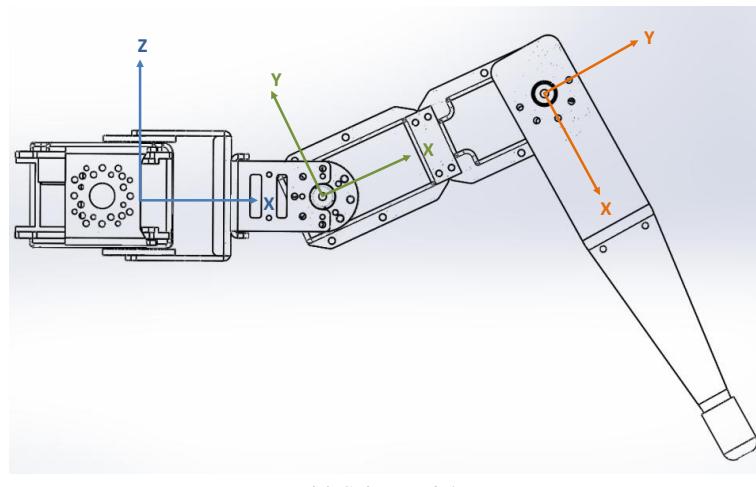


Abbildung 4.6: Die geometrischen Parameter eines Beines

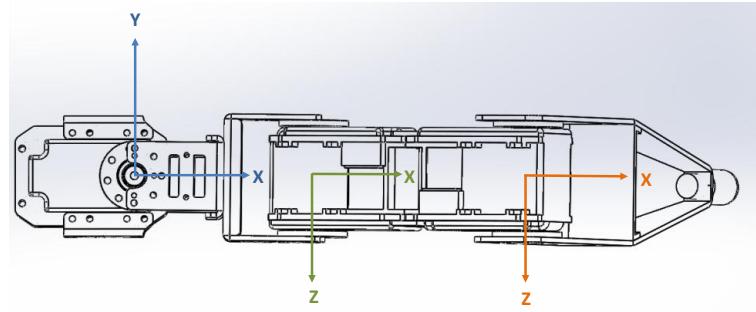
Tabelle 4.1: Geometrische Parameter

Parameter	Wert
A	72 mm
B	97 mm
C	163 mm

Zunächst musste das Anfangskoordinatensystem, in diesem Fall das Beinkoordinatensystem für die Herleitung der Kinematik festgelegt werden (Abb. 4.7). Der Ursprung des Beinkoordinatensystems ($BKS^{(i)}$) ist mittig auf der ersten Gelenkachse angeordnet. Die Orientierung ist wie folgt, die Koordinatenachse $Z^{(i)}$ des Beinkoordinatensystems ist antiparallel zur Schwerkraft und die Koordinatenachse $Y^{(i)}$ liegt in der Laufrichtung. Durch die ergänzende Koordinatenachse $X^{(i)}$ entsteht ein rechtsorientiertes kartesisches Koordinatensystem. Für jedes weitere Gelenk wurde festgelegt, dass die Koordinatenachse $Z^{(i)}$ auf der Drehachse und die Koordinatenachse $X^{(i)}$ auf der Normalen zwischen den Drehachsen der Gelenke liegen.



(a) Seitenansicht



(b) Draufsicht

Abbildung 4.7: Die definierten Achsenkoordinatensysteme eines Beines

Die Tabelle 4.2 zeigt die resultierenden DH Parameter für ein Bein. Die Anordnung der Gelenkachsen ist parallel oder senkrecht aufeinander stehend, dadurch ergeben sich Vereinfachungen notwendiger Transformationsschritte. Dies ist durch die Nullwerte der DH Parameter in der Tabelle 4.2 verdeutlicht. Aufgrund der identischen mechanischen Struktur aller sechs Beine der Laufmaschine gelten die DH Parameter genauso für die restlichen fünf Beine.

Tabelle 4.2: DH Parameter eines Beines

Link	a_n	δ_n	q_n
1	A	$\frac{\pi}{2}$	α
2	B	0	β
3	C	0	γ

Nach der Definition der Koordinatensysteme in der kinematischen Kette und der Ermittlung der DH Parameter lässt sich die Transformationsgleichung für die direkte Kinematik aufstellen.

$$\begin{aligned} {}_3^0T = & Trans(X, A) \cdot Rot(X, \frac{\pi}{2}) \cdot Rot(Z, \alpha) \cdot \\ & Trans(X, B) \cdot Rot(Z, \beta) \cdot \\ & Trans(X, C) \cdot Rot(Z, \gamma) \end{aligned}$$

Die Transformationsmatrix für das gesamte Bein wird durch das Ausmultiplizieren der einzelnen homogenen Transformationsmatrizen erzielt.

$${}^0_3T = \begin{pmatrix} {}^0_3R & {}^0\vec{r} \\ 0 & 1 \end{pmatrix} \quad (4.11)$$

Bei der resultierenden Transformationsmatrix für das gesamte Bein wird zur Positionierung des Endeffektors im kartesischen Raum der Ortsvektor ${}^0\vec{r}$ genutzt. Die hergeleiteten Gleichungen des Ortsvektors sind wie folgt:

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} \cos(\alpha)(C \cos(\beta + \gamma) + \cos(\beta) + A) \\ \sin(\alpha)(C \cos(\beta + \gamma) + \cos(\beta) + A) \\ C \sin(\beta + \gamma) + B \sin(\beta) \end{pmatrix} \quad (4.12)$$

Inverse Kinematik der Beine

Die Beschreibung von erzeugten Trajektorien der Roboterbewegungen im kartesischen Raum wird mittels inverser Kinematik erreicht. Dabei wird der Endeffektor im kartesischen Raum jeweils nach einem Punkt von der erzeugten Trajektorie koordiniert und die notwendigen Gelenkstellungen der Gelenke zur Realisierung dieser Position mit Hilfe der inversen Kinematik ermittelt. Hierfür müssen zunächst die Gleichungen für die inverse Kinematik bestimmt werden. Die Lösung des inversen Problems erfolgt analytisch, da sich die inverse Kinematik damit sehr effizient

berechnen lässt. Die Vorgehensweise zur Herleitung einer analytischen Lösung ist folgende. Der kartesische Raum in dem der Endeffektor koordiniert wird, lässt sich auf zwei zweidimensionale Koordinatensysteme zurückführen:

1. Ein XY-System, das auf der von der X- und Y-Achse aufgespannten Ebene des kartesischen Koordinatensystems liegt und dessen Ursprung sich auf der ersten Gelenkachse befindet (Abb. 4.8a).
2. Ein zum XY-System senkrechttes XZ-System, das auf der von der X- und Z-Achse aufgespannten Ebene des kartesischen Koordinatensystems liegt und dessen Ursprung sich auf der ersten Gelenkachse befindet (Abb. 4.8b).

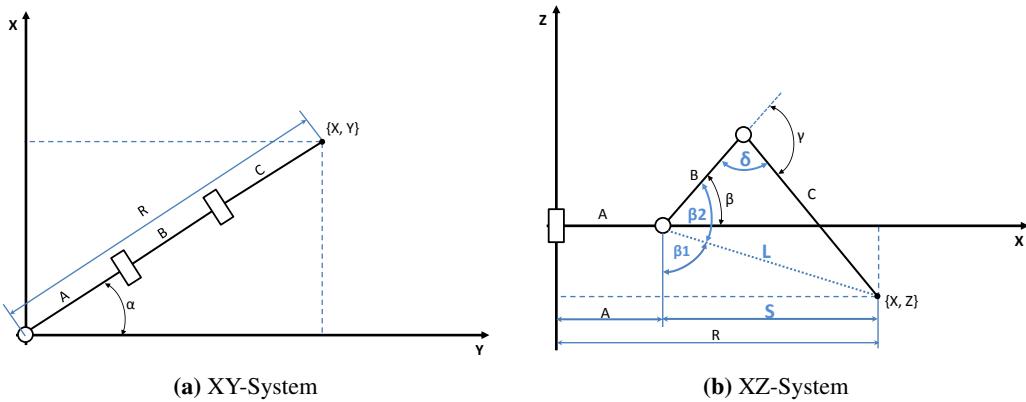


Abbildung 4.8: Bestimmung der inversen Kinematik anhand der geometrischen Beziehungen an einem Bein

Der erste Gelenkwinkel α wird mit Hilfe des XY-Systems bestimmt. Die Koordinaten sind sowohl im XY-System als auch auf Grund der mechanischen Struktur der Laufmaschine im Beinkoordinatensystem bekannt, da die Ursprünge beider Koordinatensysteme im ersten Gelenk liegen und sich schneiden. Dividiert man die ermittelte Y-Größe durch X-Größe, so erhält man die Lösung für den Gelenkwinkel α .

$$\alpha = \arctan\left(\frac{Y}{X}\right) \quad (4.13)$$

Der Term in dem Nenner kann Null werden. Daher empfiehlt sich bei der Implementierung die Funktion *atan2* zu verwenden, da diese solche Fälle behandeln kann.

Diese steht bei der C++ Programmiersprache zur Verfügung. Der Parameter R in der Abbildung 4.8a beschreibt den Abstand des Effektors zum Ursprung.

$$R = \sqrt[3]{X^2 + Y^2}$$

Die Abbildung 4.8b stellt die geometrischen Beziehungen des Beins im XZ-System dar. Mittels diesem System werden die fehlenden Gelenkwinkel β und γ bestimmt. Für die Berechnung dieser werden zunächst die Hilfsvariablen $S, L, \beta_1, \beta_2, \delta$ definiert.

$$\begin{aligned} S &= R - A \\ L &= \sqrt[3]{S^2 + Z^2} \\ L &= \sqrt[3]{(R - A)^2 + Z^2} \end{aligned}$$

Der Gelenkwinkel β wird in β_1 und β_2 aufgeteilt. Der Hilfswinkel β_1 lässt sich mittels der Variable S und der X-Koordinate berechnen.

$$\beta_1 = \arctan \left(\frac{S}{Z} \right) \quad (4.14)$$

Zur Ermittlung des β_2 Winkels wird der Kosinussatz genutzt. Dabei werden die gegebenen Seiten des durch geometrische Größen A, D des Beins und die berechnete Hilfsvariable L aufgespannten Dreiecks, genutzt.

$$\beta_2 = \arccos \left(\frac{1}{2} \cdot \frac{C^2 + L^2 - B^2}{CR} \right) \quad (4.15)$$

Der resultierende β_2 Winkel und β_1 bilden den gesamten β Winkel.

$$\beta = \beta_1 + \beta_2 \quad (4.16)$$

Ebenfalls wird der letzte fehlende Gelenkwinkel γ mit Hilfe des Kosinussatzes berechnet. Hierfür wird zunächst der Hilfswinkel δ ermittelt.

$$\delta = \arccos \left(\frac{1}{2} \cdot \frac{C^2 + B^2 - Z^2}{CB} \right) \quad (4.17)$$

Durch die Subtraktion von 180° wird der Gelenkwinkel γ definiert.

$$\gamma = \delta - 180^\circ \quad (4.18)$$

Somit liegen für alle drei Gelenkwinkel die analytischen Lösungen vor. Für die Gelenke α und β_1 ist die erwähnte *atan2* Funktion zu verwenden, da die Nenner in der Gleichung (4.13) und (4.14) Null werden können.

4.3 Nummerierung der Motoren

Für die Implementierung der in diesem Kapitel beschriebenen Fortbewegungen, sowohl vom Roboterkörper als auch der Beine nach den vordefinierten Trajektorien werden die hergeleiteten Gleichungen eingesetzt. Die Berechnung der Resultate dieser Gleichungen müssen dem Laufroboter bekannt gegeben werden. Im Genauesten werden beispielsweise die mittels inverser Kinematik errechneten Gelenkstellungen oder die durch direkte Kinematik berechnete Positionierung des Endeffektors eines Beines an die Steuerungselektronik der Dynamixel RX64-Motoren weitergereicht, um die gezielte Positionierung der Motoren auszuführen. Hierfür muss jeder einzelne Motor mit einer Identifikationsnummer versehen werden und zur Ansteuerung im ROS eindeutig identifizierbar sein. Dazu zeigt die Abbildung 4.9 die Gesamtstruktur der nummerierten Gelenke an der Laufmaschine.

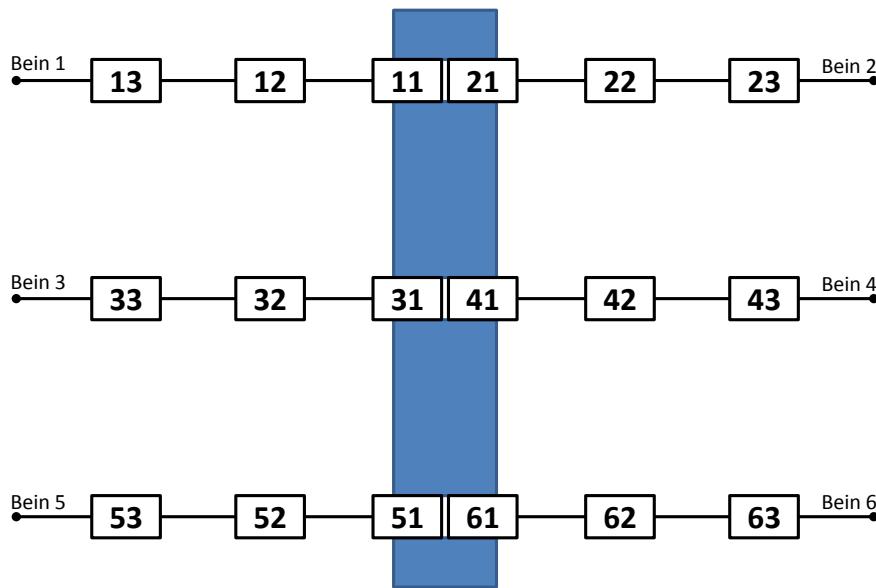


Abbildung 4.9: Nummerierung der Motoren des Laufroboters

Kapitel 5

Implementierung des Steueralgorithmus

In diesem Kapitel wird die Umsetzung der im vorhergehenden Kapitel vorgestellte Konzepte unter dem Robot Operating System (ROS) beschrieben. Das ROS System bietet bereits implementierte Algorithmen, Bibliotheken und Treiber die in Stacks und Packages versammelt sind und je nach Robotersystem einzelne bestimmte Funktionalitäten zur Verfügung stellen. Die Steueralgorithmen wurden in der Programmiersprache C++ implementiert, welche einer der von dem ROS Framework unterstützten Programmiersprachen ist.

Zunächst werden der Aufbau und die Organisation des im Rahmen dieser Arbeit entwickelten ROS Packages auf der Datei-Ebene dargestellt. Dabei werden wichtige Konfigurationsdateien zur Anbindung der Hardware erläutert. Hierzu werden die verwendeten Stacks des ROS Systems und deren Funktionalitäten zur Umsetzung des Steuerungskonzepts präsentiert.

Der zweite Teil beschreibt die Implementierung der Steuerungsalgorithmen anhand eines Nodes, im einzelnen sind die Methoden und deren Funktionen bzw. Aufgaben dargelegt. Hierfür wird die Umsetzung direkter und inverser Kinematik mittels ROS Bibliotheken, sowie die Laufmustern präsentiert. Ebenfalls wird auf die Ansteuerung der Motoren, sowie die Anbindung des Joysticks unter ROS erläutert.

5.1 Akrobat Paket

Zur Erzeugung der Fortbewegung und somit die Steuerung der aufgebauten Laufmaschine ist im Rahmen dieser Thesis ein ROS Package entwickelt worden. Dieses Paket stellt neben der Strukturierung der entwickelten Algorithmen, Bibliotheken, Code auf der Datei-Ebene die und die Funktionalitäten für das Robotersystem zur Verfügung. Das Ziel dabei ist eine möglichst überschaubare Funktionseinheit mit einer Gewährleitung ferner die flexible Weiterentwickelung des Robotersystems vornehmen zu können, zu erschaffen.

Organisation

Die Strukturierung des entwickelten Pakets auf der Datei-Ebene ist in der folgenden Abbildung 5.1 dargestellt.

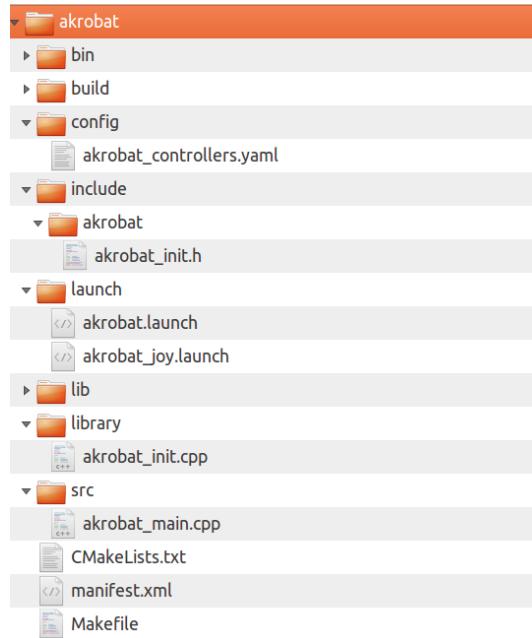


Abbildung 5.1: Der Aufbau des Akrobot-Pakets

Dabei sind die Inhalte folgender fünf Verzeichnisse, die im Rahmen der Thesis entwickelten Steuerungskonzepts relevant sind, zu betrachten.

config/: Dieses Ordner beinhaltet die Konfigurationsdatei zu RX64 Motoren.

include/akrobot/: Hier befindet sich die entwickelte Header-Datei `akrobot_init.h`.

launch/: Die Launch-Dateien werden in diesem Ordner abgelegt.

library/: Diese Ordner enthält die implementierte Datei `akrobot_init.cpp`.

src/: In diesem Ordner ist `akrobot_main.cpp`, der eigentliche Node zum gesamten Robotersystem.

Konfigurationen

Config Um die RX64-Motoren ansteuern zu können, sind zunächst die Parameter der Servomotoren einzustellen. Dies ist mittels der Datei `config/akrobot_controllers.yaml` zu realisieren. Der Ansatz hierfür ist dem `dynamixel_tutorials` Paket entnommen. In der `akrobot_controllers.yaml` können die Namen der Controller mit den dazugehörigen Identifikationsnummern von Servomotoren vergeben werden. Mit dem Controller wird die interne Elektronik der RX64 Motoren bezeichnet, die die Steuerung dieser Motoren abwickelt. Die Motoren sind im Gelenkmodus eingestellt und die Namen der Gelenke sind in der Konfigurationsdatei definiert. Die Initialisierung des minimalen, maximalen Arbeitsbereichs des Gelenks, sowie der Anfangsposition bzw. neuen Nullpunkts sind ebenfalls in der yaml Datei festgelegt. Die entsprechenden Einstellungen wurden für alle 18 Gelenke vorgenommen. Der Abschnitt der `config/akrobot_controllers.yaml` zeigt die eingestellten Parameter anhand eines Gelenks.

```
controller_m11:
  controller:
    package: dynamixel_controllers
    module: joint_position_controller
    type: JointPositionController
  joint_name: pan_joint_m11
  joint_speed: 1.0
  motor:
    id: 11
    init: 545
    min: 207
    max: 682
```

Launch Nachdem die Konfigurationen für jedes einzelne Gelenk bereit sind, kommen die Treiber zum Einsatz, um über die Controller die Motoren bewegen zu können. Hierfür stehen die Launch Dateien zur Verfügung, die es ermöglichen weitere Einstellungen vorzunehmen. Alle zuständigen Nodes für die Treiber können bereits mit einer Launch Datei aktiviert und die Kommunikation mit den Motoren aufbaut werden. Die `launch/akrobot.launch` startet drei implementierte Nodes aus folgenden Stacks: zwei aus dem `dynamixel_motor` und einen aus dem

`joystick_drivers`. Zuerst wird der Node mit dem Namen `dynamixel_manager` gestartet. Dieses Node aktiviert die Treiber und baut die Kommunikation zu jedem Controller der Motoren auf. Für diesen Node werden folgende Parameter konfiguriert:

1. Der Port Name (`/dev/ttyUSB0`) für den Kommunikationsaufbau, welcher mit dem USB- RS485 Konverter übereinstimmen muss.
2. Die Baud Rate, die mit der eingestellten Baud Rate der Motoren übereinstimmt.
3. Der Bereich in dem die Identifikationsnummern (`min/max_motor_id`) liegen.

Für die gefundenen IDs in diesem Bereich werden die Treiber eine Kommunikation aufbauen. Wichtig ist dabei, dass der Bereich mit der kleinsten Identifikationsnummer des gesamten Robotersystems anfängt und mit der größten endet. Außerdem erzeugt dieses Node einen Topic `/motor_states/pan_tilt_port`, welcher den aktuellen Zustand der Motoren liefert. Im Genauren können die Informationen zur aktuellen Winkelposition, dem eingestellten Drehmoment, der Geschwindigkeit und der Betriebstemperatur überwacht werden. Dieses Node ermöglicht jedoch keine Ansteuerung der Motoren. Im Folgenden ist der Abschnitt der Launch Datei mit dem `dynamixel_manager` Node dargestellt.

```
<!-- -*- mode: XML -*- -->

<launch>
  <node name="dynamixel_manager" pkg="dynamixel_controllers"
    type="controller_manager.py" required="true" output="screen">
    <rosparam>
      namespace: dxl_manager
      serial_ports:
        pan_tilt_port:
          port_name: "/dev/ttyUSB0"
          baud_rate: 57142.0
          min_motor_id: 0
          max_motor_id: 44
          update_rate: 20
    </rosparam>
  </node>
```

Im zweiten Schritt ist der Node `dynamixel_controller_spawner` zu starten. Mit der Aktivierung dieses Nodes kann die Ansteuerung der Motoren umgesetzt werden, da für jeden Controller zwei Topics in ROS erzeugt werden. Einer der Topics ist `/controller_mxx/command`, mit Hilfe dessen die neue Positionierung des Motors bzw. Gelenks stattfindet. Der zweite Topic `/controller_mxx/state` liefert den Betriebszustand eines Motors. Alle definierten Controller in `config/akrobot_controllers.yaml` werden als Parameter an diesen Node überführt, damit dieser für diesen Controller die erwähnten Topics erzeugt. Folgender Abschnitt der Launch Datei stellt den zweiten Node dar.

```
<!-- Load controller configuration to parameter server -->
<rosparam file="$(find akrobot)/config/akrobot_controllers.yaml" command="load"/>

<!-- start specified joint controllers -->
<node name="dynamixel_controller_spawner" pkg="dynamixel_controllers"
      type="controller_spawner.py"
      args="--manager=dxl_manager
            --port=pan_tilt_port
            --type=simple
            controller_m11
            controller_m12
            controller_m13"
            ...
            controller_m63
            output="screen" >
</node>
```

Im Anschluss wird der Joy Node gestartet. Hierfür muss der Port Name konfiguriert werden, um die Kommunikation mit dem Joystick Empfänger über die USB Schnittstelle aufzubauen. Folgender Abschnitt aus der `akrobot.launch` zeigt den Joy Node.

```
<!-- start joy -->
<node respawn="true" pkg="joy" type="joy" name="akrobotJoy">
    <param name="dev" type="string" value="/dev/input/js1" />
</node>
</launch>
```

Manifest Die Datei `manifest.xml` stellt die Information zum entwickelten Akrobat Paket, die Abhängigkeiten von anderen Paketen, Compiler Flags usw. zur Verfügung. Der Aufbau der `manifest.xml` Datei ist wie folgt:

```
<package>
  <description brief="akrobot">
    akrobot
  </description>
  <author>stud</author>
  <license>BSD</license>
  <review status="unreviewed" notes="" />
  <url>http://ros.org/wiki/akrobot</url>
  <depend package="std_msgs"/>
  <depend package="rospy"/>
  <depend package="roscpp"/>
  <depend package="dynamixel_controllers"/>
  <depend package="dynamixel_driver"/>
  <depend package="dynamixel_msgs"/>
  <depend package="dynamixel_tutorials"/>
  <depend package="joy"/>
  <depend package="tf"/>

</package>
```

In `manifest.xml` sind die Abhängigkeiten von anderen Pakete zu erkennen. Die Liste der Abhängigkeiten stellt die Pakete dar, dessen Funktionalitäten für das Akrobat Paket bereitgestellt werden.

std_msgs: Dieses Paket enthält standardisierte Nachrichtentypen und Datentypen, die zum Informationsaustausch genutzt werden.

rospy: Dieses Paket bietet die Bibliotheken für die Programmiersprache Python in ROS.

roscpp: Dieses Paket bietet die Bibliotheken für die Programmiersprache C++ und ermöglicht die Implementierung der Algorithmen für ROS in C++.

dynamixel_motor: Dieser Stack besteht aus vier Dynamixel Paketen, die zur Ansteuerung der 18 Servomotoren genutzt werden.

joy: Das Joy Paket wickelt den Kommunikationsaufbau mit dem Joystick ab und ermöglicht die externe Ansteuerung der Laufmaschine.

tf: Dieses Paket stellt alle notwendigen Operationen zur Erstellung und Berechnung der Transformationsmatrizen. Dieses wird zur Transformationen zwischen Koordinatensystemen der Laufmaschine genutzt.

CMakeLists In der Datei CMakeLists.txt befinden sich die notwendigen Parameter zum Komplilieren des Akrobat Pakets. Dabei wurde diese um folgende Zeilen angepasst.

1. rosbuild_add_library(akrobot_init library/akrobot_init.cpp)
2. rosbuild_add_executable(akrobot src/akrobot_main.cpp)
3. target_link_libraries(akrobot akrobot_init)

5.2 Steuerungskonzept

Der im Verlauf dieser Arbeit entwickelte akrobot_main.cpp-Node stellt ein mathematisches Modell der Laufmaschine dar. Hiermit ist es möglich, die verschiedenen kinematischen Berechnungen des Laufroboters durchzuführen, um die Fortbewegung dieser zu realisieren. Dabei stehen Methoden zur Verfügung, die es ermöglichen die Laufmaschine per Joystick zu steuern. Im folgenden wird auf die implementierten C++ Quelldateien und deren Funktionalitäten eingegangen.

akrobot_init.hpp

Die akrobot_init.hpp Header Datei erfasst alle der Konstruktion betreffenden Parameter, um das Programm bei einer Modifikation der Roboterkonstruktion leicht anpassen zu können. Diese enthält die Werte für die mechanischen Beschränkungen der einzelnen Gelenke, die Initialposition des Lauroboters sowie weitere Parameter für den Joystick. Außerdem sind Definitionen der Koordinatensysteme und der Methoden der Klasse Akrobot festgelegt. Für die Wiederverwendbarkeit des Quelltextes sollen neue geometrische Parameter des Roboters als Konstanten in die Header Datei akrobot_init.hpp eingetragen werden.

akrobot_init.cpp

Diese Quelldatei enthält alle implementierten Funktionen der Klasse Akrobot, mittels derer die Steuerung des Roboters realisiert wird.

Zunächst werden die Koordinatensysteme des Roboters global instanziert. Dabei handelt es sich um vier Koordinatensysteme: Zentral-, Körper-, Bein- und Fußkoordinatensystem. Hierfür wurden folgende Structs definiert:

```

struct coordinateSystemStruct{
    legStruct leg[numberOfLegs]; //array of legs
};

struct legStruct{
    Vector3 footPresPos;           //present position
    Vector3 footInitPos;          //init position
    Vector3 footGlobPos;          //globale position
    Vector3 trajectoryPresPos;    //trajectory present positon
    floatJointStruct jointAngles; //joint angles
};

```

Jedes Koordinatensystem enthält ein Array der Größe sechs vom Typ legStruct. Jedes einzelne legstruct besteht aus fünf Variablen, vier davon sind vom Typ Vector3 und beschreiben die Positionen bezogen auf die verschiedenen Koordinatensysteme, während die letzte die Winkel (α, β, γ) der Gelenke enthält. Die Klasse Vector3 wurde für die Positionen gewählt, damit später die Transformationen mit dem tf Stack ausgeführt werden können.

Bevor die Steuerungsmethoden genutzt werden können, müssen im nächsten Schritt die entsprechenden Subscriber und Publisher Topics im Konstruktor der Klasse Akrobot erzeugt werden.

```

Akrobot::Akrobot()\{
    subJoy = n.subscribe<sensor_msgs::Joy>("joy", 10, &Akrobot::callRumblePad2Back, this);
    pubLeg1Joint1 = n.advertise<std_msgs::Float64>("/controller_m11/command",1);
    pubLeg1Joint2 = n.advertise<std_msgs::Float64>("/controller_m12/command",1);
    ...
}

```

Im Konstruktor wird an erster Stelle ein Subscriber mit dem Namen subJoy erzeugt. Dieser Subscriber abonniert die aktuellen Daten vom Joy Topic mit dem Messagetype `<sensor_msgs::Joy>`, wenn der Joystick betätigt wird. Die abgerufenen Sensordaten werden an die Funktion `Akrobot::callRumblerPad2Back()`

weitergereicht. So stellt die Funktion `Akrobot::callRumblerPad2Back()` eine Einheit dar, in der die externen Steuerungssignale für bestimmte Roboteroperationen verarbeitet und interpretiert werden.

Für die Ansteuerung der Motoren werden nach dem Start der `akrobot.launch` Datei vom Node `dynamixel_controller_spawner` zwei Topics für jedes einzelne Gelenk erzeugt. Einer der Topics ist `/controller_mxx/command`, mit Hilfe dessen die neue Positionierung des Motors bzw. Gelenks vorgenommen werden kann. Um beispielsweise das erste Gelenk (ID: 11) neu zu positionieren, muss die Übergabe der ermittelten Gelenkwinkel an den Topic `/controller_m11/command` mit dem Messagetype `<std_msgs::Float64>` stattfinden. Hierfür wird ein Publisher Namens `pubLeg1Joint1` erzeugt, welcher die berechneten Gelenkwinkel zur Ansteuerung des Motors bzw. der Positionierung des Gelenks an den Topic `/controller_m11/command` veröffentlicht.

Der Publisher `pubLegJoint1` wird in der Funktion `Akrobot::moveLeg(...)` zur Ansteuerung der Gelenke verwendet.

```

int Akrobot::moveLeg(float alpha, float beta, float gamma, int legNum){

    if(CoxaJointLimit){      // [COXA JOINT LIMITS] -- control the joint max and min limits
        if(FemurJointLimit){ // [FEMUR JOINT LIMITS] -- control the joint max and min limits
            if(TibiaJointLimit){ // [TIBIA JOINT LIMITS] -- control the joint max and min limits

                std_msgs::Float64 aux; // [STANDARD MESSAGE TYPE] -- publishing data type
                // [...publish(..)] -- function to publish data

                switch(legNum){
                    case 0: // [LEG 1] -- set angle values and publish

                        aux.data = from_degrees(alpha);
                        pubLeg1Joint1.publish(aux);

                        aux.data = from_degrees(beta);
                        pubLeg1Joint2.publish(aux);

                        aux.data = from_degrees(gamma);
                        pubLeg1Joint3.publish(aux);

                        break;

                    ... \\

                } // int Akrobot::moveLeg(float alpha, float beta, float gamma, int legNum)
            }
        }
    }
}

```

Die Funktion `Akrobat::moveLeg()` hat vier Übergabeparameter. Dies sind die Sollpositionen der drei Gelenkwinkel (α, β, γ) und die Variable `legNum`, welche definiert an welches Roboterbein die drei Gelenkwinkel übergeben werden müssen. Bevor die Werte der Gelenkwinkel an die Gelenke übergeben werden, muss geprüft werden ob diese im Arbeitsbereich liegen. Die maximalen und minimalen Grenzen des Arbeitsbereichs sind in der Header Datei `akrobot_init.hpp` definiert. Für die eigentliche Übergabe der Gelenkwinkel an die Gelenke wird die Funktion `pubLeg1Joint1.publish(aux)` verwendet. Dabei hat die Variable `aux` als Datentyp den Nachrichtentyp `std_msgs::Float64`, da der Topic `/controller_m11/command` diesen benutzt. Diesen Nachrichtentyp stellt das Paket `std_msgs` des ROS Systems zur Verfügung.

Zur Ermittlung der Gelenkwinkel wurde die Funktion `Akrobat::inverseKinematics(int legNum)` implementiert. Diese stellt die im Kapitel 4.2.4 hergeleiteten Gleichungen der inversen Kinematik der Roboterbeine dar. Der Übergabeparameter `legNum` legt fest, für welches der sechs Beine die inverse Kinematik berechnet werden soll.

Die Positionen des Endeffektors im kartesischen Raum werden durch die erzeugten Trajektorien vorgegeben. Diese werden in der Funktion Dreifußgang `Akrobat::tripodGait(trajectoryStruct *tS, int legNum)` für jedes einzelne Roboterbein berechnet. Hierfür werden die Parameter der Trajektorien, im Genaueren die Werte der Amplituden, die Anzahl der Punkte im kartesischen Raum und die entsprechende Reihenfolge der Beine für den Dreifußgang über den Übergabeparameter `trajectoryStruct *tS` an die Funktion `Akrobat::tripodGait` weitergeleitet. Die Ansätze zur Umsetzung der Funktion wurden dem Projekt B.E.T.H¹ entnommen.

Um die kinematischen Operationen des Roboterkörpers ausführen zu können, muss das Zusammenspiel mehrerer Koordinatensysteme hergestellt werden. Dafür werden die im Kapitel 4.2.3 beschriebenen Transformationen notwendig. Um diese Transformationen erzeugen zu können, wurde ein weiteres Paket des ROS Systems angebunden, nämlich das `tf` Paket. Mit Hilfe der implementierten Klassen und deren Funktionen wurde die Funktion `Akrobat::transformCS(string sCS, string tCS, Vector3 rot, Vector3 trans)` entwickelt. Um diese zu nutzen, muss über die Strings `sCS` und `tCS` angegeben werden, zwischen welchen Koordinatensystemen die Transformation ausgeführt werden soll. Die rotatorischen und translatori-

¹<http://www.projectsofdan.com/?tag=phantomx>, Stand: Juli 2014

schen Parameter sind in die Vektoren `rot` und `trans` zusammengefasst. Die Definition für die Transformation erfolgt wie folgt.

```
Transform TCS;
TCS.setOrigin(trans);
Quaternion rotQuat;
rotQuat.setRPY(from_degrees(rot.x()), from_degrees(rot.y()), from_degrees(rot.z()));
TCS.setRotation(rotQuat);
```

Um die Transformation definieren zu können, wird zunächst ein Objekt `TCS` der Klasse `Transform` erzeugt. Im nächsten Schritt werden die Lage und die Ausrichtung mit den Vektoren `rot` und `trans` festgelegt. Die Lage wird mit der Methode `TCS.setOrigin(trans)` definiert. Für die Ausrichtung wird ein Quaternion deklariert und mit der Methode `TCS.setRotation(rotQuat)` dem Transformationsobjekt bekannt gegeben. Die Berechnung der Transformationen intern im ROS System geschieht über die homogenen Transformationsmatrizen mittels der homogenen Koordinaten. Die berechnete Transformation wird als Rückgabewert der Funktion `Akrobat::transformCS(string sCS, string tCS, Vector3 rot, Vector3 trans)` zurückgegeben. So kann, beispielsweise die Transformation eines Trajektoriepunkts im kartesischen Raum des Fußkoordinatensystems in das Beinkoordinatensystem überführt werden. Der Vektor `FCS.leg[legNum].trajectoryPresPos` enthält die aktuellen Koordinaten des Endeffektors und diese werden nach der Transformation dem Vektor `LCS.leg[legNum].footPresPos` zugewiesen.

```
LCS.leg[legNum].footPresPos =
Akrobat::transformCS("FCS", "LCS", Vector3(0,0,0), LCS.leg[legNum].footInitPos) *
FCS.leg[legNum].trajectoryPresPos;
```

akrobat_main.hpp

Der Abschnitt der `akrobat_main.cpp` stellt den eigentlichen Akrobat Node dar. In der Main-Methode wird ein Objekt `akrobat1` der entwickelten Klasse `Akrobat`

erzeugt. Nach der Positionierung der Laufmaschine wird in der Schleife die Funktion `akrobat1.runAkrobat()` ausgeführt. Diese umfasst die in diesem Kapitel beschriebenen Funktionen, die eine Fortbewegung des Laufroboters umsetzen. Die Verarbeitungsgeschwindigkeit bzw. die Frequenz des Akrobat Nodes kann mit der Funktion `ros::Rate r_schleife(30)` manipuliert werden. Der Wert der globalen Variable `ON` wird mittels einer Tastenkombination des Joysticks verändert, so kann die Schleife `while(ros::ok() && ON)` unterbrochen und der Akrobat Node gestoppt werden.

```
//-----MAIN-----//
int main(int argc, char** argv)
{
    ros::init(argc, argv, "akrobot_main");
    Akrobot akrobot1;
    ros::Rate r_schleife(30);
    akrobot1.initAkrobot();

    //WHILE-LOOP
    while(ros::ok() && ON){
        akrobot1.runAkrobot();
        ros::spinOnce();
        r_schleife.sleep();
    } //END WHILE
}//-----END MAIN-----//
```

Kapitel 6

Erprobung der Laufmaschine

Um die Funktionsfähigkeit der entwickelten Steuerungsalgorithmen aufzuzeigen, erfolgen die Experimente mit dem Laufroboter und die Messerfassung bei der Ausführung dieser. Die Ansätze zur Umsetzung der Experimente wurden der Dissertation [12] entnommen.

Für die Experimente wird der entwickelte Node des Pakets Akrobat und alle hierfür notwendigen Nodes für den Kommunikationsablauf mit den Sensoren und Aktoren gestartet. Dies erfolgt mittels der Launch Datei `akrobat.launch` des Pakets Akrobat. Hierfür muss folgender ROS-Befehl ausgeführt werden.

```
roslaunch akrobat akrobat.launch
```

Nachdem alle wichtigen Nodes für das Robotersystem gestartet wurden, können die Experimente ausgeführt werden. Zunächst wird der Aufbau der Topologie von dem Robotersystem in ROS aufgezeigt, welcher die einzelnen Kommunikationen zwischen den Nodes und die Richtungen des Datenaustausches unter den Subscriber und Publisher Nodes darstellt. Dabei wird die Interaktion des Akrobat-Nodes mit den einzelnen Topics `/controller_mxx/command` für jeden Controller des Motors aufgezeigt. Die Abbildung 6.1 stellt die Topologie des Robotersystems dar.

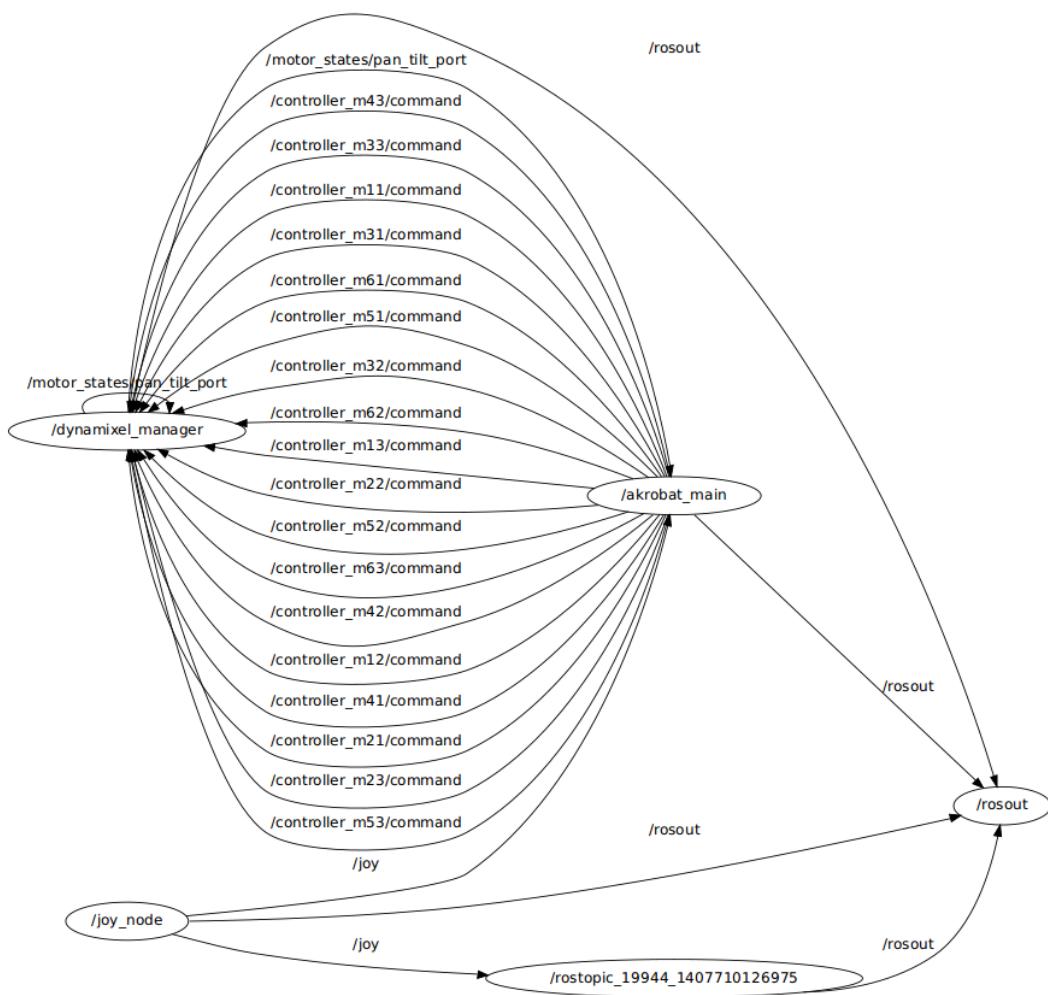


Abbildung 6.1: Interaktion zwischen dem Node Akrobot und den Motoren

Zur Visualisierung des Laufens und der Bewegungen des Roboterkörpers und der Beine kann das interne 3D Visualisierungstool des ROS Systems gestartet werden. Die Abbildung 6.2 stellt die Visualisierung der Laufmaschine dar.

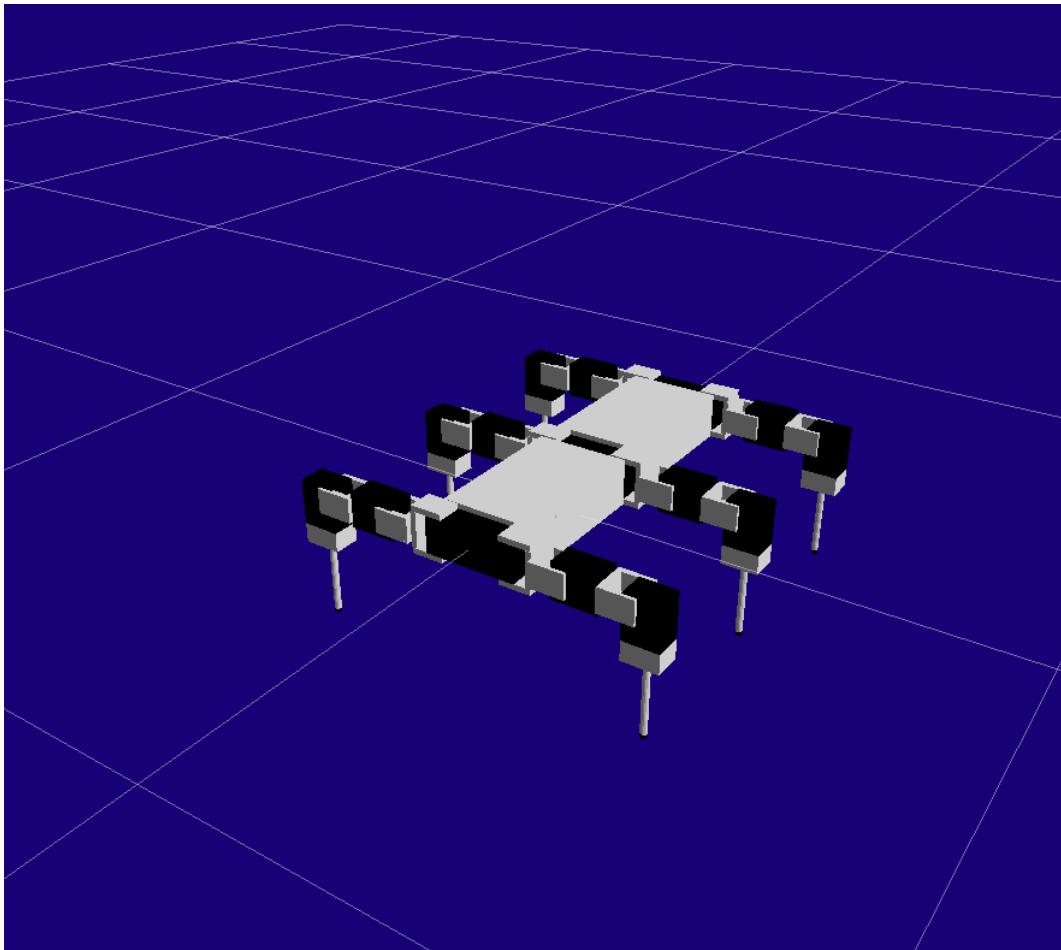


Abbildung 6.2: 3D-Visualisierung der Laufmaschine

6.1 Erprobung des Laufens

Für die Fortbewegung der Laufmaschine wurden in Kapitel 4 die Trajektorien der Beine und die Laufzyklen erläutert, die Relativbeinbewegungen ermöglichen. Um die statische Stabilität zu sichern, muss entsprechend das Laufmuster gewählt werden, bei dem zu jedem Zeitpunkt mindestens drei Füße gleichzeitig den Bodenkontakt haben und ein ausreichend großes Stützpolygon aufspannen. Hierfür wurde der Dreifußgang ausgewählt, da dieser den oben genannten Anforderungen entspricht. Die Abbildung 6.3 zeigt die Stützpolygone beim Dreifußgang (Abb. 6.3a) und (Abb. 6.3b) das Stützpolygon beim Stehen der Laufmaschine mit allen sechs Beinen (Abb. 6.3c).

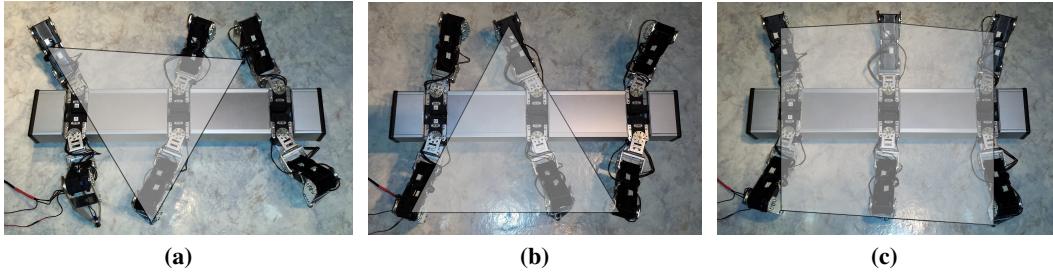


Abbildung 6.3: Schematische Darstellung der Stützpolygone beim Dreifußgang

Die ermittelten und aufgespannten Stützpolygone beim Dreifußgang als Laufmuster bilden eine große Fläche. So kann von einem statisch stabilen Laufmuster aus gegangen werden, bei dem drei der Beine gleichzeitig die Stützphase der Trajektorie ausführen und den Bodenkontakt haben, während die anderen drei sich in der Schwingphase der Trajektorie befinden.

6.2 Erprobung der Körperbewegungen

Zur Erprobung der Körperbewegungen werden mittels Joystick die Rotations- und Translationsparameter für die Körperposition nach der Gleichung (4.6) und (4.7) variiert. Die Abbildung 6.4 veranschaulicht die Beobachtungen bei der Ausführung der Körperbewegungen des Laufroboters.

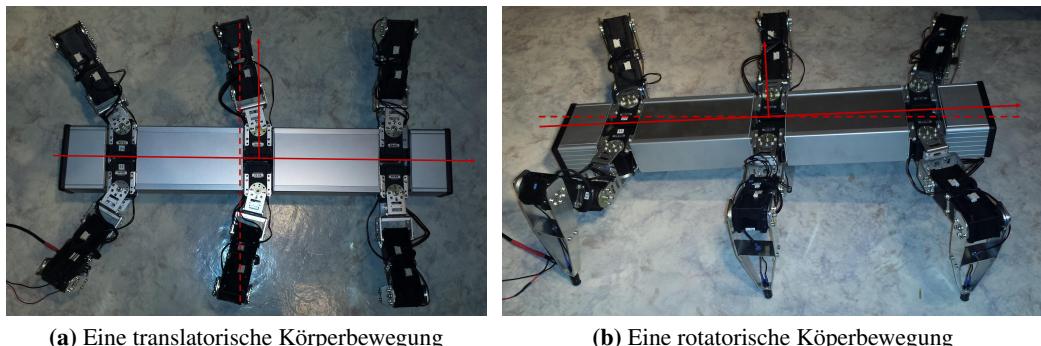


Abbildung 6.4: Erprobung der Körperbewegungen

Die Abbildung 6.4a zeigt die Körperverschiebung entlang der Körperachsen entsprechend der Operation nach Gleichung (4.6). Die Abbildung 6.4b veranschaulicht die Neigung des Roboterkörpers entsprechend der Gleichung (4.7). Der Kör-

per kann mit dem hier demonstrierten Algorithmus in allen sechs Freiheitsgraden ausgerichtet werden.

Kapitel 7

Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde an der Hochschule Mannheim eine Laufmaschine konstruiert. Für die Fortbewegung dieser wurden die Steuerungsalgorithmen konzipiert und implementiert. Die Laufmaschine soll ferner als Experimentierplattform genutzt werden.

Zur Realisierung der Fortbewegung wurden die statisch stabilen Laufmustern, beispielsweise der Dreifußgang genutzt. Dafür musste die kinematische Analyse der Beine durchgeführt und die Koordinatensysteme sowohl am Roboterkörper als auch an den Beinen definiert werden. Die definierten Koordinatensysteme an den Roboterbeinen konnten zur Ermittlung der entsprechenden, geometrischen Parameter genutzt werden. Mit Hilfe der ermittelten DH Parametern und des Denavit-Hartenberg Verfahrens wurden die Transformationsmatrizen der direkten Kinematik hergeleitet. Für die Umsetzung der im kartesischen Raum erzeugten Beintrajektorien sind mitels der analytischen Lösungen die Gleichungen der inversen Kinematik ermittelt worden.

Die Entwicklung der Steuerungsalgorithmen wurde mit dem ROS System umgesetzt. Dafür wurde ein ROS Paket entwickelt, welches das Steuerungssystem des Roboters bildet. Im Umfang dieses Pakets wird der Kommunikationsaufbau mit den Servomotoren und mit dem Joystick mittels der Integration existierender ROS Pakete und deren Funktionalitäten abgewickelt. Dieses Paket wurde in das installierte ROS System auf dem Raspberry Pi integriert, welcher als ein Steuerungsrechner zum Einsatz kommt.

Die Erprobung der Laufmaschine lässt ein positives Ergebnis, soweit dies im Rahmen eines statisch stabilen Laufmusters zu erwarten ist, aufzeigen. Der Laufrobo-

ter kann als eine funktionstüchtige Experimentierplattform für Studierende genutzt werden.

Abkürzungsverzeichnis

BKS Beinkoordinatensystem

FKS Fußkoordinatensystem

FZI Karlsruhe Forschungszentrum Informatik Karlsruhe

GPIO General-purpose input/output

HDMI High Definition Multimedia Interface

I2C Inter-Integrated Circuit

ROB Mannheim Institut für Robotik der Hochschule Mannheim

ROS Robot Operating System

UART Universal Asynchronous Receiver Transmitter

USB Universal Serial Bus

WLAN Wireless Local Area Network

ZKS Zentralkoordinatensystem

Tabellenverzeichnis

2.1	Denavit - Hartenberg Parameter	10
3.1	Mechanische Parameter des Laufroboters	24
4.1	Geometrische Parameter	40
4.2	DH Parameter eines Beines	42

Abbildungsverzeichnis

2.1	Dreifußgang	7
2.2	Wellengang	8
2.3	Kinematische Kette [7]	9
2.4	Zwei Gelenktypen [13]	9
2.5	Das Denavit-Hartenberg Verfahren [12]	10
2.6	Direkte Kinematik [7]	11
2.7	Inverse Kinematik [7]	13
2.8	Lösungen der inversen Kinematik [7]	14
2.9	Oben (v.l.n.r): Turtlebot, PR2, NXT, Roomba Unten (v.l.n.r.): Stanford Junior, CityFlyer, Parrot AR.Drone [14]	16
2.10	ROS Topic [14]	17
2.11	ROS Service [14]	18
2.12	Message-Definition: Sensor/Image [14]	19
2.13	P2P-Topologie [14]	19
2.14	Topic-Verbindungsauflbau über Namensserver [14]	20
2.15	Die Verzeichnisstruktur des Dynamixel_motor-Stacks	21
3.1	Die Laufmaschine	22
3.2	Ein CAD-Modell des Roboters mit vier Gehäusen	25
3.3	Kinematisches Model eines Beines	26
3.4	Arbeitsbereich eines Beines [12]	27
3.5	Das neue Raspberry Pi Modell B+	28
3.6	Dynamixel RX-64 Motoren [19]	30
3.7	USB-RS485-Konverter der Firma FDTI [18]	30
4.1	Die Trajektorie für die Beine	32
4.2	Die Phasen der Trajektorie	33
4.3	Definierte Koordinatensysteme am Roboter	35
4.4	Beziehungen zwischen den Koordinatensystemen am Laufroboter	37
4.5	Kinematik des Laufroboters	39
4.6	Die geometrischen Parameter eines Beines	40
4.7	Die definierten Achsenkoordinatensysteme eines Beines	41
4.8	Bestimmung der inversen Kinematik anhand der geometrischen Beziehungen an einem Bein	43
4.9	Nummerierung der Motoren des Laufroboters	45

5.1	Der Aufbau des Akrobat-Pakets	47
6.1	Interaktion zwischen dem Node Akrobat und den Motoren	59
6.2	3D-Visualisierung der Laufmaschine	60
6.3	Schematische Darstellung der Stützpolygone beim Dreifußgang . .	61
6.4	Erprobung der Körperbewegungen	61

Literaturverzeichnis

- [1] ALBIEZ, J. ; ILG, W. ; LUKSCH, T. ; BERNS, K. ; DILLMANN, R.: Learning a reactive posture control on the four-legged walking machine BISAM. In: *IEEE/RSJ International Intelligent Robots and Systems* Interactive Diagnosis- and Service Syst., Forschungszentrum Informatik Karlsruhe, Germany (Veranst.), 2001, S. 999–1004
- [2] BLUM, M. ; HECK, F.: *Studienarbeit, Projekt Sechsbeiner*. Hochschule Mannheim, Institut für Robotik, Mannheim. 2012
- [3] BOTELLO, O.E.L. ; GARCIA, M.L.B. ; ZAMBRANO, C. ; VELAZQUEZ, A.R.R.: Design of a Hexapod Robot Based on Insects. In: *IEEE Electronics, Robotics and Automotive Mechanics* Universidad Autonoma de Nuevo Leon, San Nicolas de los Garza, Nuevo Leon, Mexico (Veranst.), 2010, S. 347–354
- [4] DENAVIT, J. ; HARTENBERG, R. S.: A kinematic notation for lower-pair mechanisms based on matrices. In: *ASME Journal of Applied Mechanics*, 1955, S. 215 – 221
- [5] DONGPING, L. ; ERBAO, D. ; CHUNSHAN, L. ; MIN, X. ; JIE, Y.: Design and development of a leg-wheel hybrid robot "HyTRo-I". In: *IEEE/RSJ International Intelligent Robots and Systems* Dept. of Precision Machinery and Precision Instrum., Univ. of Sci. and Technol. of China, Hefei, China (Veranst.), 2013, S. 6031–6036
- [6] FEI, L. ; BONSIGNORI, G. ; SCARFOGLIERO, U. ; DAJING, C. ; STEFANINI, C. ; WEITING, L. ; DARIO, P. ; XIN, F.: Jumping mini-robot with bio-inspired legs. In: *IEEE International Robotics and Biomimetics (ROBIO)* State Key Lab. of Fluid Power Transm. and Control, Zhejiang Univ., Hangzhou (Veranst.), 2009, S. 933–938
- [7] FELLMANN, J.: *Entwicklung eines statisch stabilen Laufalgorithmus für einen zweibeinigen laufroboter*. Institut für Robotik der Fakultät für Informatik, Hochschule Mannheim. 2007
- [8] FERRELL, C.: Robust and adaptive locomotion of an autonomous hexapod. In: *IEEE From Perception to Action Conference Artificial Intelligence Lab.*, MIT, Cambridge, MA, USA (Veranst.), 1994, S. 66 – 67

- [9] GOEBEL, R. P.: *ROS By Example, A Do-It-Yourself Guide to the Robot Operating System.* 1. 2012 (ISBN 5-800085-311092)
- [10] GOELLER, M. ; ROENNAU, A. ; GORBUNOV, A. ; HEPPNER, G. ; DILLMANN, R.: Pushing around a robot: Force-based manual control of the six-legged walking robot LAURON. In: *IEEE International Robotics and Biomimetics (ROBIO)* FZI-Res. Center for Inf. Technol., Karlsruhe, Germany (Veranst.), 2011, S. 2647–2652
- [11] HAUN, M.: *Handbuch Robotik: Programmieren und Einsatz intelligenter Roboter.* 2. Springer Vieweg, 2013 (ISBN 978-3-642-39858-2)
- [12] IHME, T.: *Steuerung von sechsbeinigen Laufrobotern unter dem Aspekt technischer Anwendungen*, Otto-von-Guericke-Universität Magdeburg, Dissertation, 2002
- [13] IHME, T.: *Vorlesung Robotik SS2014, Script: Teilsysteme*. Institut für Robotik, Hochschule Mannheim. 2014
- [14] JÖRNS, E.: *ROS - Ein offenes Roboter-Betriebssystem*. IRP, TU Braunschweig. 2011
- [15] JUNG-MIN, Y. ; K., Jong-Hwan: Optimal fault tolerant gait sequence of the hexapod robot with overlapping reachable areas and crab walking. In: *IEEE International Systems Man and Cybernetics* Dept. of Electr. Eng., Korea Adv. Inst. of Sci. and Technol., Seoul, South Korea (Veranst.), 1999, S. 224 – 235
- [16] KAJITA, S. ; TOMIO, Y. ; KOBAYASHI, A.: Dynamic walking control of a biped robot along a potential energy conserving orbit. In: *IEEE International Robotics and Automation*. Robotics Department, Mechanical Engineering Laboratory, Ibaraki, Japan (Veranst.), 1992, S. 431–438
- [17] KARIGIANNIS, J.N. ; REKATSINAS, T.I. ; TZAFESTAS, C.S.: Fuzzy rule based neuro-dynamic programming for mobile robot skill acquisition on the basis of a nested multi-agent architecture. In: *IEEE International Robotics and Biomimetics (ROBIO)* Division of Signals, Control and Robotics, National Technical University of Athens, Greece (Veranst.), 2010, S. 312–319
- [18] LTD, Future Technology Devices I.: *Datasheet: USB to RS485 Serial Converter Cable, Version 1.4, Document No FT 000117.* 2007
- [19] LTD, ROBOTIS C.: *User's Manual: Dynamixel RX-64 , Version 1.10.* – URL http://creativemachines.cornell.edu/sites/default/files/RX-64_Manual.pdf
- [20] MARTINEZ, A. ; FERNANDEZ, E.: *Learning ROS for Robotics Programming.* 1. Packt publishing, 2013 (ISBN 978-1-78216-144-8)
- [21] NACHTIGALL, W. ; WISSE, A.: *Bionik in Beispielen.* 1. Springer Verlag, 2013 (ISBN 978-3-642-34767-2)

- [22] QUIGLEY, M. ; BERGER, E. ; ANDREW, Y. N.: *STAIR: Hardware and Software Architecture*. 2007
- [23] RUFFLER, U.: *Laufplanung basierend auf realitätsnahen Umgebungsdaten für einen sechbeinigen Laufroboter*. Institut für Robotik der Fakultät für Informatik, Hochschule Mannheim. 2006
- [24] SCHAMBUREK, J.U.: *Bewegungssteuerung bei Insekten*. Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe. 2003
- [25] SCHMUCKE, U. ; SCHNEIDER, A. ; IHME, T.: Sechsbeiniger Schreitroboter mit Kraftrückführung - Konzeption und erste Ergebnisse. In: *1. Brandenburger Workshop Mechatronik, Brandenburg* Fraunhofer-Institut für Fabrikbetrieb und -automatisierung (IFF) (Veranst.), 1994, S. 45 –53.
- [26] STELZER, M. ; STRYK, O. von: *Laufbewegungen bei Roboter, Tier und Mensch: Analyse, Modellierung, Simulation und Optimierung*. Fachgebiet Simulation und Systemoptimierung, Technische Universität Darmstadt. 2005
- [27] WLOKA, D. W.: *Robotersysteme: Technische Grundlagen*. 1. Springer-Verlag, 1992 (ISBN 3-540-54739-8)
- [28] WOERING, R.: *Simulating the first steps of walking hexapod robot*. Department Mechanical Engineering, Technische Universiteit Eindhoven. 2011
- [29] YONEDA, K. ; SUZUKI, K. ; KANAYAMA, Y.: Gait Planning for Versatile Motion of a Six Legged Robot. In: *IEEE International Robotics and Automation* Dept. of Mechano-Aerosp. Eng., Tokyo Inst. of Technol., Japan (Veranst.), 1994, S. 1338–1343
- [30] ZAVGORODNIY, Y.: *Konstruktion und Steuerung von Schreitrobotern mit ballistischem Laufverhalten*, Otto-von-Guericke-Universität Magdeburg, Dissertation, 2009