OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

MASTERARBEIT

Tamara Szecsey

Intelligentes Schuhwerk für den humanoiden NAO Roboter basierend auf Magneto- und Elektrostiction zur Verbesserung der Bodenhaftung

Fakultät:	Elektro- und Informationstechnik
Studiengang:	Master Electrical- and Microsystem Engineering
Abgabefrist:	31. März 2021
Betreuung:	Prof. Dr. Gareth Monkman
Zweitbegutachtung:	Dr. Dirk Sindersberger

Erklärung

1. Mir ist bekannt, dass dieses Exemplar der Masterarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.
2. Ich erkläre hiermit, dass ich diese Masterarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum und Unterschrift

Vorgelegt durch:	Tamara Szecsey
Matrikelnummer:	3140789
Studiengang:	Master Electrical- and Microsystem Engineering
Bearbeitungszeitraum:	1. Juni 2020 – 31. März 2021
Betreuung:	Prof. Dr. Gareth Monkman
Zweitbegutachtung:	Dr. Dirk Sindersberger

Inhaltsverzeichnis

1 Einleitung

Kapitel 2.3 erklärt die Definition und Eigenschaften von Magneto-aktiven Polymeren (MAP), welche als Sohle für den Nao Roboter eingesetzt wurden. Diese Erklärungen basieren auf einem Buch von Pelteret und Steinmann [1], welches ich für tiefergehende Lektüren empfehle.

2 Theoretischer Hintergrund

2.1 Aufbau des NAO

NAO ist ein $574\,\mathrm{mm}$ großer, humanoider Roboter (siehe Abb. 1) ursprünglich entwickelt von dem französischen Unternehmen Aldeberan Robotics, welche 2015 von Softbank Group aufgekauft [3] und in Softbank Robotics umbenannt wurde. Während NAO's große Schwester Pepper mit ihren $1,20\,\mathrm{m}$ mit einem Tablet und Rollen statt Beinen ausgestattet ist [4], gibt es NAO in verschiedenen Ausführungen, unter anderem nur ab der Hüfte aufwärts oder mit Beinen. Es handelt sich hier um Roboter, die unter anderem Kinder und Jugendlichen die Robotik näher bringen sollen und der Vorführung von Mensch-Roboter Interaktionen dienen. NAO bietet außerdem die Gelegenheit zweibeinige Robotersysteme zu studieren und ist bereits in psychologischen Studien verwendet worden (cite).

Abb. 1. NAO V6 [2]

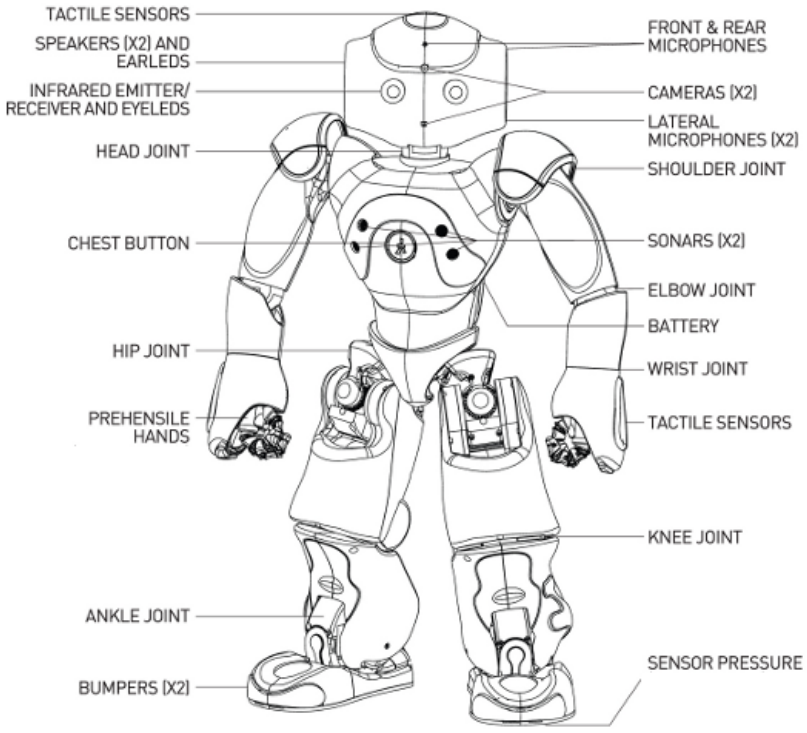

Abb. 2. Sensorenüberblick des NAO-H25 Version 6 [5, in /H25]

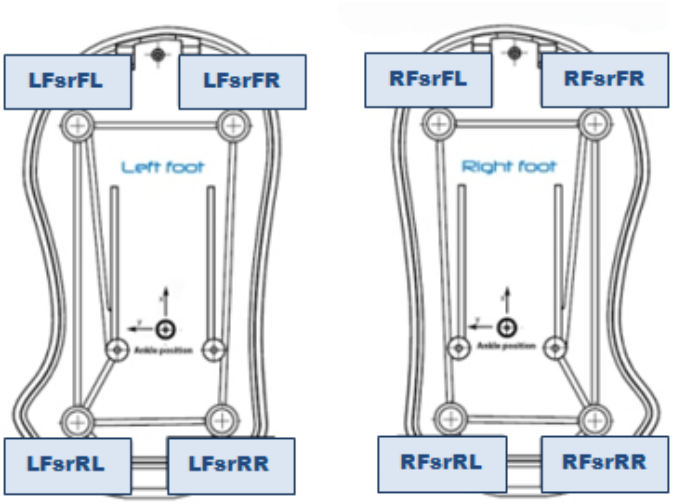

Abb. 3. Drucksensoren in den Füßen von NAO [2, in /Technical overview/FSRs]

Das hier verwendete Modell ist NAO-H25 Version 6, dessen Sensoren in Abb. 2 zu sehen sind. Im Unterschied zu anderen Ausführungen besitzt NAO-H25 Drucksensoren an Händen und Fußsohlen. Er gehört zu den kommerziellen Robotern deren Gelenke positionsbasierenden sind [6], hat 25 Freiheitsgrade und wiegt $5,4\,\mathrm{kg}$. Über die an der Brust angebrachten Sonar Sensoren, die Kameras oberhalb und unterhalb der Augen LEDs, die Vorder- und Rückseitigen Mikrofone, den Stoßfängern an den Füßen sowie den Kontaktsensoren an Händen und Kopf kann der NAO mit seiner Umwelt vielseitig interagieren. Jedes Gelenk ist mit Sensoren für die Winkelmessung, den Stromverbrauch und Temperaturmessung ausgerüstet und in seiner Brust befindet sich außerdem ein Gyroskop. Auf die in dieser Arbeit verwendeten Messausgaben wird im Folgenden genauer eingegangen.

Druckempfindlicher Widerstand

An den Fußssohlen des NAO befinden sich pro Fuß vier sogenannte *Force Sensitive Resistors* (FSR) zu sehen in Abb. 3. Diese ändern ihren Widerstand sobald Druck ausgeübt wird und messen im Bereich von 0 bis $25\,\mathrm{N}$.

Jeder dieser Sensoren kann einzeln ausgelesen werden, so zum Beispiel LFsrFL unter `Device/SubDeviceList/LFoot/FSR/FrontLeft/Sensor/Value`. Des weiteren können der berechnete zweidimensionale Massenschwerpunkt und das Gesamtgewicht ausgegeben werden. Diese Werte sind allerdings unzuverlässig, sobald wenig oder kein Gewicht auf den Sensoren lastet. In [7] haben Shayan et al. die eingebauten FSR mit barometrischen Drucksensoren verglichen und sind zu dem Schluss gekommen, dass die Sensoren, welche in NAO verbaut sind, unzuverlässig arbeiten. Deshalb werden hier die Ausgaben kritisch betrachtet, sowie die Balance des Roboters zusätzlich mit dem Gyroskop erfasst.

Eine weitere Einschränkung der Messungen ist die Kraftverteilung auf die FSRs. Denn in dem ursprünglichen unteren Teil des Schuhs liegt der obere Teil ausschließlich auf den

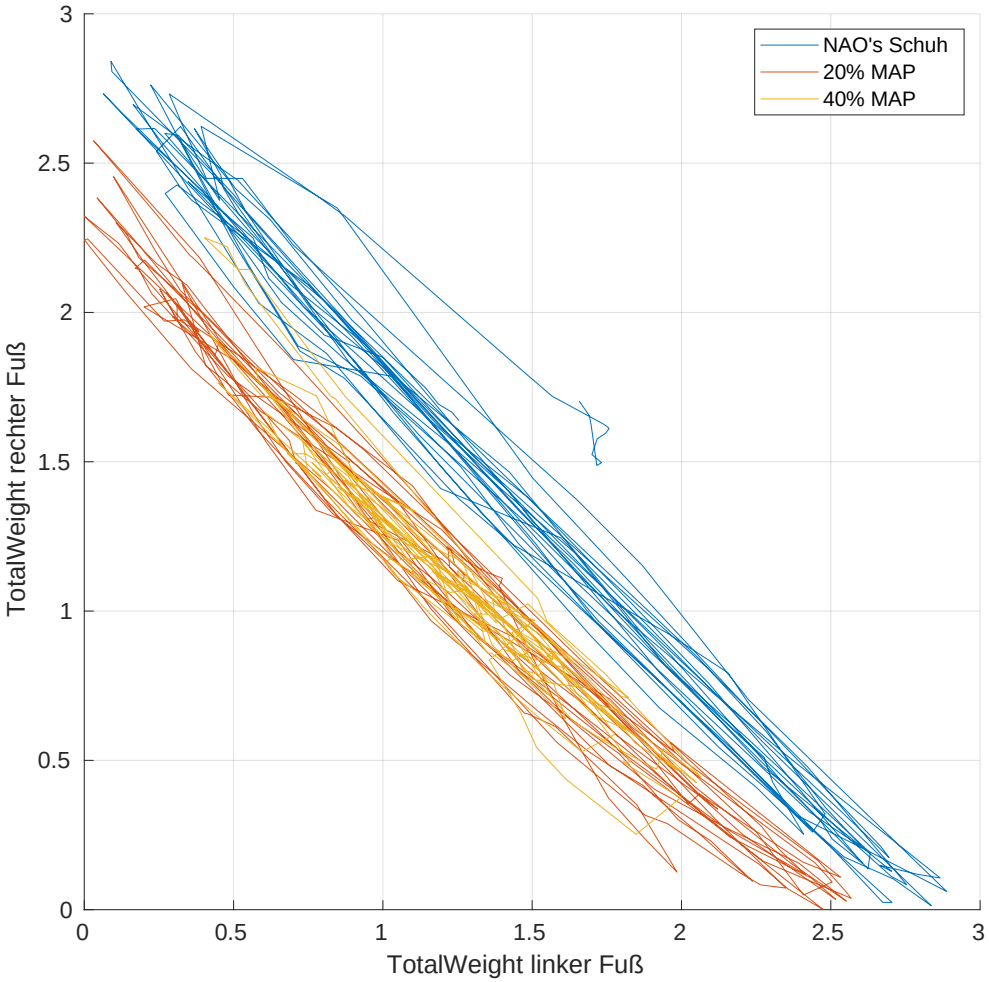

Abb. 4. Berechnetes Gesamtgewicht der Messungen für die normalen Sohlen von NAO in blau und dem Silikon versetzt mit Eisenpartikeln in 20% und 40% Anteilen. Das Gewicht wird in kg ausgegeben.

Sensoren und auf den Verbindungszylindern für die Schrauben auf. Dies ist für die Einlegesohlen aus Silikon bzw. MAP nicht der Fall, zu sehen in den Kapiteln 3.1 und folgende. Dies führt dazu, dass das Gewicht nicht mehr akkurat aufgenommen wird, wie in Abb. 4 zu sehen ist. Hier ist zu sehen, dass sich die Messung des Gewichtes etwa um $0,5\,\mathrm{kg}$ von den NAO Schuhen abweicht.

Aktoren und Sensoren der Beinen

Neben dem für diese Arbeit uninteressanten Bumper Sensoren und den bereits beschrieben Drucksensoren hat NAO eine Ausgabe für jeden eingebauten Aktor mit den Werten:

- `.../Position/Actuator/Value` (`Pos/Act`)
- `.../Position/Sensor/Value` (`Pos/Sens`)

- .../ElectricCurrent/Sensor/Value (Current)
- .../Temperature/Sensor/Value
- .../Hardness/Actuator/Value
- .../Temperature/Sensor/Status

Hierbei unterscheiden sich die Pfade bei „...“ je nach Aktor. Die Bezeichnungen in Klammern dahinter dienen der abkürzenden Benennung für spätere Kapitel.

Die Temperaturausgaben sowie die Starrheit der Motoren liefern keine verwertbaren Messausgaben, da sich beide während einer Messung nicht oder kaum ändern. Pos/Act und Pos/Sens geben ähnliche Werte in rad aus, da ersteres die Ausgabe des Programms vorgibt und zweiteres den tatsächlich gemessenen Wert an dem Gelenk ausgibt. Der Current Wert wird in Ampère gemessen und gibt an, wie viel Strom für das Erreichen der entsprechenden Aktorposition und Starrheit aufgewendet werden muss. Dies bedeutet, dass dieser Wert unter anderem eine Aussage über den Zustand des Aktors geben kann.

Bei den Probegängen des NAO stellte sich heraus, dass er in einer Kurve nach links läuft bei einem Befehl für geradeaus. Softbank Robotics betonte, dass es nicht möglich ist, dass NAO komplett gerade läuft. Grund dafür ist zum einen, dass die Motoren nicht immer komplett identisch hergestellt sind. Zum anderen ist der Gehbefehl, welcher hier ausgeführt wird, ohne Rückkopplungsschleife für Einwirkungen der Umgebung. Näheres zur Software wird in Kapitel 2.2 beschrieben.

Gyroskop

Der NAO Roboter verfügt über eine Inertialeinheit, welche sich zusammensetzt aus den 3 Achsen des Gyrometers, im einer Achsengeschwindigkeit von bis zu $500^\circ/\mathrm{s}$, sowie den 3 Achsen des Beschleunigungssensors, mit einer Beschleunigung bis zu $2\,\mathrm{g}$, zusammensetzt [2, /Technical overview/Inertial unit]. Die Z-Achse des Gyroskops ist allerdings noch nicht verfügbar. Außerdem wird durch diese beiden Sensoren der Winkel des Torsos bestimmt.

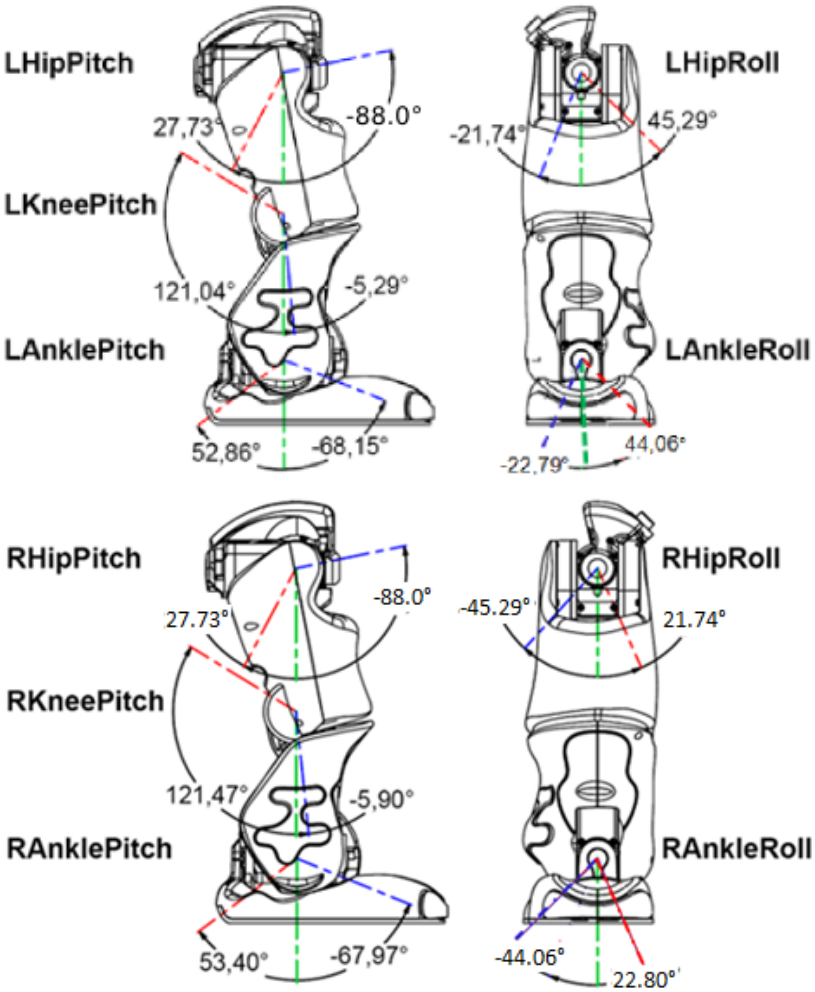

Abb. 5. Das obere Bild zeigt Vorder- und Seitenansicht der Positionen und möglichen Winkel des linken Beins. Die untere Abbildung veranschaulicht dieselben Parameter für das rechte Bein. [2, in /kinematics-data/joints]

2.2 Verwendete Software

Die Rahmenumgebung NAOqi

NAOqi wird die Hauptsoftware genannt, welche auf diesem Roboter sowie auf Pepper läuft, das Betriebssystem NAOqi OS basiert auf Linux. Der NAOqi Rahmen ist eine plattformübergreifende und sprachenübergreifende Umgebung, mithilfe derer Anwendungen für den NAO erstellt werden können. Sie kann über Windows, MacOS und Linux verwendet werden und es werden die Sprachen Python und C++ unterstützt, wobei ersteres direkt auf dem NAO kompiliert werden kann, während C++ komplizierter benutzt wird, aber mehr eingriffe erlaubt. [8, /Former NAOqi Framework/Key concepts]

Außerdem gibt es eine Desktop Anwendung namens Choregraphe, in der Dialoge und Verhaltensmuster erstellt werden können, ohne Code schreiben zu müssen. Außerdem ist

hierüber der Akkustand und aktuelle Position in Form eines virtuellen Bildes des tatsächlichen NAO einsehbar, sowie der autonome Zustand ein- und ausschaltbar. [8, /Choregraphe Suite/What is Choregraphe]

Für die in dieser Masterarbeit benötigten Anwendungen war Python am besten geeignet. Erforderlich waren eine sich wiederholende Schrittabfolge sowie die Aufzeichnung diverser bereits in NAO verbauten Sensoren. Text verarbeitende Funktionen sind in dieser Sprache leicht zu erhalten und anzupassen. Es wurde sich für den `moveTo()` Befehl als Fortbewegung entschieden, denn dieser ist durch das Objekt `post` einem sogenannten *non-blocking call*. Dies ermöglicht das Aufrufen von weiteren Befehlen parallel zur Bewegung. Der Nachteil ist, dass der Gang dadurch unbeaufsichtigt vonstatten geht, d.h. NAO kann seine Schritte nicht seiner Umgebung anpassen. Die aufgezeichneten Daten werden in eine csv Datei übertragen und mit Matlab ausgewertet. Im Anhang ist der gesamte Programmcode 1 abgebildet, welcher auf dem NAO ausgeführt wird.

Konstruktionssoftware

Das CAD-Programm Inventor von Autodesk ist für 3D Konstruktionen ausgelegt und bietet einige nützliche Simulationserweiterungen, welche unter anderem einen Shape Generator enthält [9]. Dieser kann Flächen minimieren während die Stabilität erhalten bleibt, sodass ein minimaler Anteil an Material verwendet werdet wird [10].

Für die Herstellung der Prototypen und Gussformen wurde das FFF (Fused Filament Fabrication) 3D Druckverfahren mit PLA oder PETG Filamenten verwendet und für die Vorbereitung auf den Druck der Slicer Cura von Ultimaker [11].

Datenauswertung mit Matlab

Für die Auswertung wurden neben der gewöhnlichen `plot` Funktion von Matlab [12] auch die *Statistics and Maschine Learning* Toolbox mit deskriptiver Statistik und Visualisierung verwendet [13]. Eine Messreihe ergibt sich aus 20-40 Messungswiederholungen, in denen NAO etwa einen Meter zurücklegt. Jede Auswertung enthält das arithmetische Mittel eines Sensors von allen Messungen der Messreihe. Diese Mittel wurden mit dem Befehl `mean()` berechnet und mit Werten aus weiteren Messreihen, mit anderem Schuh oder anderen Untergrundvoraussetzungen aber gleicher Schrittlänge und Sensorenmessung verglichen. Dieser Vergleich wurde mit den Funktionen `hist` und `scatterhist` der genannten Toolbox angestellt. Ersteres erstellt ein Histogramm und ist damit optimal für eine Häufigkeitsverteilung. Zweiteres erstellt zwei Histogramme von jeweils zwei Vektoren und deinen aus diesen Vektoren resultierenden Scatterplot. Dies ermöglicht eine umfangreiche Verteilungsanalyse für Sensorausgaben, die sich aus zwei Achsen zusammensetzen, wie z.B. der 2 dimensionalen Gyroskop Ausgabe.

2.3 Magneto-aktive Polymere

Begriffserklärung und Eigenschaften

Der Begriff magneto-aktive Polymere (MAP) schließt eine Gruppe von intelligenten, auf Felder ansprechende Materialien ein, welche typischerweise Kombinationen aus einer weichen, polymetrischen Grundlage und magnetisch aktiven Partikeln sind. Diese Partikel werden während dem Vernetzungsprozess des Polymers in dieses eingebettet.

Die wesentlichen Verhaltensweisen, die MAP in der heutigen Zeit attraktiv für seine Verwendung gemacht hat, wurde bereits in den 80ger Jahren von Rigbi und Jilken [14] sowie Rigbi und Mark [15] beschrieben. Ein Jahrzehnt später wurde eine genauere Analyse zum ersten Mal von Ginder und Jolly et al. [16] veröffentlicht. Diese kombinierten, aus mehreren Komponenten bestehenden Materialien stechen durch zwei Schlüsseleigenschaften heraus. Zum einen ist es das magnetostriktive Verhalten, bei dem es sich um das Phänomen der Verformung eines Materials handelt, welches durch ein Magnetfeld hervorgerufen wird. [17] Zum anderen sind es die leicht veränderbaren Materialeigenschaften wie Elastizität und Dämpfungsfaktor, welche hauptsächlich mit der Mikrostruktur des Grundlagenmaterials zusammenhängt. [18] [19]

Außerdem ist entscheidend, wie die magnetischen Partikel in das Polymer eingebettet werden. Je nachdem ob während des Vernetzungsprozesses ein Magnetfeld wirkt, können sich die Partikel kettenförmig ausrichten und dadurch dem MAP eine anisotropisches Verhalten zuführen. Isotropisches MAP hingegen enthält keine gerichteten Partikel. Diese verschiedenen Ausrichtungsarten können sowohl die Steifigkeit verändern als auch bestimmen, ob das MAP in einem Magnetfeld ausgedehnt oder zusammengedrückt wird.

Anwendungsbereiche

Weiche, mit einem Feld manipulierbare, Polymere haben diverse Anwendungsbereiche in akademischen und industriellen Bereichen. Angefangen von anpassungsfähiger Vibrationsabsorption in der Luftfahrt und Automobilindustrie durch das Einsetzen durch Scherung (cite 175,114) Windung(202) und Kompression bzw. Elongation(246) und Vibrationsisolatoren (cite 174) sowie Sensoren (174,334), Ventile und Aktoren (55,254) und anpassungsfähige Sandwichartige Strukturen (575,576,560) bis hin zur Anwendung in der Bionik wie zum Beispiel durch Mikro- und Nanoroboter und Schwimmroboter (438,561.329.219), Schlauchradpumpen (152) und Erschütterungsisolatoren (308).

Des Weiteren wird die Verhärtung bei Anlegen eines Magnetfeldes für Greifer genutzt (cite suchen), ebenso wird der 3D Druck von magneto-aktiven Polymeren (cite von uns) wird bereits erforscht.

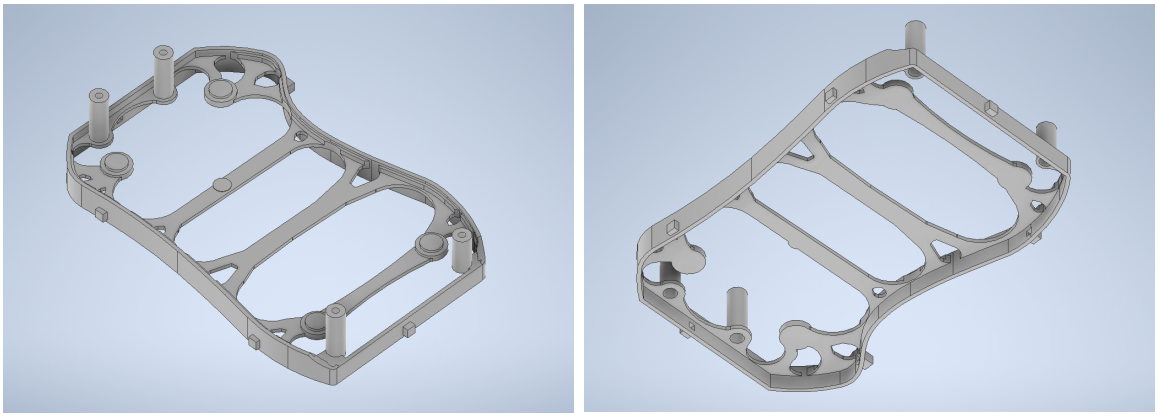

Abb. 6. Linker Schuh in Autodesk Inventors, Links von Oben, Rechts von Unten.

3 Versuchsaufbau

In diesem Kapitel wird auf alle selbstkonstruierten Komponenten eingegangen, welche für die Messungen benötigt wurden. Zum einen gibt es eine Schuhkonstruktion, die es erlaubt, die in einer Gussform angefertigten MAP Sohlen einzuhängen, zum anderen wurde ein Laufsteg mit Rampenfunktion entworfen, welche es erlaubt, Magnete unter die Lauffläche zu montieren.

3.1 Schuhkonstruktion

Die Hülle eines Fußes von NAO besteht aus einem zweiteiligen Oberteil, welches das Fußgelenk abschließt und nur mit dem unteren Teil zusammen mit 4 Schrauben fest sitzt. Die in Kapitel 2.1 beschriebenen vier Drucksensoren liegen dabei in dem unteren Teil auf Erhöhungen auf. Um die Auswirkungen anderer Sohlen für NAO messen zu können, muss der untere Teil des Fußes ausgetauscht werden. Dieser „Schuh“ welcher die ursprüngliche Fußsohle ersetzt, enthält wiederum einen Steckplatz für die MAP Sohlen in verschiedenen Stärkegraden.

In Abb. 6 sind links die vier flachen Zylinder zu sehen, auf denen die Drucksensoren aufliegen. Durch die vier Zylinder an den Seiten werden Schrauben gelegt, welche an das obere Teil des Fußes von NAO geschraubt werden. Die Außenform umschließt den oberen Teil während eine zweite Erhöhung, welche an den Innenseiten verläuft, auf dem Rand des oberen Teils aufsitzt, sodass die Passform fest ineinander greift. Da das gesamte Gewicht des NAO auf den 4 Drucksensoren lastet, kann Druckmaterial für die restliche Gesamtfläche bis auf stabilitätserhaltende Streben eingespaart werden. Diese Form wurde mit dem Shape Generator von Autodesk Inventor (cite) generiert, sodass sie längs bis zu einem $45°$ Winkel ohne zu brechen gebogen werden kann. Die instabilsten Stellen sind die Zylinder der Schraubvorrichtung, welche mit Sekundenkleber verstärkt werden können. Die Instabilität ist auf den schichteweise Druckvorgang mit einem

Die Unterseite, zu sehen in Abb. 6 rechts, ist ein Hohlraum für die MAP Sohle zusammen mit den viereckigen Steckeinlässen für die Halterung. Die Seiten des Schuhs sind so hoch,

dass das MAP, welches in der Gussform gegossen wurde, etwa $1\,\mathrm{mm}$ herausragt. Andernfalls würde NAO auf der Schuhkante laufen und nicht auf dem MAP.

3.2 Herstellung des MAP

Das verwendete Polymer als Grundlage für das MAP ist ein additiv vernetzendes Silikon von (Firma und cite einbinden). Während dem Mischverfahren ist es flüssig und muss deshalb in eine Form gegeben werden. Silikon selbst lässt sich nur sehr schlecht durch etwaige Klebstoffe nach der Vernetzung verkleben. Deshalb wird hier wie in Abb. 7 zu sehen ist, eine $2\,\mathrm{mm}$ dicke Stangenkonstruktion eingehängt, welche bis auf die 6 Enden mit MAP umschlossen wird. Diese aus PLA gedruckte Konstrukt ist flexibel und kann deshalb durch Verbiegen in die Verankerungen gedrückt werden. Nach der vollständigen Vernetzung kann die Sohle aus der Form entnommen und in den Schuh aus dem vorherigen Kapitel eingesetzt werden.

Die vier Zylinder dienen als Platzhalter um die sechs Ecken in der Halterung des Schuhs für einen besseren Halt festzukleben und dann durch die Löcher des MAPs die Schrauben lockern zu können.

3.3 Laufstegkonstruktion

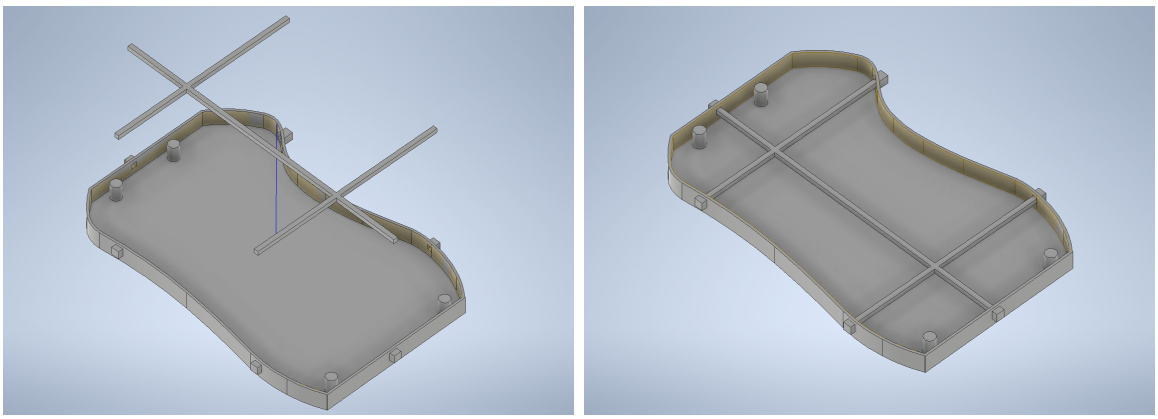

Abb. 7. Gussform der MAP Sohlen. Links ist die Innenhalterung herausgenommen, rechts ist sie eingespannt in den sechs Eckhalterungen.

Der NAO Roboter ist für den Einsatz auf geraden Bodenflächen im Innenbereich ausgelegt wobei er bei einem Bewegungsablauf ohne Anpassung an die Umwelt wie mit dem Befehl `moveTo()` durch Rutschen nicht immer die gleiche Strecke zurücklegt. Um wiederholbare Messreihen garantieren zu können ist eine Teststrecke von Nöten. Des Weiteren sind verschiedene, flache Untergründe für eine Sohlenentwicklung interessant. Außerdem kann auf das MAP nur Einfluss genommen werden, wenn ein magnetisches Feld angelegt wird. Deshalb muss wurde ein Laufsteg mit einem Hohlraum angefertigt, um unter der Fläche, auf der NAO läuft, Magneten angebracht werden.

Der Laufsteg besteht aus einer $120 \times 66,4\,\mathrm{cm}$ großen Pressholzplatte, die auf der Oberseite mit einem Aluminiumkonstrukt erweitert ist, welches die Einschubplatten von beiden Längsseiten und nach oben hin abschließt, Abb. 8 links. Auf den kurzen Seiten verriegeln jeweils Zwei drehbare Keile den Einschub, sodass die Platten eingeschlossen werden, Abb. 9.

Abb. 9. Seitenansicht der Rampe mit einer Breite von $66,4\,\mathrm{cm}$.

Auf der Unterseite sind an den Längsseiten zwei mit T-Nut versehene Aluminiumstangen angebracht, sowie eine zweiteilige Stangenkonstruktion, die eine Winkelverstellung mit Raste erlaubt, zu sehen in Abb. 8 rechts. Die einstellbaren Winkel betragen ca. $5°$ bis $17°$, oder es wird für $0°$ vollständig eingeklappt.

Man hat bereits NAO schräge Flächen gehen lassen wie in (cite). Dies erfordert einen komplett anderen Gang und hätte den Rahmen dieser Arbeit gesprengt. Die Neodymmagnete, die verwendet wurden, haben eine Haftkraft von ca. $16\,\mathrm{kg}$, eine Maße von $40 \times 40 \times 4\,\mathrm{mm}$ [20] und wurden an die Unterseite der Rampe geschraubt. Die ersten Versuche ergaben schließlich, dass das MAP nur bei einem Abstand unter den Einlageplatten reagierte. Deshalb wurde in den hiesigen Messungen nur ohne Platten gemessen.

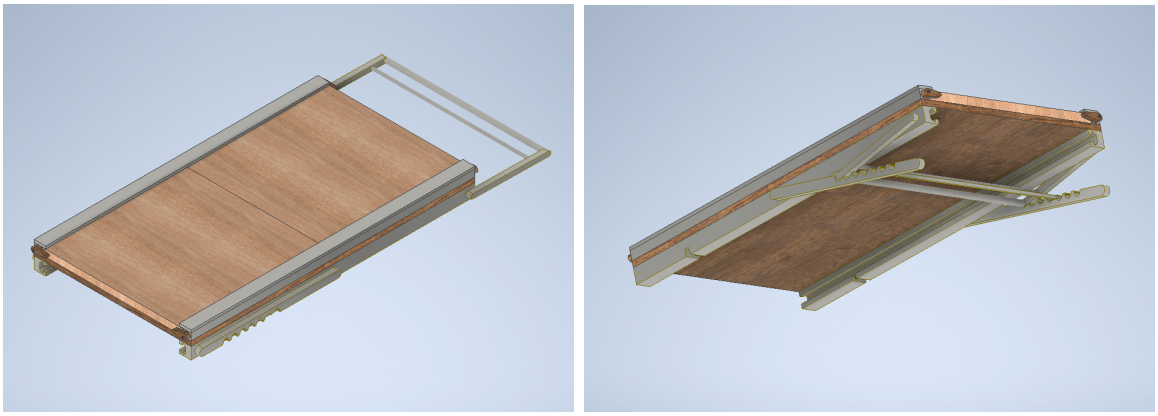

Abb. 8. Laufstegrampenkonstruktion mit zwei austauschbaren Platten und einer Winkelverstellung mit Raste. Links: Sicht von schräg oben mit eingeklappter Winkelverstellung. Rechts: Sicht von schräg unten mit niedigster Winkeleinstellung.

4 Auswertung und Interpretation

5 Anhang

Code 1. Pythonprogramm für Messaufnahmen

```
# ! /usr/bin/env python
# -*- encoding: UTF-8 -*-

"""Example: Use getData Method to Use FSR Sensors"""

import qi
import argparse
import sys
import time
import csv
import re
import shutil
from tempfile import mkstemp
import os

def sed(pattern, replace, source, dest=None, count=0):
    """Reads a source file and writes the destination file.

    In each line, replaces pattern with replace.

    Args:
    pattern (str): pattern to match (can be re.pattern)
    replace (str): replacement str
    source  (str): input filename
    count (int): number of occurrences to replace
    dest (str):   destination filename, if not given,
       source will be over written.
    """

    fin = open(source, 'r')
    num_replaced = count

    if dest:
        fout = open(dest, 'w')
    else:
        fd, name = mkstemp()
        fout = open(name, 'w')

    for line in fin:
        out = re.sub(pattern, replace, line)
        fout.write(out)

```

```
42          if out != line:
43              num_replaced += 1
44              if count and num_replaced > count:
45              break
46      try:
47          fout.writelines(fin.readlines())
48      except Exception as E:
49          raise E
50
51      fin.close()
52      fout.close()
53
54      if not dest:
55          shutil.move(name, source)
56
57
58  def zeilen_aufteilen(file):
59      sed(',platzhalter,', '\n', file)
60      sed(',platzhalter', '', file)
61
62
63  def recordData(memory_service):
64      """ Get pressure sensor data from ALMemory
65      Returns a matrix of values
66
67      """
68      print "Recording data..."
69      data = list()
70      for range_counter in range(1, 230):
71          #Gyroscope
72          GyrX = memory_service.getData("Device/SubDeviceList
                /InertialSensor/GyroscopeX/Sensor/Value")
73          GyrY = memory_service.getData("Device/SubDeviceList
                /InertialSensor/GyroscopeY/Sensor/Value")
74          data.append(GyrX)
75          data.append(GyrY)
76
77          # Adding the summary of the FSR
78          LFsrTw = memory_service.getData("Device/
                SubDeviceList/LFoot/FSR/TotalWeight/Sensor/Value"
                )
79          RFsrTw = memory_service.getData("Device/
                SubDeviceList/RFoot/FSR/TotalWeight/Sensor/Value"
                )
80
```

```
81          LFcopX = memory_service.getData("Device/
               SubDeviceList/LFoot/FSR/CenterOfPressure/X/Sensor
               /Value")
82          LFcopY = memory_service.getData("Device/
               SubDeviceList/LFoot/FSR/CenterOfPressure/Y/Sensor
               /Value")
83          RFcopX = memory_service.getData("Device/
               SubDeviceList/RFoot/FSR/CenterOfPressure/X/Sensor
               /Value")
84          RFcopY = memory_service.getData("Device/
               SubDeviceList/RFoot/FSR/CenterOfPressure/Y/Sensor
               /Value")
85          data.append(LFsrTw)
86          data.append(RFsrTw)
87          data.append(LFcopX)
88          data.append(LFcopY)
89          data.append(RFcopX)
90          data.append(RFcopY)
91
92          # LeftAnkleRoll
93          PosAct = memory_service.getData("Device/
               SubDeviceList/LAnkleRoll/Position/Actuator/Value"
               )
94          PosSens = memory_service.getData("Device/
               SubDeviceList/LAnkleRoll/Position/Sensor/Value")
95          ElectrSens = memory_service.getData("Device/
               SubDeviceList/LAnkleRoll/ElectricCurrent/Sensor/
               Value")
96          data.append(PosAct)
97          data.append(PosSens)
98          data.append(ElectrSens)
99
100         # RightAnkleRoll
101         PosAct = memory_service.getData("Device/
               SubDeviceList/RAnkleRoll/Position/Actuator/Value"
               )
102         PosSens = memory_service.getData("Device/
               SubDeviceList/RAnkleRoll/Position/Sensor/Value")
103         ElectrSens = memory_service.getData("Device/
               SubDeviceList/RAnkleRoll/ElectricCurrent/Sensor/
               Value")
104         data.append(PosAct)
105         data.append(PosSens)
106         data.append(ElectrSens)
107
108         # LeftAnklePitch
```

```
109             PosAct = memory_service.getData("Device/
                    SubDeviceList/LAnklePitch/Position/Actuator/Value
                    ")
110             PosSens = memory_service.getData("Device/
                    SubDeviceList/LAnklePitch/Position/Sensor/Value")
111             ElectrSens = memory_service.getData("Device/
                    SubDeviceList/LAnklePitch/ElectricCurrent/Sensor/
                    Value")
112             data.append(PosAct)
113             data.append(PosSens)
114             data.append(ElectrSens)
115
116             # RightAnklePitch
117             PosAct = memory_service.getData("Device/
                    SubDeviceList/RAnklePitch/Position/Actuator/Value
                    ")
118             PosSens = memory_service.getData("Device/
                    SubDeviceList/RAnklePitch/Position/Sensor/Value")
119             ElectrSens = memory_service.getData("Device/
                    SubDeviceList/RAnklePitch/ElectricCurrent/Sensor/
                    Value")
120             data.append(PosAct)
121             data.append(PosSens)
122             data.append(ElectrSens)
123
124             data.append('platzhalter')
125             time.sleep(0.05)
126         return data
127
128
129     def count_files():
130         counter = 1
131         # str.zfill schreibt vor, wie lang die Zahl mit Nullen
                davor sein soll. also zfill(3) ist 3 Zahlen lang.
132         filename = 'measurement' + str(counter).zfill(3) + '.
                csv'
133
134         # Wenn das file nicht exisiert, erstelle measurement001
                .csv
135         while os.path.exists(filename):
136             counter = counter + 1
137             filename = 'measurement' + str(counter).zfill(3) +
                    '.csv'
138         create_file(filename)
139         return filename
140
```

```
141
142 def create_file(filename):
143     with open(filename, "w") as f:
144         pass
145
146
147 def main(session):
148     """
149     This example uses the getData method to use FSR sensors
        .
150     """
151     # Get the ALProxy ALMemory and ALMotion
152     from naoqi import ALProxy
153     memory_service = session.service("ALMemory")
154     motion = ALProxy("ALMotion", "nao.local", 9559)
155
156     # wake up nao
157     motion.wakeUp()
158
159     motion.moveInit()
160     motion.post.moveTo(0.85, -0.10, -0.25, [["
          MaxStepFrequency", 0.0]])
161
162     data = recordData(memory_service)
163     filename = count_files()
164
165     output = os.path.abspath(filename)
166     with open(output, "wb") as file:
167         writer = csv.writer(file, delimiter=',')
168         writer.writerow(data)
169     zeilen_aufteilen(output)
170     print "Results written to", output
171     # go back to crouch position and sleep
172     motion.rest()
173
174
175 if __name__ == "__main__":
176     parser = argparse.ArgumentParser()
177     parser.add_argument("--ip", type=str, default="
          127.0.0.1",
178     help="Robot IP address. On robot or Local Naoqi: use
          '127.0.0.1'.")
179     parser.add_argument("--port", type=int, default=9559,
180     help="Naoqi port number")
181
182     args = parser.parse_args()
```

```
183     session = qi.Session()
184     try:
185         session.connect("tcp://" + args.ip + ":" + str(args
                .port))
186     except RuntimeError:
187         print ("Can't connect to Naoqi at ip \"" + args.ip
                + "\" on port " + str(args.port) + ".\n"
188         "Please check your script arguments. Run with -h
                option for help.")
189     sys.exit(1)
190     main(session)
```

Der Programmcode 1 kann in mehrere Funktionen aufgeteilt betrachtet werden. Die Funktion `sed` ist aus (cite) entnommen und funktioniert wie die gleichnamige Funktion unter der Linux-Bash. Sie wird benötigt, um nach jedem Durchgang der Messschleife in `recordData` eine neue Zeile in die CSV Datei zu schreiben. Dies geschieht durch die Funktion `zeilen_aufteilen`. `recordData` wurde aus den Beispielen der NAO Dokumentation (cite) entnommen und angepasst, sodass am Ende jeder Zeile von `data` ein Platzhalter eingefügt wird und alle gewünschten Sensoren abgegriffen werden. Die Funktion `count_files` sorgt dafür, dass keine vorhandenen Messungen überschrieben werden und jede Messdatei eine fortlaufende Nummerierung erhält.

In der `main` Funktion werden ALMemory und ALMotion geladen, und der Gang einschließlich des Abgreifens der Sensorwerte ausgeführt. Die Ausgabe der Messwerte während dem Gang ist nur möglich durch den Präfix `post` vor `moveTo`.

Abschließend dient die letzte `if`-Abfrage zur Verbindung mit NAO, allerdings nur, wenn dieses Pythonprogramm selbst auf dem NAO liegt. Die Methode `post` sowie die Aufnahme der Sensoren während dem Lauf der Methode `moveTo` funktionieren nur lokal, deshalb ist es in diesem Fall nicht möglich, das Programm von dem eigenen Rechner aus zu starten. Mit anderen Methoden wäre eine Programmaufrufung über eine Wlan Verbindung durchaus möglich. Um Programme direkt auf dem NAO zu starten, wird eine `ssh`-Verbindung hergestellt und darüber dann `python` ausgeführt.

Literatur

[1] Jean-Paul Pelteret und Paul Steinmann. *Magneto-Active Polymers - Fabrication, characterisation, modelling and simulation at the micro- and macro-scale*. De Gruyter, 2020. DOI: https://doi.org/10.1515/9783110418576.

[2] SoftBank Robotics. *NAO - Documentation*. URL: https://developer.softbankrobotics.com/nao6/nao-documentation/nao-developer-guide (besucht am 21. 12. 2020).

[3] URL: https://spectrum.ieee.org/automaton/robotics/humanoids/aldebaran-robotics-founder-and-ceo-steps-down-softbank-appoints-new-leader (besucht am 04. 01. 2021).

[4] URL: https://www.softbankrobotics.com/emea/en/pepper (besucht am 04. 01. 2021).

[5] *NAO (NAOqi-2.1) NAO - documentation*. URL: https://developer.softbankrobotics.com/nao-naoqi-2-1/nao-documentation/nao-technical-guide/ (besucht am 04. 01. 2021).

[6] P. Shahverdi, M. J. Ansari und M. T. Masouleh. "Balance Strategy for Human Imitation by a NAO Humanoid Robot". In: *2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)*. 2017, S. 138–143. DOI: 10.1109/ICRoM.2017.8466225.

[7] A. M. Shayan u. a. "Design and Development of a Pressure-Sensitive Shoe Platform for Nao H25". In: *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*. 2019, S. 223–228. DOI: 10.1109/ICRoM48714.2019.9071802.

[8] SoftBank Robotics. *NAOqi - Developer guide*. Accessed: 2020-12-21. URL: https://developer.softbankrobotics.com/nao6/naoqi-developer-guide (besucht am 21. 12. 2020).

[9] URL: https://www.autodesk.de/products/inventor/overview?plc=INVPROSA&term=1-YEAR&support=ADVANCED&quantity=1#internal-link-what-is-inventor (besucht am 31. 01. 2021).

[10] URL: https://knowledge.autodesk.com/support/inventor/learn-explore/caas/CloudHelp/cloudhelp/2019/ENU/Inventor-Help/files/GUID-D74F47F3-FE22-44EF-85BE-7C6B1F56DCF9-htm.html (besucht am 31. 01. 2021).

[11] URL: https://ultimaker.com/de/software/ultimaker-cura (besucht am 31. 01. 2021).

[12] Version R2020 Update 3. URL: https://de.mathworks.com/discovery/what-is-matlab.html (besucht am 31. 01. 2021).

[13] URL: https://de.mathworks.com/help/stats/exploratory-data-analysis.html (besucht am 31. 01. 2021).

[14] Z. Rigbi und L. Jilken. "The response of an elastomer filled with soft ferrite to mechanical and magnetic influences". In: *J. Magn. Magn. Mater* 37(3) (Juli 1983), 267–276. DOI: 10.1016/0304-8853(83)90055-0.

[15] Z. Rigbi und J. E. Mark. "Effects of a magnetic field applied during the curing of a polymer loaded with magnetic filler". In: *J. Polym. Sci., Polym. Phys. Ed.* 23(6) (Juni 1985), 1267–1269. DOI: 10.1002/pol.1985.180230618.

[16] J. M. Ginder. "Encyclopedia of Applied Physics". In: Bd. 16. Wiley-VCH Verlag GmbH & Co KGaA, New York, New York, 1996. Kap. Rheology controlled by magnetic fields, 487–503. DOI: 10.1002/3527600434.eap402.

[17] D. Read J. E. Martin R. A. Anderson und G. Gulley. "Magnetostriction of field-structured magnetoelastomers". In: *Phys. Rev. E* 74(5).051507 (Nov. 2006). DOI: 10.1103/physreve.74.051507.

[18] G. Filipcsei Z. Varga und M. Zrínyi. "Smart composites with controlled anisotropy". In: *Polymer* 46(18) (Aug. 2005), 7779–7787. DOI: 10.1016/j.polymer.2005.03.10.

[19] G. Filipcsei Z. Varga und M. Zrínyi. "Magnetic field sensitive functional elastomers with tuneable elastic modulus". In: *Polymer* 47(1) (Jan. 2006), 227–233. DOI: 10.1016/j.polymer.2005.10.139.

[20] URL: https://www.supermagnete.de/magnete-zum-anschrauben/quadermagnet-40-x-40-x-4mm_CS-Q-40-40-04-N (besucht am 06. 01. 2021).