Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorthesis

Magnus Bruhn

## Konzeption, Entwicklung und Implementierung einer adaptiven, konfigurierbaren Motorsteuerung

*Fakultät Technik und Informatik*

*Faculty of Engineering and Computer Science*

Magnus Bruhn

# Konzeption, Entwicklung und Implementierung einer adaptiven, konfigurierbaren Motorsteuerung

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Mechatronik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Klaus Jünemann
Zweitgutachter : Prof. Dr. Frank Kirchner

Abgegeben am March 23, 2016

**Magnus Bruhn**

## Thema der Bachelorthesis

Konzeption, Entwicklung und Implementierung einer adaptiven, konfigurierbaren Motorsteuerung

## Stichworte

Regelung, Konzeption, Konstruktion, PID-Regler, Raspberry Pi, NDLCom, Robotertestanlage, C++

## Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Steuerung und Regelung eines Laufbandes als Robotertestanlage. Hierzu werden unter anderem Softwarekomponenten entwickelt, eine Reglersynthese durchgeführt und passende Sensorik zum Erfassen der Geschwindigkeit des Laufbandes ausgewält. Die einzelnen Teile des Arbeit wurden zunächst konzeptionell erarbeitet und dann umgesetzt. Die Software wurde in C++ objektorientiert programmiert. Außerdem war die Konstruktion einer mechanischen Anbindung des Sensors an das Laufband notwendig, die unter Einhaltung gängiger Konvertionen und Konstruktionsweise durchgeführt wurde. Mit Tests wurde schließlich die Funktion des Systems geprüft.

**Magnus Bruhn**

## Title of the paper

Conception, development and implementation of an adaptive, configurable motor control unit

## Keywords

Controlling, conception, construction, PID-Controller, Raspberry Pi, NDLCom, robotic test unit, C++

## Abstract

This report works on the control of a treadmill as a robotic test unit. In this process, software components were developed, a controller synthesis was performed and sensors were choosen, which were capable of velocity capturing on the treadmill. Before implementing, the different workpackages were conceptually designed. The Software was written object-oriented in C++. Also the constuction of a mechanical connection between sensor and treadmill was necessary. It was designed in regard of established conventions and constructional advises. Finally, the system's function was checked with tests.

# Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mir im Laufe meiner Bachelorarbeit mit Rat und Tat zur Seite gestanden haben.

Mein besonderer Dank richtet sich an meinem Betreuer Hendrik Hanff, der mich in allen Phasen der Arbeit hilfreich beraten hat und immer für Fragen offen war.

Außerdem möchte ich mich ganz herzlich bei Marc Manz und Martin Zenzes bedanken, die mir insbesondere bei der Umsetzung der Arbeit geholfen haben.

Ein weiterer besonderer Dank geht an meinen Erstprüfer und Ansprechpartner Klaus Jünemann, der mir mit hilfreichen Tipps und Anregungen geholfen hat sowie meinem Zweitprüfer Frank Kirchner für seine fachmännische Beurteilung meiner Arbeit.

# Inhaltsverzeichnis

# Tabellenverzeichnis

# Abbildungsverzeichnis

# 1. Einleitung

Tests im Labor stellen im Entwicklungs- und Erprobungsprozess von autonomen Systemen einen notwendigen Schritt dar. Sie fungieren als Bindeglied zwischen reinen Funktionstests und Tests in einer dem Einsatzgebiet ähnlichen Umgebung. In dieser Arbeit geht es um die Konzeption und Umsetzung einer Regelung, sowie einer grafischen Benutzeroberfläche, mit der die Nutzung eines Laufbandes als Robotertestanlage ermöglicht werden soll.

Bei der Umsetzung dieser Aufgaben wird unter anderem auf Kenntnisse aus den Gebieten der Informatik, Regelungstechnik und Konstruktion zurückgegriffen. Es wird mit Reglersynthese, Softwareentwurf und Konstruktiven Maßnahmen in allgemein gängiger Form gearbeitet. Als Hilfsmittel kommen unter anderem CAD-Programme, Konstruktionshinweise, Tabellenbücher und Simulationssoftware zum Einsatz.

## 1.1. Motivation

Abbildung 1.1.: DFKI - Mantis [Quelle: DFKI GmbH]

Der in Abbildung 1.1 zu sehende Roboter ist der 6-beinige Laufroboter Mantis. Er ist für „Fortbewegung und Manipulation in unstrukturierten und steilen Umgebungen auf den Oberflächen extraterrestrischer Himmelskörper"[DFKI (2015)] gedacht. Die Abmaße von Mantis von etwa 2 x 2 m verhindern die Nutzung kommerzieller Laufbandsysteme als Testanlage. Deshalb wurde es notwendig, wie in den meisten Fällen bei der Roboterentwicklung, ein speziell an die vorliegenden Anforderungen angepasstes Laufband anzuschaffen.

Um dieses Laufband, welches mit Motor geliefert wurde, nun für die Erprobung von Mantis nutzen zu können war es noch notwendig, eine Regelung und ein Benutzeroberfläche zu schaffen.

## 1.2. Zielsetzung

Das Ziel dieser Arbeit ist es ein passendes, einfach zu benutzendes Testumfeld für Mantis zu schaffen. Das ist besonders auf die Funktionsfähigkeit der Regelung sowie die Intuitive

Bedienbarkeit der grafischen Benutzeroberfläche zu beziehen. Das Ziel ist als erreicht zu betrachten, wenn die Erfüllung der festgehaltenen Anforderungen nach dem Konzeptions- und Implementierungsprozess durch passende Tests nachgewiesen wird.

## 1.3. Aufbau der Arbeit

Im zweiten Kapitel werden zunächst einige Grundlagen der Regelungstechnik erläutert. Dies soll dazu dienen dem Leser einen reibungslosen Einstieg in die Arbeit zu ermöglichen.

Danach wird in Kapitel drei auf den aktuellen Stand der Technik in den Bereichen Robotertestanlagen und PID-Regler eingegangen und ein kurzer Überblick vergleichbarer Testanlagen im Bereich Test autonomer Systeme gegeben.

In Kapitel vier wird die Konzeption dargelegt. Es werden zunächst die gegebenen Anforderungen festgehalten und im weiteren Verlauf eine Zuordnung in Teilbereiche vorgenommen. In diesen Teilbereichen werden dann geeignete Konzepte zum Lösen der Aufgabenstellung erarbeitet.

Nachdem die theoretischen Überlegungen abgehandelt sind, wird in Kapitel fünf als erstes die Umsetzung der mechanischen Verbingung zwischen Laufband und Sensor skizziert. Außerdem werden die verwendeten Konstruktionsmethoden vorgestellt.

Auf die Umsetzung der Konstruktion folgt in Kapitel sechs die Umsetzung der Software bezogenen Aufgabenteile. Hier wird einerseits die verwendete Software noch einmal genauer vorgestellt, andererseits die einzelnen Bereiche zu entwickelnder Software näher betrachtet. Hierzu sind der Entwurf eines Reglers, das Sicherstellen der Kommunikation zwischen der Benutzeroberfläche und der Steuerung und dem Entwurf der grafischen Benutzeroberfläche zu zählen.

Im siebten Kapitel wird auf die Entwicklung und Durchführung von Tests des Gesamtsystems eingegangen. Mit diesen Tests wird die Funktion der Anlage, sowie die korrekte Umsetzung der Anforderungen sichergestellt.

Im abschließenden achten Kapitel wird eine Zusammenfassung der im Lauf der Arbeit gewonnen Erkenntnisse festgehalten und ein Ausblick auf zukünftig vorstellbare Weiterentwicklungen gegeben.

## 1.4. Kontext der Arbeit

Diese Arbeit wird im Rahmen des LIMES Projektes am DFKI durchgeführt. Die Entwicklung des eingangs erwähnten Weltraumdemonstrator Mantis stellt eine der Kernaufgaben des Projektes dar. Es wurden bis jetzt schon verschiedene Funktionstests von Aktoren, Sensoren und Software durchgeführt, es fehlte jedoch an einer passenden Testeinrichtung um das Gesamtsystem in einem geeigneten Rahmen zu erproben. Da diese nun vorhanden ist, gilt es jetzt die Betriebsbereitschaft herzustellen.

# 2. Grundlagen der Regelungstechnik

In diesem Kapitel werden einige der für diese Arbeit relevanten Grundlagen der Regelungstechnik erläutert. Der Fokus wird hierbei auf das Einstellen der Reglerparameter gelegt, da diese den Großteil, der hier mit Regelungstechnik in Zusammenhang stehenden Arbeit betreffen. Da der Anwendungsfall, die Steuerung eines Laufbandes als Roboter Testanlage, eine, aus regelungstechnischer Sicht, relativ geringe Komplexität darstellt, wird auch kein übermäßig komplexes Verfahren zum Bestimmen der Reglerparameter eingesetzt. Dies würde einen, dem potenziellen Resultat unverhältnismäßig gegenüberstehenden Aufwand bedeuten.

## 2.1. Fachtermini

Hier werden einige Fachbegriffe der Regelungstechnik erklärt und gegebenenfalls übersetzt, falls sie üblicherweise auf englisch verwendet werden.

- **Regler:** Ein Regler bestimmt automatisch den Eingangswert des zu regelnden Systems, mit dem nach Möglichkeit die Differenz zwischen Soll- und Istwert des zu regelnden Systems auf ein Minimum reduziert werden soll.
- **Regelstrecke:** Eine Regelstrecke ist das mit einem Regler zu beeinflussende System.
- **P-Anteil:** Der Proportional-Anteil sorgt mit dem Faktor $K_P$ für die Stärke der Regelung. Eine große Regeldifferenz erfordert ein großes $K_P$. [Lunze (2010)]
- **I-Anteil:** Der Integral-Anteil verhindert mit dem Faktor $K_I$ bei sprungförmigen Führungs- und Stellsignalen eine bleibende Regelabweichung. [Lunze (2010)]
- **D-Anteil:** Der Differenzial-Anteil reagiert mit dem Faktor $K_D$ nur auf Veränderungen der Regelabweichung. Er führt dazu, dass der Regler bereits dann mit einer großen Stellgröße reagiert, wenn die Regelabweichung stark zunimmt, selbst wenn sie noch keinen großen Wert angenommen hat. [Lunze (2010)]
- **Sprungantwort:** Die Sprungantwort ist das Ausgangssignal eines Systems, bei dem am Eingang eine Sprungfunktion angelegt wird.

## 2.2. Auslegungsmöglichkeiten

Wie in Kapitel 2 erwähnt wird in dieser Arbeit ein Einstellverfahren mit geringer Komplexität gesucht. Im Folgenden werden verschiedene Verfahren dieser Kategorie vorgestellt und ein passendes ausgewählt. Auf Einstellverfahren mit höherer Komplexität wird in Kapitel 3.2 näher eingegangen.

### 2.2.1. Empirische Dimensionierung

Oft werden Regelkreise durch einfaches Ausprobieren von Reglerparametern eingestellt. Ausgehend von Erfahrungswerten werden Reglerparameter gewählt und variiert. Diese Methode ist allerdings nur zur Dimensionierung einfacher Systeme geeignet. [Bate (2009)]

### 2.2.2. Dimensionierung nach Einstellregeln

Es wird bei der Dimensionierung nach Einstellregeln zunächst unterschieden, ob die Art der Regelstrecke, sowie ihre Streckenparameter bekannt sind oder nicht. Es gibt verschiedene Verfahren, einige zum Identifizieren der Streckenparameter und andere zum Bestimmen der Reglerparameter. [Bate (2009)]

#### Identifikation der Streckenparameter

Sollten die Streckenparameter Streckenverstärkung $K_S$, Verzugszeit $T_u$, Ausgleichzeit $T_g$ bekannt sein, können die Reglerparameter direkt nach Einstellregeln bestimmt werden. Sind die Streckenparameter nicht bekannt, ist es zunächst notwendig diese mit einem der folgenden Verfahren zu bestimmen. [Bate (2009)]

**Wendetangentenmethode** Ein sehr weit verbreitetes Verfahren ist das Wendetangentenverfahren. Bei diesem Verfahren wird die Sprungantwort aufgenommen um $T_u$ sowie $T_g$, mit dem Anlegen der Wendetangente, wie in Abbildung 2.1, zu ermitteln.

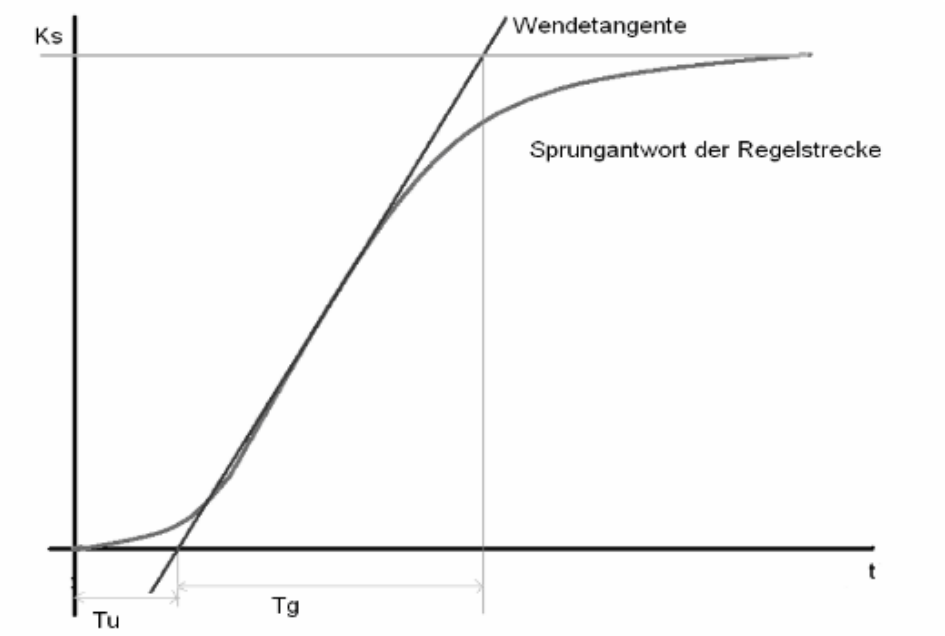



Abbildung 2.1.: Wendetangentenverfahren [Bate (2009)]

**P-T1-Tt-Approximation** Hier wird die Strecke durch eine Reihenschaltung aus einem P-$T_1$-Glied mit der Zeitkonstante $T_1$ und einem Totzeitglied mit Totzeit $T_t$ modelliert. Die Zeitkonstanten werden dann so gewählt, dass die Sprungantwort gut angenähert wird. [Bate (2009)] Es ist auch möglich Parameter für diese Betrachtung aus der Sprungantwort mit dem Wendetangentenverfahren abzuleiten: $K_p = K_p$, $T_1 = T_g$, $T_t = T_u$. [L.Billmann (2010)]

**T-Summenverfahren** In diesem Verfahren wird die Summenzeitkonstante $T_{\sum}$ als Kenngröße für die Schnelligkeit der Strecke verwendet. Diese Konstante ist als die Differenz der Summe aller verzögernder Zeitkonstanten zuzüglich der Totzeit zur Summe der differenzierenden Zeitkonstanten definiert. [Bate (2009)]

**Betragsoptimum (P-$T_n$-Approximation)** Hierbei wird die Regelstrecke als P-$T_n$-Glied mit n gleichen Zeitkonstanten $T_1$ betrachtet. Es wird zunächst aus der Sprungantwort ermittelt, zu welchen Zeiten die Regelgröße 10%, 50% und 90% ihres Endwertes erreicht hat. Nun muss $\mu = \frac{t_{10\%}}{t_{90\%}}$ berechnet werden und die Parameter $\alpha_{10}$, $\alpha_{50}$, $\alpha_{90}$ sowie die Ordnung der Strecke n aus einer geeigneten Tabelle abgelesen werden. [Bate (2009)]

### Reglerparameter

Sobald die Streckenparameter bekannt sind, kann damit fortgefahren werden die Reglerparameter anhand einer Einstellregel zu ermitteln. Diese Werte können über die Zusammenhänge $K_P = K_R$, $K_I = K_R/T_N$ und $K_D = K_R*T_V$ in Proportionalverstärkung, Integral- oder Differenzialzeitkonstante umgerechnet werden. Im Folgenden werden einige Einstellregeln vorgestellt.

**Ziegler/Nichols** Diese Tabelle zeigt die Einstellregeln nach Ziegler/Nichols.

| Regler | $K_R$ | $T_N$ | $T_V$ |
|---|---|---|---|
| P | $(1 / K_S) * (T_1 / T_t)$ | | |
| PI | $(0{,}9 / K_S) * (T_1 / T_t)$ | $3{,}3 * T_t$ | |
| PID | $(1{,}2 / K_S) * (T_1 / T_t)$ | $2 * T_t$ | $0{,}5 * T_t$ |

Tabelle 2.1.: Einstellregeln nach Ziegler/Nichols [Waste (2005)]

**Chien/Hrones und Reswick** Diese Tabelle zeigt die Einstellregeln nach Chien/Hrones und Reswick.

| Regler | | Aperiodischer Verlauf | | 20% Überschwinger | |
|---|---|---|---|---|---|
| P | $K_R$ | $(0{,}3/K_S)*(T_g/T_u)$ | $(0{,}3/K_S)*(T_g/T_u)$ | $(0{,}7/K_S)*(T_g/T_u)$ | $(0{,}7/K_S)*(T_g/T_u)$ |
| PI | $K_R$ | $(0{,}6/K_S)*(T_g/T_u)$ | $(0{,}35/K_S)*(T_g/T_u)$ | $(0{,}7/K_S)*(T_g/T_u)$ | $(0{,}6/K_S) * (T_g/T_u)$ |
| | $T_N$ | $4*T_u$ | $1{,}2*T_u$ | $2{,}3*T_u$ | $T_g$ |
| PID | $K_R$ | $(0{,}95/K_s)*(T_g/T_u)$ | $(0{,}6/K_s)*(T_g/T_u)$ | $(1{,}2/K_s)*(T_g/T_u)$ | $(0{,}95/K_s)*(T_g/T_u)$ |
| | $T_N$ | $2{,}4*T_u$ | $T_g$ | $2*T_u$ | $1{,}35*T_g$ |
| | $T_V$ | $0{,}42*T_u$ | $0{,}5*T_u$ | $0{,}42*T_u$ | $0{,}47*T_u$ |

Tabelle 2.2.: Einstellregeln nach Chien/Hrones und Reswick [Bate (2009)]

**Oppelt** Diese Tabelle zeigt die Einstellregeln nach Oppelt.

| Regler | $K_R$ | $T_N$ | $T_V$ |
|---|---|---|---|
| P | $(1/K_S)*(T_g/T_u)$ | | |
| PI | $(0{,}8/K_S)*(T_g/T_u)$ | $3*T_u$ | |
| PID | $(1{,}2/K_S)*(T_g/T_u)$ | $2*T_u$ | $0{,}42*T_u$ |

Tabelle 2.3.: Einstellregeln nach Oppelt [Bate (2009)]

**Rosenberg** Diese Tabelle zeigt die Einstellregeln nach Rosenberg.

| Regler | $K_R$ | $T_N$ | $T_V$ |
|---|---|---|---|
| P | $(1/K_S)*(T_g/T_u)$ | | |
| PI | $(0{,}91/K_S)*(T_g/T_u)$ | $3{,}3*T_u$ | |
| PID | $(1{,}2/K_S)*(T_g/T_u)$ | $2*T_u$ | $0{,}44*T_u$ |

Tabelle 2.4.: Einstellregeln nach Rosenberg [Bate (2009)]

**T-Summenverfahren** Diese Tabelle zeigt die Einstellregeln nach T-Summenverfahren: für normale Einstelung:

| Regler | $K_R$ | $T_N$ | $T_V$ |
|---|---|---|---|
| P | $(1/K_S)$ | | |
| PI | $(0{,}5/K_S)$ | $0{,}5*T_\sum$ | |
| PD | $(1/K_S)$ | | $0{,}33*T_\sum$ |
| PID | $(1/K_S)$ | $0{,}6*T_\sum$ | $0{,}167*T_\sum$ |

Tabelle 2.5.: Einstellregeln nach T-Summenverfahren normal [Bate (2009)]

für schnelle Einstellung:

| Regler | $K_R$ | $T_N$ | $T_V$ |
|---|---|---|---|
| PI | $(1/K_S)$ | $0{,}7*T_\sum$ | |
| PID | $(2/K_S)$ | $0{,}8*T_\sum$ | $0{,}194*T_\sum$ |

Tabelle 2.6.: Einstellregeln nach T-Summenverfahren schnell [Bate (2009)]

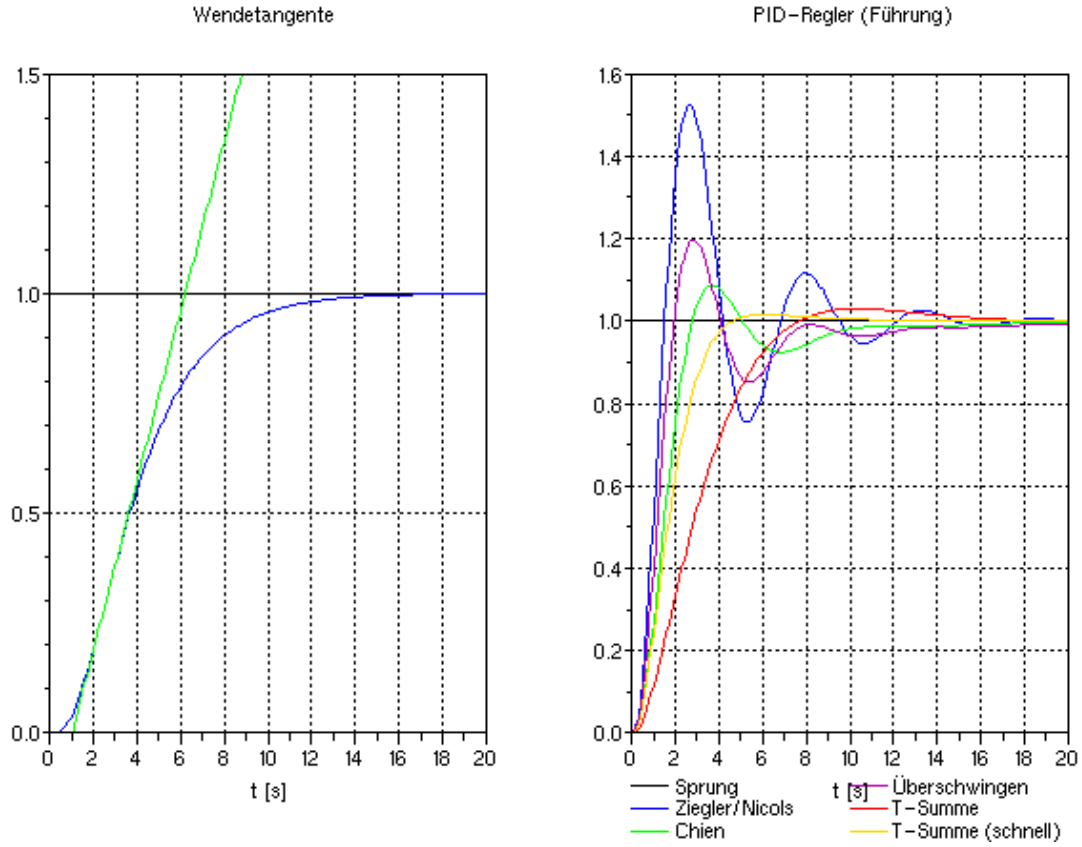



Abbildung 2.2.: Vergleich von einigen Einstellregeln [Wikipedia]

Anhand von Abbildung 2.2 werden die unterschiedlichen Regelergebnisse zwischen einigen der Einstellverfahren gezeigt werden.

# 3. Stand der Technik

Im Folgenden wird der aktuelle Stand relevanter technischer und wissenschaftlicher Zusammenhänge aufgeführt. Testanlagen für Robotersysteme sind im Allgemeinen keine „Off-the-Shelf" Produkte, die von der Industrie erworben werden können. Dies ist unter anderem darauf zurückzuführen, dass die zu testenden Robotersysteme zumeist sehr spezielle Anforderungen haben und somit darauf zugeschnitte Testumgebungen benötigen. Die, aus dieser Tatsache resultierenden, Spezialanfertigungen und deren Spezifikationen werden üblicherweise nicht öffentlich zugänglich gemacht. Dies erschwert das Analysieren dieser Systeme um sie als Stand der Technik heranzuziehen.

## 3.1. Robotik Testanlagen

Es gab am DFKI bereits Laufbänder als Teststände für Laufbewegungen von Robotern, welche hier zum Stand der Technik herangezogen werden. Das LegLab verfügt über ein Laufband, sowie Einrichtungen zur Messung von Winkelposition der Gelenke, des Stromverbrauchs, Geschwindigkeit des Laufbandes und ähnliche.

Das Laufband verfügt über eine nutzbare Lauffläche von ca. 150 cm Länge und ca. 50 cm Breite, sowie einen stufenlos regelbaren Geschwindigkeitsbereich von 0.14 $\frac{m}{s}$ - 0.97 $\frac{m}{s}$. Geregelt wird der Teststand von einem Atmel Microcontroller mit USB-Anschluss, verschiedenen Elektronikbausteinen und einem Messrad-Encoder. [DFKI (2012)]

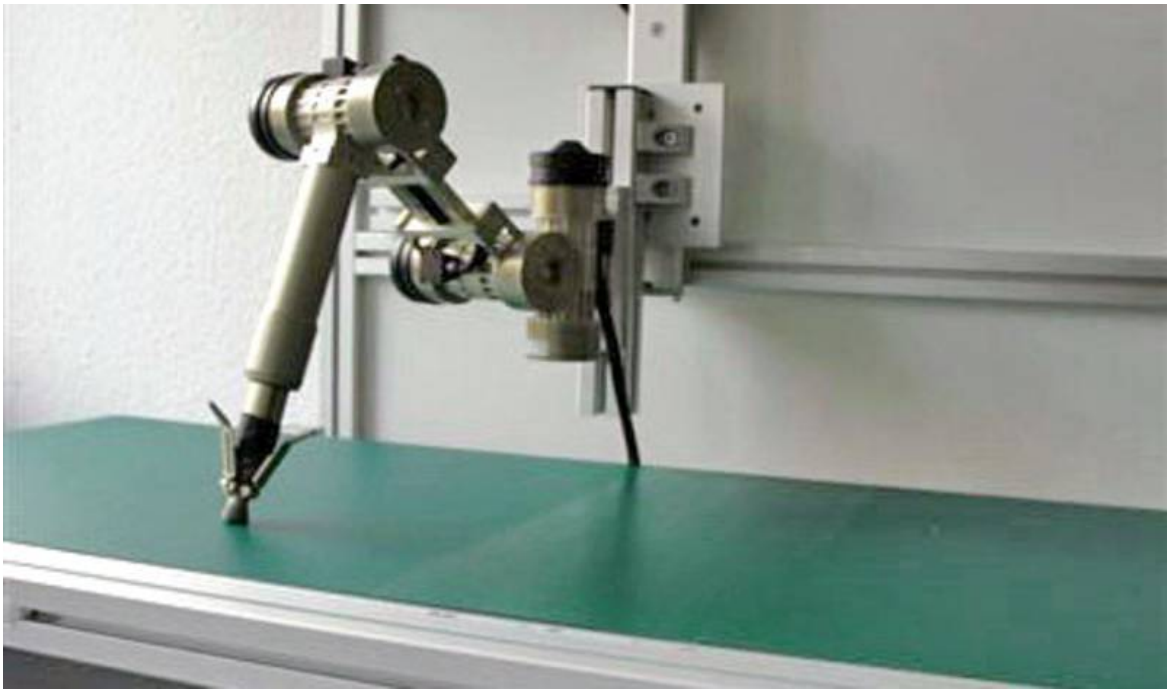


Abbildung 3.1.: DFKI - LegLab [DFKI (2012)]

## 3.2. PID-Regler

Der PID-Regler ist einer der ältesten und meistgenutzten Regler überhaupt. Das Konzept und die Einstellregeln hierfür existieren schon seit über 70 Jahren [Ziegler und Nichols (1942)]. Dieses und andere empirische Einstellverfahren werden zwar nach wie vor genutzt, komplexe Systeme und variierende Parameter machen es jedoch teilweise notwendig einen anderen, moderneren Ansatz zu wählen.

Zu den moderneren Ansätzen zählen direkt adaptive Systeme, mit Kontrollalgorithmen wie iterativem Feedbacktuning (iterative feedback tuning), und der Methode der periodischen Zielungleichheit (method of recurrent objective inequalities). Außerdem gibt es indirekte Adaptionsalgorithmen wovon viele eine Weiterentwicklung von automatisierten Tuningalgorithmen darstellen. Zuletzt gibt es noch frequenzadaptive PID-Regler, wobei das Identifizieren der Parameter durch die Auswahl von Frequenzen und Amplituden für unterschiedliche Testsignale vorgenommen werden. [Alexandrov und Palenov (2014)]

# 4. Konzeption

Die Konzeption war der erste Schritt beim Durchführen des Projektes. Hier werden im ersten Schritt die Anforderungen aufgeführt und daraus Aufgaben abstrahiert. Auf der Grundlage dieser Aufgaben wird als nächstes ein Gesamtkonzept vorgestellt, das eine Lösung der Aufgabe als Ganzes umreißt. Im Gesamtkonzept werden auch schon Lösungansätze für Teilaufgaben genannt, diese aber noch nicht genauer ausgeführt. Dies geschieht in den Unterkapiteln Reglerkonezpt, Sensorkonzept und Interfacekonzept. Abschließend wird noch die Verdrahtung des Systems in Form eines Verkabelungsplanes vorgestellt.

## 4.1. Anforderungen

Die Anforderungen an eine Testeinrichtung für autonome Robotersysteme sind naturgemäß vielzählig und hoch. Da die zu erprobenden Systeme hochintegriert und komplex sind ist es von größter Wichtigkeit eine geeignete Testumgebung zu schaffen, um möglichst effizient, exakte Ergebnisse erzeugen zu können.

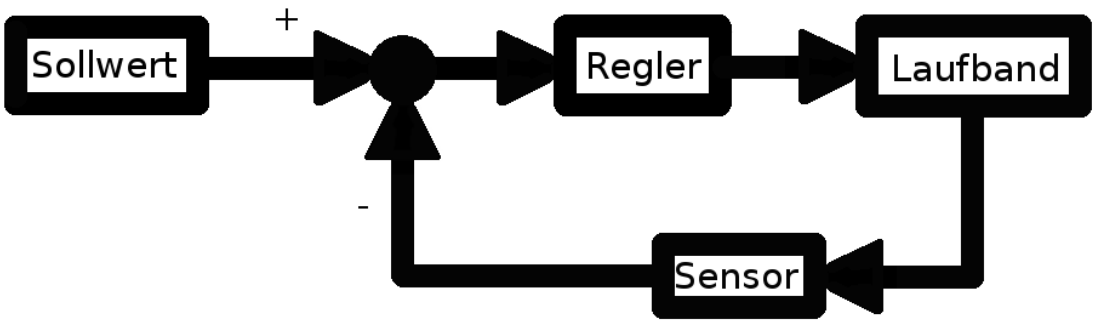



Abbildung 4.1.: Blockschaltbild des Signalflusses des geregelten Systems

Die Anforderungen lauten:

- Entwicklung eines Gesamtkonzepts zur Laufbandregelung
- Stufenlose Geschwindigkeitsregelung zwischen minimaler und maximaler Laufbandgeschwindigkeit
- Bestimmung eines geeigneten Reglers zur Kontrolle des Laufbandes
- Implementierung des Reglers
- Bestimmung der Strecken- und Reglerparameter
- Auswahl geeigneter Sensorik
- Auslesen der Sensorik
- Konzeption und Konstruktion von mechanischen Komponenten, wie z.B. der mechanischen Anbindung von Sensoren an das Laufband
- Programmieren einer GUI zur Ansteuerung des Reglers unter Verwendung einer ”Institutsvorlage” in C++

## 4.2. Ausgangslage

Das Robotic Innovation Center (RIC) des Deutsche Forschungsinstitut für künstliche Intelligenz (DFKI) in Bremen ist führend in verschiedenen Bereichen der Robotik. Am DFKI werden zum Beispiel Unterwasserroboter zur autonomen Inspektion von Pipelines und Pipelineventilen entwickelt, andere zum Kartografieren von Schwarzen Rauchern in der Tiefsee als Vorbereitung für einen potenziellen Rohstoffabbau. Das DFKI ist aber auch in der Weltraumrobotik tätig. Es werden Roboter zum Transport von anderen Robotern über große Strecken auf anderen Himmelskörpern entwickelt, sowie Systeme zur Exploration und Manipulation auf Himmelskörpern.

Abbildung 4.2.: Space Explorationshalle [DFKI GmbH/Florian Cordes]

Das in der Einleitung erwähnte Laufband steht in der Space Exploration Hall des RIC. Siehe Abbildung 4.2. Der Motor, eine Ansteuerung für diesen und damit einhergehende Sicherheitsfunktionen, sowie die Energieversorgung waren bereits vorhanden.

In einem vorausgegangenen Projekt wurde außerdem bereits die manuelle Geschwindigkeitssteuerung per Steuerflasche implementiert.

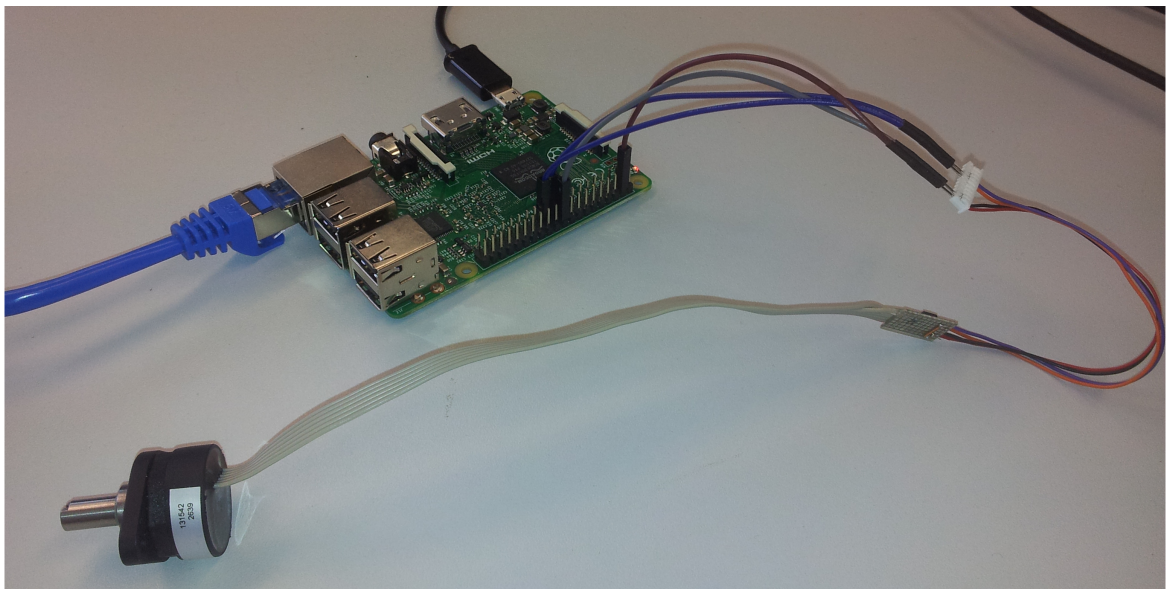


Abbildung 4.3.: Raspberry Pi2 mit Sensor

Desweiteren wurde der Einplatinencomputer bereits im Rahmen des vorausgegangenen Projektes ausgewählt und angeschafft, sodass auf die Auswahl kein Einfluss mehr genommen werden konnte. Der ausgewählte Computer ist ein Raspberry Pi2 mit einer Raspbian Wheezy Linux Distibution als Betriebssystem. Die Kommunikation zwischen Pi und Motorsteuerung wurde ebenfalls im Vorweg mit einem RS-485 Shield auf dem Pi sichergestellt.

## 4.3. Gesamtkonzept

Die Rahmenbedingungen für das Gesamtkonzept sind die Aufgabenstellung in Kapitel 1.1 und die gegebenen Anforderungen in Kapitel 4.1.

Dieses Kapitel soll einen Überblick darüber geben, in welcher Form die einzelnen Komponenten interagieren.

Im nachfolgenden Blockschaltbild 4.4 ist zu sehen, dass der Sensor am Laufband befestigt ist und eine elektrische Verbindung zum Raspberry Pi hat. Außerdem ist erkennbar, dass die Motorsteuerung direkt mit dem Raspberry Pi verbunden ist. Der Raspberry Pi selbst ist mit dem lokalen Netzwerk verbunden, worüber der Benutzer eine Remoteverbindung zu ihm aufbauen kann. Er kann über diese Netzwerkverbindung außerdem das Laufband, mit

der zur Verfügung gestellten grafischen Benutzeroberfläche, steuern.

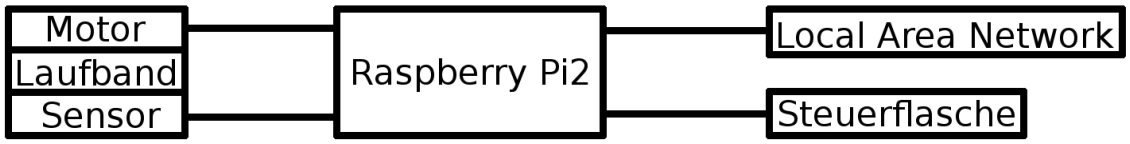



Abbildung 4.4.: Blockschaltbild des Gesamtkonzeptes

## 4.4. Reglerkonzept

Die gängigen Formen von implementierten Reglern sind P-, PI-, PD- und PID-Regler. Die drei erstgenannten Reglertypen können sehr einfach mit einem PID-Regler umgesetzt werden, in dem einzelne Zeitkonstanten auf 0 gesetzt werden. Deshalb bietet es sich an einen PID-Regler zu nutzen, da dieser eine große Flexibilität bietet und der Aufwand beim Implementieren keinen großen Mehraufwand bedeutet.

## 4.5. Sensorkonzept

Die zur Regelung des Laufbandes interessante physikalische Größe, die zur Bildung des aktuellen Regelfehlers benötigt wird, ist die Laufbandgeschwindigkeit. Für das Aufnehmen dieser Größe gibt es unterschiedliche Möglichkeiten. Es wäre zum Beispiel denkbar ein Messrad auf dem Laufband anzubringen, was aber keine besonders elegante Lösung darstellt. Besser und kompakter ist es eine Messwelle an einer der Wellen des Laufbandes zu befestigen. An dieser Welle kann dann ein Drehgeber befestigt werden, dessen Ausgabewerte können dann in den Regler gespeist werden.

Um größere Messungenauigkeiten durch Montagetoleranzen zu vermeiden, bietet es sich an einen Sensor auszuwählen, der über einen integrierten Messwertgeber und ein montagefertiges Gehäuse verfügt. Aus dieser Kategorie würde zum Beispiel der CTS Electronics Series 288 Rotary Encoder in Frage kommen. Er verfügt über eine 4-Bit Absolutwert Kodierung und eine mechanische Bauweise. Es gibt jedoch einen Sensor, der hier am Institut bekannt und verfügbar ist, was den Zeit- und Entwicklungsaufwand deutlich reduziert. Er verfügt mit seiner, vom Kommunikationsprotokoll abhängigen, 12- beziehungsweise 14-Bit Absolutwert

Kodierung auch über eine deutlich höhere Auflösung. Dies kommt dem Projekt, ob der verhältnismäßig geringen Drehgeschwindigkeiten entgegen. Außerdem handelt es sich bei dem Megatron ENS22F um einen Halleffekt Sensor, der ohne größeren Verschleiß betrieben werden kann, da der Magnet, der als Messwertgeber genutzt wird fest kugelgelagert auf einer Welle befestigt ist. Die Lebensdauer des CTS Sensors ist begrenzt. Der deutlich höhere Preis des Megatron Sensors wird aufgrund der immensen Vorteile als legitim eingestuft.

## 4.6. Interfacekonzept

Das Interface, in Form einer grafischen Benutzeroberfläche, ist die Schnittstelle zwischen dem Benutzer und der Regelung. Sie soll ihm eine intuitive Benutzung des Laufbandes zur Verfügung stellen. Hierbei sollte sichergestellt werden, dass, neben dem Geschwindigkeitssollwert auch die Werte für Proportionalverstärkung, Integrator- und Differentiatorzeitkonstante eingestellt werden können. Es ist außerdem sinnvoll den aktuellen Geschwindigkeitswert des Laufbandes in der GUI anzuzeigen. Die intitutsintern gängige CommonGUI stellt eine passende Möglichkeit dar, potentiellen Nutzern diese Funktion in gewohnter Nutzungsumgebung zugänglich zu machen.

## 4.7. Verkabelungsplan

Der folgende Verkabelungsplan 4.5 zeigt die genauen elektrischen Verbindungen zwischen den einzelnen Komponenten. Solange es sich nicht um Standardverbindungen handelt (zum Beispiel RJ45 Patchverbindung) werden die Verbindungskabel, nach Institusstandard mit Farbe und Nutzungsart der einzelnen Litzen gekennzeichnet.

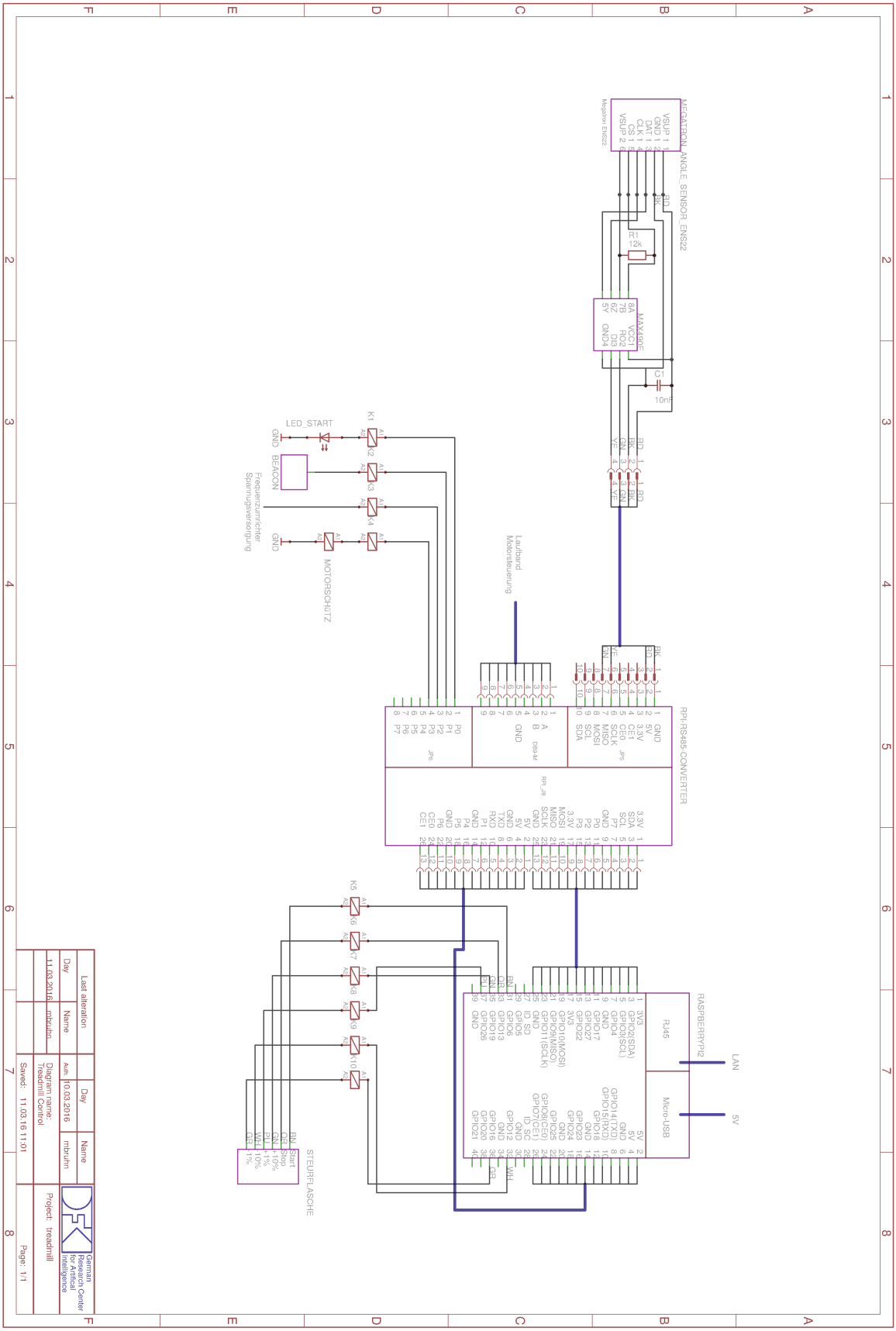



Abbildung 4.5.: Verkabelungsplan der Steuerungs- und Regelungskomponenten

## 4.8. Mechanisches Konzept

Im folgenden Kapitel 5 wird erläutert, wie die Verbindung zwischen Sensor und Laufband realisiert wurde. Es wird dargelegt, warum und auf welcher Grundlage die einzelnen Teile so konstruiert wurden.

# 5. Konstruktion

Die Verbindung zwischen Laufband und Sensor konnte nicht ohne die Verwendung eigens konstruierter Teile hergestellt werden. Der ausgewählte Megatron Sensor verfügt bereits über eine Welle, was den Konstruktionsaufwand auf ein geringes Maß reduziert hat. So musste lediglich eine mechanische Verbindung hergestellt werden und nicht noch eine penible Ausrichtung von Sensor und Messwertgeber vorgenommen werden.

Zunächst galt es die Art der Verbindung festzulegen. Es wurde eine Welle-Kupplung-Welle Verbindung ausgewählt, bei der eine passende Welle an der vorhandenen Welle des Laufbandes befestigt wird. Dies verspricht eine zweckmäßige, kompakte Bauform, bei einem geringen Aufwand an zu fertigenden Teilen.

Der Sensor verfügt über eine Welle mit dem Durchmesser 6 mm, wodurch es sinnvoll ist den Durchmesser der zweiten Welle auch auf 6 mm festzulegen. Die in Abbildung A.1 gezeigte Welle wurde unter dieser Annahme konstruiert. Außerdem ist ein M6 Gewinde zum Einschrauben in die Laufbandwelle mit einem dazugehörigen Gewindefreistich nach Konstruktionshinweisen [Bode (2013)] vorgesehen.

Als Kupplung wurde eine drehstarre Kupplung HF mit Sackloch und Klemmnarbe von Mädler aus Abbildung C.1 ausgewählt. Die Kupplung ist zur Verbindung zweier 6 mm Wellen geeignet und bietet einen Versatzausgleich von 0,5 Grad und 0,2 mm radial. Sie kann ein Drehmoment von 1,7 Nm übertragen, wobei das tatsächlich übertragene deutlich geringer sein sollte. Ein Moment tritt lediglich durch den Widerstand der Sensorwellenlagerung auf.

Die Befestigung des Sensors bedurfte des größten konstruktiven Aufwandes. Aufgrund der Bauform des Laufbandes bietet es sich an, den Sensor am Gehäuse des Getriebes zu befestigen. Das Gehäuse besteht unter anderem aus zwei Abdeckungen in Form von gebogenen Blechen, die leicht demontierbar sind. Das eine der Bleche liegt in der Flucht der Laufbandwelle. Es gilt also eine Verbindung zwischen dem Sensor und diesem Blech herzustellen, die genug Raum für die Kupplung lässt. Unter Berücksichtigung der Fertigungskomplexität wurde als Grundform ein rotierter Körper gewählt, da dieser mit überschaubarem Zeit- und Arbeitsaufwand gefertigt werden kann.

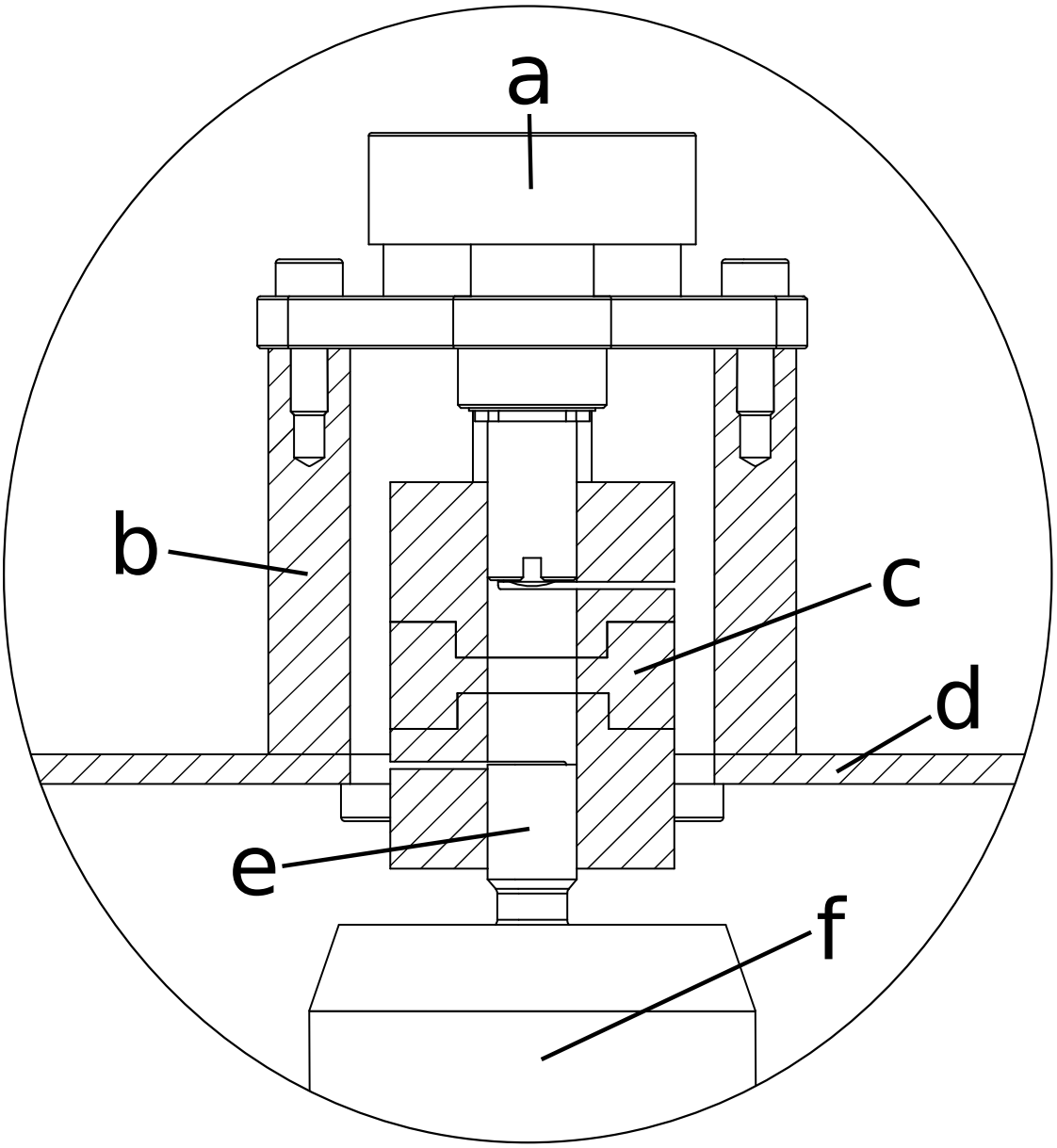



Abbildung 5.1.: Zusammenbauzeichnung der Sensor-Laufband Verbindung. a: Sensor, b: Buchse, c: Kupplung, d: Adapterwelle, e: Gehäuse des Laufbandes, f: Welle des Laufbandes

Unter demselben Aspekt wurde die Aussparung zum Montieren der Kupplung an der Sensorwelle als Form konstruiert, die mit einer Fräse problemlos herzustellen ist. Die Maße der hier konstruierten Buchse orientieren sich an den Maßen der Kupplung, des Sensors und

den im Folgenden ausgelegten Schrauben inklusive Reserven und Toleranzen.

Die Auslegung der Schrauben basiert primär auf dem Gewicht der verbauten Komponenten, da diese im ungünstigsten anzunehmenden, wenn auch äußerst unwahrscheinlichen Fall, voll auf der Sensor-Gehäuse Befestigung lasten würden. Mit den in Anhang B.1.2 angefügten Rechnungen nach der Schraubenberechnung ergibt sich mit dem Sicherheitsfaktor, der gewählten Anzahl von Schrauben und den angenommenen Massen der Teile eine Klemmlänge von ca. 8 mm bei einem M2,5 Gewinde.

Die nun vollständig konstruierte Verbindung ist in Abbildung 5.1 als Ausbruchzeichnung zu sehen.

# 6. Programmierung

In diesem Kapitel wird der Entwurf und die Umsetzung der Programmierung gezeigt. Die Ansprüche an eine Software zur Regelung einer Testanlage sind als hoch zu betrachten, da sich Fehler, die beim Programmieren gemacht werden unmittelbar auf die Testergebnisse auswirken. Aus diesem Grund wird die im Rahmen dieser Arbeit erzeugte Software nach dem Maßstab aktueller Softwareentwicklungsrichtlinien, wie dem C++ Style Guide von Google, geschrieben.

Die Wahl der Programmiersprache fiel auf C++, da dies ein gängiger Standard ist und flexibel eingesetzt werden kann. So kann jede der benötigten Funktionen wie Regelung, Sensorwerte aufnehmen oder eine grafische Betriebsoberfläche generieren durch die Nutzung vorhandener APIs und Bibliotheken implementiert werden.

Der in objektorientierten Programmiersprachen wie C++ übliche Aufbau mit Klassen, gewährleistet auch hier eine übersichtliche, funktionsorientierte Struktur. Durch diesen annähernd modularen Aufbau ist es unproblematisch im späteren Verlauf weitere Funktionen zu implementieren oder gegebenenfalls Fehler zu entfernen.

## 6.1. Software

Bevor näher auf den Entwurf und die Implementierung eingegangen wird, wird hier die Software vorgestellt die im Verlaufe der Arbeit genutzt wurde.

### 6.1.1. Solidworks

Die CAD-Software Solidworks von Dassault Systems wurde für die Konstruktion der mechanischen Komponenten verwendet. Solidworks ist eine übliche und weit verbreitete 3D-CAD-Software. Diese Software wurde auch zum Anfertigen der Konstruktionszeichnungen, auf deren Grundlage die Teile gefertigt wurden, genutzt.

### 6.1.2. Linux

Die Linux-Distribution Ubuntu wurde, mit Ausnahme der CAD Arbeiten, im gesamten Projektverlauf als Betriebssystem verwendet. Den Anstoß zur Entscheidung für Linux gaben die weitverbreitete Nutzung in der Softwareentwicklung, sowie die Tatsache, dass Linux auch am Institut von vielen zu diesem Zweck genutzt wird.

Neben der Nutzung als Betriebssystem auf dem Arbeitsplatzrechner wurde die Linux-Distribution Raspbian als Betriebssystem für den Raspberry Pi ausgewählt.

Der Großteil der hardwarenahen Programmierung fand über eine Remoteverbindung direkt auf dem Pi statt. Beim Anfertigen der Dokumetation war Texmaker ein nützliches Tool. Es handelt sich um eine IDE zum erzeugen von Latex Dokumenten, die Zugriff auf diverse Funktionen aus dem Texbereich hat und ermöglicht mit seiner Strukturansicht eine gute Übersicht, innerhalb der Dokumentenstruktur.

### 6.1.3. WiringPi

WiringPi ist eine Werkzeugbibliothek, die Funktionen für die Benutzung der Kommunikationsschnittstellen des Pi bereitstellt. Ohne diese Bibliothek wäre es notwendig gewesen eben diese Funktionen zu implementieren.

### 6.1.4. Qt

Qt ist ein sehr umfangreiches Werkzeug, das Funktionen von Tastatureventerkennung bis Datumseinbindung beinhaltet. Hier werden allerdings primär die Funktionen zur grafischen Darstellung genutzt.

### 6.1.5. Matlab/Simulink

Matlab und Simulink sind Werkzeuge, mit denen Kalkulationen und Simulationen durchgeführt werden können. Simulink verfügt über eine große Bibliothek, mit Funktionsblöcken für Funktionen von arithmetischer Addition bis zu mechanischen Aktoren.

### 6.1.6. Git

Als Versionierungssoftware wurde Git benutzt. Hiermit ist das Festhalten der Änderungshistorie sichergestellt und außerdem kann die entwickelte Software dem Endnutzer als Repository einfach zugänglich gemacht werden.

## 6.2. Modell der Strecke

Bei der Regelstrecke handelt es sich um eine Strecke mit $PT_1$ Verhalten. Dies wird aus der Annahme gewonnen, dass sich in der Regelstrecke nur ein Energiespeicher, in Form der Spule des Elektromotors befindet. Ein $PT_1$-Glied wird dadurch charakterisiert, dass sie ein proportionales Übertragungsverhalten mit einer Verzögerung erster Ordnung aufweißt.

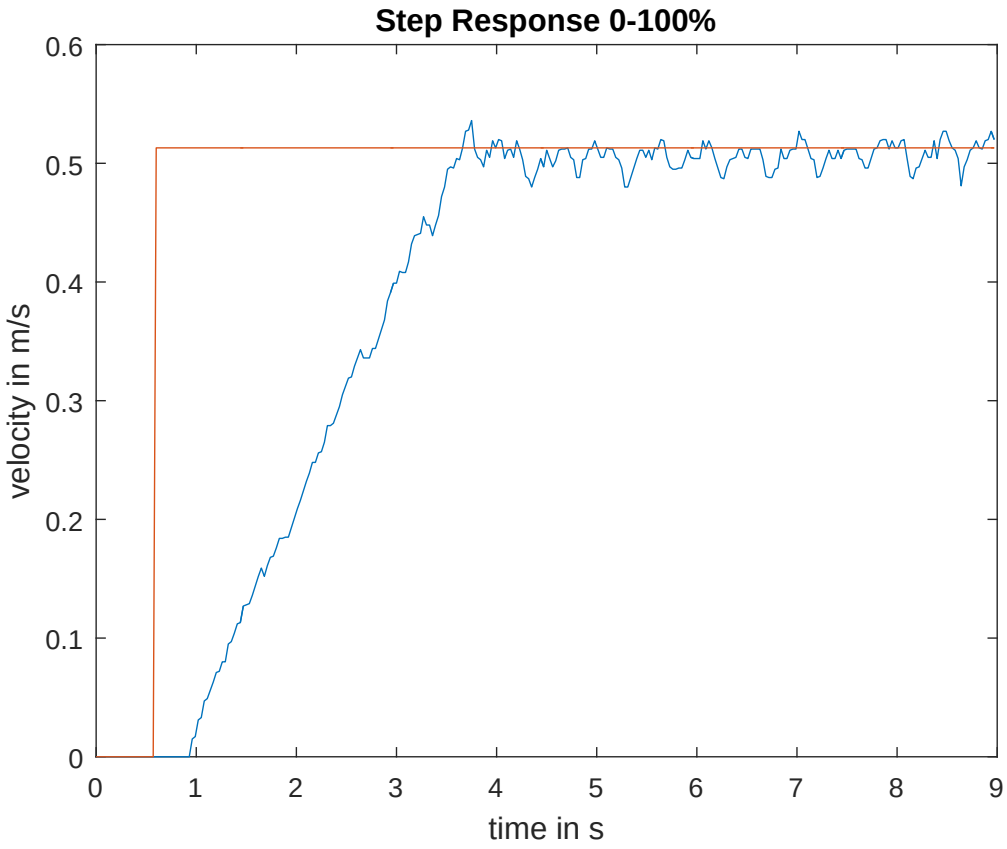



Abbildung 6.1.: Sprungantwort der Strecke bei einem Sprung auf 0.513 m/s

Mit der in Abbildung 6.1 gezeigten Sprungantwort lassen sich die Streckenparameter $T_u$ = 0,32 s und $T_g$ = 2,54 s, mit dem Wendetangentenverfahren identifizieren.

## 6.3. Reglerentwurf

Bevor der tatsächliche Reglerentwurf vorgenommen wird, müssen einige grundlegende Annahmen getroffen und Entscheindungen gefällt werden.

- Zum Identifizieren der Streckenparameter wurde die Sprungantwort der Regelstrecke aufgenommen. Aus diesen werden die Reglerparameter, anhand einer Einstellregel abgeleitet.
- Als Einstellregel wird die nach Chien/Hrones und Reswick für 20% Überschwinger und gutes Führungsverhalten gewählt, da diese einen guten Kompromiss zwischen schnellem Erreichen des Endwertes und geringem Überschwingen bietet.

Mit den in 6.2 ermittelten Werten und dem ausgewählten Einstellverfahren können die Reglerparameter wie folgt berechnet werden.

- $K_P = K_R = (0,6/K_s) \cdot (T_g/T_u) = 4,76$
- $T_N = T_g$
- $K_I = K_R/T_N = 1,87$ 1/s

Das System zeigt mit PID-Auslegung ein merkwürdiges Verhalten. Mit dem eingestellten D-Anteil, entsprechend der Einstellregeln, erreicht das System einen, nicht der Vorgabe entsprechenden Endwert. Dieses Verhalten war bei wiederholten Versuchen mit Sprüngen am Eingang reproduzierbar. Ursächlich hierfür könnten nicht berücksichtigte Auswirkungen der vorhanden Motorelektronik auf den Regelkreis sein. Um dennoch eine Regelung vornehmen zu können wurde deshalb stattdessen die PI-Auslegung gewählt, mit der ein nachvollziehbares und den Vorgaben entsprechendes Ergebnis erzielt werden konnte.

## 6.4. Kommunikation

Dieses Kapitel zeigt den Aufbau der Kommunikation zwischen Raspberry Pi und Benutzeroberfläche. Hierzu wurde der institutsinterne Standard NDLCom verwendet. Mit NDLCom ist es möglich verschiedene System IDs zu definieren und sie so direkt anzusprechen.

Implementiert wurde die Kommunikation so, dass zunächst eine NDLCom-Bridge eingerichtet wird, die die Kommunikation auf den festgelegten Ports überwacht. Wird ein NDLCom Telegramm mit der passenden ID „Treadmill" erkannt, wird das Telegramm verarbeitet. NDLCom Telegramme vom Typ PID-Controller enthalten den Geschwindigkeitssollwert, die Proportionalverstärkung, die Intergratorzeitkonstante, die Differentiatorzeitkonstante und den aktuellen Geschwindigkeitswert. Diese Werte werden extrahiert und an die entsprechenden Schnittstellen des Reglerprogramms weitergeleitet. Der Sollwert wird lediglich beim Empfangen verarbeitet und der aktuelle Geschwindigkeitswert nur beim Versenden befüllt.

## 6.5. GUI-Entwurf

Eine grafische Benutzeroberfläche sollte den Anspruch erfüllen, dem Endnutzer die gewünschte Funktion so intuitiv und mit so wenig Erklärungsaufwand wie möglich zur Verfügung zu stellen. Unter diesem Gesichtspunkt wurde die Oberfläche aufgeräumt und mit wenigen Bedienelementen entworfen.

So wurde beispielsweise auf einen Knopf zum Versenden der Werte verzichtet. Die Übermittlung findet statt, wenn einer der einstellbaren Werte geändert wird, beziehungsweise das Laufband gestartet oder gestoppt wird. Wie in Abbildung 6.2 zu sehen ist, können die in Kapitel 6.4 beschriebenen Reglerwerte eingestellt werden. Eine Ausnahme bildet hier der Wert für die aktuelle Geschwindigkeit. Dieser wird unidirektional empfangen und lediglich angezeigt.

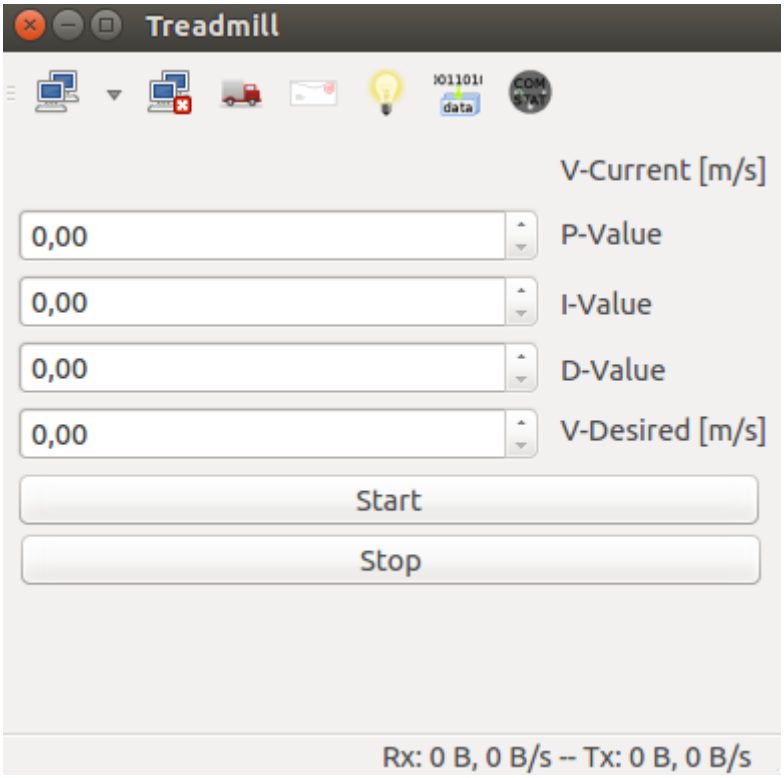



Abbildung 6.2.: Grafische Benutzeroberfläche zur Nutzung des Laufbandes

Es gibt allerdings notwendige Konfigurationen, die vor einer Inbetriebnahme der Regelung durch die Benutzeroberfläche vorgenommen werden müssen.

- Es muss sichergestellt werden, dass sich der Computer auf dem die Benutzeroberfläche ausgeführt wird im selben Netz wie der Raspberry Pi befindet.
- Die IP-Adresse des Raspberry Pi muss identifiziert werden.

Der Raspberry Pi hat den Hostname „treadmill" und ist in der institutstypischen Domäne .dfki.uni-bremen.de. Man kann sich also entweder mit

```
ssh pi@treadmill.dfki.uni-bremen.de
```

mit dem Pi verbinden und die IP-Adresse mit dem Befehl

```
ifconfig
```

herausfinden, oder man nutzt das Werkzeug nmap mit dem Befehl

```
nmap -sn 10.250.*.0/22 | grep treadmill
```

um die IP-Adresse herauszufinden.

- Diese IP-Adresse muss unter UDP->hostname eingetragen werden. Hier muss außerdem der Senderport auf 34000 und der Empfängerport auf 34001 eingestellt werden.

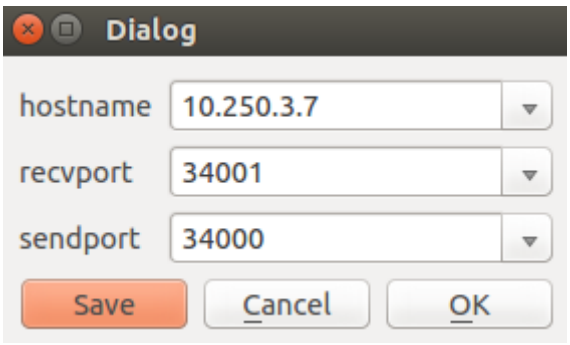



Abbildung 6.3.: Konfiguration der Verbindung zum Regler

- Die Verbindung zum Raspberry Pi muss mit einem Klick auf OK hergestellt werden.

Solange das Reglerprogramm auf dem verbundenen Raspberry Pi läuft, sollte nun eine Steuerung des Reglers durch die Benutzeroberfläche möglich sein.

Teile des Codes der im Zusammenhang mit der Nutzung von NDLCom und der GUI steht, stammen von [Zenzes u. a. (2016)].

# 7. Test der Anlage und Diskussion

Der Test der Anlage, beziehungsweise des Gesamtsystems stellt den abschließenden Schritt dar. Zunächst soll jedoch überprüft werden, ob die Anforderungen korrekt umgesetzt wurden. Bei manchen Anforderungen kann auch ohne Test festgestellt werden, ob diese korrekt umgesetzt wurden.

## 7.1. Überprüfung der Anforderungen

### 7.1.1. Entwicklung eines Gesamtkonzepts zur Laufbandregelung

Das Gesamtkonzept wurde in Kapitel 4.3 aufgestellt und dargelegt. Es enthält alle wesentlichen Bestandteile einer Regelung. Ein Sollwert wird durch den Benutzer in der Benutzeroberfläche vorgegeben, die Differenz von Soll- und Istwert kann durch die Rückspeisung des Sensorwertes der aktuellen Geschwindigkeit gebildet werden. Der Aktor kann durch einen Ausgangswert, der in die Motorsteuerung gespeist wird beeinflusst werden.

### 7.1.2. Stufenlose Geschwindigkeitsregelung zwischen minimaler und maximaler Laufbandgeschwindigkeit

Eine stufenlose Einstellung war leider aus mehreren Gründen nicht realisierbar. Zum einen gibt es die Begrenzung durch die Software, da der eingestellte Sollwert nicht beliebig fein aufgelöst werden kann. Durch den genutzten Datentyp Float ist die Auflösung auf 32-Bit beschränkt. Nach dem IEEE 754 können damit mindestens 7 Nachkommastellen dargestellt werden. Aus dem möglichen Drehzahlbereich von 0 bis 1,56 1/s (0,513 m/s) ergibt sich eine maximale Auflösung von 0,0000001 1/s.

Dies ist jedoch nicht die einzige Begrenzung. Der Sensor ist in der Lage 360° in $2^{14}$ Werten darzustellen. Das bedeutet, dass ein Bit etwa 0,022° entspricht. Die Abtastfrequenz wird nun so gewählt, dass das Nyquist-Shannon-Abtasttheorem erfüllt wird und die Geschwindigkeit möglichst hoch aufgelöst wird. Die im Folgenden als als $f_{Laufband}$ bezeichnete

Frequenz des Laufbandes bezieht sich auf die Drehfrequenz der Rolle, die das Laufband antreibt und an der der Sensor befestigt ist.

$f_{Laufband,max} = 25,92Hz$

$f_{s,min} > 2 \cdot f_{Laufband,max} = 51,84Hz$

$f_s = 52Hz$

Die gewünschte Abtastfrequenz zum Erreichen einer möglichst hoch aufgelösten Geschwindigkeit, entspricht also 52 Hz, da die Auflösung sich umgekehrt proportional zur Abtastfrequenz verhält. Damit ergibt sich eine maximale Auflösung von ca. 0,001 m/s (0,02 rad/s), die erfasst werden kann. Damit wird bei einer Laufbandgeschwindigkeit von 10 rad/s ein maximal überstrichener Winkel von 0,573° erreicht. Damit kann auch, mit einer sehr großen Reserve, auf keinen Fall mehr als eine Umdrehung während der Abtastzeit vorkommen. Hieraus ergibt sich, dass die Begrenzung des float Datentyps gar nicht ins Gewicht fällt, da die Auflösung, die durch den Sensor erreicht werden kann, um 4 Zehnerpotenzen kleiner ist als die von float.

Der dritte begrenzende Faktor ist der Aufbau der Motorsteuerung. Sie erlaubt es nur die Motorgeschwindigkeit zwischen 0 und 16393 einzustellen. Das bedeutet, dass die Geschwindigkeit theoretisch auf 0,000031 m/s genau eingestellt werden kann. Laut Bedienungsanleitung soll der Sollwert jedoch lediglich in 100 Schritten zwischen 0 - 100/0,0061 eingestellt werden.

Da diese Begrenzungen durch äußere Bedingungen vorgegeben sind und die Einstellung nahezu stufenlos ist wird die Anforderung dennoch als erfüllt betrachtet.

### 7.1.3. Bestimmung eines geeigneten Reglers zur Kontrolle des Laufbandes

In dem in Kapitel 6.3 vorgenommenen Reglerentwurf wird dargelegt, wieso der bestimmte Regler zur Kontrolle des Laufbandes geeignet ist. Dies ist ein geeigneter Beleg für die Umsetzung der Anforderung.

### 7.1.4. Implemetierung des Reglers

Die Implementierung des Reglers kann durch die Gegenüberstellung der Sprungantworten des ungeregelten und des geregelten Systems belegt werden. Diese Gegenüberstellung findet in Abbildung 7.1 statt.

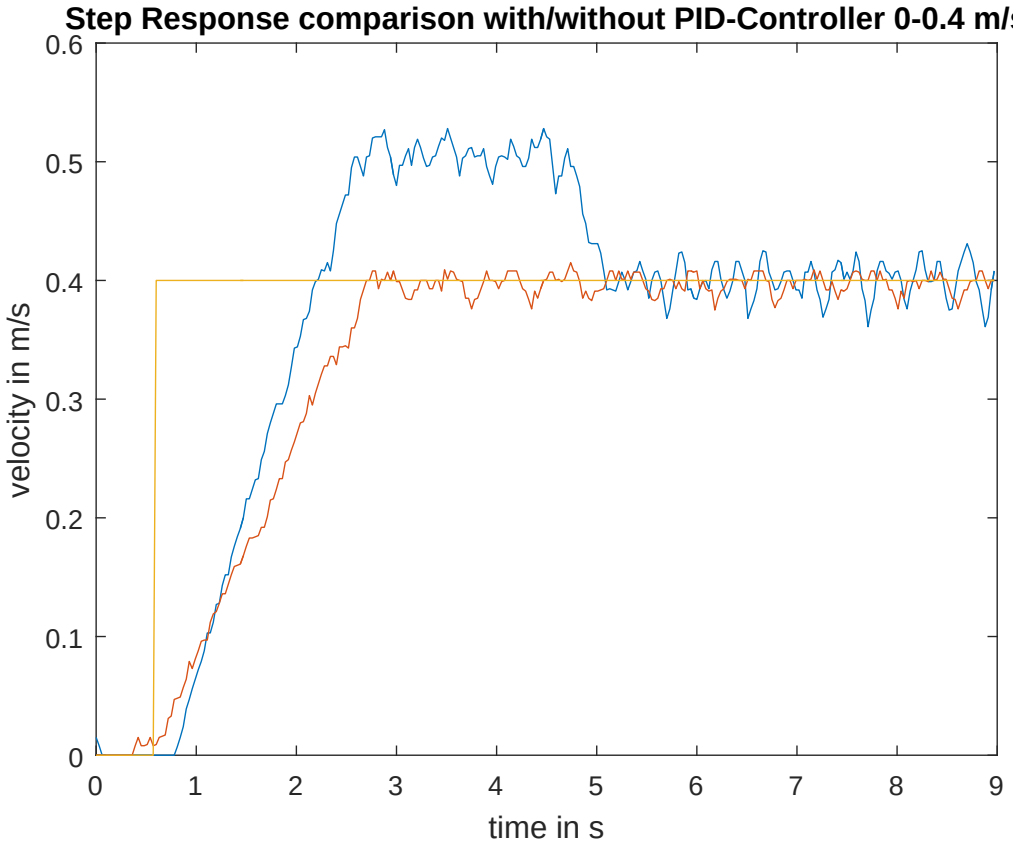



Abbildung 7.1.: Vergleich der geregelten (blau) und ungeregelten (orange) Sprungantwort auf 0,4 m/s

### 7.1.5. Bestimmung der Strecken- und Reglerparameter

Die Bestimmung der Strecken- und Reglerparameter wird in Kapitel 6.3 gezeigt. Die Parameter wurden bestimmt.

### 7.1.6. Auswahl geeigneter Sensorik

Die in Kapitel 4.5 ausgewählte Sensorik ist geeignet, da sie durch den Raspberry Pi ausgelesen werden kann und in der Lage ist, alle Drehzahlen die das Laufband erreichen kann zu erfassen.

### 7.1.7. Auslesen der Sensorik

Als Test des korrekten Auslesens wurde hier eine Vergleichsmessung mit einer Referenzzeit durchgeführt. Zum Ermitteln der Referenzzeit wurden im Abstand von einem Meter zwei Markierungen auf dem Laufband angebracht. Die Zeit wurde genommen, ab dem Zeitpunkt des Passierens der ersten Markierung des Messpunktes, bis zum Passieren der zweiten Markierung des selben Punktes. Die Messungen ergaben, bei einem fest eingestellten Sollwert, den Wert 0,513 m/s beim Auslesen des Sensors und den Wert 0,5128 m/s durch die gemessene Referenzzeit. Diese Werte stimmen annähernd überein, was ein Beleg für das korrekte Auslesen unter Berücksichtigung von Messungenauigkeiten ist.

### 7.1.8. Konzeption und Konstruktion von mechanischen Komponenten wie z.B. der mechanischen Anbindung des Sensors an das Laufband

Die Konzeption und Konstruktion der mechanischen Anbindung des Sensors wurde in Kapitel 5 vorgenommen. Die entsprechenden Teile wurden angeschafft und gefertigt, sodass der Sensor wie in Abbildung 7.2 zu sehen ist, an dem Laufband befestigt werden kann.

### 7.1.9. Programmieren einer GUI zur Ansteuerung des Reglers unter Verwendung einer „Institusvorlage” in C++

Die GUI wurde wie in Abbildung 6.2 zu sehen ist programmiert.

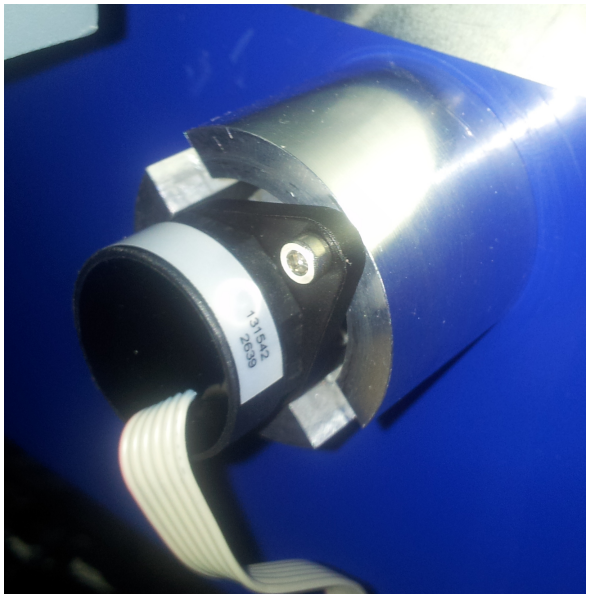



Abbildung 7.2.: Mechanische Sensoranbindung

## 7.2. Test des Gesamtsystems

Hier soll nun getestet werden, ob das System erwartungsgemäß funktioniert. Um dies tun zu können müssen zunächst zu testende Funktionen definiert werden, die dazu geeignet sind eine korrekte Funktion des Gesamtsystems festzustellen.

Diese Funktionen, die getestet werden müssen um eine ordnungsgemäße Funktionsweise des Systems sicherzustellen, sind:

- Korrekte Kommunikation zwischen Benutzeroberfläche und Regler
- Erwartungsgemäße Funktion des Reglers

Die korrekte Kommunikation wurde durch die Ausgabe der empfangenen Daten auf dem Zielsystem, sowie dem Vergleich der erwarteten mit den tastsächlichen Werten überprüft.

Die Funktion des Reglers wurde durch eine Simulation überprüft. Die Sprungantwort des geregelten Systems 7.3 wird dabei mit der simulierten Sprungantwort 7.4 verglichen. Dies zeigt, dass der Regler funktioniert, im Ergebnis aber etwas schlechter performt als die Simulation. Das könnte daran liegen, dass sich der implementierte Regler wahrscheinlich von dem in der Simulation verwendeten PID-Reglerblock von Simulink unterscheidet. Weitere mögliche Gründe sind Umwelteinflüsse, die in der Simulation nicht berücksitigt wurden und

die vorhandene Motorelektronik, deren Einfluss auf das Verhalten der Regelstrecke nicht präzise bestimmt werden konnte. Belastungstests haben gezeigt, dass der Regler, bei einer Belastung des Laufbandes durch eine Person, eine wesentlich konstantere Geschwindigkeit erreicht, als das ungeregelte System.

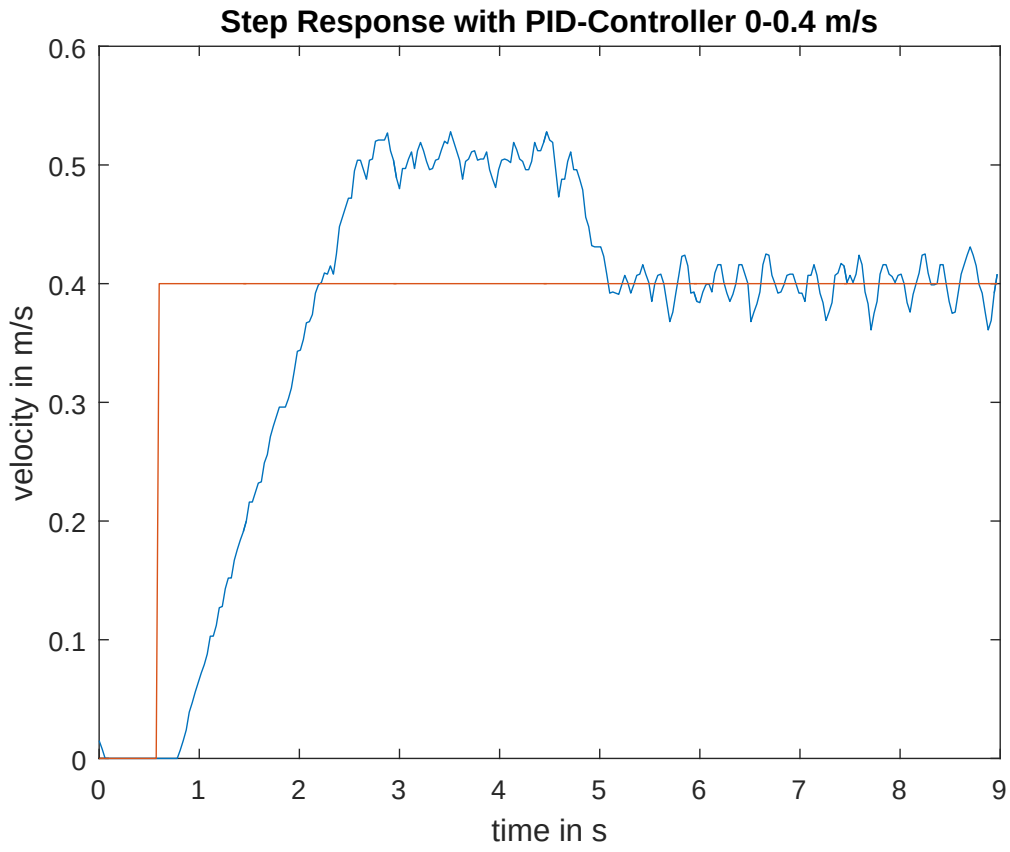



Abbildung 7.3.: Sprungantwort des geregelten Systems auf 0,4 m/s

## 7.3. Diskussion

Wie in Grafik 7.1 zu Sehen ist, weißt das ungeregelte System ein Rauschen beim halten des erreichten Endwertes auf, die der Regler nicht in der Lage ist auszuregeln.

Ein möglicher Grund hierfür könnte sein, dass es nicht möglich ist den Motor beliebig schnell anzusteuern. Bei einer Ansteuerung mit mehr als 4 Hz kann nicht mehr gewährleistet werden, dass jedes Kommando von der Motorsteuerung angenommen wird. In den

durchgeführten Versuchen wurde mit einer höheren Frequenz gearbeitet, um eine möglichst gute Performance zu erreichen.

Denkbar wäre auch, dass es in der mechanischen Kraftübertragung zwischen Motor und Laufband ein Spiel gibt. Dieses könnte dazu führen, dass bei der Umsetzung der Kraft eine unregelmäßige, ungleichmäßige Belastung auftritt.

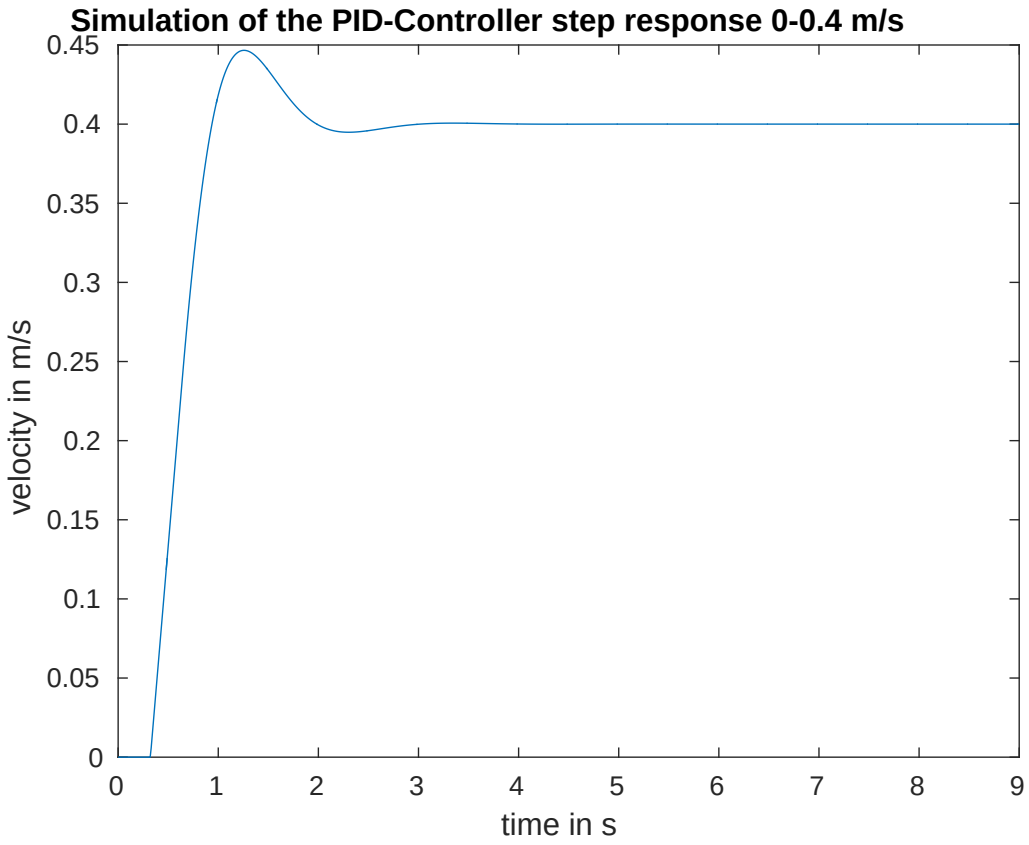



Abbildung 7.4.: Simulierte Sprungantwort des regelten Systems

# 8. Zusammenfassung und Ausblick

Das Ziel dieser Arbeit sollte es sein eine adaptive, konfigurierbare Motorsteuerung zu konzipieren und zu implementieren.
Einleitend wurde die Motivation für diese Arbeit erläutert und der vorgefundene Istzustand beleuchtet.
Dann wurden einige der Grundlagen, auf die im Verlauf der Arbeit zurückgegriffen wurde, dargelegt und erklärt.
Der Stand der Technik wurde aufgeführt, wobei auf kein besonders großes Kontingent an bestehenden Inhalten zurückgegriffen werden konnte.
Die gestellten Anforderungen wurden genannt und analysiert. Außerdem wurde aus ihnen die Struktur für das Vorgehen, sowie den Aufbau des Konzepts abgeleitet.
Nachdem die Anforderungen im Gesamtkonzept und entsprechenden Konzepten, für die der einzelnen Komponenten verarbeitet wurde, konnten diese unter Anwendung gängiger Verfahren implementiert werden.
Auf die Implementierung und Umsetzung der Konzepte folgten schlussendlich Tests, mit denen die ordnungsgemäße Umsetzung der Anforderungen sowie die Funktionsfähigkeit der Anlage überprüft wurde.

Es konnte gezeigt werden, dass das System mit den ermittelten Reglerparametern geregelt werden kann. Die Performance ist dabei allerdings noch nicht ideal. Die Gründe hierfür, die in Kapitel 7.3 angesprochen wurden, sollten in zukünftiger Arbeit validiert oder widerlegt werden, da dies im Rahmen dieser Arbeit nicht mehr möglich war. Um zu testen, ob die Ansprechzeit des Motors tatsächlich für das nicht Ausgleichen der Schwingung verantwortlich ist, kann der Regler an einem Referenzsystem, ohne eine solche Beschränkung getestet werden. Damit herausgefunden werden kann, ob ein mechanisches Spiel vorliegt, könnte der Motor auf einem Teststand montiert werden und dort entsprechend belastet und getestet werden.

Das Ergebnis dieser Arbeit ist grundsätzlich dazu geeignet in der Zukunft auch mit anderen Systemen als speziell diesem Laufband eingesetzt zu werden.

# Literaturverzeichnis


[Alexandrov und Palenov 2014] ALEXANDROV, Al'bert G. ; PALENOV, Maksim V.: Adaptive PID controllers: State of the art and development prospects. In: *Automation and Remote Control* 75 (2014), Nr. 2, S. 188–199

[Bate 2009] BATE, Boulent: *Spezialgebiete der Steuer- und Regelungstechnik WS 2008/2009*. 2009

[Bode 2013] BODE, Erasmus: *Konstruktionsatlas: Werkstoffgerechtes Konstruieren/Verfahrensgerechtes Konstruieren*. Springer-Verlag, 2013

[DFKI 2012] DFKI: *LegLab*. 2012. – URL http://robotik.dfki-bremen.de/de/forschung/testanlagen/leglab.html. – [Online; Stand 01. März 2016]

[DFKI 2015] DFKI: *LIMES - Lernen intelligenter Bewegungen kinematisch komplexer Laufroboter für die Exploration im Weltraum*. 2015. – URL http://robotik.dfki-bremen.de/en/research/projects.html?jumpurl=uploads%2Ftx_dfkiprojects%2FProjektblatt_LIMES_DE.pdf&juSecure=1&locationData=75%3Apages%3A75&juHash=0f530d398abe4d7e267cb8ffb87a95a0cc0ea4d7. – Zeile 2-4 [Online; Stand 01. März 2016]

[L.Billmann 2010] L.BILLMANN, Prof.Dr.-Ing.: *Die PT1-Tt-Prozeßbeschreibung*. 2010. – URL http://www.billmann-web.de/research/control/prz4expl.htm. – [Inline; Stand 29. Februar 2016]

[Lunze 2010] LUNZE, Jan: *Regelungstechnik 1–Systemtheorietische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Bd. 8. 2010. – 710 S

[Muhs u. a. 1997] MUHS, Dieter ; WITTEL, Herbert ; BECKER, Manfred: *Roloff/Matek Maschinenelemente Formelsammlung*. Springer, 1997

[Waste 2005] WASTE: *Regelungstechnik*. 2005. – URL http://rn-wissen.de/wiki/index.php/Regelungstechnik. – [Online; Stand 29. Februar 2016]

[Wittel u. a. 2009] WITTEL, Herbert ; MUHS, Dieter ; JANNASCH, Dieter ; VOSSIEK, Joachim: Roloff/Matek Maschinenelemente. In: *Vieweg+ Teubner* (2009)

[Zenzes u. a. 2016] ZENZES, Martin ; KAMPMANN, Peter ; STARK, Tobias ; SCHILLING, Moritz: *NDLCom: Simple Protocol for Heterogeneous Embedded Communication Networks*. 2016. – URL http://robotik.dfki-bremen.de/de/forschung/publikationen/8214.html. – [Online; Stand 16.03.2016]

[Ziegler und Nichols 1942] ZIEGLER, John G. ; NICHOLS, Nathaniel B.: Optimum settings for automatic controllers. In: *trans. ASME* 64 (1942), Nr. 11

# A. Konstruktionszeichnungen

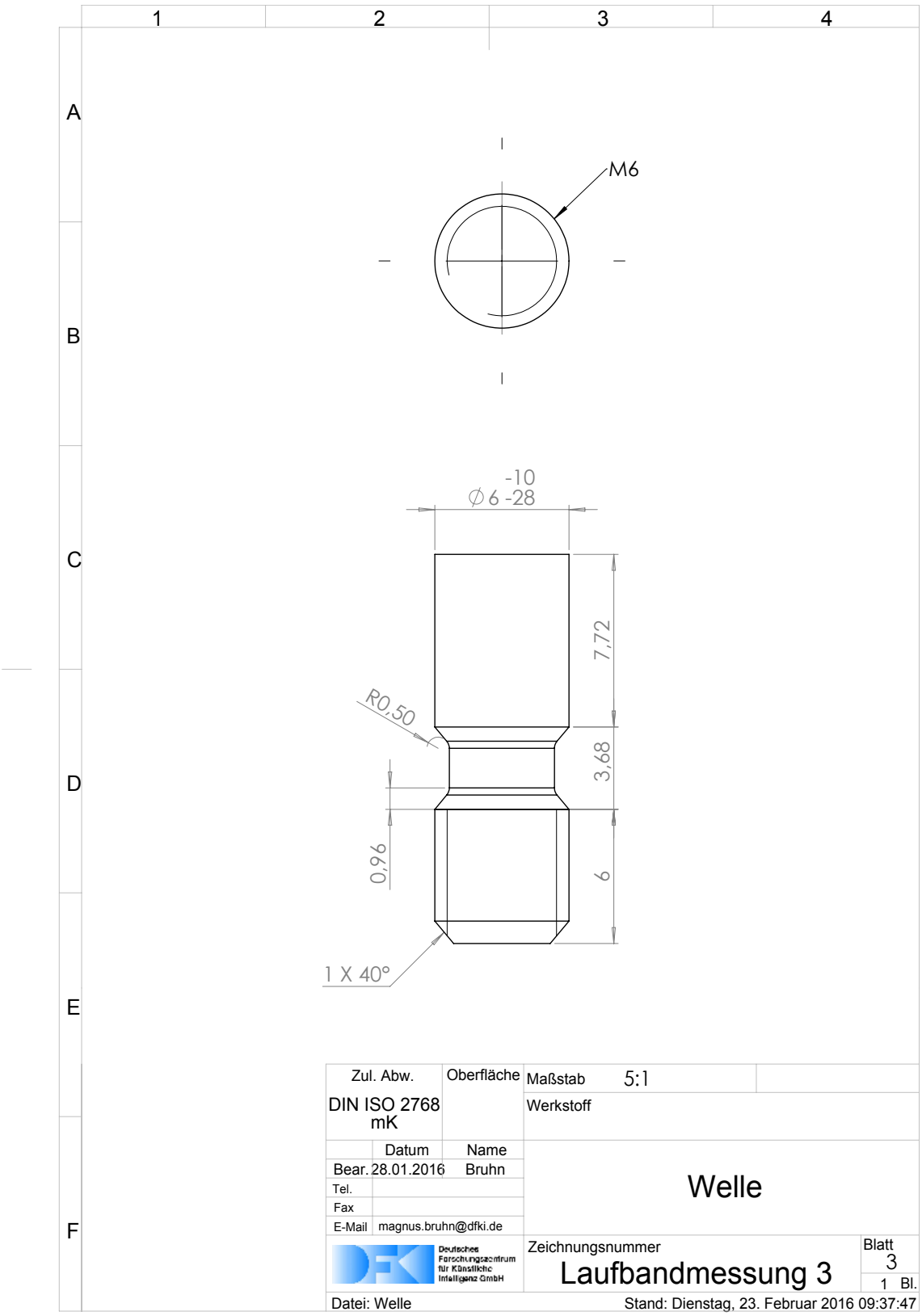



Abbildung A.1.: Welle der Sensor-Laufbandverbindung

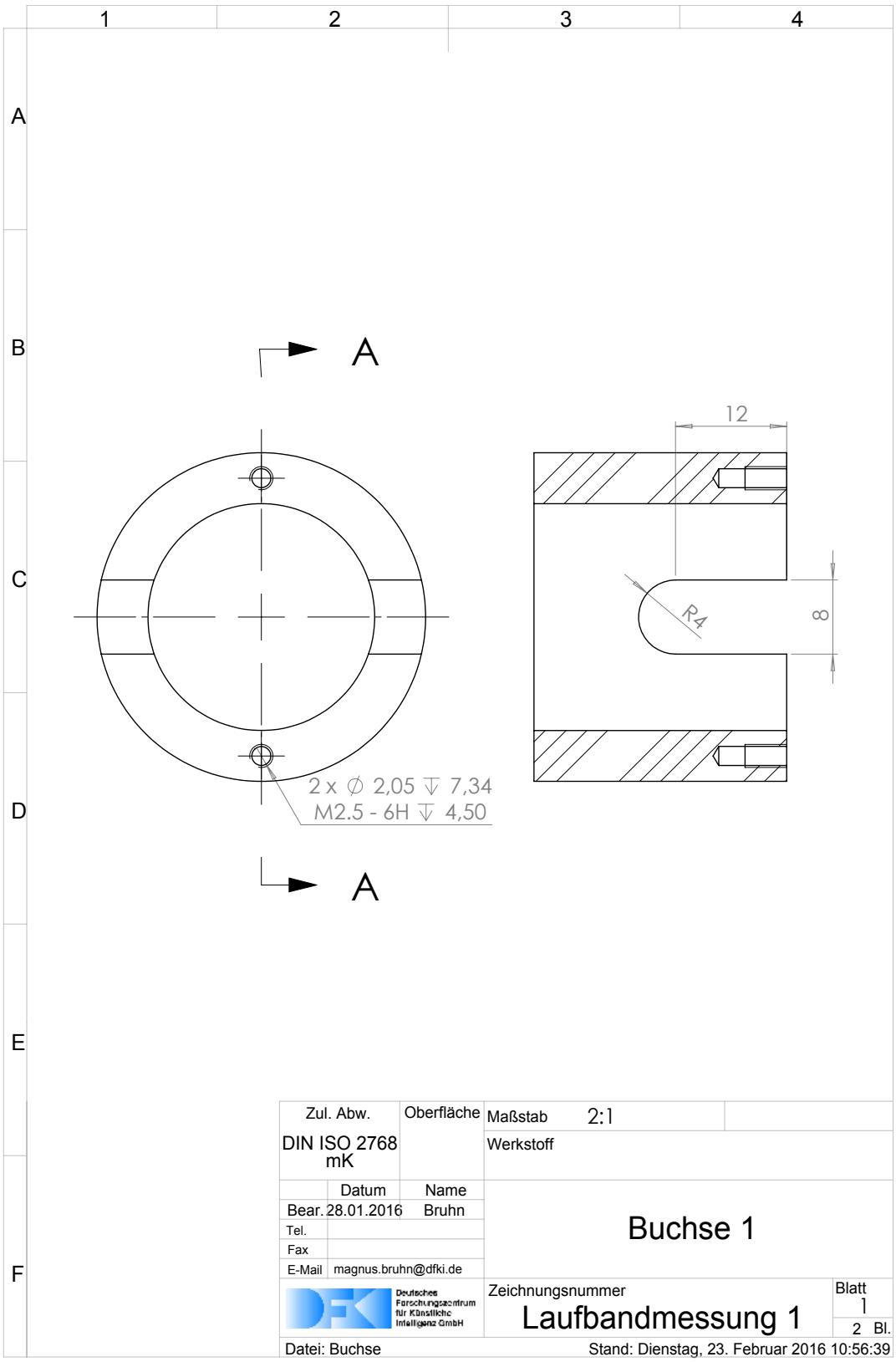



Abbildung A.2.: Buchse Ansicht 1 der Sensor-Laufbandverbindung

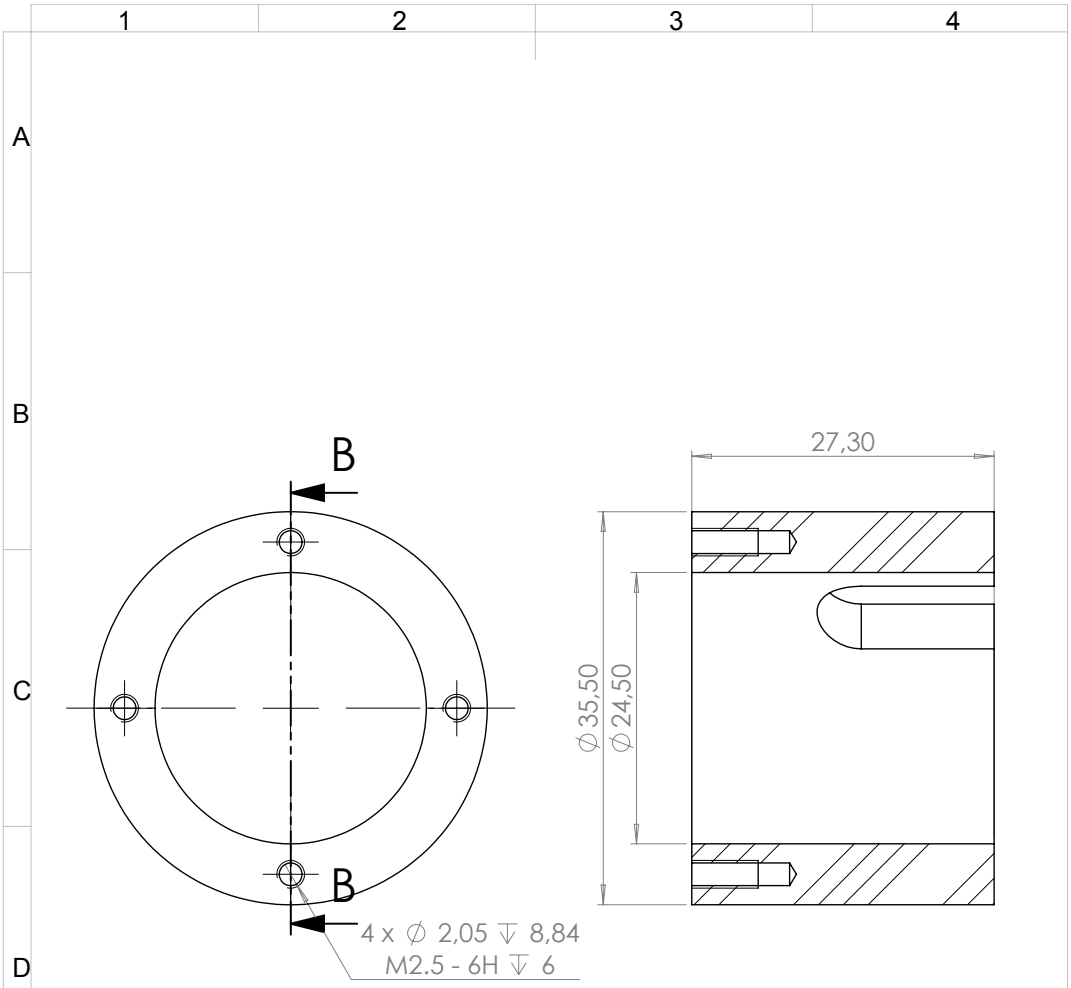



Abbildung A.3.: Buchse Ansicht 2 der Sensor-Laufbandverbindung

# B. Rechnungen

## B.1. Schrauben- und Gewindeberechnung

Hier werden die Berechnungen zum Gewinde der Welle A.1, sowie die Auslegung der Schrauben zur Verbindung des Sensors mit der Buchse A.2 und der Buchse mit dem Laufbandgehäuse dargelegt.

### B.1.1. Gewindefreistich der Welle

Auslegung nach [Wittel u. a. (2009) S.228].

Annahmen:

- Gewinde = M6
- P = 1

Bekannte Werte:

- r = 0,5
- g = 3,5
- $d_3$ = 5
- $d_g$ = 4,7
- $\alpha \geq 30^\circ$

### B.1.2. Schraubenauslegung

Auslegung nach [Muhs u. a. (1997) S.72 ff.].

Bekannte Werte:

- $m_{Sensor}$ = 6,04 g
- $m_{Buchse}$ = 54,18 g
- $m_{ges} \approx$ 60 g

Annahmen:

- Gewinde = M2,5
- P = 0,45
- $F_B \approx 0$
- $z_{Gehäuse}$ = 4
- $z_{Sensor}$ = 2
- $\mu \approx$ 0,2 (Stahl auf Stahl)
- S = 2
- $R_{p0,2} \approx$ 640 N/mm$^2$ (8,8)
- $\kappa$ = 1,41
- $k_A$ = 1,6
- $\beta$ = 0,8
- E = 210000 N/mm$^2$
- $f_z$ = 0,011 mm

#### Buchse-Gehäuse

$$F_t = m_{ges} \cdot g = 0,06kg \cdot 9,81m/s^2 = 0,59N$$

$$F_{kl} \approx \frac{S \cdot F_t}{\mu \cdot z_Gehuse} = 1,472N$$

$$l_k \geq \frac{\beta \cdot E \cdot f_z}{\frac{R_{p0,2}}{\kappa \cdot k_A} - \frac{F_{kl}}{A_T}} = 6,53mm => 8mm$$

### Buchse-Sensor

$F_t = m_{ges} \cdot g = 0,006kg \cdot 9,81m/s^2 = 0,059N$

$F_{kl} \approx \frac{S \cdot F_t}{\mu \cdot z_S ensor} = 0,294N$

$l_k \geq \frac{\beta \cdot E \cdot f_z}{\frac{R_p 0,2}{\kappa \cdot k_A} - \frac{F_{kl}}{A_T}} = 6,52mm => 8mm$

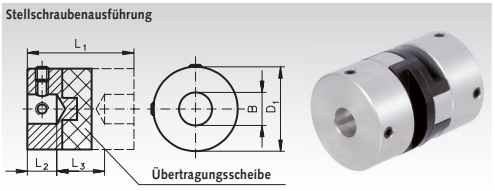
# C. Datenblätter

## Drehstarre Kupplungen HZ mit Sackloch

**Werkstoff:** Naben aus Alu-Legierung mit Aluchrom Oberflächenbehandlung. Übertragungselement aus schwarzem Azetal.

Diese 3-teiligen spielfreien und elektrisch isolierenden Kupplungen bestehen aus 2 Naben und einer Übertragungsscheibe.
Sie sind vielseitig einsetzbar und robust gebaut. Hoher Radialausgleich, einfache Montage auch bei schwer zugänglichen Einbausituationen.

Anwendung: Ideal für Schrittmotoren durch die dämpfenden Eigenschaften der Übertragungsscheibe aus Kunststoff. Positionierantriebe, Stellungsgeber wie Inkremental- oder Absolutencoder, Pumpen etc.
Temperaturbereich: -20°C bis +60°C.




**Bestellangaben z.B.: Art.-Nr. 601 201 00, Kupplung HZ, 2 mm Bohrung**

| Artikel-Nr. | Drehmoment max.[1] Nm | Statisches Bruchmoment Nm | Bohrung $B^{+0,03}$ mm | $L_1$ mm | $L_2$ mm | $L_3$ mm | $D_1$ mm | max. Verlagerung bei 3000 min$^{-1}$ Winkel ±Grad | radial ±mm | Verdrehsteifigkeit Nm/rad | Gewicht g | Artikel-Nr. Ersatzteil Kreuzscheibe | Gewicht g |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **601 201 00*** | 0,06 | 0,7 | 2 | 12,7 | 3,8 | 5,1 | 6,4 | 0,5 | 0,1 | 10 | 2,5 | **601 237 00** | 0,1 |
| **601 202 00*** | 0,06 | 0,7 | 3 | 12,7 | 3,8 | 5,1 | 6,4 | 0,5 | 0,1 | 10 | 2,5 | **601 237 00** | 0,1 |
| **601 203 00*** | 0,21 | 2 | 3 | 12,7 | 3,8 | 5,1 | 9,5 | 0,5 | 0,1 | 30 | 4 | **601 238 00** | 0,1 |
| **601 204 00*** | 0,21 | 2 | 4 | 12,7 | 3,8 | 5,1 | 9,5 | 0,5 | 0,1 | 30 | 4 | **601 238 00** | 0,1 |
| **601 206 00*** | 0,5 | 4 | 3 | 15,9 | 4,3 | 7,3 | 12,7 | 0,5 | 0,1 | 65 | 11 | **601 239 00** | 0,5 |
| **601 207 00*** | 0,5 | 4 | 4 | 15,9 | 4,3 | 7,3 | 12,7 | 0,5 | 0,1 | 65 | 11 | **601 239 00** | 0,5 |
| **601 208 00*** | 0,5 | 4 | 6 | 15,9 | 4,3 | 7,3 | 12,7 | 0,5 | 0,1 | 65 | 11 | **601 239 00** | 0,5 |
| **601 301 00** | 1,7 | 8 | 4 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 302 00** | 1,7 | 8 | 6 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 303 00** | 1,7 | 8 | 8 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 305 00** | 4 | 13 | 6 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 306 00** | 4 | 13 | 8 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 307 00** | 4 | 13 | 10 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 308 00** | 9 | 53 | 8 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 309 00** | 9 | 53 | 10 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 310 00** | 9 | 53 | 12 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 312 00** | 17 | 57 | 10 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,7 |
| **601 313 00** | 17 | 57 | 12 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,7 |
| **601 315 00** | 17 | 57 | 16 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,7 |

* Naben aus Messing.

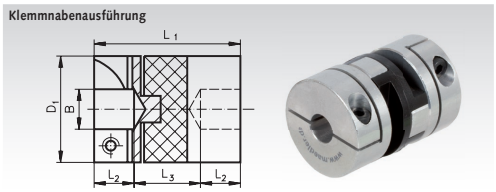
## Drehstarre Kupplungen HF mit Sackloch

**Werkstoff:** Naben aus Alu-Legierung mit Aluchrom Oberflächenbehandlung. Übertragungselement aus schwarzem Azetal.

Diese 3-teiligen spielfreien und elektrisch isolierenden Kupplungen HF bestehen aus 2 Naben und einer Übertragungsscheibe.
Sie sind vielseitig einsetzbar und robust gebaut. Hoher Radialausgleich, einfache Montage auch bei schwer zugänglichen Einbausituationen.

Anwendung: Ideal für Schrittmotoren durch die dämpfenden Eigenschaften der Übertragungsscheibe aus Kunststoff. Positionierantriebe, Stellungsgeber wie Inkremental- oder Absolutencoder, Pumpen etc.

Temperaturbereich: -20°C bis +60°C.




**Bestellangaben: z.B.: Art.-Nr. 601 401 00, Kupplung HF, 4 mm Bohrung**

| Artikel-Nr. | Drehmoment max.[1] Nm | Statisches Bruchmoment Nm | Bohrung $B^{+0,03}$ mm | $L_1$ mm | $L_2$ mm | $L_3$ mm | $D_1$ mm | max. Verlagerung bei 3000 min$^{-1}$ Winkel ±Grad | radial ±mm | Verdrehsteifigkeit Nm/rad | Gewicht g | Artikel-Nr. Ersatzteil Kreuzscheibe | Gewicht g |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **601 401 00** | 1,7 | 8 | 4 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 402 00** | 1,7 | 8 | 5 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 403 00** | 1,7 | 8 | 6 | 22 | 6,3 | 9,4 | 19,1 | 0,5 | 0,2 | 115 | 12 | **601 242 00** | 1,5 |
| **601 407 00** | 4 | 13 | 6 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 408 00** | 4 | 13 | 8 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 409 00** | 4 | 13 | 10 | 28,4 | 8,6 | 11,2 | 25,4 | 0,5 | 0,2 | 205 | 31 | **601 244 00** | 2,7 |
| **601 411 00** | 9 | 53 | 8 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 412 00** | 9 | 53 | 10 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 413 00** | 9 | 53 | 12 | 48 | 13 | 22 | 33,3 | 0,5 | 0,2 | 615 | 86 | **601 246 00** | 8 |
| **601 415 00** | 17 | 57 | 10 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,2 |
| **601 416 00** | 17 | 57 | 12 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,2 |
| **601 418 00** | 17 | 57 | 16 | 50,8 | 16,7 | 17,4 | 41,3 | 0,5 | 0,25 | 1200 | 148 | **601 248 00** | 12,2 |

[1] Betriebsfaktoren (ohne Wellenverlagerung):

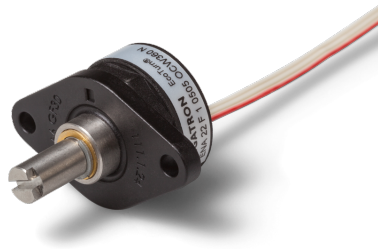
| Lastdauer | Betriebsfaktor |
|---|---|
| kurzzeitig | 1 |
| 1 Stunde pro Tag | 2 |
| 3 Stunden pro Tag | 4 |
| 6 Stunden pro Tag | 6 |
| 12 Stunden pro Tag | 8 |



Abbildung C.1.: Kupplung der Sensor-Laufbandverbindung

# Data Sheet for Angle Sensors

**Hall-Effect Single-Turn Rotary Encoder with Serial Output** — **Series ENS22 F**



- SPI- or SSI- Interface
- Resolution up to 14 bit (16384 positions)
- Life expectancy > 25 x $10^6$ shaft revolutions
- Potted electronics

The Series ENS22 F closes the gap between potentiometric sensors and high end sensors with ball bearings. Due to the easy connection of the flat ribbon cable, e.g. using IDC, the cost of soldering can be reduced as well. For safety critical applications there is a redundant version available.

| Electrical Data | | |
|---|---|---|
| Effective electrical angle of rotation 1.) | 0..360° | |
| Independent linearity (best straight line) 1.) | ± 0,5 % | |
| Output signal | SPI | SSI |
| Theoretical Resolution | 14 bit | 12 bit |
| Updaterate | 0,2 ms | 0,1 ms |
| Supply voltage | 5 VDC ± 10 % | 9-30 VDC |
| Power consumption (no load) | ≤ 30 mA | |
| Insulation voltage 1.) | 1000 VAC @ 50 Hz, 1 min | |
| Insulation resistance 1.) | 2 MOhm @ 500 VDC, 1 min | |

| Mechanical Date, Environmental Conditions, Miscellaneous | |
|---|---|
| Mechanical angle of rotation 1.) | Without stops |
| Lifetime 2.) | > 25 x $10^6$ rotations<br>Depending on the application - values determined at room temperature +20 ° C, with a radial load of 1 N |
| Bearing | Sleeve bearing |
| Max. operational speed | 4000 rpm |
| Starting torque @ ambient temperature 1.) 2.) | < 0,6 Ncm |
| Operating temperature range | -40..+85 °C (fixed cable, extended temperature range on request) |
| Storage temperature range | -40..+105 °C |
| Protection grade (IEC 60529) | IP65 |
| Sealing shaft / bearing | no sealing (IP40) |
| Vibration (IEC 68-2-6, Test Fc) | ±1,5 mm / 20 g / 10 bis 2000 Hz / 16 frequency cycles (3x4 h) |
| Shock (IEC 68-2-27, Test Ea) | 50 g / 11 ms / half sine (3x6 Shocks) |
| Max. radial load | 1 N |
| Mass | approx. 19 g |
| Material shaft | stainless steel |
| Material housing | plastic |



Abbildung C.2.: Datenblatt Megatron ENS22F

# D. Quellcode

## D.1. Main

### D.1.1. treadmill.cpp

```
#include "ndlcom/Node.h"
#include "ndlcom/Bridge.h"
#include "ndlcom/InternalHandler.hpp"
#include "representations/PIDcontrol.h"
#include "representations/id.h"
#include "ndlcom/ExternalInterfaceParseUri.hpp"
#include "include/pidcontroller.hpp"
#include "include/vsens.hpp"
#include "include/serialcom.hpp"
#include <unistd.h>

class PrivateHandler : public ndlcom::NodeHandlerBase {
  public:
    PrivateHandler(struct NDLComNode &_node) : NodeHandlerBase(_node) {
      Pval = 0;
      Ival = 0;
      Dval = 0;
      Vset = 0;
      Stop = 1;
    }
    float Pval;
    float Ival;
    float Dval;
    float Vset;
    int Stop;

    void handle(const struct NDLComHeader *header, const void *payload) {
```

```
        struct representations::Representation *repr =
            (struct representations::Representation *)payload;

        if (repr->mId ==
            REPRESENTATIONS_REPRESENTATION_ID_RepresentationsPIDcontrol) {

            struct representations::PIDcontrol *msg =
                (struct representations::PIDcontrol *)payload;

            Pval = msg->Pvalue;
            Ival = msg->Ivalue;
            Dval = msg->Dvalue;
            Vset = msg->setPoint;

            return;
        } else if (repr->mId ==
                    REPRESENTATIONS_REPRESENTATION_ID_STOP) {

            Stop = 1;

        } else if (repr->mId ==
                    REPRESENTATIONS_REPRESENTATION_ID_RESUME) {

            Stop = 0;

        } else {
            out << "skipit\n";
            return;
        }
    }
};

void sendMessage(struct NDLComNode *node, float velo) {

    struct representations::PIDcontrol msg;
    msg.mBase.mId =
        REPRESENTATIONS_REPRESENTATION_ID_RepresentationsPIDcontrol;

    msg.measureValue = velo;
```

```
    ndlcomNodeSend(node, 1, &msg, sizeof(msg));
}

int main(int argc, char *argv[]) {
    struct NDLComNode node;
    struct NDLComBridge bridge;
    ndlcomBridgeInit(&bridge);
    ndlcomNodeInit(&node, &bridge, 198);

    class PrivateHandler handler(node);

    ndlcom::ParseUriAndCreateExternalInterface(std::cerr, bridge,
                                    "udp://localhost:34000:34001");

    VSens vs;
    PID pid;
    SerialCom sc;
    float pidout, measure;

    //setting the lighting options
    sc.beacon(1);
    sc.control_light(0);

    //enable for logging
    /*FILE * file;
    char filename[512];
    sprintf(filename,"vmeasurement.csv");

    if((file = fopen(filename,"a+")) == NULL) {
      return 0;
    }*/

    while (1) {
      usleep(10000);

      //check for new data
      ndlcomBridgeProcess(&bridge);

      measure = vs.VMeasurement();
```

```
        //std::cout << "V-Measured: " << measure << std::endl;
        sendMessage(&node, measure);

        //enable for logging
        //fprintf(file,"%f,%f,%f\n",cnt*0.03,measure,set);
        //printf("%f: Velocity: %f %f\n",cnt*0.03,measure,set);

        pidout = pid.Pidout(handler.Pval, handler.Ival,
                            handler.Dval, handler.Vset, measure);

        if((pidout > 0) && (handler.Stop != 1))
          sc.serial_communication(pidout, 6);
        else
          sc.serial_communication(pidout, 2);
      }
      //enable for logging
      //fclose(file);
      return 0;
}
```

## D.2. PID-Regler

### D.2.1. pidcontroller.hpp

```
#pragma once

struct Params {
  float kp;         //proportional gain
  float ki;         //integrator gain
  float kd;         //differentiator gain
  float tau;        //sampling interval
  float setpt;      //setpoint
  float highlim;    //upper limit controller output
  float lowlim;     //lower limit controller output
  //float rate_limit;
};

struct State {
  float integral;  //Summation of setpoint errors
```

```
  float deriv;      //Previous setpoint error
};

class PID {
  public:
    PID();
    float Pidout(float p, float i, float d, float set, float measure);
//counter signal
    Params param;
    State state;
};
```

### D.2.2. pidcontroller.cpp

```
#include "../include/pidcontroller.hpp"

//This code mainly is taken from
//http://www.mstarlabs.com/apeng/techniques/pidsoftw.html

PID::PID() {
  param.kp = 4.76;
  param.ki = 1.87;
  param.kd = 0.0;
  param.setpt = 0.0;
  param.highlim = 16393.0;
  param.lowlim = 0.0;
  param.tau = 0.03;
  //param.rate_limit = 0.0;
  state.integral = 0.0;
  state.deriv = 0.0;
}

float PID::Pidout(float p, float i, float d, float set, float measure) {
  param.kp = p;
  param.ki = i;
  param.kd = d;
  param.setpt = set;

  float out = 0.0, err = 0.0, change = 0.0, anti_windup = 0.15;

  err = measure - param.setpt);
  out = err;

  //integrator rate limiting
  //comment in rate limit in params
  //float ichange;
  //ichange = err;
  //if (ichange > pid.params.rate_limit)
  //  ichange = pid.params.rate_limit;
  //else if (ichange < -pid.params.rate_limit)
  //  ichange = -pid.params.rate_limit;
```

```
  out += state.integral*param.tau*param.ki;
  //pid.state.integral += ichange
  state.integral += err;

  //to avoid derivator spikes change err to measure
  change = err - state.deriv;
  out += change * param.kd / param.tau;
  state.deriv = err;

  out *= -param.kp;

  //scaling the controller output
  out /= (16393/0.513);

  //Security functions
  //coping with Limits
  //anti windup reduction factor between 0.05 - 0.25
  if(out >= param.highlim) {
    out = param.highlim;
    state.integral += anti_windup * err;
  }
  else if(out <= param.lowlim) {
    out = param.lowlim;
    state.integral += anti_windup * err;
  }
  //integrator Latching
  else
    state.integral += err;

  return(out);
}
```

## D.3. Sensor

### D.3.1. vsens.hpp

```
#pragma once
#include <wiringPiSPI.h>
#include <iostream>
```

```
#include <stdint.h>
#include <stdlib.h>
#include <iostream>
#include <cstring>
#include <stdio.h>
#include <unistd.h>

class VSens {
  public:
    VSens();
    int SensorRead();
    float VMeasurement();
};
```

### D.3.2. vsens.cpp

```
#include "vsens.hpp"

VSens::VSens() {
  if(wiringPiSPISetup(0, 125000)==-1) {
    std::cout << "Couldn't setup SPI" << std::endl;
    exit(EXIT_FAILURE);
  }
}

int VSens::SensorRead() {
  int len = 2, erg;
  uint16_t resi = 0;
  unsigned char  data[2] = {0,0};
  //reading the sensor
  wiringPiSPIDataRW(0, data, len);
  resi = data[1]|(data[0]<<8);
  //converting grey-code to binary
  for(erg = resi >> 1; erg != 0; erg = erg >> 1) {
    resi = resi ^ erg;
  }
  return(resi & (0x3FFF));
}

float VSens::VMeasurement() {
  int sensorval[2];
  float velocity = 0, v1 = 0, v2 = 0;
  sensorval[0] = SensorRead();
  usleep(20000);
  sensorval[1] = SensorRead();
  v1 = (float)sensorval[0];
  v2 = (float)sensorval[1];
  //in case the sensor passes the 0 position
  //the value gets corrected
  if((v1 - v2) < 0.0) {
    v1 += 16383;
  }
  //if the sensor position has not changed the
  //value zero is returned
```

```
  else if((v1 - v2) == 0) {
    return(0.0);
  }

  //calculation of the velocity in m/s
  //0,001 = 0,00038 rad * 0,0525 m / 0,02 s
  //      = 1 step in rad, radius, sample time
  velocity = abs(v1 - v2) * 0.001;

  return(velocity);
}
```

## D.4. GUI

### D.4.1. Treadmill.cpp

```
/**
 * @file Treadmill/src/Treadmill.cpp
 * @author mbruhn
 * @date 2016-01-22
 */

#include "widgets/Treadmill.h"
#include "ui_Treadmill.h"

#include <QDebug>

#include <iostream>
/* singleton to ndlcom-main-widget */
#include "ndlcom_qt/InterfaceContainer.h"
#include "ndlcom_qt/NDLCom.h"
/* id-database generated from xml-file */
#include "RobotConfig/Devices.h"
/* example representations "Ping" datatype */
#include "representations/Ping.h"
/* payload ids generated from names.c */
#include "representations/id.h"
#include "representations/Commands.h"
#include "representations/PIDcontrol.h"
```

```
using widgets::Treadmill;

Treadmill::Treadmill(QWidget* parent):
    QWidget(parent),
    mpUi(new Ui::Treadmill)
{
    /* set up the parts of the gui which were defined in qtcreator */
    mpUi->setupUi(this);

    mpUi->doubleSpinBoxP->setMaximum(10.0);
    mpUi->doubleSpinBoxI->setMaximum(5.5);
    mpUi->doubleSpinBoxD->setMaximum(15.2);
    mpUi->doubleSpinBoxV->setMaximum(20.0);



    /* for receiving all kinds of messages */
    //connect(ndlcom_qt::getInstance(),
    //        SIGNAL(rxMessage(const ndlcomId),
    //             this, SLOT(rxMessage(const QByteArray&)));

    /* for receiving certain representations */
    connect(ndlcom_qt::getInstance(),
            SIGNAL(rxRepresentation(const NDLComHeader &,
                                    const representations::PIDcontrol &)),
            this,
            SLOT(rxRepresentation(const NDLComHeader &,
                                  const representations::PIDcontrol &)));

    /* sending ndlcom packages */
    connect(this, SIGNAL(txMessage(const NDLComId, const QByteArray &)),
            ndlcom_qt::getInstance(),
            SLOT(txMessage(const NDLComId, const QByteArray &)));

    connect(mpUi->doubleSpinBoxP,
            SIGNAL(valueChanged(double)), this, SLOT(update()));
    connect(mpUi->doubleSpinBoxI,
            SIGNAL(valueChanged(double)), this, SLOT(update()));
    connect(mpUi->doubleSpinBoxD,
```

```
            SIGNAL(valueChanged(double)), this, SLOT(update()));
    connect(mpUi->doubleSpinBoxV,
            SIGNAL(valueChanged(double)), this, SLOT(update()));
    connect(mpUi->stopButton,
            SIGNAL(released()), this, SLOT(stop()));
    connect(mpUi->startButton,
            SIGNAL(released()), this, SLOT(start()));
}


Treadmill::~Treadmill()
{
}

/* for receiving certain representations */
void Treadmill::rxRepresentation(const NDLComHeader &hdr,
                                 const representations::PIDcontrol &data)
{
        if(hdr.mSenderId==198)mpUi->Tacho->setNum(data.measureValue);
}

/* for receiving all kinds of messages */
//void Treadmill::rxMessage(const QByteArray&)
//{}

/* sending ndlcom packages */
void Treadmill::update()
{

    /*{
        representations::STOP pkg;
        pkg.mBase.mId = REPRESENTATIONS_REPRESENTATION_ID_RESET;
    }*/

    /* prepare and set up a message to be sent */
    representations::PIDcontrol pkg;
    pkg.mBase.mId =
       REPRESENTATIONS_REPRESENTATION_ID_RepresentationsPIDcontrol;
    pkg.Pvalue = mpUi->doubleSpinBoxP->value();
    pkg.Ivalue = mpUi->doubleSpinBoxI->value();
```

```
    pkg.Dvalue = mpUi->doubleSpinBoxD->value();
    pkg.setPoint = mpUi->doubleSpinBoxV->value();
    /* and gone! */
    emit txMessage(robotconfig::Treadmill,
                   QByteArray((const char *)&pkg, sizeof(pkg)));
}

void Treadmill::stop()
{
    representations::STOP pkg;
    pkg.mBase.mId = REPRESENTATIONS_REPRESENTATION_ID_STOP;

    emit txMessage(robotconfig::Treadmill,
                   QByteArray((const char *)&pkg, sizeof(pkg)));
}

void Treadmill::start()
{
    representations::RESUME pkg;
    pkg.mBase.mId = REPRESENTATIONS_REPRESENTATION_ID_RESUME;

    emit txMessage(robotconfig::Treadmill,
                   QByteArray((const char *)&pkg, sizeof(pkg)));
}
```

# Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

| Hamburg, 23. März 2016 | |
|---|---|
| Ort, Datum | Unterschrift |