

# Aufbau und Regelung eines Ballbots

Florian Müller

Markus Lamprecht

Michael Suffel

Projektseminar – 16. Februar 2018

Betreuer: Dr.-Ing. Eric Lenz



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

REGELUNGSTECHNIK *rtm*  
UND MECHATRONIK



---

## Aufgabenstellung

---

Für schriftliche Arbeiten (Pro-/Projektseminar, Studien-, Bachelor-, Master-, Diplomarbeiten, etc.) soll Studierenden ein L<sup>A</sup>T<sub>E</sub>X-Dokument zur Verfügung gestellt werden, das die Vorgaben aus den *Richtlinien zur Anfertigung von Studien- und Diplomarbeiten* [?] umsetzt. Die Dokumentation soll die Funktionen des Dokumentes beschreiben und Hinweise zu ihrer Anwendung geben.

Grundlage ist die tudreport-Klasse. Die damit erstellten Arbeiten müssen sowohl zum Ausdrucken geeignet sein als auch für die Bildschirmdarstellung und die elektronische Archivierung als PDF-Datei.

Beginn: 16. Oktober 2017

Ende: 16. Februar 2018

Seminar: 16. Februar 2018

---

Prof. Dr.-Ing. Ulrich Konigorski

---

Dr.-Ing. Eric Lenz

Technische Universität Darmstadt  
Institut für Automatisierungstechnik und Mechatronik  
Fachgebiet Regelungstechnik und Mechatronik  
Prof. Dr.-Ing. Ulrich Konigorski

Landgraf-Georg-Straße 4  
64283 Darmstadt  
Telefon 06151/16-4167  
[www.rtm.tu-darmstadt.de](http://www.rtm.tu-darmstadt.de)





---

## **Erklärung**

---

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 16. Februar 2018

---

Florian MüllerMarkus LamprechtMichael Suffel

---

## Kurzfassung

---

Das  $\text{\LaTeX}$ -Dokument `sada_tudreport` ist eine Vorlage für schriftliche Arbeiten (Proseminar-, Projektseminar-, Studien-, Bachelor-, Master- und Diplomarbeiten, etc.) am Institut für Automatisierungstechnik der TU Darmstadt. Das Layout ist an die *Richtlinien zur Anfertigung von Studien- und Diplomarbeiten* [?] angepasst und durch Modifikation der Klasse `tudreport` realisiert, so dass in der Arbeit die gewohnten  $\text{\LaTeX}$ -Befehle benutzt werden können. Die vorliegende Anleitung beschreibt die Klasse und gibt grundlegende Hinweise zum Verfassen wissenschaftlicher Arbeiten. Sie ist außerdem ein Beispiel für den Aufbau einer Studien-, Bachelor-, Master- bzw. Diplomarbeit.

**Schlüsselwörter:** Studienarbeit, Bachelorarbeit, Masterarbeit, Diplomarbeit, Vorlage,  $\text{\LaTeX}$ -Klasse

---

## Abstract

---

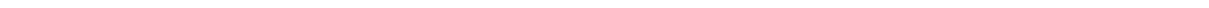
The  $\text{\LaTeX}$  document `sada_tudreport` provides a template for student's research reports and diploma theses ("Proseminar-, Projektseminar-, Studien-, Bachelor-, Master- und Diplomarbeiten") at the Institute of Automatic Control, Technische Universität Darmstadt. The layout is adapted to the "*Richtlinien zur Anfertigung von Studien- und Diplomarbeiten*" [?] and is implemented by modification of the standard `tudreport` class, so that common  $\text{\LaTeX}$  commands can be used in the text. This manual describes the class and dwells on general considerations on how to write scientific reports. Additionally, it is an example for the structure of a thesis.

**Keywords:** Research reports, diploma theses, template,  $\text{\LaTeX}$  class

---

# Inhaltsverzeichnis

<b>Symbole und Abkürzungen</b>	<b>vii</b>
<b>1. Einleitung</b>	<b>1</b>
<b>2. Modellbildung und Regelung</b>	<b>3</b>
2.1. Model . . . . .	3
2.2. Energien . . . . .	3
2.3. Bewegungsgleichungen . . . . .	5
2.4. Zustandsraumdarstellung . . . . .	5
2.5. Reglerentwurf . . . . .	7
<b>3. Konstruktion</b>	<b>9</b>
<b>4. Simulation</b>	<b>11</b>
4.1. 3D Simulatoren . . . . .	11
4.2. Simulation von omnidirektionalen Rädern . . . . .	11
4.3. Simulations Aufbau . . . . .	14
4.3.1. Kommunikation zwischen ROS und Gazebo . . . . .	14
4.3.2. Plugins der Simulation . . . . .	15
4.3.3. Simulierte Dynamik-Eigenschaften . . . . .	17
4.3.4. Visualisierungsprogramme . . . . .	17
<b>5. Auswertung</b>	<b>19</b>
<b>A. Parameterliste</b>	<b>21</b>
<b>Literaturverzeichnis</b>	<b>23</b>





---

# Symbole und Abkürzungen

## Lateinische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
$I$	Strom	A
$R$	Widerstand	$\Omega$
$U$	Spannung	V

## Griechische Symbole und Formelzeichen

Symbol	Beschreibung	Einheit
$\Psi$	Datenmatrix	
$\sigma$	Standardabweichung	
$\omega$	Kreisfrequenz	$s^{-1}$

## Abkürzungen

Kürzel	vollständige Bezeichnung
Dgl.	Differentialgleichung
LS	Kleinste Quadrate ( <i>Least Squares</i> )
PRBS	Pseudo-Rausch-Binär-Signal ( <i>Pseudo Random Binary Signal</i> )
ZVF	Zustandsvariablenfilter



---

# 1 Einleitung

4 und 3 ballbots



---

## 2 Modellbildung und Regelung

---

### 2.1 Model

---

Für die Modellbildung wird der dreidimensionale Ballbot in drei unabhängige planare Modelle aufgeteilt. In jeder Ebene wird das System vereinfacht als Zusammensetzung von drei starren Körpern bestehend aus einer Kugel, ein virtuelles Rad und einen Körper betrachtet und besitzt somit zwei Freiheitsgrade, die sich in eine Translation bzw. Rotation des Balles und eine Rotation des Körpers aufteilen lassen[ETHZ].

Um ein möglichst vereinfachte Modelle der drei Ebenen zu erhalten, werden weitere Annahmen getroffen, die im Folgenden erläutert werden:

- **Kein Schlupf:** Das System besitzt zwei Kontaktpunkte, in denen ein Schlupf auftreten kann. Hierzu zählt zum einen der Kontaktpunkt von Ball und Boden und zum zweiten der Kontaktpunkt zwischen den Rädern und dem Ball. Damit dies gewährleistet wird, müssen die angelegten Drehmomente begrenzt werden.
- **Keine Reibung:** Der einzige Vorgang im System, bei dem die Reibung nicht vernachlässigt wird, ist bei der Rotation des Balles um die  $z$ -Achse. Bei den anderen Vorgängen, bei der in der Realität auch Reibung auftritt, wird im Modell vernachlässigt.
- **Keine Deformation:** Bei dem eingesetzten Ball handelt es sich nicht um eine hohle Stahlkugel, sondern um ein elastischen, mit Luft befüllbaren Ball. Deshalb wird die Deformation des Balles in der Modellbeschreibung nicht mit einbezogen, um die Komplexität gering zu halten.
- **Schnelle Motorendynamik:** Für die Gleichgewichtsstabilisierung des Roboters ist es wichtig, dass die Motoren eine schneller Dynamik als das System aufweisen.
- **Horizontale Bewegung:** Das System wird für die horizontale Bewegung auf einer flachen Oberfläche ohne starken Neigungen ausgelegt. Somit wird die vertikale Bewegung vernachlässigt.

Mit den getroffenen Annahmen ist es möglich, das Modell des Ballbots aufzustellen.

---

### 2.2 Energien

---

Für das Aufstellen der Bewegungsgleichungen des Systems wird der Lagrange Ansatz angewendet. Dazu müssen im Voraus die potentiellen und kinetischen Energien der einzelnen Körper aufgestellt werden. Dabei ist darauf zu achten, dass die Formeln der potentiellen und kineti-

schen Energien für die  $yz$ - und der  $xz$ -Ebene identisch sind und dagegen sich die Energie  $xy$ -Ebene unterscheiden. Deshalb werden im Folgenden die Formeln der  $yz$  und  $xy$ -Ebenen angegeben.

Zunächst werden die kinetischen und potentiellen Energien des Balles in den entsprechenden Ebenen betrachten. Für die kinetische Energie werden folgenden Formeln der jeweiligen Ebenen angegeben.

$$T_{S,yz} = \frac{1}{2} \cdot m_S \cdot (r_S \cdot \dot{\varphi}_x)^2 + \frac{1}{2} \cdot I_S \cdot \dot{\varphi}_x^2 \quad (2.1)$$

$$T_{S,xy} = \frac{1}{2} \cdot I_S \cdot \dot{\varphi}_z^2 \quad (2.2)$$

Da der Ursprung des Weltkoordinatensystems in den Mittelpunkt des Balles gelegt wird, besitzt das System sowohl in der  $yz$  als auch in der  $yx$  Ebene keine potentielle Energie.

$$V_{S,yz} = 0 \quad (2.3)$$

Die kinetischen Energien des virtuellen Rades der jeweiligen Ebene werden mit den folgenden Formeln angegeben.

$$T_{W,yz} = \frac{1}{2} \cdot m_W \cdot ((r_S \cdot \dot{\varphi}_x)^2 + 2 \cdot (r_S + r_W) \cdot \cos(\vartheta_x) \cdot \dot{\vartheta}_x \cdot (r_S \cdot \dot{\varphi}_x) + (r_S + r_W)^2 \cdot \dot{\vartheta}_x^2) + \frac{1}{2} \cdot I_W \cdot \left( \frac{r_S}{r_W} \cdot (\dot{\varphi}_x - \dot{\vartheta}_x) - \dot{\vartheta}_x \right)^2 \quad (2.4)$$

$$T_{W,xy} = \frac{1}{2} \cdot I_W \cdot \dot{\Psi}_z^2 \quad (2.5)$$

Die potentielle Energie des virtuellen Rades in der  $yz$  Ebene kann folgendermaßen berechnet werden.

$$V_{W,yz} = m_W \cdot g \cdot (r_S + r_W) \cdot \cos(\vartheta_x) \quad (2.6)$$

Die kinetischen Energien für die  $yz$  und der  $xy$  Ebene für den Roboterkörper sind ähnlich der des virtuellen Rades und werden folgendermaßen berechnet.

$$T_{B,yz} = \frac{1}{2} \cdot m_A \cdot ((r_S \cdot \dot{\varphi}_x)^2 + 2 \cdot l \cdot \cos(\vartheta_x) \cdot \dot{\vartheta}_x \cdot (r_S \cdot \dot{\varphi}_x) + l^2 \cdot \dot{\vartheta}_x^2) + \frac{1}{2} \cdot I_A \cdot \dot{\vartheta}_x^2 \quad (2.7)$$

$$T_{B,xy} = \frac{1}{2} \cdot I_{W,xy} \cdot \dot{\Psi}_z^2 \quad (2.8)$$

Auch die dazugehörige potentielle Energie ähnelt der des virtuellen Rades. Der Unterschied liegt zum einem im Gewicht des Körpers  $m_B$  und der Höhe  $l$  von Ballmittelpunkt zum Schwerpunkt des Roboteraufbaus.

$$V_{B,yz} = m_A \cdot g \cdot l \cdot \cos(\vartheta_x) \quad (2.9)$$

## 2.3 Bewegungsgleichungen

(matlab) Mit den kinetischen und potentiellen Energien können nun die Bewegungsgleichungen des Ballbots für die jeweiligen Ebenen mit Hilfe der LAGRANGESchen Gleichungen zweiter Art hergeleitet werden. Als Beispiel wird hier die Herleitung für die  $yz$ -Ebene angegeben. Zunächst wird aus den einzelnen Summen der kinetischen und potentiellen Energien die LAGRANGESche Funktion gebildet.

$$L = T - V \quad (2.10)$$

Die Bewegungsgleichungen für den Ballbot resultieren aus dem Lösen der LAGRANGESchen Gleichungen zweiter Art.

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L}{\partial \mathbf{q}} = \mathbf{f}_{NP,yz} \quad (2.11)$$

Mit mathematische Umformungen können diese Bewegungsgleichungen in eine Matrixform überführt werden.

$$\begin{aligned} \mathbf{M}_x &= \begin{bmatrix} m_{tot} \cdot r_S^2 + I_S + \left(\frac{r_S}{r_W}\right)^2 \cdot I_W & -\frac{r_S}{r_W^2} \cdot r_{tot} \cdot I_W + \gamma \cdot r_S \cdot \cos \vartheta_x m_{tot} \\ -\frac{r_S}{r_W^2} \cdot r_{tot} \cdot I_W + \gamma \cdot r_S \cdot \cos \vartheta_x m_{tot} & \frac{r_S^2}{r_W^2} \cdot I_W + I_B + m_b \cdot l^2 + m_W \cdot r_{tot}^2 \end{bmatrix}, \quad \mathbf{M}_x \in \mathbb{R}^{2 \times 2} \\ \mathbf{C}_x &= \begin{bmatrix} -r_S \cdot \gamma \cdot \sin \vartheta_x \cdot \dot{\vartheta}_x^2 \\ 0 \end{bmatrix}, \quad \mathbf{C}_x \in \mathbb{R}^{2 \times 1} \\ \mathbf{G}_x &= \begin{bmatrix} 0 \\ -g \cdot \sin \theta_x \cdot \gamma \end{bmatrix}, \quad \mathbf{G}_x \in \mathbb{R}^{2 \times 1} \end{aligned} \quad (2.12)$$

## 2.4 Zustandsraumdarstellung

Bei den hergeleiteten Bewegungsgleichungen handelt es sich um nichtlineare Funktionen. Um ein Zustandsraummodell zu erhalten, müssen zunächst die Zustände bzw. der Zustandsvektor festgelegt werden. Als Beispiel wird auch hier die  $yz$ -Ebene betrachtet. Mit  $\mathbf{x} = [\varphi_x, \vartheta_x, \dot{\varphi}_x, \dot{\vartheta}_x]$  können die nichtlinearen Funktionen in folgende Form überführt werden.

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}_x^{-1} \cdot (\mathbf{f}_{NP} - (\mathbf{C}_x + \mathbf{G}_x)) \end{bmatrix} \quad (2.13)$$

Das Ziel dieser Arbeit ist das Auslegen eines Reglers, damit der Roboter innerhalb eines kleinen Bereiches um den Arbeitspunkt  $\mathbf{x}_0 = [0, 0, 0, 0]$  auf dem einen Ball balanciert. Hierzu muss

zunächst das nichtlineare Modell um den Arbeitspunkt mit der Taylor-Reihenentwicklung linearisiert werden, um die lineare Zustandsraumdarstellung

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \\ \mathbf{y} &= \mathbf{C} \cdot \mathbf{x} + \mathbf{D} \cdot \mathbf{u}\end{aligned}\tag{2.14}$$

zu erhalten. Mit den zum Roboter dazugehörigen Parametern und dem eingesetzten Arbeitspunkt lauten die konkreten Matrizen für die Zustandsraumdarstellung der  $yz$ -Ebene folgender Maßen:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.4886 & 0 & 0 \\ 0 & 17.1119 & 0 & 0 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 63.5516 \\ -10.6945 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Damit der Roboter auf einen Ball die eigene Gleichgewichtslage einhalten kann, ist es zunächst nebensächlich, welche Position und Geschwindigkeit der Ball dabei einnimmt, da in dem linearisierten Modell nur geringe Abweichungen um die Ruhelage zulässig sind. Dadurch sind für den späteren Entwurf einer Regelung die Zustände des Roboters von Bedeutung und die Zustände für den Ball müssen nicht für eine Regelung berücksichtigt werden. Mit dem neuem Zustandsvektor ... wird die Ordnung des Modells 2.14 reduziert, das zu folgenden Zustandsraummatrizen führt:

$$\mathbf{A}^* = \begin{bmatrix} 0 & 1 \\ 17.1119 & 0 \end{bmatrix} \quad \mathbf{B}^* = \begin{bmatrix} 0 \\ -10.6945 \end{bmatrix} \quad \mathbf{C}^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{D}^* = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Die Prüfung auf Stabilität der beiden Modellen erfolgt anhand der Lage der Eigenwerte. Das ursprüngliche als auch das reduzierte Modell besitzt folgende Eigenwerte:

$$\lambda = \begin{pmatrix} 0 \\ 0 \\ 4.1367 \\ -4.1367 \end{pmatrix} \quad \lambda^* = \begin{pmatrix} 4.1367 \\ -4.1367 \end{pmatrix}\tag{2.15}$$

Das ursprünglich Modell besitzt zwei grenzstabilen Eigenwerte  $\lambda_{1,2} = 0$  auf der Imaginären Achse und einen instabilen Eigenwert  $\lambda_3 = 4.1367$  in der rechten Halbebene des Bildbereiches. Somit ist das System instabil. Bei dem reduzierten Modell handelt es sich ebenfalls um ein instabiles System, da es auch einen Eigenwerte  $\lambda_1^* = 4.1367$  besitzt.



---

Die Untersuchung der Steuer- und Beobachtbarkeit erfolgt in beiden Modelle mit den entsprechenden Kriterien von KALMANN. In beiden Fällen besitzen die Steuerbarkeits- und Beobachtbarkeitsmatrizen vollen Rang, wodurch beide Modelle vollständig steuer- und beobachtbar sind. Mit diesen Systemeigenschaften ist mit Hilfe eines Reglers möglich, die entsprechende instabile System zu stabilisieren. Das Auslegen eines geeigneten Regler wird nachfolgend erläutert. [ETHZ,MGR]

---

## 2.5 Reglerentwurf

---

Der Entwurf eines geeigneten Regler zur Stabilisierung der Gleichgewichtslage des Roboters wird im Folgenden auf das reduzierte Model beschränkt. Für die beiden Modelle der  $yz$ - und  $xz$ -Ebene wird jeweils eine Linear-quadratische Regelung (LQR) ausgelegt, die auf die Minimierung des Gütemaß

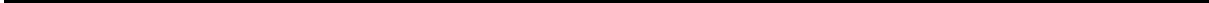
$$J = \int_0^{\infty} \mathbf{x}^T(t) \cdot \mathbf{Q} \cdot \mathbf{x}(t) + \mathbf{u}^T(t) \cdot \mathbf{R} \cdot \mathbf{u}(t) \quad (2.16)$$

durch einen vorgegebenen Stellgrößenverlauf basiert. In das Güteintegral geht quadratisch zum einen der Trajektorienverlauf  $\mathbf{x}(t)$  und der Stellgrößenverlauf  $\mathbf{u}(t)$  ein. Dabei handelt es sich bei der symmetrischen und positiv semidefiniten Matrix  $\mathbf{Q}$  um eine Gewichtungsmatrix für den Trajektorienverlauf. Die symmetrische und positiv definite Matrix  $\mathbf{R}$  gewichtet die Stellgröße.[AD]



---

## 3 Konstruktion



---

## 4 Simulation

Die Simulation des Ballbot's wurde mittels Gazebo realisiert und kann mittels eines globalen ROS launch files gestartet werden. Sie steht frei zur Verfügung und der Programmcode kann über [?] abgerufen werden. Zudem gibt es ein Youtube Video [1] das zeigt, wie man die Simulation auf einem Ubuntu-Betriebssystem mit ROS-Kinetic und Gazebo7 testen kann.

---

### 4.1 3D Simulatoren

Für eine 3D Simulation des Ballbot's bieten sich grundsätzlich zwei 3D Simulatoren an: Gazebo und V-Rep. Die Unterschiede dieser beiden Simulatoren sind in Tabelle 4.1 aufgeführt.

Auf dem Up-level Computer (dem UP-Board) des Ballbots soll ROS<sup>1</sup> laufen. Aus diesem Grund haben wir uns für den Gazebo Simulator entschieden, denn dieser ist der standard Simulator von ROS und daher besser integriert als V-REP[2, S. 1]

---

### 4.2 Simulation von omnidirektionalen Rädern

Bei der Simulation des Ballbot's in Gazebo stellt sich zunächst die Frage wie man die omnidirektionalen Räder simulieren soll. Hierzu gibt es zwei Möglichkeiten:

Die erste Möglichkeit besteht darin, das Omnidirektionale Rad ohne freidrehende kleine Räder zu simulieren. Um die freidrehenden kleinen Räder zu simulieren gibt man dem Rad zwei verschiedene Reibungskoeffizienten für die unterschiedlichen Reibungsrichtungen vor. Die Reibungskoeffizienten werden in gazebo  $\mu_1$  und  $\mu_2$  genannt. Der erste Reibungskoeffizient muss hierbei so gewählt werden, dass er der Reibung des Balls entspricht. Der zweite Reibungskoeffizient muss zu 0 gewählt werden, denn dieser simuliert die freidrehenden kleinen Räder des Omnidirektionalen Rades. Zusätzlich muss man noch den  $fdir_1$  parameter von gazebo setzen. Dieser gibt an in welche Richtung des aktuellen Gelenks (Joints) der  $\mu_1$  parameter zeigen soll.

In Abbildung 4.1 a) ist die einfachste Modellierungs-Möglichkeit eines Ballbot's dargestellt. In dem Ausschnitt in Abbildung 4.1 b) sieht man, wie man die Reibungskoeffizienten  $\mu_1$  und  $\mu_2$  sowie den  $fdir_1$  Parameter für dieses Rad einstellen müsste, um unendlich viele freie

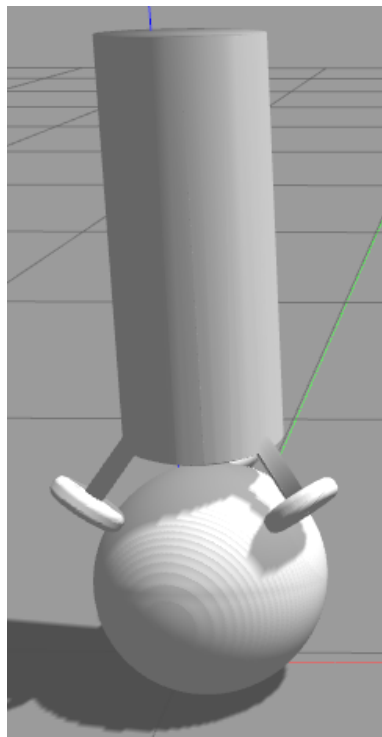
---

<sup>1</sup> Das Robot Operating System (ROS) ist eine Open Source Middleware. ROS hat eine riesige Community und ermöglicht es die verschiedenen Komponenten eines Roboters (Sensoren und Aktoren) geschickt miteinander zu verbinden. So ist es möglich einen Roboter sehr elegant zu simulieren und mittels eines UP-Level Computers zu steuern.

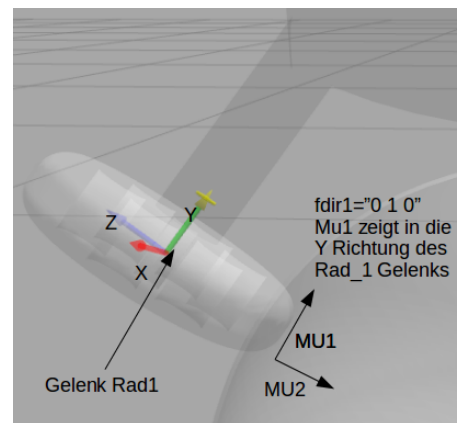
**Tabelle 4.1.:** Unterschiede zwischen den 3D Simulatoren V-REP und Gazebo.

Kriterium	Gazebo-Simulator	V-Rep-Simulator
Lizenz	Open Source Programm	Kommerzielle und kostenlose Version
ROS Integration	Gazebo ist der Standard Simulator von ROS. Gazebo wird als ein ROS Node behandelt und kann daher sehr gut in ROS integriert werden.	V-REP hat keine direkte Anbindung an ROS. Es läuft neben ROS in einem extra Terminal. Jedoch existiert ein Plugin mit dem auf ROS Topics und Services zugegriffen werden kann.
Plugins für Sensoren	Gazebo stellt bereits einige Plugins für Kameras, Laser Scanner etc. bereit. Diese können in einem xml file definiert werden.	V-REP stellt eine sehr benutzerfreundliche graphische Methode zur Verfügung um ein Modell mit Sensoren auszustatten.
CPU Auslastung	Gazebo lastet die Hardware sehr stark aus und ist bis zu 20% langsamer als V-REP.	V-REP hat im Gegensatz zu Gazebo eine konstante CPU Auslastung beim Zugriff auf ROS Nodes.
Community	Gazebo hat eine riesige Community die sehr viele Plugins für neue Sensoren selbst entwickelt und zur Verfügung stellt. Fragen gestellt werden.	V-REP ist nicht ganz so bekannt und hat lediglich 2570(01.2018) Forenmitglieder. Gazebo hat dagegen 3200.

Räder zu simulieren. Leider hat diese einfache Modellierungsmöglichkeit beim Testen nicht funktioniert, da in Gazebo7 der `fdir1` Parameter nicht richtig funktioniert (siehe ....).<sup>2</sup>



**(a)** Übersicht des gesamten Aufbaus des Ballbots.



**(b)** Simulation der freien Räder des Omnidirektionalen Rades.

**Abbildung 4.1.:** Die einfachste Simulations Möglichkeit eines Ballbots in Gazebo7.

Die zweite Möglichkeit ist sehr viel aufwendiger, denn sie besteht darin, das echte Omnidirektionale Rad mit allen kleinen Rädern zu simulieren. Hierfür muss zunächst das Omnidirektionale Rad ohne die freilaufenden kleine Subräder in die Gazebo Simulation geladen werden. Anschließend müssen die Subräder mit richtiger Orientierung und Position ebenfalls in die Simulation geladen werden. Das Omnidirektionale Rad sowie ein einzelnes Subrad wurde hierfür zunächst in Solid Edge konstruiert und anschließend als .stl exportiert. Diese .stl Dateien werden dann mittels einer .xml Datei in die Gazebo Simulation geladen.

Für unsere finale Ballbot Simulation haben wir die zweite Möglichkeit benutzt, denn sie ist der Realität sehr viel Näher als die Modellierung von unendlich vielen kleinen Subrädern. Nachteilig bei der Simulation aller 30 Subräder ist jedoch, der deutlich größere Berechnungsaufwand, der in unserem Falle die Simulation auf einen RealTime Faktor von 0.2 verlangsamt hat. Das heißt die Simulation lief fünf mal langsamer als sie in Echtzeit laufen würde.

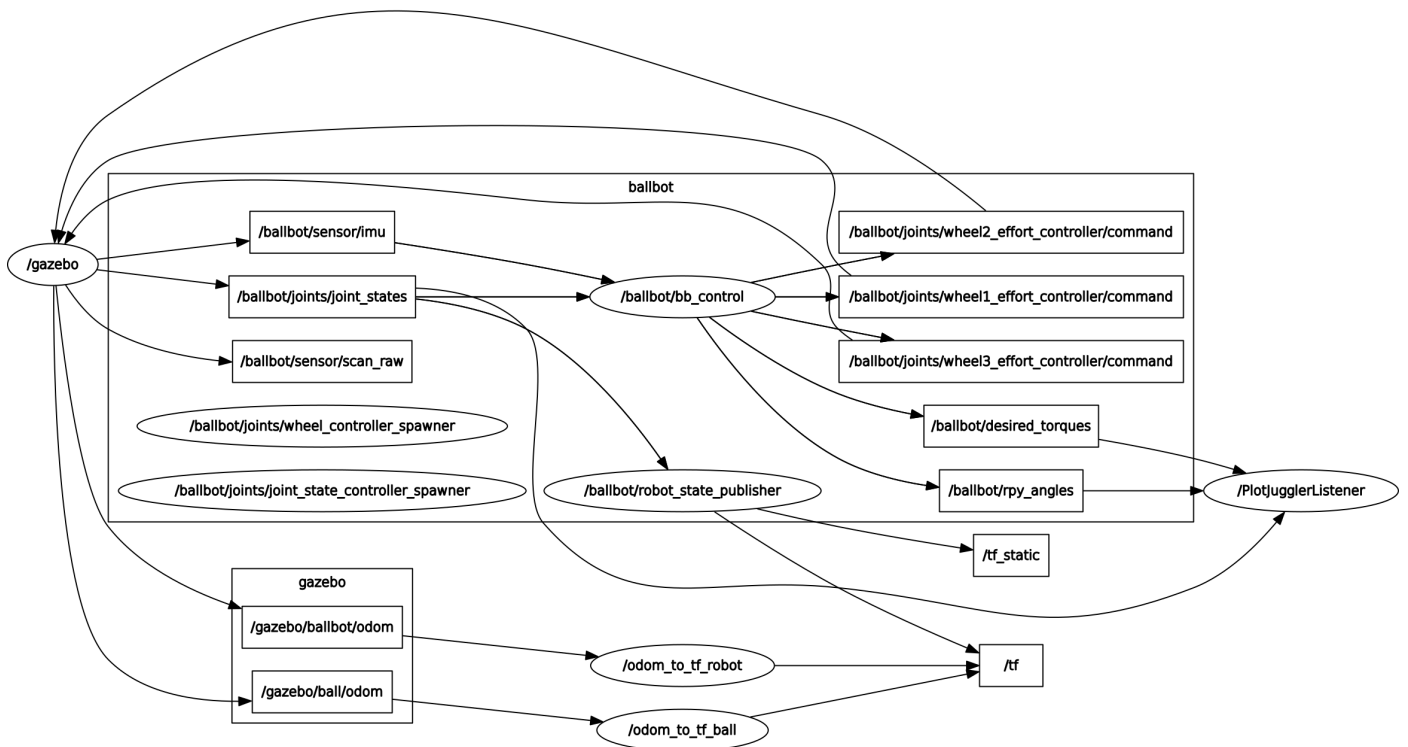
<sup>2</sup> Seit Gazebo8 sollte der `fdir1` Parameter wieder richtig funktionieren. Dies wurde jedoch in dieser Arbeit nicht weiter betrachtet.

## 4.3 Simulations Aufbau

Dieses Kapitel zeigt zunächst exemplarisch die Kommunikation zwischen ROS und Gazebo. Anschließend wird näher auf die verwendeten Plugins eingegangen. Zudem wird auf dynamische Eigenschaften der Simulation wie etwa die Steifigkeit einzelner Elemente eingegangen. Zum Schluss wird noch auf zwei Visualisierungsprogramme(RVIZ und PlotJuggler) eingegangen.

### 4.3.1 Kommunikation zwischen ROS und Gazebo

Die Simulation mittels Gazebo wurde komplett in ROS aufgebaut. Sie besteht aus mehreren Nodes (Teilprogrammen) die alle durch ein globales ROS launch file gestartet werden. Beim Ausführen der Simulation sind sehr viele Teilprogramme(Nodes) aktiv die untereinander über sogenannte Topics Nachrichten austauschen. Abbildung 4.2 zeigt den sogenannten



**Abbildung 4.2.:** Übersicht aller Teilprogramme(Nodes) und deren Topics, die beim Starten der Simulation aktiv sind und Nachrichten austauschen. Hierbei sind die Nodes mit Ellipsen und die Topics mit Rechtecken gekennzeichnet. Das Bild wurde mit dem ROS Programm `rqt_graph` erstellt.

ROS Graph der aktiven Nodes und deren Topics nach dem Starten des globalen ROS launch files. In der Mitte dieser Abbildung sieht man das den Node `/ballbot/bb_control` des namespaces



---

ballbot. Dies ist das Teilprogramm das die Regelung des simulierten Ballbots beinhaltet. Hierfür liest(subscribt) es Nachrichten der Topics `/ballbot/joints/joint_states`<sup>3</sup> und `/ballbot/sensor/imu` ein, berechnet damit die entsprechenden Drehmomente für die einzelnen Räder und veröffentlicht (published) die Nachrichten mit den berechneten Drehmomenten auf den Topics `/ballbot/wheelx_effort_controller/command`. Der Node `/gazebo` wiederum liest diese Drehmoment Befehle der einzelnen Omnidirektionalen Räder ein und dreht entsprechend in der sichtbaren Simulation die Räder.

Es sei noch darauf hingewiesen, dass es möglich ist Gazebo mit ROS zu synchronisieren. Möchte man zum Beispiel einen Regler implementieren, der eine sehr große Update Frequenz (bzw. eine geringe Sample Time) aufweist, die Berechnung der Verstärkungsfaktoren dieser Regelung jedoch länger dauert als die Sample Time, so muss die Regelung mit Gazebo synronisiert werden. Hierfür gibt es die Möglichkeit, Gazebo pausiert zu starten und auch die ganze Zeit pausiert zu lassen. Ist nun die Berechnung der Regelung fertig, schickt man einen rosservice call an gazebo um die Simulation einen Schritt weiterlaufen zu lassen. Anschließend wird der nächste Regelunswert berechnet. Bei dem simulierten Ballbot wurde eine Update Frequenz von 100Hz benutzt. Die Verstärkungsfaktoren der 2D Regelung wurden jedoch mit mindestens 1000Hz berechnet. Daher musste Gazebo nicht mit dem entsprechendem ROS Node, der die Regelung darstellt (`/ballbot_controll`) synronisiert werden.

---

#### 4.3.2 Plugins der Simulation

---

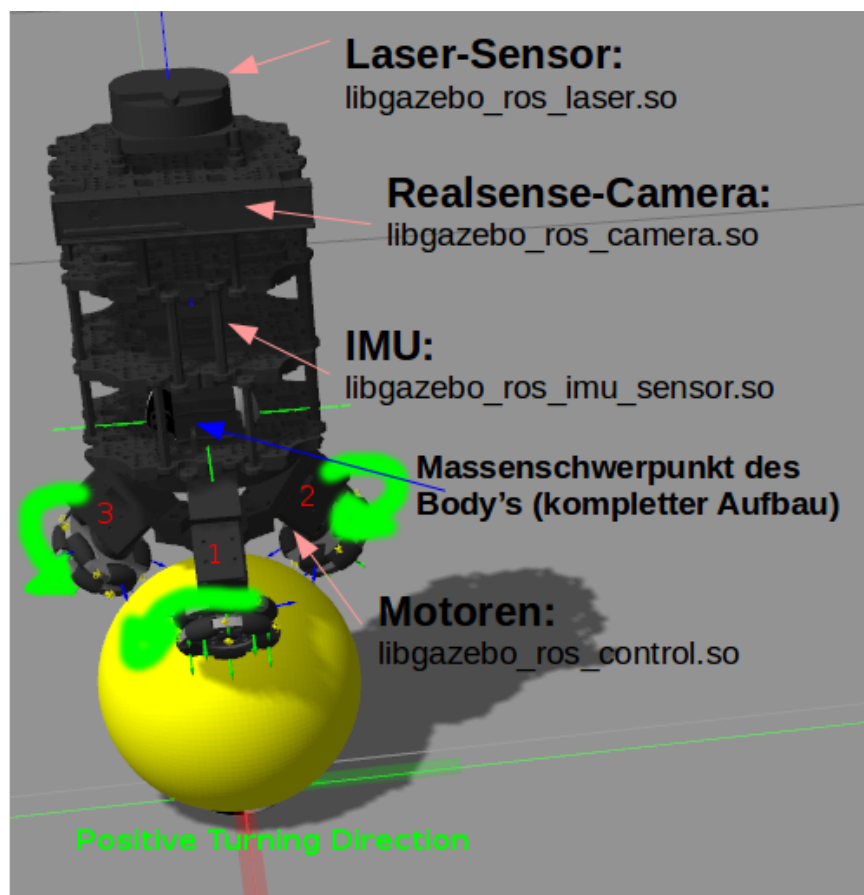
Die Sensoren des Ballbot's können in Gazebo durch Plugins simuliert werden. Abbildung 4.3 zeigt die simulierten Sensoren und deren Plugin Bibliotheken. Hierbei wurde die IMU mit einer `update_rate` von 200Hz sowie mit einem Gaußschen Rausschen von 0.01 simuliert. Als Motoren Interface wurde das Joint Effort Interface verwendet, welches über das `ros_control` Paket<sup>4</sup> zur Verfügung steht. Die Motoren wurden dabei mit einem PID Regler simuliert. Dabei wurden die Verstärkungsfaktoren zu  $P=100$ ,  $I=0.01$  und  $D=10$  gewählt. Für weitere Details zur Implementierung der Real-Sense Kamera sowie des LDS Laser Scanners sei auf den Programmcode<sup>5</sup> verwiesen.

---

<sup>3</sup> Dieses Topic enthält eine Nachricht die die aktuellen Rad Drehmomente, Geschwindigkeiten [rad/sec] und deren absolute Positionen enthält. Für die Regelung der Odometrie werden jedoch nur die Rad Drehgeschwindigkeiten verwendet.

<sup>4</sup> Dieses Paket ist standardmäßig bei der ROS-Kinetic Version dabei. Weitere Informationen zu diesem Paket gibt es hier: [http://wiki.ros.org/ros\\_control](http://wiki.ros.org/ros_control) .

<sup>5</sup> siehe: [https://github.com/CesMak/bb/blob/master/src/ballbot/bb\\_descrpition/urdf/bb\\_double\\_wheel.gazebo.xacro](https://github.com/CesMak/bb/blob/master/src/ballbot/bb_descrpition/urdf/bb_double_wheel.gazebo.xacro)



**Abbildung 4.3.:** Der Simulierte Ballbot mit den simulierten Sensoren und deren Plugin Bibliotheken.

---

### 4.3.3 Simulierte Dynamik-Eigenschaften

---

	name(xacro)	description
Nicht an dieser tabelle orientieren.	mu1	is the Coulomb friction coefficient for the first friction direction
	mu2	is the friction coefficient for the second friction direction
	fdir1	3-tuple specifying direction of mu1 in the collision local
	kp	spring constant equivalents of a contact as a function of
	kd	spring damping constant equivalents of a contact as a f
	cfm	Constraint Force Mixing parameter.
	erp	Error Reduction Parameter.
	min_depth	Minimum depth before ERP takes effect.
	max_Vel	Maximum interpenetration error correction velocity. If s
	slip1	Artificial contact slip in the primary friction direction
	slip2	Artificial contact slip in the secondary friction direction.

Ball so hart wie möglich machen! in der simulation da wie in annahmen gesagt.

---

### 4.3.4 Visualisierungsprogramme

---

RVIZ, RQT-Multiplot



---

## 5 Auswertung



---

# A Parameterliste

NICHT AN DIESER TABELLE orientieren.

Parameter	Variable	Wert	Quelle
Masse Ball	$m_S$	0,3280 kg	Gemessen
Masse Motor	$m_M$	0,0820 kg	Datenblatt
Masse omnidirektionales Rad	$m_{OW}$	0.0520 kg	Gemessen
Masse virtuelles Rad	$m_W$	0,4020 kg	Gemessen
Masse Roboterkörper (mit Motoren/Räder)	$m_B$	1,603 kg	Gemessen
Masse Roboterkörper (ohne Motoren/Räder)	$m_B$	1,2010 kg	Gemessen
Radius Ball	$r_S$	0,0800 m	Datenblatt
Radius virtuelles Rad	$r_W$	0,0300 m	Datenblatt
Radius Körper	$r_B$	0,0703 m	Gemessen
Höhe Massenschwerpunkt	$l$	? m	SolidEdge
Höhe Körper	$h$	? m	SolidEdge
Trägheitsmoment Ball	$I_S$	0,0013 $kgm^2$	Berechnet
Trägheitsmoment Rotor	$I_M$	3.8e-8 $kgm^2$	Datenblatt
Trägheitsmoment omnidirektionales Rad	$I_{OW}$	2,34e-5 $kgm^2$	Berechnet
Trägheitsmoment virtuelles Rad (yz-Ebene)	$I_{W,yz}$	0.00357 $kgm^2$	Berechnet
Trägheitsmoment virtuelles Rad (xz-Ebene)	$I_{W,xz}$	0.00357 $kgm^2$	Berechnet
Trägheitsmoment virtuelles Rad (xy-Ebene)	$I_{W,xy}$	0.0143 $kgm^2$	Berechnet
Trägheitsmoment Körper (yz-Ebene)	$I_{B,yz}$	0.0880 $kgm^2$	SolidEdge
Trägheitsmoment Körper (xz-Ebene)	$I_{B,xz}$	0.0880 $kgm^2$	SolidEdge
Trägheitsmoment Körper (xy-Ebene)	$I_{B,xy}$	0.0070 $kgm^2$	SolidEdge
Übersetzungsverhältnis	$i$	353,5	Datenblatt
Erdbeschleunigung	$g$	9,81 $\frac{m}{s^2}$	Datenblatt



---

# Literaturverzeichnis