THE UNIVERSITY OF TOKYO

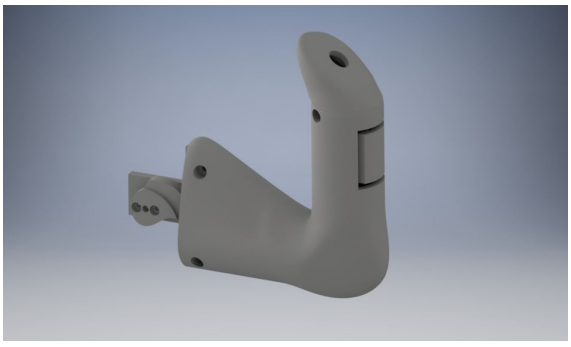
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Haptic Feedback Controller
Palm Pressurization by Series Elastic Actuators in a Handheld Device

Master Thesis

Author:
Roman Oechslin
Department:
Microengineering

Director of project:
Prof. Hannes Bleuler
Supervising professor:
Prof. Akio Yamamoto



August 13, 2018

Haptic Feedback Controller – Palm Pressurization by Series Elastic Actuators in a Handheld Device

Roman Oechslin, Section Microtechnique

Assistant:

Professeurs: Bleuler Hannes et Yamamoto Akio

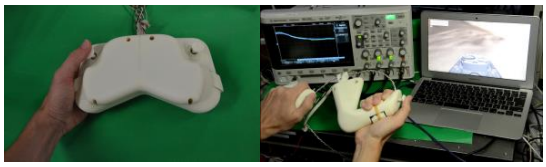
Ce projet est basé sur une recherche à l'université de Tokyo, proposant un appareil haptique, avec lequel l'utilisateur peut piloter un robot mobile. La nouveauté de cette recherche est, que les forces du feedback haptique sont substituées par une pression dans les paumes.

Dans ce travail, le but était de remplacer les moteurs à bobines mobiles par des actionneurs élastique série (AES). Ceux-ci permettent d'augmenter considérablement la force à la sortie. De plus, les éléments élastiques sont capables de stocker de l'énergie, qui résout le problème de dissipation d'énergie et de chauffage dans l'ancien design. Comme désavantage, la réaction de ces systèmes est très lente, dû aux éléments mécaniques.

Dans cet œuvre deux nouvelles conceptions d'appareils portables ont été étudiées et testées, pour prouver la fonctionnalité des AES dans des applications haptiques.

Contrôleur proposé par [Nakamura, et al. 2016]

Le premier prototype a été inspiré par des contrôleurs conventionnels. Après avoir dessiné et imprimé ce prototype, les paramètres majeurs ont été identifiés, analysés et documentés. Ces paramètres étaient en effet la force maximale, la latence de la réponse mécanique par rapport à la référence, la stabilité et la transparence. Basé sur ces performances, un deuxième contrôleur inspiré par un manche à balai d'un avion a été conçu.



Les deux contrôleurs utilisant des AES

Les contrôleurs proposés ont été testés non seulement dans la simulation, mais aussi en utilisant un vrai robot. Pour faire cela, une application avec une interface graphique a été conçue qui traite les messages de communication entre le robot et l'ordinateur.

Les résultats ont confirmé que, dépendant de l'application, l'implémentation d'un système d'actionnement utilisant des éléments élastiques ne risque pas la transparence.

	Requis	Proto 1	Proto 2
Force [N]	>10	10.8	18
Latence [ms]	<160	50	90-180
Taille [mm^3]	<200x 150x 100	220x 110x 70	155x 155x50
Poids [kg]	<0.6	0.55	0.5

Tableau de résultats

Pour soutenir ces résultats, un modèle mathématique a été créé, qui permet de simuler l'appareil et d'effectuer une recherche dans le domaine des paramètres.

Department of Microengineering
LSRO - *Laboratoire de Systèmes Robotiques*
LSRO - *Laboratory of Robotic Systems*

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

	Master Thesis - Summer 2018
Title:	Haptic Feedback Controller - Palm Pressurization by Series Elastic Actuators in a Handheld Device
Candidate:	Roman Oechslin
Professor:	Hannes Bleuler
Supervisor:	Akio Yamamoto
Department:	Microengineering

Short Introduction

In 2016 a haptic feedback controller has been designed at the University of Tokyo, which uses voice coil motors to substitute a force by palm pressurization. Using this controller, one can navigate a tank in a Unity based simulation. The feedback law was based on the roll and pitch angles, giving the user an intuitive feeling of the robot's orientation within the simulation. Even though the current application is the navigation of a mobile robot, the target application shall not be specified at this point, to ensure a certain flexibility.
The voice coil motors (VCMs) can only produce a low output force and create a lot of extra heat when maintaining a certain force. Therefore, a new research project aims at replacing this actuation system by a novel design based on the principle of series elastic actuators (SEAs). These SEAs have the advantage of high output forces and simple control.

Project Objectives and Requirements

In the scope of this project, several controllers implementing SEAs shall be designed, prototyped and tested. Important design-related parameters shall be identified and reported, such that the suggestions can be used as a guideline for future projects.
It is expected that the output force of the controller designs is bigger than 2N, the maximum of the VCMs. The control speed of the actuation system shall be fast enough to ensure a smooth and intuitive feeling, subjective to the user. The weight of the controller does not have to be optimized, but the new designs should not exceed 0.6kg.
The controllers shall be tested in the virtual environment and on a real robot. This requires a certain controller modularity to be able to use it for any kind of robot controllable with 2 input-joysticks. The size of this robot is not specified.

Abstract

This master thesis is based on the work from [1] and further investigates the research in haptic feedback in handheld devices. The previous work has focused on providing a solution for limited information about a remotely controlled robot, for example in a search-and-rescue mission or a hazardous inspection site. The controller provided haptic feedback using palm pressurization as a pseudo-force to the operator. The force is proportional to the inclination, such that the operator can intuitively feel obstacles or rising slopes in the palms. The controller used voice coil motors which could only provide very low output forces. Furthermore, a constant current had to be delivered to maintain a target force. This resulted in high energy dissipation and heated up the system.

In this thesis, an alternative actuation system, based on series elastic actuators, has been studied. Several designs have been proposed and extensively tested. An underlying mathematical model has been created, which helped identifying friction and damping coefficient in the system. To complement the limited sources of literature in this field of study, design suggestions based on the parameter testing have been made.

The requirements of higher output forces with acceptable control speed could successfully be met. The common haptics-related issues of stability and transparency could be overcome.
This work shows that, depending on the target application, an implementation of series elastic actuators can significantly increase the output force while not limiting the control speed. This design can not only be used for remotely operated vehicles (robots and drones) but also handheld manufacturing tools or manipulation setups, where haptic feedback can immensely increase the performance.

Acknowledgements

This project would not have been feasible without the help of various people. At this point, I would like to speak out my gratitude towards them.
I would first like to thank my thesis supervisors Prof. Akio Yamamoto and Prof. Hannes Bleuler. I am thankful for the friendship between the two professors who both made it possible for me to gain this valuable experience in Japan.
The project supervision from Prof. Yamamoto and the constructive feedback and ideas that he has provided were particularly helpful. Together with the fruitful discussions and inputs this project could be realized.
I would also like to thank Prof. Bleuler who has helped me before coming to Japan in matters of preparation and organization, as well as for his correspondence with the Ecole Polytechnique Federale de Lausanne.

I would also like to thank the secretary Mariko Kawamura who has helped immensely in all organizational affairs, such as filling out immigration and residence forms.
Lastly, I would like to thank all the members of the advanced mechatronics laboratory, who have taught me language and cultural aspects. They have welcomed me warmly in their circle and even though this was an exchange program, I almost felt at home. Thank you.

Preface

Between February and August in 2018 I was given the chance to go on an exchange program to the University of Tokyo. Sent by Prof. Hannes Bleuler, I had the opportunity to join the advanced mechatronics laboratory led by Prof. Akio Yamamoto, to work on an applied haptics project.
The University of Tokyo provided a language course throughout the semester, to facilitate the cultural exchange and living in Japan. I was very happy to have come here, not only to learn about the culture, traditions and habits in Japan, but also to gain valuable experience in my domain of expertise.

I first started reading into the previous project and current literature in this field of study. Then a thorough brainstorming session about device designs was followed by the actual 3D-design and 3D-printing phase. With extensive testing I could improve the design and analyze the important parameters.
Then I wrote a small software in Processing to handle the messages and communicate with the robot. Parallel, the Arduino had to be programmed to read out the device's data. To finalize the connections, I had to solder and design small circuits and implement electric filters.
A thorough frequency response analysis was conducted for two different devices and their results have been used to create a mathematical model. The data has been analyzed and represented writing python scripts. The mathematical model could be used to simulate the setup and identify parameters in Simulink.
On the control part, a PID gain tuning was necessary. Then the devices have been put to a test with a real robot, as well as a simulated environment. The performance has been evaluated and reported in this thesis.
For future research, a brief section of suggestions has been written, which can be used as a guideline for similar projects.
A major challenge was the fact that the robot's user manual and documentation was written in Japanese. Hardware-related issues were mainly encountered with the robot's battery that had to be removed and an alternative power supply had to be used.

In this project I gained valuable experience in all aspects of robotics. Working individually and autonomously, I could improve my practical skills and technical knowledge. Furthermore, the organized lab-internal presentation meetings helped me in presenting my work to a group of fellow researchers and communicate encountered problems. All in all, I had a lot of fun with an interesting project and great laboratory students, creating a healthy work environment, while at the same time, discovering a wonderful country.

Contents

check completeness of appendices
make dots bigger in bode plots
reread design suggestions

1 Introduction

1.1 Haptics

The Oxford Dictionaries define haptics as

> "Relating to the sense of touch, in particular relating to the perception and manipulation of objects using the senses of touch and proprioception." [2]

It dates the term back to the late 19th century and states its origin from the greek words *haptikos* meaning 'able to touch or grasp' and *haptein*, 'fasten'.
In engineering, the research in the haptic field has started to emerge in late nineteen-eighties [3]. With Thomson Reuters' *Web of Science*[1], one can find the number of papers published, containing the keywords *'haptic feedback'*. The graph can be seen in figure 2 and clearly shows the increase in popularity of this research topic.

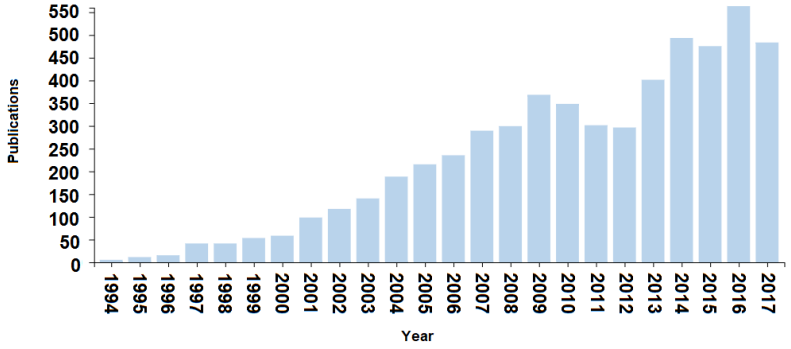


Figure 2: Trend of published papers containing the keywords *'haptic feedback'*.

Before that, man-machine interaction was mainly limited to keyboard and mouse input, which is rather unidirectional and passive and it soon has become clear that this requires a rather skilled user for all kinds of operations. This directly leads to limitations in performance. Haptics can extend this unilateral interaction by providing tactile or kinesthetic information. This combination can overcome the user's limitations and improve performances of high-precision tasks or high-force tasks drastically.
In [4] it is stated that, even though the research in the haptic domain has significantly increased in the past ten years, further investigations are necessary for the "quest for realism", especially in medical telesurgery applications where realism is key to performance. Other application domains include but are not limited to space operations, manufacturing, physical rehabilitation, arts or simply entertainment related devices.
The critical elements for stability of haptic setups, mainly focusing on haptic simulations are discussed in [5]. It also motivates exploration of alternative control techniques due to the unpredictability of the human operator and the environment model.

[1] accessed (2018, July 26th), `https://apps.webofknowledge.com/WOS_GeneralSearch_input.do?product=WOS&search_mode=GeneralSearch&SID=C3bYefMpEMD84LZwaAU&preferencesSaved=`

1.2 Major Issues in Haptics

In general, the issues in haptic devices are related to the stability and the transparency. On one hand, high transparency is desired, on the other hand however, this can only be achieved by jeopardizing the system's stability[6].
In literature ([7], [8], [9]) the importance of stability and transparency is often emphasized, stating that they are the main problems related to haptics and haptic teleoperation applications. For conventional master slave operation systems Salcudean explains transparency as

> [...] the master should feel to the operator as if the task were being manipulated directly.[7]

Even though the controller does not meet the definition of a conventional master-slave teleoperation setup, the problem settings of stability and transparency still apply. Often, the stability issues arise from master-slave mass mismatch and stiffness of the environment. But high inertia also poses problems for higher frequencies. In this research however, the masses and frequencies are relatively low.
The stability of this controller is not only affected by the control scheme and the mechanical setup, but also by the operator, whose grasp can render a system stable or unstable [9]. Additionally, communication delays can also cause instability.

Another common issue in haptic application is the fact that counteracting forces are created. In conventional designs, these forces are dealt with, by using grounded devices. If the device is handheld however, the counteracting forces will also affect the user and their effect need to be taken into consideration.

The previous research aimed at developing a new design of haptic feedback systems that can overcome these issues. In the handheld device, the reaction forces are transmitted to the user. However, the research group has presented a method to deal with those reaction forces, such that the operator's perception of the target forces is not distorted. Furthermore, the problem of stability is overcome by decoupling the input and output elements, which yields a stable open-loop system. A detailed explanation of the previous research is given in the following paragraph.

1.3 Project Description and Status Quo of Previous Projects

Research on a handheld controller has been conducted at Yamamoto's lab in 2016 and 2017 ([10], [1], [11]). This controller is capable of giving a feedback to the user about the state of a remotely controlled robot. As stated in the papers, the feedback law is based on the roll and pitch angle of the robot. For the feedback two voice coil motors (VCMs) have been used. The disadvantage of these motors is that a constant current has to be applied to maintain a constant output force. This leads to an energy-inefficient system and produces a lot of extra heat. Furthermore, the motors were rated at 13.6V with 2N/A. The output force of each palm stimulator was exactly 2N. Due to the magnets in the VCMs the controller was rather heavy, which could fatigue the operator if used for too long.
The advantage of these motors however, was the speed at which they were able to feed the force back to the operator. Due to the fact that the output force of the voice coil motors depends on the current, the speed of this system is governed by the amplifiers that convert the reference voltage to a target current.
The testing environment for this controller consisted in a Unity program that let the user control a tank with crawlers in an artificial landscape. When driving over the hills and smaller bumps, the user feels the magnitude of the tilt (roll and pitch orientation angles) as a pressure in the palms. It has already been shown in the previous work that this pressure successfully substitutes the force and is perceived as a pseudo-force.

Figure 3: Developed controller in previous research[11].

In figure 3 one can see the final controller and its testing environment. The joysticks are used to control the navigation commands. In this case, the tank's crawlers can be controlled individually in both directions and the speed is linear to the joystick's position. The simulation renders the robot in a hilly environment, including reaction and gravitation forces. From the orientation (pitch and roll) the desired feedback (the forces that should be felt by the operator) are calculated and sent to the controller. The law is such that if one is driving over an obstacle on the right-hand side, one can feel a higher force on that side.
The feedback is read out by an Arduino (not shown in the figure) and controls teh voice coil motors (VCM) with the help of an amplifier (not shown in the figure). These motors will move the stimulator back and forth, in a uni-lateral direction. This direction can be fine-adjusted by the hinge in the middle of the controller. The stimulators are in direct contact with the operator, creating a pressure in the palms. They are henceforth also referred to as the palm pads.

1.4 Target Application

The Fukushima Daiichi nuclear disaster in 2011 was a shock for the whole world. The radioactive environment leaves a wide area inaccessible. However, an on-site inspection can give important information for all kinds of research fields. These inspections have to happen using remotely controlled robots.
A common feature is to provide a visual feedback with an on-board camera, as it is used in the Inachus project[2], for example. However, in earthquake affected areas, the view can be obstructed due to smoke, dust and rubble[12].
Consequently, the feedback controller designed in this lab can be used for facilitating navigation in such environments. The intuitive and straightforward feedback method yields higher performance with no trade-off.

1.5 Psychological Aspects

The device that has been developed in the previous research[11] feeds back a force to the user. However, according to Newton's third law, every action force induces an equal reaction force. This means that the user should feel a pushing force on the palms, directed towards the user, as well as a pulling force on the fingers, away from the user. But the research with this device has also shown, that this reaction force is felt much less by the user due to the non-linear force sensation characteristics of humans.
Due to the fact that the reaction force is spread over the whole area of contact with the controller, the pressure is much lower, even though the magnitude of the forces stay the same. This means that the user only believes to perceive a pushing force, which allows to give a more intuitive experience. Thus the ubiquitous problem in all haptic applications of dealing with the reaction forces can be overcome. This means that the device does not have to be grounded in order to dissipate the reaction force, but can be handheld.

[2]accessed (2018, August 8th), `https://www.inachus.eu/`

1.6 Project Outline

In order to overcome the weaknesses and disadvantages of the voice coil motors, a novel actuation system shall be investigated. In the scope of this project, several haptic feedback controllers shall be designed, implementing series elastic actuators (SEAs). These controllers shall be prototyped and tested in the Unity simulation, as well as on a real robot. The tests of the controller shall be made on the commercially available crawler robot developed by topy[3] using two tracks as means of transportation. In addition, different springs and spring arrangements shall be analyzed.
The performance of the controller is expected to have a higher output force than the one from the previous research, which implemented voice coil motors. It is namely 2N per palm pad. It should be shown, whether or not the series elastic actuation principle is the bottleneck of its implementation in terms of control speed and therefore transparency and real-feel.
The controller shall be capable of providing intuitive feedback (defined in section 1.9) to the user. The weight does not have to be optimized, but it should certainly not exceed the weight of the current controller (ie. 0.6kg).
From the results a design parameter guidance in form of suggestions shall be written, that future research can use as a reference and guideline.
The given time for this project was from February until August 2018, a total of 25 weeks.

1.7 Performance Requirements

Here, the performance requirements from the previous paragraph are summed up and a rough reference value is calculated. It seemed reasonable to have an output force of 10N, a value five times higher than in the VCM-based controller. This corresponds roughly to 1kg against gravity.
The control delay was based on the research of [13], which suggested a maximum arm movement frequency of 6Hz. With the target application of an inspection robot in mind, the low operating speed justifies the comparably small delay time.
The maximum size of the controller has been calculated as 110% of the size of the controller from the previous research.
A quantitative requirement on transparency cannot be given at this point, however an intuitive feeling related to the transparency is expected. Furthermore, the stability of the controller has to be guaranteed.

Designator	Value	Unit
Output force	10	[N]
Control speed	–	[full stroke/s]
Response delay	160	[ms]
Maximum size	$200 \times 150 \times 100$	[mm×mm×mm]
Maximum weight	0.6	[kg]

Table 1: Performance requirements of the controller.

1.8 Approach and Expected Challenges

As an initial approach and to be consistent with the previous research in this area [1], the robot's orientation (pitch and roll angles) have been taken as state of interest and are fed back to a handheld controlling device. As an alternative measure, the current in the two crawler motors can be fed back to the user, to have a rough idea of the used energy. With feedback on the current consumption it is also detectable if the robot is stuck or being blocked somewhere.
Note at this point, that in this work *feedback* is referring to the force that is determined by the robot's state and created by the controller, acting on the operator. The term feedback is generally

[3]accessed (2018, May 25th), `http://www.topy.co.jp`

avoided when talking about the closing of the control loop that is necessary to achieve or maintain the aforementioned force or control variable, for that matter.
Current literature and research projects show little to no guidance on how to build an SEA based haptic feedback device. Furthermore, the human hand, acting as environment, has uncertain mechanical parameters. Due to this, the design of the controller and the implementation of the SEA system tend to be rather iterative processes. Furthermore, the design of the previous controller cannot be recycled, since the actuation system differs completely.
In addition to that, a lot of parameters are present in the controller. These include but are not limited to the mechanical parameters, such as the springs and spring constants, the motors, feedback direction and the software and electrical parameters, such as filters, control scheme and control frequency.
The performance evaluation of such a controller can be manifold. Depending on the target application, one might favor low-latency and fast feedback over high output force, or vice versa. Therefore, it is important to keep the performance specifications of the target application in mind when deciding on design of the controller and when evaluating. Concerning the stability issues (detailed explanation in section 1.2), it is not always possible to fully classify a given controller and setup into stable and non-stable, since it might also depend on the user's perception and handling. Also, an assessment of the intuitiveness and transparency can only be done subjectively.

1.9 Literature Review

The major applications for haptic feedback devices can be found in (minimally invasive) surgical teleoperated medical systems ([9], [14], [15], [16]), mobile augmented reality ([17], [18]) and teleoperated robotic systems (for example in space) [8]. However this research project focuses on implementing a feedback on a handheld controller for a non-specified group of grounded wheeled robots. The controller uses pressurization feedback as a substitution for force feedback to give an intuitive feeling about the intrinsic state of the robot. In [19], intuitiveness is said to be achieved when the user immediately knows what the system's intentions are.
The medical domain with its minimal invasive surgery (MIS) and teleoperated system with haptic feedback is a common research field. The research group in [9] states, that the absence of haptic feedback is the reason that impede further spread of surgical robots. They give a broad overview of current research in this field, focusing not only on different haptic feedback methods and interface types, but also on control schemes and the device types.
The aspects of minimal invasive surgery are also explained in [14]. Furthermore, it proposes the design of the master and slave in such a teleoperated setup. It uses force feedback and is dealing with a high degree of freedom (ie. DOF = 5). The device is a handheld pen that is connected to a robotic operator, which can change the pen's orientation.

Research on haptic feedback in handheld devices has been conducted not only with force feedback ([20], [21]), but also with pressure stimulation ([10], [11]), [22] or vibrotactile feedback ([22], [23]).
[20] investigated the substitution of kinesthetic and cutaneous force feedback by cutaneous feedback only. They used a wearable finger stretching device for the task of needle insertion. In [21] however, a vibrotactile force feedback system was designed and evaluated. Its application domain is in minimal invasive surgery (MIS), where feedback can facilitate surgical operations. The output device sent vibration signals to the operator's foot.
A custom-made pressure cuff was designed in [22] that could give feedback when manipulating an object with a prosthetic hand. Again, the feedback device is decoupled from the main interaction device which can be seen as a disadvantage. In addition to the pressure feedback, the SoftHand uses vibrotactile feedback to inform the user about the surface properties.
In the work of [23], the research group mainly focused on having an intuitive feedback in a game controller for a flight simulator. The intentions were to be low-cost and versatile. The vibrotactile feedback is coming from the controller itself, which should make the users feel as if they have become the aircraft itself. The device is called the haptic glove, again in the domain of wearable

haptics. An evaluation and usability study of this device has not been conducted.
In [24] a wearable haptic display is proposed, using the principle of deforming and stretching the fingers' skin. It can generate vertical and shearing stress to the finger to give the impression of holding an object with a certain mass and to give perception of a pseudo grip force.

In the domain of series elastic actuators, [25] introduces the suggestion of using an elastic element in a normal actuation system. It states its advantages for robots performing natural tasks and presents a control system for force or impedance control. Among the advantages are the low-pass filtering of shock loads, facilitation of force control, the increased achievability of stability and possibility of energy storage.
The principle of SEA is integrated in a legged robot and explained in [26]. The paper suggests to implement this actuation system in cases where little is known about the environment, such as with legged robotic walking, wearable exoskeletons and also haptic interfaces. Then it introduces other research projects where series elastic actuators have enabled legged robots to walk.
In the research of [27], an electric series actuator is compared with a hydraulic SEA. The target application is a four-legged walking robot. The paper also provides a detailed analysis of the dynamics of the actuator, taking into account the motor parameters as well as the mechanical setup. The results are purely theoretical, where different frequency responses are compared. It shows a cut-off frequency around 30Hz for the investigated actuation design.
A variant of the SEA is the rotary elastic actuator. The impedance control of such an actuator is discussed in [28]. Again, the frequency response function has been used for performance analysis.

The joint topics of haptics that implement the series elastic actuators show only little research. For one, there is the implementation of SEA in a laparoscopic haptic device from [29]. The device has 4 degrees of freedom. In the frequency response analysis both, the experimental and analytical data show a force bandwidth of around 12Hz. They considered damping coefficients up to 1Ns/m for the springs.
In the work of [30], they address the control challenges of a rehabilitation robot based on SEAs. The conditions guaranteeing stability and passivity of such a haptic device are investigated in their paper. The analysis is applied to parallel robot structures with added elastic elements.
In [31] an educational SEA based robot (a haptic paddle) is studied, for understanding human robot interaction under specific conditions. It is used to teach students about closed-loop force control. The paper shows the dynamic model of the setup and analyzes it. The cut-off frequency is around 12Hz.

From this literature review it can be seen that a lot of research has been done for haptic minimal invasive surgery devices. Also, the domain of wearable haptics seems to be very popular. However, research in handheld haptic feedback devices where the device itself is producing the feedback are more scarce. Even more so, when the feedback method is a pressure based force substitution.
Coupled with the implementation of the series elastic actuators, this work becomes a novelty in both research fields combined.

1.10 Thesis Delineate

This thesis sums up and represents the methods and results of the work during one semester. The thesis' structure is kept in chronological order, where it makes sense. Due to the high innovation factor of the research and the fact that there is only little research from literature that can be used as base knowledge, the work turned out to be very practice oriented and iterative. For this reason for example, the first controller prototype was built and analyzed before a mathematical model could be created. Chapter 2 explains the working principle of SEAs and the general design phase of the first controller. Discarded and elected controller designs are introduced. Electrical and mechanical components of vital importance are explained.
Following this design introduction, and to keep this thesis' structure consistent with general re-

search papers, the mathematical model is presented. The mathematical analysis has mainly been applied to the first controller but can be extended to be used as a tool for identification of general controller design parameters. This chapter shows the equations of motion that have been used to create the model and show the implementation in Simulink for the analytical results. For reasons of completeness and simplicity, the experimental results of the first controller are also shown in the same chapter.
Chapter 4 introduces the robot that has been used for testing and explains the environment for the test setups.
The next chapter talks about how the controller has been tested and depicts the performance. It explains major decisions that influenced the design of the second controller and discusses the results to some extent.
Chapter 6 is discussing design suggestions, based on the results of the design and testing of the first controller. The results of the second controller have also been considered, albeit to a smaller degree.
The next two chapters, chapter 7 and 8 discuss the design phase and testing of the second controller, respectively. The design choices are based on the findings of the performance about the first controller. The testing was conducted in the same manner as with the first controller for the sake of direct comparison to the first controller.
The discussion of the project and results can be found in chapter 9, and a final conclusion is given in chapter 10.

2 Design Phase of the First Controller

2.1 Principle of Series Elastic Actuators (SEA)

The principle of series elastic actuators is relatively simple. The concept is explained in [25] which also mentions a more accurate and stable force control and the possibility of energy storage. An SEA is a normal actuation setup with an integrated elastic element. For this purpose, a simple spring or a set of springs can be used. As it can be seen in figure 4 the control scheme is reduced to a simple distance-based control, while the output remains an equivalent force, given by Hooke's law.

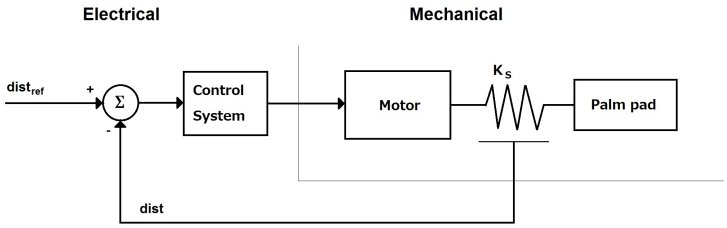


Figure 4: Control scheme for the haptic feedback system.

An additional advantage is, that the elastic element is shock-absorbent and stable for different environments. This means that the controller can be stable even when used with different grip stiffnesses.
A thorough study of an SEA design and its analysis has been made in [27], stating that the most important parameter in the system is the spring constant. Therefore, springs with several different spring constants have been used for the analysis of this research project.

2.2 Discarded Designs

Before the first controller could be designed, a suitable implementation of this actuation system had to be thought of. For this purpose, a brainstorming for actuation methods has been done. Then the individual design suggestions were analyzed and carefully evaluated, with minor calculations for speed and output force. The major discarded designs can be seen in the following sections.

2.2.1 Car-Jack

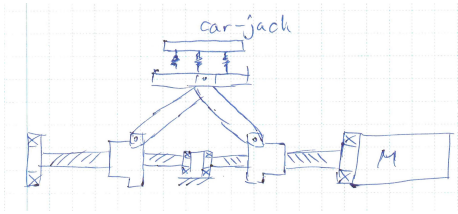
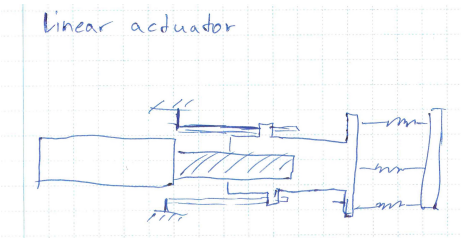


Figure 5: Discarded idea: Car-jack.

In the lab there was a different kind of controller from a previous project that used this kind of system as actuation.
The previous work has shown that using voice coil motors, this shape for a handheld controller is appropriate, albeit with a low output force. The continuation of that project was that the VCMs have been replaced by an SEA system. The results showed that the controller was too slow to provide an intuitive feedback, due to the small pitch of the screw.
Even though it has been proven to work, due to its large size and slow response, this idea has been discarded.

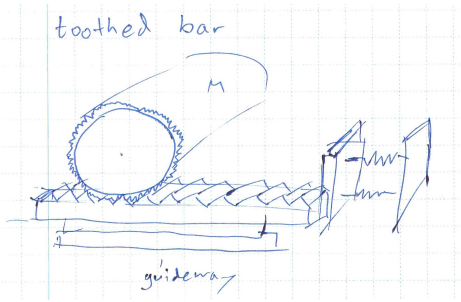
2.2.2 Linear Actuator



This is a rather straightforward approach that uses the same principle as a bench vice. The advantage is its simplicity. However, this system might turn out to be too large, especially if one considers a perpendicular feedback force towards the user.

Figure 6: Discarded idea: Linear actuator.

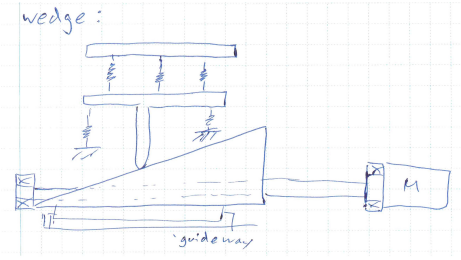
2.2.3 Toothed Bar



Here the principle is a rack and pinion approach where the rack is seen as a carriage in a guideway.
The advantages are the orthogonal actuation, that can solve eventual space issues. Furthermore, the speed can be adjusted easily by choosing a different set of rack and pinion up to the point, where only few rotations are needed for the full desired stroke. The disadvantage is the eventual pre-load that is needed.

Figure 7: Discarded idea: Toothed bar.

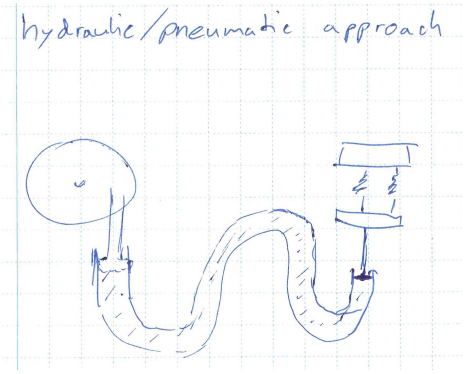
2.2.4 Wedge



The wedge system is similar to the cam-follower. Here the carriage is a wedge that simply pushes the palm pad out of its way. On the downside, there is a lot of friction involved and a pre-load was necessary. Even though the angle of the wedge varies the speed of the system, the achievable stroke was too small. Furthermore, a guideway for the wedge and a guiding system for the actuated element would have been needed.

Figure 8: Discarded idea: Wedge.

2.2.5 Hydraulic or Pneumatic



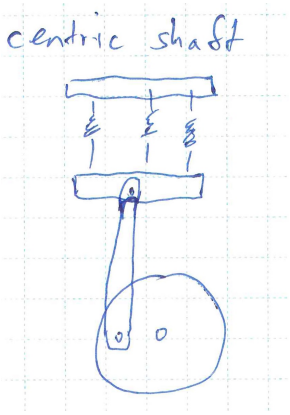
This idea was mainly for completeness of the suggestions. The disadvantages are the difficulty in control and the hassle of handling the hydraulic or pneumatic issues. Even though it can be implemented in practically any volume and shape, it seemed too difficult to further investigate.

Figure 9: Discarded idea: Hydraulic or pneumatic.

2.3 Accepted Designs

After some calculations and evaluation of the proposed designs, the following designs have been opted for.

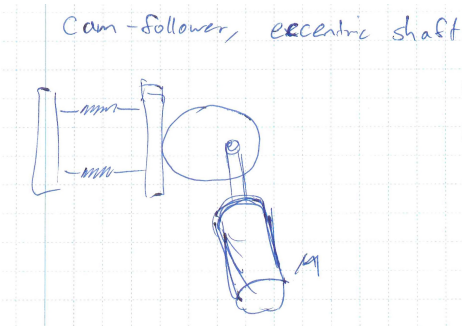
2.3.1 Centric Shaft



In this design, the motor is attached to a rotary disc or rotary lever, that pushed a carriage in its guideway (not depicted). This solution seemed relatively easy, even if it involves an element that corrects the parasitic movements.
The motor can be assembled orthogonally to the feedback direction, which can be advantageous for solving space issues. The speed and output force can be varied easily by changing the length of the lever. Due to the singularities in the system, the motor shaft has to turn less than 180°. With this, the whole system can be quite fast.

Figure 10: Accepted idea: Centric shaft.

2.3.2 Cam-follower with Eccentric Shaft



This design is based on a conventional cam-follower principle, but with an eccentric shaft. The main disadvantages of this implementation is the high friction involved and the required torque being rather high. Since the motors have already been chosen, there was no margin on that aspect. The advantages are however, that only very small movements are necessary and the system would be rather fast. The space occupation and volume of this system was to be further investigated.

Figure 11: Accepted idea: Cam follower with eccentric shaft.

2.4 Inspiration of the Design

It is not far-fetched to search for inspiration in the domain of entertainment, if a user-friendly handheld controller is needed. The probably most famous controller is the one designed by PlayStation [4], which can be used for a variety of applications. Due to this flexibility and the simple but elegant design, this work has opted for a similar design and used the PlayStation controller, as well as the design of the previous research as models. From now on, this controller design will be referred to as the game controller.

2.5 Implementation

After evaluating these designs, the centric shaft design has been tackled at first. It was designed as a clamp link based linear guideway controller (see figure 12), since it seemed to fit best into the casing designs from the previous work. Also, the calculated force of 14N was promising. The singularities in the design restrict the movement of the clamp link to 90° and it was furthermore calculated, that the maximum stroke could be achieved within 100ms.

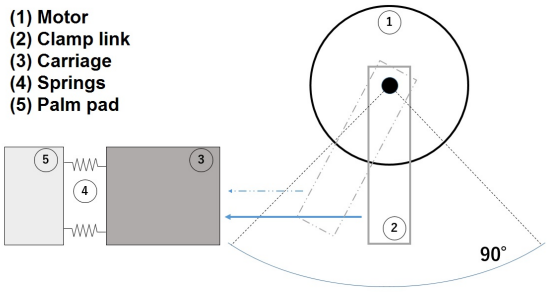


Figure 12: Initial schematic of the actuation design.

2.6 Design and Prototyping

The 3D design of the PS controller can be seen in figure 13.
This design of the controller aims at having a natural position for the hands, such that the user can hold the controller for a long time without having the hands in an awkward position. The palm

[4] accessed (2018, July 23rd), `https://www.playstation.com/en-us/`

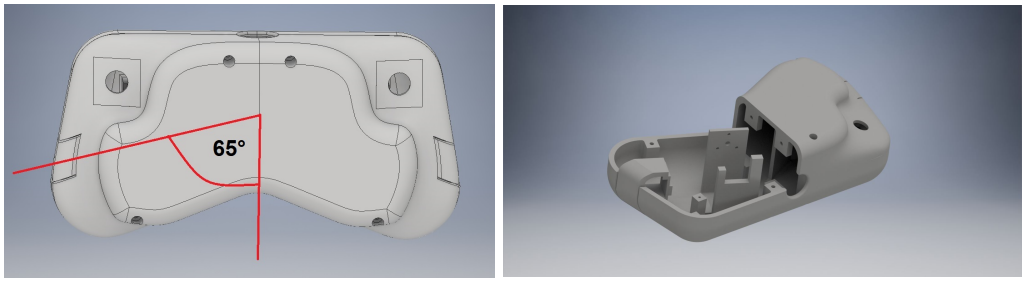
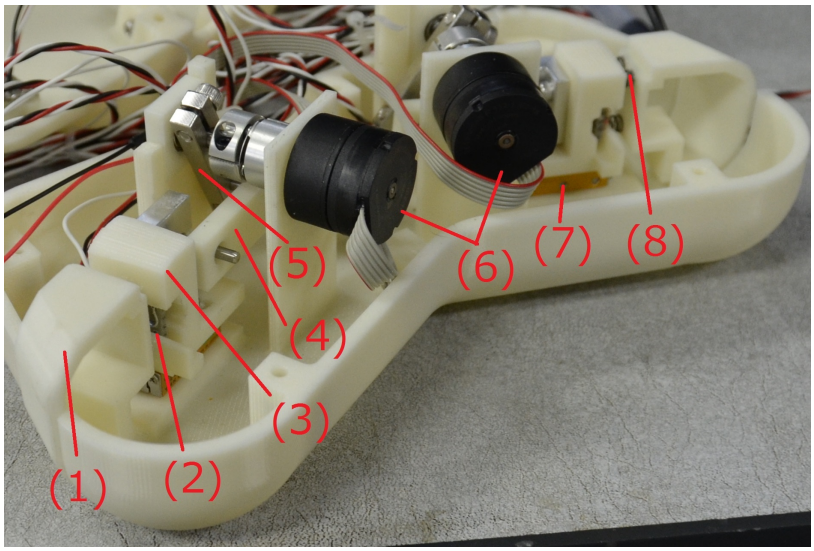


Figure 13: 3D model and rendering of the game controller.

pads that transmit the feedback pressure to the palms are not directed perpendicularly towards the user, but rather outwards at a certain angle (refer to figure 13). This can be seen as a drawback, since the user is expected to have the most intuitive feeling with a feedback opposing the driving direction (ie. perpendicular).
The dimensions of the controller are 220mm in length, 110mm in width and 70mm in height. The controller is unusually thick due to the feedback actuation design. The assembled actuation system can be seen in figure 14. The motor shaft is rotating a clamp link which is attached to a 3D-printed part, called connecting link. This 3D-printed part is attached to the carriage, also a 3D-printed part called L-plate, and is screwed to the linear guideway. The L-plate is connected via a set of springs (different spring constants have been tested) to the palm pad, thus making it an SEA. The linear guideway keeps the palm pad in a linear motion.



- (1) : Palm pad
- (2) : Photoreceptor
- (3) : L-plate
- (4) : Connecting link
- (5) : Clamp link
- (6) : Motors
- (7) : Linear guideway
- (8) : Springs

Figure 14: Actuation system with legend (and top cover removed).

2.7 Electrical Components

To have a functional controller, several requirements had to be met. First of all, an actuator was necessary to provide the feedback. In addition to that, the user shall be able to navigate the robot, which has been realized with potentiometer-based joysticks. To close the loop and create the desired force output, it was furthermore necessary to measure the compression of the springs. This has been realized with a simple photoreceptor as a distance sensor.

2.7.1 Motors

The motors play an important role on the output force of the controller, as well as the achievable control speed. There were motors with two different reduction gear ratios available in the laboratory. They stem from Faulhaber's series called **2619 024 SR IE2-16** and their reduction ratios were *33:1* and *112:1*. The maximum intermittent output torques are 100mNm and 180mNm respectively. Due to the reduction stages, the efficiency is around 60%. The maximum applicable voltage is of 24V, for motor protection however, the motor voltage has been limited to 20V during normal operation.

2.7.2 Joysticks

The joysticks are conventional 2-axes potentionmeters. The series number is **P-04048** and they have been bought at Akizukidenshi store. In the implementation of these sensors, a dead-zone has been created, such that the joystick value is converted to a zero-speed command, even if it is not exactly in the middle position. This can be seen in figure 15.

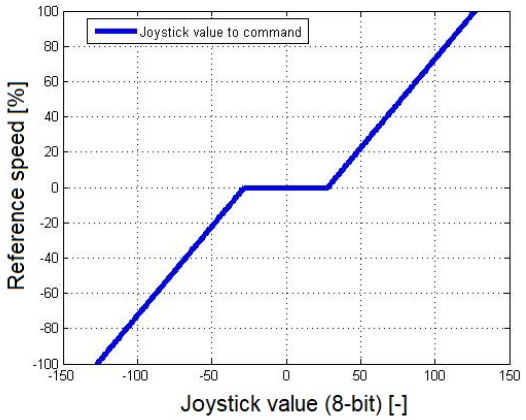


Figure 15: Joystick to command conversion with indicated dead-zone.

2.7.3 Photoreceptors

The photoreceptors are the **TPR-105** from *GENIXTEK CORP*. The circuit can be seen in figure 16. The components chosen are $R_1 = 330\Omega$ and $R_2 = 27k\Omega$.

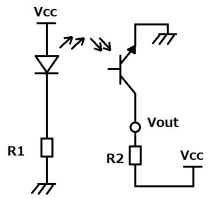


Figure 16: Photoreceptor circuit.

The photoreceptors' working principle is based on detecting the amount of reflected light. This controls the base current i_B of the transistor in the schematic. This base current then determines the collector current i_C. Specific to this setup, the distance and therefore the output force controls the amount of light that is reflected on the wall of the palm pad. Therefore, we have:

$$F_{output} = k_{eq}\Delta x \propto i_B \tag{1}$$

$$i_C = h_{FE} i_B \tag{2}$$

$$V_{out} = V_{CC} - h_{FE} R_2 i_B = V_{CC} - K R_2 \Delta x \tag{3}$$

Where h_{FE} is the forward current gain and K is a constant given by $h_{FE} k_{eq}$.
The two resistor values have been empirically found to have the highest sensitivity but not saturating the measurement. The sensitivity decreases with a smaller resistor, since at a certain point, the Arduino cannot detect a change in voltage anymore. Saturation occurs, when the value of R_2 is too high. Also, it has been opted for keeping the operational distance in the linear range of the sensing position characteristic (see datasheet in appendices).

Identification of Operational Range (Photoreceptor) To find the receptor values at maximum compression of the springs, a simple test has been made, where one time 0V has been applied to the motors, and another time 20V has been applied. The results can be seen in figure 17.

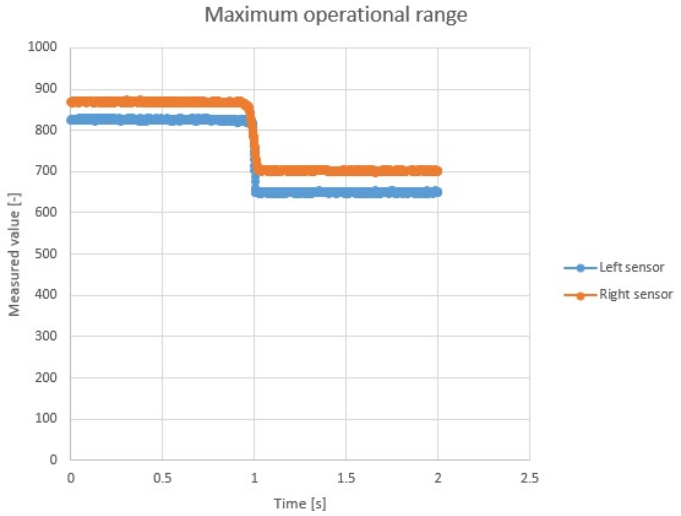


Figure 17: Operational range of photoreceptors with springs at rest and in full compression.

The values that have been found to be the limiting values are resumed in table 2.

	Sensor reading MAX (rest)	Distance (rest)	Sensor reading MIN (compression)	Distance (compression)	Max output force
Left side	830	2.35mm	650	4mm	9.9N
Right side	870	2.2mm	700	4mm	10.8N

Table 2: Identified operational range for the photoreceptors in the game controller.

Using these values, the photoreceptor measurements can be mapped to this range with an 8-bit value. This means that 0 is no compression and 255 is maximum achievable compression. In the Arduino software these values have been slightly adapted (see table 4), to ensure a natural feeling, even when the sensor has some peak values due to noise.

Voltage Interference In the test setup for the frequency response analysis (explained in section 5.3), the Arduino was directly connected to a wave generator. This had an interference with the photoreceptors, as their voltage level was strongly correlated with the input voltage. In order to uncouple the measurement from the input, and to stabilize the measured voltage, a voltage follower with unitary gain (operational amplifier, **LF-412**) has been implemented. Thence, with inactive control, no correlation between input and output has been measured.

2.8 Mechanical Components

From a mechanical point of view, the spring implementation is the most essential part. However, there are other components that have an impact on the output force, the clamp link for instance, whose length acts as a lever. Other minor but non-negligible components are the linear guideway and the coupling.

2.8.1 Springs

The springs are the most crucial part of the entire system. Not only is the output force determined by the spring constant and the compression rate, but also the length and arrangement of the springs have a major impact on the feeling of the controller. Intensive testing has shown that the arrangement has to be symmetric with preferably uniform springs. If springs with two different spring coefficients are used at the same time, the compression becomes rapidly asymmetric and the feedback feels unnatural. Also, if the springs are too long, then a bending into one direction occurs, which causes a buckling effect, felt by the user. This may even result in contact with the casing of the controller and therefore friction or blocking.
For the final controller design, a total of four springs (**WT4-5**), arranged symmetrically on the palm pad have been used. The springs have a spring constant of $k_s = 1.5\text{N/mm}$ which corresponds to an equivalent spring constant of $k_{eq} = 6\text{N/mm}$. They are distributed around the palm pad, where in their middle the distance sensor has been attached, to measure the distance between the L-plate (carriage) and the palm pad. It therefore measures the compression of the springs, which can be related to the output force by Hooke's law as:

$$F = k_{eq}x \tag{4}$$

The setup of the carriage with the springs and the palm pads can be seen in figure 18.

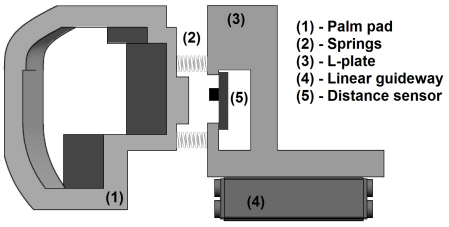


Figure 18: Schematic of the spring system setup.

2.8.2 Linear Guideway

The linear guideway is the model **LS-827** from *THK*. Its maximum stroke is 13mm, which is enough for compression the springs, as well as indenting the operators palms to a certain extent. This guideway restricts the movement of the carriage to a unidirectional one.

2.8.3 Coupling

The coupling (**SCPS16-3-5**) and its functionality are straightforward.

2.8.4 Clamp Link

The clamp link acts as a lever pushing the carriage in its guideway. Different lengths have been tested, but the model in the final design is **CLKWS5-3-20**. It converts the rotational motion of the motor shaft to an almost linear motion on the other end. To compensate for the parasitic motions, an additional element called connecting link (see figure 14) has been designed.

2.9 Printed Version

The printed and assembled game controller can be seen in figure 19.

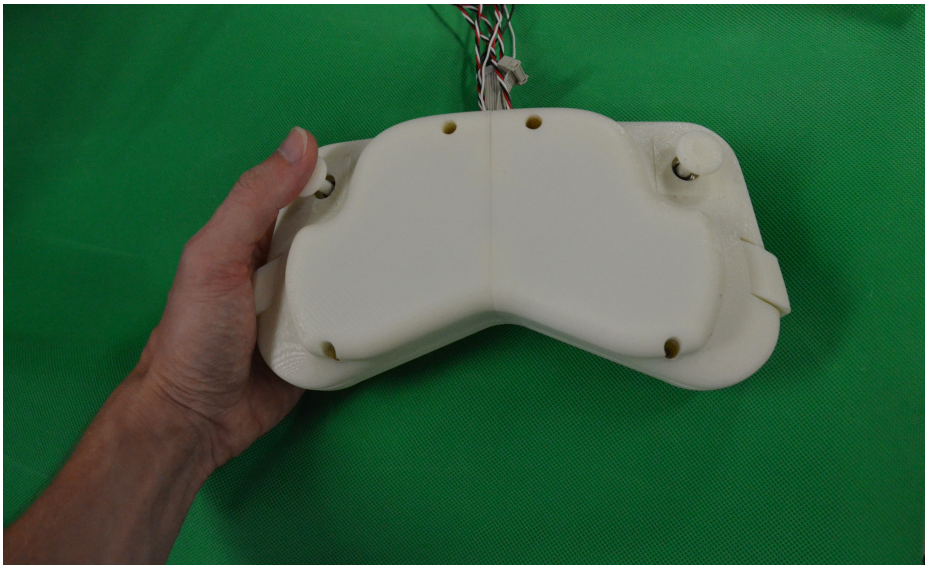


Figure 19: Printed and assembled game controller.

The controller was assembled with 6 screws and nuts and contains the two motors with the complete actuation system.
Once all the design choices and parameters have been fixed, one can create a mathematical model, in order to back the experimental data from the testing phase with the theory.

3 Mathematical Model of the Game Controller

Not only is it interesting to have an underlying mathematical model of the setup for a thorough parameter search, but it can also help to identify important setup parameters that cannot really be measured, such as the operators stiffness and the skins damping coefficient.
To come up with a theoretical analysis of the transfer function, a simplified mechanical schematic has been drawn. This schematic can be seen in figure 20.

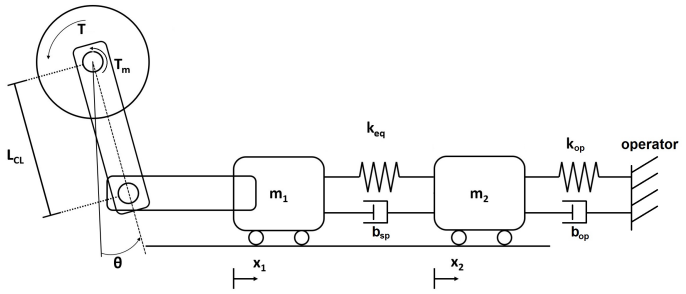


Figure 20: Simplified mechanical schematic of the actuation system with the stimulator (m_2).

The equations of motion can be formulated with the major parameters defined in the schematic. A full explanation of all parameters can be seen in table 3. The variables with subscript 1 refer to the first mass element, the carriage in its guideway, whereas variables with subscript 2 refer to the stimulator, also known as the palm pad. For the motor, the subscript m has been used.

Designator	Explanation	Unit
T_m	Motor torque	[Nm]
T	Output torque acting on the carriage	[Nm]
θ_m	Motor shaft angle (before reduction gear)	[rad]
θ	Clamp link angle	[rad]
L_{CL}	Clamp link length	[m]
m_1	Mass of the carriage in its guideway	[kg]
m_2	Mass of the stimulator	[kg]
x_1	Position of the carriage in its guideway	[m]
x_2	Position of the stimulator	[m]
k_{eq}	Equivalent spring constant	[N/m]
b_{sp}	Spring damping coefficient	[Ns/m]
n	Reduction gear ratio	[-]
k_{op}	Spring constant of the operator	[N/m]
b_{op}	Damping coefficient of the operator	[Ns/m]
J_T	Total inertia of mechanical setup	[kgm^2]

Table 3: Setup parameters

3.1 Assumptions

First of all, it is important to mention that the transfer function is non-linear, due to the motor angle θ_m that determines the force acting on the carriage with mass m_1. As an initial approach however, this effect has been neglected. More specifically, it is assumed that $\theta \ll 1$ and $\cos(\theta)\frac{T_m}{L_{CL}} = F_{carr}$ becomes $\frac{T_m}{L_{CL}} \simeq F_{carr}$. Here the angle θ is the angle of the lever, pushing the carriage (ie. $\theta_m = n\theta$). Furthermore, there are several types of friction in the system: the intrinsic friction within the

motor and its reduction gear, inside the bearings and the carriage in its guideway. Additionally, the springs have a non-negligible damping coefficient. In this work the overall friction and the spring damping have been merged and are represented by the friction coefficient b_{sp}. The stimulator, is not in contact with the controller's casing, but only with the operator. To model the damping of the skin of the operator and the friction between the skin and the palm pad, the damping coefficient b_{op} has been introduced. Similarly, the spring constant of the operator's skin is modeled by k_{op}.

3.2 Spring Constant and Damping Coefficient of the Operator's Hands

The order of magnitude of the two coefficients k_{op} and b_{op} are discussed in [32] [33] [34]. They all indicate parameters varying in the same order of magnitude, namely $k_{op} \simeq 400\text{N/m}$ and $b_{op} \simeq 5\text{Ns/m}$.

3.3 Identification of the Spring Damping Coefficient

The damping coefficient of the spring b_{sp} can be found by comparing the theoretical results of the frequency response analysis with the experimental findings. In fact, for the experimental setup, the stimulator has been fully blocked and therefore the operator's spring coefficient can be seen as infinitely stiff.
By varying b_{sp} and Bode-plotting the results of the analytical transfer function, the coefficient's order of magnitude can be found. In order to do so, the analytical transfer function has to be identified.

3.4 Expected Transfer Functions

The system can be sub-structured into two major transfer functions. The block diagram including these two transfer functions is depicted in figure 21.

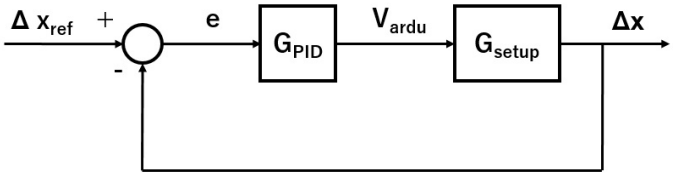


Figure 21: Block diagram with two different transfer functions.

According to this figure one can obtain a transfer function of the following form:

$$F(s) = G_{PID}(s)G_{setup}(s) = \frac{V_{ardu}}{E}\frac{\Delta X}{V_{ardu}} = \frac{\Delta X(s)}{E(s)} \tag{5}$$

where V_{ardu} is the voltage output of the Arduino for controlling the motors. Using this form one can calculate the individual transfer functions and finally relate the compression of the springs Δx to the compression given as reference Δx_{ref}, since $\Delta X/\Delta X_{ref} = F/(1+F)$.

3.4.1 PID Transfer Function

The transfer function given by the PID-controller is very straight-forward and can be taken out of any control theory book [35]. Since the Arduino has limited resolution with floating point operations, the error and reference have been converted into micrometers. To keep the gains consistent between the Arduino code and the matlab script for analytical analysis, the multiplication factor $K_{\mu m}$ to get

from the reference distance in meters to micrometers has been introduced. Furthermore, an offset has been added ($K_{bit-offset} = 128$) to allow for negative armature voltages (motor voltages) and lastly the PID value has been converted to the Arduino output voltage, where 255 corresponds to 5 V. The transfer function is given in equation 6.

$$G_{PID}(s) = \frac{V_{ardu}(s)}{E(s)} = (K_{\mu m}(K_P + \frac{K_I}{s} + K_D s) + K_{bit-offset})K_{PID2Vardu} \tag{6}$$

Finally, there is also the gain of the amplifier in voltage mode, which converts the voltage of the Arduino into the voltage applied to the motors. This gain is $K_{ampl} = 10\text{Volt/Volt}$. To this voltage an offset voltage of $V_{offset} = -25\text{V}$ is added.

3.4.2 Motor Equations

The second transfer function relates the motor torque T_m to the Arduino voltage as well as the output Δx to T_m. Due to the back electromotive force these two parts are related and have to be treated as a whole.
The output torque T_m of the motor can be calculated using the sum of all torques and the conversion parameters intrinsic to the motor.
Similar to the setup and analysis in [27] the equations of the motor are given as:

$$L_a \frac{di_a}{dt} + R_a i_a + K_{emf}\dot{\theta}_m = V_a \tag{7}$$

where L_a is the armature inductance, R_a the armature resistance and i_a the armature current of the motor. K_{emf} is the back electromotive force constant also given by the motor. V_a is the armature voltage and θ_m is the angle of the motor shaft.
Furthermore, with Newton's second law for rotation, the sum of all torques must be zero, or:

$$J_T\ddot{\theta}_m - \frac{k_{eq}L_{CL}}{n}\Delta x - \frac{b_{sp}L_{CL}}{n}(\dot{x}_2 - \dot{x}_1) = T_m = K_\tau i_a \tag{8}$$

In equation 8 the parameter J_T stands for the total equivalent inertia of the motor, the clamping link and the carriage of mass m_1. K_τ is the proportional current torque gain constant. The moment of inertia can either be calculated as the sum of all inertias seen by the motor shaft, or measured in a simple test.

3.5 Analytical Inertia Identification

The total inertia of the system is determined by the inertia of the rotor and gears J_m, the inertia of the clamp link J_{CL} as well as the inertia of the carriage assembly with mass m_1. The last one can be found by simplifying the load to a point mass at distance of the clamp link's length L_{CL}, which yields a moment of inertia of $J_{carr} = m_1 L_{CL}^2$. The gear box affects the inertia seen by the motor shaft by the square of its ratio n:

$$J_{reflected} = \frac{J_{load}}{n^2} \tag{9}$$

We therefore have a total inertia of:

$$J_T = J_m + \frac{J_{CL}}{n^2} + \frac{m_1 L_{CL}^2}{n^2} \tag{10}$$

where J_{CL} can be calculated by approximating it as a cantilever with an off-center axis of distance $L_{CL}/2$ [5]:

[5] (2018, June 19th) retrieved from `http://www.orientalmotor.com/technology/motor-sizing-calculations.html`

$$J_{CL} = \frac{1}{12} m_{CL}(A^2 + B^2 + 12l^2) \tag{11}$$

where A and B are the width and length respectively.
The calculated total moment of inertia for the motor with a reduction ration of $n = 112$ is:

$$J_T = 6.87 \times 10^{-8} kgm^2$$

3.6 Experimental Inertia Identification

Alternatively, one can approximate the total moment of inertia by applying a constant current on the motor and measuring the acceleration. In this case the traveled distance has been derivated twice to find the acceleration, which results in an amplification of errors. Furthermore, the constant current has been kept very low, (between 20 and 100mA), which led to a slow movement and therefore higher friction impact on the measurements. However, the results are consistent with the theoretically calculated values:

$$J_T = \frac{7.5 \times 10^{-4}}{n^2} = 5.98 \times 10^{-8} kgm^2$$

3.7 Relating ΔX to θ

The conversion between the angle θ and the carriage's traveled distance x_1 can be found by assuming that the horizontal displacement of the carriage is given by $L_{CL} sin(\theta) = x_1$. For small angles of θ the Taylor expansion gives:

$$L_{CL}\theta \simeq x_1 \tag{12}$$

The output Δx is the compression of the springs and is given by $\Delta x = x_2 - x_1$. For finding x_2 the equation of motion given by Newton's second law has been considered.

$$m_2\ddot{x}_2 = -k_{eq}(x_2 - x_1) - b_{sp}(\dot{x}_2 - \dot{x}_1) - k_{op}x_2 - b_{op}\dot{x}_2 \tag{13}$$

In the case where the stimulator has been blocked by a wall, $\dot{x}_2$ has been forced to zero. Using the Laplace transform and equation 13 one finds the expression of x_2:

$$X_2 = -\frac{k_{eq} + b_{sp}s}{s^2 m_2 + b_{op}s + k_{op}} \Delta X \tag{14}$$

3.7.1 Motor and Spring Transfer Function

Combining all the equations one can find the final block diagram, which can be seen in figure 22
From this diagram and the equations mentioned above, one can obtain the transfer functions that relate the output Δx and input Δx_{ref} as introduced in equation 5, where $\Delta X(s)$ and $\Delta X_{ref}(s)$ are the Laplace transforms of the output and input functions respectively.
It is thus possible to study the frequency response by simulating this setup with the assumptions mentioned earlier.

3.8 Main Equations for Analytical Transfer Function

To sum up, the equations leading to the analytical transfer function are represented here.

$$\Delta X_{ref} - \Delta X = E \tag{15}$$

$$((K_P + K_D s + \frac{K_I}{s})EK_{\mu m} + K_{bit-offset})K_{PID2Vardu}K_{ampl} + V_{offset} = V_a \tag{16}$$

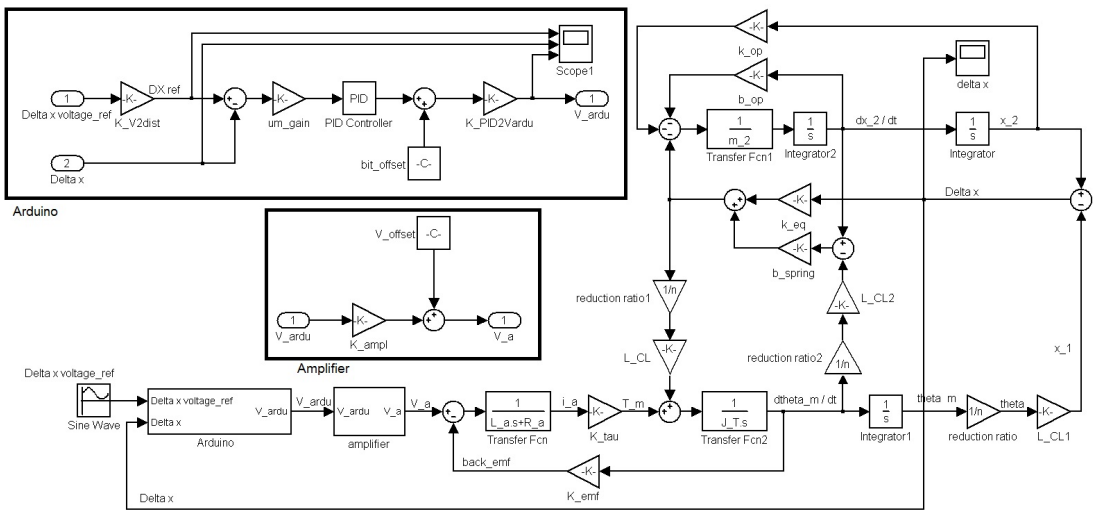


Figure 22: Complete block diagram relating the output Δx to the input Δx_{ref}.

$$\frac{V_a - K_{emf}\dot{\theta}_m}{L_a s + R_a} K_\tau + F_{coupled} = J_T \ddot{\theta}_m \tag{17}$$

$$F_{coupled} = (k_{eq} + b_{sp}s)\frac{L_{CL}}{n}\Delta X \tag{18}$$

$$\theta_m = \frac{n}{L_{CL}} X_1 = -\frac{n}{L_{CL}}(1 + \frac{k_{eq} + b_{sp}s}{m_2 s^2 + b_{op}s + k_{op}})\Delta X \tag{19}$$

Note that the constant offsets in equation 16 are for pure symmetrical reasons and will cancel each other out, since $K_{bit-offset}K_{PID2Vardu}K_{ampl}+V_{offset} = 0$ and the equation becomes $((K_P+K_D s+\frac{K_I}{s})EK_{\mu m})K_{PID2Vardu}K_{ampl} = V_a$. This is the equivalent to a standard PID form with gains K'_P, K'_D and K'_I.
In the case of the experimental setup the palm pad has been blocked and therefore x_2 has been forced to be constant. The last equation becomes thus: $\theta_m = -\frac{n}{L_{CL}}\Delta X$.

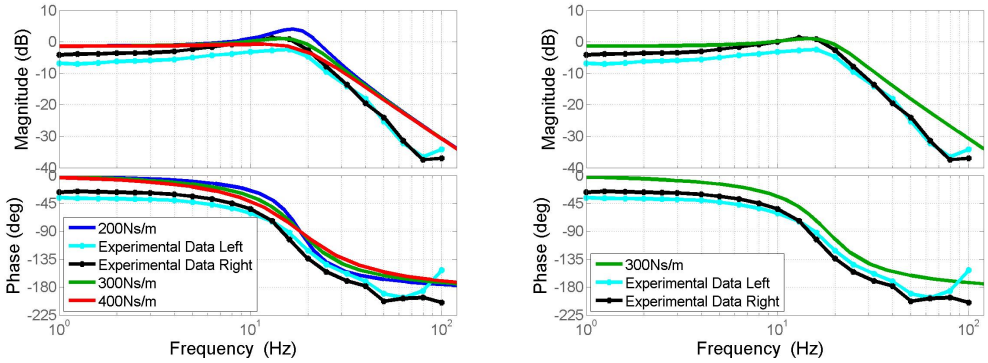
3.9 Analytical Results

With the identified mathematical model, one can simulate the setup in *matlab*. With the assumption, that all known or directly measurable parameters have been correctly identified and that the mathematical model is correct, the unknown parameters can be studied. Simulations have been done to identify the unknown spring damping coefficient. A parameter search for this coefficient (b_{sp}) has been undergone to obtain the corresponding Bode plots. This damping coefficient also accounts for the friction in the system, since these two effects cannot be discriminated in the experiments.
Comparing the Bode plots with the experimentally gathered data, one can identify the best match and thus obtain a physical value for the two (henceforth referred to as spring damping coefficient). For this however, it was necessary to have fully built the controller and created an experimental frequency response analysis of it. Even though the steps taken for this are explained in the later section, the results can be seen below.

3.9.1 Comparison of Results - P-Controller and Experimental Data

The analytical transfer function has only been calculated for a wide set of spring damping coefficients b_{sp}. However, only few potentially best matches are represented in these figures. The results for the uniformly spaced values between 200Ns/m and 400Ns/m are depicted in figure 23a. The gains that have been used are for a P-controller, it was namely $K'_P = 39.2\text{V/mm}$. The simulation result that was closest to the experimental setup can be seen in figure 23b and the corresponding transfer function is:

$$\frac{\Delta X}{\Delta X_{ref}} = \frac{1.23 \times 10^3}{3.24 \times 10^{-6}s^3 + 6.67 \times 10^{-2}s^2 + 8.48s + 1.23 \times 10^3} \tag{20}$$



(a) Several different b_{sp}.

(b) $b_{sp} = 300\text{Ns/m}$.

Figure 23: Bode plot comparison of experimental data of the P-controller and analytical transfer functions for different spring damping coefficients (b_{sp}).

Also in figure 23b the gain for lower frequencies of the analytical transfer function is marginally less than 0 and a slight resonance top becomes visible around 18Hz. The phase shift of the experimental data has a non-negligible offset with respect to the analytical curves. At low frequencies, the offset or phase lag is $-45°$ and it reaches a value around $-205°$ for the highest tested frequencies, whereas the analytical model asymptotically approaches $-180°$. Due to the setup constraints, the analytical results of higher frequencies have not been considered.
The identified spring damping coefficient that seems to best match the experimental data of the P-controlled setup is $b_{sp} = 300\text{Ns/m}$.
Due to this non-perfect tracking, especially for low frequencies, a better performing controller had to be studied.

3.9.2 Comparison of Results - PID-Controller and Experimental Data

With the gains stated in table 5, section 5.10, the frequency response can be calculated again. This has been done using the motor with a reduction ratio of $n = 112$. The comparison between the analytical results and the experimentally gathered data can be seen in figure 24.
This time the analytical model does not match the experimental data equally well. Even though the same resonance top is present for a spring damping coefficient of $b_{sp} = 100\text{Ns/m}$, the high frequency prediction is quite different from the actual data.
The analytical system from figure 24 still shows a second order behavior, since the phase lag will asymptotically go to $-180°$, but the graph has been cut at the frequency range of the test setup.

Comparing the analytical and experimental data, one sees that for low frequencies the experimental data still has a rather high phase lag, ie. $-45°$. This constant delay seems quite big (ie. a delay of

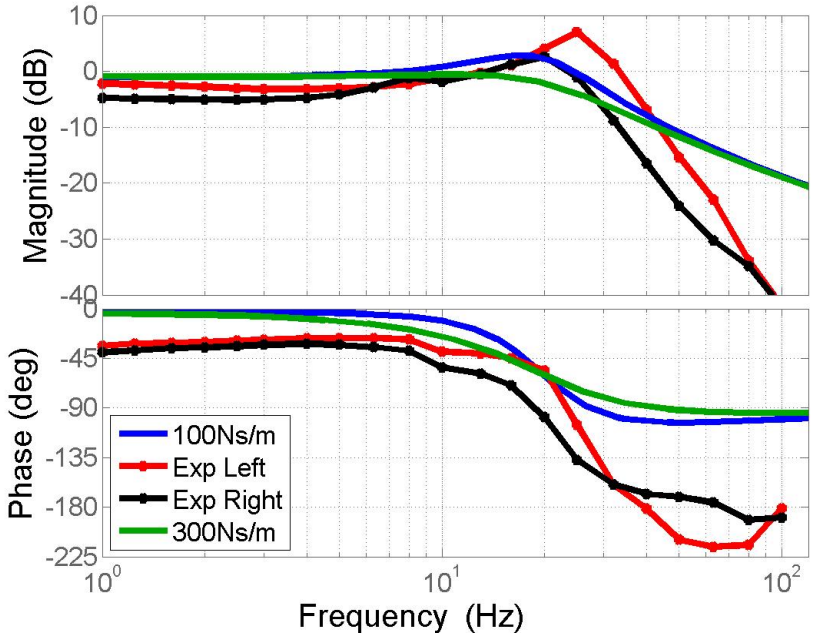


Figure 24: Bode plot comparison of experimental data of the PID-controller and analytical transfer function for $b_{sp} = 100$Ns/m and $b_{sp} = 300$Ns/m.

0.125s for 1Hz) and the cause of it cannot be identified with full certainty.
An initial guess would be, that there is a static friction in the system, that needs to be overcome and therefore introduces a delay. Also, the mechanical aspects like backlash or movable parts that should be fixed might account for this phase lag. In addition to this, mechanical misalignments can introduce stress in the system, which causes this kind of behavior. The most plausible explanation however, is the presence of a hysteresis effect (a more in-depth analysis of this can be seen in section 8.5).

The additional offset for the middle and high frequency range (compared with expected theoretical behavior) is governed by different factors. Here the delay stems from the fact that several filters have been put in place. The first one is an electric RC-circuit that filters the PWM signals sent to the amplifiers. However, the cut-off frequency is at 330Hz and the delay should have no impact for the tested frequencies. The second filter is within the Arduino code and is used to filter the measured distance. This is a very basic digital filter explained in more detail in equation 26 in section 5.12.
This filter was necessary to reduce the impact of noise amplification in the derivative part of the PID-controller. Three different filters with cut-off frequencies from $f_1 = 1.6$Hz, $f_2 = 4.2$Hz and $f_3 = 8.8$Hz have been tested. These filters did not yield significantly different Bode plots compared with each other. The performance of these filters can be seen in figure 43 in section 5.12. But with such low cut-off frequencies, a non-negligible delay for low frequencies is induced which might account for the high phase lag.
Another source of this error is the fact that the discretization becomes more important in the PID-controlled setup. However, even when one tries to analytically discretize the system, the results do not match the empirical data. To improve the theoretical model, it is suggested to implement the effects of this digital filter in the analytical transfer function.
For the sake of the argument, the derivative part has been lowered by a factor of 10% and the

results can be seen in figure 25.

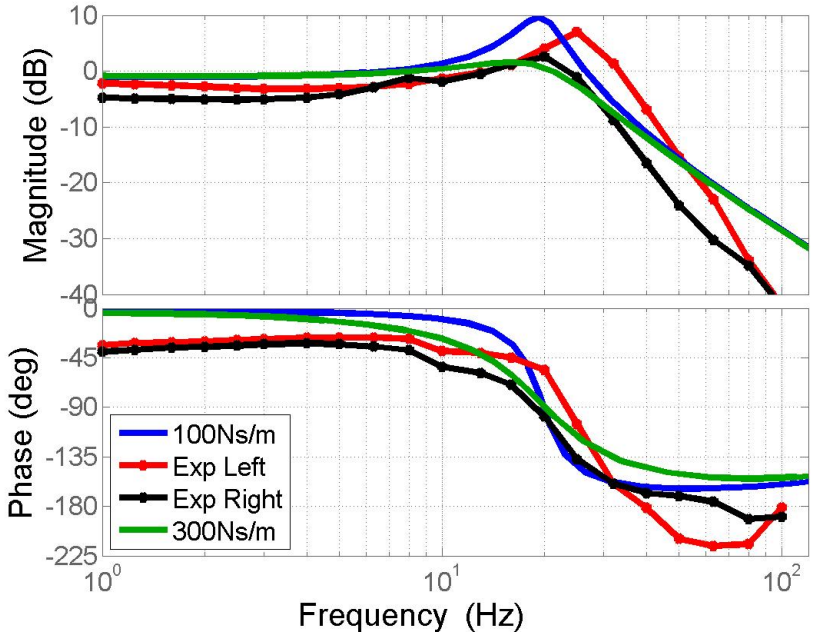


Figure 25: Bode plot comparison of experimental data of the PID-controller and analytical transfer function for $b_{sp} = 100$Ns/m and $b_{sp} = 300$Ns/m. The derivative gain is 24.7Vs/mm.

Judging by figure 23a, one can also suggest a control structure without a derivative gain. In fact, the small resonance peak but steady-state offset (gain smaller than 0dB) for low frequencies suggest, that an integral action is reasonable. Therefore, the system has also been tested with a PI control scheme, where the proportional gain was $K_P = 31.7$V/mm and the integral gain was $K_I = 0.183$V/mm/s. The results are shown in figure 26.

Note that this test has only been done in the end of the project. For this reason, the analysis and experimentation in the following chapters is mainly based on the P- and PID-controller. For future work, it is suggested to adopt the PI control scheme.

3.10 Analytical PID Gain Domain Study

In order to understand the influence and order of mangitude of applicable gains, a theoretical analysis has been conducted, using the mathematical model. It is important to keep in mind, that the digital filter has not been modeled, which is why a strong discrepancy between experimental data and theoretical analysis will arise when focusing on the derivative gain.
When varying the proportional gain, one shifts the Bode plot in a horizontal direction. A higher gain will yield a better frequency response and therefore shifts the graphs towards the right-hand side. However, a too high gain causes saturation of the system (ie. motor voltage) or instability, which is not included in this simulation. Moreover, a higher proportional gain lifts the Bode plot on the y-axis, resulting in sharper resonance peaks. The tested proportional gain range is between 39.2V/mm and 400V/mm. The upper end will yield an oscillation frequency of 55Hz and a gain of 20dB at said frequency.

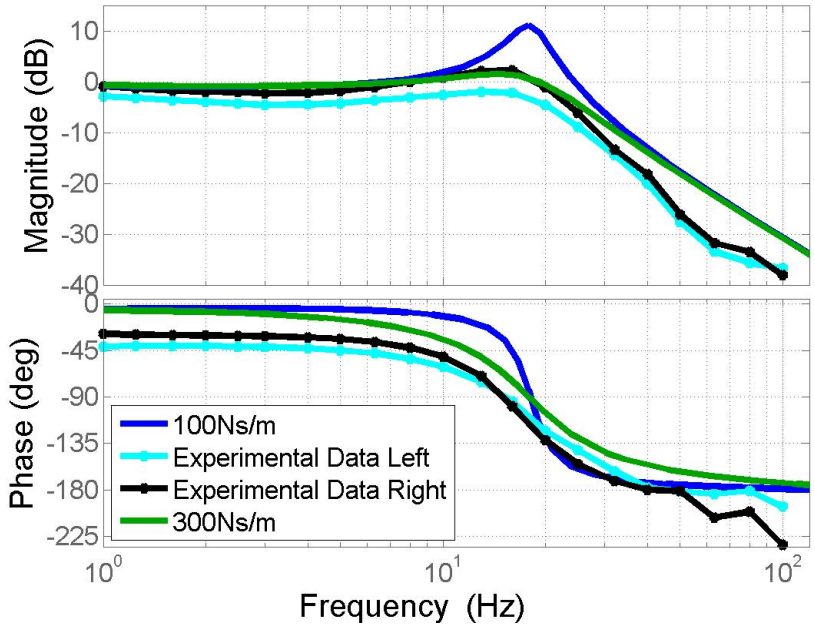


Figure 26: Bode plot comparison of experimental data of the PI-controller and analytical transfer function for $b_{sp} = 100$Ns/m and $b_{sp} = 300$Ns/m.

A higher integral gain sharpens the resonance peak (higher value) and yields a phase plot that resembles a step-function (rectangular-shaped). The gains varied between 0.06V/mm/s and 1.2V/mm/s.

The derivative part has a very different impact for experimental and analytical data. In practice, a gain of 247Vs/mm has been chosen. The simulation using this K_D is also represented in figure 24. The system is of second order and reaches $-180°$ at 10^5Hz. In practice however, the Bode plot looks rather different.
In general, a lower derivative gain provokes a higher resonance peak value, with almost no vertical shift of the Bode plot. The phase curve sharpens and approaches a more step-like function with lower gains. Furthermore, for too high gains (ie. 25Vs/mm), the phase starts to not directly approach $-180°$ and swings around this value at high frequencies (between 100Hz and 10^5Hz). The tested values are between 2.5Vs/mm and 250Vs/mm. Even though a gain of 2.5Vs/mm seems reasonable according to the simulation, the finally implemented gain was of 247Vs/mm. This decision has been based on the experimental results and evaluation of the controller.

When varying the damping coefficient, one can see that a lower damping coefficient results in a bigger peak value for the resonance top. On the phase plot, the curve for a lower damping coefficient becomes sharper.

The final parameter that has been tested in the analytical simulation is the spring coefficient itself. Figure 27 shows the different results when varying the spring constant for the identified P-controller. A stiffer spring leads to a weaker magnitude value, but shifts the magnitude plot towards higher frequencies. The phase curve is only affected to a small extent, where the major change is in a horizontal shift as well. The oscillation frequency for 1N/mm is around 17Hz, whereas the oscillation frequency for 100N/mm lies at 35Hz. The damping coefficient that has been used in this case was

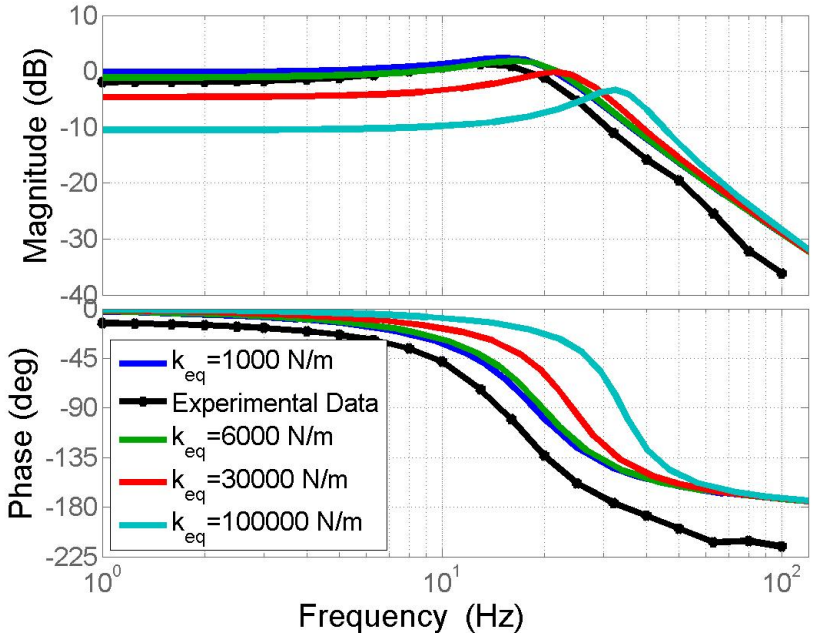


Figure 27: Bode plot comparison of experimental data of the P-controller and analytical transfer function for different spring coefficients (k_{eq}).

of 300Ns/m.

4 Robot and Environment

To test the controller, a robot developed by Topy was at disposal (see figure 28).

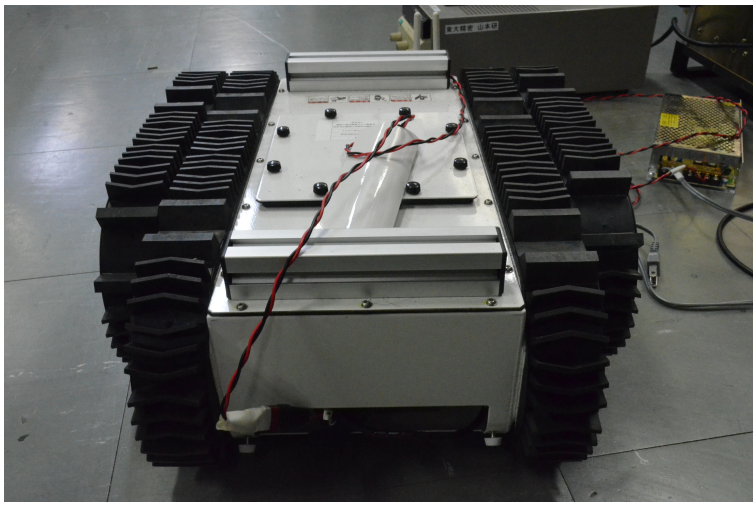


Figure 28: Topy robot for real-world controller test setup.

Figure 29: Topy controller.

This is a commercially available robot with its own controller. In order to replace this controller with the haptic feedback controller from this project, it was necessary to write a properly working environment. The topy robot is communicating via a wireless serial link. It has a prespecified communication message that consists in 70 bytes for sending from the robot to the PC and 44 bytes for the message to receive. These messages include the bytes reserved for proper starting and ending as well as the checksum. The full documentation of the messages and robot usage can be seen in the Topy manual (in Japanese only).
It was thus necessary to write an application that reads out the position of the two joysticks and construct a message including these joystick values as speed reference for the robot. Then the message has to be sent over the wireless serial link to the robot and the answer has to be received. The important sensor readings (inclination, current in the crawlers, passed time and battery level) have to be read out and a voltage command for the desired feedback according to the chosen feedback law has to be sent to the motors.
For this purpose, the programming language processing [36] has been used to create a graphical user interface and to establish the serial connection. For low-level motor control purposes, an Arduino Uno has been used. The parameters that can be set for testing purposes can be seen in table 4.

4.1 About the Robot

The robot is an all-terrain, multi-purpose robot, capable of driving over obstacles and slopes with its main crawlers. Attached to the crawlers it has two flippers that also allow to overcome bigger obstacles or even climb stairs.
The robot measures 45cm in length, 37cm in width and 17cm in height. It weighs around 13kg and has a 10Ah Li-Fe battery rated at 13.2V on board. However, this battery was not in its best condition and had to be replaced with an alternative power source. Furthermore, the robot is equipped with a WiFi transceiver that can establish a serial connection to the computer, such that it can be controlled by the controller plugged into the computer.
Due to the all-terrain accessibility of the robot, it seemed a perfect target to develop an intuitive haptic feedback controller for it, to gain more insight over the obstacles the robot is driving over. The robot can be seen in figure 28.

Setting	Value	Units
Baud rate for robot serial link	250000	[bps]
Baud rate for Arduino serial link	250000	[bps]
Update rate of the processing GUI	5	[Hz]
Update rate of the Arduino motor controller	1000	[Hz]
Max voltage for motor	20	[V]
PWM frequency	31372.55	[Hz]
Proportional motor gain	47.6	[V/mm]
Integral motor gain	0.124	[V/mm/s]
Derivative motor gain	247	[Vs/mm]
Left photoreceptor MIN value	640	[-]
Left photoreceptor MAX value	840	[-]
Right photoreceptor MIN value	700	[-]
Right photoreceptor MAX value	880	[-]

Table 4: Software parameters.

4.2 Hardware Setup

The hardware schematic is shown in figure 30. The main connections are the serial link cable between the Arduino and the computer, the wireless serial link between the computer and the robot, as well as the power lines from the amplifiers to the built-in motors in the haptic controller.

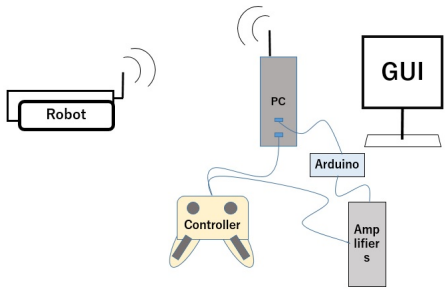


Figure 30: Hardware connection schematic.

4.3 GUI in Processing

The graphical user interface shows the current feedback method as well as the magnitude. It also indicates which driving direction (forward, backward or halt) is sent to the robot. Since there is no control of the battery charge on-board, the robot includes battery information in its message to the PC. The processing program reads out the charge of the battery and warns the user if it is low. It stops the program, if the battery state is critical. The feedback method can be changed by a mouse-click anywhere in the window. The GUI can be seen in figure 31.

4.3.1 Message Handling

This application handles the messages sent to and from the robot. The communication between the GUI and the Arduino takes place over the serial cable with a fixed baud rate (see table 4). The same baud rate has been used for the communication between the GUI and the real robot. The latter communication frequency is of 5Hz, which has been suggested by the Topy user manual.

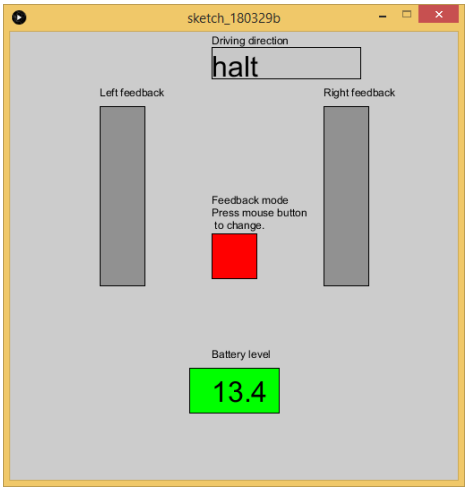


Figure 31: Graphical user interface written in processing.

The message description from the manual and its translation can be seen in the appendices. The message size for sending is 70 bytes long, while the message from the robot is of 44 bytes.

4.4 Control Scheme

At a first state, the control scheme is based on a simple proportional controller. The full scheme can be seen in figure 32 which implements the part of the control scheme that was already shown in figure 4. In the test setup (see section 5, figure 33) the reference signal $dist_{ref}$ is given by the sinusoidal function generator. This reference signal is directly treated as desired feedback. In the operational mode, this corresponds to the target output force, also called the haptic feedback force that should be felt by the user. Ideally, this shall be a function of orientation (roll and pitch) as well as the current in the two crawlers.

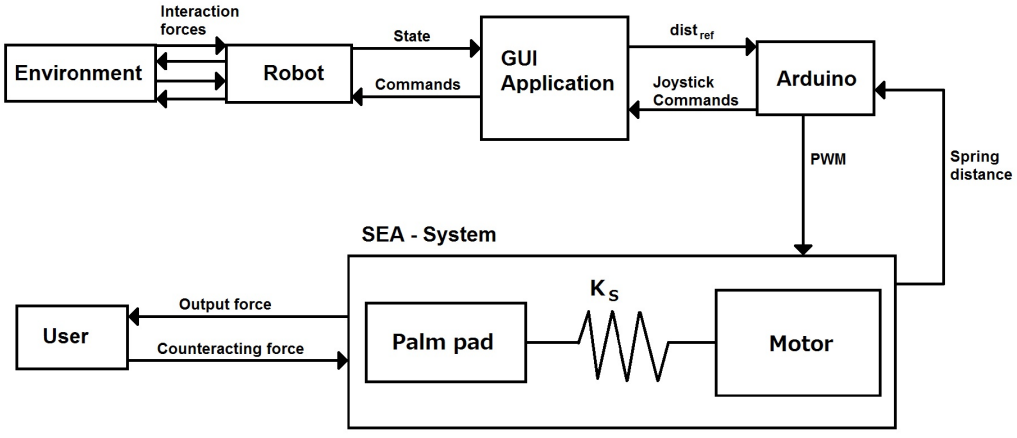


Figure 32: Complete system scheme of the robot in its environment and the user.

4.5 Electrical low-pass Filter

Before the PWM signal from the Arduino is sent to the amplifiers, where it is amplified to control the motors, it has been filtered with a simple RC low-pass filter. The resistor has a value of 3.9 kΩ and the capacitor of 0.1 μF. This smooths out the PWM signal and has been implemented in order to prevent the amplifier from trying to follow the Arduino signal with unwanted precision, eventually causing too much heat. The cut-off frequency is therefore 204Hz.

4.6 Feedback Laws

The previous research in this project suggested to use a feedback law, that couples both the roll and pitch orientation of the robot. The underlying law of this idea is given in equation 21, where θ_p and θ_r are the pitch and roll angles of the tank in the Unity simulation respectively.

$$F = K_1(\sin\theta_p \pm \sin\theta_r) \tag{21}$$

In the processing application (GUI) the feedback of the robot is extracted from the received message. By simply clicking somewhere in the window, the user can switch between three feedback laws. The first one is the same as in the previous paper, with the difference that the operator $\pm$ has been replaced by a simple + which results in uniform feedback for both sides. The second and third feedback laws are indicated in equation 22 and 23. In the latter, i_{cl} and i_{cr} are the currents in the left and right crawler respectively. For testing, mainly the last law has been used, since there were not so many obstacles to overcome in the testing environment and therefore the orientation did not change much.

$$F = K_2(\sin\theta_p \sin\theta_r) \tag{22}$$

$$F = K_3(i_{cl} + i_{cr}) \tag{23}$$

5 Testing of the Game Controller

5.1 Frequency Response Function

For an in-depth performance analysis and evaluation of the controller a series of experiments have been conducted. A standard approach is to measure the frequency response function of the controller. In order to find this frequency response, one should consider the amplitude correlation between the input signal and the output signal, as given in the following equation:

$$H = \frac{F^*(j\omega)}{F(j\omega)} \tag{24}$$

Where $F^*(j\omega)$ is the transfer function of the output signal and $F(j\omega)$ of the input signal.
The results of this testing can be plotted in a Bode-plot which is a standard representation. This not only shows for which frequencies the reference signal can be tracked nicely, but it also indicates any potential oscillation frequencies. In addition, one can read out the order of the system when looking at the phase plot of the frequency response function. In the phase plot the phase lag between the reference signal and the output signal is represented. Furthermore, the bode plots allow for a direct cross-platform comparison with other controllers and devices.

5.2 Test Setup

In this experiment a thorough frequency response analysis shall be done on the controller. On the left-hand side, the motor with a reduction ratio of *33:1* has been used, whereas for the right-hand side, the motor had a reduction ratio of *112:1*.
A reference signal is fed into the Arduino, which then controls the motors to match the compression of the springs with the reference. The operational distance of the photoreceptors to the palm pads is 2 to 4mm which lies within the more sensitive region of the sensor.

5.3 Control Scheme

The reference signal is given by the Function Generator **SG-4115**. This generator has an intrinsic output impedance of 50 Ohm. This means that it expects to have a device connected to it with the same value as input impedance. If this is the case, these two elements form a simple voltage divider and only half of the voltage is applied to the target device. However, this is not the case for the Arduino, since it has a considerably higher input impedance. Therefore, the settings made on the function generator result in double the voltage on the Arduino. From this point on, this issue shall be neglected and all future voltage indications refer to the voltage level as seen by the Arduino.
The function generator produces a sine wave between 0 and 5V with a frequency ranging from 1 to 100Hz. The Arduino reads this voltage and controls the motor to have a proportional spring compression accordingly. In this case, 0V as reference signal is 0% compression and 5V corresponds to 100% compression. The control scheme of this setup can be seen in figure 33. In this case the counteracting force from the user is infinite (ie. pseudo-infinite stiffness), since the palm pads have been blocked by a wall. Tests have also been made where the palm pads have been blocked by the operators hands, but without any significant difference. For simplicity, the wall blocked test setup has been used for these investigations.

5.4 Tracking Behavior of the P-Controller

In order to measure the tracking performance of the controller, the Arduino sent its data to the serial link. This data has then been read and treated in a python script and represented with the *matplotlib* library.
The tracking behavior of the P-controlled ($K_P = 39.2$V/mm) setup can be seen in the following figures.

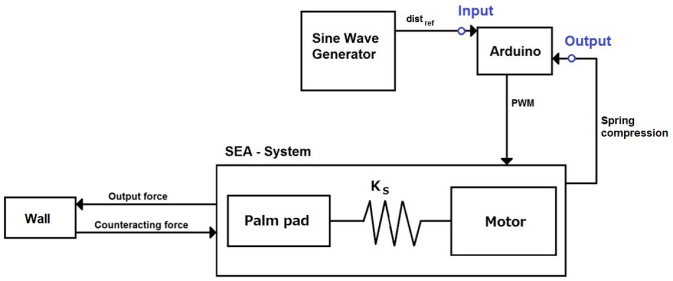
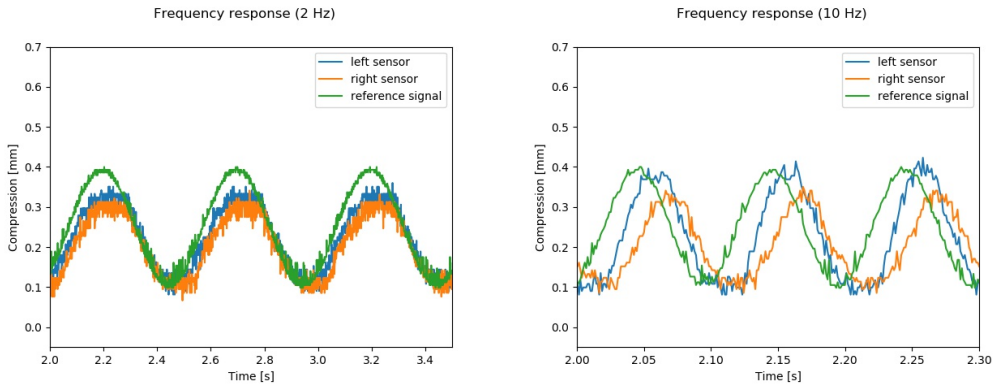


Figure 33: Control scheme for the frequency response analysis.



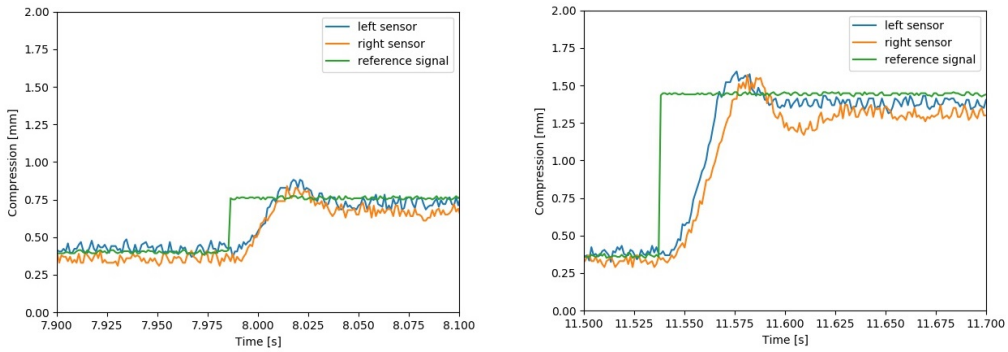
(a) $K_P = 39.2$V/mm for 2Hz.

(b) $K_P = 39.2$V/mm for 10Hz.

Figure 34: Tracking behavior of the P-controller for different frequencies.

5.5 Step Response

In order to gain more insight in the controller's behavior and performance, a step response analysis has been done. When using the P control scheme, the performance indicated in figure 35 can be obtained.



(a) Low reference values.

(b) High reference values.

Figure 35: Tracking behavior of the P-controller for different step response input.

When comparing this result with the analytical data, one obtains the graph from figure 36.

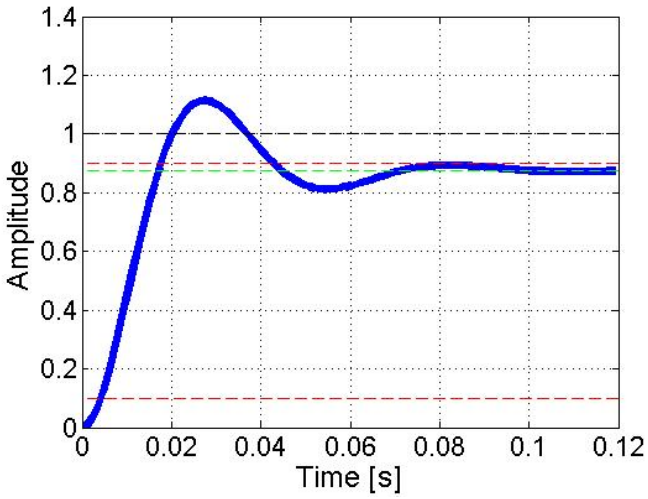
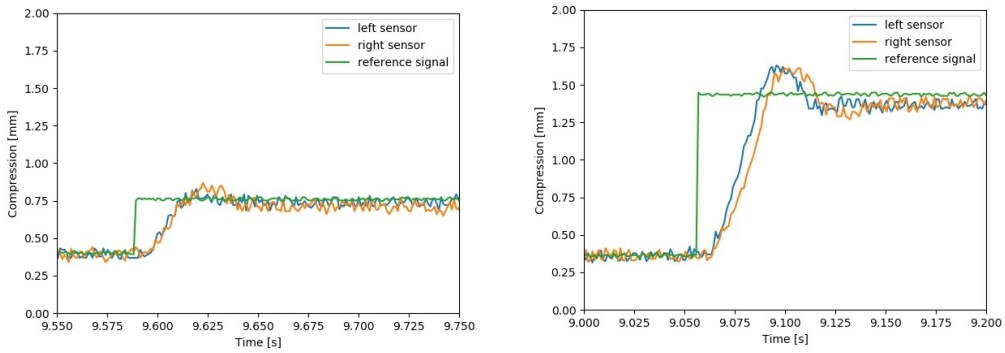


Figure 36: Analytic unit step response of the P-controller.

In figures 35 and 36 the response time has been calculated using the threshold values of 10% and 90% (also indicated in analytical results). This yields the response time of 10ms and 15ms for the experimental data, and 15ms for the analytical.

The same analysis has been done for a PI controlled setup, where the proportional gain was $K_P = 32\text{V/mm}$ and the integral gain was $K_I = 0.183\text{V/mm/s}$. The results can be seen in figures 37 and 38.



(a) Low reference values.

(b) High reference values.

Figure 37: Tracking behavior of the PI-controller for different step response input.

The delays are 12ms and 18ms for the experimental setup and 23ms from the analytical results. This time, the measured compression goes asymptotically towards the reference compression. The stead-state offset has successfully been removed.

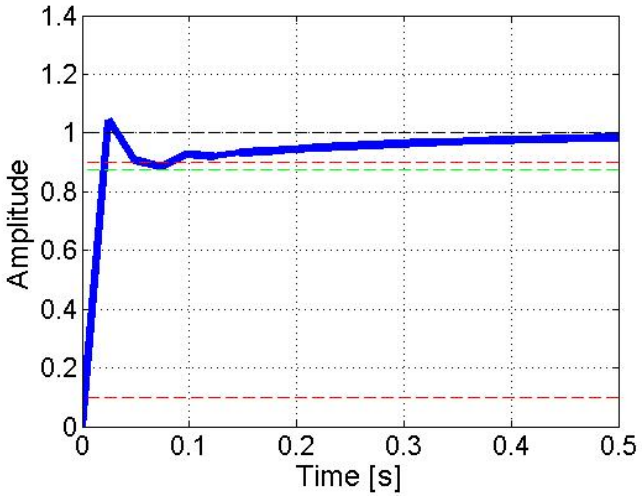


Figure 38: Analytic unit step response of the PI-controller.

5.6 Sine wave Fitting

As described in this example[37], one can fit a sine wave with the linear least square method to the samples. For finding the amplitude, the second norm has been used. This function returns the phase, amplitude and the bias. In this case, the difference in phase between input and output, as well as the amplitude ratio is needed in order to plot a Bode diagram.

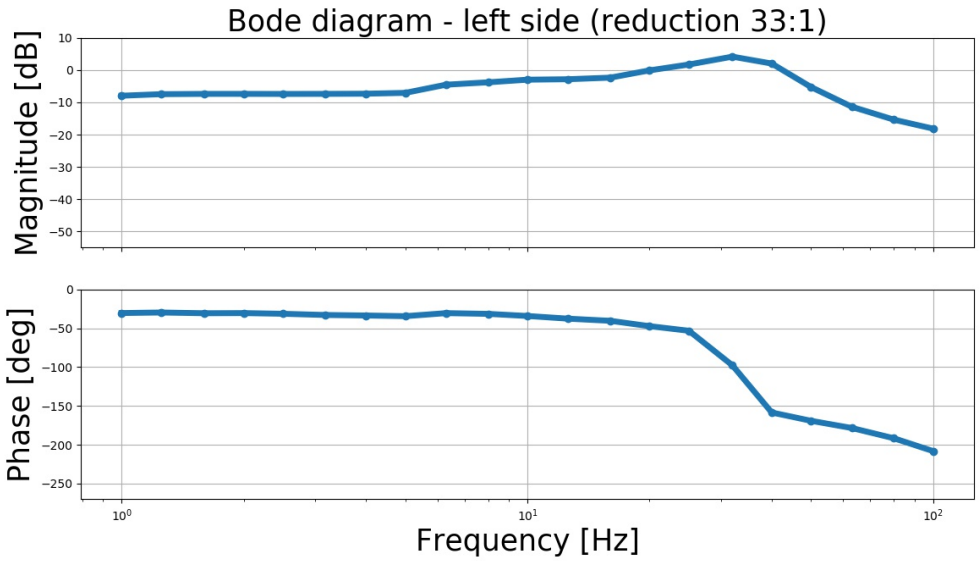
5.7 Bode Diagram

The results of the frequency response analysis are shown in figures 39a and 39b.
For the left-hand side, a resonance peak at around 30Hz can be found, with a gain of roughly -8dB for lower frequencies. At higher frequencies a slope of roughly -30dB/dec has been calculated. For doing so the values in the range of 50Hz and 100Hz have been used. The phase shifts from $-35°$ to $-210°$, a total shift of roughly 175°.

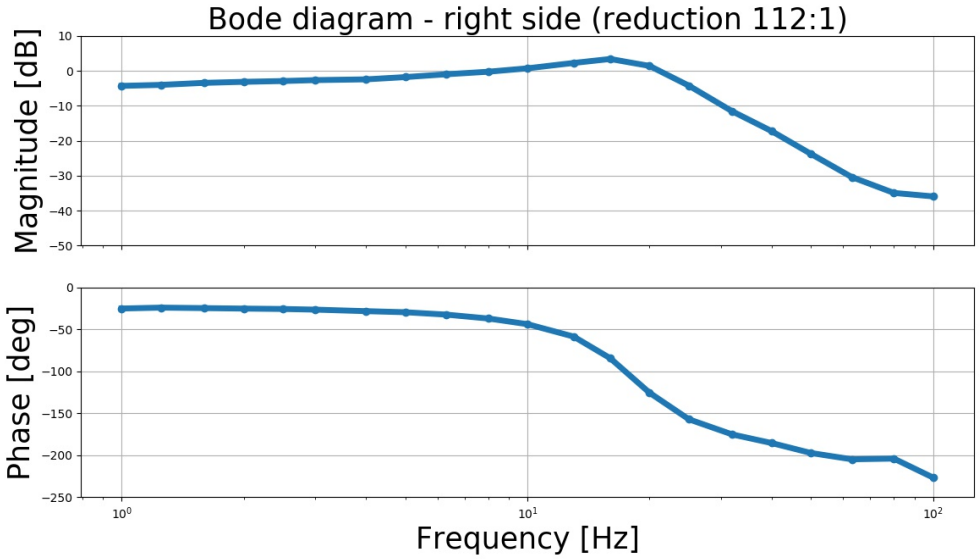
For the right-hand side, the resonance peak is less extreme and the cut-off frequency is around 13Hz, and a constant gain of roughly -4dB for lower frequencies can be identified. At higher frequencies the same slope of -30dB/dec has been calculated, again, using the values in the range of 50Hz and 100Hz. The phase shifts from $-25°$ to $-225°$, a total shift of roughly 200°.
From the phase shift one can calculate the delay for the desired frequencies. For a frequency of 1Hz, the delay is 70ms. Even though this seems like a big delay, it meets the requirements stated in section 1.7.

5.8 Discussion of the P-Tracking

As shown in figures 34a and 34b, the signal is not followed perfectly. For one, the magnitude ratio even at low frequencies is at roughly 0.85 for the right side. This shows that the proportional gain is too low. This would not only lift up the curve indicated in the Bode diagram, but also shift it towards higher frequencies (to the right).
From the Bode diagrams one can conclude several things. First of all, the gain magnitude at lower frequencies should be around 0dB to have perfect following of the reference signal. This can be influenced by tuning the proportional value of the controller, or introducing an integral part.
Second, the two motors show different characteristics. While the weaker motor (left-hand side)



(a) Left-hand side, Motor reduction ratio *33:1.*



(b) Right-hand side, Motor reduction ratio *112:1.*

Figure 39: Bode diagrams for the two motors.

shows a resonance top around 30 Hertz, the stronger motor behaves differently.
Furthermore, the first pole of the system can be found around 13 Hertz. The communication frequency between robot and graphical user interface is suggested (according to the datasheet of the robot) to be between 2Hz and 5Hz. This shows that the series elastic actuator system as it is implemented in this experiment, is not the limiting factor in the robotic system. The controller therefore successfully meets the previously stated requirements.

When wanting to determine the order of the system, one can look at the slope at the tail of the magnitude curve in the Bode plot. In our case they are -30dB/dec for both sides. Since the slopes can only be a multiple of 20dB/dec, it can be concluded, that the $2\xi\omega_0$ term from equation 25 has a non-negligible influence on the tail of these Bode plots. The equation of a second order transfer function is as follows:

$$H(s) = \frac{\omega_0^2}{s^2 + 2\xi\omega_0 s + \omega_0^2} \tag{25}$$

Therefore one should also have a look at the phase lag diagram. The phase shift of roughly $-180°$ suggests a second order transfer function.

5.9 Motor Comparison

When comparing the two motors at hand, one can conclude that the motor with the higher reduction ratio has a higher output force with a speed trade-off. Since one of the critical elements of the SEA system is its actuation speed, it is essential to push the boundaries as far as possible. However, the high gain frequency response in the experimental setup starts to drop at much higher frequencies than the actual operating frequency (which is given by the rather slow communication speed between the robot and the control device of $f_{op} \simeq 2 - 5$Hz).
Due to the fact that even with the stronger reduction gear motor, the springs cannot be compressed to their limits, a higher possible output force has been favored and therefore the stronger motor seemed more appropriate. From this point on, only the motor with reduction ratio of *112:1* has been considered for this controller.
When one replaces the left motor to have two identical motors, the following Bode plot can be obtained (see figure 40). The two curves overlap nicely, except for the phase values for the highest frequency (100Hz). This is due to the noise in the photoreceptors that becomes dominant at such low gain magnitudes and has no further significance.

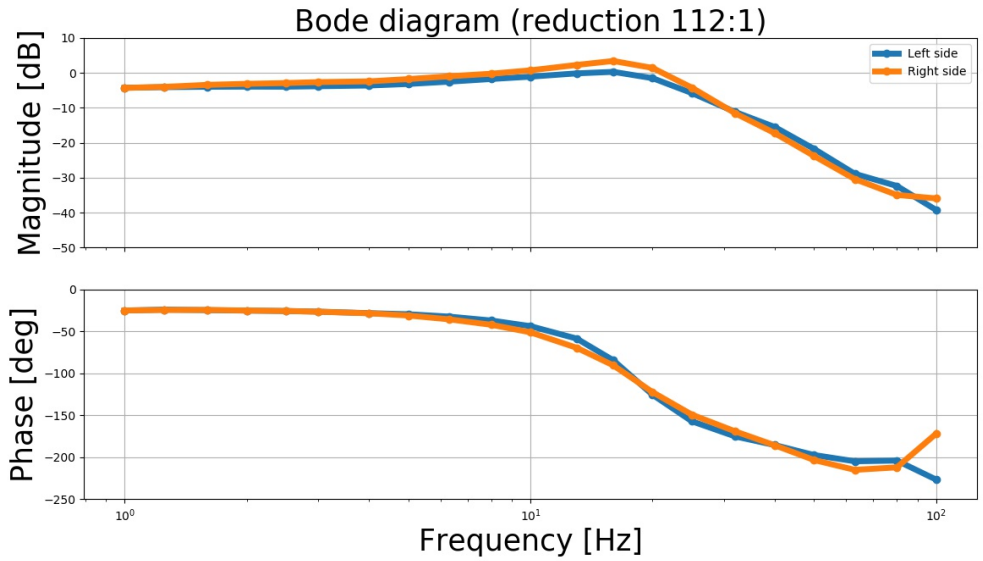


Figure 40: Bode plot for both motors with same reduction ratio, P-controlled.

5.10 Experimental PID Tuning

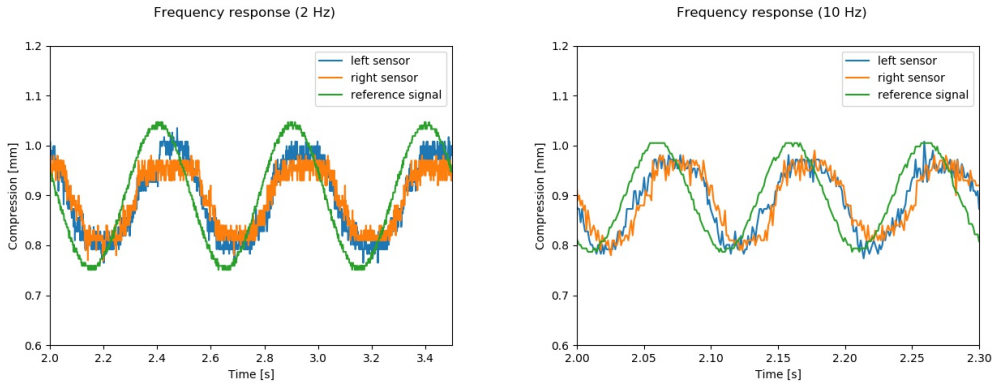
In order to compare the P-controller to potentially better controllers, it has been extended to a PID control scheme.
At first, the Ziegler Nichols tuning method has been used to identify the PID gains for the setup. However, due to the non-linear behaviour, these gains resulted in a rather poor tracking performance of the reference signal.
Since an educated tuning of the gains is the very core problem of all control engineering, a lot of different approaches exist to find optimal or sub-optimal gain values. Given the complexity of the setup, it seemed reasonable to opt for the simple trial and error approach, where the gains have been tuned and the tracking performance was shown in real-time. This approach led to the following gain coefficients:

Designator	Value	Unit
K'_P	47.6	[V/mm]
K'_I	0.124	[V/mm/s]
K'_D	247	[Vs/mm]

Table 5: Trial and error PID tuning.

5.11 Tracking Behavior of the PID-controller

The tracking behavior of the PID-controlled setup can be seen in figures 41a and 41b.



(a) $f = 2$Hz. (b) $f = 10$Hz.

Figure 41: Tracking behavior of the PID-controller (see table 5) for different frequencies.

The same experiment for measuring the frequency response function has been conducted and the Bode plot can be seen in figure 42.

5.12 Discussion of the PID-Tracking

The performance of this controller has not been improved drastically compared to the P-controller. There is still a small negative gain at lower frequencies. Opposed to the previous Bode plot in figure 40, the two sides show different characteristics in figure 42. This might be due to several asymmetries, such as the different sensor thresholds or noise values, or the fact that the gains have been kept the same for both sides.
However, this time the Bode plot has been shifted to the right and a resonance top is more clearly

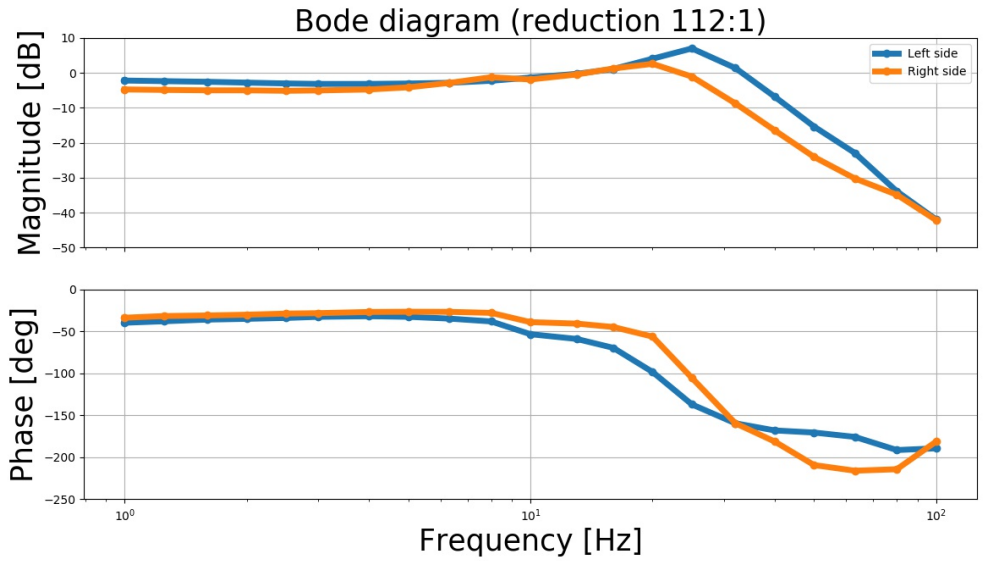


Figure 42: Bode plot for both motors with same reduction ratio, PID-controlled.

visible at 24-25Hz. Similarly, the phase lag is considerably big for low frequencies, starting at $-40°$, but ending around $-180°$ for higher frequencies. This behavior is difficult to understand.
As it has been mentioned in the theoretical analysis, this might be due to a hysteresis effect. A possible source of this hysteresis is the internal friction in the motors' reduction gear ratios. When the reduction gear is not in motion, it takes more force to overcome the static friction. However, figure 39a does not show a significantly increased performance compared to figure 39b.

Conclusively, all these Bode plots suggest, that the controller itself already has a latency per se for low frequencies. The origin of this can stem from some non-linearities in the setup, or the fact that a hysteresis curve is present between input and output. This phenomenon has been tested and explained for the second controller (see section 8.5).
The tracking behavior in figures 41a and 41b is still not perfect, which suggests further gain tuning.

A thorough case testing with several filters implemented can be seen in figure 43. The filter that has been changed here is the one that filters the error of the tracking for the derivative part.
This is a simple digital filter based on the formula in equation 26, where $x[t]$ is the measured distance at time step t and $y[t]$ is the filtered measured distance.

$$y[t+1] = \alpha x[t] + (1-\alpha)y[t] \tag{26}$$

α determines how conservative the filter is, where the cut-off frequency can be calculated as:

$$f_c = \frac{\alpha}{(1-\alpha)2\pi\Delta T}$$

The cut-off frequencies do not have a major impact on the tracking performance, which is why the first filter has been implemented ($f_c = 1.6$Hz). This digital filter has a slightly weaker performance than a general first order lag of the form $H(s) = \frac{1}{1+\tau s}$.

5.13 Performance Evaluation

After having found gains that seem to work well enough for the desired application, the game controller setup was put to the test. At first, the controller was connected to the computer to

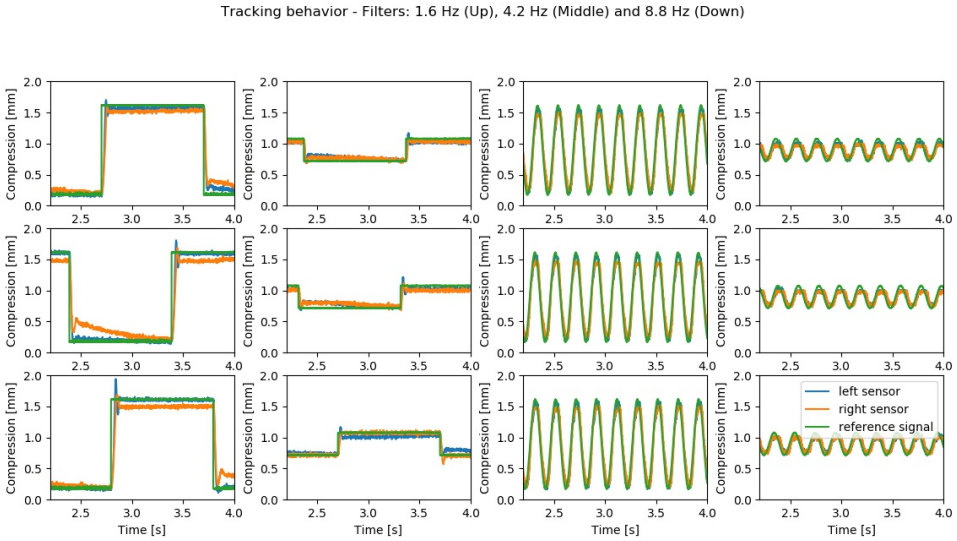


Figure 43: Tracking performance of PID game controller with different derivative filters.

communicate with the Topy robot.

5.13.1 Latency

The controller was successfully able to navigate the robot, albeit with a delay of roughly 700ms. The commercial controller, that was developed for the Topy robot, also had a certain delay of almost 500ms. This delay is the difference between the instant when the joystick is pushed forward, and the moment when the robot starts moving. The feedback methods that have been tested varied between a pure pitch feedback, a combined pitch and roll feedback, and a current consumption feedback law (see section 4.6).
The effect and magnitude of the command latency, but also of the feedback latency can be seen in figures 44 and 45.
In these figures, several interesting things have to be mentioned. First of all, the delay between sending the commands and receiving the consumed current value, which is then used as feedback value, is roughly 700ms. The delay between the received feedback value and the response of the distance sensor is much smaller and depends on the situation of the robot. When the robot starts to move, it takes some time until the current has built up, and the latency between the reference compression and actual compression is roughly 130ms for the P-controlled, and 430ms for the PID-controlled setup (see first red vertical lines). However, when the robot is stopped, the latency is 20 - 30ms (second and third red lines).
This latency is the actual controller's response latency and shows repeatedly, that the controller with its actuation implementation is not the bottleneck of the system.
In the first green lines, an external force has been applied to the robot, simulating an obstacle, which increased the consumed current and therefore also increased the desired feedback value. Again, the latency (between reference compression and actual compression) was of roughly 30ms. The second green lines indicate where the external force has been removed and the current abruptly dropped back to its normal value. The latency here was below 10ms.
Also indicated in figures 44 and 45, one can see that the feedback has been turned off when driving backwards (black lines). However, in this scenario the feedback has been activated when standing still, ie. sending 0 as driving speed. Since the robot is only sending the magnitude but not the direction of the current in the crawlers, the remaining current from the backward motion is fed

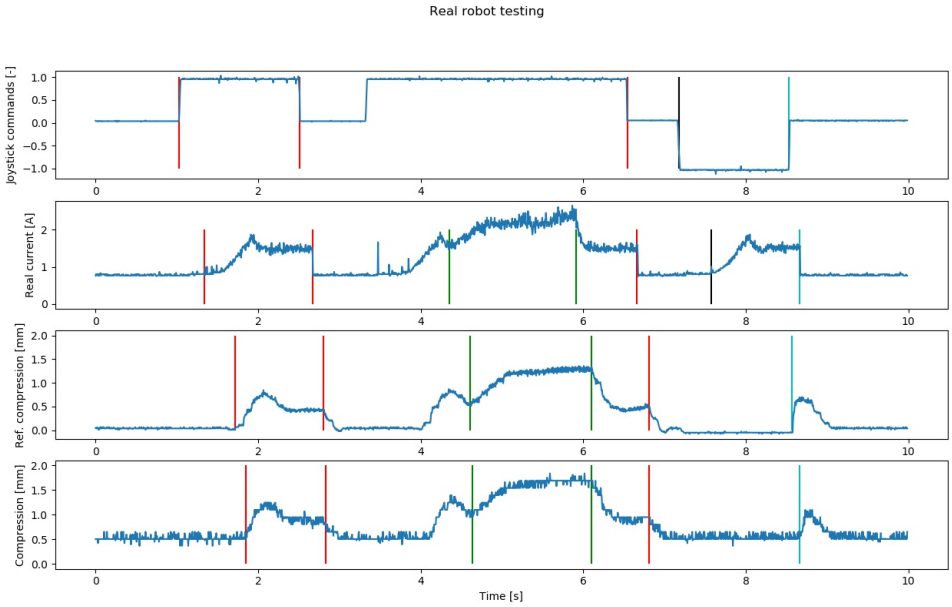


Figure 44: Command and behavior of Topy robot with P-controlled game controller.

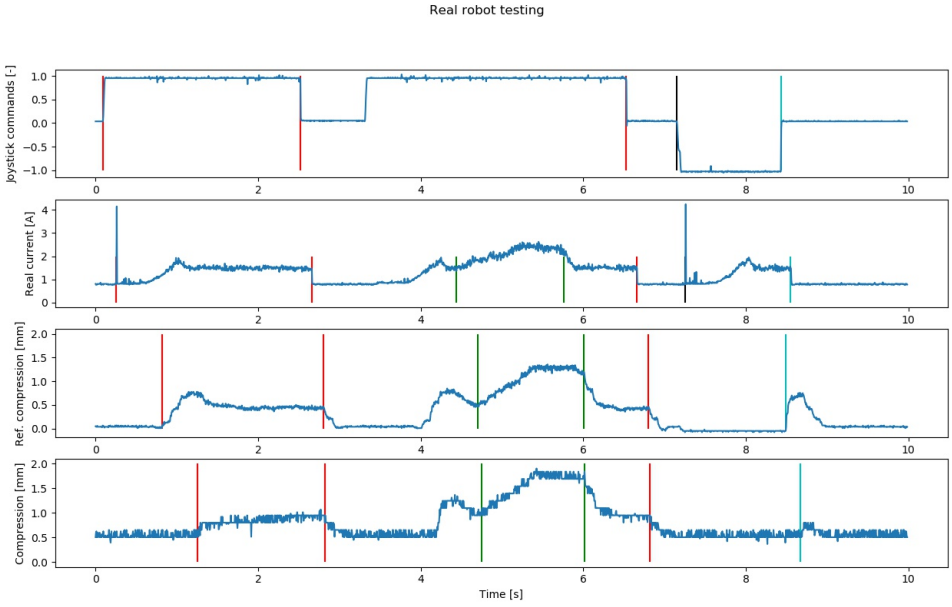


Figure 45: Command and behavior of Topy robot with PID-controlled game controller.

back. This mode can easily be turned off, so that no feedback is felt when the robot is halting. For the sake of completeness however, this has not been turned off and it is interesting to note, that the time between switching the command values and the reception of the feedback value has been reduced to roughly 50ms.
From these experiments one can conclude that the robot takes a long time to start driving, mainly due to internal implementation of the crawler control scheme. This is completely decoupled from

the controller and cannot be changed in this work.
The total delay for this initial start-up is 820ms for the P-controlled and 1160ms for the PID-controlled version. The communication delay (joystick command and robot's reaction) is 210ms on average. The time between sending the joysticks commands and feeling the feedback is 240ms (P) and 280ms (PID), when excluding the initial start command. The full data has been gathered with an oscilloscope and can be seen in table 6.

	Start	Stop	Ext. force	No force	Stop	Back	Stop	Control
Joy-stick commands sent	1.03	2.51	-	-	6.54	7.18	8.53	P
Measured current in robot	1.34	2.67	4.35	5.91	6.66	7.57	8.66	P
Desired compression	1.72	2.80	4.61	6.10	6.81	-	8.57	P
Measured compression	1.85	2.83	4.64	6.10	6.81	-	8.66	P
Joy-stick commands sent	0.09	2.52	-	-	6.53	7.15	8.43	PID
Measured current in robot	0.26	2.66	4.44	5.76	6.66	7.26	8.55	PID
Desired compression	0.82	2.80	4.70	6.01	6.80	-	8.49	PID
Measured compression	1.25	2.83	4.75	6.02	6.82	-	8.67	PID

Table 6: Latency table for sending joystick commands, current measured on the real robot, desired compression and measured compression. All values indicated in [s].

5.13.2 Real-Feel and Intuitiveness

Despite the command delay, the feedback from the robot can be felt almost instantly in the user's palms. For example, as soon as the robot is moving forward, one can feel the current building up in the current consumption feedback mode.
The difference between the PID and P-controlled controllers is small, but can still be felt. In the PID-control scheme, one can feel an asymmetry between the left and right palms. This is due to the distance sensors that have different threshold values and the fact that the PID gains have been tuned for one side only. To avoid this asymmetry, it is recommended to identify different gains for the two sides or use sensors with more similar threshold values.
The proportional control scheme is already capable of giving an intuitive feedback of the robot's state. Especially since the feedback value is continuous and does not change much over time for all feedback modes. For these reasons, it seems appropriate to leave the controller P-controlled only. Complementarily, one can also test a PI-controlled setup.

5.13.3 Stability

Both control schemes are stable for various grips and feedback values, but in some cases one can feel and hear a slight oscillatory behavior in the PID-controlled setup. This suggests that the gains are not optimal and that they can further be tuned in future research. However, it is not possible to render the setup unstable even when the user is deliberately trying to do so, acting as a non-passive element. This has to do with the fact that the input using the joysticks is completely decoupled from the output (palm pads).

5.13.4 General Performance

The overall performance evaluation of this controller is rather subjective. The output force of the SEA setup is much bigger than for the voice-coil implementation, as it has been anticipated in the design phase. Since the output force is distributed over the whole area of contact of the stimulators, the user technically feels a pressure instead of the force itself. However, one can call the feedback a pseudo-force as demonstrated in the previous research of this project [10]. One

important psychological aspect is the area of the stimulators. If they are too big, the pressure is much weaker and the pseudo-force is too weak. If the area is too small, the feedback becomes punctual and the edges distort the feeling of the feedback, making it uncomfortable to use and counterintuitive. In this controller design the stimulators have the right ratio between output force and area of contact.
Another psychological aspect is the direction of the feedback. This controller has an angle of roughly 115° with respect to the operator's orientation. During operation the forces tend to push the palms outwards which is not entirely intuitive if one is expecting a force opposing the movement of the robot (180°). This however depends strongly on the target application and the controller design can be well-suited for other environments.

However, the overall feeling is rather good and the performances have shown that the controller was able to successfully meet all design requirements from section 1.7.

Having said that, the biggest issue in this test setup though, is the delay of the command messages. In order to get an idea of the delay of the mechanical setup (the SEA implementation), the controller has also been tested in a purely simulated environment. This eliminates (almost) all potential delays in communication between robot and controller and the results and evaluation can be seen in the next section.

5.14 Unity Simulation

In the previous research, a controller testing environment has been created. It consists of several different landscapes, that include a crawler-based tank, which can be navigated in its world. The feedback is purely based on the roll and pitch angle of the tank.
The environment that has been used for testing is a dune-like desert with small hills to overcome. The feedback that is sent to the controller is different on each side, in order to increase the intuitiveness when driving with a tilt (roll angle). This allows for easy detection of obstacles, even when no visual feedback is given.

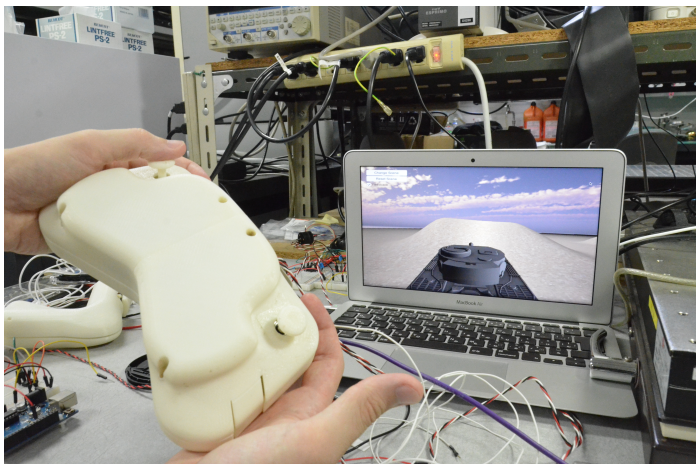


Figure 46: Unity simulation test setup.

For testing the controller in the simulated world, the Unity program from the previous research of this project has been used. With minor modifications of the control software, the controller could be tested with different feedback for the two stimulators. This time, only the combined roll-pitch feedback law has been used.

5.14.1 Latency

Similarly to the real-world testing of the controller, one can analyze the reaction time of the feedback with respect to the user's input. This time however, the current of the tank has not been measured, instead the vehicle's orientation was used. Furthermore, the joysticks do not have a direct impact on the feedback.
Nevertheless, in figure 47 one can see that the change in the joystick commands happens almost simultaneously with the change of the feedback. This is because the robot was stopped briefly, as soon as the hill was overcome. Also in this figure, one can see a slight oscillatory behavior in the feedback around 5s. This stems from the simulated damping system of the tank, which keeps oscillating and changing its orientation after being stopped abruptly.

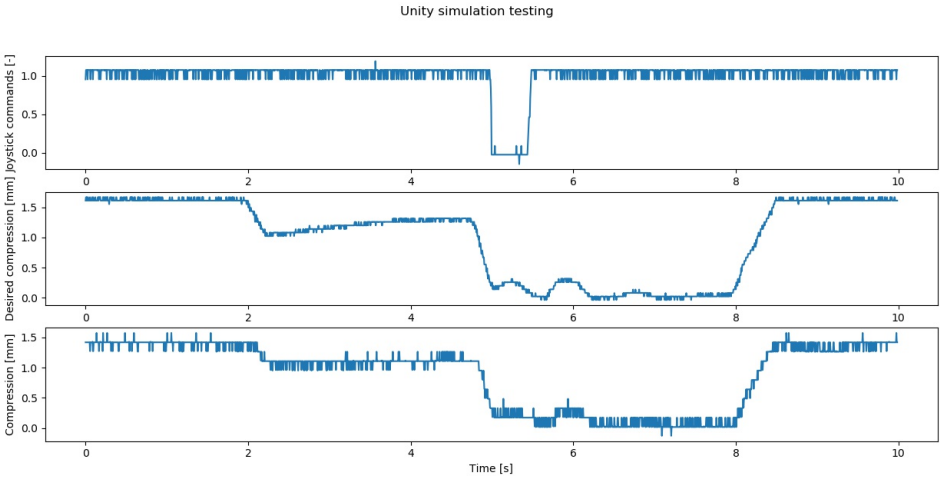


Figure 47: Command and feedback in Unity with P-controlled game controller.

Figure 48 shows almost the same behavior, this time with a bit more delay between the moment where the joysticks are released and the moment where the feedback drops. This can be due to the integral effect of the control scheme, as well as the filters used, or much more likely due to imprecise robot navigation. In this case, two filters have been used in the software, one for the derivative parts on the distance measurement, and one for the feedback value. As it can be seen at around 9 seconds, there is almost no oscillatory behavior after the robot has reached the horizontal plateau.

Important Remark on Unity Results Figures 47 and 48 are to be read with a grain of salt, since no direct correlation exists between the joysticks commands and the feedback values. The environment cannot be represented on these plots and no further variables could be measured, due to restricted access to the simulation code.

5.14.2 Unity Performance Evaluation

In the Unity simulation all hardware latencies have been eliminated and the simulated tank reacted almost instantly to the user's command and its orientation angles are fed back. Only the serial link cable is left as bottleneck of the speed and latency performances. Even though the serial cable is capable of communicating with a Baud rate of 250000 bits per second, this might slow down the system, if a lot of messages are being sent back and forth. The overhead reduces the achievable control frequency of 4kHz (calculated for sending 8 bytes per communication frame) and the error rate increases with higher frequencies. For comparison, the message size with the real robot is 115 bytes, not counting the overhead (maximum communication frequency of 250Hz).

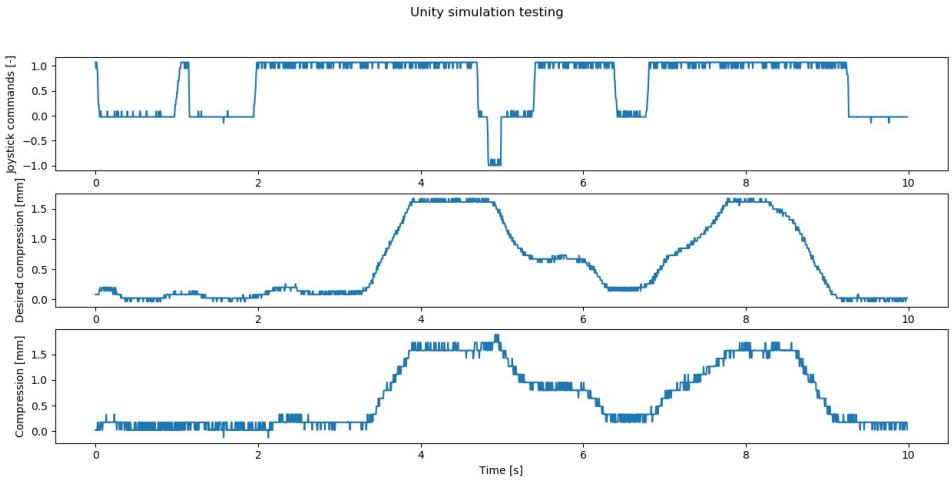


Figure 48: Command and feedback in Unity with PID-controlled game controller.

However, there is no latency perceptible, neither when sending the commands, nor when receiving the feedback. This simulation-only test shows that the latency reported in the previous section does not stem from the implementation of the series elastic actuators and justifies the choice of this actuation system.
The tracking performance is good for both P and PID control schemes and the feedback feels intuitive. The real-feel of the controller is better than in the experiment with the real robot and even the maximum output force could be achieved when driving over the steep slopes of the environment. The range of the output force is appropriate for the desired feedback. Obstacles or slopes can be felt individually and intuitively on both sides.

5.15 Overall Evaluation of the Game Controller

Overall, one can say that the controller has achieved the performance requirements stated in the beginning of the project (section 1.7).
The advantages of using SEAs are mostly the increased maximum output force and the decreased weight (VCMs tend to be very heavy due to the magnets). A drawback however is the size of the controller. The controller has been designed with several margins for dimensions and distances, leading to this bulky setup. If necessary, the design can be optimized in future work, which would result in a much smaller controller.
Both control schemes (SEA and VCM) are straightforward and the output force can be controlled easily.
The generic disadvantages of SEA implementations are the control speed and reaction time of the mechanical setup. However, as it has been shown in both testing environments, the mechanical response time is not the bottleneck for this particular application.

Even though the maximum output force is enough to give a good feedback, it can be further increased. The motors that are used currently are not capable of compressing the springs to their maximal deflection. Therefore, it is enough, to switch out the motors to have a higher torque. When doing so, it should be verified, that the control speed of the motors do not affect the overall response time.
From the psychological points of view, the only shortcoming is the feedback direction, which is application-specific. For this reason, and in order to write and test the parameter choice sugges-

Designator	Requirement	Achieved	Unit
Output force	> 10	10.8	[N]
Response delay	< 160	50	[ms]
Maximum size	$< 200 \times 150 \times 100$	$220 \times 110 \times 70$	[mm×mm×mm]
Maximum weight	< 0.6	0.55	[kg]

Table 7: Controller performance requirements and achieved performances.

tions for future research projects based on the current findings, a second controller has been designed.

6 Design Suggestions

This section provides a guideline for important design parameter choices, if one wants to design a similar haptic feedback controller based on series elastic actuators. It is mainly based on the findings from the game controller but also includes the main results of the second controller, called yoke controller.
Since the design strongly depends on the target application, it is assumed to use the controller for robots similar to the Topy robot.

6.1 Software and Control Choices

In order to introduce no latency for control commands, it is important to have a high communication speed with no delay. In the current Arduino setup, one can decrease the command delay by using interrupts for the joystick commands. For immediate feedback, one can implement a feedforward control scheme that is fed from the joysticks' positions directly to the stimulators.
The bottleneck of this setup was the fixed communication frequency of the Topy robot of maximum 5Hz. Due to the small changes of the feedback value, this operation frequency is still acceptable[6]. However, if one expects a highly fluctuating feedback, a higher communication frequency has to be opted for.
The suggested control scheme is a normal proportional controller, mainly due to the fact that it is bothersome to fine-tune the PID gains and because the PID performance benefits do not outweigh the performance of the P control scheme.
For haptic applications, it is suggested to have a motor control rate of at least 1kHz which is the limit for the Arduino. If one opts for higher control frequencies, it is recommended to switch to an *Mbed* device or similar devices.
The joystick to feedback (ie. input to output) latency is around 300ms. To reduce this latency, one can think about implementing a feedforward from the joysticks in the control algorithm. The theoretical latency has been evaluated in the Unity test environment, but the results are to be taken with a grain of salt. If one is looking for a in-depth analysis of the latency, a more thorough testing is suggested. Also, it seems unavoidable to alter the Unity test software in order to gain access to main parameters, such as the terrain details (ground truth inclination), reception of commands or sent feedback both on the simulated tank-side.

6.2 Mechanical Parameters

One of the first mechanical design choices is the design of the controller itself. The PlayStation like controller has been chosen for consistency with the previous research, but also for the fact that most conventional controllers are based on this design. It is not required to copy this design, which is why a different approach has been chosen for the second controller design.
The feedback direction is target-application-specific and results from the design. For the desired application, it is recommended to have a direct movement-opposing force, as it is the case for the yoke controller design.
The weight only plays a minor role and any weight seems to be acceptable, as long as the operator can comfortably handle the controller.

The target point of contact with the user's palm is between the Mars and Venus region[7]. This area is sensitive enough and can have a typical indentation of 5mm which needs to be taken into account when calculating the necessary stroke of the stimulator. The palm pads' areas should measure around 5 to 10cm^2.

[6] This has to do with the fact that the current can only change continuously and comparably slowly in the two crawlers. For other applications this cannot always be assumed.

[7] accessed (2018, July 26th), `https://www.yourchineseastrology.com/palmistry/mounts.htm`

Based on the findings from the previous study in the same lab, the handheld controller can successfully overcome the issue of common haptic interfaces where the reaction forces have to be dealt with. It has been shown that the device does not have to be grounded in order to only create the feeling of being pushed and not pulled.

Another important element is the spring system. It is recommended to have a symmetric arrangement with springs of equal spring constant. In addition, one should assemble the springs with utmost precision to ensure symmetry, in order to reach linear behavior during compression. The length of the springs does not seem to be very important, as long as the compression is constrained to the perpendicular axis only. With the tested springs it has been concluded, that a length of 10mm is at the limit of the acceptable range, but shorter spring lengths are suggested. Shorter springs also result in a more compact design which is generally favorable in most designs.
Testing different deflection ratios has shown that the target stroke (not necessarily full compression) must be achievable and that it is better to have a margin if one wants to change the motors used to increase the output force.
The spring coefficients can vary and tests between 2 and 24N/mm have shown promising results[8]. The designer can choose the springs according to the desired output force and the target compression of the springs.

The motors greatly influence the performance of the controller. The reduction ratio should be high enough to guarantee the target output force, while ensuring the desired control speed. The target output torque can be used to calculate the output force approximately. However, one should keep in mind that some energy is lost in friction and efficiency of the motor and gear assembly.
Furthermore, non-backdrivability in the motors can lead to a more energy-efficient system, since the voltage level only has to be provided to load the springs and the energy is then stored in the compressed springs, being blocked by the irreversible gears of the motors.

6.3 Electrical Parameters

The electric components that have been used are mainly the potentiometer for the joysticks and the photoreceptors for distance sensing.
The photoreceptors have very different characteristics and all thresholds need to be identified individually. Furthermore, the output-to-distance function is not linear for a big range of distances and the sensitivity does not stay constant. If a high performance is expected from the controller, it is recommended to find a different solution or a better performing sensor.
Furthermore, it is advisable to filter the motor commands to convert the Arduino output PWM to a more steady voltage level, to protect the amplifiers from overfitting the signal.
Further, it was necessary to implement a voltage follower (a simple operational amplifier with unitary gain) in order to reduce interfering effects on the distance sensors. The sensors namely showed a very strong correlation with the reference signal given by the wave generator.

[8]These results stem from subjective evaluation of the output force, as well as the theoretical calculation of the transfer function.

7 Design Phase of the Yoke Controller

7.1 Working Principle

The second controller uses a cam follower principle as actuation transmission. A mechanical cam is attached to the motor shaft, which pushes a lever. This lever is directly linked to the springs and the palm pads. The SEA system stays globally the same as in the first controller, only the transmission principle has changed. In this case the motor can only provide positive forces, which leads to a compression of the springs. It cannot decompress the springs faster, even when a negative voltage is applied. This is because no mechanical pre-load has been applied to the springs and the follower (lever) only touches the cam with no linkage.

7.2 Design and Parameters

When designing this controller, the design of an airplane yoke has been used as a model. Therefore it is from now on referred to as the yoke controller. The first draft of the design can be seen in figure 49.

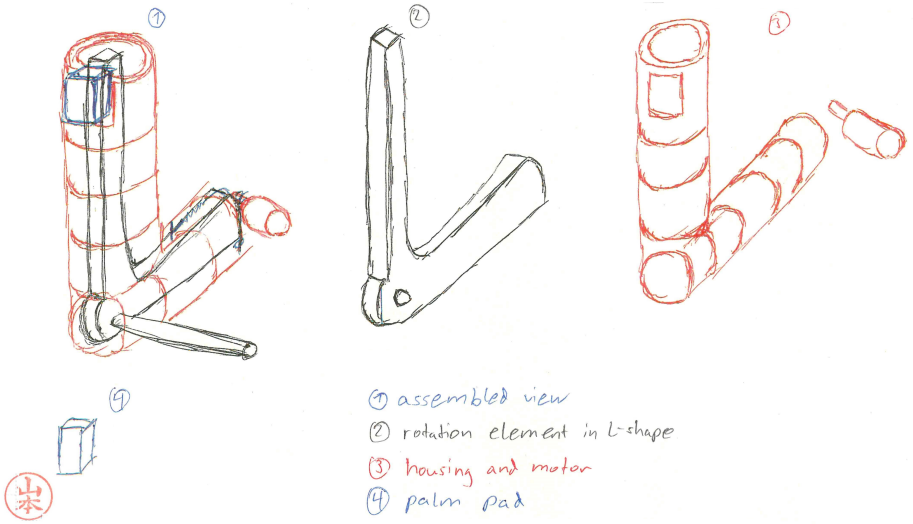


Figure 49: First draft of the yoke controller.

In order to have a nice fit to the palms, the circumference has been chosen to be close to the airplane yokes' or conventional car steering wheels' circumferences. The joysticks' position is such that the stimulator touches the same area of the palm as in the previous controller and research (Venus and Mars area of the hand). This time the feedback direction is perpendicular towards the user, which ensures a more intuitive feeling for the desired application.
The most important parameters in this system are the length of the lever and the springs. For the prototype, the lever's length has been fixed to roughly 100mm and a set of four springs with 1N/mm each have been used. Their length is 10mm and they have an allowable deflection ratio of 40%.
The stimulators' area is kept almost the same with 7.5cm^2.
The motors play an important role again. For this design, the motors with the reduction gear of *33:1* have been implemented.
It has been calculated that an output force of 10N can be achieved if the rotation cam has a radius of 5mm and the lever has a length of 90mm between the axis of rotation and the motor, and 60mm between the axis of rotation and the springs.
In order to achieve a bigger output force, a smaller radius of the rotation cam can be chosen.
The angle of the rotational L that was targeted is 10° for the full stroke. With the current design of

the rotation cam, the motor has to move roughly 100° to achieve the full stroke. With the motors characteristic turn speed, the palm pad control speed of 110ms for the full stroke can be calculated. This corresponds to a speed of 10cm/s.
For designing the rotational cam part, a spiral-like freeform has been drawn to keep the angle-to-distance ratio as linear as possible.

The operational distance for the photoreceptors is between 5.5 and 10mm. As before, the operational range of the photoreceptors can be identified.

	Sensor reading MAX (rest)	Distance (rest)	Sensor reading MIN (compression)	Distance (compression)	Max output force
Left side	780	10.5mm	550	5.5mm	20N
Right side	763	10mm	600	5.5mm	18N

Figure 50: Identified operational range for the photoreceptors in the yoke controller.

The 3D model of the left-hand side yoke-based controller can be seen in figure 51. To see the inside of the controller, the cover has been removed. For assembly, three sets of screws and nuts are necessary, making it easy to assemble and take apart. Moreover, this design is smaller in volume than the game controller. It is therefore more slender, lighter, cheaper and faster to manufacture.

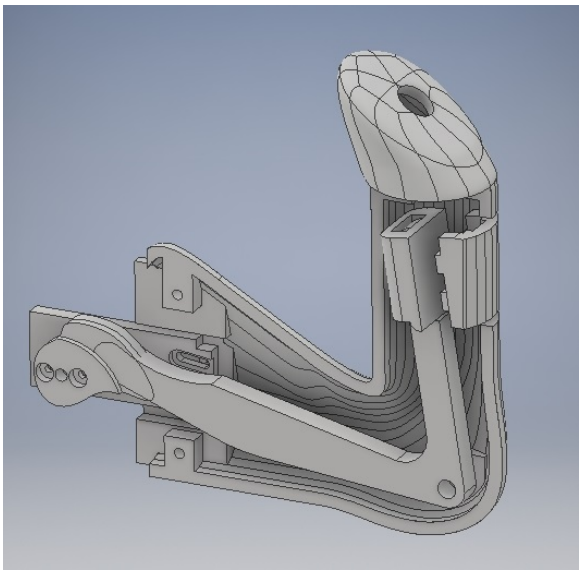
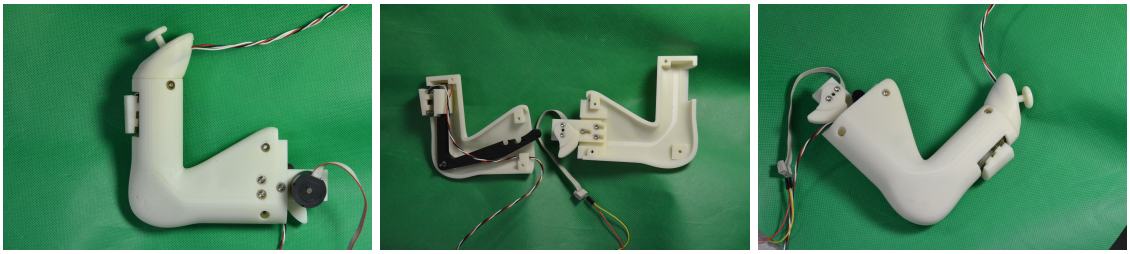


Figure 51: 3D model of the left-hand side controller with a removed cover.

In this design, the feedback motion is perpendicular towards the user and opposing the direction of motion. In fact, there is a slight angle at which the force is acting, due to the rotational design in the controller. However, due to the large lever, this effect can be neglected.

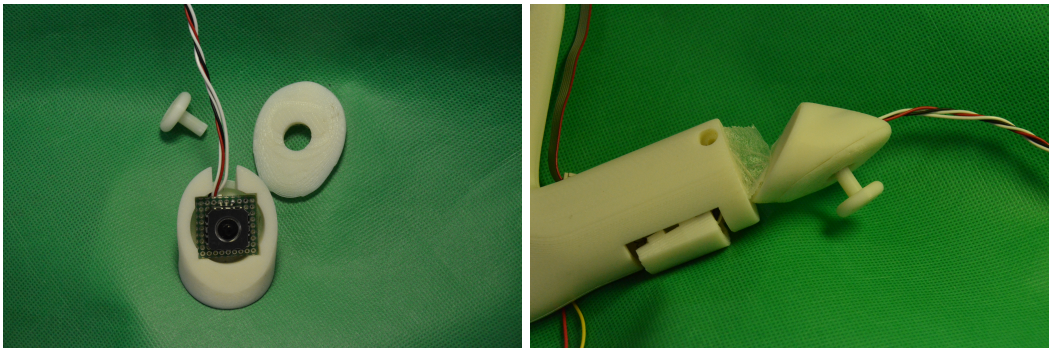
7.3 Printed Version

The printed version of this controller can be seen in figure 52.
As it can be seen in figure 52b, the controller consists of very few parts. For one, there is the joystick mount, which has preliminarily been taped to the cap of the casing of the controller. This can be seen in figure 53.



(a) Yoke controller front side. (b) Yoke controller opened. (c) Yoke controller back side.

Figure 52: 3D printed yoke controller assembled and with cover removed.

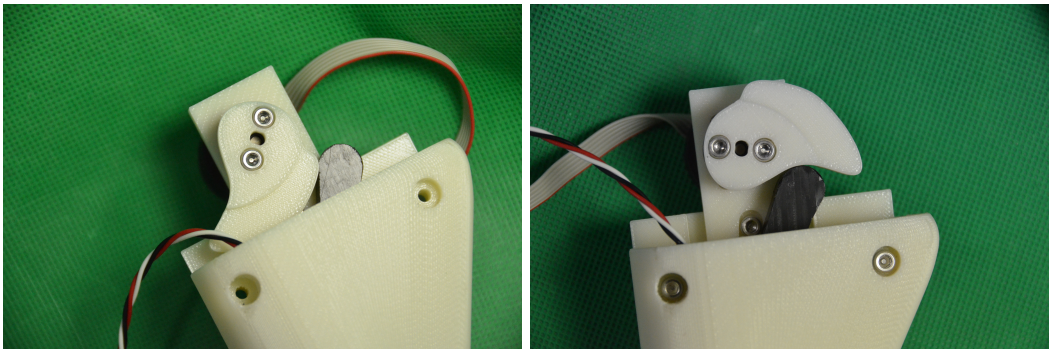


(a) Joystick mount disassembled. (b) Joystick mount taped to casings cap.

Figure 53: Joystick mount for the yoke controller.

Then there is the lever element, referred to as the rotational L (black part in figure 52a), to which the spring plate is attached. The springs are on one side glued to this spring plate and on the other side glued to the palm pad. A slot within the palm pad serves as a housing chamber for the photoreceptor and its circuit, thus allowing to measure the distance to the spring plate and therefore the compression of the springs.
The motor is screwed to an element called the motor plate, which is attached to one side of the controller casing. The oblong holes in this plate are used to adjust the distance of the motor with respect to the lever. Therefore the output force and control speed can be fine-adjusted.

Another important element is the rotational cam. This is directly attached to a shaft collar, which is fixed to the motor shaft with two set screws.
The rotational cam actuates the rotational L by simply pushing it. This is subject to frictional effects and can have some impact in the final performance. However, this was a risk that has been accepted with the design choice.
Figure 54 shows the assembled version of the rotational cam.
As it can be seen in this figure, the rotational cam can rotate and actuate the lever until the lever is blocked by the walls of the controller itself (figure 54a). However, if a negative voltage is applied, it rotates into the other direction, which causes it to lose contact with the lever (figure 54b). Since it is only blocked by the rotational L on the other side (ie. $-180°$ rotation of the cam), this may cause a long delay in operational mode. This phenomenon has been explained in chapter 8.8 and is termed the build-up latency.
When the controller is in use, the two sides can be held individually and are not linked to each other. This allows for more flexibility and simplicity of use. Figure 55 depicts this situation. The area of

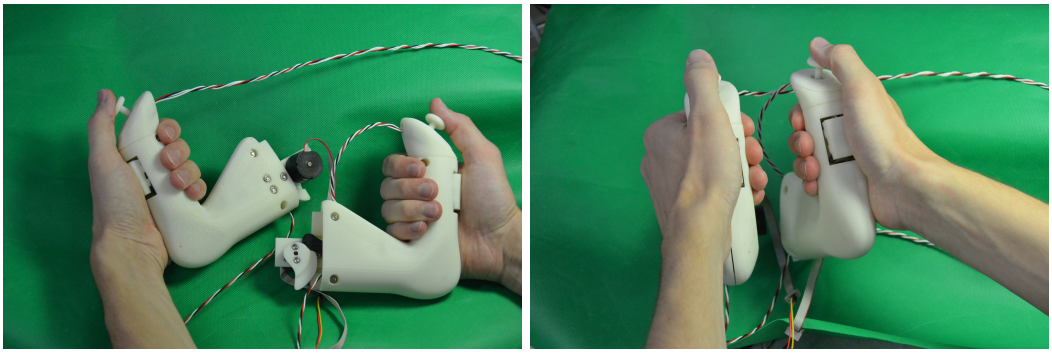


(a) Operational mode: Cam pushing the rotational L.

(b) Build-up latency: Back-rotation until blocked by rotational L.

Figure 54: The rotation cam element (black-side view).

contact with the hand, as well as the feedback direction can also be seen.



(a) Side view.

(b) Front view.

Figure 55: The pilot controller in use.

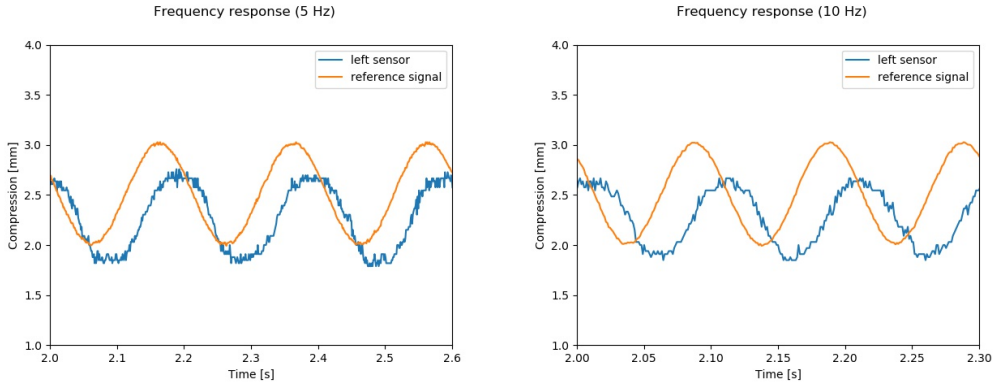
8 Testing of the Yoke Controller

8.1 Test Setup

Similar to the test setup described in section 5.3, the yoke controller has been tested. The major changes that have been made were the threshold values of the photoreceptors and the gain values. Again the Function Generator **SG-4115** has been used to create a reference signal and the same frequency range has been tested. The palm pad has been blocked by a wall. Due to the fact that the controller is virtually identical on both sides, only one side has been evaluated. For the sake of the argument, we shall call the tested controller the left side.

8.2 Tracking Behavior of the P-controller

The tracking behavior of the P-controlled ($K_P = 10\text{V/mm}$) yoke controller can be seen in figure 56.



(a) $f = 5\text{Hz}$. (b) $f = 10\text{Hz}$.

Figure 56: Tracking behavior of the P-controller ($K_P = 10\text{V/mm}$) for different frequencies.

Analogously to the previous testing, the tracking performance has been studied for different reference input. The test cases where a sine wave as well as a square wave, for both, low and high amplitude. The results can be seen in figure 57 and 58. Due to the nature of the P-controller, the low reference values cannot be tracked well enough for a low gain. This can be seen in figure 57 on the left-hand side. In order to show that the software control is not the problem, the motor voltage has been plotted as well.
This also shows that there is a non-negligible static friction in the system, since the motor voltage of up to 10V does not result in any movement of the cam.
For a higher gain, small and high reference values are tracked, albeit with a time offset. The trade-off for this parameter tuning is the speed and stability in favor of controllability of the output force.

8.3 Bode Diagram

Again, the results of the frequency response testing can be plotted in a Bode diagram. In figure 59 one can see that the tracking is acceptable for low frequencies. The gain does not fall below -4dB until it reaches the resonance frequency around 8Hz. The cut-off frequency is around 8Hz. The phase lag starts at $-30°$ and reaches $-200°$ for the far end of the tested frequency range. The anomaly at 100Hz can be explained with the noise in the distance sensors, that becomes non-negligible for such small displacements.
The identified phase lag at 1Hz corresponds to a delay of 80ms. This can be confirmed by the

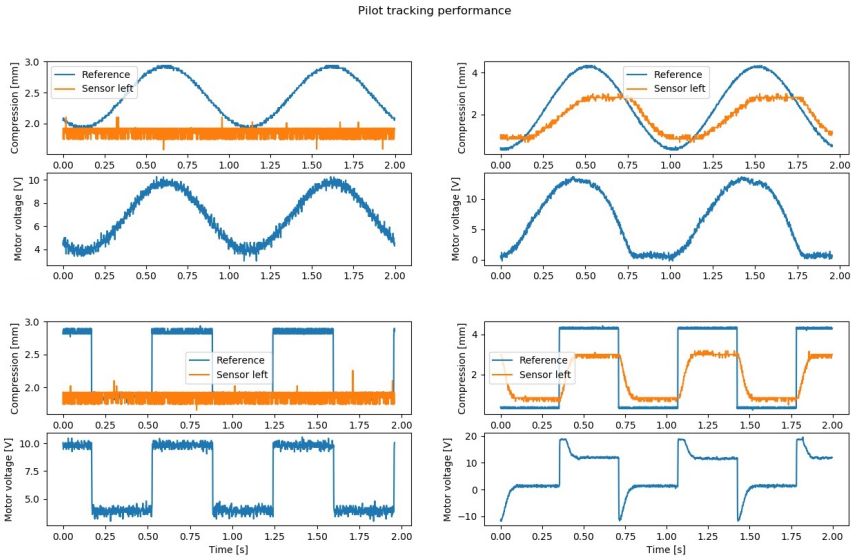


Figure 57: Tracking performance for different reference signals with a $K_p = 5\text{V/mm}$.

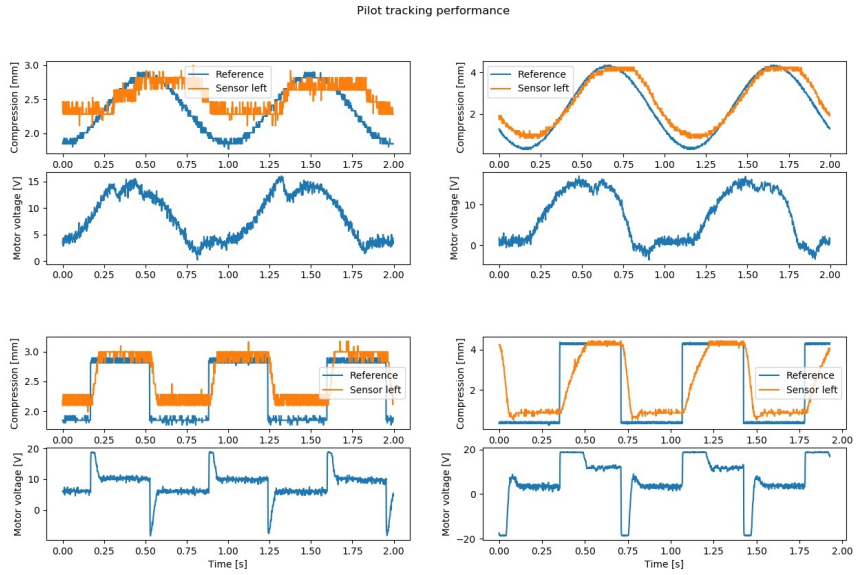


Figure 58: Tracking performance for different reference signals with a $K_p = 15\text{V/mm}$.

sine wave tracking performance in figure 58. The square wave confirms this delay only for small to medium compression references (ie. up to 2mm).

8.4 Discussion of P-Tracking

As explained in the previous testing set of the first controller, it has been suggested that the PID controller does not significantly increase the performance of the feedback. For this reason, only a P-controlled yoke console has been tested in this case.

In figure 60 a saturation behavior arises for very low frequencies. One could argue, that this stems

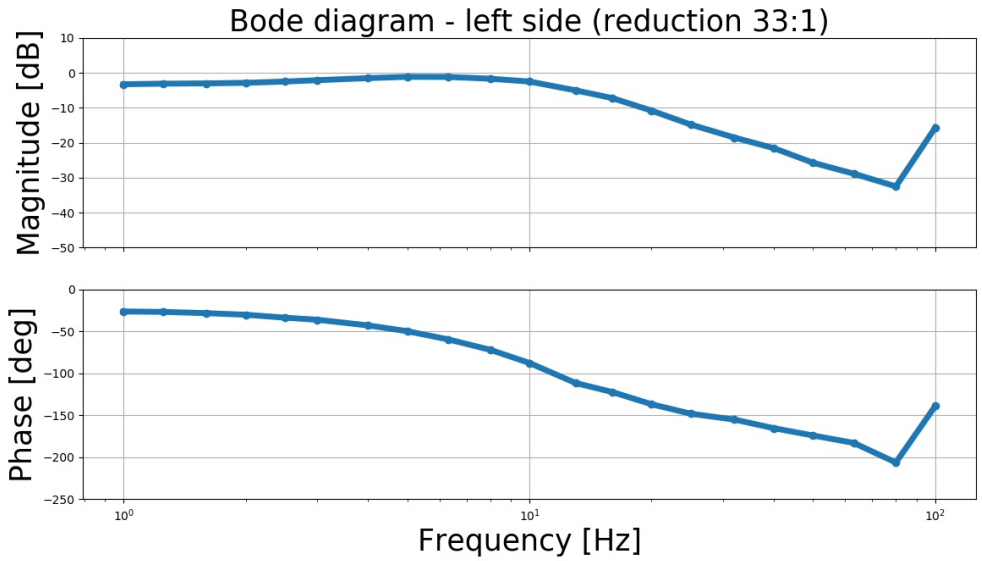


Figure 59: Left-hand side, Motor reduction ratio *33:1*

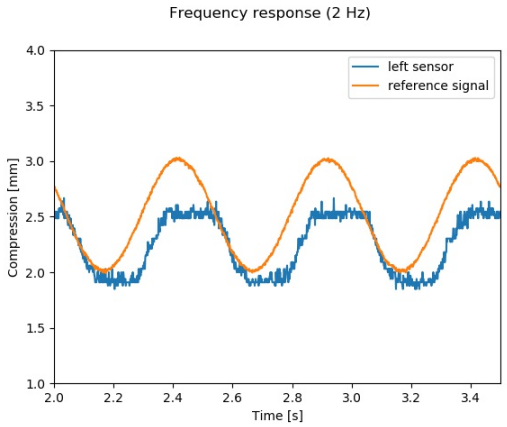
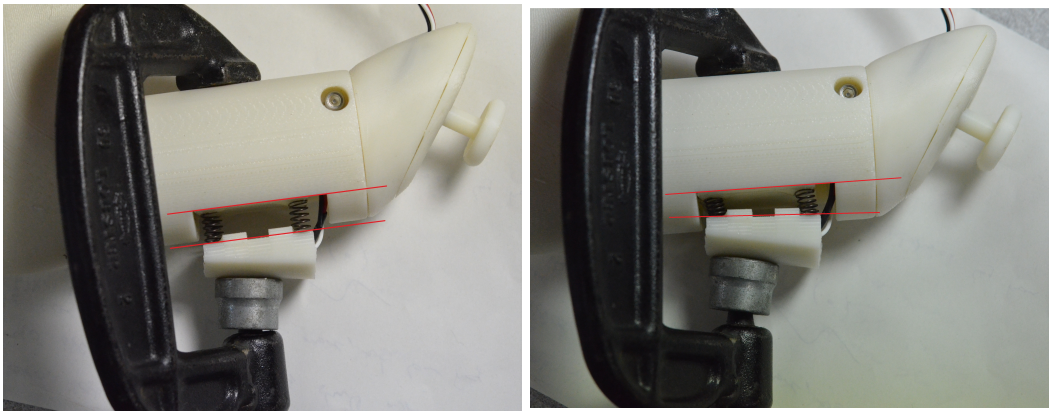


Figure 60: Tracking behavior of the P-controller ($K_P = 10\text{V/mm}$) for 2Hz.

from the fact that the reference value that has been fed into the system was too high and the motors' capacities have been saturated[9]. However, this does not apply here, since the maximum reference of 5mm has been measured as the steady-state 20V resulting displacement. Therefore, it should theoretically be achievable. From this one can conclude that either there is a non-repeatability in the system, the system shows a non-linear behavior, or a combination of both. Additionally, one can also argue that there is a significant difference between steady-state and dynamic operation for low frequencies.
Furthermore, due to the length of the springs, and since the compression procedure of the springs is not guided, it seems plausible to assume that the palm pad's movements are not completely perpendicular to the spring plate. In order to understand this issue, figure 61 can be studied. Therefore, the distance does not change linearly and the full compression cannot be achieved all the time. For the test setup a C-clamp has been used in order to block the palm pad. Since this

[9]Note, that for the frequency response analysis, a smaller reference signal has been used

produces only a punctual counterforce, this effect is magnified. In operational mode however, it is expected to have a more linear behavior, since the palms of the user can create a more uniformly distributed counterforce.



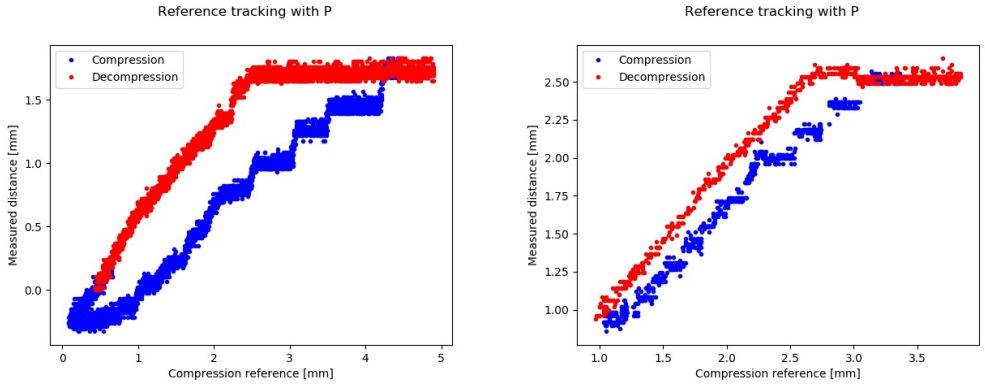
(a) Before compression. (b) After compression.

Figure 61: Indicated angles of palm pad with respect to spring plate to show the C-clamp effect.

In figure 60, it is visible, that the real compression is constantly slightly below the reference value. These findings have been confirmed in the Bode diagram from figure 59. This is due to the implementation of the P-only control scheme with no integral part that could remove the steady-state offset.
However, since the steady-state offset between reference and tracking is of less than 0.2mm (ie. 0.8N) it is negligible and therefore, an implementation of integral and derivative part does not seem necessary.
For the right-hand side, it is assumed that the tracking behavior stays similar and the gains shall be identical.

8.5 Hysteresis Effect

As it has already been explained, there is an expected hysteresis effect which causes a non-linearity and therefore also a latency in the system.
This hysteresis can be shown by simply plotting the input to output tracking for a linearly increasing reference. In theory, a perfect controller should have an identity function where $x = x_{ref}$, the output matches the input. However, for this system this is not the case. In fact, there is a hysteresis effect which does not map a single x value to a given x_{ref} value. The effect of this can be seen in figure 62.
From figure 62, one can see the counter-clock wise rotation of the hysteresis, which is an indicator for the phase lag. In figure 62a the measured distance goes below 0mm which is caused by non-perfect threshold identification of the photoreceptor values, and has no impact on the performance at this point.
Another key conclusion that can be drawn from figure 62b is, that the tracking is almost perfect when decompressing the springs (almost identity function). While in the other direction (compression), there is a slight offset, for example when 3mm is given as a reference, only 2.5mm can be achieved. This shows that a gain value of 15V/mm leads to a better tracking of the reference value, at least for the decompression direction. Additionally, there is a horizontal line visible for references between 3mm and 3.8mm.
This horizontal line seems to be the maximal achievable compression for this setup. This is where



(a) For a $K_P = 5\text{V/mm}$.

(b) For a $K_P = 15\text{V/mm}$.

Figure 62: The hysteresis effect for input-output tracking performance of different gains.

an increase in voltage does not result in a change in compression of the springs. Also, when testing, the voltage for high compression references (ie. 3.7 - 3.8mm) has been saturated at 20V, yet again indicating the maximum achievable compression.

In the case where the weaker gain has been tested, the tracking performance is also worse. The decompression slope is not optimal and neither is the one for compression. Already at mid-distance references, the output is far from what it should be. Also, with a weaker gain, the hysteresis effect seems to have a bigger impact.
Another effect that can be observed, is that the compression happens step-wise. It is assumed that this is due to the friction present between the rotational L and the cam, or the friction in the reduction gear. Strangely enough, this phenomenon is not visible in the other direction.

8.6 Other Non-Linearities

Apart from the hysteresis effect, there are other non-linearities present in the system. Based on the fact that the springs are decompressed more easily than compressed (due to the naturally occurring spring force, that tries to keep the compression at 0mm), the slope of the compression is less steep than the decompression and happens more slowly[10]. This will result in a skewed sine wave for a regular sine reference.
As a result, this impacts on the analysis, notably in the Bode-diagram. The sine interpolation of the measured distance is shifted to the right, due to the left-skewedness.
This leads to an additional latency in the system. Even though this phenomenon is present in any reference signal, the psychological effect has not been taken into account in this work. It is expected though, that the slower compression will be felt as normal delay.

In addition to that, the lever has been created using an 8mm thick poly-acetal (POM) plate. Even though this material shows excellent specific strength and specific stiffness properties[38], the material is deforming towards the direction of actuation. This non-infinite stiffness behavior induces further non-linearities in the system, that are difficult to cope with. The measured deflection at the distance of the spring plate in the cantilever-like rotational L is 1.5mm for 20N. These effects however, have not been taken into account in this work.

[10]Even though this is linked to the hysteresis effect, they are not the same and another issue arises, which has to be treated individually.

8.7 Motor Choice

For this yoke controller the motor with the lower reduction gear ratio has been used. This means that it is faster than the other motor in terms of tracking speed. Since the cut-off frequency from the frequency response analysis is already lower than in the previous controller, and given the fact that the output force is high enough (according to the requirement specifications in the beginning of this work), this motor seems more appropriate.
In fact, the cut-off frequency is very close to the maximum robot communication speed. To guarantee that the actuation system is not the bottleneck of the control speed, one can consider motors with even lower reduction ratios.

8.8 Performance Evaluation

The yoke controller has also been tested on the real robot. The key design difference from the first controller is the fact that the motor cannot decompress the springs faster than their natural decompression speed. When a negative voltage is applied to the motor, the rotation cam will just rotate the other way, until it is blocked by the structure of the controller. However, if one wants to increase the speed of this controller, and especially not introduce a latency if the rotation cam has traveled all the way back, for instance, a pre-load is recommended. From now on this latency will be referred to as the initial build-up latency.

8.8.1 Latency

For now, one can neglect the build-up latency, as the rotation cam stays in contact with the lever during current feedback operation with non-zero feedback (the crawler currents are never really zero).
In this operation mode, the same analysis has been done as for the previous controller. Figure 63 shows the reaction and latencies of the real robot current, the desired spring compression and the real measured compression with respect to the joystick commands. The color coding is the same as in figure 44. This time however, it seemed more difficult to find the right points in time, since the signal to noise ratio of the measured spring compression has decreased.

However, the latencies can still be analyzed. Again, the robot needs a long time to start up after the initial forward commands. 0.4s after sending the signal, the robot's current starts to increase. This is only detected and used as a feedback 0.7s after the commands have been sent. Almost 0.93s later the distance sensors measure a compression of the springs. This delay is partially due to the build-up latency that has been described earlier.
When the stopping signal is sent, the user feels a drop in feedback force with a delay of roughly 0.6s. For completeness, the entire data can be seen in table 8.
The mechanical delay, measured between the moment when the desired compression is received, to the moment where the compression is measured, is of roughly 0.3s. This seems to be much bigger than in the previous controller and is not satisfying the performance requirements. To be able to make a conclusion however, it is suggested to further analyze this delay, since only 5 data points could be measured.

8.8.2 Real-Feel and Intuitiveness

When using the controller on the real robot, the time delay between sending the commands and feeling the feedback becomes non-negligible. Even though the feedback is perpendicular to the user, and the controller fits the hands and palms of the user smoothly, the feedback does not feel very intuitive.

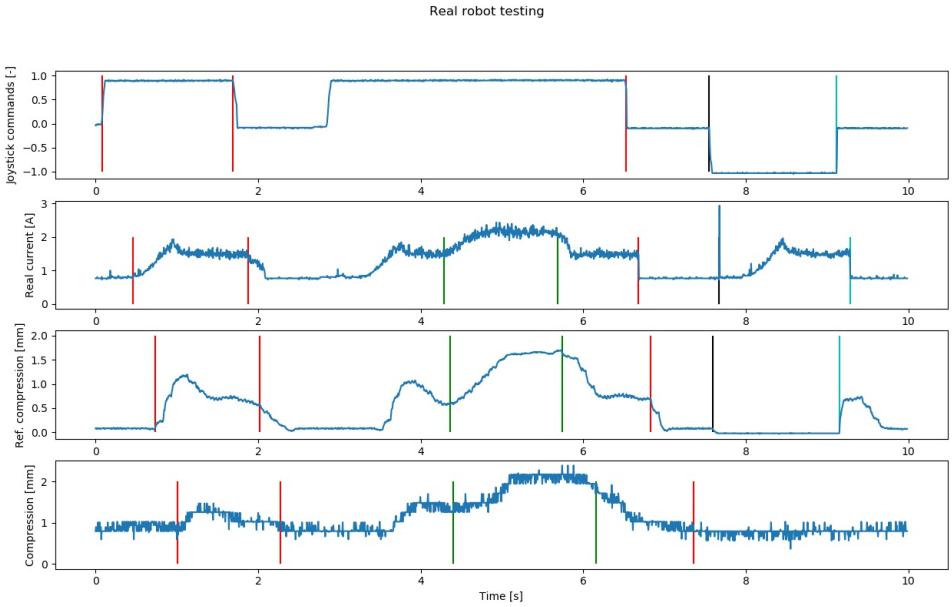


Figure 63: Command and behavior of Topy robot with P-controlled yoke controller.

	Start	Stop	Ext. force	No force	Stop	Back	Stop	Control
Joy-stick commands sent	0.08	1.69	-	-	6.53	7.55	9.11	P
Measured current in robot	0.46	1.88	4.29	5.68	6.68	7.67	9.28	P
Desired compression	0.73	2.02	4.36	5.74	6.83	7.59	9.15	P
Measured compression	1.01	2.27	4.4	6.16	7.36	-	-	P

Table 8: Latency table for sending joystick commands, current measured on the real robot, desired compression and measured compression. All values indicated in [s].

8.8.3 Stability

In terms of stability there is no problem. The controller stays stable in different situations and can resist shock or changes of the counteracting force of the user. As expected for this SEA type and implementation, stability is not an issue.

8.8.4 General Performance

The controller works, but the felt feedback is not intuitive. The time-delay deteriorates the user experience. Even though the design of the controller is visually more appealing and geometrically more appropriate for the desired tasks, the final performance of this controller seems to be somewhat worse.
This controller has traded off speed of the actuation system in favor of high output force and therefore the latency has increased as well. For a better experience it is recommended to either implement a feedforward method to overcome the latency issues and create a pseudo-feedback, or to change some parameters in the system. For one, the springs can be switched out to a set of lower equivalent stiffness. Or the necessary stroke can be reduced by replacing the 10mm long springs by springs of 5mm, while doubling the spring constant. Alternatively, the motor can be changed to have a lower reduction ratio and higher control speed. Also, it is recommended to pre-load the rotation cam and link it to the lever, to get rid of the build-up latency effects.

8.9 Unity Simulation

When testing this controller in the Unity Simulation, almost the same procedure has been used as in the previous testing part. Again the latency has been analyzed, in the assumption that all the latencies in this system come from the controller side and that the communication delay over the serial link can be neglected. Later, the real-feel and intuitiveness of the controller have been evaluated.

8.9.1 Latency

In order to evaluate the latency of the mechanical setup coupled to the Arduino, the robot has been navigated in the landscape of the Unity simulation. This time, the joystick values have not been recorded and have been replaced by the motor voltage level instead.
This time two of the ten evaluated test case scenarios are represented in figure 64 and 65.

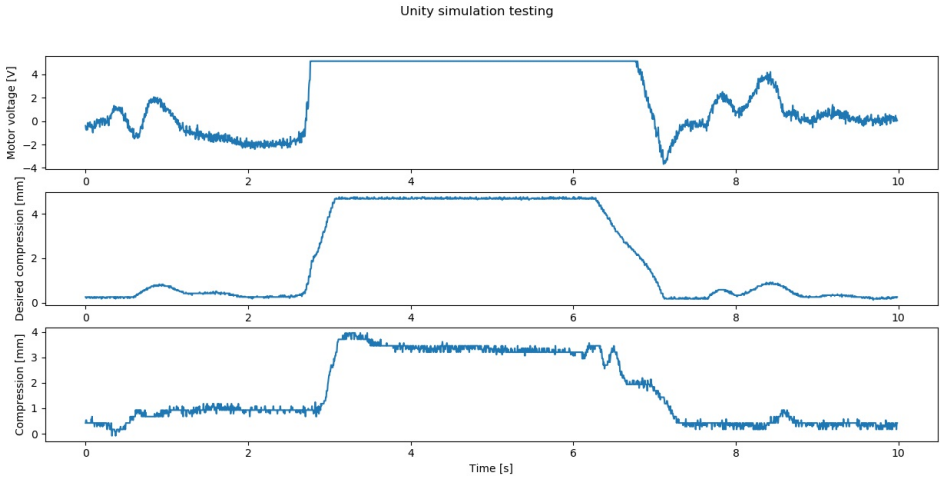


Figure 64: Command and feedback in Unity with P-controlled yoke controller, test case 1.

These plots not only show the tracking performance of the P-controller, but also the influence of the motor voltage in the compression of the springs, as well as a notion of latency. When evaluating the test cases, it has been found, that the latency (between voltage level and spring compression) is of roughly 180ms with a standard deviation of 150ms. The problem here is that again, it was difficult to find the exact points in time where the signal changes.
Furthermore, one can see in figure 64, that for a steady motor voltage, the compression oscillates. This is due to the fact that the user cannot provide an overall constant reaction force and changes the grip from time to time. This makes it additionally harder to evaluate the tracking performance of the controller.

8.9.2 Unity Performance Evaluation

When using the controller in the Unity simulation, it has been shown that the right direction of the feedback is important for the feeling of the application. The perpendicular feedback and the high force fidelity (up to a high output force) give a good overall feeling. The feedback reference can be matched well enough, with only small oscillations. These may be due to the P-controller or simply to the fact that the user cannot maintain a constant grip throughout the testing time. This means that the output force changes, depending on the grip of the user.

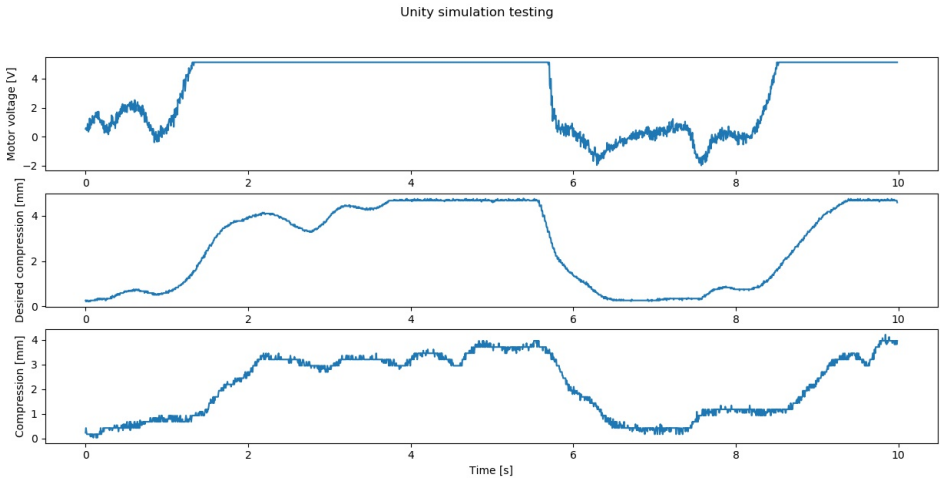


Figure 65: Command and feedback in Unity with P-controlled yoke controller, test case 2.

The latency in this simulation-only test does not distort the transparency and the real-feel of the controller is quite good. The range of the output force could almost be fully exploited and the range again seems appropriate for the purpose of this application.

8.9.3 Rubber Band Unity

state problem, idea, approach, picture, and results

8.10 Overall Evaluation of the Yoke Controller

This time, the controller has only partially satisfied the performance requirements.
The Bode plot has shown that a delay of 80ms is present for low frequencies. Testing in the real-world however, shows a much bigger delay of 300ms. Nevertheless, the Unity simulation as well as the tracking performance evaluation have suggested a lower response time of 180ms and 80ms respectively. For these reasons, it is difficult to come to a final conclusion about the latency requirements.
Nonetheless, the maximum achievable output force is much larger than in the VCM system and the changes in force feedback can be felt very smoothly (high force fidelity).
The geometry and shape of this controller is very intuitive and fits the hand well enough. Furthermore, the separated left- and right-hand side allow relatively free movements of both hands and arms of the user. In a future research, it might be interesting to make the controller wireless and battery-powered to have two completely separated portable controllers.
Again, the Topy robot's communication speed is lower than the systems control speed. However, the major issue is the previously described build-up latency. Even though it happens very rarely, the maximally felt latency can increase drastically. It is recommended to avoid this by coupling the rotational L and the cam (pre-loading).
The choice of the motors seems to be appropriate.

Designator	Requirement	Achieved	Unit
Output force	> 10	18	[N]
Control speed	–	9	[full stroke/s]
Response delay	< 160	$90 - 180$	[ms]
Maximum size	$< 200 \times 150 \times 100$	$155 \times 155 \times 50$	[mm×mm×mm]
Maximum weight	< 0.6	0.5	[kg]

Table 9: Controller performance requirements and achieved performances.

9 Discussion

This chapter summarizes the main results from the design and testing phases of both controller models. It is complementing the discussions and evaluations that have already been mentioned in the testing sections.

9.1 Game Controller

The game controller has shown promising results, especially in terms of output force and control speed. The frequency response analysis shows that the PID control scheme suddenly introduces a large lag. Hence, when choosing between the proposed control schemes, a proportional controller is advisable.
For low frequencies up to the maximum robot communication frequency, acceptable tracking behavior has been observed. Given the fact that the controller's performance strongly depends on a variety of parameters, the system can be seen as rather complex. Furthermore, the PID tuning was done in an experimental trial and error manner, which yielded only suboptimal gains. For improving the performance, a more thorough gain parameter search is advisable.
The controller clearly behaves like a second order system and shows typical characteristics, such as an asymptotic phase lag of $-180°$.
The main part of the latency is due to the communication delay and the robot's internal speed control. The controller itself does only account for an average latency around 80ms. This latency analysis was based on visual identification of rising and falling edges, which was subject to high noise in the system. Therefore, it is recommended to conduct a more in-depth analysis to precisely identify the controller's mechanically induced delay.
The overall feeling and transparency has been evaluated in a real-world scenario, as well as a simulated environment. In both cases, the controller was able to produce the desired feedback, albeit with the aforementioned delay. The user successfully feels the overcome obstacles on both sides, even though the feedback direction is not perfectly intuitive.
The stability of the controller was not an issue for any of the test cases.
This implementation of SEAs shows that this actuation system is no bottleneck for the target application, and justifies using elastic elements. The initially stated requirements concerning output force, weight and speed could be met.

9.2 Theoretical Model

The theoretical model has been used to identify a parameter that was difficult to measure. This was namely the combination of the springs' damping coefficient and the friction effects.
The P-controlled frequency response matches the analytically anticipated data. When integral and derivative parts are implemented in the control scheme however, the data differs from the analytical results.
The theoretical model has mainly been used to analyze the built controller. For future research, the mismatch between experimental and analytical data can be studied, in order to use the model as a tool for fast gain identification.

9.3 Yoke Controller

The yoke controller has a more intuitive design and feedback geometry. The output force is even higher, as it has traded off its control speed. The spring's length yield a rather large stroke which increases the control speed. Furthermore, testing has shown that the build-up latency has a big impact on the performance and it is thus recommended to link the rotation cam to the rotational L. The tracking performance of the P-controller strongly depends on the gain and reference values. However, a constant delay at low frequencies has been observed for all gains. The cam principle introduces non-negligible friction effects which causes very poor tracking behavior for small gains. Furthermore, the non-linear hysteresis effect has been studied. For small gains this effect is stronger. These and further discussed non-linearities in the system make it difficult to evaluate the controller. However, the frequency response analysis suggests that the series elastic actuation system can be fast enough in the target frequency range. A target-specific trade-off between speed and output force has to be made. For this controller it is suggested to opt for higher speed rather than output force.
The latency of this controller is dominated by the build-up effect. This can be overcome by the proposed mechanical solution. The pure mechanical latency of this controller is higher than for the game controller. However, a more thorough latency analysis is prudent.
The intuitiveness of the controller and provided feedback was given not only in terms of feedback direction, but also output force and force fidelity. Obstacles of different sizes could be distinguished successfully. The stability was not an issue either. The intuitiveness was impaired in some scenarios, where the friction caused step-wise motion and therefore step-wise changing feedback. This controller met the performance requirements, too.

9.4 Identified Potential and Applications

From testing the two different controllers, the importance of the feedback direction has become clear. For the target application, where the feedback should act as a pushing force, a perpendicular force towards the user is more intuitive. However, one may find other applications, where the feedback direction of the first controller (ie. pushing the hands outwards) is more appropriate.
An example for this could be in a handheld controller for navigating a drone. By implementing haptic feedback one can inform the operator about external forces such as wind or wind gusts in an intuitive way. In this case a combination of both directions might even be considered.
This type of haptic feedback does not necessarily have to be implemented in a remote control for mobile robots. As a matter of fact, haptic feedback can be useful in a multitude of applications. These application fields can vary from the manufacturing domain, where handheld tools can yield important information about the object being handled, over maneuvering vehicles, for example in construction site vehicles, where the feedback can tell a how much the load in an excavators shovel or forklift weighs.
This haptic feedback system could also be implemented in a steering wheel of a car, to warn the driver about objects being too close to the car, to overcome the lateral blind spots.

10 Conclusion

This work has shown that a haptic feedback controller based on series elastic actuators is, without question, accomplishable. The designed and tested controllers were successful, not only in terms of output force, but also in terms of control speed (response delay) requirements and transparency expectations.
Key elements in building an SEA-based haptic feedback controller have been analyzed and studied. Valuable insights have been gained throughout the iterative design and building phases of two different controllers. Based on this, design suggestions were discussed for future research projects in the same environment setting.
The controllers have been tested extensively and albeit, proven to work, weaknesses in design and control have been identified. To overcome these, individual solutions have been proposed.
For future research, it might be interesting to study the valid ranges from which the setup parameters can be chosen, in order to facilitate the design process. To create a full design framework, a more thorough testing is advisable.

List of Figures

List of Tables

References

[1] Asada. Study on pseudo-haptic device using tactile stimulation. Prethesis, 2016.

[2] Oxford Online Dictionaries. Definition of haptics. https://en.oxforddictionaries.com/definition/haptic, July 2018.

[3] Mandayam A Srinivasan. What is haptics? *Laboratory for Human and Machine Haptics: The Touch Lab, Massachusetts Institute of Technology*, pages 1–11, 1995.

[4] Vincent Hayward, Oliver R Astley, Manuel Cruz-Hernandez, Danny Grant, and Gabriel Robles-De-La-Torre. Haptic interfaces and devices. *Sensor Review*, 24(1):16–29, 2004.

[5] Richard J Adams, Manuel R Moreyra, and Blake Hannaford. Stability and performance of haptic displays: Theory and experiments. In *Proceedings ASME international mechanical engineering congress and exhibition*, pages 227–234, 1998.

[6] Dale A Lawrence. Stability and transparency in bilateral teleoperation. *IEEE transactions on robotics and automation*, 9(5):624–637, 1993.

[7] Septimiu E Salcudean, Ming Zhu, Wen-Hong Zhu, and Keyvan Hashtrudi-Zaad. Transparent bilateral teleoperation under position and rate control. *The International Journal of Robotics Research*, 19(12):1185–1202, 2000.

[8] Göran Anders Viking Christiansson. *Hard master, soft slave haptic teleoperation*. PhD thesis, TU Delft, Delft University of Technology, 2007.

[9] Nima Enayati, Elena De Momi, and Giancarlo Ferrigno. Haptics in robot-assisted surgery: challenges and benefits. *IEEE reviews in biomedical engineering*, 9:49–65, 2016.

[10] Yamamoto A. Asada T., Nakamura T. Investigation on substitution of force feedback using pressure stimulation to palm. In *Haptics Symposium*, 2016.

[11] Taku Nakamura, Shota Nemoto, Takumi Ito, and Akio Yamamoto. Substituted force feedback using palm pressurization for a handheld controller. In *International AsiaHaptics conference*, pages 197–199. Springer, 2016.

[12] Panagiotis Agrafiotis, Anastasios Doulamis, George Athanasiou, and Angelos Amditis. Real time earthquake's survivor detection using a miniaturized lwir camera. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, page 21. ACM, 2016.

[13] Mohamed Elgendi, Flavien Picon, Nadia Magnenat-Thalmann, and Derek Abbott. Arm movement speed assessment via a kinect camera: a preliminary study in healthy subjects. *Biomedical engineering online*, 13(1):88, 2014.

[14] Linda van den Bedem, Ron Hendrix, Nick Rosielle, Maarten Steinbuch, and Henk Nijmeijer. Design of a minimally invasive surgical teleoperated master-slave system with haptic feedback. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pages 60–65. IEEE, 2009.

[15] Mahdi Tavakoli, RV PateI, and Mehrdad Moallem. Design issues in a haptics-based master-slave system for minimally invasive surgery. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 371–376. IEEE, 2004.

[16] Ilana Nisky, Assaf Pressman, Carla M Pugh, Ferdinando A Mussa-Ivaldi, and Amir Karniel. Perception and action in teleoperated needle insertion. *IEEE transactions on haptics*, 4(3):155–166, 2011.

[17] Carlos Bermejo and Pan Hui. A survey on haptic technologies for mobile augmented reality. *arXiv preprint arXiv:1709.00698*, 2017.

[18] Shoichi Hasegawa, Ishikawa Toshiaki, Naoki Hashimoto, Marc Salvati, Hironori Mitake, Yasuharu Koike, and Makoto Sato. Human-scale haptic interaction with a reactive virtual human in a real-time physics simulator. *Computers in Entertainment (CIE)*, 4(3):9, 2006.

[19] Winfred Mugge, Irene A Kuling, Eli Brenner, and Jeroen BJ Smeets. Haptic guidance needs to be intuitive not just informative to improve human motor accuracy. *PloS one*, 11(3):e0150912, 2016.

[20] Domenico Prattichizzo, Claudio Pacchierotti, and Giulio Rosati. Cutaneous force feedback as a sensory subtraction technique in haptics. *IEEE Transactions on Haptics*, 5(4):289–300, 2012.

[21] Ryan E Schoonmaker and Caroline GL Cao. Vibrotactile force feedback system for minimally invasive surgical procedures. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 3, pages 2464–2469. IEEE, 2006.

[22] Arash Ajoudani, Sasha B Godfrey, Matteo Bianchi, Manuel G Catalano, Giorgio Grioli, Nikos Tsagarakis, and Antonio Bicchi. Exploring teleimpedance and tactile feedback for intuitive control of the pisa/iit softhand. *IEEE transactions on haptics*, 7(2):203–215, 2014.

[23] Jacques Foottit, Dave Brown, Stefan Marks, and Andy M Connor. An intuitive tangible game controller. In *Proceedings of the 2014 Conference on Interactive Entertainment*, pages 1–7. ACM, 2014.

[24] Kouta Minamizawa, Souichiro Fukamachi, Hiroyuki Kajimoto, Naoki Kawakami, and Susumu Tachi. Gravity grabber: wearable haptic display to present virtual mass sensation. In *ACM SIGGRAPH 2007 emerging technologies*, page 8. ACM, 2007.

[25] Gill A Pratt and Matthew M Williamson. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995.

[26] Jerry E Pratt and Benjamin T Krupp. Series elastic actuators for legged robots. In *Unmanned Ground Vehicle Technology Vi*, volume 5422, pages 135–145. International Society for Optics and Photonics, 2004.

[27] Arnaldo Gomes Leal Junior, Rafhael Milanezi de Andrade, and Antônio Bento Filho. Series elastic actuator: Design, analysis and comparison. In *Recent Advances in Robotic Systems*. InTech, 2016.

[28] Wilian M dos Santos and Adriano AG Siqueira. Impedance control of a rotary series elastic actuator for knee rehabilitation. *IFAC Proceedings Volumes*, 47(3):4801–4806, 2014.

[29] E Basafa, M Sheikholeslami, A Mirbagheri, F Farahmand, and GR Vossoughi. Design and implementation of series elastic actuators for a haptic laparoscopic device. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 6054–6057. IEEE, 2009.

[30] Jakob Oblak and Zlatko Matjačić. Design of a series visco-elastic actuator for multi-purpose rehabilitation haptic device. *Journal of neuroengineering and rehabilitation*, 8(1):3, 2011.

[31] Ata Otaran, Ozan Tokatli, and Volkan Patoglu. Hands-on learning with a series elastic educational robot. In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*, pages 3–16. Springer, 2016.

[32] Katherine J Kuchenbecker, June Gyu Park, and Gunter Niemeyer. Characterizing the human wrist for improved haptic interaction. In *ASME 2003 International Mechanical Engineering Congress and Exposition*, pages 591–598. American Society of Mechanical Engineers, 2003.

[33] Jaebum Park, Nemanja Pažin, Jason Friedman, Vladimir M Zatsiorsky, and Mark L Latash. Mechanical properties of the human hand digits: Age-related differences. *Clinical Biomechanics*, 29(2):129–137, 2014.

[34] John E Speich, Liang Shao, and Michael Goldfarb. Modeling the human hand as it interacts with a telemanipulation system. *Mechatronics*, 15(9):1127–1142, 2005.

[35] Ken Dutton, Steve Thompson, and Bill Barraclough. *The art of control engineering*. Addison Wesley Harlow, 1997.

[36] Ben Fry and Casey Reas. Processing. https://www.processing.org/, July 2018.

[37] exnumerus. How to fit a sinewave, an example in python. http://exnumerus.blogspot.com/2010/04/how-to-fit-sine-wave-example-in-python.html, April 2010.

[38] Christopher C Ibeh. *Thermoplastic materials: properties, manufacturing methods, and applications*. CRC Press, 2011.

11 Appendices

Arduino Code - Both Controllers

```
/*
  Roman Oechslin
  Robot controller with haptic feedback
  2018/08/07  Tokyo
  Version 1.0
*/
#define FASTADC 1

// defines for setting and clearing register bits
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

#define joystick_left_pin A0
#define photo_left_pin A2
#define joystick_right_pin A1
#define photo_right_pin A3
int motorPinLeft = 3;     // Motor connected to digital pin 3 (PWM)
int motorPinRight = 11;     // Motor connected to digital pin 11 (PWM)

float dist_ref = 0.0;
float FILTER_CST = 0.5; // filter when reading new feedback value
int OUTPUT_PER_VOLT = 255/5; // the arduino can output 5V max
int MOTOR_MAX_VOLTAGE = 20; // change this according to motor
int LEFT_HAND_AMPLIFIER_GAIN = 10;
int RIGHT_HAND_AMPLIFIER_GAIN = 10;
const int OUTPUT_RANGE_L = 2 * MOTOR_MAX_VOLTAGE * OUTPUT_PER_VOLT /
    LEFT_HAND_AMPLIFIER_GAIN;
const int OUTPUT_RANGE_R = 2 * MOTOR_MAX_VOLTAGE * OUTPUT_PER_VOLT /
    RIGHT_HAND_AMPLIFIER_GAIN;

float error_left = 0.0;
float error_left_last = 0.0;
float sum_error_left = 0.0;
float numerator_l_last_f = 0.0;
float error_right = 0.0;
float error_right_last = 0.0;
float sum_error_right = 0.0;
float numerator_r_last_f = 0.0;
float dT = 0.001;
float filter_coeff = 0.05; // filter on derivative part

int photo_value_left_raw = 0;
int photo_value_right_raw = 0;

const int PHOTO_MIN_LEFT = 640;
const int PHOTO_MAX_LEFT = 840;
const int PHOTO_MIN_RIGHT = 700;
const int PHOTO_MAX_RIGHT = 880;
const float MAX_DISPLACEMENT_MM = 1.8; // [mm], has been measured

// If Yoke controller is used:
/*
const int PHOTO_MIN_LEFT = 550;
const int PHOTO_MAX_LEFT = 780;
const int PHOTO_MIN_RIGHT = 600;
const int PHOTO_MAX_RIGHT = 763;
const float MAX_DISPLACEMENT_MM = 5.0; // [mm], has been measured
```

```
*/

unsigned long TIME_BEGIN = 0;
unsigned long TIME_NOW = 0;
unsigned long TIME_CYCLE = 1000; // this is the time_step in [microseconds] that
    determines the running frequency


void setup() {
  Serial.begin(250000);
  pinMode(motorPinLeft, OUTPUT);
  pinMode(motorPinRight, OUTPUT);

  pinMode(joystick_left_pin, INPUT);
  pinMode(photo_left_pin, INPUT);
  pinMode(joystick_right_pin, INPUT);
  pinMode(photo_right_pin, INPUT);
  pinMode(testPin, OUTPUT);
  pinMode(controlPin, OUTPUT);

  pinMode(LED_BUILTIN, OUTPUT);

  TCCR2B = TCCR2B & B11111000 | B00000001; // for PWM frequency of 31372.55 Hz
  #if FASTADC
  // set prescale to 16
  sbi(ADCSRA, ADPS2) ;
  cbi(ADCSRA, ADPS1) ;
  cbi(ADCSRA, ADPS0) ;
  #endif
}

void loop() {
  TIME_BEGIN = micros();

  int joy_val_left = analogRead(joystick_left_pin);
  joy_val_left = joy_val_left>>2; // the joystick value is [0..1023] converts it to 8
    bit
  int joy_val_right = analogRead(joystick_right_pin);
  joy_val_right = joy_val_right>>2; // the joystick value is [0..1023] converts it to
     8bit

  photo_value_left_raw = analogRead(photo_left_pin);
  photo_value_right_raw = analogRead(photo_right_pin);

  int pwmValueLeftSym = 128;
  int pwmValueRightSym = 128;

//  float k_p = 0.243;
//  float k_i = 0.63;
//  float k_d = 0.00126;
  float k_p = 0.3;
  float k_i = 0.0;
  float k_d = 0.0;

  // If Yoke controller is used:
  /*
   float k_p = 5.0; // [V/mm]
   float k_i = 0.0;
   float k_d = 0.0;

   k_p = k_p * 255 / 40; // to convert it to arduino values
   */

  // Writing to Processing file
  Serial.write(joy_val_left);
  Serial.write(joy_val_right);
```

```
123   Serial.write((joy_val_left + joy_val_right)%256); // this is the checksum
124   if (Serial.available() > 0) {
125     dist_ref = (int)(FILTER_CST*dist_ref + (1-FILTER_CST)* Serial.read()*(
        MAX_DISPLACEMENT_MM) / 255);
126   } else {
127     dist_ref = FILTER_CST*dist_ref;
128   }
129   error_left = dist_ref - sensor2dist(photo_value_left_raw, PHOTO_MIN_LEFT,
        PHOTO_MAX_LEFT);
130   error_right = dist_ref - sensor2dist(photo_value_right_raw, PHOTO_MIN_RIGHT,
        PHOTO_MAX_RIGHT);
131
132   sum_error_left = error_left + sum_error_left;
133   sum_error_right = error_right + sum_error_right;
134   float numerator_l = error_left - error_left_last;
135   float numerator_r = error_right - error_right_last;
136   float numerator_l_f = numerator_l*filter_coeff + numerator_l_last_f * (1-
        filter_coeff);
137   float numerator_r_f = numerator_r*filter_coeff + numerator_r_last_f * (1-
        filter_coeff);
138
139   numerator_l_last_f = numerator_l_f;
140   numerator_r_last_f = numerator_r_f;
141   error_left_last = error_left;
142   error_right_last = error_right;
143
144   int des_mot_volt_left = k_p * error_left + k_i * sum_error_left*dT + k_d *
        numerator_l_f/dT;
145   int des_mot_volt_right =  k_p * error_right + k_i * sum_error_right*dT + k_d *
        numerator_r_f/dT;
146
147   // symmetric feedback left and right, (allowing for negative motor values)
148   pwmValueLeftSym = pid2pwm_sym(des_mot_volt_left, OUTPUT_RANGE_L);
149   pwmValueRightSym = pid2pwm_sym(des_mot_volt_right, OUTPUT_RANGE_R);
150
151   analogWrite(motorPinLeft, pwmValueLeftSym);
152   analogWrite(motorPinRight, pwmValueRightSym);
153
154   while ((micros() - TIME_BEGIN) < TIME_CYCLE) {  } // do nothing until we reach the
        time step of TIME_CYCLE
155 }
156
157 float sensor2dist(int sensor_value, int min_val, int max_val) {
158   // maps the measured value to the distance (assumed linearity) in millimeters
159   return MAX_DISPLACEMENT_MM - (sensor_value - min_val) * (MAX_DISPLACEMENT_MM ) / (
        max_val - min_val) ;
160 }
161
162
163 int limit_value(int value, int lower, int upper) {
164   if (value < lower) {
165     value = lower;
166   } else if (value > upper) {
167     value = upper;
168   }
169   return value;
170 }
171
172 int pid2pwm_sym(int des_voltage, int range) {
173   // converts motor voltage to pwm output where 128-range/2 is full power in one
        direction and 128+range/2 is
174   // full power in the other
175   return limit_value(des_voltage, -range/2, range/2) + 128;
176 }
```

Processing Code

```
/*
//  Controller for Topy's Crawler Robot
 //  Author: Roman Oechslin
 //  Master project - haptic feedback controller - Yamamoto's lab, Uni Tokyo
 //
 //  Date 2018/08/07
 //  Version 1.03
*/

import processing.serial.*;
Serial crawlerPort;
Serial arduinoPort;

PFont warningFont;

boolean DEBUG = true; // some cryptic terminal output with a lot of numbers
boolean VERBOSE = false; // verbose output in terminal
boolean IGNORE_COM = false; // if no port available

int msg_stop[] = {0xFF, 0xEE, 0x00, 0x00, 0x00, 0x00, 0x01, 0x4E, 0x01, 0x6E, 0x01,
                  0x2D, 0x32, 0x32, 0x02, 0x17, 0x02, 0x17, 0x37, 0x00, 0x00, 0x00,
                  0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x12, 0x11,
                  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x1E, 0xED, 0x0D};
int msg_forward[] = {0xFF, 0xEE, 0x00, 0x46, 0x00, 0x46, 0x01, 0x4E, 0x01, 0x6E, 0x01
    ,
                    0x2D, 0x32, 0x32, 0x02, 0x17, 0x02, 0x17, 0x37, 0x00, 0x00, 0x00,
                    0x07, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x02, 0x12, 0x11,
                    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x1E, 0x83, 0x0D
    };
int my_msg[] = msg_stop;
int driving_speed_left = 0x46; // default driving speed of 70%
int driving_speed_right = 0x46; // default driving speed of 70%
boolean time_init = false;
int program_init_time[] = {hour(), minute(), second()}; // to know how long the
    program has been running
int offset_sec = 0; // offset between computer program and robot

int pitch_angle = 0;
int roll_angle = 0;
final int MAX_ANGLE = 25; // maximum angle for some feedback laws (25 is appropriate)
final int DEADZONE_CST = 27; // deadzone for joystick values (27 is appropriate)
final int ONE_BYTE_LENGTH = 256;
int ARDU_MSG_LENGTH = 3; // number of bytes received from arduino
int RX_ROBOT_MSG_LENGTH = 70; // number of bytes in msg from robot to pc

int crawler_current_left = 0;
int crawler_current_right = 0;
int current_offset[] = {529, 550}; // offset values can vary over time (529, 550 is
    suggested)

final int STOP_MODE = 0; // both stop
final int FORWARD_MODE = 1; // both forward
final int BACKWARD_MODE = 2; // both backward
final int BL_FR = 3; // backward left forward right
final int FL_BR = 4; // forward left backward right
int driving_mode = STOP_MODE;
int feedback_law = 0; // 0 -> pitch + roll, 1 -> pitch*roll, 2 -> current
int FEEDBACK_MAX = 3; // number of possible different feedback laws
int SAFETY_FACTOR = 1; // factor by which final motor command is diveded by 2 (1 is
    appropriate)
int ARDU_MSG_SHIFT = 2; // allows to shift msg by 2

int global_loop_counter = 0;
boolean critical_battery_level = false; // is true when voltage drops below 11.5 V
```

```
60
61 void setup() {
62   // setup loop
63   if (!IGNORE_COM) arduinoPort = new Serial(this, "COM3", 250000);
64   if (!IGNORE_COM) crawlerPort = new Serial(this, "COM4", 250000);
65   size(500, 500);
66   frameRate(50); // 50 is appropriate
67   println("Started up program!");
68   if (IGNORE_COM) println("ignoring com ports");
69   if (DEBUG) println("DEBUG MODE ON");
70   else println("DEBUG MODE OFF");
71   if (VERBOSE) println("VERBOSE MODE ON");
72   else println("VERBOSE MODE OFF");
73
74   warningFont = createFont("Arial", 16, true);
75
76   fill(0);
77   text("Left feedback", width/4-width/20, height/6-height/45);
78   text("Right feedback", 3*width/4-width/20, height/6-height/45);
79   text("Driving direction", width/2-width/20, height/20-height/45);
80   text("Feedback mode\nPress mouse button\n to change.", 9*width/20, 9*height/20-3*
       height/45);
81   text("Battery level", 9*width/20, 15*height/20-height/45);
82 }
83
84 void draw() {
85   // main loop
86   draw_direction_rectangle();
87   if (critical_battery_level) {
88     warn_battery();
89     send_over_serial(msg_stop);
90   } else {
91     if (global_loop_counter == 5) { // 5 is appropriate
92       global_loop_counter = 0;
93       construct_msg();
94       send_over_serial(my_msg);
95       receive_robot_msg();
96     }
97     receive_arduino_msg();
98     draw_feedback_rectangles();
99
100    send_arduino_feedback();
101    global_loop_counter +=1 ;
102  }
103 }
104
105
106 void mouseClicked() {
107   // change feedback_law by mouse_click
108   feedback_law = (feedback_law + 1) %FEEDBACK_MAX;
109 }
110
111 void draw_feedback_rectangles() {
112   // visualize feedback on GUI with different feedback_law
113   float feedback = get_feedback(roll_angle, pitch_angle, crawler_current_left,
       crawler_current_right);
114   float bar_height = height*0.4;
115   switch (feedback_law) {
116   case 0:
117     // left hand feedback
118     fill(145);
119     rect(width/4-width/20, height/6, width/10, height*0.4);
120     fill(0, 0, 255);
121     rect(width/4-width/20, height/6+(1-feedback)*bar_height, width/10, bar_height*
       feedback);
122
```

```
123     // right hand feedback
124     fill(145);
125     rect(3*width/4-width/20, height/6, width/10, height*0.4);
126     fill(0, 0, 255);
127     rect(3*width/4-width/20, height/6+(1-feedback)*bar_height, width/10, bar_height*
        feedback);
128
129     // feedback type indication
130     fill(255, 0, 0);
131     rect(9*width/20, 9*height/20, width/10, height/10);
132     break;
133   case 1:
134     // left hand feedback
135     fill(145);
136     rect(width/4-width/20, height/6, width/10, height*0.4);
137     fill(0, 0, 255, 255*feedback);
138     rect(width/4-width/20, height/6, width/10, height*0.4);
139
140     // right hand feedback
141     fill(145);
142     rect(3*width/4-width/20, height/6, width/10, height*0.4);
143     fill(0, 0, 255, 255*feedback);
144     rect(3*width/4-width/20, height/6, width/10, height*0.4);
145
146     // feedback type indication
147     fill(0, 255, 0);
148     rect(9*width/20, 9*height/20, width/10, height/10);
149     break;
150   case 2:
151     // left hand feedback
152     fill(145);
153     rect(width/4-width/20, height/6, width/10, height*0.4);
154     fill(0, 255, 0);
155     rect(width/4-width/20, height/6+(1-feedback)*bar_height, width/10, bar_height*
        feedback);
156
157     // right hand feedback
158     fill(145);
159     rect(3*width/4-width/20, height/6, width/10, height*0.4);
160     fill(0, 255, 0);
161     rect(3*width/4-width/20, height/6+(1-feedback)*bar_height, width/10, bar_height*
        feedback);
162
163     // feedback type indication
164     fill(0, 0, 255);
165     rect(9*width/20, 9*height/20, width/10, height/10);
166     break;
167   }
168 }
169
170
171 float get_feedback(int roll, int pitch, int crawler_current_left, int
        crawler_current_right) {
172   // sends back a feedback between 0 and 1
173   // uses different input or functions depending on feedback_law
174   float feedback = 0.0;
175   if (driving_mode == FORWARD_MODE || driving_mode == STOP_MODE) {
176     float K_P = 0.5/sin(MAX_ANGLE*TWO_PI/360); // proportional gain
177     float K_P1 = 1/(sin(MAX_ANGLE*TWO_PI/360)*sin(MAX_ANGLE*TWO_PI/360)); //
        proportional gain
178     float K_P2_inv = 4000; // 4000 seems to be a good unification factor, since
        AmpMax = 2.0A
179
180     roll = abs(roll);
181     pitch = abs(pitch);
182     if (roll > MAX_ANGLE) roll = MAX_ANGLE;
```

```
183     if (pitch > MAX_ANGLE) pitch = MAX_ANGLE;
184
185     switch (feedback_law) {
186     case 0:
187       feedback = (float) K_P*(sin(roll*TWO_PI/360) + sin(pitch*TWO_PI/360));
188       break;
189     case 1:
190       if (roll > MAX_ANGLE && pitch > MAX_ANGLE) {
191         // obsolete due to limiting to MAX_ANGLE, but better safe than sorry
192         feedback = (float) K_P1*(sin(MAX_ANGLE*TWO_PI/360) * sin(MAX_ANGLE*TWO_PI
    /360));
193       } else if (pitch > MAX_ANGLE) {
194         feedback = (float) K_P1*(sin(roll*TWO_PI/360) * sin(MAX_ANGLE*TWO_PI/360));
195       } else if (roll > MAX_ANGLE) {
196         feedback = (float) K_P1*(sin(MAX_ANGLE*TWO_PI/360) * sin(pitch*TWO_PI/360));
197       } else {
198         feedback = (float) K_P1*(sin(roll*TWO_PI/360) * sin(pitch*TWO_PI/360));
199       }
200       break;
201     case 2:
202       feedback = (float) (crawler_current_left + crawler_current_right) / K_P2_inv;
203       break;
204     default:
205       feedback = 0.0;
206       break;
207     }
208   } else {
209     feedback = 0.0;
210   }
211   if (feedback > 1) feedback = 1.0;
212   return feedback;
213 }
214
215
216 void receive_arduino_msg() {
217   // read the message from the arduino over serial port
218   int joystick_value_left = 127;
219   int joystick_value_right = 127;
220   int received_msg[] = new int[ARDU_MSG_LENGTH + ARDU_MSG_SHIFT];
221   int counter = 0;
222
223   while (!IGNORE_COM && arduinoPort.available() > 0 && counter < ARDU_MSG_LENGTH +
      ARDU_MSG_SHIFT) {
224     received_msg[counter] = arduinoPort.read();
225     counter++;
226   }
227
228   if (counter == ARDU_MSG_LENGTH + ARDU_MSG_SHIFT) {
229     int shift = 0;
230     while ((received_msg[0+shift] + received_msg[1+shift])%256 !=
231         received_msg[2+shift] && shift < ARDU_MSG_SHIFT) {
232       shift += 1;
233     }
234     if (shift < ARDU_MSG_SHIFT) {
235       joystick_value_left = received_msg[shift];
236       joystick_value_right = received_msg[1 + shift];
237     } else { // if checksum does not check
238       if (DEBUG) println("Checksum is wrong");
239       return;
240     }
241   } else {
242     return;
243   }
244
245   if (!IGNORE_COM) arduinoPort.clear();
246
```

```
247   // shift joystick values by 127 (half a byte) to make it symmetric around 0
248   joystick_value_left = joystick_value_left - 127;
249   joystick_value_right = joystick_value_right - 127;
250
251   // check for left joystick
252   if (abs(joystick_value_left) > DEADZONE_CST) {
253     if (joystick_value_left > DEADZONE_CST) { // driving forwards
254       if (VERBOSE) println("left going forward");
255       driving_mode = FORWARD_MODE;
256       driving_speed_left = joystick_value_left - DEADZONE_CST;
257       // joystick_value_left is bounded between (27..127)
258     } else if (-joystick_value_left > DEADZONE_CST) { // driving backwards
259       if (VERBOSE) println("left going backwards");
260       driving_mode = BACKWARD_MODE;
261       driving_speed_left = abs(joystick_value_left + DEADZONE_CST);
262       // joystick_value_left is bounded between (-127..-27)
263     }
264   } else {
265     if (VERBOSE) println("left stopping");
266     driving_mode = STOP_MODE;
267     driving_speed_left = 0;
268   }
269
270   // check for right joystick
271   if (abs(joystick_value_right) > DEADZONE_CST) {
272     if (joystick_value_right > DEADZONE_CST) { // driving forward
273       if (VERBOSE) println("right going forward");
274       if (driving_mode == BACKWARD_MODE) {
275         driving_mode = BL_FR;
276       } else {
277         driving_mode = FORWARD_MODE; // NOTE if this is dropped, one can forbid one-
      sided turns
278       }
279       driving_speed_right = joystick_value_right - DEADZONE_CST;
280       // joystick_value_right is bounded between (27..127)
281     } else if (-joystick_value_right > DEADZONE_CST) { // driving backwards
282       if (VERBOSE) println("right going backwards");
283       if (driving_mode == FORWARD_MODE) {
284         driving_mode = FL_BR;
285       } else {
286         driving_mode = BACKWARD_MODE; // FIXME if i drop this i can forbid one sided
      turns
287       }
288       driving_speed_right = abs(joystick_value_right + DEADZONE_CST);
289       // joystick_value_right is bounded between (-127..-27)
290     }
291   } else {
292     if (VERBOSE) println("right stopping");
293     driving_speed_right = 0;
294   }
295   // Safety factor division to drive the robot slower than full speed
296   driving_speed_left = driving_speed_left >> SAFETY_FACTOR;
297   driving_speed_right = driving_speed_right >> SAFETY_FACTOR;
298   if (DEBUG) println("driving speed = " + driving_speed_left + " and " +
      driving_speed_right);
299 }
300
301
302 void send_arduino_feedback() {
303   // write the feedback according to the specified feedback_law to the arduino serial
       port
304   float feedback = get_feedback(roll_angle, pitch_angle, crawler_current_left,
      crawler_current_right);
305   int arduino_feedback = (int) (feedback*255);//(feedback*1800);//FIXME this should
      be a constant
306   if (!IGNORE_COM) arduinoPort.write(arduino_feedback);
```

```
307   if (DEBUG) println("send feedback to arduino: " + arduino_feedback);
308 }
309
310 void receive_robot_msg() {
311   // read the message received from the robot over the serial port
312   // if first message: assign initial offsets
313   int received_msg[] = {};
314   if (VERBOSE) println("\nread: (in receive robot msg)");
315   int msg_length = 0;
316   if (!IGNORE_COM) {
317     while (crawlerPort.available() > 0) {
318       msg_length += 1;
319       int inByte = crawlerPort.read();
320       received_msg = append(received_msg, inByte);
321       if (VERBOSE) print(inByte + ' ');
322     }
323   }//end IGNORE_COM
324   if (VERBOSE) {
325     print("\n");
326     if (msg_length >= RX_ROBOT_MSG_LENGTH) {
327       if (VERBOSE) println("elapsed time is: " + received_msg[67]);
328       //67 is the index of elapse time, beware of second byte in front!!
329     }
330   }
331
332   if (!time_init && msg_length >= RX_ROBOT_MSG_LENGTH) {
333     time_init = true;
334     offset_sec = received_msg[66]*ONE_BYTE_LENGTH + received_msg[67] -
335         ((hour() - program_init_time[0])*60+ minute() - program_init_time[1])*60 +
336         second() - program_init_time[2];
337     if (VERBOSE) println("I initialized the time with " + offset_sec);
338   }
339
340   if (msg_length >= RX_ROBOT_MSG_LENGTH) {
341     roll_angle = convert_angles(received_msg[23], received_msg[24]);
342     pitch_angle = convert_angles(received_msg[21], received_msg[22]);
343     // read and convert measured crawler currents
344     crawler_current_left = abs(convert16bit(received_msg[16], received_msg[17]) -
345         (msg_forward[16]*256) - msg_forward[17]);
346     crawler_current_left *= 16000/1024;
347     crawler_current_right = abs(convert16bit(received_msg[12], received_msg[13]) -
348         (msg_forward[16]*256) - msg_forward[17]);
349     crawler_current_right *= 16000/1024;
350     if (VERBOSE) println("crawler current = " + (crawler_current_right +
      crawler_current_left));
351     check_battery_status(received_msg[36]);
352   }
353
354 }
355
356 int convert_angles(int high_byte_val, int low_byte_val) {
357   // This function converts a 16bit value for the angles to an integer
358   int angle = 0;
359
360   if (high_byte_val > 127) {
361     angle = -(255 - high_byte_val) << 8;
362   } else {
363     angle = high_byte_val << 8;
364   }
365   if (low_byte_val > 127) {
366     angle = angle -(255 - low_byte_val);
367   } else {
368     angle = angle + low_byte_val;
369   }
370   return angle;
371 }
```

```
372
373 int convert16bit(int high_byte_val, int low_byte_val) {
374   // This function converts a 16bit value to an integer
375   int value = 0;
376   value =  (high_byte_val<<8) + low_byte_val;
377   if (VERBOSE) println("value = " + value + " high = " + high_byte_val + " low = " +
      low_byte_val);
378   return value;
379 }
380
381 void check_battery_status(int bat_msg) {
382   float battery_level = (float) bat_msg *30/255; // values from robot manual
383   if (VERBOSE) println("battery status = " + battery_level);
384
385   if (battery_level > 13.4) { // everything is fine
386     critical_battery_level = false;
387     fill(0, 255, 0);
388   } else if (battery_level > 12.5) { // mid range
389     critical_battery_level = false;
390     fill(255, 255, 0);
391   } else if (battery_level > 12.1) { // critical, warn
392     critical_battery_level = false;
393     fill(255, 100, 0);
394   } else if (battery_level > 11.5) { // critical, stop
395     critical_battery_level = true;
396     fill(255, 0, 0);
397   } else if (battery_level > 0.01) { // very critical, stop
398     print("battery so critically low: "+battery_level);
399     critical_battery_level = true;
400     fill(255, 0, 0);
401   } else {
402     println("battery problem: battery level at: " + battery_level + "V!");
403   }
404   // battery charge indication
405   rect(8*width/20, 15*height/20, width/5, height/10);
406   fill(0);
407   text(floor(battery_level) + "." + (int) ((battery_level - (float) floor(
      battery_level))*10),
408     9*width/20, 16.5*height/20);
409 }
410
411 void draw_direction_rectangle() {
412   // indicates in which direction the robot is commanded
413   // (green = forward, red = backward, grey = no command)
414   fill(200);
415   rect(width/2-width/20, height/20-height/60, width/3, height/14);
416   textFont(warningFont, 32);
417   switch (driving_mode) {
418   case STOP_MODE:
419     fill(0, 0, 0);
420     text("halt", width/2-width/20, height/10);
421     break;
422   case FORWARD_MODE:
423     fill(0, 255, 0);
424     text("forward", width/2-width/20, height/10);
425     break;
426   case BACKWARD_MODE:
427     fill(255, 0, 0);
428     text("backward", width/2-width/20, height/10);
429     break;
430   default:
431     fill(0, 0, 0);
432     break;
433   }
434 }
435
```

```
436 void construct_msg() {
437   // assigns the driving mode and driving speed for the two crawlers
438   switch (driving_mode) {
439   case STOP_MODE:
440     my_msg = msg_stop;
441     if (VERBOSE) println("stop mode\n");
442     break;
443   case FORWARD_MODE:
444     my_msg = msg_forward;
445     my_msg[2] = 0;
446     my_msg[4] = 0;
447     my_msg[3] = driving_speed_right;
448     my_msg[5] = driving_speed_left;
449     if (VERBOSE) println("forward mode\n");
450     break;
451   case BACKWARD_MODE:
452     my_msg = msg_forward;
453     my_msg[2] = 255;
454     my_msg[4] = 255;
455     my_msg[3] = 255-driving_speed_right;
456     my_msg[5] = 255-driving_speed_left;
457     if (VERBOSE) println("backward mode\n");
458     break;
459     //FIXME complete this list
460   case FL_BR:
461     my_msg = msg_forward;
462     my_msg[2] = 255;
463     my_msg[3] = 255-driving_speed_right;
464     my_msg[4] = 0;
465     my_msg[5] = driving_speed_left;
466     if (VERBOSE) println("FL BR mode\n");
467     break;
468   case BL_FR:
469     my_msg = msg_forward;
470     my_msg[2] = 0;
471     my_msg[3] = driving_speed_right;
472     my_msg[4] = 255;
473     my_msg[5] = 255-driving_speed_left;
474     if (VERBOSE) println("BL FR mode\n");
475     break;
476   default:
477     my_msg = msg_stop;
478     if (VERBOSE) println("Error, no known driving mode\n");
479     break;
480   }
481 
482   if (time_init) {
483     int passed_time = ((hour() - program_init_time[0])*60 + minute() -
484       program_init_time[1])*60 + second() - program_init_time[2] + offset_sec;
485     if (passed_time > ONE_BYTE_LENGTH-1) {
486       my_msg[40] = passed_time/ONE_BYTE_LENGTH;
487     }
488     my_msg[41] = passed_time % ONE_BYTE_LENGTH;
489     if (VERBOSE) println("\nupdated time to " + passed_time);
490   }
491 }
492 
493 void send_over_serial(int my_msg[]) {
494   // calculates the check_sum and sends message over serial port to robot
495   int checksum = 0;
496   if (VERBOSE) println("sent message is:\n");
497   for (int i=0; i<=43; i++) {
498     if (i<42) {
499       checksum = checksum + my_msg[i];
500     } else if (i == 42) {
501       my_msg[i] = checksum%ONE_BYTE_LENGTH;
```

```
502     }
503     if (!IGNORE_COM) crawlerPort.write(my_msg[i]);
504     if (VERBOSE) print(my_msg[i] + " ");
505   }
506   if (VERBOSE) println();
507 }
508
509 void warn_battery() {
510   fill(100);
511   rect(width/20, height/20, 9*width/10, 9*height/10);
512   textFont(warningFont, 40);
513   fill(255, 0, 0);
514   text("Battery charge critical!\nCharge battery!", width/16, height/2);
515 }
```

Matlab Code - Bode Comparison

```
spring_damping_vec = [100 150 200 250 300 350 400];
%k_op = 200
%b_op = 20
K_P = 0.2
K_I = 0.0
K_D = 0.0

K_PID2Vardu = 1/51 % [V]
um_gain = -1000000 % [um/m]
K_ampl = 10 % V/V
n = 112 % reduction ration [-]
J_T = 1.25E-3 / n / n % kg*m^2
L_CL = 20E-3
k_eq = 6000 % N/m
m_2 = 7E-3 % kg
L_a = 5.8E-3 % H
R_a  = 118.6 % Ohm
efficiency = 0.6 % [-]
K_tau = efficiency*31.4E-3 % [Nm/A]
K_emf = 3.29E-3 / 60 % [V/s]
s = tf('s');
%z = tf('z',dT) % Used for discretization
%s = (1-z^-1)/dT % Used for discretization
dT = 0.001;

%experim K_P = 0.2; K_I = 0; K_D = 0
freq_vec = [1.25, 1.6, 100.0, 10.0, 13.0, 16.0, 1.0, 2.5, 20.0, 25.0,
    2.0, 32.0, 3.0, 40.0, 4.0, 50.0, 5.0, 6.3, 63.0, 80.0, 8.0]
amplitude_factor_l = [0.8021762720531634, 0.8054151390569485,
    0.015545101680621341, 1.0773667825505355, 1.1642926077392948,
    1.1103451508176316, 0.8001758873970981, 0.814056941660425,
    0.8809965954959127, 0.5443027780892209, 0.8080825031462602,
    0.2796958251080355, 0.8204548508774807, 0.16228564296355955,
    0.8471466599918783, 0.10547366728162677, 0.8705112782697394,
    0.9245682579252669, 0.05343340955353186, 0.024603192099363968,
    0.9978075966054786]
phasediff_l = [-13.338637077482488, -14.09045739088748,
    -215.85673669598538, -48.04116200227554, -73.34441078067667,
    -100.25217048165923, -13.120541509581287, -16.085908849961044,
    -133.37888476020055, 201.4902985359118, -14.874321828402891,
    -176.09087916984802, 342.54636240739, 172.0658999860665,
    339.59240015589626, 160.15771192477212, -23.728405952504453,
    -28.567395522204393, 147.9917576755808, 149.19385839692242,
    -36.037447598503036]
amplitude_factor_r = [0.7259689909042475, 0.735561475713898,
    0.011847747061743118, 0.7807616430080885, 0.6652857160264002,
    0.4684605400670259, 0.7127722160221738, 0.7461571487258002,
    0.28422778003510263, 0.18648071556469148, 0.7439708501454618,
    0.10739962887725536, 0.7525359007020068, 0.07014225593098215,
    0.7786568428021221, 0.04585663312767849, 0.8100457909516678,
```

```
    0.8245672101098731, 0.026406209520878O5, 0.017672001900720043,
    0.8172666256439353]
phasediff_r = [-16.647751016192103, -18.17293749217521,
    -237.48339557890603, -84.206485995129, -110.7789491948499,
    -133.2352471671927, -15.490775194720683, -22.931547230698484,
    -154.12969771290278, 194.1127967504471, -20.088130391155154,
    -178.43503982578886, 334.0056876428433, 172.21922748081028,
    327.9483761537956, -190.87105537800483, -39.46722121664876,
    -49.247314509247005, 153.90271170189118, 138.0318425466486,
    -64.3717766560425]


phasediff_l = -(phasediff_l>0)*360 + phasediff_l;
phasediff_r = -(phasediff_r>0)*360 + phasediff_r;
[f_sort, I] = sort(freq_vec)

for k = 1:length(spring_damping_vec)
    b_sp = spring_damping_vec(k)

    % blocked by walls
    a = um_gain*(K_P + K_D*s + K_I/s);
    b = K_PID2Vardu*K_ampl;
    c = -n/L_CL;
    d = (L_a*s + R_a);
    e = (k_eq + b_sp*s)*L_CL/n;
    my_transfer = tf(K_tau*a*b/(c*d*s^2*J_T - e*d + K_emf*K_tau*c*s + a*
        b*K_tau))
    % Plot figure and save
    fig = figure('units','normalized','outerposition',[0 0 1 1]);
    semilogx(f_sort, 20*log10(amplitude_factor_l(I')), '*-', 'Color', 'r
        ')
    hold on
    semilogx(f_sort, 20*log10(amplitude_factor_r(I')), '*-', 'Color', 'k
        ')
    PlotHandle= bodeplot(my_transfer);
    semilogx(f_sort, phasediff_l(I'), '*-', 'Color', 'r')
    semilogx(f_sort, phasediff_r(I'), '*-', 'Color', 'k')
    h = gcr
    ylims{1} = [-50, 10];
    ylims{2} = [-250, 10];
    setoptions(h,'FreqUnits','Hz','YLimMode','manual','YLim',ylims);
    set(findall(fig, 'type','axes'),'fontsize',16)
    PlotOptions= getoptions(PlotHandle);
    PlotOptions.Title.String= '';
    PlotOptions.XLabel.FontSize= 20;
    PlotOptions.YLabel.FontSize= 20;
    PlotHandle.setoptions(PlotOptions)
    legend(strcat('Analytical: b_{sp} = ', num2str(spring_damping_vec(k)
        ), 'Ns/m'), 'Experimental data left','Experimental data right',
        'Location', 'SouthWest')

    set(findall(gcf,'Type','line'),'LineWidth',4)
```

```
69      xlim([1  120])
70      grid on
71      hold on
72      filename = ['figures/bode_sp_damp' num2str(spring_damping_vec(k))];%
            backward_
73      saveas(gcf, [filename  '.jpg'])
74      dot_pos = strfind(filename,'.');
75      if (dot_pos)
76          saveas(gcf, [filename(1:dot_pos-1) filename(dot_pos+1:end)  '.m'
                ])
77      else
78          saveas(gcf, [filename  '.m'])
79      end
80      close(fig)
81  end
```

Matlab Code - Parameter File for Simulink Model

```
clc
close all
clear all
% Simulation
freq = 1 % [Hz]

% Motor
R_a = 118.6 % [Ohm]
L_a = 5.8E-3 % [Henri]
K_tau = 31.4E-3 % [Nm/A]
K_emf = 3.29E-3 / 60 % [V/s]
n = 112 % [-] reduction ratio

% Mechanical
J_T = 1.25E-3 / n / n % [kg m^2]
m_2 = 7E-3 % [kg]
L_CL = 20E-3 % [m]

% Springs
k_eq = 6E3 % [N/m]
b_spring = 300 % [Ns/m] damping coefficient of springs

% Operator // can be very high to simulate a wall or moderate for the
    real
% operator evnironment
k_op = 200 % [N/m] (stiffness of operators hand)
b_op = 20 % [Ns/m] (damping of operators hand)

% Arduino
MOTOR_MAX_V = 20 % [V]
dist_min = 2.2E-3 % [m]
dist_max = 4E-3 % [m]
dx_max = dist_min - dist_max % [m]
um_gain = -1000000 % [um/m]
K_V2dist = dx_max/5 % [m/V]
K_PID2Vardu = 1/51 % [V]
bit_offset = 128 % [-]

% Amplifier
K_ampl = 10 % [V/V]
V_offset = -25 % [V]
```

Matlab Code - Calculation of Moment of Inertia

```
%calculate J_TOT
n = 112
m_CL = 18.5E-3 % kg
A = 10E-3 % m
B = 6E-3 % m
L_CL = 20E-3 % m
m_base = 7E-3 % kg
m_BL = 2E-3 % kg
m_guideway = 5E-3 % kg
m_shaft = 2E-3 % kg
m_carr = m_base + m_BL + m_guideway + m_shaft % kg

J_motor = 6.8E-8 % kg*m^2
J_CL = m_CL/12 * (A^2 + B^2 + 12*(L_CL/2)^2) % kg*m^2
J_carr = m_carr*L_CL^2 % kg*m^2

J_TOT_motor = J_motor + J_CL/n^2 + J_carr/n^2 % kg*m^2
%J_TOT_refl = J_TOT_load / n^2
```

Simulink Models

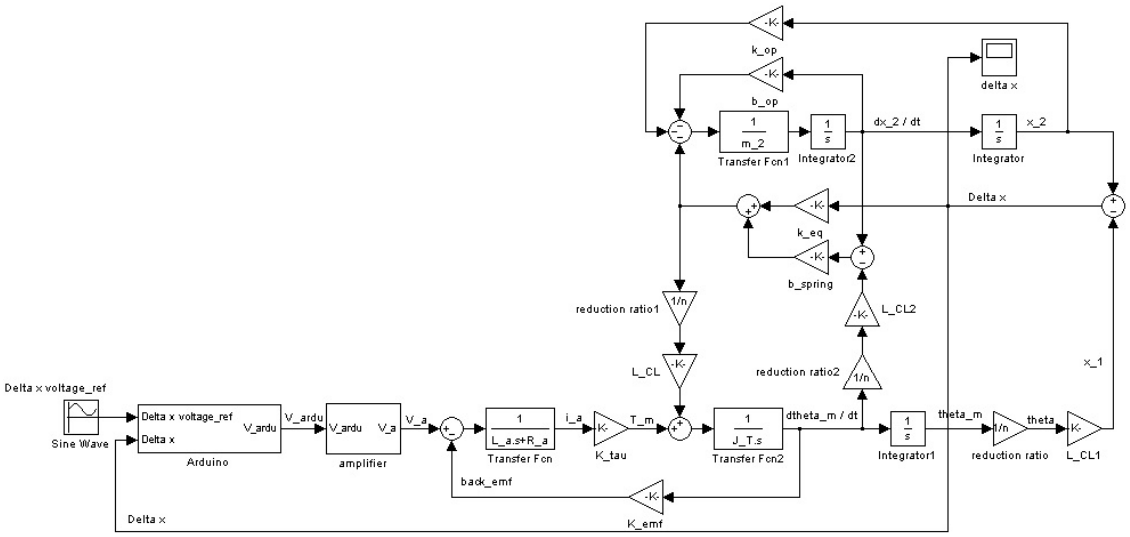


Figure 66: Main Simulink file.

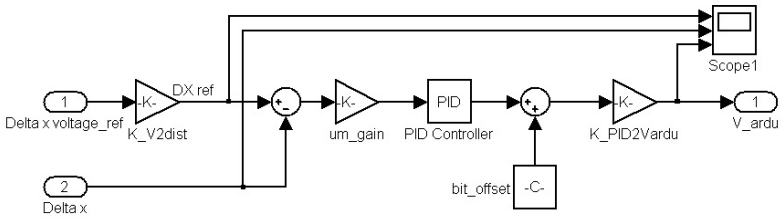


Figure 67: Simulink Arduino block.

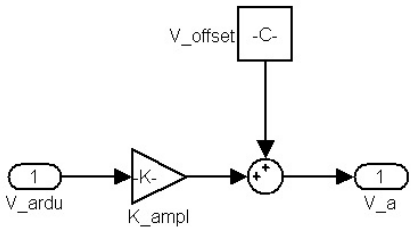


Figure 68: Simulink amplifier block.

Python Code - Create Different Files

```
"""
This program takes 1 long  raw data text file
    and creates several different text files
    according to the predefined frequency vector

Version: 1.0
Roman Oechslin
Master Thesis - University of Tokyo

"""

import shutil
directory = "tmp_text_files/"
src_filename = "raw_data.txt"
freq_vec = ["1", "1.25", "1.6", "2", "2.5", "3", "4", "5", "6.3", "8", "10", "13",
            "16", "20", "25", "32", "40", "50", "63", "80","100"]
import os

if not os.path.exists(directory):
    os.makedirs(directory)
source= open(src_filename, "r")
num_lines = sum(1 for line in open(src_filename))
print("number of lines in file = " + str(num_lines))
iter_len = num_lines / len(freq_vec)
print("iter len = " + str(iter_len))
loop = 0
ignore_lines = 1500

for frequency in freq_vec:
    print(frequency)
    destination = open(directory + "f" + frequency + ".txt", "w")

    with open(src_filename) as fp:
        for i, line in enumerate(fp):
            if i >= loop * iter_len + ignore_lines:
                if i < (loop + 1) * iter_len - ignore_lines:
                    destination.write(line)
            elif i < loop * iter_len + ignore_lines:
                pass
            else:
                break

    destination.close()
    loop = loop + 1
source.close()
```

Python Code - Pre-Processing Raw Data

```
"""
This program reads out the raw data saved in hexadecimal format in the text files
    and converts it to integer values and saves it as _csv.csv files.
    Due to high errors in the initial few lines, the first 100 lines are cut
    out and also the last two lines are left out.

Version: 1.0
Roman Oechslin
Master Thesis - University of Tokyo

"""

import shutil
import os

directory = "20180727_fra_logs_pilot_P10Vmm/raw_data/"


for filename in os.listdir(directory):
    print(filename)

    source= open(directory + filename, "r")
    destination= open("20180727_fra_logs_pilot_P10Vmm/" + filename[:-4] + "_csv"
        + ".csv", "w")
    count = 0
    num_lines = sum(1 for line in open(directory + filename))
    for line in source:
        count = count + 1
        if (count > 100 and count < num_lines - 2):
            buf = ""
            for character in line:
                if character == ";":
                    destination.write(str(int(buf, 16)) + ";")
                    buf = ""
                elif character == "\n":
                    destination.write("\n")
                    buf = ""
                else:
                    buf = buf + character
    source.close()
    destination.close()
```

Python Code - Frequency Response Analysis

```
import os
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle
from scipy import signal

from pylab import *
from math import atan2

PLOT_BOOL = True
CALC_OPER_FREQ = False
SHORT_CUT = True  # change this, if you want to skip all the calculation
only_this_file = "f"  # "f" if all frequencies shall be tested, otherwise "f2_csv"

INTMAX = 65535
PHOTO_MIN_LEFT = 630
PHOTO_MAX_LEFT = 795
PHOTO_MIN_RIGHT = 550
PHOTO_MAX_RIGHT = 765
MAX_DISPLACEMENT_UM = 5
directory = "20180727_fra_logs_pilot_P10Vmm/"  # change, if new data shall be analyzed


def get_freq_from_filename(filename):
    if filename[0] == "f" and filename[-8:] == "_csv.csv":
        return filename[1:-8]
    else:
        return ""


def sensor2dist(sensor_value, min_val, max_val):
    # maps the measured value to the distance (assumed linearity) in micrometers
    return (MAX_DISPLACEMENT_UM - (sensor_value - min_val) *
        MAX_DISPLACEMENT_UM / (max_val - min_val) )


def fitSine(tList, yList, freq):
    # from: http://exnumerus.blogspot.jp/2010/04/how-to-fit-sine-wave-example-in-python.html
    '''
        freq in Hz
        tList in sec
        returns
        phase in degrees
    '''
    b = matrix(yList).T
    rows = [[sin(freq * 2 * pi * t), cos(freq * 2 * pi * t), 1] for t in tList]
    A = matrix(rows)
    (w, residuals, rank, sing_vals) = lstsq(A, b)
    phase = atan2(w[1, 0], w[0, 0]) * 180 / pi
```

```
    amplitude = norm([w[0, 0], w[1, 0]], 2)
    bias = w[2, 0]
    return phase, amplitude, bias


phasediff_r = []
amplitude_factor_r = []
phasediff_l = []
amplitude_factor_l = []
freq_vec = []
for file in os.listdir(directory):
    if SHORT_CUT:
        amplitude_factor_l = [0.699, 0.706, 0.164, 0.750,
                              0.561, 0.435, 0.687, 0.751,
                              0.288, 0.182, 0.719, 0.118,
                              0.784, 0.084, 0.840, 0.052,
                              0.876, 0.873, 0.036, 0.024,
                              0.824]
        phasediff_l = [-26.744, -28.252, -139.027, -87.623,
                       -111.603, -122.307, -26.264, -33.660,
                       -136.889, 211.819, -30.130, 204.827,
                       -36.105, -165.420, -42.835, 185.875,
                       -49.770, -59.484, -183.017, 153.644,
                       -71.902]
        amplitude_factor_r = [0.003, 0.002, 0.043, 0.003,
                              0.002, 0.001, 0.005, 0.002,
                              0.004, 0.002, 0.001, 0.002,
                              0.001, 0.004, 0.001, 0.001,
                              0.002, 0.002, 0.001, 0.007,
                              0.003]
        phasediff_r = [54.935, -104.989, 18.806, 9.827,
                       -27.094, 44.405, -218.037, 4.153,
                       0.311, 314.340, -315.484, -28.529,
                       54.380, -9.592, -91.283, 99.501, 23.370,
                       -317.901, -244.663, 150.724, 22.131]

        freq_vec = [1.25, 1.6, 100.0, 10.0, 13.0, 16.0, 1.0, 2.5, 20.0, 25.0,
            2.0, 32.0, 3.0, 40.0, 4.0, 50.0, 5.0, 6.3, 63.0, 80.0, 8.0]
        continue

    if ".csv" not in file:
        continue
    frequency = get_freq_from_filename(file)
    frequency_float = float(frequency)
    print(frequency)

    # to restrict to first file only
    if only_this_file not in file:
        continue

    df = pd.read_csv(directory + file, delimiter=';')
    # make timeline continuous (no overflow), change [us] to [s] convert sensor val to [mm]
```

```
for i in range(len(df.iloc[:,3]) -1):
    if df.iloc[i,3] > df.iloc[i+1,3]:
        df.iloc[i+1:,3] = df.iloc[i+1:,3] + INTMAX
for i in range(len(df.iloc[:,3])):
    df.iloc[i, 3] = df.iloc[i, 3] / 1000000  # convert to seconds
    df.iloc[i,0] = df.iloc[i,0] / 1000  # convert reference to mm
    # convert sensor values to mm
    df.iloc[i,1] = sensor2dist(df.iloc[i,1], PHOTO_MIN_LEFT, PHOTO_MAX_LEFT)
    df.iloc[i,2] = sensor2dist(df.iloc[i,2], PHOTO_MIN_RIGHT, PHOTO_MAX_RIGHT)

print(df.iloc[:,1])
# just to have an average us length of a step:
if CALC_OPER_FREQ:
    step_sum = 0
    for i in range(len(df.iloc[:,3]) - 1):
        step_sum = step_sum + df.iloc[i+1,3] - df.iloc[i,3]
    # avg_step_len = step_sum / len(df.iloc[:,3])  # more precise mean
    avg_step_len = (df.iloc[-1,3] - df.iloc[0,3]) / len(df.iloc[:,3])  # global mean
    print("average step length for " + frequency + "Hz is " + str(avg_step_len))

if PLOT_BOOL:
    print("plotting freq " + frequency)
    fig = plt.figure()
    # plot left and right sensor data,zoom and save
    plt.plot(df.iloc[:, 3], df.iloc[:, 1])
    plt.plot(df.iloc[:, 3], df.iloc[:, 0] )  # plot reference
    fig.suptitle('Frequency response (' + frequency + ' Hz)')
    plt.xlabel('Time [s]')
    plt.ylabel('Compression [mm]')
    plt.axis([2, 3.6, 1.0, 4])
    plt.legend(["left sensor", "reference signal"])
    figure_directory = 'figs/f' + frequency
    if not os.path.exists(figure_directory):
        os.makedirs(figure_directory)
    fig.savefig(figure_directory + '/' + frequency + 'plot_zoom_fixed_time.jpg')
    plt.axis([2, 2 + 3 / frequency_float, 1.0, 4])
    fig.savefig(figure_directory + '/' + frequency + 'plot_zoom.jpg')
    with open(figure_directory + '/' + frequency + '_raw.pkl', "wb") as fp:
        pickle.dump(fig, fp, protocol=4)
    plt.close(fig)  # for not plotting all the pictures

# frequency response analysis
# input
yMeasured_in = df.iloc[:, 0]
tSamples = df.iloc[:, 3]
(phaseEst_in, amplitudeEst_in, biasEst_in) =
    fitSine(tSamples, yMeasured_in, frequency_float)

# output left
yMeasured_out_l = df.iloc[:, 1]
(phaseEst_out_l, amplitudeEst_out_l, biasEst_out_l) =
    fitSine(tSamples, yMeasured_out_l, frequency_float)
```

```
    # output right
    yMeasured_out_r = df.iloc[:, 2]
    (phaseEst_out_r, amplitudeEst_out_r, biasEst_out_r) =
        fitSine(tSamples, yMeasured_out_r, frequency_float)
    # magnitude and phase offset
    phasediff_l.append(phaseEst_out_l - phaseEst_in)
    amplitude_factor_l.append(amplitudeEst_out_l / amplitudeEst_in)
    phasediff_r.append(phaseEst_out_r - phaseEst_in)
    amplitude_factor_r.append(amplitudeEst_out_r / amplitudeEst_in)
    freq_vec.append(frequency_float)

# plot Bode diagram (freq_vec, magn_vec) and (freq_vec, phase_offset_vec)
# save figures
print("freq_vec = ")
print(freq_vec)
print("amplitude_factor_l = ")
print(amplitude_factor_l)
print("phasediff_l = " )
print(phasediff_l)
print("amplitude_factor_r = " )
print(amplitude_factor_r)
print("phasediff_r = ")
print(phasediff_r)
indices = np.argsort(freq_vec)
amplitude_factor_l = 20 * np.log10(amplitude_factor_l)
amplitude_factor_r = 20 * np.log10(amplitude_factor_r)



left_bode, axarr_l = plt.subplots(2, sharex=True)
left_lines = axarr_l[0].semilogx([freq_vec[x] for x in indices],
    [amplitude_factor_l[x] for x in indices], 'o-', linewidth=5.0, label='Left side')
axarr_l[0].set_title('Bode diagram - left side (reduction 33:1)', fontsize=25)
"""
right_lines = axarr_l[0].semilogx([freq_vec[x] for x in indices],
    [amplitude_factor_r[x] for x in indices], 'o-', linewidth=5.0, label='Right side')
axarr_l[0].legend()
axarr_l[1].semilogx([freq_vec[x] for x in indices], [phasediff_r[x] if
    phasediff_r[x] < 0 else phasediff_r[x] - 360 for x in indices], 'o-',
    linewidth=5.0, label='Right side')
axarr_l[0].set_title('Bode diagram (reduction 112:1)', fontsize=25)
#"""
axarr_l[1].semilogx([freq_vec[x] for x in indices], [phasediff_l[x] if
    phasediff_l[x] < 0 else phasediff_l[x] - 360 for x in indices], 'o-',
    linewidth=5.0, label='Left side')
#axarr_l[1].semilogx([freq_vec[x] for x in indices], [phasediff_l[x]  for x in
indices])
axarr_l[1].set_xlabel('Frequency [Hz]', fontsize=25)
axarr_l[0].set_ylabel('Magnitude [dB]', fontsize=25)
axarr_l[1].set_ylabel('Phase [deg]', fontsize=25)
axarr_l[0].tick_params(axis='x', labelsize=10)
```

```
axarr_l[1].tick_params(axis='x', labelsize=10)
axarr_l[0].tick_params(axis='y', labelsize=10)
axarr_l[1].tick_params(axis='y', labelsize=10)
axarr_l[0].set_ylim(-50,10)
axarr_l[1].set_ylim(-250,0)
axarr_l[0].grid()
axarr_l[1].grid()
figure_directory = 'figs/'
mng = plt.get_current_fig_manager()
mng.resize(*mng.window.maxsize())
plt.show()
left_bode.savefig(figure_directory + '/bode_left.jpg')

right_bode, axarr_r = plt.subplots(2, sharex=True)
axarr_r[0].semilogx([freq_vec[x] for x in indices], [amplitude_factor_r[x] for x
    in indices], 'o-', linewidth=5.0)
axarr_r[0].set_title('Bode diagram - right side (reduction 33:1)', fontsize=25)
axarr_r[1].semilogx([freq_vec[x] for x in indices], [phasediff_r[x] if
    phasediff_r[x] < 0 else phasediff_r[x] - 360 for x in indices], 'o-',
    linewidth=5.0)
#axarr_r[1].semilogx([freq_vec[x] for x in indices], [phasediff_r[x] for x in
indices])
axarr_r[1].set_xlabel('Frequency [Hz]', fontsize=25)
axarr_r[0].set_ylabel('Magnitude [dB]', fontsize=25)
axarr_r[1].set_ylabel('Phase [deg]', fontsize=25)
axarr_r[0].tick_params(axis='x', labelsize=10)
axarr_r[1].tick_params(axis='x', labelsize=10)
axarr_r[0].tick_params(axis='y', labelsize=10)
axarr_r[1].tick_params(axis='y', labelsize=10)
axarr_r[0].set_ylim(-50,10)
axarr_r[1].set_ylim(-250,0)
axarr_r[0].grid()
axarr_r[1].grid()
figure_directory = 'figs/'
mng = plt.get_current_fig_manager()
mng.resize(*mng.window.maxsize())
plt.show()
right_bode.savefig(figure_directory + '/bode_right.jpg')
```

6. 通信コマンド Communication command

国立大学法人 東京大学殿向け 実験用レスキューロボットシステム S2-P1 (2フリッパ特注) 通

●仮想ＣＯＭポート設定「Com Port Redirector」にて設定した仮想ＣＯＭポート（設定したIPアドレス）にて、下記の送受信を行います。
●送受信文字（コマンド）は、バイナリ形式です。Transmit/Receive characters (command) are in binary format.
●送信文字（コマンド）は、0.1～0.8秒の送信間隔で送信してください。Send the transmitted character (command) at the transmi
●ロボット本体は、操作用パソコンから１秒以上、正常な通信コマンド（サムチェック含む）が受信できないと、自動的に走行（クローラー）とフリ
●ロボット本体からの返信コマンドの中の「経過時間（秒）」を折り返し操作用パソコンから送信し、その「受信した時間」と「ロボット本体の内部カウント時間」の差が送信
●ロボット本体からは、操作用パソコンからのコマンドを正常に受信できた場合（サムチェック正常時）にのみ、操作用パソコンに返信（１回のみ
※コマンド仕様は、変更になる場合があります。From the robot body we will reply (only once) to the operating PC only

Transmission data

【送信データ】（操作用パソコン ⇒ ロボット本体）

PC for operation => robot body

1	2	3	4	5	6	7	8	9	10	11	12
First character 先頭文字 (固定) (fixed) FFEEh		Right crawler speed command 右クローラー速度指令 -99～99 (%)		Left crawler speed command 左クローラー速度指令 -99～99 (%)		Acceleration X offset 加速度Xオフセット		Acceleration Y offset 加速度Yオフセット		Acceleration Z offset 加速度Zオフセット	

23	24	25	26	27	28	29	30	31	32	33	34
Crawler command クローラー コマンド	Flipper command 1 フリッパー コマンド1	Flipper command 2 フリッパー コマンド2	Flipper speed command フリッパー速度指令 -99～99 (%) -99 ~ 99 %		Flipper angle command フリッパー角度指令 -44～＋224 (°)		unused 未使用 (固定) 00h (fixed)	Flipper current フリッパー電流オフセット値 offset value		フリッパー電流制限値 10倍 (A)	unused 未使用 (固定) 00h fixed

(1-2) ■ 先頭文字（固定） first character — (fixed)
固定値（FFEEh）を設定します。

(3-6) ■ （右左）クローラー速度指令 (Right and left) crawler speed command set value
クローラーの速度指令値（-99～99%）を設定します。-99～99%
※超信地旋回時は上限速度がマイコン側にて最大30%に制限されます。ultra fine speed the upper limit speed is limited to 30% at micro computer side when turning

(7-8) ■ 加速度Xオフセット Acceleration X offset
ロボット水平時の加速度Xの値を設定してください。
Please set the value of acceleration X when the robot is horizontal

(9-10) ■ 加速度Yオフセット Acceleration Y offset
ロボット水平時の加速度Yの値を設定してください。

(11-12) ■ 加速度Zオフセット Acceleration Z offset
ロボット水平時の加速度Z+164の値を設定してください。
Set value to +164 for horizontal offset

(13-14) ■ 前進／後退傾斜制御角度 Forward/Backward tilt control angle sets the stop angle when
前進／後退時における、停止角度を設定します。advancing/backing up
走行中、傾斜角度が設定範囲を超えた場合に停止します。Running operation stops when the angle of inclination exceeds the set range
swivel operation is impossible

(15-16) ■ 全体電流オフセット値 Overall current offset value for calculating total current. The
全体電流算出のオフセット値を設定します。offset value of the current sensor may fluctuate depending on surrounding temperature, etc. Enter value of
電流センサのオフセット値は、周辺の温度等により変動する可能性があります。
（ロボット停止時の値：約520を入力してください）about 520 when stopping the robot
※厳密に設定する必要はありません。It is not necessary to set it strictly (exact)

(17-18) ■ クローラー電流オフセット値 Crawler current offset value Set the offset value
クローラー電流算出のオフセット値を設定します。for calculating the crawler current
電流センサのオフセット値は、周辺の温度等により変動する可能性があります。may fluctuate depending on
（ロボット停止時の値：約520を入力してください）temperature. Enter the value of
※厳密に設定する必要はありません。about 520 when stopping the robot. It is not necessary to set exact value

19 ■ クローラー電流制限値 Crawler current limit value, sets current limit value
走行時の電流制限値を設定します。（10倍値）10 times value / when running, operation
走行中、電流が制限値を超えた場合には停止します。stops when value is exceeded
※通常は、設定値に60を設定してください。（6A×10倍）normally set to 60
※設定可能な最大値は80（8A×10倍）です。(6A×10) maximum value is 80.
上記の設定値以上を入力した場合、上記の最大値になります。
When a higher value is input, it becomes 80.

(23) ■ クローラーコマンド Crawler command (flipper current limit value)

1bit	動力電源 ON/OFF (1/0) when power supply c ビットをONにすると、動力電源をONします。power 動力OFFの場合、クローラーやフリッパーが動作せず、フリー状 （姿勢を維持しません）does not maintain posi
2bit	クローラーモータイネーブル ON/OFF (2/0) Crawler motor en ビットをONにすると、左右のクローラーモータをイネーブル状態に 旋回異常制限解除 (4/0) turn abnormality restricti ビットをONにすると、フリッパーが地面に接地した状態
3bit	走行傾斜制限解除 ON/OFF (8/0) traveling tilt li ビットをONにすると、走行中（前進/後退時）の傾斜停 傾斜制限を解除すると、傾斜を走行中に転倒する 通常は、解除しないでください。 Stop limit dis
4bit	階段モード ON/OFF (16/0) staircase mode (ena 階段モードを有効にする際にビットをONします。turn o ※階段モードではフリッパーが地面に接地した状態で走行 （旋回操作は不可）staircase mode enables turning
5bit	クローラー電流制御解除 ON/OFF (32/0) crawler current c ビットをONにすると、クローラーモータの電流制限が解除さ 電流制御を解除すると、クローラーモータの電流値が抑制 モータの寿命低下や破損につながる恐れがあります。
6bit	クローラ温度制限解除 (64/0) Crawler temperature lim ビットをONにすると、クローラーモータの温度制限が解除さ 温度制限を解除するとモータが焼損する恐れがありま
7bit	走行傾斜速度制限解除 ON/OFF (128/0) running inc ビットをONにすると、走行中（前進/後退時）の傾斜速 when bit is turned ON, inclination speed limit dur
8bit	

(24) ■ フリッパーコマンド1 Flipper command 1

1bit	フリッパーモータOFF Flipper motor OFF フリッパーモータをフリーにします。Free flipper motor 通常出力 normal output, rotate the motor w 速度制御や位置制御を行わずモータを回転させます。 モータ停止中に外力が加わる場合、姿勢を維持できま
2bit	位置制御（速度指令）Position control (speed com 速度指令を行いたい場合に、設定してください。 位置制御（角度指令）Position control (angle co 角度指令を行いたい場合に、設定してください。
3bit	
4bit	
5bit	角度リセット Angle reset current flipper 現在のフリッパー角度を0° に設定します。Set angle

(25) ■ フリッパーコマンド2 flipper command 2

1bit	移動量制限解除 ON/OFF (1/0) release limit rest フリッパーの移動量制限（＋約224° ～－約44° ※クロー を解除します。 ※転倒する恐れがあるため、通常は解除しないで
2bit	フリッパー電流制限解除 ON/OFF (2/0) Flipper c ビットをONにすると、フリッパーモータの電流制限が解除さ 電流制御を解除すると、フリッパーモータの電流値が抑制 モータの寿命低下や破損につながる恐れがあります。
3bit	フリッパー温度制限解除 (4/0) Temperature lim ビットをONにすると、フリッパーモータの温度制限が解除さ 温度制限を解除するとモータが焼損する恐れがありま

コマンド

13	14	15	16	17	18	19	20	21	22
傾斜角度	後退傾斜制御角度	全体 電流オフセット値		クローラー 電流オフセット値		クローラー 電流制限値 10倍 (A)	未使用 (固定) 00h	未使用 (固定) 00h	未使用 (固定) 00h

35	36	37	38	39	40	41	42	43	44
使用 定) 0h	未使用 (固定) 00h	未使用 (固定) 00h	未使用 (固定) 00h	未使用 (固定) 00h	比較時間 (秒)	経過時間 (秒)		チェックサム	末端文字 (固定) 0Dh

■ フリッパ-速度指令
【通常出力、位置制御(速度指令)時】
フリッパ-の速度指令値(-99～99%)を設定します。
【位置制御(位置指令)時】
フリッパ-の速度指令値(0～99%)を設定します。

■ フリッパ-角度指令
フリッパ-位置制御時の指令角度(-44～224°)を設定します。
フリッパ-の角度が指令角度に到達すると動作を停止します。

■ フリッパ-電流オフセット値
フリッパ-電流算出のオフセット値を設定します。
電流センサのオフセット値は、周辺の温度等により変動する可能性があります。
(ロボット停止時の値：約520を入力してください)
※厳密に設定する必要はありません。

■ フリッパ-電流制限値
フリッパ-の電流制限値を設定します。(10倍値)
フリッパ-動作中、電流が制限値を超えた場合には停止します。
※通常は、設定値に15を設定してください。(1.5A×10倍)
※設定可能な最大値は、17(1.7A×10倍)

■ 比較時間(秒)
比較時間(1～255)を秒で設定します。通常は2秒を設定してください。
※ロボット本体からの返信コマンドの中の「経過時間(秒)」を操作パソコンから送信し、その「受信した時間」と「ロボット本体の内部カウント時間」の差が送信データの比較時間(秒)以上あった場合、走行とフリッパ-を停止します。

■ 経過時間(秒)
ロボットが起動してからのおおよその経過時間(0～28800秒)を設定します。
※経過時間は、8時間(28800秒)でリセットされます。

■ チェックサム
送信データの1～42byteの足し合わせした値を設定します。

■ 末端文字(固定)
固定値(0Dh)を設定します。

Receive data robot main body → operating personal computer

【受信データ】（ロボット本体 ⇒ 操作用パソコン）

1	2	3	4	5	6	7	8	9	10	11	12	13
First character 先頭文字 (固定) FFEEh (fixed)		unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	Overall current value 全体 電流値		Right cr current 右 電

26	27	28	29	30	31	32	33	34	35	36	37	38
Acceleration X 加速度X		Acceleration Y 加速度Y		Acceleration Z 加速度Z		Right crawler 右クローラー 温度 temperature	unused 未使用 (固定) 00h (fixed)	Left crawler 左クローラー 温度 temperature	unused 未使用 (固定) 00h (fixed)	internal 内部温度 temperature	battery バッテリ 電圧 voltage	crawler status クローラー ステータス

51	52	53	54	55	56	57	58	59	60	61	62	63
Flipper encoder value フリッパー エンコーダ値				Flipper angle フリッパー 角度(°)		Flipper current value フリッパー 電流値		Flipper current フリッパー 電流 制限値 (%) limit value	Flipper フリッパー 温度 temperature	Flipper フリッパー ステータス status	manufacturer メーカ確認用	

(1-2) ■ 先頭文字(固定) The first character (fixed value) is set
固定値(FFEEh)が設定されます。

(11-12) ■ 全体電流値 overall current value, value of total current is stored, calculation is possible with current (A) = (received value - offset value) * 48 / 1024. offset value may
全体電流の値が格納されます。
電流(A) = (受信値 − ｵﾌｾｯﾄ値) ×48 / 1024 にて算出可能です。fluctuate, please enter the value when robot stopped
ｵﾌｾｯﾄ値は、周辺の温度等により変動する可能性があります。
(ﾛﾎﾞｯﾄ停止時の値を入力してください)

(13-14) (17-18) ■ ｸﾛｰﾗｰ電流値 crawler current value current (A) = (received value - offset value) * 16/1024
ｸﾛｰﾗｰの電流値が格納されます。 offset value may fluctuate due to temperature change
電流(A) = (受信値 − ｵﾌｾｯﾄ値) × 16 / 1024 にて算出可能です。
ｵﾌｾｯﾄ値は、周辺の温度等により変動する可能性があります。 please enter the value when stopping the robot
(ﾛﾎﾞｯﾄ停止時の値を入力してください)

(21) ■ ｸﾛｰﾗｰ電流制限値(%) crawler current limit value(%) when current limit value is exceeded
ｸﾛｰﾗｰﾓｰﾀの電流制限値を超えた場合にﾓｰﾀへの出力を制限し、その値が格納されます。 the output to the motor is limited and the value is stored. (The value in the
(制限されていない状態での値は100%です) unrestricted state is 100%.

(22-23) (24-25) ■ 前後/左右角度 pitch / roll angle the angles of pitch and roll are stored
ﾛﾎﾞｯﾄ本体に搭載された加速度ｾﾝｻの値から算出した傾斜角度の値(°)が格納されます。

(26-27) (28-29) (30-31) ■ 加速度(XYZ) Acceleration XYZ the value of the acceleration sensor of the robot body is stored.
ﾛﾎﾞｯﾄ本体の加速度ｾﾝｻｰの値が格納されます。 Acceleration (G) = (Receive) value - offset
加速度(G) = (受信値 − ｵﾌｾｯﾄ値) / 1023 × 5 / 0.8 value) / 1023 * 5 / 0.8

(32) (34) (36) ■ 温度(℃) Temperature
温度ｾﾝｻの値(°)が格納されます。 The temperature sensor value is stored

(37) ■ ﾊﾞｯﾃﾘ電圧 Battery voltage value of battery voltage is stored
ﾊﾞｯﾃﾘ電圧の値が格納されます。 Battery voltage (V) = received value / 255 * 30
ﾊﾞｯﾃﾘ電圧(V) = 受信値 / 255 × 30

(38) ■ ｸﾛｰﾗｰｽﾃｰﾀｽ crawler status

1bit	右ｸﾛｰﾗｰﾓｰﾀｲﾈｰﾌﾞﾙ中(1/0)	Right crawler motor enabled, bit turned ON = ENABLE
	右ｸﾛｰﾗｰﾓｰﾀがｲﾈｰﾌﾞﾙ(Enable)状態になるとﾋﾞｯﾄがONします。	
2bit	左ｸﾛｰﾗｰﾓｰﾀｲﾈｰﾌﾞﾙ中(2/0)	Left crawler motor enabled, bit turned ON = ENABLE
	左ｸﾛｰﾗｰﾓｰﾀがｲﾈｰﾌﾞﾙ(Enable)状態になるとﾋﾞｯﾄがONします。	
3bit	走行傾斜制限中(4/0)	running elimination limit in progress, bit turned ON when
	走行傾斜制限中状態になるとﾋﾞｯﾄがONします。	traveling tilt restricted state is entered
4bit	未使用	unused
5bit	未使用	
6bit	未使用	
7bit	未使用	
8bit	ｸﾛｰﾗ温度異常(128/0)	crawler temperature abnormality
	ｸﾛｰﾗ温度が80℃を超えた場合に走行が停止し、ﾋﾞｯﾄがONします。	when the crawler temperature exceeds 80°C Operation stops and bit is turned ON

(40) ■ ｸﾛｰﾗｰｽﾃｰﾀｽ2 crawler status 2

1bit	走行傾斜速度制限中(1/0)	traveling tilt speed restricted
	走行傾斜が15°以上になると走行速度を約0.1km/h以下に制限し、ﾋﾞｯﾄがONします。	when the running inclination ≥ 15° running speed is limited to 0.1 km/h or less bit = ON
2bit	ｸﾛｰﾗ旋回異常(2/0)	crawler turning abnormality
	ﾌﾘｯﾊﾟｰが地面についた状態(0°以下、180°以上)で旋回操作を行うと、ｸﾛｰﾗが停止し、ﾋﾞｯﾄがONします。	when the turning operation is performed in a state where the flipper is attached to ground (0° or 180°) crawler stops, bit = ON
3bit	ｸﾛｰﾗ旋回警告(4/0)	crawler turning warning
	ﾌﾘｯﾊﾟｰが地面についた状態(0°以下、180°以上)でﾋﾞｯﾄがONします。	bit = ON when flipper is on ground (0° or 180°)
4bit	未使用	unused
5bit	未使用	
6bit	未使用	
7bit	未使用	
8bit	未使用	

(39) ■ ﾏｲｺﾝｽﾃｰﾀｽ microcomputer status

1bit	動力電源指令ON中(1/0)	Power su
	動力電源指令ON中に、ﾋﾞｯﾄがONしま	
2bit	動力電源状態ON中(2/0)	Power s
	動力電源状態ON中(確認信号ON)に、	
3bit	ﾏｲｺﾝCAN通信異常(4/0)	microcompa
	ﾛﾎﾞｯﾄ本体内のﾏｲｺﾝ間に通信異常が	
4bit	経過時間差(通信間隔)異常(8/0)	el
	操作ﾊﾟｿｺﾝから送信した経過時間と、	
	の差が送信ﾊﾟﾗﾒｰﾀの比較時間(2秒	
	ﾋﾞｯﾄがONします。	and the elapse parameter the
5bit	未使用	not used
6bit	未使用	
7bit	未使用	
8bit	未使用	

(51-54) ■ フリッパー エンコーダ値 Flipper encoder va
回転量検出ｾﾝｻ(ｴﾝｺｰﾀﾞ)の値が格納されます。

(55-56) ■ フリッパー角度 Flipper angle Angle
フリッパー角度(°)の値が格納されます。
※ﾛﾎﾞｯﾄ電源投入時、または角度ﾘｾｯﾄ時のﾌﾘｯ

(57-58) ■ フリッパー電流値 Flipper current value
フリッパーの電流値が格納されます。 Calculatio
電流(A) = (受信値 − ｵﾌｾｯﾄ値) × 16 / 102
ｵﾌｾｯﾄ値は、周辺の温度等により変動する可能
(ﾛﾎﾞｯﾄ停止時の値を入力してください)

(59) ■ フリッパー電流制限値(%) Flipper current
フリッパーモータの電流制限値を超えた場合にﾓｰﾀへ
(制限されていない状態での値は100%です) T

(60) ■ フリッパー温度 flipper temperature
温度ｾﾝｻの値(°)が格納されます。 tempe

(61) ■ フリッパーステータス flipper status

1bit	フリッパーモータイネーブル中(1/0)	flipper
	フリッパモータがｲﾈｰﾌﾞﾙ(Enable)状態にな	
2bit	フリッパー角度ﾋﾞｯﾄﾘｾｯﾄ完了(2/0)	flipp
	フリッパー角度のﾋﾞｯﾄﾘｾｯﾄが完了するとﾋ	
3bit	フリッパーエンコーダ異常(4/0)	flipper e
	フリッパーの位置制御中にｴﾝｺｰﾀﾞ異常が	
4bit	フリッパー温度異常(8/0)	temperatur
	フリッパー温度が80℃を超えた場合にフリ	
5bit	フリッパー上昇リミット(16/0)	flippe a
	フリッパーが上昇限界角度(224°)を超え	
6bit	フリッパー下降リミット(32/0)	flipper lo
	フリッパーが上昇限界角度(-44°)を超え	
7bit	フリッパー角度指令異常(64/0)	flipper
	フリッパー角度が上昇限界角度(225°)以	
	フリッパーが停止し、ﾋﾞｯﾄがONします。	
8bit	未使用	unused

1 box = 1 byte

13	14	15	16	17	18	19	20	21	22	23	24	25
ght crawler rrent value 右クローラー 電流値		unused 未使用 (固定) 00h (fixed)	(unused) 未使用 (固定) 00h (fixed)	left crawler current value 左クローラー 電流値		unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	Crawler クローラー 電流 制限値 (%) current limit value	pitch angle 前後角度		(Right and left) roll angle 左右角度	

38	39	40	41	42	43	44	45	46	47	48	49	50
wler tus ローラー テータス	microcomputer status マイコン ステータス	Crawler status 2 クローラー ステータス2	for manufacturer confirmation メーカ確認用						unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)

63	64	65	66	67	68	69	70
rver confirmation 確認用		unused 未使用 (固定) 00h (fixed)	unused 未使用 (固定) 00h (fixed)	elapsed time (seconds) 経過時間(秒)		check sum チェックサム	end character 末端文字 (固定) 0Dh (fixed)

us
er supply command (ON/OFF)
がONします。 bit = ON: Power = ON
er supply status: bit is turned ON while power supply statu
ON) に、ビットがONします。 is turned ON (confirmation signal ON)
computer CAN communication error. bit = ON if commu-
異常が発生するとビットがONします。cation error occurred
8/0) elapsed time difference (communication interval
時間と、ロボット本体にて算出している経過時間 abnormality
引 (2秒等) 以上ある場合、走行やフリッパーを停止して、 when the difference between the
e elapsed time calculated by the robot body is ≥ than comparison time (2 seconds) of the transmission
the robot operation is stopped (running and flipper), the bit turns ON

(67-68)
■ 経過時間 (秒) ~~Disconnection~~ elapsed time (sec) approximate elapsed time (0 to 2880
ロボットが起動してからのおおよその経過時間 (0~28800秒) が格納されます。 since the robot started, i
※経過時間は、8時間 (28800秒) でリセットされます。 The elapsed time is reset after 8 hours
(69)
■ チェックサム check sum, the summed value of 1~68 bytes of the checksum
受信データの1~68byteの足し合わせた値が格納されます。 received data is stored
(70)
■ 末端文字 (固定) end character (fixed value)
固定値 (0Dh) が格納されます。

er value
れます。 The value of the rotation amount detection sensor is stored (encoder).

Angle of flipper is stored
時のフリッパ角度が0° になります。 The angle of the robot is 0° when power is turned ON or angle is reset.

value , current value of flipper is stored
culation is possible with current [A] = (received value - offset value) * 165 / 1024
i / 1024 にて算出可能です。offset value may fluctuate (because of temperature)
する可能性があります。 please enter the value as offset when robot is stopped.
い)

urrent limit value free when the motor limit value is exceeded, the output to the motor
こモータへの出力を制限し、その値が格納されます。 is limited and the value is stored
です) The unrestricted value is 100%.

ure
emperature sensor value is stored

lipper motor enabled =ON
状態になるとビットがONします。
) flipper 0-resetting angle completed (=ON)
するとビットがONします。
per encoder error
異常が発生するとフリッパ動作が停止し、ビットがONします。 operation stops and bit turns ON if error detected
perature abnormality
場合にフリッパーが停止し、ビットがONします。 when 80°C exceeded : bit = ON, operation stops
per current limit
) を超えてフリッパー上昇指令を出すとフリッパーが停止し、ビットがONします。 when flipper exceeds the rising limit, bit is turned ON (22.4°)
per lowering limit
) を超えてフリッパー下降指令を出すとフリッパーが停止し、ビットがONします。 when flipper exceeds the lowering limit, operation stopped, bit turned ON (-44°)
flipper angle command error
225°) 以上、または下降限界角度 (-45°) 以下の時に角度指令を出すと if outside of limits commands are sent, flipper stops, bit = ON
ます。

6

How to reboot microcomputer that uses Xport I/O port

<Xport の I/O ポートを利用したマイコンの再起動方法>

Perform the processing with following flow

以下の流れで、処理を行ってください。

①COM ポートを閉じる。 Close COM port

②IP アドレスとポート番号を指定して、Xport とソケット通信を行う。 Specify IP address and port number and perform socket communication with Xport
IPアドレス:190.160.0.1
ポート番号:30704

③以下の送信データで送信処理を行う。(9byte 送信) Transmission processing is performed with transmission data of 3 or less. (9 byte transmission)
※XPORT の I/O ポート(P3)を ON に設定 Set the I/O port (P3) of Xport to ON

1Byte	2Byte	3Byte	4Byte	5Byte	6Byte	7Byte	8Byte	9Byte
1BH	07H	00H	00H	00H	07H	00H	00H	00H

④受信処理を行う。(5byte 受信) receive the value (5 bytes received)
※受信した値の 2byte 目に 4bit(04H)が ON しているか確認 confirm whether 4 bit is ON at 2nd byte of the received value

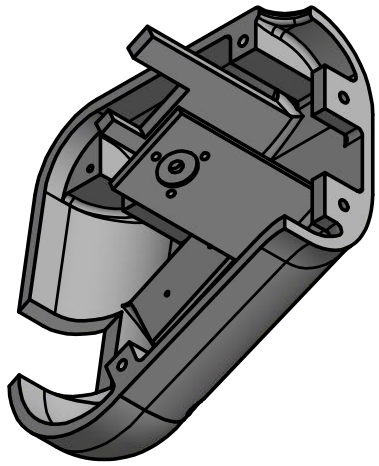
⑤約 1 秒待つ。 wait 1 second

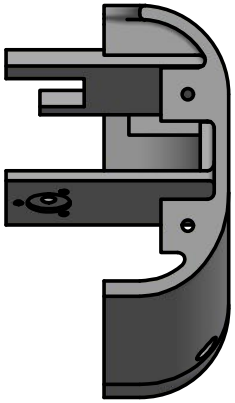
⑥以下の送信データで送信処理を行う。(9byte 送信) transmission processing is performed with the following transmission data (9 byte transmission)
※XPORT の I/O ポート(P3)を OFF に設定 Set the I/O port (P3) of Xport to OFF

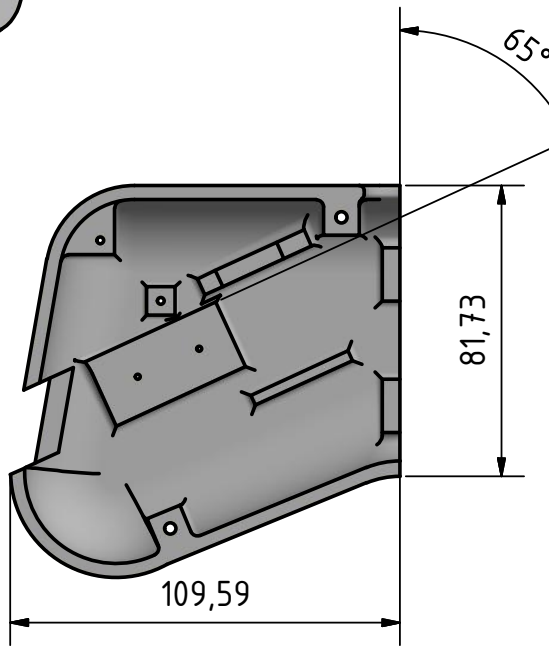
1Byte	2Byte	3Byte	4Byte	5Byte	6Byte	7Byte	8Byte	9Byte
1BH	07H	00H	00H	00H	00H	00H	00H	00H

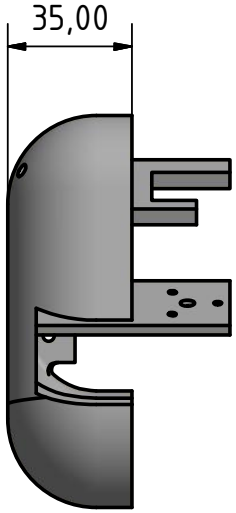
⑦受信処理を行う。(5byte 受信) receive the value (5 bytes received)
※受信した値の 2byte 目に 4bit(04H)が OFF しているか確認
Confirm whether 4 bit is OFF at the 2nd byte of the received value.

6

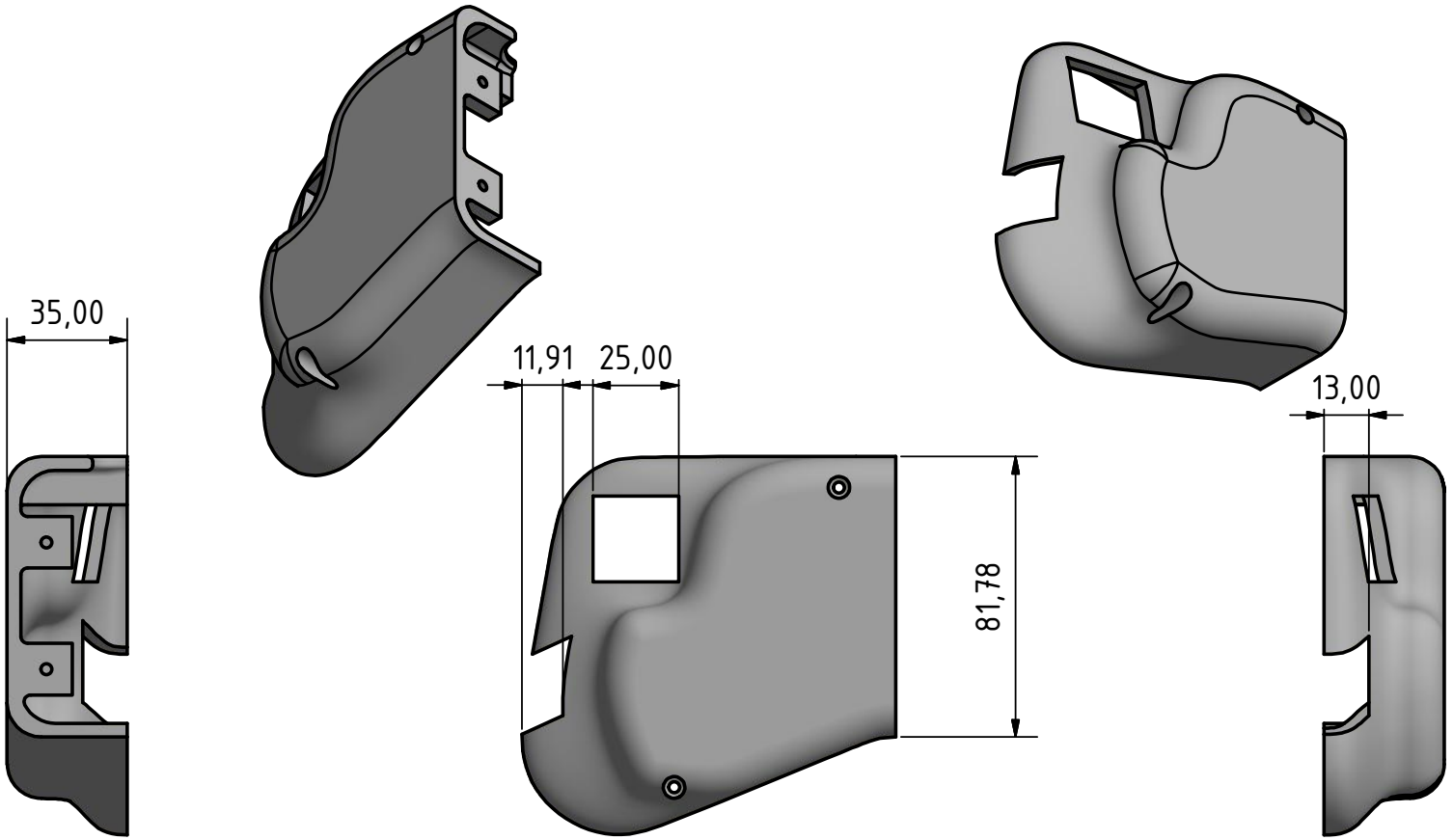




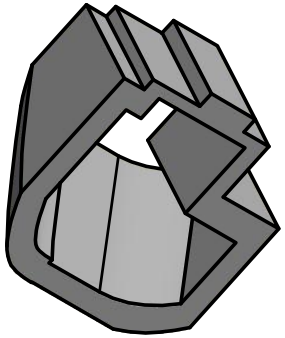


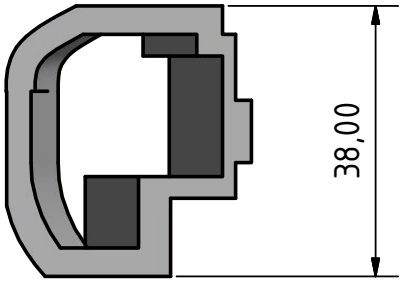


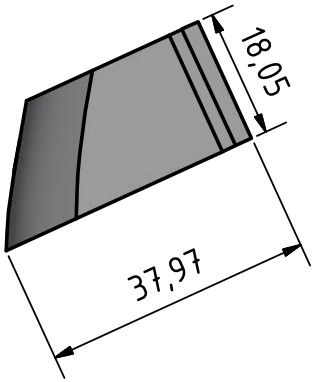
DRAWN oechslin	8/7/2018	EPFL
CHECKED		TITLE: Left Base
QA		
MFG		
APPROVED		
		SIZE A4 | DWG NO Game Controller | REV
		SCALE 1 : 2 | SHEET 2 OF 4

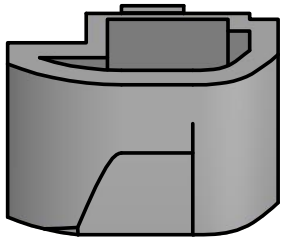


DRAWN oechslin	8/7/2018	EPFL
CHECKED		TITLE: Left Cover
QA		
MFG		
APPROVED		
		SIZE A4 / DWG NO Game Controller / REV
		SCALE 1 : 2 / SHEET 3 OF 4

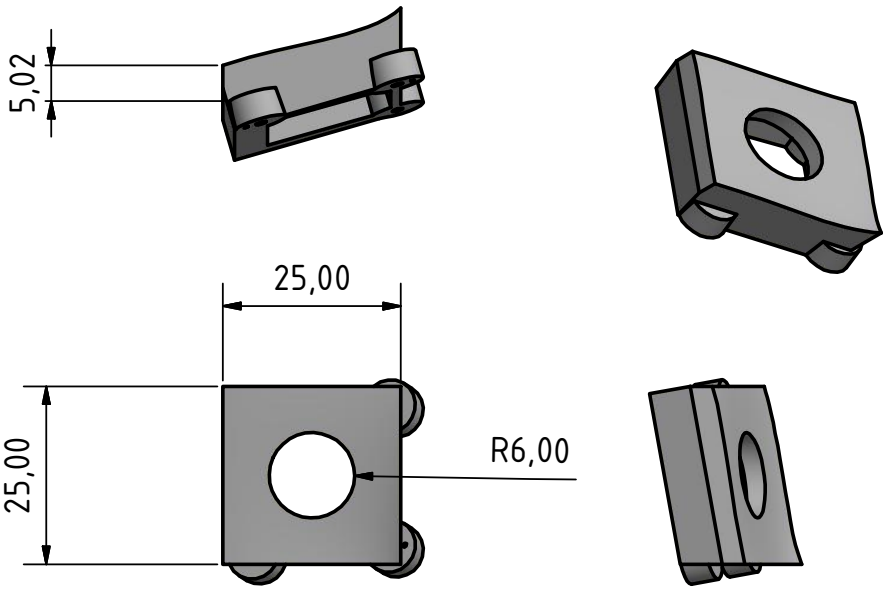




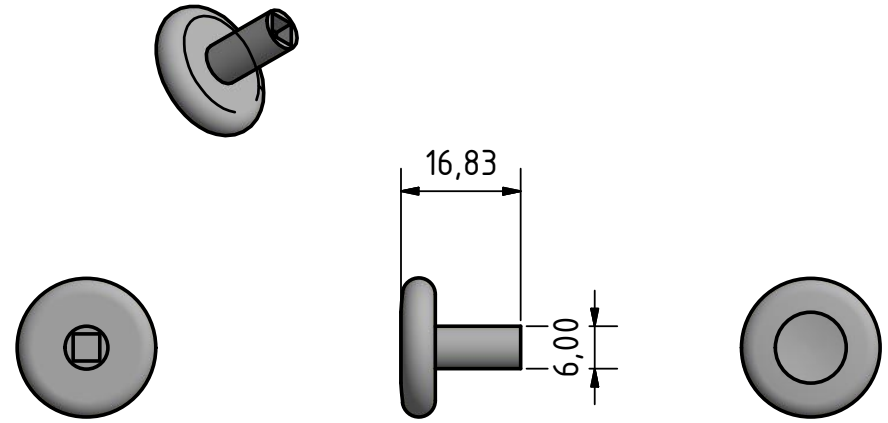




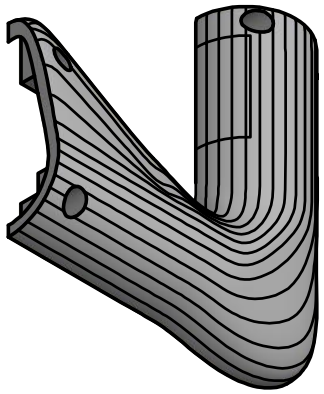
DRAWN oechslin	8/7/2018	EPFL		
CHECKED		TITLE		
QA		Left Palm Pad		
MFG				
APPROVED		SIZE A4	DWG NO Game Controller	REV
		SCALE 1 : 1		SHEET 1 OF 4

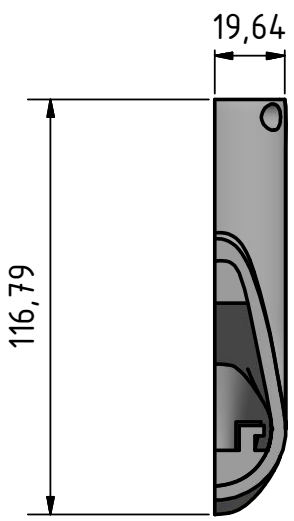


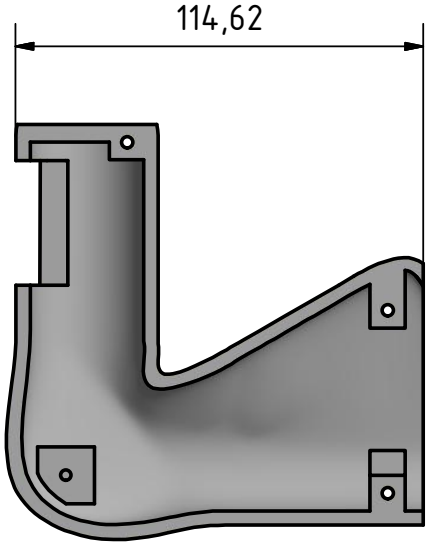
DRAWN oechslin	8/7/2018	EPFL		
CHECKED		TITLE		
QA		Left Joystick Mount		
MFG				
APPROVED		SIZE A4	DWG NO Game Controller	REV
		SCALE 1 : 1	SHEET 4 OF 4	



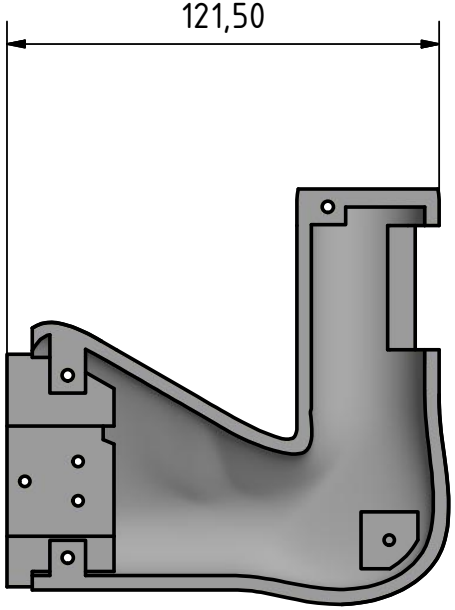
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA				
MFG		Joystick Nibble		
APPROVED		SIZE	DWG NO	REV
		A4	Yoke Controller	
		SCALE 1 : 1	SHEET 6 OF 10	

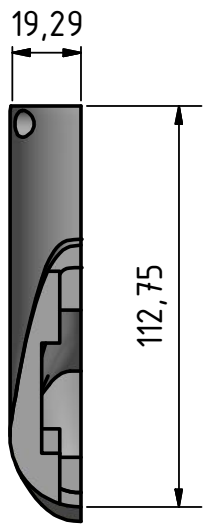


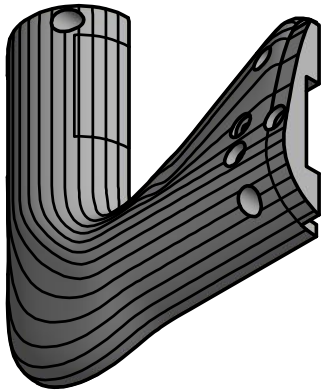




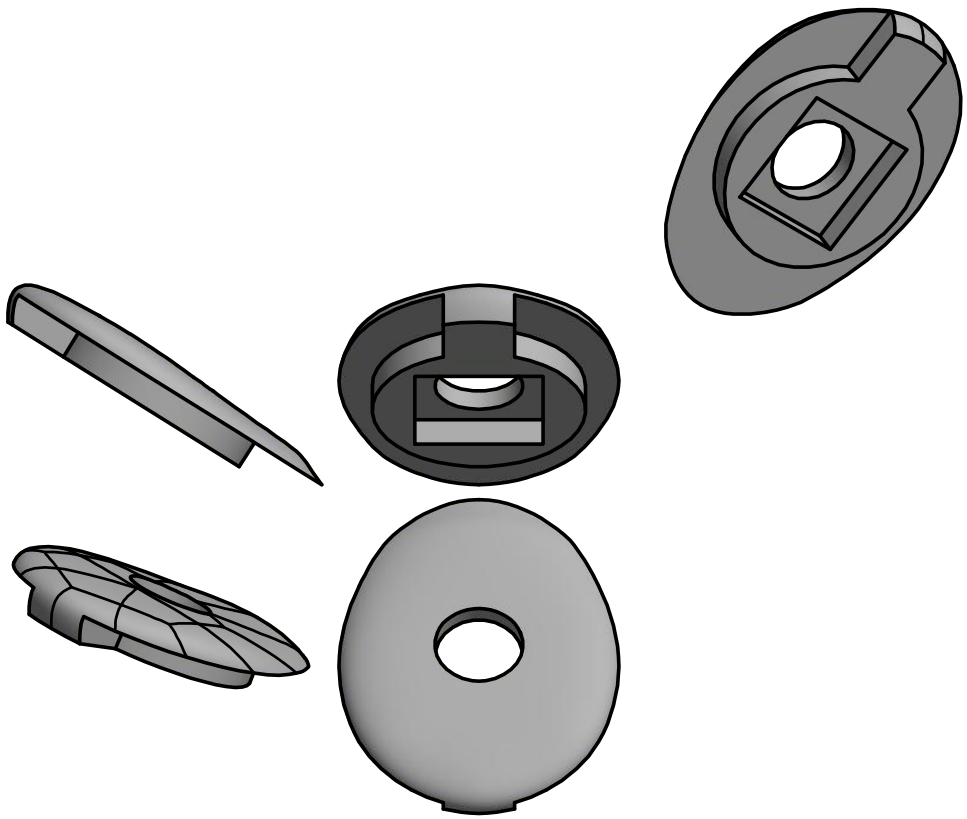
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE Cover		
QA				
MFG				
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 2	SHEET 9 OF 10	



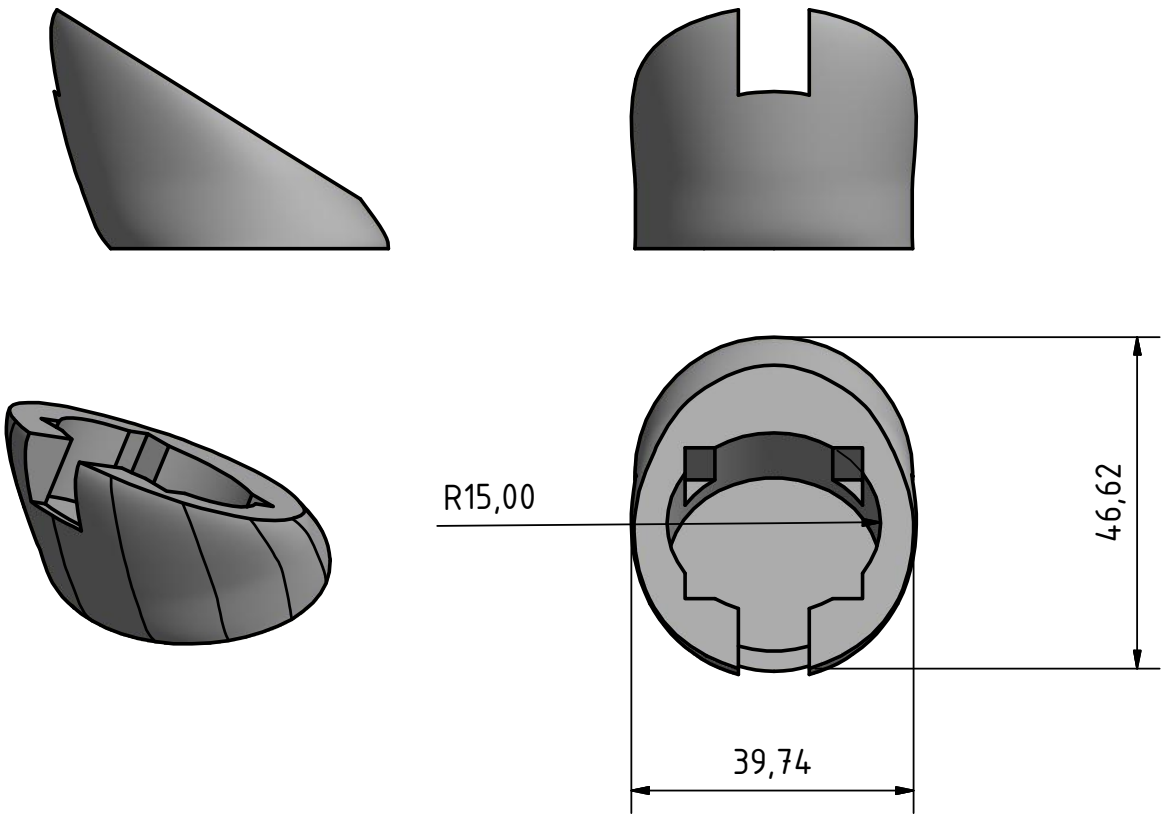




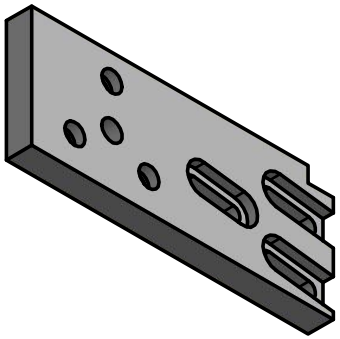
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA		Base		
MFG				
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 2	SHEET 10 OF 10	

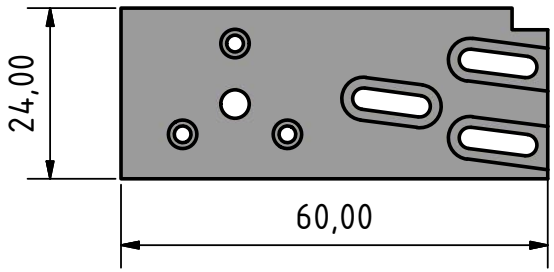


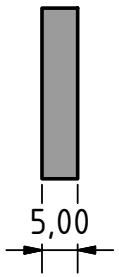
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA		Joystick Cap		
MFG				
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 1	SHEET 8 OF 10	



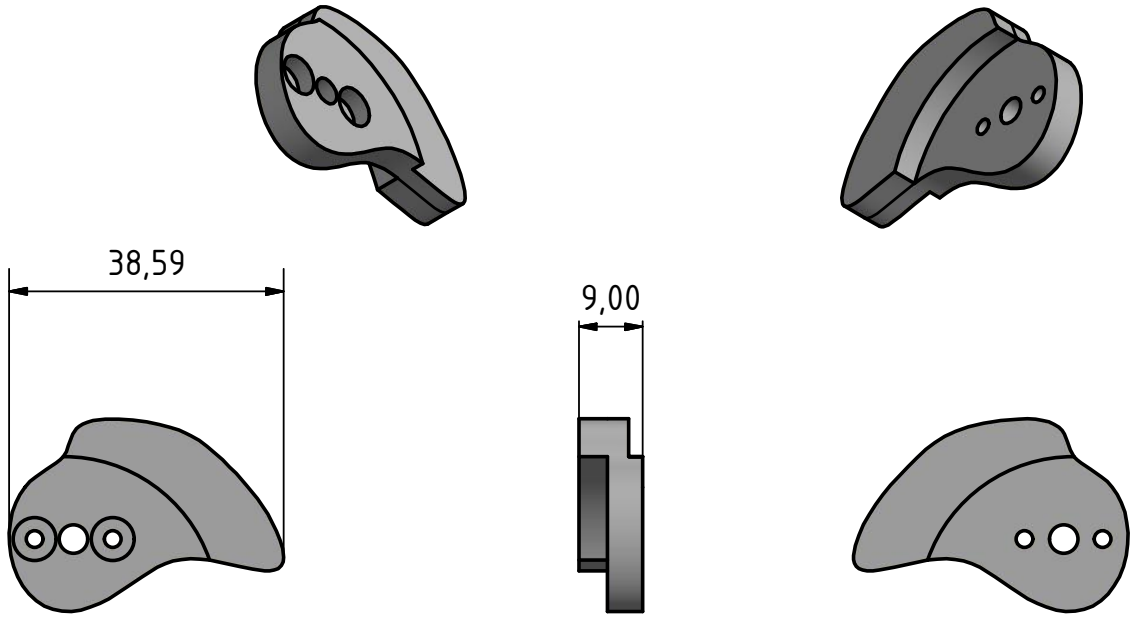
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA				
MFG		Joystick Mount		
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 1	SHEET 7 OF 10	



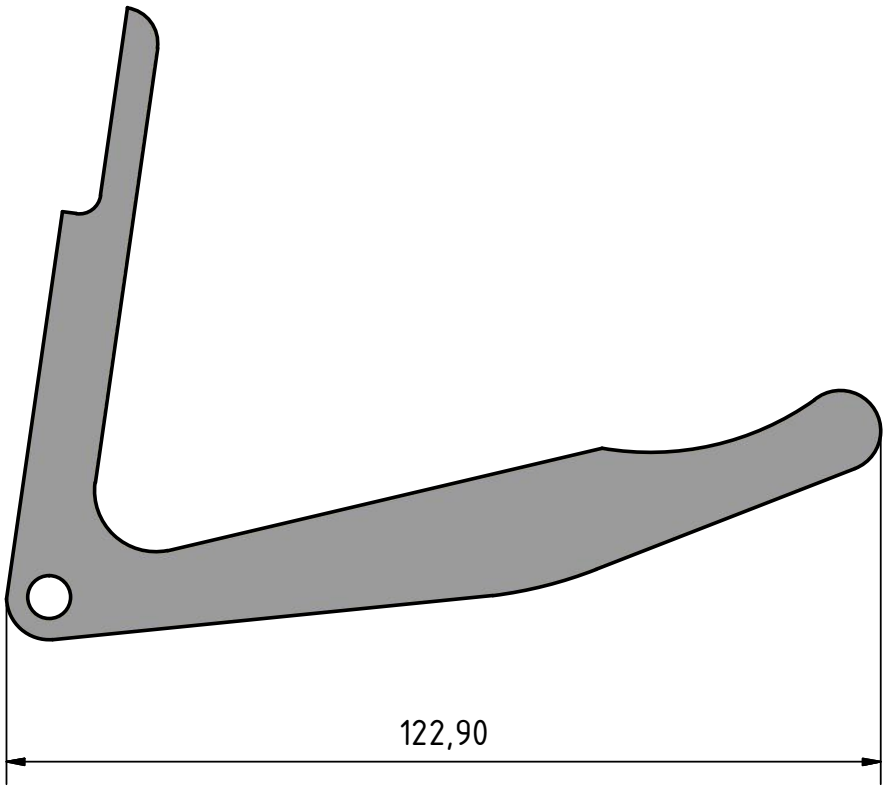


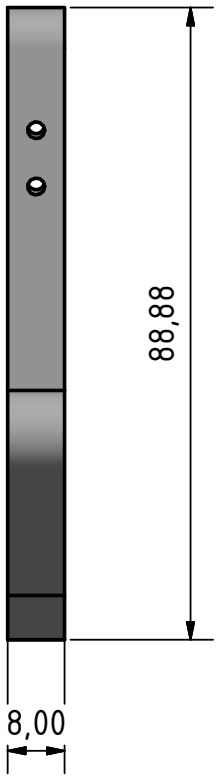


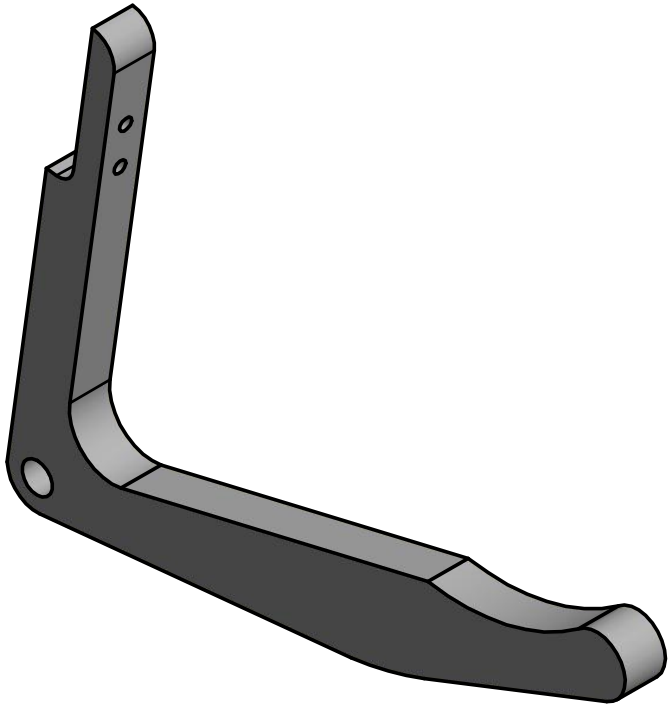
DRAWN oechslin	8/6/2018	EPFL
CHECKED		TITLE
QA		
MFG		Motor Plate
APPROVED		
		SIZE A4 / DWG NO Yoke Controller / REV
		SCALE 1 : 1 / SHEET 5 OF 10



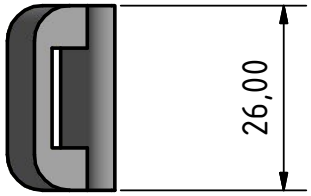
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA		Rotation Cam		
MFG				
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 1	SHEET 3 OF 10	

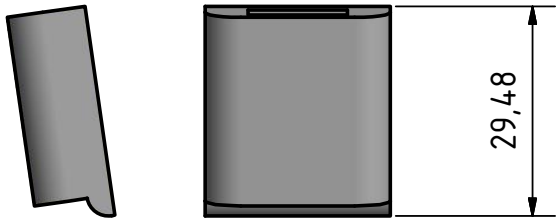


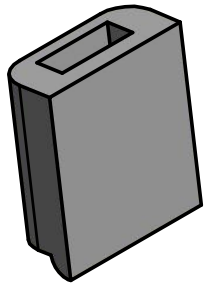




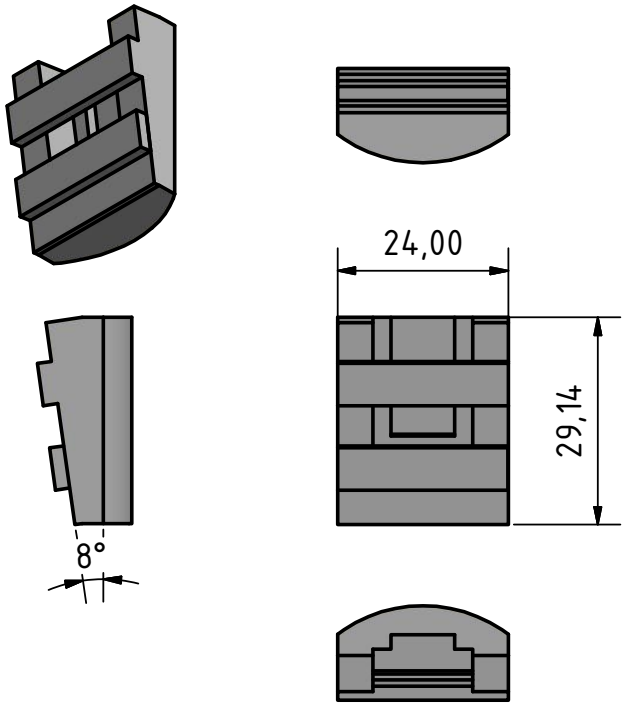
DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA				
MFG		Rotational L		
APPROVED				
		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 1	SHEET 2 OF 10	







DRAWN oechslin	8/6/2018	EPFL
CHECKED		TITLE
QA		
MFG		Spring Plate
APPROVED		
		SIZE A4 / DWG NO Yoke Controller / REV
		SCALE 1 : 1 / SHEET 1 OF 10



DRAWN oechslin	8/6/2018	EPFL		
CHECKED		TITLE		
QA		Palm Pad		
MFG				
APPROVED		SIZE A4	DWG NO Yoke Controller	REV
		SCALE 1 : 1	SHEET 4 OF 10	

FAULHABER

Datasheet, Motor
DC-Gearmotors

100 mNm

Precious Metal Commutation
with integrated Encoder

For combination with
Drive Electronics:
Speed Controller

Series 2619 ... SR ... IE2-16

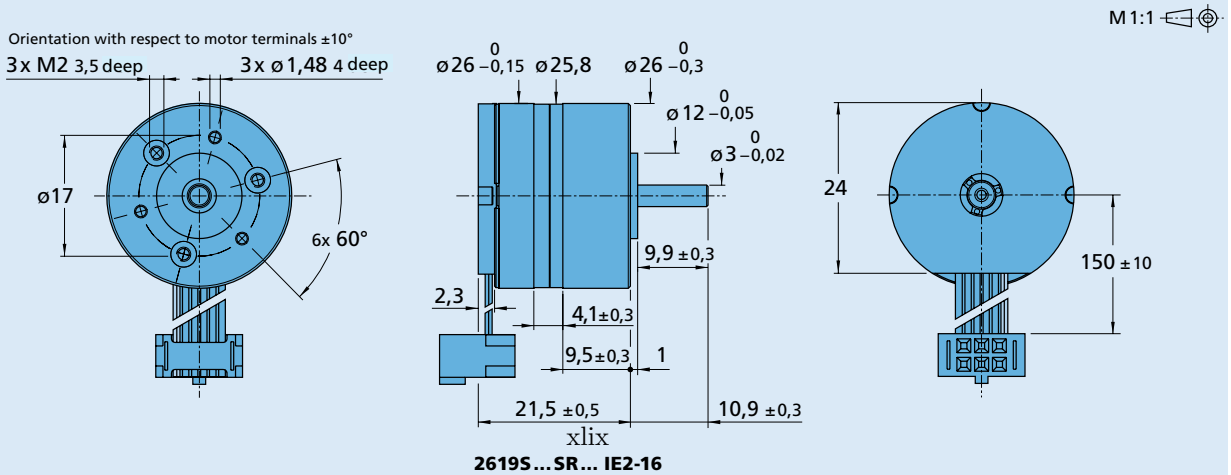
Values at 22°C and nominal voltage	2619 S		006 SR	012 SR	024 SR	IE2-16
Nominal voltage	U_N		6	12	24	Volt
Terminal resistance	R		8	31,2	118,6	Ω
No-load speed (motor)	n_o		6 700	6 900	7 200	min^{-1}
Speed constant	k_n		1 130	582	304	min^{-1}/V
Back-EMF constant	k_E		0,884	1,72	3,29	mV/min^{-1}
Torque constant	k_M		8,44	16,4	31,4	mNm/A
Current constant	k_I		0,118	0,061	0,032	A/mNm
Slope of n-M curve	$\Delta n/\Delta M$		1 060	1 090	1 110	min^{-1}/mNm
Rotor inductance	L		420	1 600	5 800	µH
Rotor inertia	J		0,68	0,68	0,68	gcm^2

Housing material		plastic		
Geartrain material		metal		
Backlash, at no-load	≤	4		°
Bearings on output shaft		brass / ceramic bearings	ball bearings, preloaded	
Shaft load max.:		(standard)	(optional)	
– radial (5 mm from mounting face)	≤	3,5	10,5	N
– axial	≤	2	5	N
Shaft press fit force, max.	≤	10	10	N
Shaft play:				
– radial (5 mm from mounting face)	≤	0,07	0,03	mm
– axial	≤	0,25	0	mm
Operating temperature range		0 ... + 70		°C

Specifications

reduction ratio (rounded)	output speed up to n_{max} min^{-1}	weight with motor g	output torque continuous operation M_{max} mNm	output torque intermittent operation M_{max} mNm	direction of rotation (reversible)	efficiency %
8 : 1	635	25	9	30	=	81
22 : 1	223	26	23	75	≠	73
33 : 1	151	26	30	100	=	66
112 : 1	44	27	93	180	≠	59
207 : 1	24	27	100	180	=	53
361 : 1	14	27	100	180	=	53
814 : 1	6	28	100	180	=	43
1 257 : 1	4	29	100	180	=	43

Note: output speed at 5000 min^{-1} input speed. Based on motor 2607 ... SR.



2619S ... SR ... IE2-16

For notes on technical data and lifetime performance refer to "Technical Information".
Edition 2018

Integrated optical Encoder		IE2-16	
Lines per revolution	*N*	16	
Signal output, square wave		2	channels
Supply voltage	U_{DD}	3,2 ... 5,5	V DC
Current consumption, typical (U_{DD} = 5 V DC)	I_{DD}	typ. 8, max. 15	mA
Output current, max. allowable (at U_{out} < 1,5V)	I_{OUT}	5	mA
Pulse width [1]	*P*	180 ± 45	°e
Phase shift, channal A to B [1]	*Φ*	90 ± 45	°e
Signal rise/fall time, max. (C_{LOAD} = 50 pF)	*tr/tf*	2,5 / 0,3	µs
Frequency range [2], up to	*f*	4,5	kHz

[1] Ambient temperature 22°C (tested at 1kHz)
[2] Velocity (min^{-1}) = f (Hz) x 60/N

Features

In this version, the DC-Micromotors have an optical encoder with two output channels. A code wheel on the shaft is optically captured and further processed. At the encoder outputs, two 90° phase-shifted rectangular signals are available with 16 impulses per motor revolution.

The encoder is suitable for the monitoring and regulation of the speed and direction of rotation and for positioning the drive shaft.

The supply voltage for the encoder and the DC-Micromotor as well as the two channel output signals are interfaced through a ribbon cable with connector.

Full product description

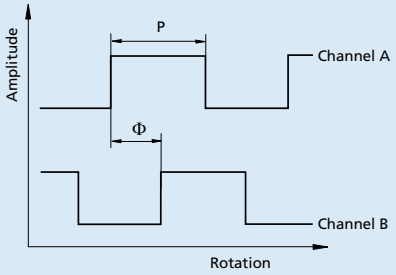
- Examples:

  2619S006SR 8:1 IE2-16

  2619S024SR 1257:1 IE2-16

Output signals / Circuit diagram / Connector information

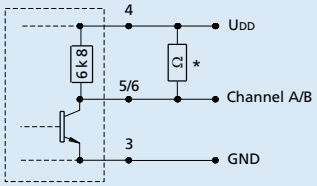
Output signals
with clockwise rotation as seen from the shaft end



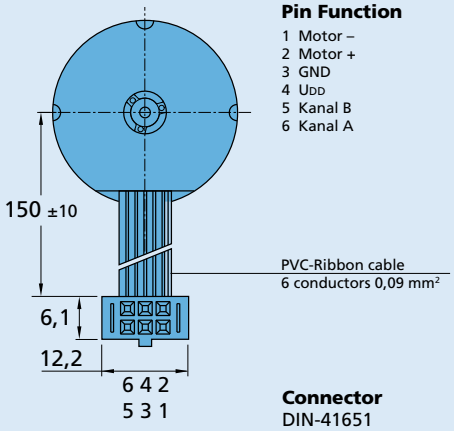
Admissible deviation of phase shift:

$$\Delta\Phi = \left| 90° - \frac{\Phi}{P} * 180° \right| \leq 45°$$

Output circuit



* An additional external pull-up resistor can be added to improve the rise time. Caution: I_{OUT} max. 5 mA must not be exceeded!



Pin Function

1 Motor –
2 Motor +
3 GND
4 UDD
5 Kanal B
6 Kanal A

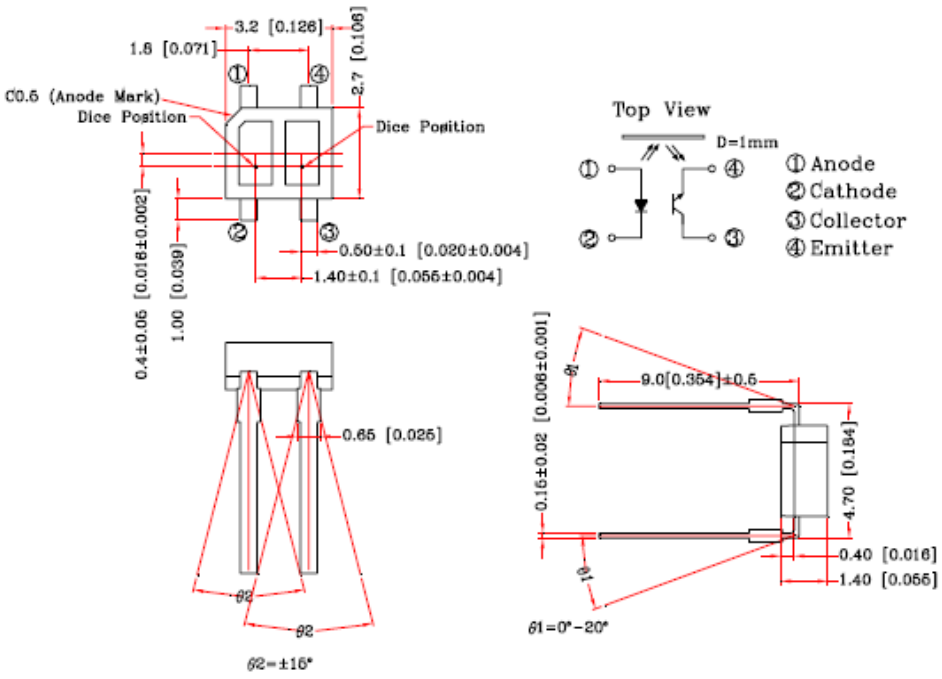
Connector
DIN-41651
grid 2,54 mm

GENIXTEK CORP.

Datasheet Photoreceptor DEVICE NO.:TPR-105F

This photo interrupters is non-contact switching and for direct pc board or dual-in-line socket mounting. It offers Fast switching speed. And this product doesn't contain restriction substance, comply ROHS standard.

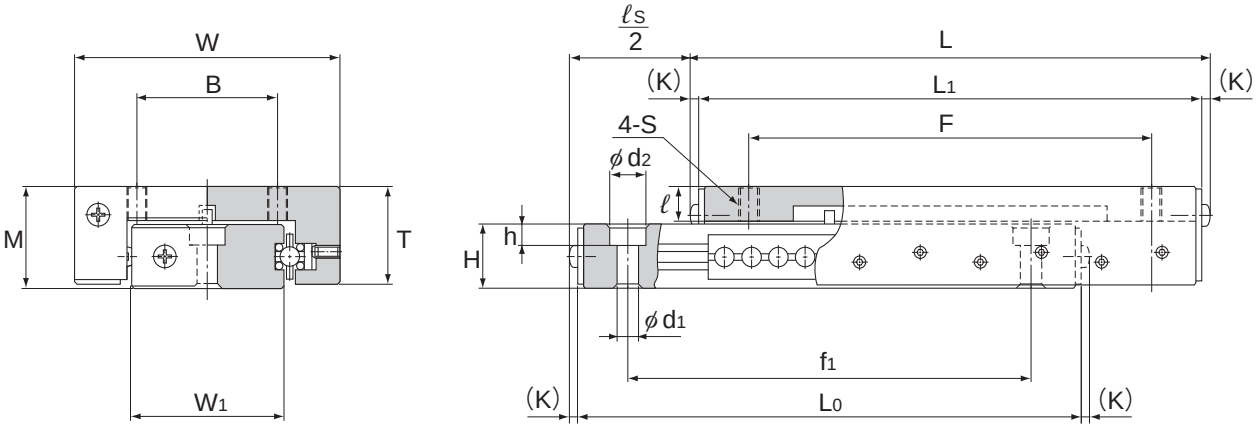
PACKAGE DIMENSIONS:



NOTE:

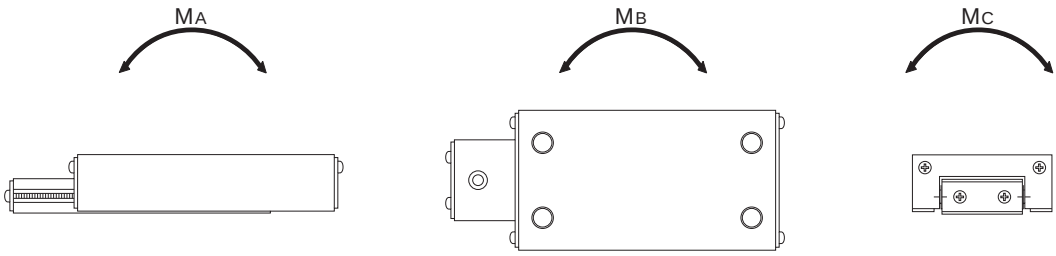
All dimensions are in millimeters
Tolerance is ±0.25 mm unless otherwise noted
Lead spacing is measured where the leads emerge from the package.

Model LS
Datasheet Guideway



Model No.	Slider dimensions									
	Max. Stroke ℓs	Height M ±0.25	Width W ±0.25	Length L	T	L_1	(K)	B	F	S×ℓ
LS 827	13	8	14.2	28.7	7.6	27	0.85	5.5	16	M2×3
LS 852	25	8	14.2	53.7	7.6	52	0.85	5.5	41	M2×3
LS 877	50	8	14.2	78.7	7.6	77	0.85	5.5	66	M2×3
LS 1027	13	10	19	28.7	9.2	27	0.85	8.5	16	M3×3.5
LS 1052	25	10	19	53.7	9.2	52	0.85	8.5	41	M3×3.5
LS 1077	50	10	19	78.7	9.2	77	0.85	8.5	66	M3×3.5

Datasheet Guideway



Unit: mm

	Base dimensions					Static permissible moment*		Basic load rating		Mass
	Width W_1	Height H	$d_1 \times d_2 \times h$	Length L_0	f_1	M_A, M_B N-m	M_C N-m	C N	C_0 N	g
	6.2	4.7	2.2×3.9×1.4	27	19	0.2	0.29	39.2	68.6	9
	6.2	4.7	2.2×3.9×1.4	52	35	0.49	0.39	68.6	118	15
	6.2	4.7	2.2×3.9×1.4	77	60	0.88	0.59	98	167	21
	9.6	6.2	3.3×6×3.1	27	19	0.29	0.59	58.8	108	13
	9.6	6.2	3.3×6×3.1	52	35	0.78	1.08	108	186	23
	9.6	6.2	3.3×6×3.1	77	60	1.47	1.57	157	275	34

Note) *M_A, M_B and M_C each indicate the permissible moment per LM system, as shown in the figure above.