Hochschule für Technik, Wirtschaft und Kultur Leipzig
Fakultät Informatik, Mathematik und Naturwissenschaften

Prüfungsprojekt für das Wahlpflichtmodul Robotik

Dokumentation des CM-BOT

B.Sc. Matthias Rick
B.Sc. Christof Pieloth

28. Dezember 2010

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Als Praxisaufgabe für das Wahlpflichtmodul Robotik an der HTWK Leipzig ist ein Hexapod zu entwickeln.

Für jedes Beinpaar dieses sechsbeinigen Roboters ist ein eigener Controller zu verwenden. Die Controller sollen über eine geeignete Kommunikationsstruktur Daten austauschen können. Außerdem ist die direkte und inverse Kinematik zu lösen. Unter Nutzung der zuvor erarbeiteten Erkenntnisse ist abschließend ein Laufalgorithmus zu implementieren.

1.2 Vorbetrachtung

1.2.1 Bioloid Premium Robot Kit

Ausgangspunkt für das Praxisprojekt des Moduls Robotik ist das Premium Robot Kit von Bioloid. Dieser beinhaltet 18 AX-12 Servomotoren der Marke Dynamixel, ein Controllermodul, ein Zigbee-Fernsteuermodul und noch einige andere Peripherie. Gedacht ist dieses Robot Kit für humanoide Roboter, die mittels der mitgelieferten Software programmiert werden können. Diese ermöglicht es einfache Bewegungen mittels statischer Abläufe zu modellieren.

Da im Rahmen des Projektes ein Hexapod entwickelt werden soll, der sich mittels der Lösung des inversen kinematischen Problems bewegen soll, werden im wesentlichen nur die *Dynamixel AX-12*-Servomotoren genutzt. Das bedeutet, dass sämtliche Mechanik und Elektronik neu entwickelt werden muss.

1.2.2 Atmel ATXmega

- //TODO
- Controller

1.2.3 Projekt auf BerliOS

BerliOS ist ein Webportal der Fraunhofer-Gesellschaft zur Verwaltung quelloffener Softwareprojekte. Das Portal stellt unentgeldlich viele nützliche Funktionen für Entwickler bereit:

- Versionsverwaltung: CVS, SVN, GIT, Mercurial
- FTP & Webspace
- Wiki
- u.m.

Für diese Praxisaufgabe wurde auf BerliOS das Projekt „CM-Bot" erstellt, welches unter `http://developer.berlios.de/projects/cm-bot/` erreichbar ist. Der Projektname leitet sich aus den Anfangsbuchstaben der beiden Entwickler Christof Pieloth und Matthias Rick her. Auf BerliOS ist der gesamte Quellcode in der Versionsverwaltung SVN verfügbar. Des Weiteren ist eine Wiki eingerichtet, in dem Spezifikationen der verwendeten Hardware, Handbücher, Dokumentationen sowie die zur Entwicklung benötigten Programme aufgelistet sind.

2 Hardware & Mechanik

2.1 Chassis

- Höhe: 60 mm
- Länge: 500 mm
- Breite: 250 mm
- Vorderes/Hinteres Bein: 41 mm (von außen)
- Mittleres Bein: 250 mm (von außen)

2.2 Netzteil

Es werden für die verschiedene Komponenten drei unterschiedliche Versorgungsspannungen benötigt:

- 12 V
  - VCC der Servomotoren
- 5 V
  - Steuerleitung für die Servomotoren
  - Levelshifting-ICs von 3,3 V auf 5 V
- 3,3 V
  - Mikrocontrollermodule
  - Levelshifting-ICs von 5 V auf 3,3 V
  - Tri-State-ICs für Kommunikation

2.3 Eagle-Schaltungen

2.3.1 Dynamixel-Anbindung

Abbildung 2.1 zeigt die Anbindung der Dynamixel, inklusive der Spannungsversorung und dem Levelshifting.

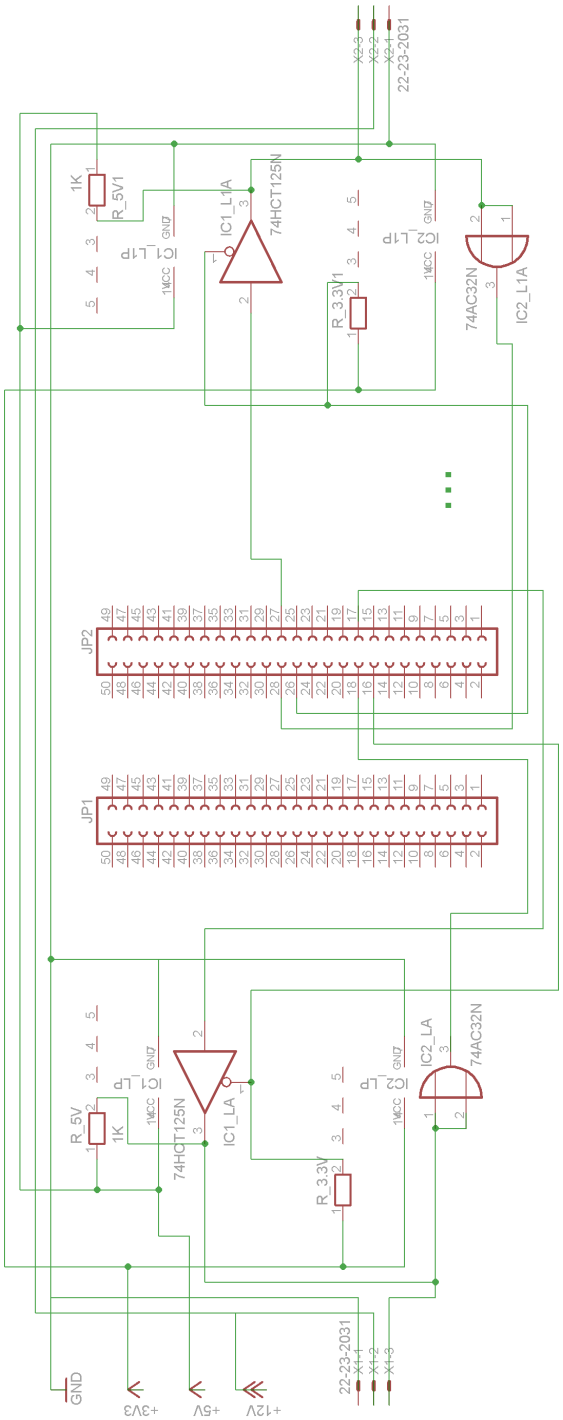

Abbildung 2.1: Dynamixel-Anbindung

2.3.2 Kommunikation - Bus

Abbildung 2.2 zeigt die angedachte Schaltung, die als Master-Slave-Bussystem angedacht war. Aufgrund eines Hardwaredefekts des Tristate-Treibers wurde aber zunächst die Schaltung wie unter 2.3.3 beschrieben, realisiert.

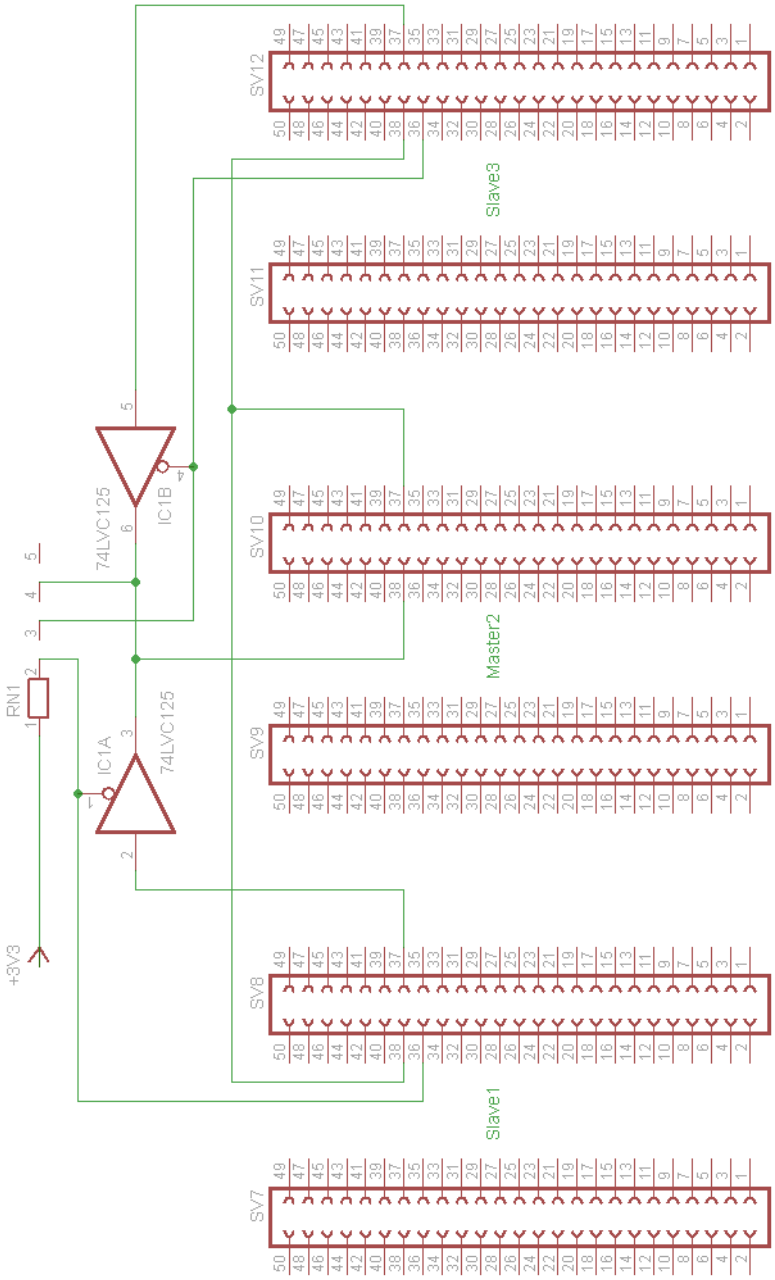

Abbildung 2.2: Kommunikation - Bus

2.3.3 Kommunikation - Alternativschaltung

Abbildung 2.3 zeigt die konkrete Umsetzung, die aus einem Hardwaredefekt des Tristate-Treibers resultiert.

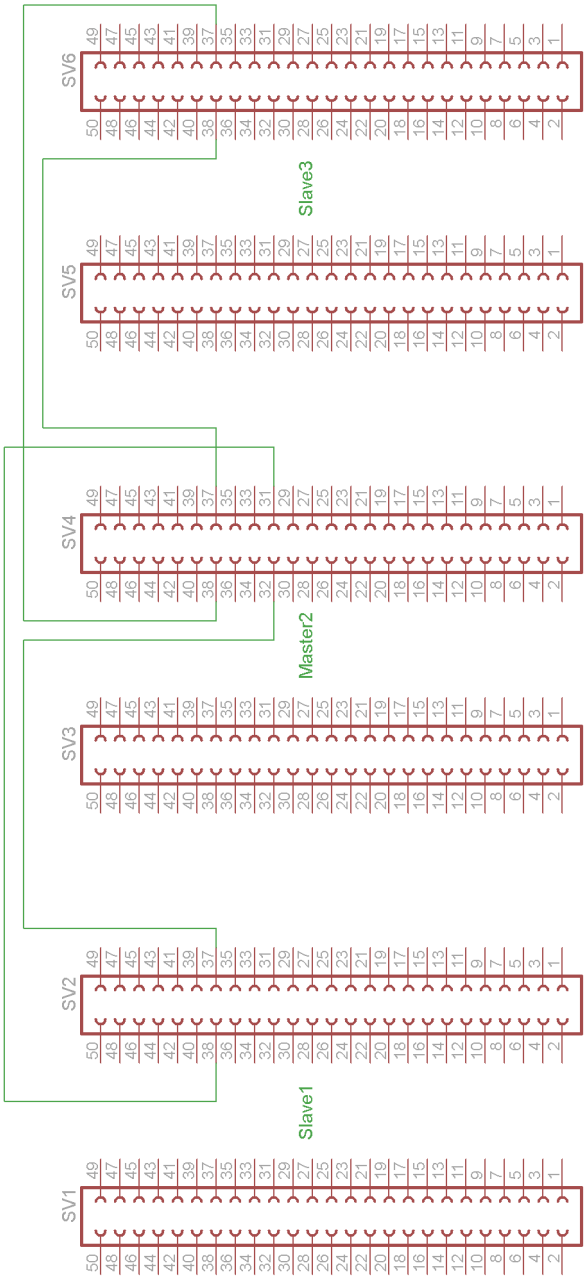

Abbildung 2.3: Kommunikation

3 Kinematik

Um koordinierte Bewegungen des CM-Bot zu ermöglichen, wird den Bewegungen ein mathematisches Modell zugrunde gelegt. Dabei spielen zwei Begriffe eine wichtige Rolle. Zum einen sind das die Roboterkoordinaten, die eine Repräsentation der einzelnen Gelenkwinkel des Roboters darstellen, und zum anderen die karthesischen Weltkoordinaten. Während die Weltkoordinaten den Arbeitsraum des Roboters definieren, bestimmen die Roboterkoordinaten den Konfigurationsraum. Mit Hilfe der *Direkten* und *Inversen Kinematik* können diese beiden Koordinatendarstellungen ineinander überführt werden.

3.1 Direkte Kinematik

Die *Direkte Kinematik* steht für die Umrechnung von Roboterkoordinaten in Weltkoordinaten. Dazu wird hierbei die Koordinatentransformation nach Denavit-Hartenberg angewendet.

3.1.1 Denavit-Hartenberg-Parameter

Aus den physischen Gegebenheiten der einzelnen Beinglieder des Roboters ergeben sich die Paramter für die Koordinatentransformation nach Denavit-Hartenberg die in Tabelle 3.1 eingetragen sind.

	G_1	G_2	G_3
l	-14	0	0
a	50	85	55
α	-90	0	0
ϑ	v	v	v

Tabelle 3.1: Denavit-Hartenberg-Parameter

3.1.2 Berechnung der Denavit-Hartenberg-Matrix

Aus den Denavit-Hartenberg-Parametern ergeben sich die folgenden Matrizen:

$$D_{01} = \begin{pmatrix} \cos(\vartheta_1) & 0 & -\sin(\vartheta_1) & 50\cos(\vartheta_1) \\ \sin(\vartheta_1) & 0 & \cos(\vartheta_1) & 50\sin(\vartheta_1) \\ 0 & -1 & 0 & -14 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D_{12} = \begin{pmatrix} \cos(\vartheta_2) & -\sin(\vartheta_2) & 0 & 85\cos(\vartheta_2) \\ \sin(\vartheta_2) & \cos(\vartheta_2) & 0 & 85\sin(\vartheta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$D_{23} = \begin{pmatrix} \cos(\vartheta_3) & -\sin(\vartheta_3) & 0 & 55\cos(\vartheta_3) \\ \sin(\vartheta_3) & \cos(\vartheta_3) & 0 & 55\sin(\vartheta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.1.2.1 Die komplette Denavit-Hartenberg-Matrix des CM-Bot

Nach ausmultiplizieren von $D_{01} \cdot D_{12} \cdot D_{23}$ erhält man:

$$D_{03} = \begin{pmatrix} c(\vartheta_1)\ c(\vartheta_2)\ c(\vartheta_3) - c(\vartheta_1)\ s(\vartheta_2)\ s(\vartheta_3) & -c(\vartheta_1)\ c(\vartheta_2)\ s(\vartheta_3) - c(\vartheta_1)\ c(\vartheta_3)\ s(\vartheta_2) & -s(\vartheta_1) & 50\,c(\vartheta_1) + 85\,c(\vartheta_1)\ c(\vartheta_2) - 55\,c(\vartheta_1)\ s(\vartheta_2)\ s(\vartheta_3) + 55\,c(\vartheta_1)\ c(\vartheta_2)\ c(\vartheta_3) \\ c(\vartheta_2)\ c(\vartheta_3)\ s(\vartheta_1) - s(\vartheta_1)\ s(\vartheta_2)\ s(\vartheta_3) & -c(\vartheta_2)\ s(\vartheta_1)\ s(\vartheta_3) - c(\vartheta_3)\ s(\vartheta_1)\ s(\vartheta_2) & c(\vartheta_1) & 50\,s(\vartheta_1) + 85\,c(\vartheta_2)\ s(\vartheta_1) - 55\,s(\vartheta_1)\ s(\vartheta_2)\ s(\vartheta_3) + 55\,c(\vartheta_2)\ c(\vartheta_3)\ s(\vartheta_1) \\ -c(\vartheta_2)\ s(\vartheta_3) - c(\vartheta_3)\ s(\vartheta_2) & s(\vartheta_2)\ s(\vartheta_3) - c(\vartheta_2)\ c(\vartheta_3) & 0 & -85\,s(\vartheta_2) - 55\,c(\vartheta_2)\ s(\vartheta_3) - 55\,c(\vartheta_3)\ s(\vartheta_2) - 14 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3.1.2.2 Einzelne Terme der Matrix

Aus den einzelnen Terme der D_{03}- Matrix ergibt sich das folgende Gleichungssystem:

$x_{ex} = f_{11} = \cos(\vartheta_1)\ \cos(\vartheta_2)\ \cos(\vartheta_3) - \cos(\vartheta_1)\ \sin(\vartheta_2)\ \sin(\vartheta_3)$

$y_{ex} = f_{12} = -\cos(\vartheta_1)\ \cos(\vartheta_2)\ \sin(\vartheta_3) - \cos(\vartheta_1)\ \cos(\vartheta_3)\ \sin(\vartheta_2)$

$z_{ex} = f_{13} = -\sin(\vartheta_1)$

$p_x = f_{13} = 50\ \cos(\vartheta_1) + 85\ \cos(\vartheta_1)\ \cos(\vartheta_2) - 55\ \cos(\vartheta_1)\ \sin(\vartheta_2)\ \sin(\vartheta_3) + 55\ \cos(\vartheta_1)\ \cos(\vartheta_2)\ \cos(\vartheta_3)$

$x_{ey} = f_{21} = \cos(\vartheta_2)\ \cos(\vartheta_3)\ \sin(\vartheta_1) - \sin(\vartheta_1)\ \sin(\vartheta_2)\ \sin(\vartheta_3)$

$y_{ey} = f_{22} = -\cos(\vartheta_2)\ \sin(\vartheta_1)\ \sin(\vartheta_3) - \cos(\vartheta_3)\ \sin(\vartheta_1)\ \sin(\vartheta_2)$

$z_{ey} = f_{23} = \cos(\vartheta_1)$

$p_y = f_{23} = 50\ \sin(\vartheta_1) + 85\ \cos(\vartheta_2)\ \sin(\vartheta_1) - 55\ \sin(\vartheta_1)\ \sin(\vartheta_2)\ \sin(\vartheta_3) + 55\ \cos(\vartheta_2)\ \cos(\vartheta_3)\ \sin(\vartheta_1)$

$x_{ez} = f_{31} = -\cos(\vartheta_2)\ \sin(\vartheta_3) - \cos(\vartheta_3)\ \sin(\vartheta_2)$

$y_{ez} = f_{32} = \sin(\vartheta_2)\ \sin(\vartheta_3) - \cos(\vartheta_2)\ \cos(\vartheta_3)$

$z_{ez} = f_{33} = 0$

$p_z = f_{33} = -85\ \sin(\vartheta_2) - 55\ \cos(\vartheta_2)\ \sin(\vartheta_3) - 55\ \cos(\vartheta_3)\ \sin(\vartheta_2) - 14$

3.2 Inverse Kinematik

Die *Inverse Kinematik* ist das Gegenstück zur *Direkten Kinematik*. Das bedeutet, dass sie zur Umrechnung von Weltkoordinaten in Roboterkoordinaten dient. Dafür gibt es mehrere Lösungsverfahren. Beispielsweise könnte man das unter 3.1.2.2 notierte Gleichungssystem verwenden. Da es sich dabei um ein Nichtlineares Gleichungssystem mit drei Unbekannten handelt wird im vorliegenden Projekt die Lösung durch ein geometrisches Verfahren berechnet.

Durch Abbildung des 3-dimensionalen Koordinatensystems auf zwei 2-dimensionale Koordinatensyteme können die Gelenkwinkel mit Hilfe der Trigonometrie berechnet werden.

In den folgenden Formeln und Erläuterungen werden diese Bezeichnungen verwendet:

- $H \triangleq$ Hüftgelenk, $\Theta_1 \triangleq$ Winkel des Hüftgelenks
- $K \triangleq$ Kniegelenk, $\Theta_2 \triangleq$ Winkel des Kniegelenks
- $F \triangleq$ Fußgelenk, $\Theta_3 \triangleq$ Winkel des Fußgelenks
- $T \triangleq$ Endpunkt
- $HK \triangleq$ Abstand bzw. Strecke Hüft- zu Kniegelenk
- $KF \triangleq$ Abstand bzw. Strecke Knie- zu Fußgelenk
- $FT \triangleq$ Abstand bzw. Strecke Fußgelenk zu Endpunkt
- $P_1 \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \triangleq$ anzufahrender Punkt

Schritt 1

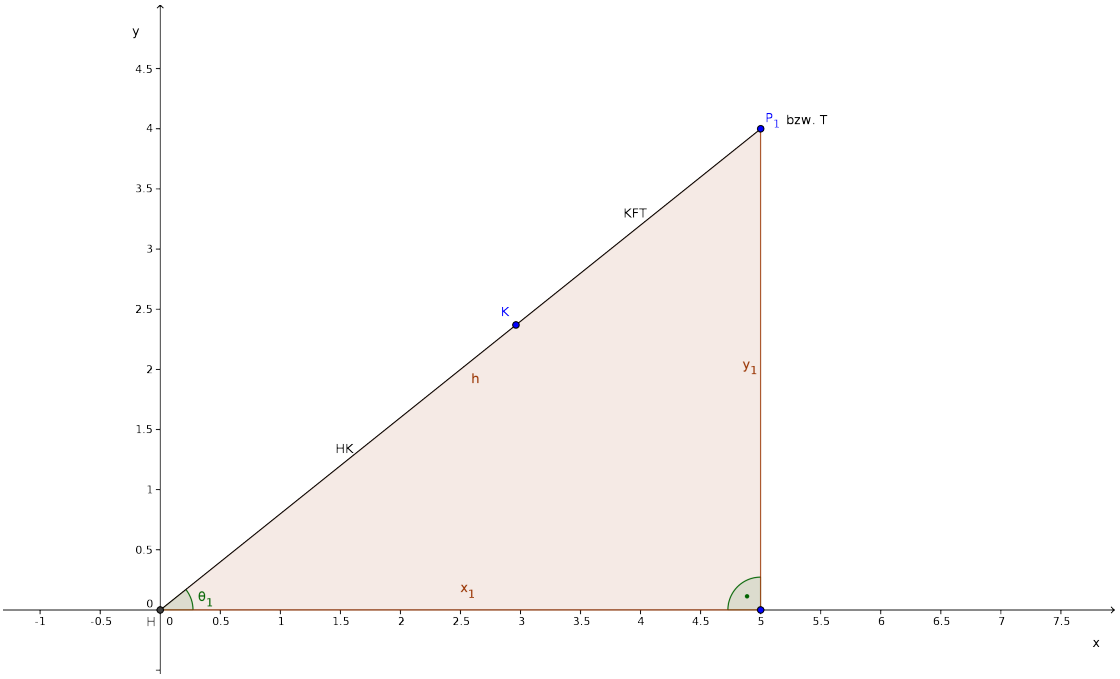

Abbildung 3.1: Schritt 1: x-y-Ebene

In Abbildung 3.1 ist der erste Schritt der Berechnung dargestellt. Hierbei wird das Bein in der x-y-Ebene ausgerichet, sodass eine gedachte Gerade, welche durch alle Achsen

eines Beines verläuft, den anzufahrenden Punkt in dieser Ebene schneidet. Dafür muss der Winkel für das Hüftgelenk berechnet werden.

Die Strecken x_1, y_1 und HK sind bekannt. Somit können der Winkel für das Hüftgelenk Θ_1 und die Strecke HT, oder kurz h, berechnet werden:

- $\Theta_1 = \arctan\left(\frac{y_1}{x_1}\right)$
- $h = \sqrt{{x_1}^2 + {y_1}^2}$

Schritt 2

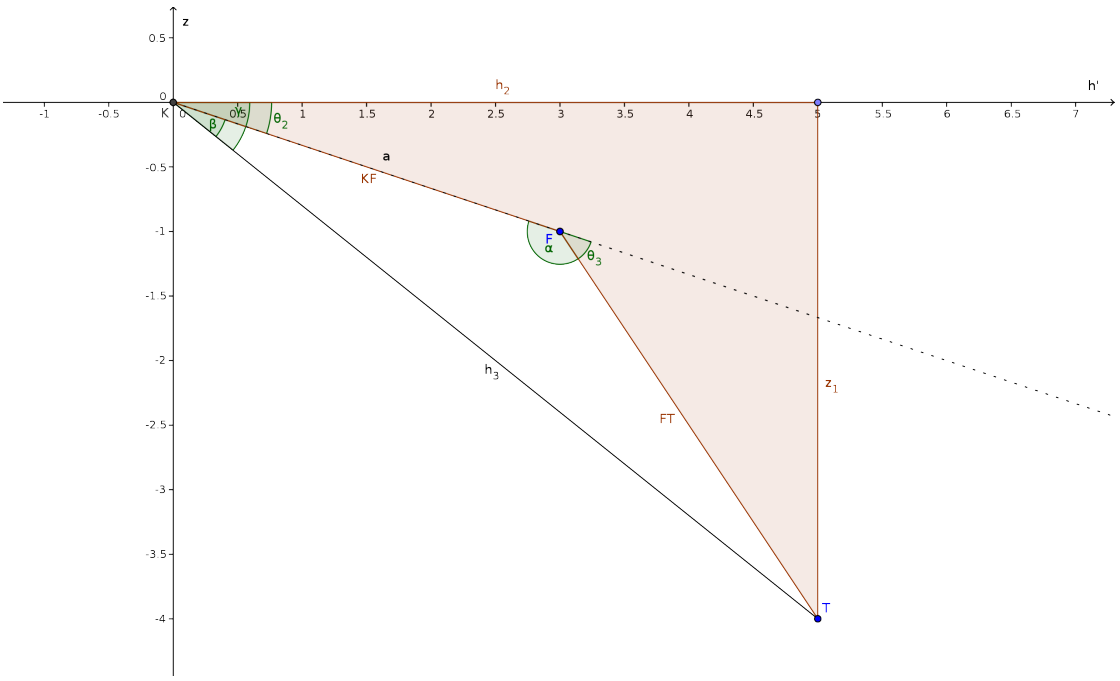

Abbildung 3.2: Schritt 2: h-z-Ebene

In Abbildung 3.2 ist der zweite Schritt der Berechnung dargestellt. Hierbei bildet die gedachte Gerade, aus dem Schritt 1, eine neue Koordinatenachse h'. Das Kniegelenk ist der Nullpunkt auf dieser Achse. Die zweite Achse in dem gedachten Koordinatensystem bildet die z-Achse. Um den Punkt P_1 anfahren zu können, muss mit Hilfe des Knie- und des Fußgelenks noch die „Entfernung" auf der h'-Achse, sowie die „Höhe" auf der z-Achse eingestellt werden. Da in dieser Ebene zwei Freiheitsgrade vorhanden sind, kann der Punkt P_1 auf zwei Wegen angefahren werden. In dieser Berechnung wird ein Weg fest vorgegeben.

Im zweiten Schritt sind die Strecken KF, FT und z_1 bekannt. Um die „Entfernung" auf der h'-Achse anfahren zu können, muss zunächst h_2 berechnet werden. Anschließend können die Winkel Θ_2 für das Kniegelenk und Θ_3 für das Fußgelenk berechnet werden:

- $h_2 = h - HK$
- $h_3 = \sqrt{{h_2}^2 + {z_1}^2}$
- $\alpha = \arccos\left(\frac{-({h_3}^2)+FT^2+KF^2}{2*FT*KF}\right)$
- $\beta = \arcsin\left(\frac{FT/h_3}{\sin\alpha}\right)$
- $\gamma = \arcsin\left(\frac{|z_1|}{h_3}\right)$

- $\Theta_2 = \gamma - \beta$
- $\Theta_3 = \pi - \alpha$

4 Programmierung

In diesem Kapitel wird die Programmierung des Roboters beschrieben. Zu Beginn wird kurz die erstellte API vorgestellt. Anschließend wird die Umsetzung der Laufalgorithmen erläutert.

4.1 API

Eine genaue Beschreibung der Dateien und deren Funktionen kann aus den Sourcecode mit Hilfe des Tools Doxygen[1] erstellt werden. Der Sourcecode sowie die aktuelle API im HTML-Format ist im BerliOS-Projekt[2] verfügbar.

communication.h Methoden zur Kommunikation der CPUs

datatypes.h Abstraktion der elementaren Datentypen und Definition von benötigten Strukturen

dynamixel.h Methoden und Protokoll zur Ansteurung der Dynamixel AX-12

kinematics.h Lösungsmethoden für inverse sowie direkte Kinematik

movement.h Allgemeine Methoden für den Ablauf eines Laufalgorithmus

remote.h Methoden für die Nutzung des Gamepads

usart_driver.h Erweiterter Beispieltreiber von Atmel für die USART

utils.h Hilfsmethoden zur Ausgabe und Umrechnung

xmega.h Spezifische Methoden zur Ansteuerung und Initialisierung der Komponenten des Mikrocontrollers ATXmega

4.2 4-Punkte-Lauf

Der 4-Punkte-Lauf ist ein sehr einfacher Laufalgorithmus, der den Roboter ein Geradeauslaufen durch das Anfahren von vier verschiedenen Koordinaten ermöglicht. Bei einem Hexapod müssen sich zu jedem Zeitpunkt mindestens drei Beine und mindestens eins auf jeder Seite auf dem Boden befinden.

Jedes der sechs Beine benötigt entweder vier Roboterkoorinaten oder vier Weltkoordinaten, welche koordiniert angefahren werden müssen. In Abbildung 4.1 ist diese Bewegung verdeutlicht. Alle Beine mit einem ausgemalten schwarzen Kreis befinden sich auf dem Boden und alle anderen haben keinen Bodenkontakt.

In der konkreten Implementierung, `movement4Points.c`, wird mit Weltkoordinaten gearbeitet. Der Nullpunkt des Weltkoordinatensystems ist die Mitte des Chassis und ein Punkt dieses Systems wird im folgendem als globaler Punkt bezeichnet. Jedes Bein besitzt sein eigenes Roboterkoordinatensystem, dessen Punkte als lokale Punkte bezeichnet

[1] `http://www.doxygen.org`
[2] `http://developer.berlios.de/projects/cm-bot/`

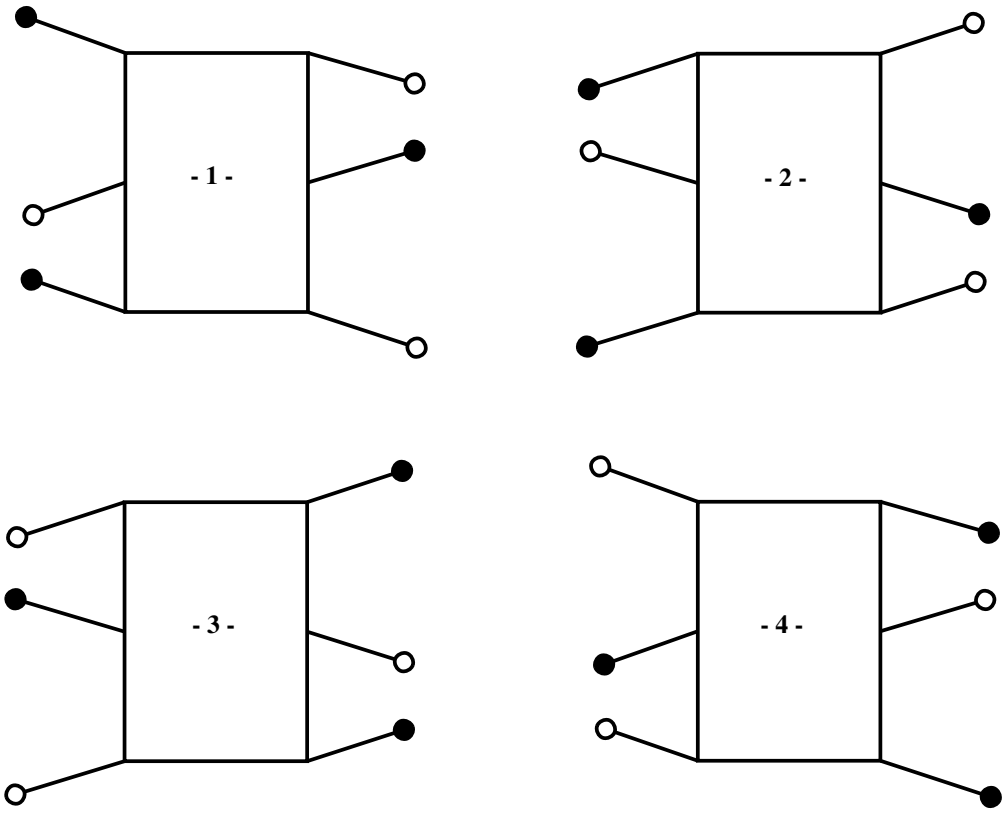

Abbildung 4.1: Darstellung einer Laufbewegung in vier aufeinander folgenden Schritten.

werden. Aus diesen lokale Punkten können die Roboterkoordinaten, dass heisst die anzufahrenden Gelenkwinkel für dieses Bein, errechnet werden.

Theoretisch könnte nur mit vier Roboterkoordianten geareitet werden, da für jedes Bein die selben Roboterkooridnaten genutzt werden könnten. In der praktischen Umsetzung werden jedoch globale Punkte genutzt, um die Transformation dieser in lokale Punkte und die nötige Berechnung der Roboterkoordinaten zu demonstrieren.

Damit nicht für jedes Bein vier Weltkoordinaten ermittelt werden müssen, wird das rechte mittlere Bein als Basis genutzt. Aus den globalen Punkten für dieses Bein können durch Addition bzw. Subtraktion der bekannten Abstände in der x-y-Ebene der Beine zueinander die globalen Punkte für jedes weitere Bein errechnet werden. Diese globalen Punkte werden durch einen CPU, den Master, in einer bestimmten Reihenfolge an die anderen CPUs, die Slaves, gesendet. Diese Punkte müssen nun zuerst von allen CPUs in lokale Punkte für das bestimmte Bein transformiert werden, um anschließend aus diesen die Roboterkoordinaten zu berechnen und die geforderte Position anfahren zu können.

Der Master steuert hierbei die gesamte Bewegung, indem er jedem Beinpaar die korrekten Punkte in einer festen Reihenfolge zuteilt. Die Slaves haben in dieser Implementierung keine Kenntnis über den verwendeten Laufalgorithmus, da sie nur die empfangenen Befehle des Masters ausführen. Somit kann auf allen Slaves das selbe Programm verwendet werden und durch eine Umprogrammierung des Masters, kann der Laufalgorithmus geändert werden.

4.3 Evolutionäres Laufen

Im Rahmen des Wahlpflichtmoduls Evolutionäre Algorithmen entstand noch ein weiterer Algorithmus. Dieser bekam die Bezeichnung Evolutionäres Laufen. Dieser basiert im wesentlichen auf einer Startpunktsuche für Bewegungen bei vorgegebenem Vektor im Raum. Der Startpunkt ist dabei der Punkt, wo ein Bein des Roboters auf den Boden aufsetzt und die Bewegung beginnt. Diese Bewegung soll in Richtung des Vektors liegen, innerhalb des Arbeitsraumes liegen und eine maximale Länge haben. Als Laufmuster wird dabei der Tripod-Gait zugrunde gelegt, der vom Prinzip her, wie auch der 4-Punkt-Laufalgorithmus (siehe 4.2), ein Bewegung so ausführt, dass dabei drei Beine in der Luft sind und die anderen drei auf dem Boden.

Der Evolutionäre Algorithmus liefert für einen vorgegebenen Vektor einen Startpunkt, sowie den maximal anfahrbaren Punkt, der vom Startpunkt ausgehend in Vektorrichtung liegt. Dies muss jeweils für ein Bein auf jeder Seite des Roboters ausgeführt werden. Realisiert wird dies, in dem der Evolutionäre Algorithmus einmal mit dem normalen Vektor und einmal mit dem invertierten Vektor ausgeführt wird. Da sich daraus normalerweise zwei unterschiedliche Längen ergeben und zwischen den einzelnen Beinen Abhängigkeiten bestehen, wird die kürzere Distanz als die zurückzulegende Distanz gewählt. Daraufhin werden noch die Mittelpunkte zwischen den Start- und Endpunkten berechnet, damit beliebige Bewegungen möglich sind.

Anschließend wird dann ein Schritt nach dem folgenden Algorithmus ausgeführt:

1. Inaktive Beine in die Luft
2. Inaktive Beine fahren in der Luft Startpunkt an
3. Alle Beine auf den Boden setzen
4. Beine wechseln (aktive Beine werden inaktiv und inaktive Beine werden aktiv)
5. Inaktive Beine in die Luft
6. Aktive Beine führen Bewegung aus
7. Alle Beine wieder auf den Boden

Dieser Algorithmus ermöglicht es dem CM-Bot Bewegungen in beliebige Richtungen auszuführen.

Nähere Informationen zum Evolutionären Algorithmus können unter `http://www.imn.htwk-leipzig.de/~mrick1` eingesehen werden.