KTH Mechatronics Advanced Course
MF2059
2020 FINAL REPORT

SOCKETSENSE HK

TEAM:
ELLEN ANDERSON, ALEJANDRO BERGSTRAND, DAVID DANELIA, SHUO FU, HANNA HERMANSSON, JAKOB JANSSON, CHARIKLIA PANTELI, MIKAEL SJÖBERG, AXEL ZEDIGH

[13-12-2020]

Abstract

This HK project is part of a European Union project, called SocketSense. The goal of this HK project was to develop a test rig for above-knee prosthetic sockets which would allow realistic pressure readings of human movement. To solve this a Stewart platform was built. The platform could recreate the corresponding forces and torques inside the socket-to-stump interface, given prerecorded motion data of an amputee's gait cycle. If pressure sensors were to be added in the socket, multiple tests can be made without the need of the patient. Which could be used in order to ensure the best possible fit of the prosthetic and avoid discomfort. Position control was successfully implemented in the final test rig while the force control implementation was lacking. The stump made for testing the rig also proved to be inadequate resulting in further difficulty verifying the force control of the rig.

Sammanfattning

Detta HK-projekt är en del av ett EU-projektet SocketSense. Målet med HK-projektet var att utveckla en testrigg för lårbensproteser som skulle möjliggöra realistiska tryckavläsningar av benets rörelse. För att lösa detta byggdes en Stewart-plattform. Plattformen kan återskapa motsvarande krafter och vridmoment som verkar mellan stumpen och protesen, givet förinspelade rörelsesdata från en amputerad person. Om trycksensorer skulle läggas till i protesen kan flera tester göras utan patientens närvaro. Vilket skulle kunna användas för att säkerställa bästa möjliga passform för protesen och undvika obehag. Positionsreglering lyckades implementeras i den slutliga testriggen medan kraftregleringen inte implemtenrades fullt ut. Stumpen som gjordes för att testa riggen visade sig också vara otillräcklig vilket resulterade i ytterligare svårigheter att verifiera kraftregleringen.

Acknowledgements

We would like to thank our stakeholders Dejiu Chen, Fredrik Asplund and SocketSense for giving us the opportunity to work on this project. We would also like to thank Lars Hässler for helping us and supervising the project. In addition we would like to thank all the people who helped us during this project by giving us guidance and insight information on how to best control the platform. Lastly, we want to thank the Mechatronics department for giving an opportunity to their students to work on a project like this and cooperate with a company, in hope to evolve as engineers, get to know the industry and develop a useful product.

Contents

List of Figures

List of Tables

Acronyms

AC Alternative Current.

AKLS Above Knee Leg Stump.

CAD Computer Aided Design.

COM Compare Output Mode.

CS Clock Select.

CTC Clear Timer on Compare Match.

DC Direct Current.

DOF Degrees of Freedom.

EMG Electromyograph.

EU European Union.

FEA Finite Element Analysis.

FF Foot Flat.

HK Högre Kurs.

HO Heel-Off.

HS Heel Strike.

ISR Interrupt Service Routine.

KTH Kungliga Tekniska Högskolan.

MEMS Micro Electro-Mechanical Systems.

MIMO Multiple Input-Multiple Output.

MS Mid-Stance.

MSW Mid-Swing.

OCIE Output Compare Interrupt Enable.

OCR Output Compare Registers.

PM Parallel Manipulator.

PWM Pulse with Modulation.

QTSS Quantum Technology SuperSensors.

SM Serial Manipulator.

SOTA State of the Art.

SP Stewart Platform.

TCCR Timer/Counter Control Register.

TF Transfemoral: above-knee amputee.

TIMSK Timer/Counter Interrupt Mask.

TO Toe-Off.

WGM Wave Generate Mode.

Chapter 1

Introduction

1.1 Background

Today over 30 million people all over the world live with a need for prosthetic or orthotic device [1]. The loss of a limb has a major impact on the overall quality of life of amputees. By using an prosthetic limb, the functional mobility can be restored and both the physical and mental health increased. However, this places certain demands on the fit of the prosthetic socket and the stump so as not to cause any additional pain or other complications, e.g., skin problems on the stump [2][3]. Several surveys have shown the majority of issues experienced by prosthetic users are caused by the prosthetic socket, where an uncomfortable socket has been the main complaint [4][5][6][7][8].

KTH is part of the EU funded project SocketSense aimed to develop a novel pressure sensing technology for lower limb amputees. This will enable manufacturing of comfortable sockets tailored to each individual patients needs. The technology consists of real-time monitoring of the pressure between the socket and stump. This is made possible by using advanced embedded sensors in the socket for data collection. At this stage the challenge is to find a method to efficiently make the acquired pressure readings available and analysable to clinicians.

The goal of the SocketSense HK project is to develop a test rig which can be used to obtain the sensor readings in an efficient and consistent way. The purpose of the test rig is to simulate realistic forces inside the socket-to-stump interface, by using motion data captured from a patient. In this way, the patient only need to preform the motion once and the test rig can preform the motion with different sockets to find the perfect fit for each individual patient.

1.2 Project Description

The goal of the SocketSense HK project is to build a test rig for above-knee prosthetic sockets which allows realistic pressure readings for a gait cycle. The test rig should consist of a platform with 6 Degrees of Freedom (DOF), with the above-knee leg stump mounted on it. The stump should also be fitted inside a socket, which is rigidly mounted on the test rig. The platform shall be able to recreate the corresponding forces and torques inside the socket-to-stump interface given by prerecorded data from OpenSim of an amputee executing a full gait cycle.

The test rig should be designed to enable future upgrades such as adding a second platform with 6 DOF and to also simulate other scenarios like jumping, running, and climbing, and also be able to change the stump and socket.

1.3 Requirements

The stakeholder requirements for the test rig are given below and the technical requirements can be seen in Appendix B.

1.3.1 Stakeholder Requirements

The following requirements were provided from the SocketSense stakeholders:

- An Above Knee Leg Stump (AKLS) part consisting of a platform with 6 Degrees of Freedom (DOF).
- A simulation-to-rig interface, compatible with OpenSim and MATLAB.
- Use an ischial socket with pin lock for testing.
- The socket and stump should be interchangeable.
- The test rig should be able to recreate forces and torques for a gait cycle
- The test rig should be designed assuring the possibility for expanding of the rig with an additional 6 DOF platform for the socket, and also investigating other scenarios like jumping, running, and climbing.
- The rig should be able to make motions and all the motors should be able to move at the same time.

1.4 Delimitations

Due to the allocated time for the project it was decided, together with the stakeholder, that the test rig will only consist of one platform with 6 DOF. The walking

scenario was investigated in this project. Furthermore, it was instead set as an requirement that the test rig should be expandable in future work in these two areas.

This project does not look into how the prerecording or data gathering in OpenSim of the amputee works. The data used in the project are from the same patient.

This project has not looked into the socket design nor how to measure and receive the pressure data in the socket-to-stump interface.

The test rig does not apply the corresponding forces and torques in the same time frame as the original gait cycle. It is instead moving in a slower rate to minimize vibrations of the test rig.

Chapter 2

Literature Review and State of the Art

2.1 Biomechanics

Biomechanics is the study of mechanical motion and its underlying forces in biological systems. This project pertains to the biomechanics of human movement.

2.1.1 Gait Cycle

The gait cycle is the physiological term for a single step during walking. It is usually described by 6 phases and those are visualized in Figure 2.1:

1. Heel Strike (HS)
2. Foot Flat (FF)
3. Mid-Stance (MS)
4. Heel-Off (HO)
5. Toe-Off (TO)
6. Mid-Swing (MSW)

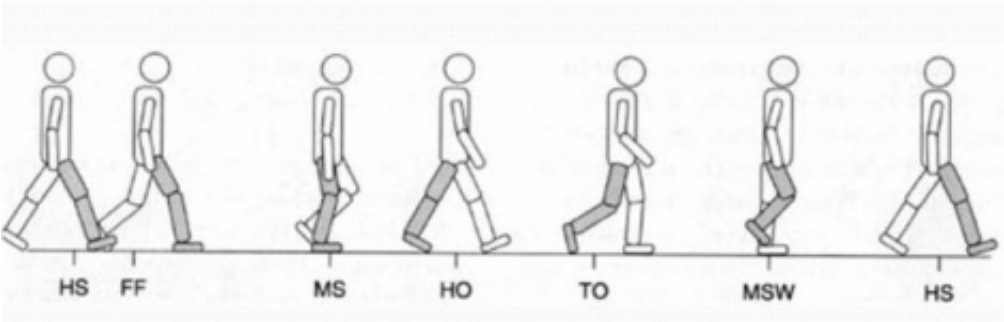


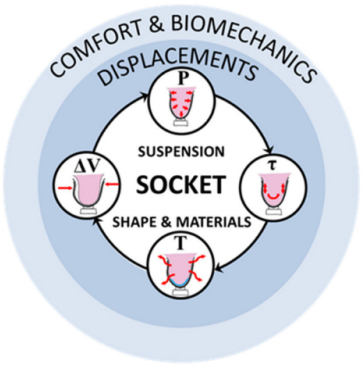
Figure 2.1: Different phases during one gait cycle [9].

Moreover, it is also common to use a percentage of the gait cycle, were 0 % is at the start and 100 % is at the end. The gait cycle is the motion that the rig tries to simulate. It is the forces and torques from the gait cycle which are used as references when controlling the rig and the goal to be achieved.

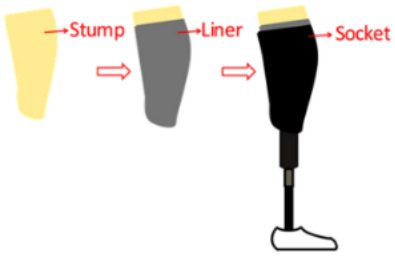
2.2 Lower Limb Transfemoral Sockets

One of the most important challenges in the field of prosthesis is the human-machine interface, namely the socket. A large number of amputees still reject prostheses or points, out of a low satisfaction level, due to a sub-optimal interaction between the socket and the residual limb tissues. A suitable socket has to ensure efficient fitting, appropriate load transmission, stability and control. It often constitutes a key factor for the success or failure of the prosthesis itself [10]. Figure 2.2a is a schematic representation of the main factors affecting the stump-socket interface and their interplay. Suspension in this case, is the system used to guarantee the adhesion of the residual limb to the socket, P stands for pressure, τ is shear stress, T is temperature, ΔV is volume fluctuations and displacements is relative movements between the stump and the socket. A liner is normally used to between the stump and socket in order to increase the stability of the stump and the socket, see Figure 2.2b.

The choice of suspension system is one of the key factors that influence user satisfaction. Several suspensions are available at present, see Figure 2.3. The socket provided by the stakeholders is using a pin-lock.



(a) Schematic representation of the main factors affecting the stump-socket interface.



(b) Socket with liner.

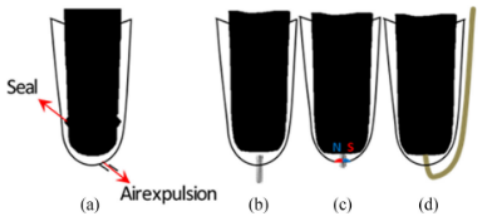


Figure 2.3: Suspension systems, from left to right: seal-in (a), pin-lock (b), magnetic-lock (c), Lanyard-strap (d) [10].

2.2.1 Ischial Containment Socket

For transfemoral (above knee) sockets there is mainly two variants in use today: the quadrilateral and the ischial containment sockets, which was used for this project.

In an ischial containment sockets the weight bearing takes place all over the surface of the stump without localizing one specific point; hence, generating more comfort, better control over the prosthesis and security for the user. The ischial tuberosity does not suffer from direct, complete and permanent weight bearing. The principal peculiarity of this design, apart from the exact volume determination, is the medial wall/border of the socket that contains the ischial ramus. The suspension is provided by negative pressure (suction) generated by adequate fitting of the socket over the stump. For the moment, this socket is in favor worldwide,replacing the quadrilateral one [11]. The socket with a pin-lock is seen in Figure 2.4.

Figure 2.4: Ischial Containment Socket [11].

2.2.2 Pressure Distribution In Socket

The pressure distribution in the socket is of great importance for the mobility, function and acceptance of the device. Pressure distribution over a greater surface, diminishes the load and provides more comfort during the use of the prosthesis.

Socket designs should allow forces to be distributed over as large a surface area as possible and should be applied as evenly as possible over pressure tolerant areas. See Figure 2.5 for highlighted pressure sensitive areas. Multiple fittings are at times necessary to assure the best possible design with a comfortable and effective fit. It is in this step of the process, that the test rig could potentially contribute with valuable data to the clinician.

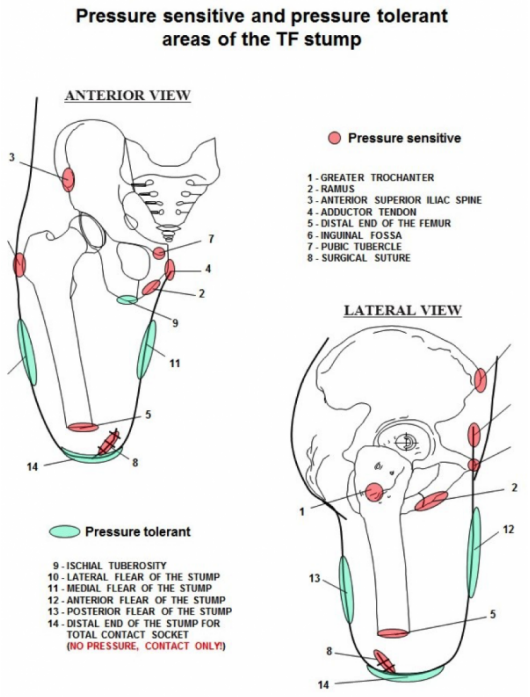


Figure 2.5: Pressure sensitive areas of TF stump. [11]

2.3 Sensors

2.3.1 Quantum Technology SuperSensors

One of the partners of the SocketSense EU project is company Quantum Technology SuperSensors, who will provide a product of theirs called Quantum Technology SuperSensors (QTSS), which is a new type of pressure sensors. QTSS are according to its makers environmentally friendly, thin, lightweight, economical, and robust sensors. When the sensor is under pressure its materials change from insulator to conductor, proportional to the applied forces. The sensors can come in various shapes and forms to be applied in many different fields. They can for example be used on textile, plastic, paper, metal and more. Furthermore, they can read multiple pressure signals simultaneously as well as their positions [12].

A prototype version of the sensors have been tested as part of a master thesis at

KTH. They conclude that the QTSS could be hard to calibrate and that they show uncertainty in repeat-ability. When it comes to static drift, their testing showed a drift from 29.4 % after one minute to 109.5 % after ten minutes on soft surfaces. For hard surfaces it was 4.4 % after one minute to 11.1 % after ten minutes. Both tests were made with a pressure of 185 kPa. The sensors displayed similar behavior when it comes to dynamic drift. The difference in sensor output between loading and unloading, known as hysteresis, was also tested and indicated a value of 63.1 % in hysteresis. It was also concluded that the fabric based QTSS could cause false negatives as the output when loading and unloading differed [13]. A picture of the QTSS can be seen in Figure 2.6.

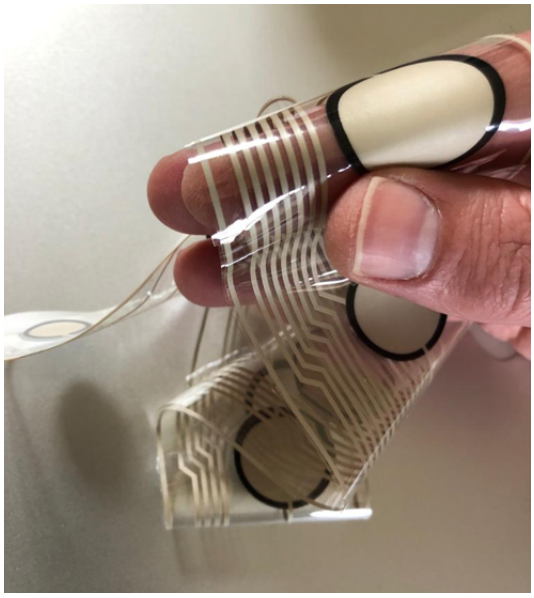


Figure 2.6: QTSS (the circular sensel pads), the image was sent from the stakeholder.

There is no commercial QTSS shear pressure sensor sensel pad available at the moment but there is development being done.

2.3.2 Load Cell

A load cell is a metal object with a strain gauge attached and is regularly used in the industry to measure force. Strain gauges are sensors which can measure the strain of a material. The strain gauge has a thin piece of metal on it that will expand or contract when force is applied on the object. The deformation of the metal will lead to a change of electrical resistance which can be measured. The change in voltage output of the sensor is linear and can be used to calculate the applied force [14].

2.4 Simulation

2.4.1 OpenSim

OpenSim is a software that is used for various biomechanical purposes. The program can analyze the data from a motion track rig and calculate the forces and the position of the skeleton joints. With imported recordings from sensors such as ground reaction plates, Electromyograph (EMG)'s or pressure sensors, the program can calculate the forces and moments applied on the joints of the skeleton during the motion. It can also simulate the muscle force, muscle activation, the stretching of ligaments, muscle and ligament defects and much more, making it a powerful tool for a physical therapist.

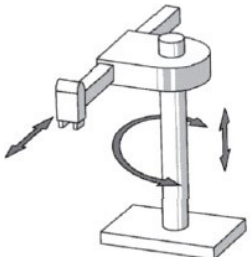
2.4.2 Simulink

The motion and force data generated by the OpenSim model can be exported to MATLAB software's modeling and simulating environment. If a complex system is wished to be modelled and simulated a Simscape Multibody model can be used. The model can either be created by importing an existing 3D model or make a new one by choosing electrical and mechanical components. This software enables simulation of the whole model in a realistic way and several input and output signals from the model can be recorded by the software to ensure good insight and control of the model.

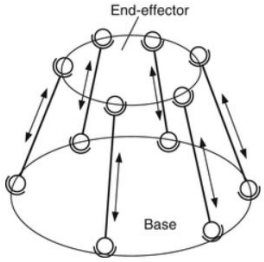
2.5 Robot Manipulators

Robot manipulators can be divided into three different kinematic structures; serial, parallel, and hybrid, which has their own advantages and disadvantages depending on the application.

Serial manipulators (SM) have a so-called open-loop kinematic structure, consisting of one or several links connected by joints, see Figure 2.7a. This structure ensures one of the SMs main advantages, they possess a large work-space. The opposite applies to parallel manipulators (PM) with their closed-loop kinematic structure with several independent limbs connected between the moving platform and the base, seen in Figure 2.7b. Due to the possibility of collision between the separate limbs this causes the work-space to become much smaller. PMs also suffer from singularities in the work-space. At these points the manipulators might lose their stiffness which cause them to become shaky in their movements and inaccurate in their position [15].



(a) A serial manipulator [15].



(b) A parallel manipulator [16].

Figure 2.7: The two different manipulator kinematic structures.

Nevertheless, PMs are generally seen as more accurate and have higher payload to weight ratio. This is due to the error propagation and cantilever structure of the SM, which tend to make the limb bend at high load and vibrate at high speed [17].

The motors can be mounted close to the base reducing the moving mass and lower the inertia. This enables higher acceleration and speed and overall a better dynamic performance [16].

When looking at controlling the manipulators, different mathematical methods can be used. The inverse kinematic method was used to obtain the coordinates of the joints, given the position and orientation of the end of the kinematic chain. This method is easier to use on PM than their counterpart. Forward kinematics is another mathematical method obtaining the position and orientation of the claw or platform using the known coordinates for the joints. This method is instead easier to use on the SM [15].

The hybrid kinematic structure is a combination of the parallel and serial manipulators. It therefore has the advantages of both, giving a large work-space with high rigidity and loading capacity. However, there are some issues concerning the complexity of the design of the hybrid manipulator and motion control, that seem to be harder to achieve [18].

2.5.1 Stewart Platform

SPs' are often used as typical examples to explain PM. The general SP consists of a fixed base connected by six legs to a moving platform [19]. The actuators on the legs may differ depending on the chosen design, but the most common ones are:

- Linear actuators: where the motors on the base frame convert the rotational to a linear motion, which causes the piston in the cylindrical leg to extend or retract, see Figure 2.8.

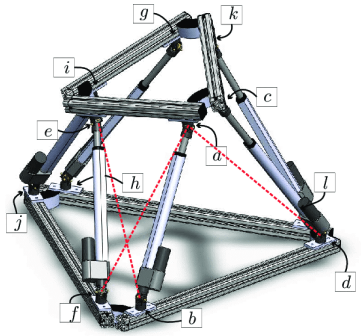


Figure 2.8: Stewart Platform: Linear actuators [20].

- Electrical motors: where the motors are rotating an arm, on which the legs of the frame are connected, see Figure 2.9.

Figure 2.9: Stewart Platform: Servo motors [21].

The concept of the SP was first introduced in 1949, and today there are over 1400 research articles about manipulators with similar structures concerning their workspace, kinematic analysis, singularities and design. They work as transducers and force feedback devices, or used as models for locomotion and gaits, and much more

[22]. The application of the platforms have been varied, being used in everything from flight simulators [19] to precision surgeries [23][24][25].

2.5.2 Cable-Driven Parallel Manipulators

The limited workspaces of the general PM have led to the development of cable-driven parallel manipulators. In these manipulators the rigid links and actuators have been replaced by flexible cables and cable drivers, leading to a larger workspace [15][26], seen in Figure 2.10. They also have higher payload-to-weight ratio and smaller inertia than the rigid-link manipulators, enabling higher acceleration and speed [26].

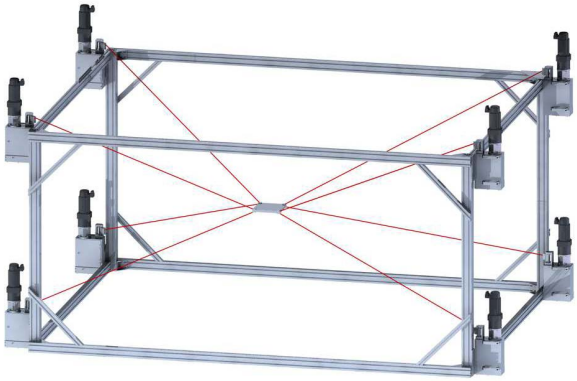


Figure 2.10: A 6 DOF cable-driven parallel manipulator [27].

However, there are some challenges concerning the design of this manipulators. The stiffness and accuracy depend highly on the tension and properties of the cables. This causes vibrations and sagging long cables to adversely affect the position accuracy [15][26].

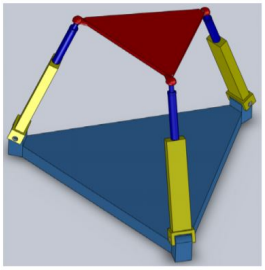
2.5.3 Micro-Manipulators

Since Micro Electro-Mechanical Systems (MEMS) play an essential role in many areas such as transportation, health care, defense systems, and automated manufacturing, the demand for micro positioning mechanisms have increased. As the name suggests the displacement of these manipulators are in the range of micro-millimeters.

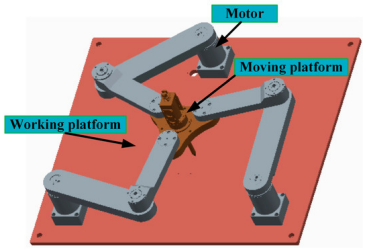
Most of the 6 DOF micro-manipulators use a similar configuration as the SP [28]. But in recent years many of the new manipulator designs use the hybrid structure because of its advantages.

These hybrid designs often consist of an upper stage parallel structure, and a lower stage parallel structure, mounted in series. For example, in Figure 2.11a the upper

stage is a 3-RPS parallel structure, and the lower stage a 3-RRR parallel structure, in Figure 2.11b. These are then creating a 6 DOF hybrid structure, see Figure 2.11c [29]. There are other micro-manipulators where the upper stage and lower stage are other kinds of parallel structures or they are switched as it is shown in Figure 2.11 [30].



(a) The 3-RPS upper stage parallel manipulator [31].



(b) The 3-RRR lower stage parallel manipulator [32].

(c) The complete structure of the micro-manipulator [29].

Figure 2.11: A configuration of a micro-manipulator.

The most essential parts of the design of these manipulators are the flexible hinges. They are used instead of joints to remove friction, backlash, and low displacement resolution [29].

2.6 Actuators

An actuator is a component that can move a system and is part of the control loop. In order to make an actuator work, a control signal and a source of electrical energy is required. There are many kinds of actuators, including hydraulic actuators, pneumatic actuators, thermal actuators, electric actuators, etc. [33].

In this project, considering the convenience of the accessibility of power source, electric actuators were taken into account. There are two kinds of actuators that

were considered. One is the kind that can convert rotational to linear motion and the other one can generate rotational motion. The typical examples of these two kinds of actuators are linear actuators and electric motors.

The restrictions of choosing actuators include the price, the mechanical structure, precision, efficiency, etc. The ideal actuator will have the characteristics of high precision, high efficiency and low price. However, it is difficult to make an actuator have all the characteristics that was desirable. Thus, choosing the actuators is a process of trade-off.

2.6.1 Linear Actuators

The linear actuators can convert rotational motions to linear motions of an Alternative Current (AC) or Direct Current (DC) motor. The main parts of linear actuators contain motors, screws and nuts. The motors can transmit torque to the screw via gears. Then the screw can drive the nut, or a pair of nuts, to do an axial motion. The structure of a linear actuator is shown in Figure 2.12 [34].

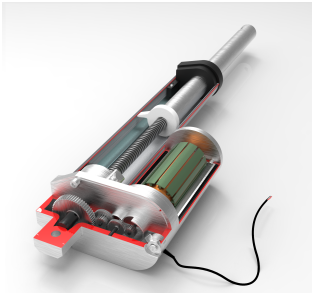


Figure 2.12: Inside structure of a linear actuator.

Sensors can be attached to a linear actuator, which makes it possible for feedback control. The feedback signal can be the elongation of the linear actuator or the angular position of the motor in the linear actuator. They have the advantage of small installation space, high precision and full synchronization. A linear actuator is directly driven by a motor, without other mechanical structure or source. The appearance of a linear actuator is shown in Figure 2.13 [35].

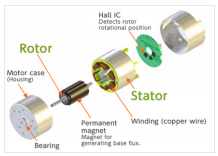
Figure 2.13: Linear actuator.

2.6.2 Electric Motors

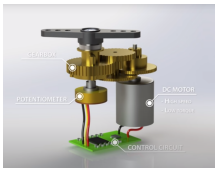
A electric motor is a kind of motor that can convert electric energy into mechanical rotational motion [36]. There are many kinds of electric motors.

A DC motor is a electric motor that is driven by direct current. There are shunt-wound, series-wound, compound-wound and permanent magnet DC motors [37]. DC motors have a wide range of speed control [38]. Based on the structure, DC motors can be divided into brushed DC motors and brushless DC motors. Comparing with brushed DC motors, brushless DC motors have high efficiency, high output power and long life span. However, brushless DC motors are more expensive than brushed DC motors and the structure is more complicated. Furthermore, a brushed DC motor's output speed is linear to the applied voltage [39]. The construction of a DC brushless motor is shown in Figure 2.14a [40]. Unlike DC motors, whose control signals are analog, stepper motors and servo motors have digital control signals. Steppers control the angular position through the number of pulse. Servo motors are close-loop motor with encoders to provide feedback control. A servo motor contains DC motor, gear box, encoder and control circuit. The controller will send an error signal to the motor if the signal from encoder provides that the angular position is wrong. The construction of stepper motors and servo motors are shown in Figure 2.14b 2.14c [41][42][43].

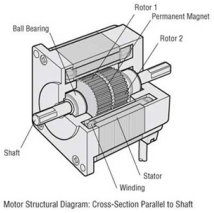
Different kinds of motors have different characteristics. It is still a trade-off process to choose the right kind of motors. Furthermore, since linear actuators and electric motors have different ways, there will be differences in accuracy, velocity, acceleration and efficiency, etc. The final decision of the kind of actuator is based on later researches and experiments.



(a) DC brushless motor [44].



(b) Servo motor [45].



(c) Stepper motor [46].

Figure 2.14: Motor types.

2.7 Existing Test Rigs

2.7.1 SocketSense HK Project 2019

A similar test-rig to the one that this project aimed to construct, has already been built, with the key difference being that it moved with 1 DOF. The project used QTSS and demonstrated the importance of testing and validating the sensors. A load cell was used to measure the sensor data but it was also considered using a load pin or scale. They implemented a filter in their software to correct the load cells output. The data was collected using a Raspberry Pi connected to a Influx database via TCP/IP and HTTP. An ABB robot was also considered but was deemed incapable of handling the required forces [47]. A picture of their final prototype can be seen in Figure 2.15.

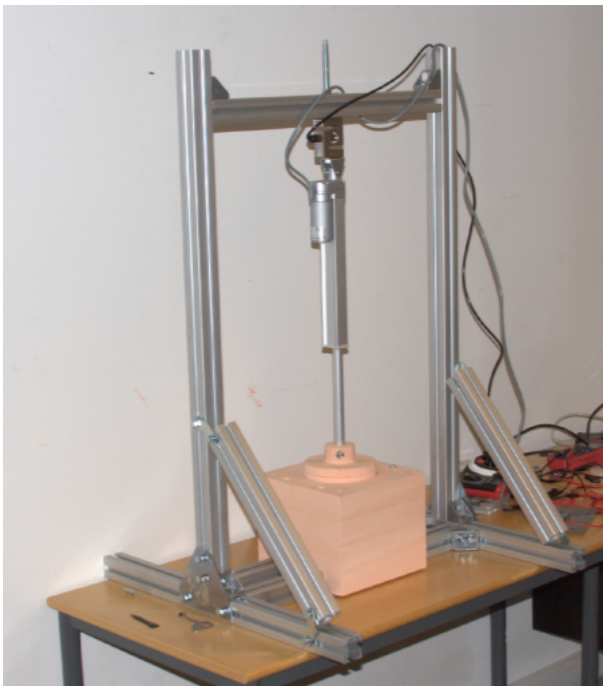


Figure 2.15: A test bench with 1 DOF constructed in a similar project 2019 [47].

2.7.2 Joint Simulators

There exist some machines that simulate the forces and movements of joints. Two examples are the Bionix Knee Wear Simulator and AMTI's VIVO which is shown in Figure 2.16. AMTI's VIVO provides 6 DOF for a single joint and can simulate forces in all directions. It uses six servo-hydraulic actuators to simulate the movement and forces. To measure the contact forces and moments, a six-axis force sensor is used and imperfections in the measurements are corrected by the control system. It also

has precision displacement sensors to give position feedback to the control system [48].

Figure 2.16: AMTI machine [48].

Chapter 3

Methodology

3.1 Project Organization

The team have worked with an agile workflow, inspired by the Scrum Framework [49]. The project have been divided into smaller subgroups, listed below.

- **Software:** Consisted of the parts regarding the interface for the OpenSim, MATLAB and Simulink data. The connection and communication between the Arduino and computer were included in the subgroup.
- **Prototype:** This area encompassed the physical implementation and the building of the real-life model. This included 3D modeling, buying, or building the necessary parts for the whole test rig, such as metal parts, plastic parts and the stump.
- **Testing:** All different tests that were conducted throughout the project. Tests were done for the components, software and the whole model.
- **Control:** This area encompassed all the work with the model and control of the rig. The design of the controller and how it was implemented.
- **Administrative:** The work connected to the report, presentation and planning.

The amount of people that worked in the groups varied over time. In addition, rotations in the groups occurred every third week. The duration is called a sprint. After every finished sprint, a meeting was held and the tasks were evaluated. The upcoming sprint were also planned. The constellation of the subgroups were to be altered every sprint so that the whole team acquired experience in every area.

The tools used during the project were Trello, Google Drive, Github and Discord for communications. Trello was used to manage tasks. Each task had three different

states; To Do, Doing and Done. Everything that needed to be done eventually, was categorized as 'To Do' in the Product Backlog. The tasks being worked on were categorized as 'Doing' and the completed tasks were marked as 'Done'. Google Drive was used for organizing files, assignments etc. Github was used for versioning of code in the project. And finally, Discord was used for communications within the group, where each subgroup had a channel for their related work.

3.2 Planning

Throughout the project an Agile project managing approach were used, in regards to Scrum [50]. But in addition, not commonly used at the same time as Scrum a GANTT-chart was made for the fall, see appendix A. It had key dates and estimations on how much time to spend each part of the project.

Every week a group meeting were held, where progress where discussed and the upcoming tasks. In the meeting there was a secretary who wrote down in the meeting template things that where discussed and decided, in addition, there were one chairman of each meeting who made sure the meeting stayed on topic and that protocol was followed. The roles where changed every meeting within the group. The group got together two times each week, one with the whole group other within the sub group. Continuous work with the report where done and discussed. The work were checked against the GANTT-chart to ensure the worked proceed as planned.

3.3 Engineering Approach

The development of a mechatronic system is a combination of several domains. For solving the problems, the well established V-model was used. It is used for a generic design of a mechatronic system. An overview of the model is presented in Figure 3.1.

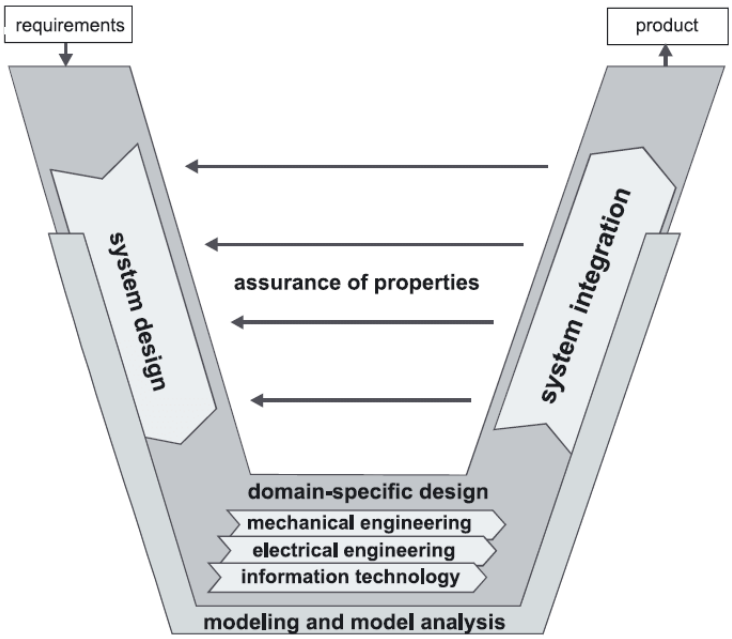


Figure 3.1: V-Model according to guideline VDI2206 [51].

The first stage of the process was the requirements from the stakeholder. The given task was clarified, defined and described. The requirements where later used for verification and validation of the project. During the second stage, system design, the physical and logical characteristics where described. The system was also divided into sub-subsystems. The third step of the V-model is system integration. The results where implemented and analysed. The process was continuously checked with the requirements and solutions to check if it would fulfill the characteristics and requirements. Depending on the complexity of the product, the amount of times run through the steps will differ.

Chapter 4

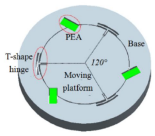
Concept Design

4.1 Discarded Designs

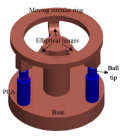
During the design phase existing concepts where evaluated to see if they would fulfill the requirements. In addition, new concepts design where considered by the members of the group. Here are the design concepts that were evaluated and discarded based on critical judgement from the SocketSense group.

4.1.1 Design 1: Two Stage Platform

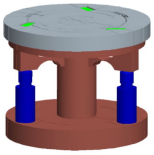
The first design concept is something that already exists in the market and provides 6 DOF, as required. This platform, as better explained in 2.5.3, consists of two stages, the upper stage is a 3-RRR parallel manipulator seen in Figure 4.1a, which then is in series with the lower stage consisting of a 3-RPS parallel manipulator seen in Figure 4.1b and 4.1c. The reason this design was discarded is because of the small movement possibility (µm) [30].



(a) The 3-RRR upper stage parallel manipulator [30].



(b) The 3-RPS lower stage parallel manipulator [30].



(c) The complete structure of the hybrid manipulator [30].

Figure 4.1: The two stage platform.

4.1.2 Design 2: Linear Parallel Robot

This design also exists in the industry and fulfills the technical requirements regarding the DOF [52]. It uses a metal spiral rod to convert the rotational movement of the motors to a linear movement as well. The yellow squares can move along the rod axis and therefore move the legs. Due to the difficulty to achieve the required movement and possibility to expand this concept got discarded.

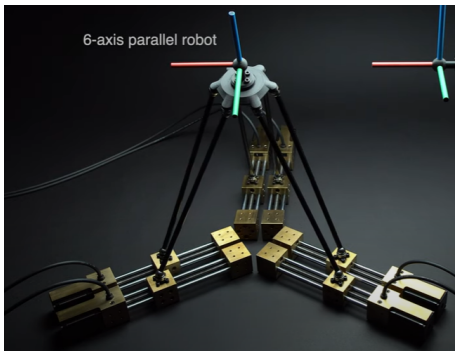


Figure 4.2: Linear axis robot [52].

4.1.3 Design 3: 3D Printer Setup

The third design was inspired by the movement of a 3D printer. The base platform has 2 DOF, likewise to a 3D printer can move left and right as well as front and back. On that platform, several telescopic rods are used as a SP to reach 3 DOF. At the top, there is a rotational platform, see Figure 4.3. Combined all the platforms provide 6 DOFas requested.

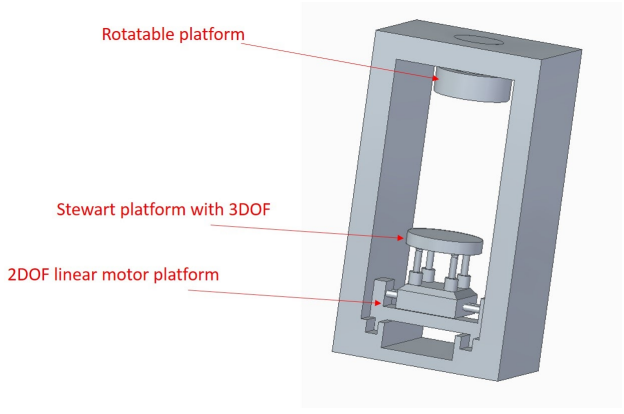


Figure 4.3: 3D printer setup.

After further discussion with the stakeholders, it was understood that all DOF must

come from a single platform. This would have given the possibility to expand to 12 DOF in the future. This was the reason that this design concept got discarded.

4.1.4 Design 4: Table Tennis Robot

Design 4 was inspired both by the 3D printer setup and a table tennis robot [53]. As shown in the Figure 4.4, the body of interest (stump and socket) can be controlled by four motors. The socket is connected to a ball joint which can move freely on a 2D axis setup, whereas the stump is connected to a frame which can extend to ensure 6 DOF.

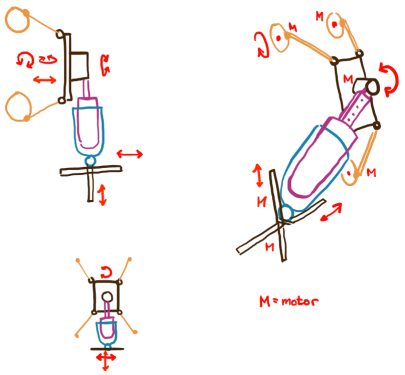
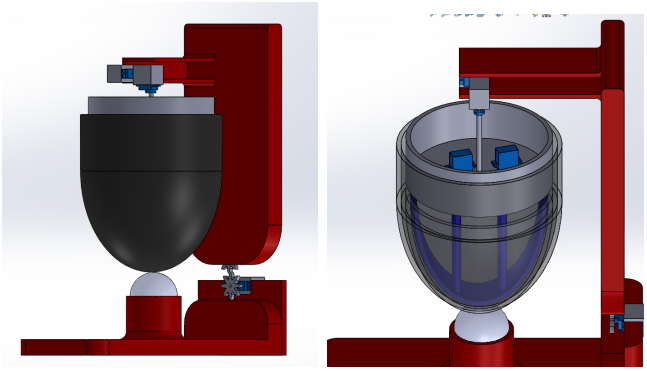


Figure 4.4: Table tennis robot.

This design had the same problem as the previous designs. Not all DOF can be provided by a single platform, since it needs the base axis to move as well. Ergo it was also discarded.

4.1.5 Design 5: Stepper Motors

This concept idea was created in an effort to utilize the stepper motors instead of linear actuators. The stump and socket are again connected to a ball joint, to provide free movement in any direction and the stump was connected to three stepper motors. The first one, right above the stump, can rotate independently from the other two but when the second or the third were actuated the first steppers' case would move as well. This setup could only give 5 DOF in total, since the body could not move up and down (along the Y axis). Along with the fact that, the movement did not come from a single frame, design 5 was also discarded.



(a) Normal view. (b) Transparent view.

Figure 4.5: Design 6: Stepper motors.

4.1.6 Design 6: Juggling Machine

Design 6 was inspired from a Juggling machine [54], where a square socket was moved by 10 wires in order to throw and catch a ball. The functionality is based on the Cable-Driven Manipulators, as explained in 2.5.2. Using this set up the wires provide controlled movement in any direction. The downside with this concept is that it takes a lot of space and it is not easily expandable.

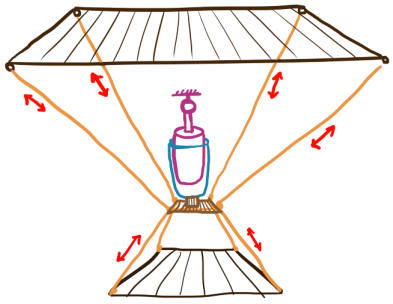


Figure 4.6: Juggling machine.

4.2 Final Design Proposal

Comparing all the design concepts, while taking into consideration the insights from SOTA as well as feedback from experts, the Stewart platform configuration with linear actuators was chosen. The SP fulfills all the requirements of the project and it is also the most common solution for such problems. This was an advantage, due to the large amount of documentation available concerning the mathematics and control of the SP. For example, MATLAB has its own built-in dynamic model of a SP with linear actuators, in the Simscape environment.

The proposal shown in Figure 4.7, consisted of two Stewart platforms for 12 DOF. Their fixed bases were connected to a singular rigid frame for stability. The stump and socket were mounted on the respective moving platform, the stump on the upper one and the socket on the lower one. The two platforms represented the hip and knee joint with 6 DOF each. This emulated the real configuration of the human joints, resulting in a realistic movement.

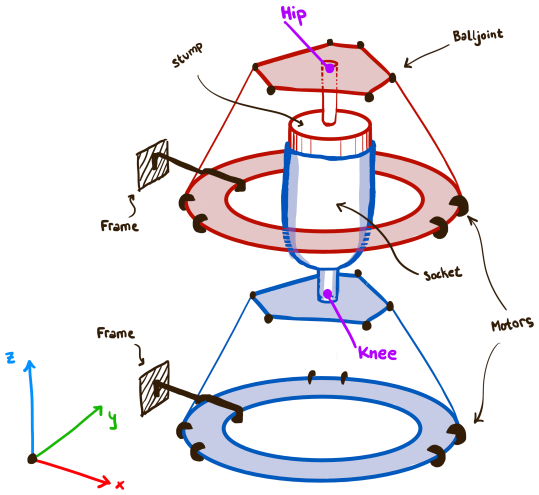


Figure 4.7: Design proposal 12 DOF.

For this project, only the upper SP was constructed due to limitations on complexity, time, experience and budget. However, the test rig was designed and constructed to allow the expansion of another SP according the proposal.

Chapter 5

Implementation

5.1 Final Design

The final solution, based on the proposal illustrated in Figure 4.7, aimed to realise the upper SP on a fixed frame. The design was modelled using a Computer Aided Design (CAD) software called Solid Edge. The design contains all detailed models of required components, as shown in Figure 5.1

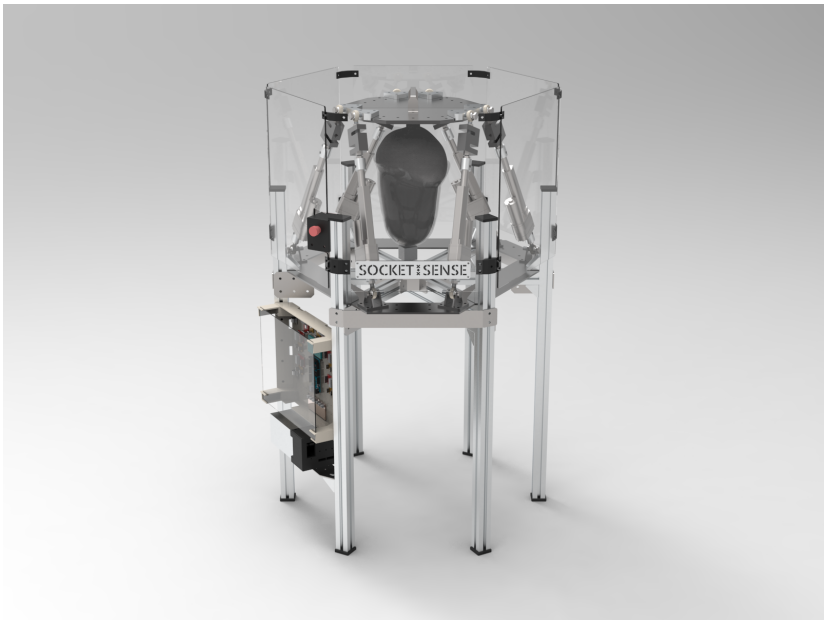


Figure 5.1: Render of the final Design, made in Solid Edge.

As can be seen in Figure 5.2, the final design consisted of multiple sub-assemblies. The frame brought structural integrity with a solid metal plate and 6 aluminium

extrusions as structural legs. The actuating legs connected the frame to the upper platform, thus created the Stewart platform configuration. In between the upper platform and the frame was the stump and socket, which could be detached, adjusted and replaced. The electrical box was mounted on one side of the frame, between two aluminium legs.

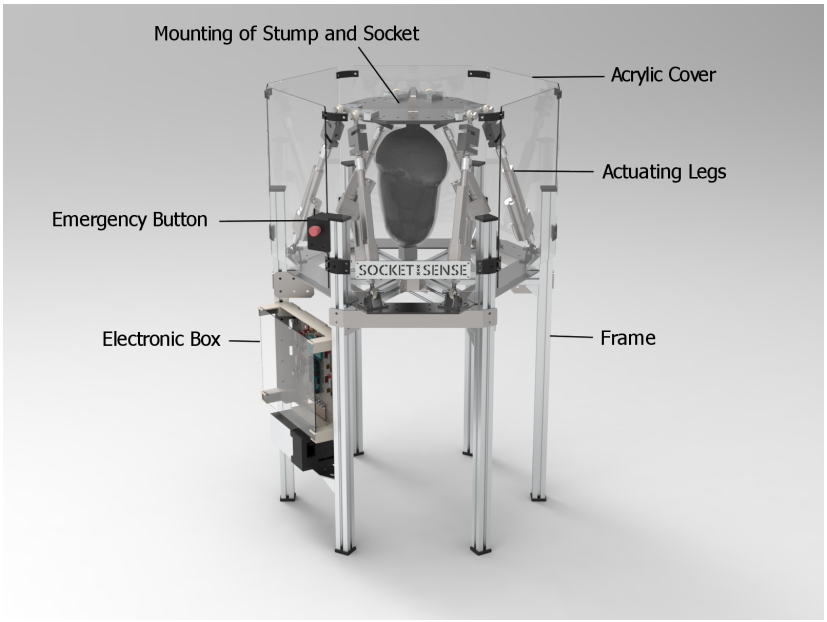
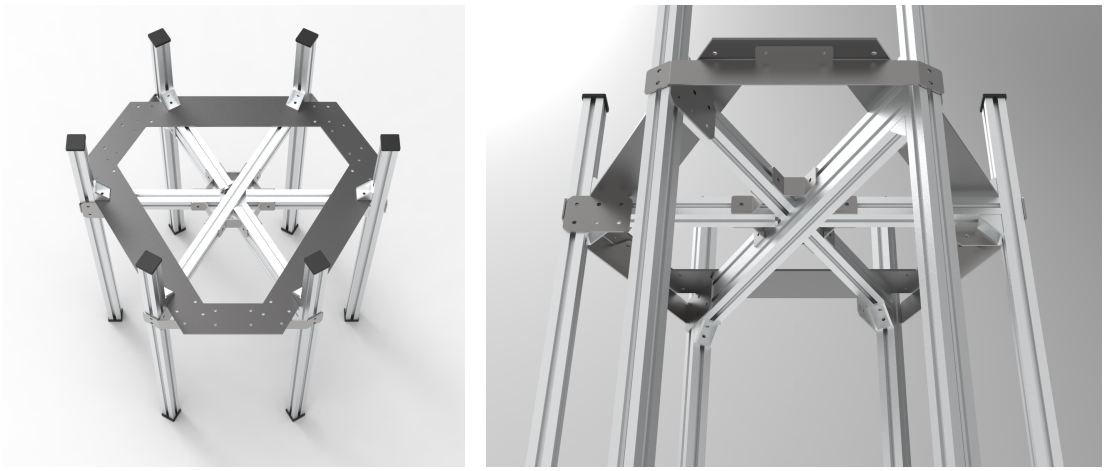


Figure 5.2: Render of the Final Design, made in Solid Edge. Points out the different sub-assemblies.

5.1.1 Frame

The frame's core origin from a waterjet-cut metal sheet of 5 mm. It connected the structural legs, actuation legs, and the base for the socket's mounting. It was designed in a triangular hexagon manner based on the requirements of a Stewart platform, allowing the actuating legs to be connected in alternating pairs. The structural legs, made from 40x40 mm aluminium extrusions, was secured with corresponding angle brackets to the frame. The 6 legs was also connected with crossing aluminium extrusions, resulting in a 6-way cross which can be seen in Figure 5.3a. Two extrusions crossed the full length, and shorter ones filled the gap to ensure that one united plane of extrusion could be achieved. This plane is discussed later in the Mounting of socket sub-chapter. 60 degrees angle brackets of 3 mm waterjet-cut steel connected the extrusions in the middle. The aluminium was chosen because of their ease of assembly and modular properties. Allowing mounting of additional components e.g cables, acrylic covers, emergency stop button and electrical boards.

Flanges was created to reduce stress on the frame underneath the connection-points



(a) Above view of the frame sub-assembly, note the base metal plate and crossing aluminium extrusions.

(b) Below view of the frame sub-assembly, note the flanges and splice plates.

Figure 5.3: Render of the frame sub-assembly, made in Solid Edge.

of the actuating legs. Splice plates was made to secure the crossing aluminium to the structural legs, uniting the parts. These can be seen in Figure 5.3b and was made from 3 mm waterjet-cut steel.

Small feet was 3D printed and placed on either end of the structural legs to prevent the sharp edges from damaging the user or the floor.

5.1.2 Actuating Legs

The actuation legs consisted of a series of multiple components in order to connect the Rod ends and Load cell to the linear actuator. A demonstration can be seen in Figure 5.4.

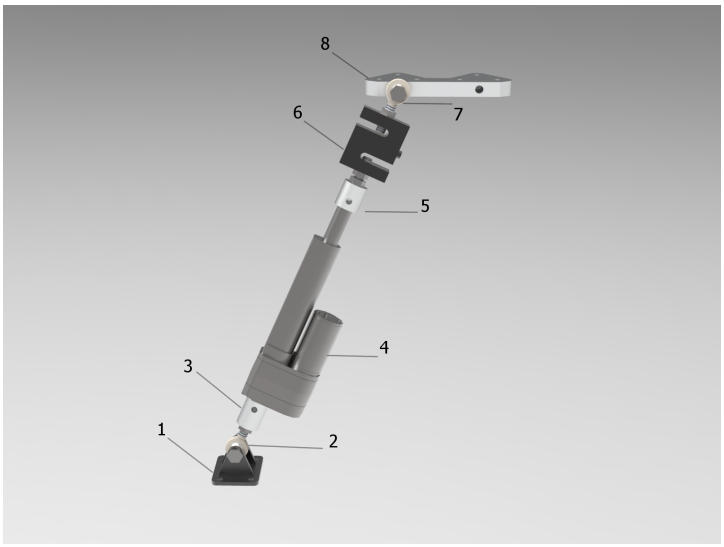


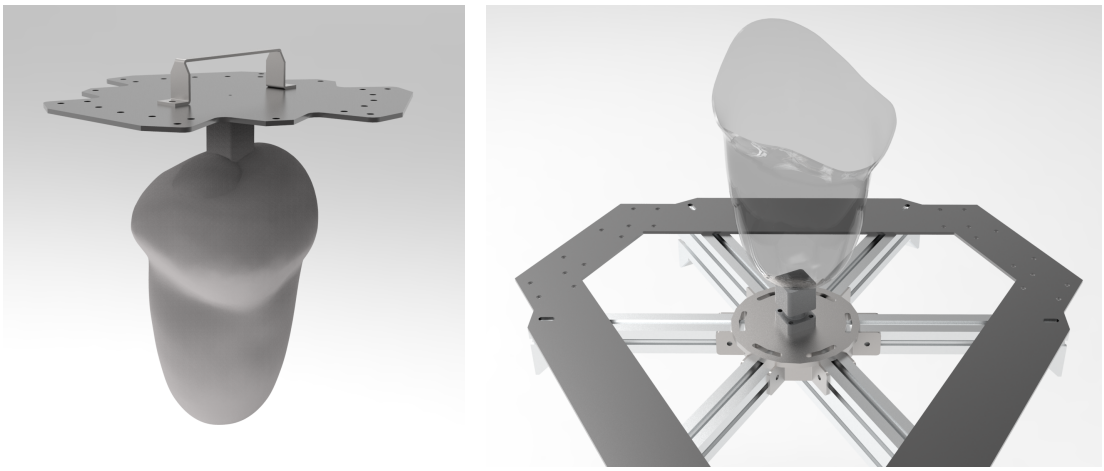
Figure 5.4: Render of the actuating leg sub-assembly, made in Solid Edge. 1) Mounting bracket. 2) Rod end. 3) Connector female. 4) Linear actuator. 5) Connector male. 6) Load cell. 7) Rod end. 8) Aluminium mount.

It was important that the ball joints (of the rod ends), load cell and linear actuator was concentric so the measured force correlated to the actuated force, and no unnecessary torque tension occurred. Special housings made from aluminium was created to attach the components to the linear actuator. The Mounting bracket was made of three waterjet-cut 5 mm steel plates, which were welded together. The Aluminium mount was also waterjet-cut but were made of 18 mm thick aluminium. Two holes on the side was drilled and tapped for the M12 bolt. The both mounting components also had spacers, made from a 13 mm steel pipe, that constrained the rod ends to the center of the M12 bolts, allowing for unrestrained bending of the joint.

5.1.3 Mounting of Socket and Stump

The socket was secured on top of a 8 mm steel disc. The disc had tracks along its circumference which lines up with the 6-way cross of aluminium extrusions mentioned earlier in the Frame chapter. It was secured with T-nuts and bolts, which allowed the disc to move and rotate as the T-nuts slid in the tracks. This was made so the socket could be easily adjustable after the stump. This can be seen in Figure 5.6b.

The stump was made replaceable by splitting the upper platform into two parts; a ring and a lid. The stump was attached to the lid, along with a handle, and could be lifted out of the test rig through the ring. See Figure 5.5 for a demonstration and Figure 5.6a for the configuration.



(a) The handle, lid and stump. (b) The socket, disc and the 6-way cross.

Figure 5.6: Render of the socket and stump sub-assemblies, made in Solid Edge.

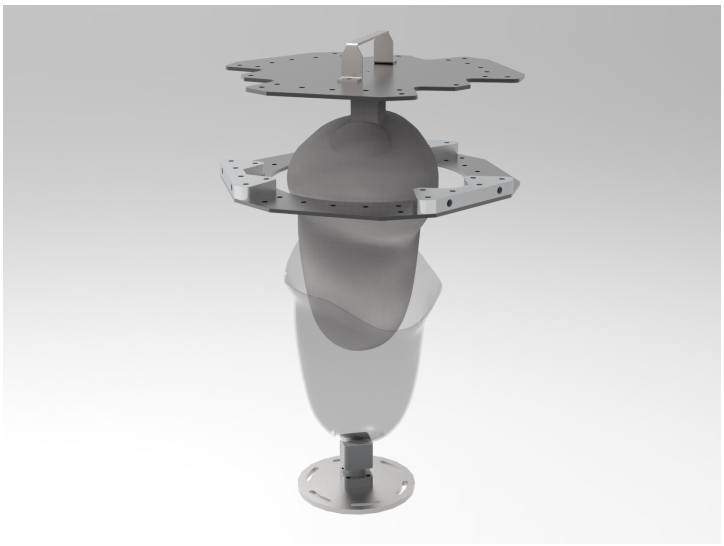
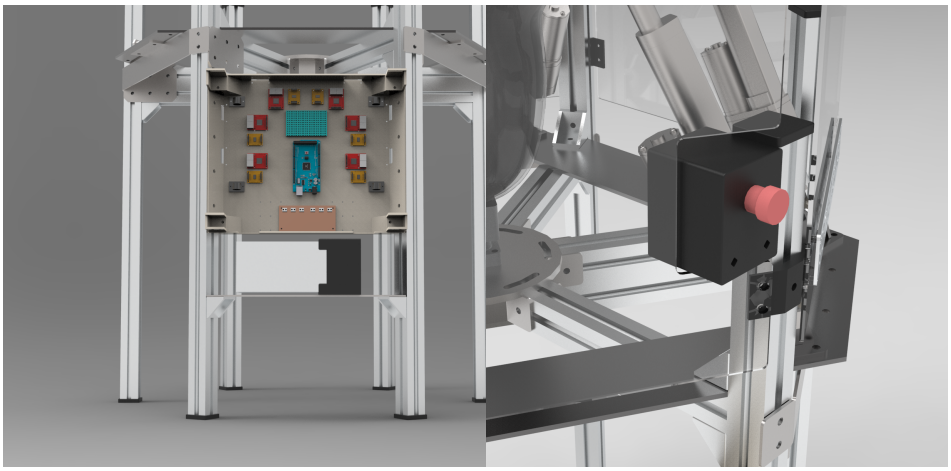


Figure 5.5: Render of the Final Design, made in Solid Edge. Demonstrates the removal of the stump.

5.1.4 Electrical Box and Emergency Button

An electrical box was created to hold all the electrical components. It was made with laser-cut 4 mm plywood, which were glued and screwed with brackets made of 30 mm square wood. The holes for the components was placed in the CAD software. 6 small openings was placed in the back of the box to allow the cords from the actuator and load cell to enter the box.



(a) Front view of the Electrical Box. (b) The Emergency button.

Figure 5.7: Render of the Final Design, made in Solid Edge. Shows the Electrical box and Emergency button.

To protect the components, some holders was created to attach a acrylic sheet in front of the opening. They were made from laser-cut 4 mm plywood with thoughtfully placed teeth and crevasses to glue them together.

The power supply, described in the following Electronics chapter, was placed upon a shelf beneath the Electronic box. The shelf was created from a 3 mm waterjet-cut steel plate and was attached to the structural legs with two angle brackets. The power supply was secured on the self with a 3D printed mount. In order to protect the electrical connections of the power supply, and to add a on/off switch with a outlet, a 3D printed cover was created.

The Emergency stop button configuration was made with two 3D printed parts. It was constructed to easily attach it to the aluminium extrusion of the Frame. Using a hinge and internally mounted bolts, the construction could be combined with two bolts. There was also two holes for the attachment of Cable glands, which protects the wire connection from tugging. These sub-assemblies can be seen in Figure 5.7.

5.1.5 Acrylic Covers

A protective ring of 6 mm acrylic sheets was laser-cut and placed around the Stewart platform. This was to protect the user from unforeseen parts, since the actuated forces was relatively high, but also prevents the user from reaching into restricted areas during actuation.

The acrylic sheets was mounted with 3D printed holders on the aluminium extrusion of the Frame. The sheets were also connected at the top with similar holders. 4

sides was identical, while 1 had 4 extra holes to attach a SocketSense logotype plate, made from a waterjet-cut aluminium sheet. The last one had an extra cutout for the Emergency button. See Figure 5.8 for a visual demonstration.

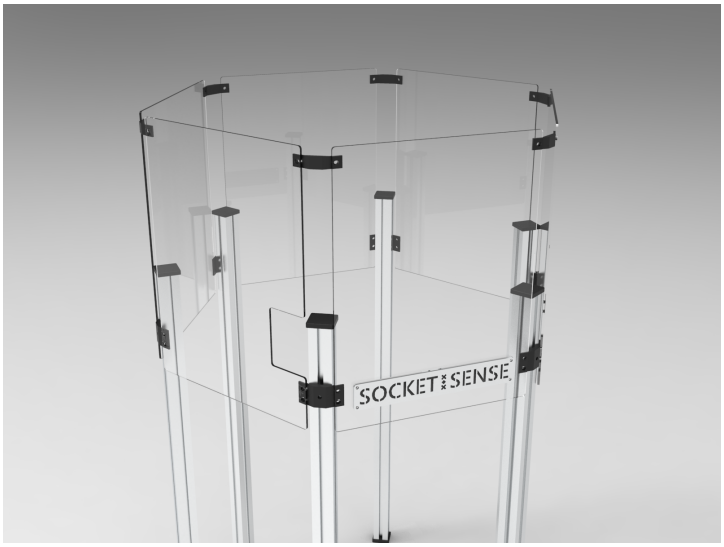


Figure 5.8: Render of the final Design, made in Solid Edge. Demonstrates the protective layer of acrylic sheets.

5.2 Electronics

After simulating the gait cycle in OpenSim and collecting the relevant data the components were carefully selected to satisfy the force and torque requirements needed. To avoid unnecessary technical issues and to allow easier future development, a safety factor was used during the selection of each module.

5.2.1 Actuators

In order to choose the appropriate actuator type, the decision was between servo motors, stepper motors and linear actuators, which are furthered explained in section 2.6. All three types mentioned were sufficient for the scope of this project, with main difference to be the cost. To get easier and more intuitive control the linear actuators were used, since many control examples were already available. The actuators and the load cells were assembled to be aligned. This way the load cells, with the help of an amplifier, can be used to measure the forces produced by the actuators. The most important technical specifications for the linear actuator were the operating voltage, current, nominal dynamic load, maximum static load and speed. Nominal dynamic load is the maximum load the linear motor can move, in this case the maximum load the motor can pull down the stump with. Since the linear actuators are self-locking, they will not draw any current during stall load. The

maximum static load can be ignored since the static loads in this system are much lower compared to dynamic loads. The voltage and current for the actuators was provided by a power supply, whereas for the motor drivers and load cell amplifiers the power was provided by the micro-controller.

Figure 5.9: Linear actuator [55].

5.2.2 Drivers

To perform the required movements the linear actuators needed to be able to move in both directions, forward and backwards. For that reason a full H-bridge motor driver was used for every actuator, as seen in Figure 5.10. It is also important to have H-bridge that is physically able to provide the power that the actuators need to operate at full potential. There are two types of inputs to motor drivers. To control the direction of the actuator one sends High or Low signal to specific pins on motor driver, and that changes the polarity (direction). To control speed of the actuator a value between 0-100 was sent to the driver that indicates the duty cycled of PWM signal.

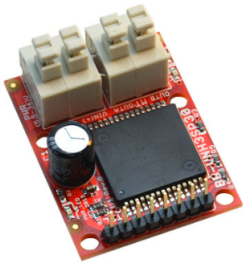


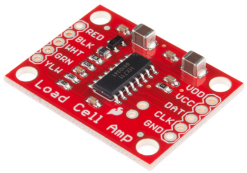
Figure 5.10: Motor Driver [56].

5.2.3 Load Cell and Load Cell Amplifier

Since the goal of this project was to emulate the correct forces and torques within the socket, it was important to have force sensors for feedback control. To achieve that, S-shaped load cells were used to measure the force along the legs (actuators). The test rig is designed for a maximum of 1200 N of vertical force and that force will be divided among six actuators. The starting angle of each leg is the same and the maximum load for each leg is defined. Hence for each load cell the maximum

load was calculated and the right component was chosen accordingly, Figure 5.11a. For a load cell sensor an amplifier is needed to magnify the signal output and feed it into a micro controller, Figure 5.11b.

(a) Load cell [57].



(b) Load cell amplifier [58].

Figure 5.11: Product pictures of load cell and load cell amplifier.

5.2.4 Power Supply

Another electronic component was the power supply. The specifications were that the output voltage should be the same as the actuators' operating voltage and that the output current should be at least six times the amount that each motor needs. This model of 'MeanWell' company, had 3 sets of variable outputs (voltage and ground). This means that even thought the maximum voltage that the power supply could provide was 27 V, it was possible to decrease it to 24 V by changing a variable resistance it contains inside. Into the 3 sets of output pins, 6 wires were connected that lead into a voltage splitter PCB. The PCB had 3 sets of double molexes as input from the power supply, which were then splitted into 6 sets of double molexes as an output to the 6 motor drivers. The PCB was also mounted with a custom 3D part around it.

Figure 5.12: Power Supply [59].

5.2.5 Microcontroller Arduino Mega

In order to control the actuators and to collect the sensor data, from both the potentiometers inside the motors and the load cells, an Arduino Mega was used.

The Arduino Mega was chosen due to its large number of analog/digital pins and PWM pins. Furthermore, due to its performance (fast and reliable) as well as its extensive third-party libraries. The Arduino powered all six load cell amplifiers, six motor drivers and six potentiometers which required 5 V power. All 18 electronic components had their power and ground connected on a breadboard which was then connected to the 5 V power and ground of Arduino with only 2 wires. All six sensor inputs from the actuators' potentiometers were connecting to the Analog ping of Arduino (A1-A6). The data pins form the load cell amplifiers were connecting to the Digital pins of Arduino (22-27) but all the clock signals from the amplifiers were connecting to a single digital pin (31). Then each driver had to be connected to 5 digital pins resulting to a total of 45 wires on Arduino Mega.

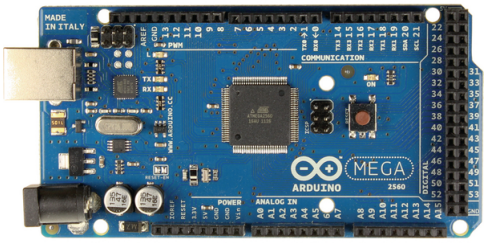


Figure 5.13: Arduino Mega [60].

5.3 System Architecture

A brief overview of the system architecture can be seen in Figure 5.14.

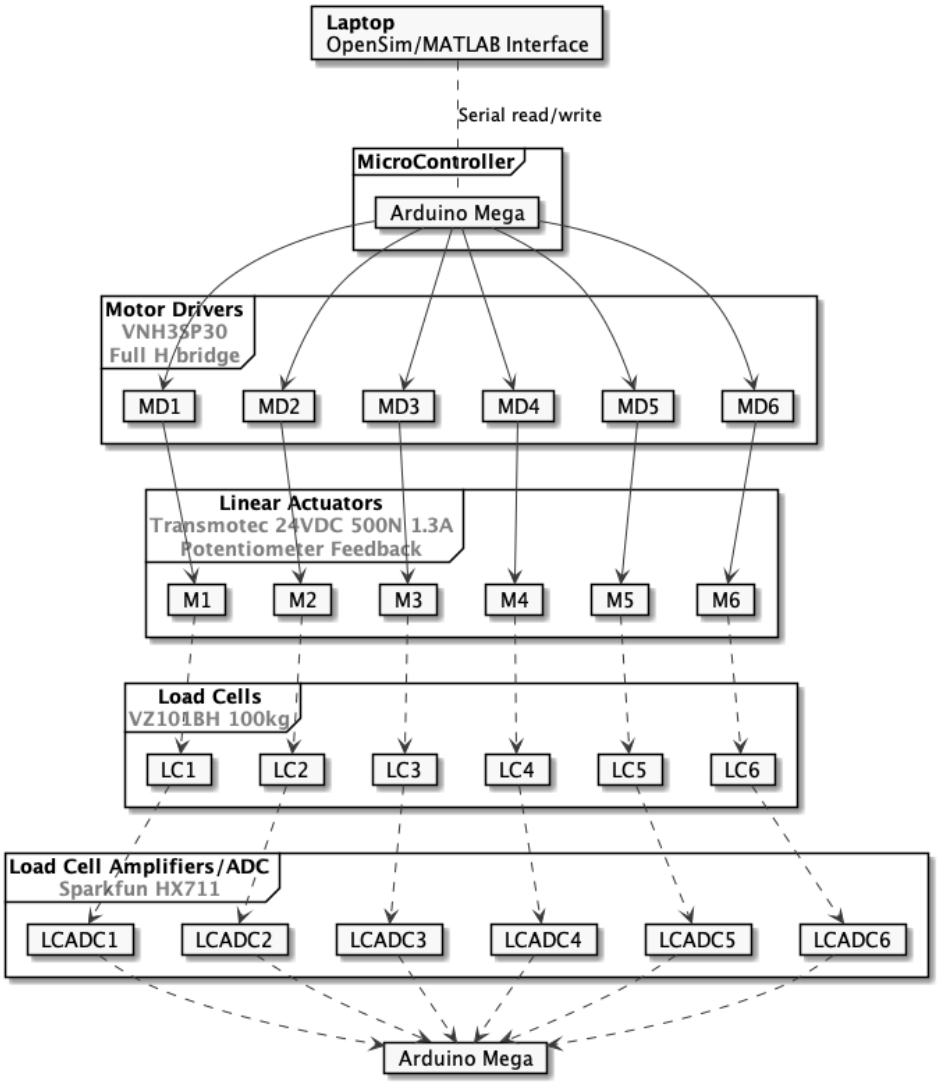


Figure 5.14: Overview [59].

5.4 Test Rig

The components were assembled according to the final design and can be seen in Figure 5.15.

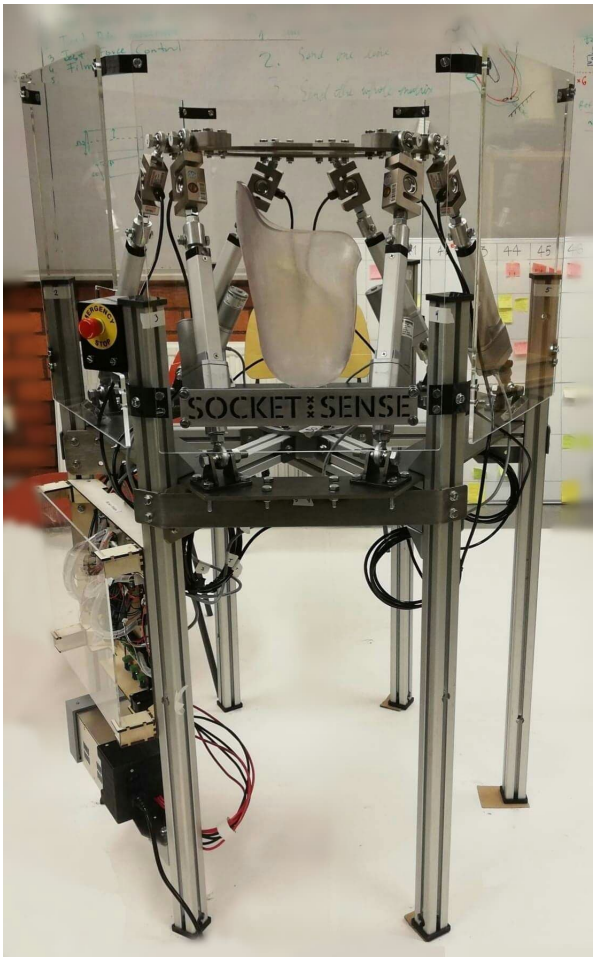


Figure 5.15: Assembled test rig.

5.4.1 Stump

To build a stump which could be used in the rig for the testing, the silicone socket was used. The silicon socket was provided by the stakeholder and had previously been used to pump it with air when testing where done with the socket. The silicon were to resemble a stump. It was filled with sealing foam and a metal square pipe which had metal wings added to it to create a more sturdy design and give the foam more places to attach to. The sealing foam hardens when in contact with air and filled out the gaps within the socket. The lid had a larger square pipe welded on to it to ensure the stump would be secured and centered on the lid. The square pipe

had four holes in it aligning with the holes on the lid for firm attachment. In Figure 5.16, the stump is visible from a side view with the lid attached. At the bottom there is the a screw for which the socket will clamp on to, securing it in the middle of the socket and providing vertical reaction forces.

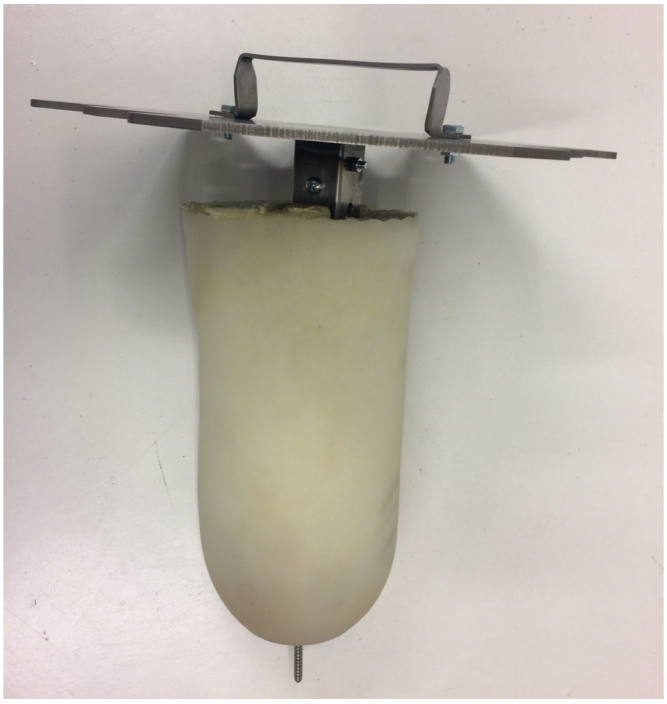


Figure 5.16: Manufactured stump attached to the lid.

5.5 Dynamic Modelling and Control

To be able to control the test rig a dynamical model had to be constructed to obtain the required reference signals for each of the legs in the rig. The built-in SP in MATLAB was used as inspiration when creating the model. It was also used to obtain the blocks to derive the required reference signals for recreating specific motions with the test rig. To ensure that the rig reached these references a controller was also designed.

5.5.1 Simscape Multibody

Because the model of the test rig was used to get the reference signals it was important to resemble positions, lengths and masses of components of the test rig in the model. This was to ensure that the model and test rig behaved in the same way.

Since a SP consists of several parts which reappears multiple times in the construction a library was created in the Simscape Multibody environment. This simplified both the construction and editing of the model. By copying a block from the library into a separate Simulink file, the block would stay linked to the library, making it possible to edit several linked blocks by just changing the corresponding blocks in the library. The library of the test rig can be seen in Figure 5.17.

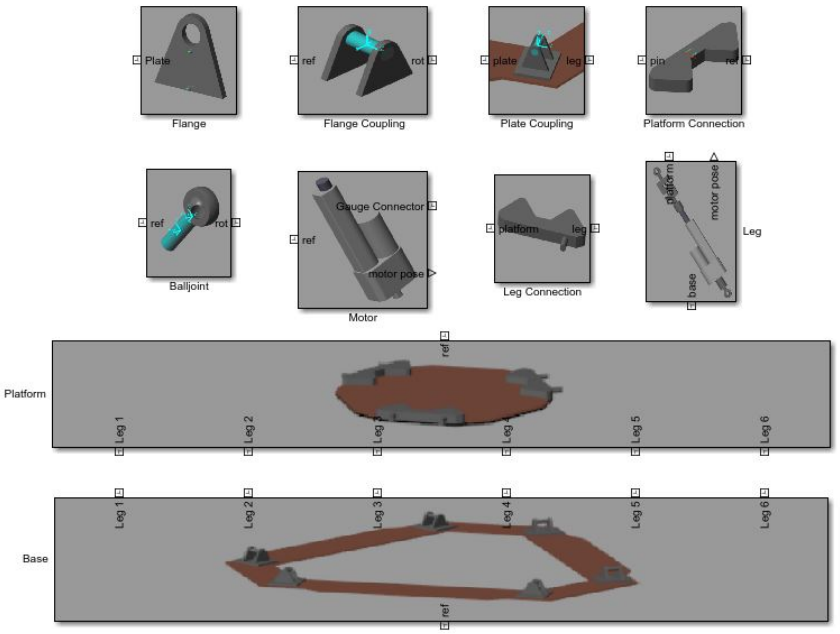


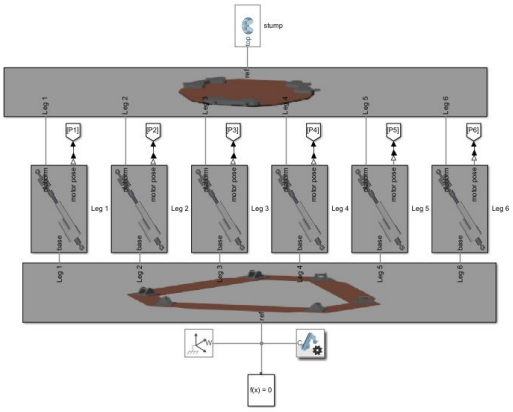
Figure 5.17: The library of the Simscape model.

The linear actuators in the rig were modeled as prismatic joints, providing one translational degree of freedom. This joint had several possible actuating and sensing attributes, making it possible to receive for example the extension of each leg during simulation. The initial position of the prismatic joint was set as half of the stroke length of the linear actuator, i.e. 50 mm, enabling the platform to move in either direction.

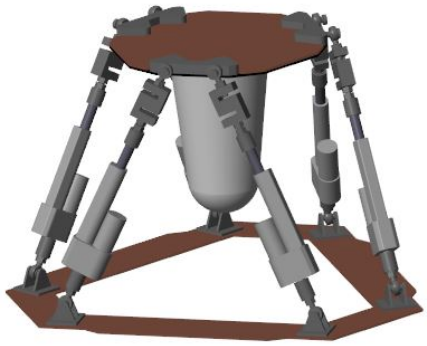
The final model of the test rig with the connected library blocks can be seen in Figure 5.18a. Figure 5.18b shows what the model actually looks in the simulation environment.

Identify Force References

The forces and torques from the gait cycle in OpenSim were desired to be applied at the center of the platform. In the model it is achieved by adding a 6-DOF joint having the capacity of actuating forces and torques at the desired point, using the data from OpenSim as input. This can be seen at the top of Figure 5.19.



(a) The library blocks connected to create the test rig model.



(b) The test rig in simulation view.

Figure 5.18: The Simscape model of the test rig.

These external forces and torques makes the legs compress, which can be sensed by the prismatic joints in each of the legs. By adding a spring and damping system in the prismatic joints, the force for each individual leg can be calculated using Hooke's law

$$F = k\Delta z, \tag{5.1}$$

where k is the spring coefficient and Δz the change in extension of the leg. This calculation can be seen to the right in the Figure 5.19. These forces were set as the reference forces for each leg in the test rig to achieve the desired total force and torque from OpenSim at the center of the top platform. It is also suggested to choose a damper in the prismatic joint to reduce oscillations.

Since the load cells used in the test rig only measures the changes in force compared to when initialized, the mass of the top platform was not included in the force readings. To recreate about the same behavior in the Simscape model the gravity

of the simulation environment were set to zero. Otherwise the gravity acting on the top platform in the model would cause a change in extension of the legs, adding to the force from the external force acting on each leg.

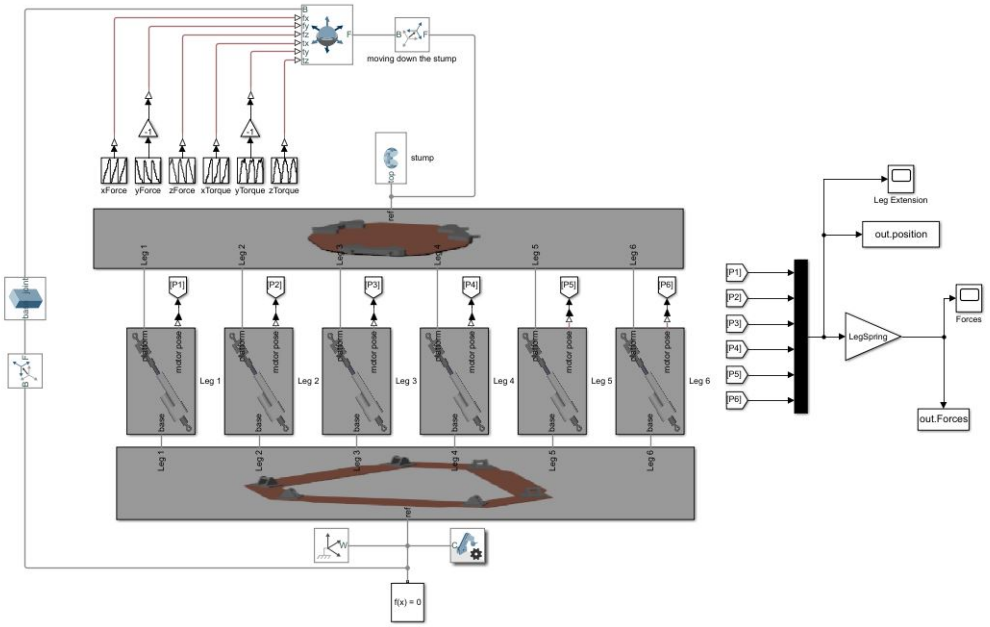


Figure 5.19: Simscape model blocks of the test rig with external forces and torques and the calculation of the force in each leg.

Inverse Kinematics

One of the requirements from the stakeholder was to ensure that the test rig could move around in some kind of controlled motion. To achieve this the reference positions for each of the legs had to be calculated throughout the whole desired motion.

The Simulink blocks used is shown in Figure 5.20 and were part of the built-in SP model in MATLAB. The Reference Trajectory block were used to define the desired motion of the top platform according to a time dependent sinusoidal wave. The user could set the desired point in space which the platform would move towards, resulting in a vector d. The user could also set the desired rotation of the platform relative to the base frame around each coordinate axis, resulting in a rotation matrix R. Each motion were defined by its amplitude, frequency, phase shift and offset, all editable by the user. Because of the time dependency of the motion it could be redefined as a desired position of the platform at each time frame, resulting in the output signals d and R being updated continuously for the different time frames.

The Inverse Kinematics block took the vector d and matrix R for each time frame

to calculate the new length of each leg, using this equation

$$l = d + RP - B, \tag{5.2}$$

where P is the coordinates of the platform joints connected to the top parts of the legs, and B is the coordinates of the base joints connected to the bottom part of the legs. To then find the necessary extension of each leg to reach the desired position the length l were subtracted by the initial length of the legs.

By saving the extensions of all the legs throughout the whole motion it could be sent as reference positions to the test rig.

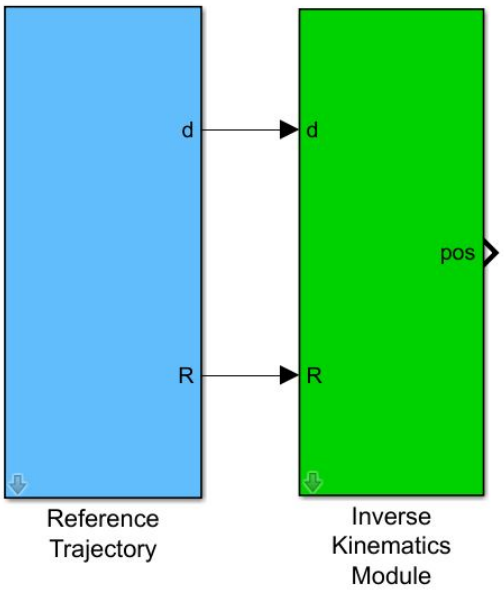


Figure 5.20: Simulink blocks of the trajectory planner and inverse kinematics to obtain position references.

5.5.2 Control of Motors

Position Control

It was considered a benefit to design and implement the position control on the test rig first, since position is easier to observe than the force, which simplified the troubleshooting to make sure all components behaved as expected. Since there were no requirement on the performance of the test rig's capability of moving around in space a simple P-controller was made.

By taking the position reference and the position sensor data from each actuator the error of the position was calculated. The speed of each actuator were then decided by how big the position error was. The actuator with the largest position error were

then set to get the maximum speed. The speed for the other actuators were then calculated by dividing the error of each actuators position with the largest error and multiplying it with the maximum speed. In that way all actuators should reach their position references at the same time. In that way the accumulation of error for a individual actuator is avoided and the movement of the whole platform will be more accurate.

Force Control

When designing the test rig, it was assumed that the stump would be stiff and have linear characteristics, which would result in the test rig performing only very small movements while applying the forces.

Due to the assumed small movements and the fact that the linear actuators moved relatively slow, resulting in almost no overshoot, and no requirement on the rise time of the system, a simple static P-controller was considered enough for force control. So, when the actuators reached the references the pressure data would be collected and the force error calculated, resulting in a change in speed of each actuator depending of the force error.

In the case of the stump being linear it also provides the possibility of creating a model of it in Simscape Multibody. In this way the stump could be represented by an spring and damping system, just like the system in the prismatic joints. This means that it would also be compressed when being affected by an external load, causing the legs in the model of the test rig to extend. It is then possible to use the extensions of the legs as positions references to recreate the same external load in the real test rig.

By knowing extensions of the legs when the stump is affected by the external load through the 6-DOF-joint in the model,

5.6 Software

5.6.1 MATLAB GUI

In this project there were many different software's in use when gathering simulation data, processing it and to send/receive data to/from the micro-controller. In order to have full control of the whole process via just one software an interface communicating with all other software's was conceived. The interface would be of use when performing/analyzing integration tests of the rig as well as a tool used by the end-user. See Figure 5.21 for an overview of how the GUI is communicating with other software's.

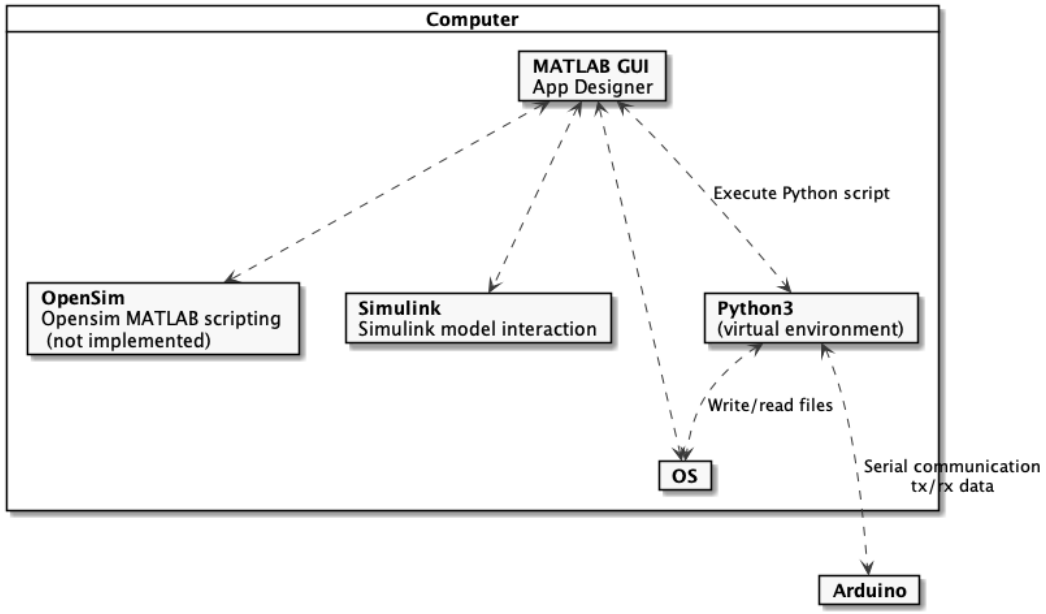


Figure 5.21: Overview [59].

The following features was initially requested by the team for the GUI:

- OpenSim: Create an OpenSim-model using scripting in order to:
  - Simulate walking/running/jumping-movement
  - Change the weight of the model
  - Export one gait-cycle of the simulated movement to a .mat file
  - Visualize the data
- Simulink

  - Choose and import exported OpenSim .mat-files (with force/torque data) into the Simulink-model of the test rig.
  - Change the step-size of the solver in Simulink
  - Change the simulation-time (in order to control how many gait-cycles to perform)
  - Visualize the resulting force and positions of the actuators in the model
  - Store resulting force/position to be used as reference signals in a .mat-file
- Arduino
  - Send new reference signals to Arduino via serial communication and store it locally on the micro-controller
  - Set option in Arduino for if it should perform force- or position-control of the test rig.
  - Read sensor data (using a Python script) while the test rig is performing the movement and store it in a .mat file
  - Visualize the resulting sensor data

In order to be able to perform all of the requested features the software used for creating the interface was chosen to be MATLAB App Designer, due to its built-in capabilities to interact with the other software's.

Due to time-constraints and problems with integrating certain parts not all features was implemented. The following features was at the end on the project implemented:

- OpenSim
  - Settings for OpenSim is not done via the interface but in OpenSim.
  - Exporting gait-cycle data of the simulated movement to a .mat file is done with a separate MATLAB-script
  - A drop-down menu where the user can choose pre-exported OpenSim-movement files to import is in interface.
  - Visualization of the chosen data.
- Simulink
  - Choose and import exported OpenSim .mat-files (with force/torque data) into the Simulink-model of the test rig.
  - Change the step-size of the solver in Simulink
  - Change the simulation-time (in order to control how many gait-cycles to perform)

  - Visualize the resulting force and positions of the actuators in the model-
  - Store resulting force/position to be used as reference signals in a .mat-file

- Arduino
  - Reference data is converted to a matrix (using a Python script) and written to a h-file that the Arduino IDE imports when uploading code. (Not done in interface). See for more information.
  - Record sensor data while the test rig is performing the movement and store it in a .mat file.
  - Visualize the resulting sensor data.

In Figure 5.22 the final interface can be seen.

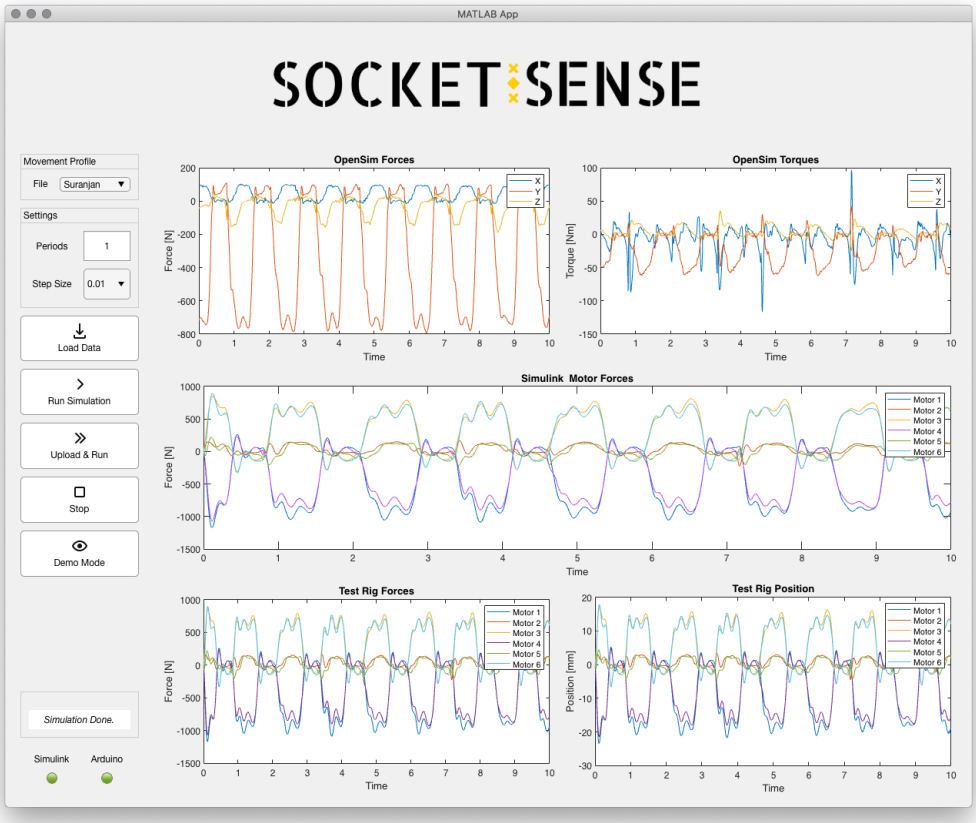


Figure 5.22: GUI interface. Made in MATLAB App Designer.

5.6.2 Arduino

In order to drive six linear actuators, 6 PWM signals with independent duty cycles should be generated by Arduino. Precise control of the test rig is needed thus, PWM signals with high precision. One way to do this is to use timers in Arduino. A timer is a piece of hardware builtin Arduino, which acts like a clock to measure time events [61]. An Arduino mega 2560 was used, based on ATmega2560. In ATmega2560, there are 6 timers including 2 8-bit timers (timer0 and timer2) and 4 16-bit timers (timer1, timer3, timer4 and timer5). Each timer includes two different PWM modes of operation: Fast PWM mode and Phase Correct PWM mode. In the project, Fast PWM mode was chosen to generate PWM signals. For each 8-bit timer, it has two PWM channels and for each 16-bit timer, it has three PWM channels. Considering the convenience of control and pin assignment, one 8-bit timer (timer0) and two 16-bit timers (timer1 and timer5) were used. Timer0 is set to Fast PWM mode and the other two 16-bit timers are set to 8-bit Fast PWM mode. To set the modes, the bits of two 8-bit Timer/Counter Control Registers (TCCR): TCCRnA and TCCRnB should be changed. The bits of TCCR and their values are shown in Table 5.1 and Table 5.2.

In these registers, Wave Generate Mode (WGM) bits are used to set the mode of timers. After setting the correct mode, the Compare Output Mode (COM) bits should be set. These bits are used to switch on/off the PWM channels of the corresponding timers. For the 8-bit timer, the two PWM channels are all needed. For the 16-bit timers, only two of the three PWM channels are needed. Since each timer will generate more than one PWM signal, the COM bits should be set to ”non-inverting mode” where COMnx1 should be set to ”1” and COMnx0 should be set to ”0”. Now Arduino can generate PWM signals from 6 channels of three different timers. The next step is to set the frequency and duty cycle of each PWM signals.

To control the frequency and duty cycle, the Clock Select (CS) bits and the Output Compare Registers (OCR) should be set. Since all the timers used to generate PWM signals are set to 8-bit mode, each timer will count 256 times in one cycle. The maximum frequency of Arduino mega is 16 MHz, then the frequency of the PWM should be 16/256 MHz, which is about 62500 Hz. However, this frequency is too big. In order to decrease the frequency, the maximum frequency will be divided by a prescaler. The value of prescaler is controlled by the Clock Select bits. The values of prescaler with different CS bits values are shown in Table 5.3. The value of the prescaler was chosen as 64. Then the frequency of PWM signal is 16/64/256 MHz, which is about 1 kHz. Now the Arduino can assign PWM signals with specific frequency. Notice that because of the mechanism of timers, the two PWM signals that are from the same timer will have the same frequency. In this project, since the six linear actuators are the same type, the frequency of six PWM signals are set to the same value: 1 kHz. Besides frequency, the duty cycle should also be assigned.

In each timer, there are two Output Compare Registers. In non-inverting mode, each PWM channel corresponds to a specific OCR. The timer starts to count from 0 and each PWM channel will have high voltage level output. When the timer counts to the value of each OCR, the corresponding PWM channel will start to have low voltage level output. The timer will count back to 0 when it counts to 255 (8-bits) and all PWM channels will generate high voltage level output. Then everything will start again. Based on this mechanism, Suppose OCRnX stands for the value of a specific OCR (n stands for the number of timer and X stands for the channel, A or B). The duty cycle of corresponding PWM D is:

$$D = \frac{1 + OCRnX}{256} \tag{5.3}$$

Besides generating PWM signals, Arudino should also be able to read data from sensors, calculate errors between sensor data and reference and modify control signals. In order to have a good performance, one solution is to use a timer to attach interrupt periodically. In ATmega2560, timers can be set to Clear Timer on Compare Match (CTC) Mode. In this mode, the timer will starts to count from 0 and compare with the value of OCRnX, when they match, the program will be interrupted and an Interrupt Service Routine (ISR) will be triggered. Then the timer will be scaled to zero and continue the loop. In this project, the 8-bit timer used is: Timer2 to attach interrupt. In ISR, Arduino will read data of sensors, calculating errors and modify control signals. By setting timer2 to CTC Mode, WGM bits should be set to "010". The principle of setting other bits in TCCR is similar to Fast PWM Mode. Furthermore, to active CTC Mode, the bits in Timer/-Counter Interrupt Mask (TIMSK) Register should also be set. In TIMSK, there are two Timer/Counter Output Compare Interrupt Enable (OCIE) bits: OCIEA and OCIEB, which are corresponding to OCRA and OCRB. Setting one of the bit to "1", the timer will then compare with the value of corresponding OCR. The value of each register of Timer2 for CTC Mode is shown in Table 5.4 [62]. When setting the registers, since each register has 8 bits, it will have too many lines of code if setting every bit of register one by one. An easy way to reduce the code is to initialise every register to "0" and then set needed bits to "1". Then code in Listing 5.1 shows the code for setting Timer0.

```
1  TCCR0A = 0x00;
2  TCCR0B = 0x00;
3  TCCR0A |= (1 << WGM00) | (1 << WGM01)| (1 << COM0B1) | (1<<COM0A1);
4  TCCR0B |= (1 << CS01) | (1 << CS00);
```

Listing 5.1: Register setting code for Timer0.

Table 5.1: TCCR Bits and Values for 8-bit Timer [62].

Bit	7	6	5	4	3	2	1	0
TCCRA	COMA1	COMA0	COMB1	COMB0	N/A	N/A	WGM1	WGM0
	1	0	1	0	N/A	N/A	1	1
TCCRB	FOCA	FOCB	N/A	N/A	WGM2	CS2	CS1	CS0
	0	0	N/A	N/A	0	0	1	1

Table 5.2: TCCR Bits for 16-bit Timer [62].

Bit	7	6	5	4	3	2	1	0
TCCRA	COMA1	COMA0	COMB1	COMB0	COMC0	COMC1	WGM1	WGM0
	1	0	1	0	0	0	0	1
TCCRB	ICNC	ICES	N/A	WGM3	WGM2	CS2	CS1	CS0
	0	0	N/A	0	1	0	1	1

Table 5.3: Prescaler Values with Different CS Bit Values [62].

CS2	CS1	CS0	Prescaler
0	0	1	1
0	1	0	8
0	1	1	64
1	0	0	256
1	0	1	1024

Table 5.4: Register Values of Timer2 for CTC Mode [62].

Bit	7	6	5	4	3	2	1	0
TCCRA	COMA1	COMA0	COMB1	COMB0	N/A	N/A	WGM1	WGM0
	0	1	0	0	N/A	N/A	1	0
TCCRB	FOCA	FOCB	N/A	N/A	WGM2	CS2	CS1	CS0
	0	0	N/A	N/A	0	1	0	1
TIMSK	N/A	N/A	N/A	N/A	N/A	OCIEB	OCIEA	TOIE
	N/A	N/A	N/A	N/A	N/A0	1	0	1

5.6.3 Importing Reference

In order to import the reference matrix into Arduino, one solution was to convert the matrix into a head file and include this head file in Arduino. However, since the reference matrix is generated from MATLAB, the format does not match with Arduino. Another way to solve this problem was to use python to operate the file. The flow chart of this process is shown in Figure 5.23.

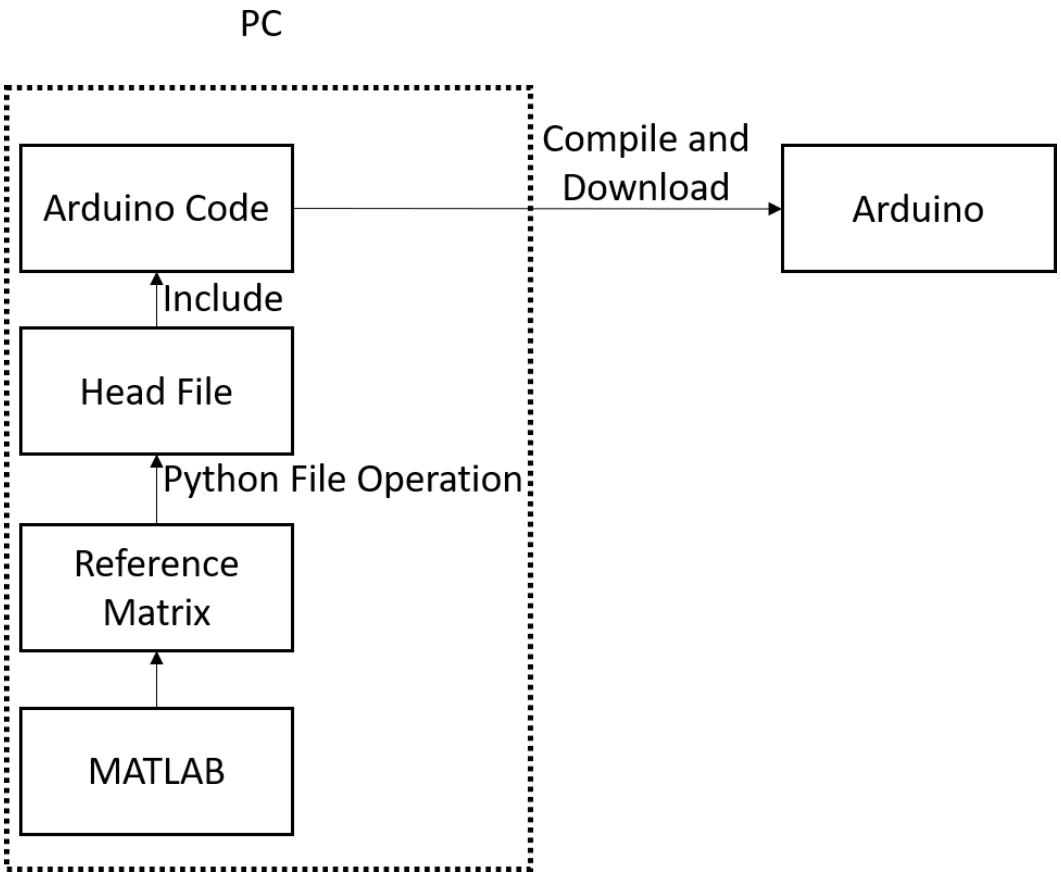


Figure 5.23: Process of Importing Reference.

5.6.4 Serial Communication

To be able to transmit reference-signals and receive sensor-data serial communication [63] was used to transfer information between the PC and micro-controller. On the Arduino Mega data was being transmitted/received via the built-in USB-to-serial-adapter (Serial port 0 and 1) [64].

Transmitting reference-data

Serial communication can be used in order to transmit reference-signals to the Arduino. See listing C.1 in Appendix C for how this can be achieved with a Python script. How to receive, parse and store the data in a matrix using Arduino can be seen in listing C.2 in Appendix C.

Receiving sensor-data

In order to send the sensor-data the `Serial.h` package, with its functions `print()` and `println()`, was used. The `Serial.print()` function converts the passing argument to char-datatype and transmit it via the serial RX-pin. The `Serial.println()` have the same functionality but adds a newline character `\r\n` to the end of the character-array. The sensor-data format was designed to be a comma-separated string and in the following order for each data-transfer:
`[FS1,FS2,FS3,FS4,FS5,FS6,MP1,MP2,MP3,MP4,MP5,MP6,R1,R2,R3,R4,R5,R6]` , where `FS` is force-sensor, `MP` is motor-position and R is current reference-value. See listing C.3 in Appendix C for how this is implemented.

In order to read, parse and store the received sensor-data string into a .mat fileformat a Python3 script was used. See Listing C.4 in Appendix C for the full code.

Chapter 6

Verification and Validation

6.1 Testing

6.1.1 Risk Identification and Testing

In order to deliver a product within the given time-frame and with acceptable end-result extensive testing was done. To reduce the risk of not meeting the requirements or delaying the project, identification of possible risks was done so that suitable test cases could be written. First off, unit testing of each electronics component (described in section 5.2) was identified as a critical step to ensure no damages or faulty operation. This could lead to unpredictable behaviour when integrating subsystems, which would lead to error-propagation and time-consuming troubleshooting. Extensive testing of the software was identified as critical at a later stage when doing integration test, especially the PC-microcontroller to actuator interaction. Faulty behaviour of controlling the rig could lead to damages on the rig, which would lead to further costs and delays. It was also noted that protective equipment such as acrylic sheets should be mounted on the rig when doing force-control on a mounted socket and stump. Faulty behaviour could in a worst case lead to damages on socket, which could potentially project splinters which in turn could damage the operator of the rig.

6.1.2 Hardware test

Linear Actuators

Each actuator was tested individually and later on connected to a motor driver. The procedure followed was:

1. Firstly check the outside of the actuator for any problems, like broken mounting holes, dust or broken teeth in the worm shaft.

2. Then it is the bearing test. The shaft connected to the motor needs to be rotated to see if it rotates smoothly. Rotor should spin quietly, freely and evenly. Also try to push and pull the shaft.

3. Furthermore, check the resistance with an ohmmeter, by putting the probes in ground and power of the actuator. The resistance should be on a scale of Mega Ohms. To check that there is no winding down to ground, with the ohmmeter or multimeter, the ohms were set on a low scale and each winding terminal and the metal casing of the motor were tested. In general, if there is any reading then the motor is broken.

4. Next, connect the actuator with a variable resistance and a power supply to see that it rotates and also to measure the output voltage with different resistance values.

5. Then, put a load on the motor and document the readings.

6. Test the encoder inside that it stops the motor when it reaches its minimum and maximum position.

7. Lastly, connect it with an embedded board and controlled with PWM. Check that the speed is changing according with PWM.

Motor Drivers

After the motors were tested successfully, each actuator was connected to an assigned motor driver which was then connected to an Arduino Uno. Arduino was sending the control signals to the driver and the driver was controlling the actuator to move upwards or downwards as well as setting the duty cycle of PWM.

Load Cell S-type And Amplifier

The load cells and the load cell amplifiers were tested together. Firstly the load cells had to be calibrated. To achieve this, calibrated weights up to 2 kg were used. By knowing the mass of an object it was easy to tune the reading of the load cell from Arduino to match it.

Power Supply

To test the power supply, first six wires were inserted in the 3 output set. Then carefully connected the multimeter probes on two of the wires and plugged it into a wall socket. The voltage was measured for all 3 sets of outputs separately. The resistance between the grounds was also measured to make sure there was no leakage. Lastly, the power supply was connected to the splitter PCB and the output of each set on the PCB was tested. The PCB was individually tested beforehand for any short-circuits.

6.1.3 Software Test

Due to the rigor testing of modern software, it was not deemed critical to test individual libraries or functions in those libraries extensively in order to ensure predictable results (though it was decided to use software that had withstand the test of time rather than novel, inexperienced technologies). Instead, primary focus was to ensure satisfiable result in the integration between different software. Parts of the proposed workflow was put under integration test and evaluated. If a test failed it was analyzed and altered or a new approach was designed and put under test.

Integration tests

1. Use "To Workspace"-block and `save()`-command to store force and position reference data from Simulink to a mat-file. Open file and compare with Sink-block.

2. Write Python script that will open the exported mat-file and print its content to terminal. Compare output with the actual mat file.

3. Send a string with Python (using `pyserial` serial communication library) to Arduino and serial monitor it in Arduino IDE. Compare results.

4. Write Python script that will open a mat-file with references, convert it to a Python list-matrix, send first row in matrix to Arduino and wait for the Arduino to send back the same row of data. Send next row and repeat. Compare results.

5. Write Python script that will send data as above, but have the Arduino storing the data in its flash memory (in a matrix consisting of pointers to pointers, which in turn points to arrays of float data). Make the Arduino send back the received matrix, one row at a time, via serial to the PC. Compare results.

6. Make the MATLAB GUI execute the Python script by using MATLAB py.-extension. Execute script 2 in this list and compare results.

7. Write Python script that will read actual sensor-data (position, force and hard-coded reference-data from Arduino using `Serial.print()` and `Serial.println()`), one line at a time, and store each row of sensor data to a mat-file. Open mat-file in MATLAB and compare results.

8. Store position reference-data in the same manner as in test 5 (matrix in Arduino flash memory), but do real position control with the received data. Send sensor-data via serial communication to PC. Compare results.

Arduino tests

1. Write Arudino code to set Timer0 to Fast PWM Mode and let Timer0 generate two 1 kHz PWM signals with different duty cycles. Use oscilloscope to check the result.

2. Write Arudino code to set Timer1 to 8-bit Fast PWM Mode and to let Timer1 generate two kHz PWM signals with different duty cycles. Use oscilloscope to check the result.

3. Write Arudino code to set Timer5 to 8-bit Fast PWM Mode and to let Timer5 generate two kHz PWM signals with different duty cycles. Use oscilloscope to check the result.

4. Write Arudino code to set Timer2 to CTC Mode and to attach interrupt every 10ms. Print "Hello" to serial port in each interrupt. Use serial monitor to check the result.

5. Make Arudino mega to generate 6 separate 1 kHz PWM signals with different duty cycles and attach interrupt every 10 ms at the same time. Print "Hello" to serial port in each interrupt. Use oscilloscope and serial monitor to check the result.

6.1.4 Final Rig

Stress Analysis

Stress analysis was performed with the inbuilt Finite Element Analysis (FEA) tool in Solid Edge. This was to ensure that the test rig could withstand our expected, worst case scenarios. The tests are as follows:

1. Frame and ring test. All linear actuators are at maximum load. The applied forces are vertical point loads of 500 N each, totaling in 3000 N.

2. Lid test. A load of 3000 N is evenly distributed along the contact points with the ring.

3. Lid test. Maximum possible torque is inflicted on the lid. This is when 3 out of 6 linear actuators are actuating maximum load of 500 N each and the rest, on the opposite side are mechanically locked. A total of 1500 N is evenly distributed along the contact points on half of the lid.

4. Find maximum accepted load if load condition exceeds yield stress.

Obtain Force References

To be able to obtain the force references of a gait cycle the corresponding OpenSim data file has to be sent in as input into the Simscape model. The SocketSense research team provided with an OpenSim data file which contained the forces and torques acting in the middle of the thigh of a 80 kg amputee during several gait cycles. Due to the fact that the coordinate systems of the OpenSim and the Simscape model didn't match the direction of some of the force and torque vectors had to be changed. It was also necessary to make the vectors shorter to only contain one gait cycle. These force and torque vectors were then sent as the input signal to the 6 DOF joint, which were placed below the top platform in the Simscape model since the OpenSim data was recorded from that position. As presented in 5.5.1 a spring is required in the prismatic joint to be able to calculate the forces acting on each leg as well as an damper to reduce the oscillations of the spring, and in this case they were chosen to be 10000 N/m and 1000 N/(m/s) respectively.

Since each vector consisted of approximately 150 data points the total simulation time of the Simscape model was set to 300 seconds and new signal from all vectors was sent every 2 seconds. This was done to make sure that each actuator reached steady state before changing the external force on the model.

Position Control

The position controller as well as the rig's capacity of performing a controlled motion were tested by generating several motion sequences using the Trajectory Planner and Inverse Kinematics block in Simulink. The different motion sequences consisted of movements along one single axis, two axes, rotation around one axis, and different combinations of the three. The maximum velocity of the actuators were set to 50 % to ensure smooth movements with just small vibrations of the test rig when ran. During the test the lid was not attached to the top platform and the load cells were calibrated to not read any loads initially. The tolerance for reaching the reference were set to 0.5 mm.

The result of the test was recorded by the Arduino and the resulting plots with the reference position and the actual position of each leg throughout the whole motion were showed in the GUI interface.

Force Control

Two different methods in obtaining the reference signals were used to test how well the test rig could apply force and how accurate, comparing to the reference force from the Simscape model. Two different objects under pressure were also used during the tests to examine the capacity of applying force on materials with different stiffness, namely the stump and a square wooden dowel rod. The square wooden dowel rod were cut to the same length as the distance between the base

frame and the lid of the test rig. It was also cut to fit into the welded square pipe on the lid, to secure it, see Figure 6.1. The rod was not attached to the bottom frame of the rig. The external force in the Simscape model were in this tests set to

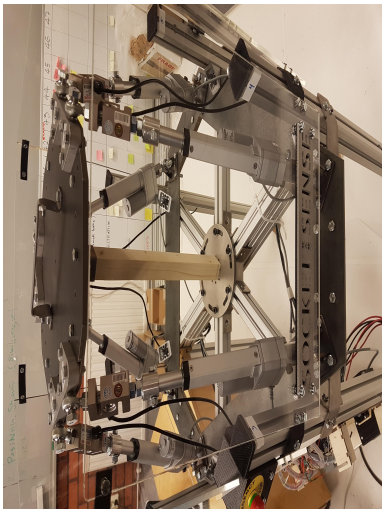


Figure 6.1: The set-up when having the wooden square dowel rod in the test rig.

be one constant force in the vertical direction applied in the 6-DOF joint beneath the top platform in the Simscape model. In this way the force reference for each motor could be obtained as described in section 5.5.1. In the test the maximum speed of the actuators were set to 50 % with the tolerance of ± 5 N for reaching the reference.

Instead of calculating the force references using the spring and damper system in the legs in the Simscape model it is also possible to create a model of the stump/rod, presented in section 5.5.2. To identify the stiffness of the stump and wooden rod the test rig were set to move to a desired position were the load cell sensor data was recorded. Both the position and the sensor data were then used as the reference to iteratively trying different spring coefficients and external forces in the Simscape model to make the model reach the same position and have close to same forces in each leg as recorded by the test rig. The chosen spring coefficient were then used to see how well the forces measured by the test rig and in the model matched each other in three different leg positions. These tests were made on both the stump and the wooden rod.

In this way it can be verified that our model can give the same force feedback but with two different types of reference.

Chapter 7

Results

7.1 Hardware Test

The results were positive, since there were no defective hardware components. All six motors, drivers, load cells and amplifiers were working properly.

The power supply was working fine, and the voltage could be decreased to the desired amount and the PCB connected to it was able to split the voltage to the six actuators equally. An interesting behaviour of the power supply was that while testing without any load connected to it, the voltage was static for at least 5 seconds after the switch off button was pushed, and then it started to reduce to zero.

Furthermore, a limitation of the Arduino that was visible from the tests was how fast it could process the position data from the potentiometer (sampling time). Without any delay in the code the readings had a small deviation from the actual position. When inserting a delay of 10 ms the deviation got smaller and with a 100 ms delay the position reading was the most accurate, as seen in the three graphs in Figure 7.1.

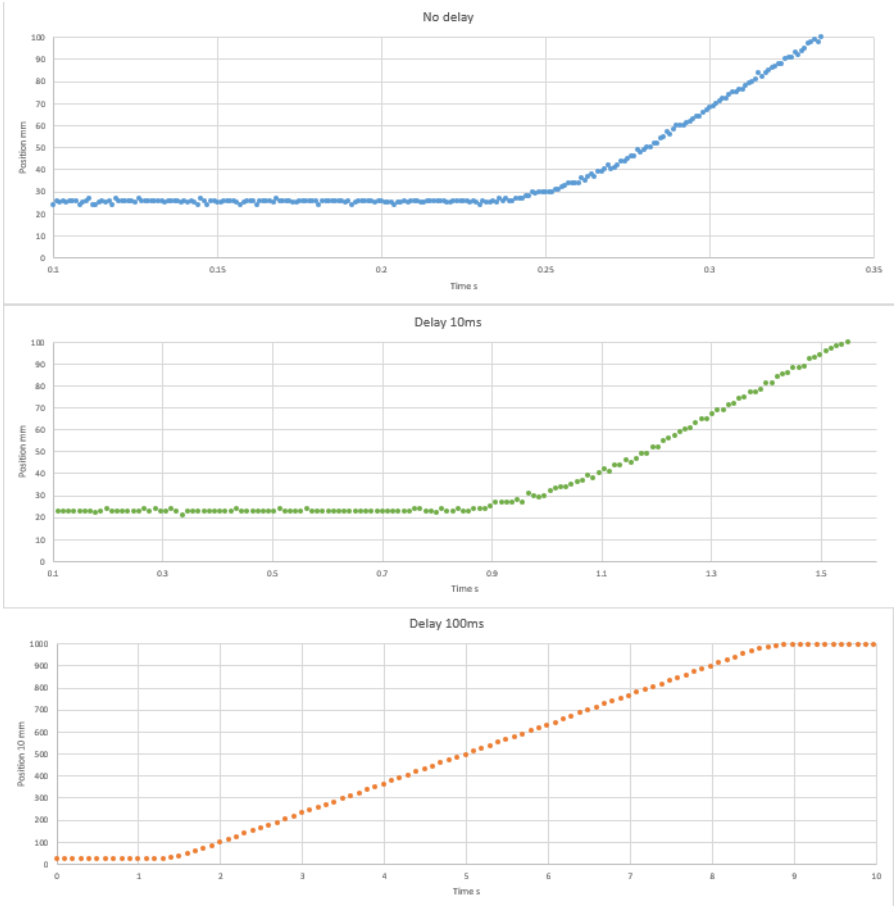


Figure 7.1: Position Data Test.

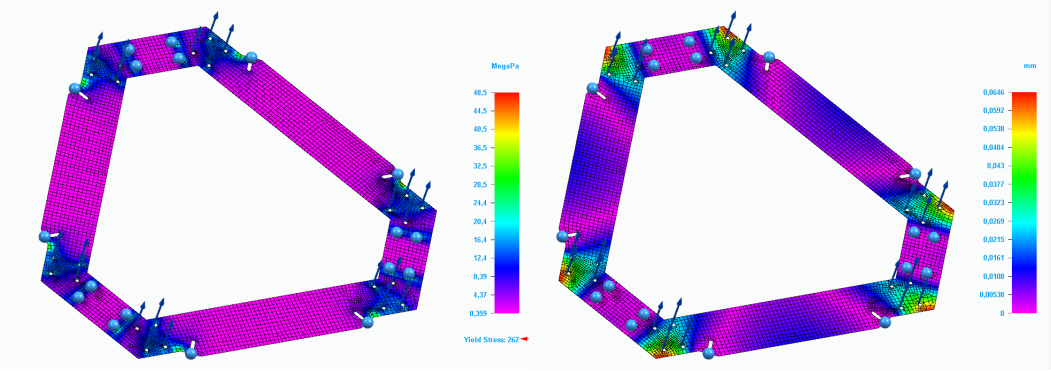
7.2 Software Test

Integration tests 1-7 were successful but integration test 8 was not. The Arduino managed to store data in a matrix (pointer to a list of pointers, which points to arrays of floats) but there was problems when using the data as reference values for the control of the actuators. Different methods for casting the data to appropriate data-types were used but did not solve the issue. Due to time-constraints an alternative solution was instead conceived where the reference-data from Simulink was imported via a Python-script and exported to a h-file. See Figure 5.23 for more details.

Arduino tests 1-5 were successful. Arduino mega could successfully use 4 timers to generate PWM signals and attach interrupt at the same time.

7.3 Stress Analysis

The first test was when maximum point loads were applied, with a total load of 3000 N. For simulated strain and displacement on the frame, see Figure 7.2a and 7.2b.

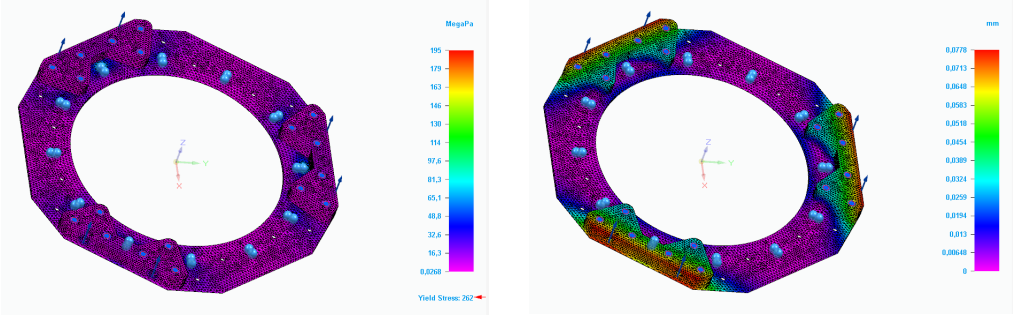


(a) FEA results of strain on the frame with 3000 N load.

(b) FEA results of displacement on the frame with 3000 N load.

Figure 7.2: FEA results of the frame with 3000 N load.

The same analysis of the strain and displacement of the ring can be seen in Figure 7.3a and 7.3b.

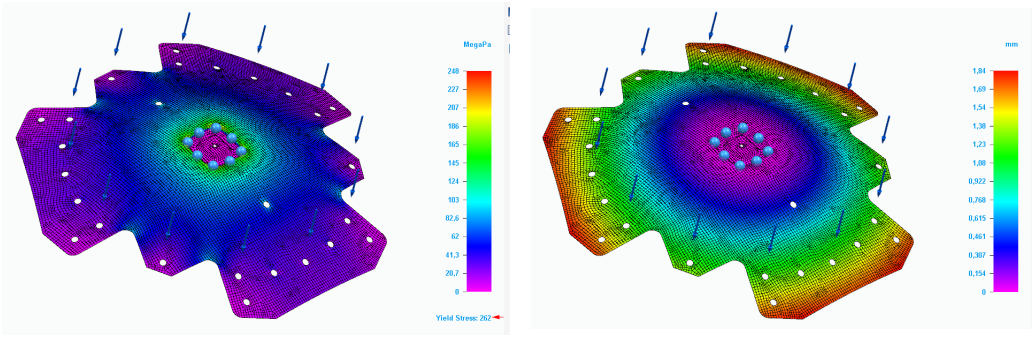


(a) FEA results of strain on the ring with 3000 N load.

(b) FEA results of displacement on the ring with 3000 N load.

Figure 7.3: FEA results of the ring with 3000 N load.

For the lid, two different scenarios were analyzed. First was that the load of 3000 N was evenly distributed along the contact points with the ring. For simulated strain and displacement see Figure 7.4a and 7.4b.

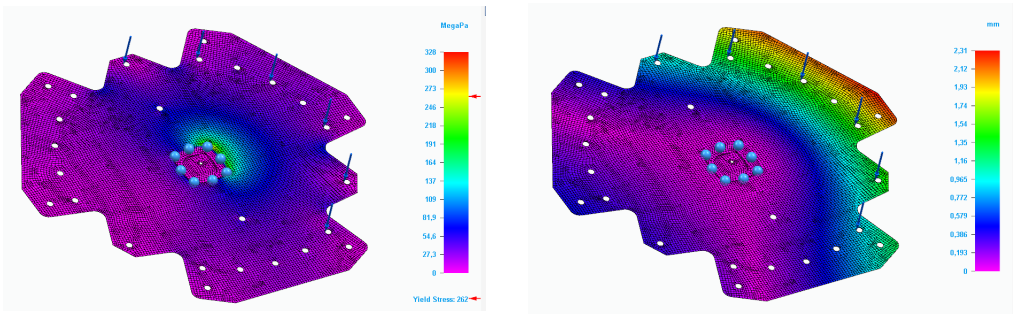


(a) FEA results of strain on the lid with 3000 N load.

(b) FEA results of displacement on the lid with 3000 N load.

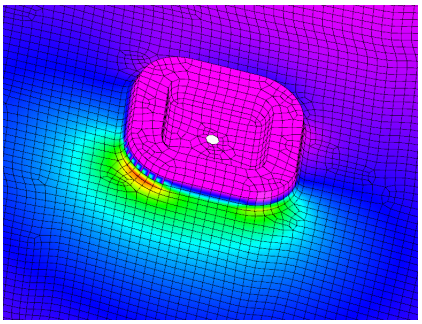
Figure 7.4: FEA results of the lid with 3000 N load.

The second test was when maximum possible torque was inflicted on the lid. For simulated strain and displacement see Figure 7.5a and 7.5b.



(a) FEA results of strain on the lid with 1500 N load.

(b) FEA results of strain on the lid with 1500 N load.



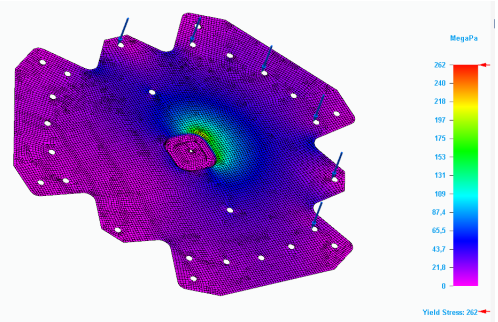
(c) FEA results of strain on the lid with 1500 N load, closeup on high stress area.

Figure 7.5: FEA results of the lid with 1500 N load on one side.

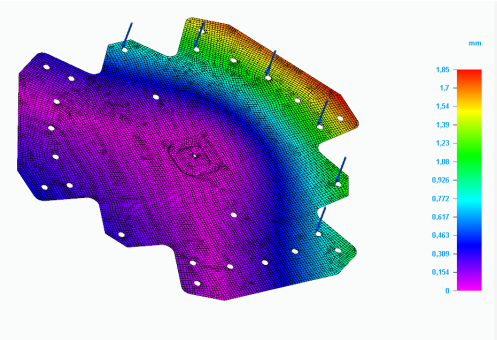
As seen in Figure 7.4a, the maximum stress was at 328 MPa. This was 125.1 % of

the materials yield stress. See Figure 7.5c for a close up of the simulated strain on the high stress area.

Further testing was done to see the maximum accepted forces at the same load condition. For simulated strain and displacement when the total load was 1200 N, see Figure 7.6a and 7.6b.



(a) FEA results of strain on the lid with 1200 N load.



(b) FEA results of displacement on the lid with 1200 N load.

Figure 7.6: FEA results of the lid with 1200 N load on one side.

This shows that the maximum allowed force is 1200 N, when the load was evenly distributed on one half of the lid.

7.4 Obtain Force References

The Simscape model properly divides the total forces and torques of the gait cycle from OpenSim between each leg to ensure that the exact same three-dimensional forces and torques will be applied on the top platform of the rig. In Figure 7.7 the forces of each leg are presented, showing how they change throughout the whole gait cycle.

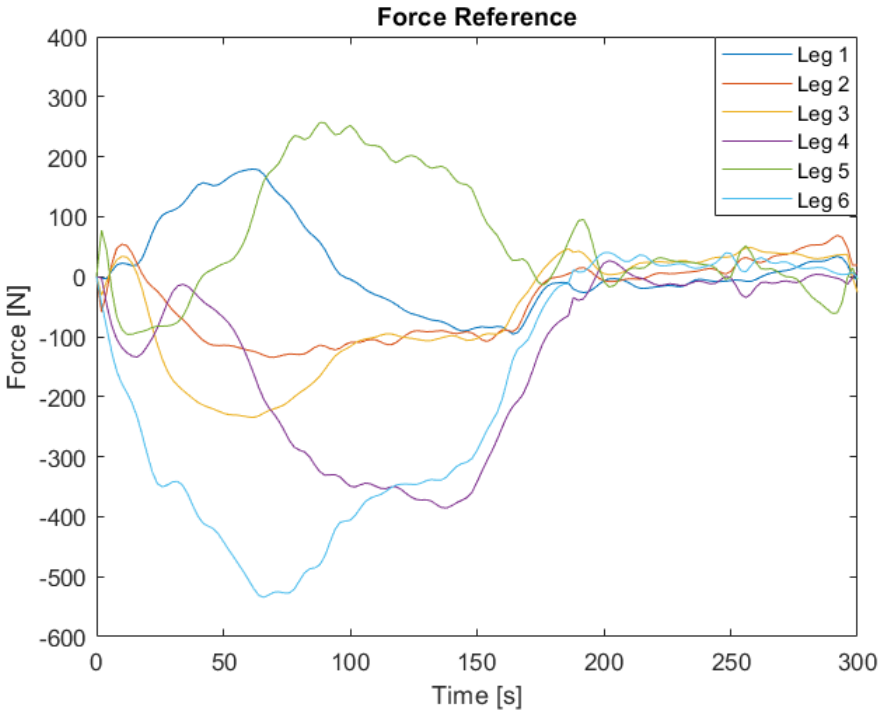


Figure 7.7: Force reference of each leg in the Simscape model.

7.5 Position Control

The Figure 7.8 is illustrating the movement of each leg of the test rig when performing a sequence of motions using position control. The orange graphs are the position references and the blue are the data from the potentiometers sensing the change in position used in the linear actuators. As can be seen in the figure, during different time periods some of the motors move in unison. This is due to symmetrical movements of the platform, like pure rotation and translation.

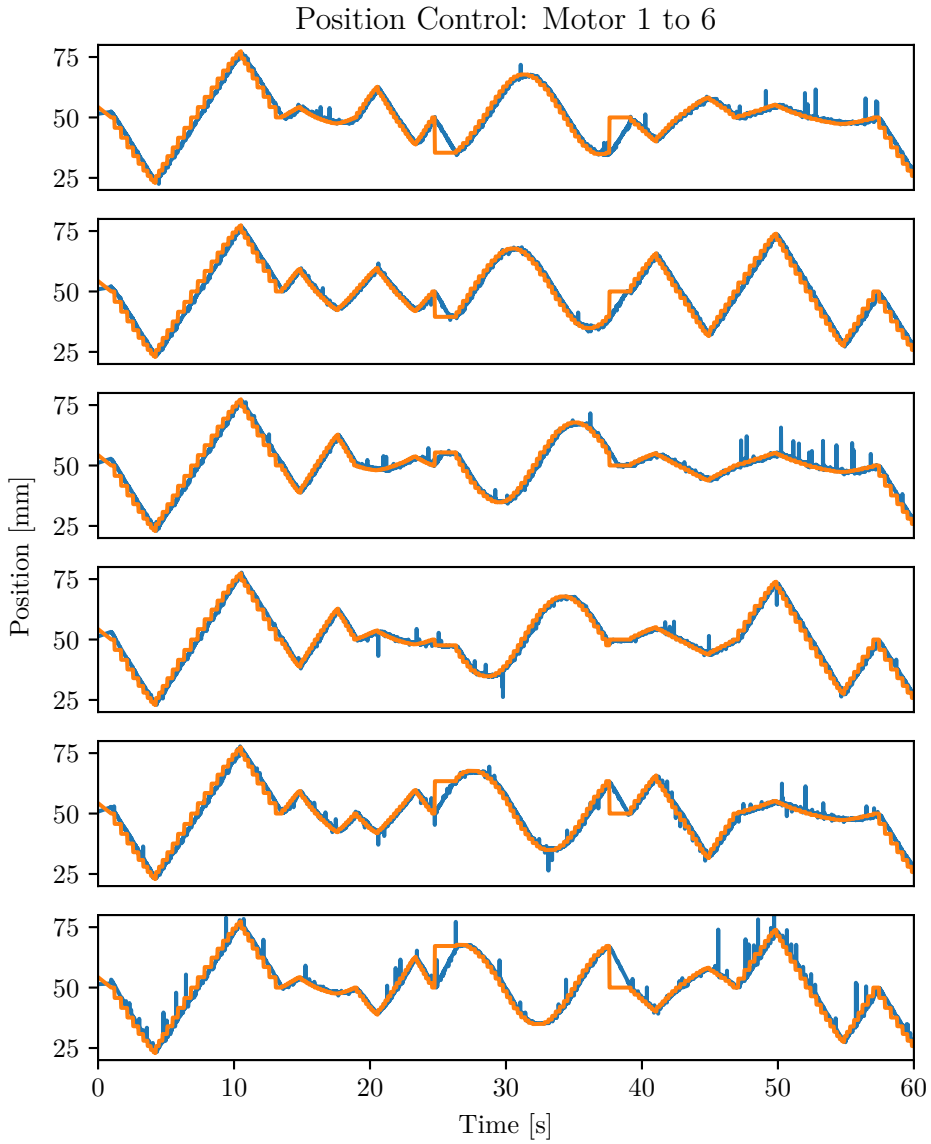


Figure 7.8: Position control of the platform without the lid attached. Tolerance for reaching the reference is ±0.5 mm. Max speed of actuators is set to 50 %. No reading of load cells in code. Average sample rate of position sensors is 100 Hz (due to timer interrupt with period of 10 ms). Position reference is shown in orange, position sensor reading is shown in blue.

To get a closer look on the position results, Figure 7.9 shows the movement of a 5 second period. As can be seen the reference points are given in discrete time since there are limited amount of reference points, and the blue graph is in the continuous time since the data is read in real time. As it can be seen the test rig follows the references follows the orange graph showing that the position control works.

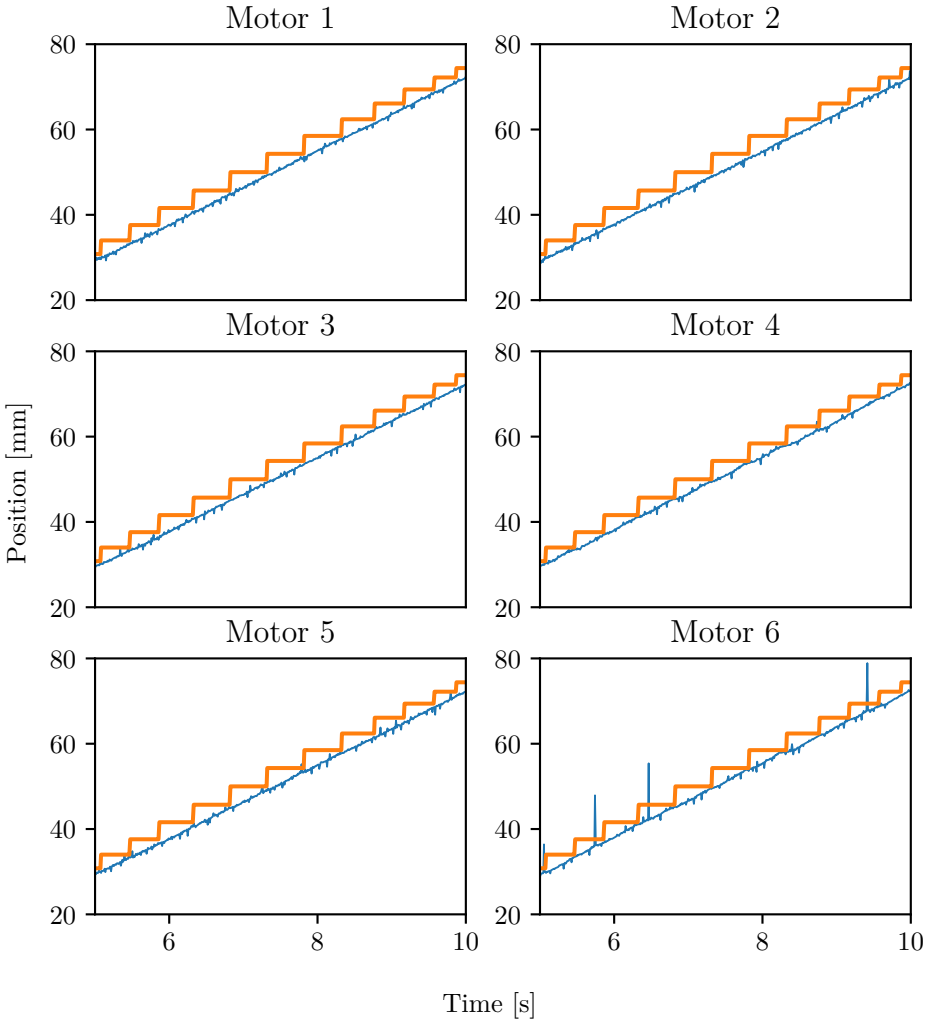


Figure 7.9: Same position control data as in Figure 7.8 but zoomed in.

7.6 Force Control

Two different methods of obtaining a reference signal which can be used to control the force of the test rig will be presented here.

7.6.1 Using Force Reference

This method consists of using the force reference data received from the Simscape model to make the test rig actuate the desired force onto the stump and wooden rod.

With Stump

Figure 7.10 shows the forces applied by each leg when the stump is inserted inside the socket. The figure shows that the legs moved in a oscillating manner with an amplitude of 200 N for most of the legs. It also seem to be some phase shift between the oscillating motions of the legs. There seem not to be any decrease in amplitude suggesting that the platform would not reach a steady state. No reading of actuator position was in code. Average sample rate of force sensors is 10 Hz. The motor oscillation frequency is approx. 0.75 Hz.

It was observed during the test that the constructed stump had varying stiffness in different directions. When attaching it to the socket it was also discovered that due to the unsymmetrical stump it was not perfectly aligned with the socket.

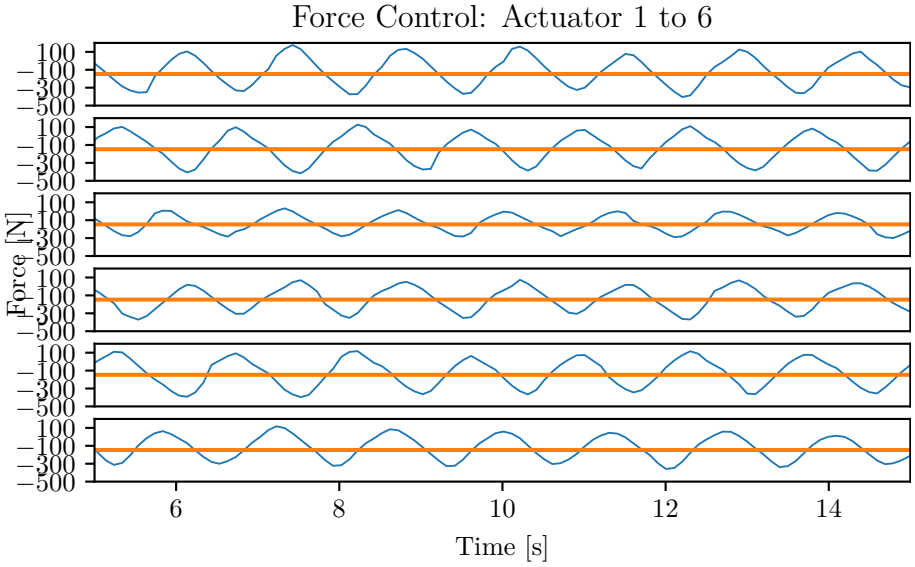


Figure 7.10: Force control of the platform with the stump attached. Tolerance for reaching the reference is ±5 N. Force reference is shown in orange, force sensor reading is shown in blue. Max speed of actuators is set to 50 %.

With square wooden dowel rod

The Figure 7.11 shows the forces applied by each leg when having the square wooden dowel rod attached between the base frame and top platform. The figure shows the rig to move in a oscillating manner, similar to when having the stump There seem not to be any decrease in amplitude suggesting that the platform would not reach a steady state. It was observed during the test that the wooden rod was not perfectly centered and a little bit angled from the vertical axis. Max speed of actuators is set to 50 %. No reading of actuator position in code. Average sample rate of force sensors is 10 Hz. The motor oscillation frequency is approx. 0.75 Hz.

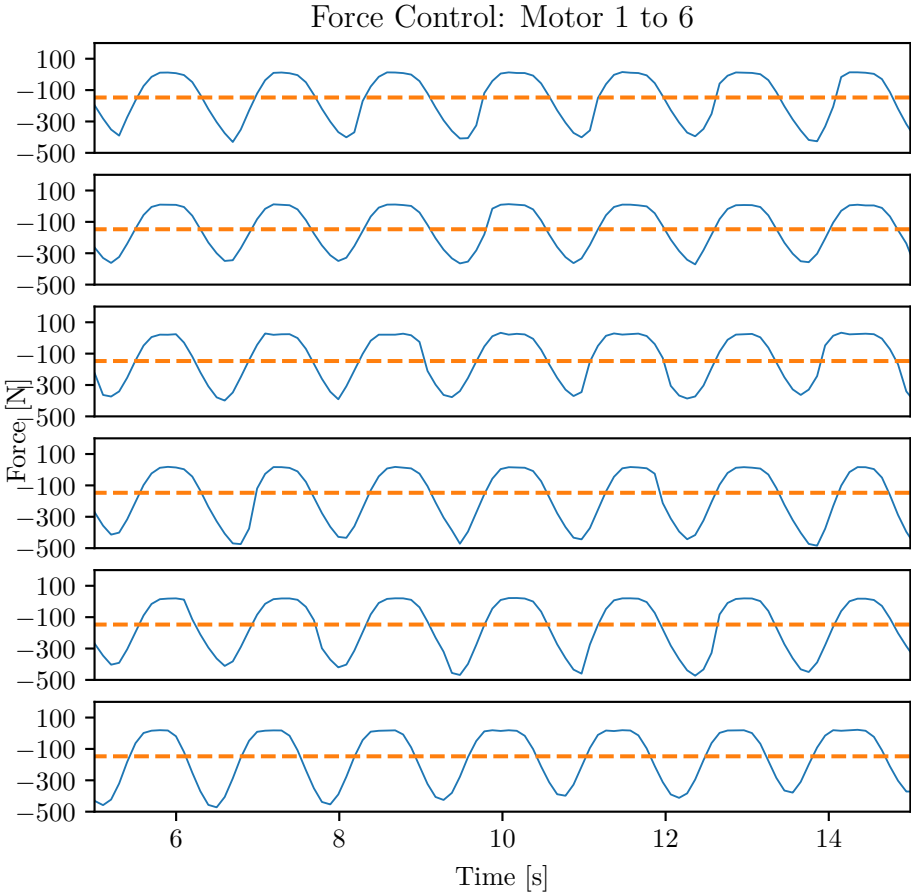


Figure 7.11: Force control of the platform with square wooden dowel rod attached. Force reference is the same for each motor. Tolerance for reaching the reference is ±5 N. Force reference is shown in orange, force sensor reading is shown in blue.

7.6.2 Using Position Reference

This method consisted of making a model out of the stump and the wooden rod to be able to use a position reference to actuated the test rig and make it apply the

desired force onto the stump and the wooden rod.

With Stump

To identify the vertical stiffness of the stump the test rig was set to go 5 mm along the negative z-axis, where the load cell data was registered at the start and the end. The position and load cell data were then plotted to show the relation between the leg extension and the force for these two data-points, illustrated by the blue lines going from the origin to the measured point in Figure 7.12 for each leg. The orange line represents the Simscape model coming sufficiently close to the desired point by having a spring coefficient of 4575 N/m.

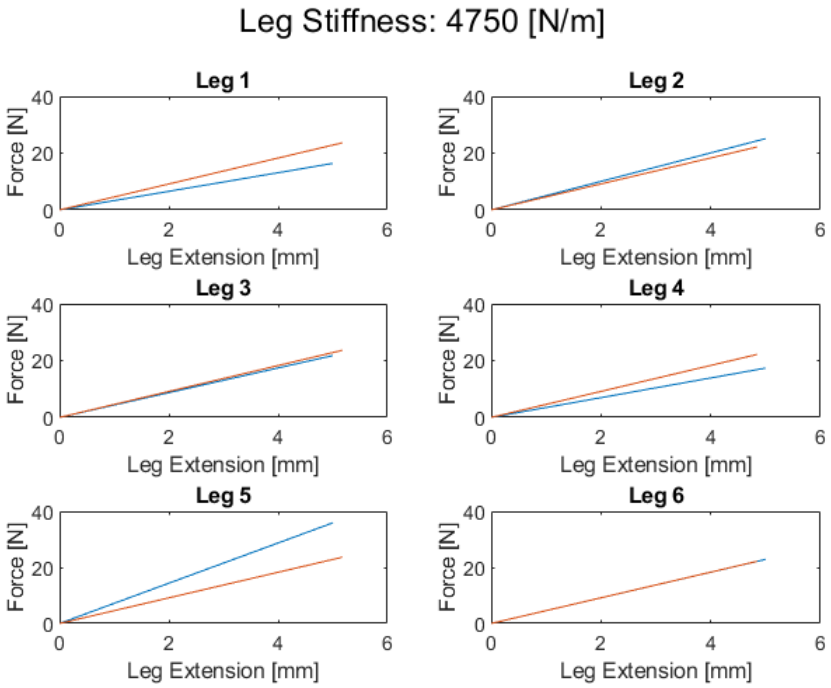


Figure 7.12: The absolute value of the force and extension of each leg in the test rig is given by the blue line. The orange line is the corresponding forces and leg extensions of each leg in the Simscape-model when having a leg stiffness of 4575 N/m and damping of 1000 N/(m/s) in the prismatic joint.

The test rig and Simscape model were then set to move to three new positions where the forces were registered for each leg, resulting in Figure 7.13, with the results from the test rig presented in blue and the Simscape-model in orange connected by a line.

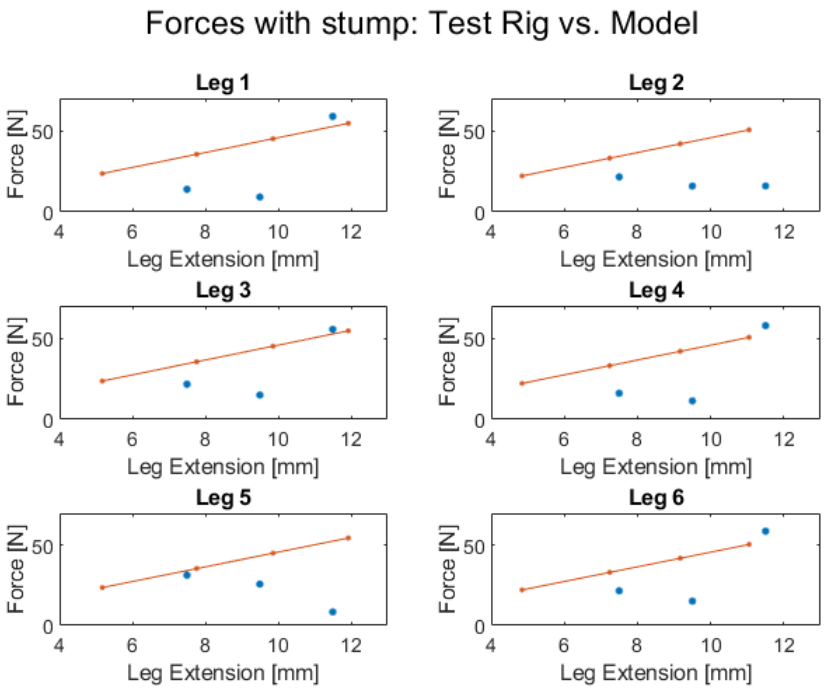


Figure 7.13: Relationship between the force and extension of each leg when using the stump, in the test rig and Simscape model.

With square wooden dowel rod

To identify the vertical stiffness of the wooden rod the test rig were set to go 4 mm along the negative z-axis, where the load cell data was registered at the start and the end. The position and load cell data were then plotted to show the relation between the leg extension and the force for these two data-points, illustrated by the blue lines going from the origin to the measured point in Figure 7.14 for each leg. The orange line represents the Simscape model coming sufficiently close to the desired point by having a spring coefficient of 77000 N/m.

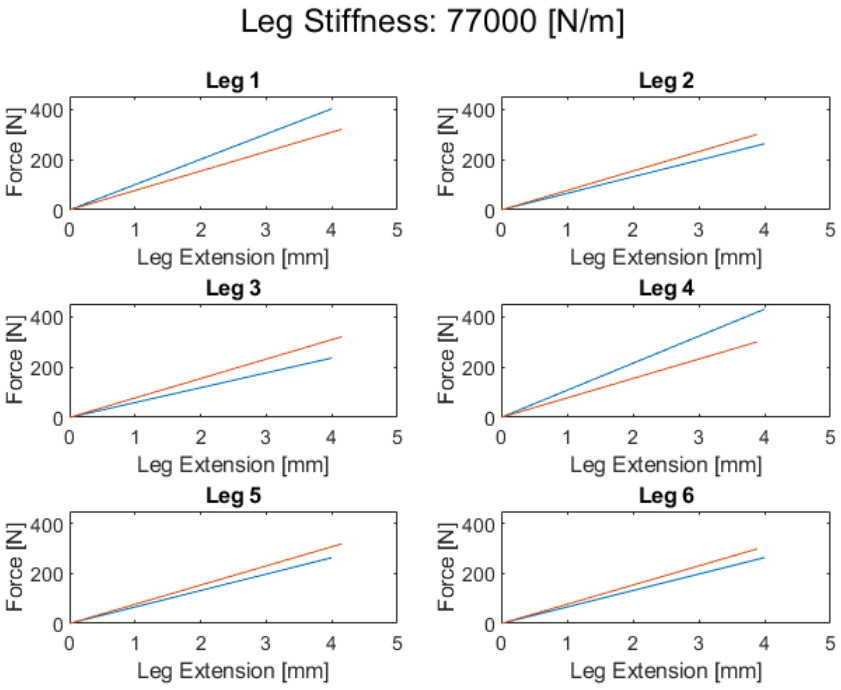


Figure 7.14: The absolute value of the force and extension of each leg in the test rig is given by the blue line. The orange line is the corresponding forces and leg extensions of each leg in the Simscape model when having a leg stiffness of 77000 N/m and damping of 1000 N/(m/s) prismatic joint.

The test rig and Simscape model were then set to move to three new positions where the forces were registered for each leg, resulting in Figure 7.15, with the results from the test rig presented in blue and the model in orange connected by a line.

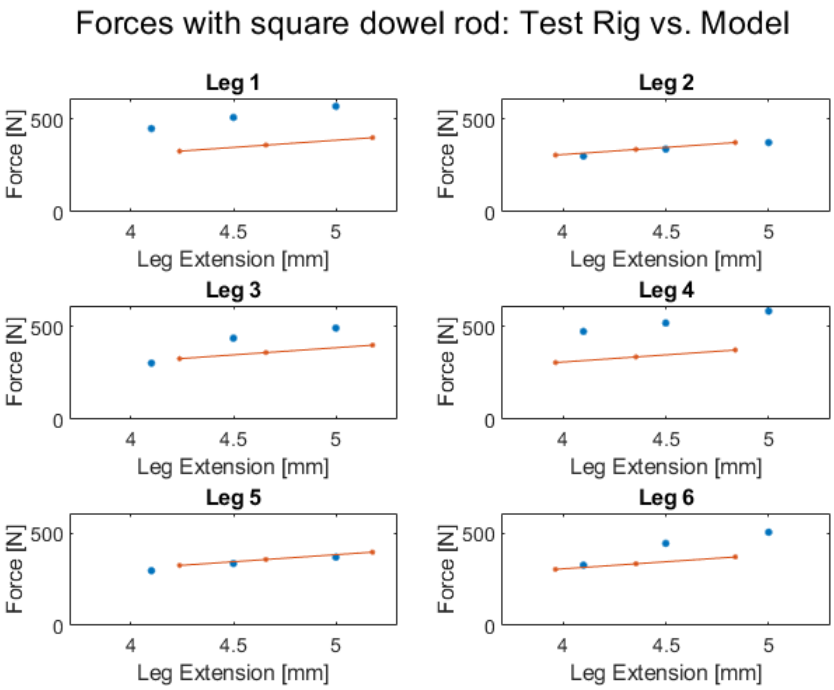


Figure 7.15: Relationship between the force and extension of each leg when using the square wooden dowel rod, in the test rig and Simscape model.

7.7 Stakeholder Requirements

The result of the project and the final test rig is presented in Figure 5.15. With regards to the requirements, the results were as follows:

> *1. A simulation-to-rig interface, compatible with OpenSim and MATLAB.*

The interface created to full fill this requirement can be seen in Figure 5.22.

> *2. Use an ischial socket with pin lock for testing.*

> *3. The socket and stump should be interchangeable.*

Both the second and third requirement are fulfilled, since the socket that was used where the one provided by the stakeholder and the rig has been designed for it to be interchangeable. The platform can be highered or lowered and the center of the socket can also be changeable.

> *4. The test rig should be able to recreate forces and torques for a gait cycle.*

The model of the test rig can identify the reference forces to recreate the forces and torques of the gait cycle. The test rig can apply forces and torques needed but not recreate the exact reference.

> *5. The test rig should be designed assuring the possibility for expanding of the rig with an additional 6 DOF platform for the socket, and also investigating other scenarios like jumping, running, and climbing.*

As seen in Figure 4.7, a design is proposed for how the current 6 DOFs rig can be changed to have in total 12 DOFs. The main point is that when changing from 6-12, the rig is duplicated at the bottom.

> *6. The rig should be able to make motions and all the motors should be able to move at the same time.*

The test rig can both be controlled by position and force. Making all the motors move at once, which means that the rig is able to perform motions. Hence, the requirement is full filled.

Chapter 8

Discussion and Conclusions

8.1 Design

The design was functional and had the option of expandability. The aluminium extrusion worked as indented, both in carrying the test rig and in fixating additional components. The test rig was fairly compact and not unnecessarily heavy. The assembly was quick and can be disassembled if needed, i.e welds was carefully placed not to lock the parts.

When testing the rotation of the platform, it was found that the load cells can collide with the lid in the most extreme angles. This was missed in the CAD since the full constraints of the rod ends could not be made.

When the actuators applied forces close to their maximum, it was observed that the structural legs started bending outwards. This was somewhat expected, since there was speculation if another frame was needed near the feet. However, the second platform's frame is planned to be mounted in that exact location, providing the needed support.

In the stress analysis of the test rig, most tests passed except one. The load scenario when maximum torque was applied lead to an area of stress that exceeded the yield stress of the material. This should mean that the material will have plastic (permanent) deformation in that area at this load condition.
The high stress area is the weld between lid and stump. The test results shown in Figure 7.6 shows that cyclic loading above 1200 N in this scenario would increase the risk of fatigue in the high stress area [65]. However, these loads are not realistic when actuating a gait cycle.

8.2 Electronics

The electronic components, for example the motor drivers, load cells and amplifiers worked fine for the purpose of the test rig. All these components were connected with a Arduino Mega with arduino-type-connector wires. These wires were not soldered on to the components which had a significant effect on the rig by redacting the overall robustness and accuracy in the readings. Because of some occasionally high vibrations, the wired connections to the Arduino Mega got loose and resulted in some actuators not working. This lead to additional time troubleshooting or re-checking the wires. This is why all the power and ground wires from all 18 components were soldered on a breadboard to avoid short circuits or open circuits.

8.2.1 Arduino Mega

Furthermore, some unexpected issues with the micro-controller were raised. One problem was that when powering the Arduino with an external power supply of 7 V it had issues connecting to the computer as well. Typically there is no issue to upload the code or monitor the data from the computer while powering the controller externally. Another difficulty was when trying to read both position and force readings at the same time. This could indicate that the controller does not have the appropriate process unit or memory to handle large amount of incoming data at the same time.

Another reason for why there was difficulty reading forces-sensor data at a desirable rate could be due to misconfiguration of the load cell amplifier HX711 and the external clock from the Arduino. As the results show the rate at which the data was being transferred from the load cell amplifiers to the Arduino is almost exactly 10 Hz. This is the rate for which the internal oscillator of the amplifier have by default, it is possible to use either an internal oscillator as clock for data transfer timing or an external. Somehow during testing of the components there could have been a misinterpretation that the external clock was in use when in fact the internal one was. Due to time-limits no investigation of this was done.

8.3 Software

Due to many changes (caused by new insights from integration testing) the decision for a final system got delayed, which in turn lead to the group deeming that a TDD-approach using a testing framework would possibly lead to more delays rather than reducing development time. The test framework Ceedling (Unity based framework for C development) [66] was proposed at a late stage but was deemed to have a steep learning curve, which in turn would take more time from actual development of the software. It was thus decided to test the C-code on the actual hardware (test one change/feature at a time) rather than use a TDD-approach with mock-ups.

Simulink Testing toolbox was investigated as a possible tool to use when testing the Simulink model [67]. Due to changes in where the control of the actuator would be (sensor-feedback from Arduino that is piped to Simulink-model via serial communication, to doing all of the control in the Arduino) Simulink Testing toolbox was deemed not critical.

8.4 Ischial Socket - Ischial Tuberosity

During the development of this rig, one important aspect was discovered with the ischial socket. The ischial tuberosity is a small support which the pelvis is resting upon during stance. This support is actually taking a significant amount of the users weight, and thus reducing the forces between the socket and stump. Unfortunately, this rig cannot simulate both the femur and the pelvis since that would require an additional actuating platform with (up to) 6 DOF. However, if OpenSim is able to isolate the stump's force from the pelvis', this rig could probably simulate them both separately. This does not affect an Quadrilateral socket, since all force is on the femur/stump.

8.5 Stump Design

The stump's deformation and behavior is not linear which caused problems when applying force control on it. The non-linearity can be seen when comparing Figure 7.13 and Figure 7.15. After discovering the non-linearity of the stump the Simscape model was used to calculate the approximate stiffness of the stump along vertical direction. Applying a known position reference to the motors, and reading the load cell data, the stiffness could in theory be changed in the Simscape model, until it reached the same position and same force both in the model and in the real rig. That way a force controller could be designed using both position and force data, making it into a cascade control. But even for a vertical motion the stump showed a non-linear behavior.

8.6 Control

The results showed that position control worked well. Some small noise was apparent in the position readings of the actuators, e.g. see Figure 7.9. This high frequency noise could be reduced by a low-pass filter (RC-circuit). The force control was unfortunately not as successful. In Figure 7.10 and 7.11 it can be seen that all motors are oscillating with the same frequency (about 0.75 Hz) while trying to reach the reference value. There's also a slight phase offset between the actuators in Figure 7.10 where actuator 1, 3, 4 and 6 are in phase and 2 and 5 are about 180° out-of-phase. The reasons for this could be many. One reason could be the interplay between the low sampling rate (10 Hz) and geometry of the placement of actuators. Since the actuators are constrained to the same top platform, the change in position

of one actuator will affect the pressure distribution of the whole platform, leading to changes in force sensor reading for each. This effect was expected and could be countered by e.g. setting reasonable tolerances for when a force reference is reached. A slight phase offset in one of the actuators could lead to a domino effect of pulling and pushing, where one actuator is pulling its neighboring actuators out of the acceptable reference value. These actuators will then compensate it, which will affect its neighboring actuators in turn etc. This ongoing tug-of-war would then find its resonating frequency, in this case it is 0.75 Hz.

This kind of behavior was expected but the assumption was made that if the stiffness of the stump was sufficiently high then the movement of each leg, required for simulating the force, would be constrained to be relatively small. That would result in errors small enough to reduce observed oscillations.

Sending one position reference and measuring forces in each motor with first attached wooden dowel and then attached stump the approximation of the stiffness of these two parts was made. It was observed that the forces are approximately linear when using the wooden dowel which meant that the load cells are working as expected, as shown in Figure 7.15. It was also observed during the test that the wooden rod was mounted slightly skewed. This probably caused the distinction of the forces between the Simscape model and the test rig in Figure 7.15. It could also be observed a similarity of the forces applied by the actuators mounted in diagonal to each other, for example 2 and 5. Given more time and resources it is probable that the force control will work as well after optimizing the controller code, improving the simulation model, improving the physical test stump and increase data rate of force sensors.

Chapter 9

Future Work

9.1 Design

There are several available options in order to reduce the occurrence of vibrations of the rig. One option is to add dampers below each feet of the rig.

To avoid the load cells colliding with the lid in the extreme angles, one solution could be to flip the actuating legs. Since the rod ends are identical, the functionality would stay the same. However, the cords of the linear actuators was very short. If flipping is desirable, the cords needs to be extended or the electrical box needs to change location.

9.2 Electronics

As future work it is suggested that a more permanent configuration of the electronics is made. During this project a lot of things were changing and separate tests had to be done after the assembly of electronics, so they were kept in an easily modified configuration. Although, to have a more robust and reliable product a large PCB could be made were all the components can be externally attached to, and the wires will be paths on the PCB instead of physical wires. This way it will be less messy and avoid any noise in the readings or EMC from crossing wires.

The wires from the load cells to the load cell amplifiers are at the moment quite long and are winded up in loops. This, in combination with low operating currents ($<$1.5 mA) could lead to susceptibility of external noise. One should shorten the wires to reduce this risk. There's a possibility of connecting a shield from the load cells to the HX711 yellow/SD input. By connecting it further noise could be reduced.

In order to increase the data rate at which the Arduino receives force data from the load cell amplifiers future work should be to investigate if one can increase the data rate by either increasing the rate of the internal oscillator or by configuring

the external clock to work properly. According to HX711 datasheet the jumper connected to the RATE pin can be opened. This will set the input of RATE pin to DVDD (or 1), which according to the datasheet will increase the data rate from 10 Hz to 80 Hz. This could have significant impact on the performance of the force control of the platform.

The reading of force sensors when using the stump, showed that at a default position of 50 mm, each motor differed significantly from one another. The reading could differ more than ± 5 kg sometimes between the highest and lowest value. The reason for this is that after running the test rig several times, the strain gauges may not have been zeroed and an additional load is calculated. The current solution is to record the force of each strain gauge and offset the reading of strain gauges to a defult value. However, this is not enough. In the future a more advanced algorithm should be implemented to eliminate additional load. Furthermore, in order to verify the performance of the test rig, it was tested many times, including some failure, which may affect the load cell. Thus, before running the test rig, the strain gauges should be re-calibrated by tuning the calibration factors.

9.2.1 Emergency Button

The purpose of the emergency button was to ensure a quick power shut down in case of any unpredicted behavior and to prevent injuries. It was observed that the power supply had lingering power output (that gradually annihilated) when turning off the input power. This discovery made the emergency button to be redundant, and was thus left unplugged (the power supply still had a on/off switch). If the Emergency button is needed in future, it could be wired between the output of the power supply and the voltage splitter for the motor drivers. Thus acting as a instant stop of power. However, a different button needs to be acquired that can handle the power supply's output current.

9.2.2 Pressure Sensors QTSS

Due to changes in the project description the team never worked with utilizing QTSS sensors in order to collect pressure data from inside the socket. Instead another team of engineer students at KTH will design and implement a solution for collecting pressure data. It would have been interesting though to use the QTSS sensors in order to validate applied forces on the rig and to compare the result with load cell values.

9.3 Stump

To ensure a proper stump with the right qualifications and characteristics, and to ensure a perfect fit within the socket. A suggestion for future work is to get a matching socket and stump from the stakeholder. Preferably a stump with a linear

behavior to be able to control it and foresee its behavior when putting on the total amount of force according to the reference.

9.4 Cascade Control

In this project, position control and force control are applied independently to the test rig. However, force control was not properly implemented in the test rig and only position control cannot prove the accuracy of force. Thus, in the future, if more accuracy is acquired, a cascade control system can be implemented. However, the cascade control can be designed when the approximate stiffness of stump is known. The inner loop of the cascade control is position control and the outer loop is force control. Figure 9.1 shows the structure of the cascade control system.

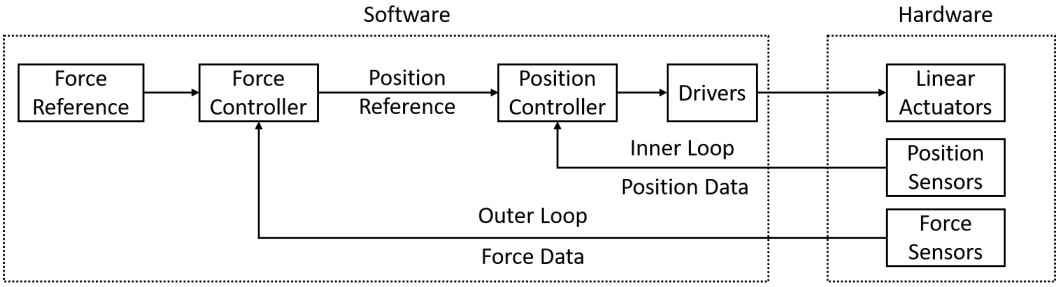


Figure 9.1: Cascade Control System.

9.5 MIMO Control

Since Stewart Platform consists of six actuators, and both position and force control have feedback data from six sensors, it is logical to implement a Multiple Input-Multiple Output (MIMO) control system. That kind of control system is mainly used when the output of one part of the system is affecting the input of other part. In this case the force that is acting on a specific force sensor is a result of both actuator attached to it and also the other actuators that are attached to the top platform. Since the actuators are affecting each other and are causing a disturbance, each motor cannot be controlled as an individual system as it was assumed in the beginning.

9.6 Software

If the current interface is to be used in the future, it should be put under test for usage in Windows and Linux environments. Currently the interface has only been tested in MacOS. Especially the Python virtual environment could differ in its functionality (or not work at all) in other environments.

OpenSim has extensive capabilities of being interacted with using either Python or MATLAB scripting. If further integration of OpenSim needs to be implemented in the interface, it is recommended to read upon it in OpenSim's Confluence documentation web page [68].

Bibliography

[1] S. Burt. "Facts About Limb Loss". In: (2018). URL: https://www.sralab.org/research/labs/bionic-medicine/news/facts-about-limb-loss.

[2] L. Paternò et al. "Sockets for Limb Prostheses: A Review of Existing Technologies and Open Challenges". In: *IEEE Transactions on Biomedical Engineering* 65.9 (2018), pp. 1996–2010.

[3] H. E. J. Meulenbelt et al. "Skin problems in lower limb amputees: A systematic review". In: (2005). URL: https://www.tandfonline.com/doi/epub/10.1080/09638280500277032?needAccess=true.

[4] L. E. Pezzin et al. "Use and satisfaction with prosthetic limb devices and related services". In: (2004). URL: https://www-sciencedirect-com.focus.lib.kth.se/science/article/pii/S0003999303008967.

[5] K. Hagberg et al. "Socket Versus Bone-Anchored Trans-Femoral Prostheses: Hip Range of Motion and Sitting Comfort". In: (2005). URL: https://journals-sagepub-com.focus.lib.kth.se/doi/full/10.1080/03093640500238014.

[6] K. Hagberg, O. Hägg, and R. Brånemark. "Questionnaire for Persons with a Transfemoral Amputation (Q-TFA): Initial validity and reliability of a new outcome measure". In: (2004). URL: https://www.rehab.research.va.gov/jour/04/41/5/pdf/hagberg.pdf.

[7] G. M. Benke et al. "Comparison of satisfaction with current prosthetic care in veterans and servicemembers from Vietnam and OIF/OEF conflicts with major traumatic limb loss". In: (2010). URL: https://pdfs.semanticscholar.org/7184/73b94465e60b0a23e8e7f199da13f94ac4e8.pdf?_ga=2.81628887.1966126157.1603821692-2063417000.1599772157.

[8] L. E. Pezzin et al. "Use and Satisfaction with Prosthetic Devices Among Persons with Trauma-Related Amputations: A Long-Term Outcome Study". In: (2001). URL: https://journals.lww.com/ajpmr/Fulltext/2001/08000/Use_and_Satisfaction_with_Prosthetic_Devices_Among.3.aspx.

[9] *Gait - physiopedia*. May 2020. URL: https://www.physio-pedia.com/Gait.

[10] S. Paterno et al. “Sockets for Limb Prostheses: A Review of Existing Technologies and Open Challenges”. In: (2018). URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8267041.

[11] *Lower Limb Prosthetic Sockets and Suspension Systems*. Apr. 2020. URL: https://www.physio-pedia.com/Lower_Limb_Prosthetic_Sockets_and_Suspension_Systems.

[12] *QTSS*. May 2020. URL: https://www.quantumtechnologysupersensors.com/.

[13] J. Danmo and A. Moustafa. “Wearable Sensors in Prosthetic Socket”. MA thesis. KTH, 2019.

[14] *Strain Gauge*. Nov. 2020. URL: https://en.wikipedia.org/wiki/Strain_gauge.

[15] H. D. Taghirad. *Parallel Robots*. CRC Press, 2013. ISBN: 9781466599284. URL: https://learning.oreilly.com/library/view/parallel-robots/9781466555778/.

[16] Y. Jin, H. Chanal, and F. Paccot. “Parallel Robots”. In: (2015). URL: https://link-springer-com.focus.lib.kth.se/referenceworkentry/10.1007/978-1-4471-4670-4_99#citeas.

[17] Y. D. Pateli and P. M. Georege. “Parallel Manipulators Applications - A Survey”. In: (2012). URL: https://www.researchgate.net/publication/235456946_Parallel_Manipulators_Applications-A_Survey.

[18] W. Guo et al. “Kinematics, dynamics, and control system of a new 5-degree-of-freedom hybrid robot manipulator”. In: (2016). URL: https://journals.sagepub.com/doi/full/10.1177/1687814016680309.

[19] M. Furqan, M. Suhaib, and N. Ahmad. “Studies on Stewart platform manipulator: A review”. In: (2016). URL: https://link-springer-com.focus.lib.kth.se/content/pdf/10.1007/s12206-017-0846-1.pdf.

[20] F. Thomas and J. Borras. “On the Primal and Dual Forms of the Stewart Platform Pure Condition”. In: (2012). URL: https://www.researchgate.net/publication/256701550.

[21] *6-RSS Parallel Platform*. May 2020. URL: http://www.nugenix.co.in/stewart-force-platform.

[22] E.F Fichter, D.R Kerr, and J. Rees-Jones. “The Gough–Stewart platform parallel manipulator: a retrospective appreciation”. In: (2008). URL: https://journals-sagepub-com.focus.lib.kth.se/doi/pdf/10.1243/09544062JMES1137.

[23] M. Walper et al. “A Stewart platform for precision surgery”. In: (2003). URL: https://journals-sagepub-com.focus.lib.kth.se/doi/pdf/10.1191/0142331203tm092oa.

[24] K.W Grace et al. “A Six Degree of Freedom Micromanipulator for Ophthalmic Surgery”. In: (1993). URL: https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=292049.

[25] G. Brandt et al. “CRIGOS: A Compact Robot for Image-Guided Orthopedic Surgery”. In: (1997). URL: https://ieeexplore-ieee-org.focus.lib.kth.se/stamp/stamp.jsp?tp=&arnumber=809169.

[26] S. Qian et al. “A Review on Cable-driven Parallel Robots”. In: (2018). URL: https://link.springer.com/article/10.1186/s10033-018-0267-9.

[27] A. Pott and V. Schmidt. “On the Forward Kinematics of Cable-Driven Parallel Robots”. In: (2015). URL: https://www.researchgate.net/publication/282849057.

[28] S. Swei P. Gao. “A six-degree-of-freedom micro-manipulator based on piezoelectric translators”. In: (1999). URL: https://iopscience-iop-org.focus.lib.kth.se/article/10.1088/0957-4484/10/4/315.

[29] D. Chao and R. Liu G. Zong. “Design of a 6-DOF Compliant Manipulator Based on Serial-Parallel Architecture”. In: (2005). URL: https://ieeexplore.ieee.org/document/1511075.

[30] K. Cai et al. “Modeling and controller design of a 6-DOF precision positioning System”. In: (2017). URL: https://www-sciencedirect-com.focus.lib.kth.se/science/article/pii/S0888327017305836.

[31] D. Verdes et al. “Kinematics analysis, Workspace, Design and Control of 3-RPS and TRIGLIDE medical parallel robots”. In: (2009). URL: http://myexs.ru/wp-content/uploads/2011/09/Kinematics-analysis-workspace-design-and-control-of-3-RPS-and-TRIGLIDE-medical-parallel-robots.pdf.

[32] W. Li and L. Sheng. “Optimization Design by Genetic Algorithm Controller for Trajectory Control of a 3-RRR Parallel Robot”. In: (2017). URL: https://www.mdpi.com/1999-4893/11/1/7/htm.

[33] *Actuator*. May 2020. URL: https://en.wikipedia.org/wiki/Actuator.

[34] Firgelli Automations Team. *Linear Actuators 101 - Everything you need to know about a linear Actuator*. URL: https://www.firgelliauto.com/blogs/news/linear-actuators-101.

[35] *Advantages of linear actuators in mechanical equipment*. May 2020. URL: http://www.gemingag.com/newsdetail/advantages-of-linear-actuators-in-mechanical-equipment.html.

[36] *Engine*. May 2020. URL: https://en.wikipedia.org/wiki/Engine.

[37] Caroline Fritz. *Definition of a DC Motor*. URL: https://sciencing.com/definition-of-a-dc-motor-13409319.html.

[38] *Why use DC motors instead of AC motors?* May 2020. URL: https://www.metmotors.com/use-dc-motors-instead-ac-motors/.

[39] *Advantages/Disadvantages of DC Motor*. May 2020. URL: http://eengineerqa.blogspot.com/2015/05/advantagesdisadvantages-of-dc-motor.html.

[40] *Brushless DC electric motor*. May 2020. URL: https://www.andmotor.com/brushless-dc-electric-motor/.

[41] *What Is the Difference Between a Stepper Motor and Servo Motor?* May 2020. URL: https://gesrepair.com/difference-stepper-motor-servo-motor/.

[42] *Stepper vs Servo*. May 2020. URL: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/stepper-vs-servo/.

[43] *2-phase and 5-phase Stepper Motor Comparison*. May 2020. URL: https://www.orientalmotor.com.sg/om/technical/stepper-motors/2-phase-vs-5-phase-stepper-motors.html.

[44] *Brushless DC electric motor*. May 2020. URL: https://www.andmotor.com/brushless-dc-electric-motor/.

[45] *A Servo Motor with a Controller: How the Two Fit Alongside*. May 2020. URL: https://rozum.com/servo-motor-controllers/.

[46] *Structure of Stepper Motors*. May 2020. URL: https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-overview.html.

[47] S. Von Schmalensee et al. "SocketSense - Final report". In: (2019). URL: https://kth.app.box.com/s/7i1dixgf9n86vppd0mgjh5qwbswqv8hi.

[48] *VIVO six D.O.F joint simulator*. May 2020. URL: https://www.amti.biz/vivo.aspx.

[49] *scrum*. May 2020. URL: https://www.scrum.org/resources/what-is-scrum.

[50] *agile*. Oct. 2011. URL: https://www.pmi.org/learning/library/agile-project-management-scrum-6269.

[51] J. Gausemeier and S. Moehringer. "Vdi 2206- a new guideline for the design of mechatronic systems." In: (2012). URL: https://www.researchgate.net/figure/The-V-Model-according-to-guideline-VDI-2206-VDI-2004-Gausemeier-and-Moehringer-2003_fig3_320729709.

[52] Oleksandr Stepanenko. *Serial robot vs. Parallel robot*. URL: https://www.youtube.com/watch?v=3fbmguBgVPA.

[53] The Verge. *Ping-pong playing robot vs human*. URL: https://www.youtube.com/watch?v=IXyKLDNzGGI.

[54] Andrej Rajnoha. *CableEndy juggler - cable-driven parallel robot*. URL: https://www.youtube.com/watch?v=7HNAL8ZKdyM.

[55] *Transmotec-Datasheet-DLA-POT*. URL: https://transmotec.se/Download/Datasheets/Transmotec-Datasheet-DLA-POT.pdf.

[56] URL: https://www.electrokit.com/produkt/motordrivare-vnh3sp30-1-kanal-5-5-36v-30a/?gclid=Cj0KCQjwit_8BRCoARIsAIx3Rj5w-cprOOgBvQ31fvaeOZbwbj50um_Sw_SFUurAjdeOZD-mYMzuxwwaAmniEALw_wcB.

[57] *VETEK-Datasheet-VZ101BH*. URL: https://www.vetek.se/Dynamics/Documents/7b99537e-c9f7-444a-881d-4f075a97e248/Datasheet_101BH_V4.pdf.

[58] URL: https://www.sparkfun.com/products/13879.

[59] *MeanWell SP-500-27*. URL: https://www.mouser.se/datasheet/2/260/mean%5C%20well_sp-500-spec-1179977.pdf.

[60] URL: https://www.google.com/search?q=arduino+mega&sxsrf=ALeKk00eUSkVb-Ze10K6ncFOoyjBgOjwLA:1607700156543&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjGofHlncbtAhUVHHcKHcikD9QQ_AUoAXoECBEQAw&biw=1536&bih=682#imgrc=1lAFmpYg5NqgMM.

[61] *Arduino 101: Timers and Interrupts*. URL: https://www.robotshop.com/community/forum/t/arduino-101-timers-and-interrupts/13072.

[62] *ATmega640/V-1280/V-1281/V-2560/V-2561/V 8-bit Microcontroller with 16/32/64KB In-System Programmable Flash DATASHEET*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf.

[63] R. Toulson. *Fast and Effective Embedded Systems Design*. Elsevier Ltd., 2017. ISBN: 9780081008805. URL: https://www.sciencedirect.com/book/9780081008805/fast-and-effective-embedded-systems-design.

[64] *Arduino Mega Serial Library*. 2020. URL: https://www.arduino.cc/reference/en/language/functions/communication/serial/.

[65] URL: https://www.comsol.com/blogs/singularities-in-finite-element-models-dealing-with-red-spots/.

[66] URL: https://www.throwtheswitch.org/ceedling.

[67] URL: https://www.mathworks.com/products/simulink-test.html.

[68] URL: https://simtk-confluence.stanford.edu/display/OpenSim/Scripting+with+Matlab.

Appendix A

Fall Gantt Chart

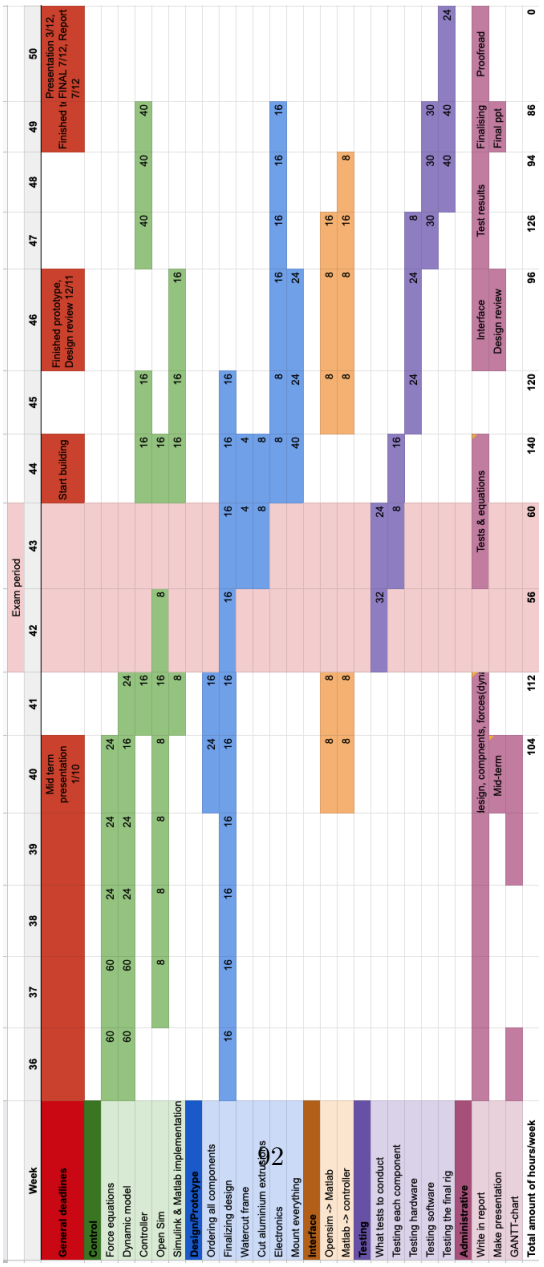


Figure A.1: Gantt chart made for the fall term

Appendix B

Technical Requirements

B.1 Test Rig

- The test rig should be easily moved by hand, max 2 persons required.
- The operation of the test rig should have a clear and easy-to-use User-Interface.
- The test rig should perform motions autonomously, according to a predefined program.
- The test rig should reach the maximum force of 500N per leg/actuator.

B.2 Micro-controller for Controlling Actuators

- Power efficiency: Low priority, electrical outlet connection.
- Temperature tolerance: -10 C/50 C
- Security: Low prio
- Processing power: Multicore would be nice
- Memory: $\geq 64KB$
- Digital Pins: ≥ 36
- Analog Pins: ≥ 6
- Output Signal Type to actuators: PWM
- Response time: $\leq$20ms from instructions to performed actuation (target position).

- Able to handle MATLAB/Simuling-generated hardware code
- Cost: $\leq$ 500SEK

Appendix C

Code

```
def serial_write_matrix(matrix, port='/dev/cu.usbmodem14101'):
    """ Sends whole matrix to Arduino."""
    # Initialize variables
    arduino = serial.Serial(port, baudrate=115200, timeout=1)
    arduino.flushInput()
    final_sensor_matrix = list()
    REF_STRING = '{},{},{},{},{},{}'
    try:
        for row in matrix:
            REF_STRING = '{},{},{},{},{},{},{},{}'.format(
                    row[0],row[1],row[2],row[3],row[4],row[5]
                    )
            print("SEND: {}".format(REF_STRING))
            arduino.write(bytearray(REF_STRING, 'utf-8'))
            time.sleep(0.1)
    except KeyboardInterrupt:
        try:
            arduino.close()
        except:
            pass
    except:
        import traceback
        traceback.print_exc()
        try:
            arduino.close()
        except:
            pass
```

Listing C.1: Python function for sending a matrix of references to the Arduino

```
int refMatrix[100][6];
bool matrix_downloaded = false;
void rxMatrixViaSerial() {
  while (matrix_downloaded == false) {
    if (Serial.available()) {
      Serial.readString().toCharArray(refs, 40);
      parseData();
```

```
      append_to_matrix(rxRow);
      change_val_increase(rxRow);
      is_matrix_downloaded(rxRow);
    }
  }
}
void parseData() {
  char * strtokIndx;
  strtokIndx = strtok(refs,","); // Get index of first comma in string refs
  strcpy(ref1_char, strtokIndx); // Copy string-part to char-array ref1_char
  ref1 = atoi(ref1_char); // Convert char to integer datatype
  // [...] Repeat for all values in string (use strtok(NULL, ",") instead)
  strtokIndx = strtok(NULL,",");
  strcpy(ref6_char, strtokIndx);
  ref6 = atoi(ref6_char);
}
void append_to_matrix(int &rxRow) {
    refMatrix[rxRow][0] = ref1; // Append ref1 to matrix
    // [...]
    refMatrix[rxRow][5] = ref6;
}
void change_val_increase(int &rxRow) {
  rxRow++;
}
void is_matrix_downloaded(int &rxRow) {
  if (rxRow == nRow) {
    matrix_downloaded = true;
  }
}
```

Listing C.2: Receive parse and store reference-data in a matrix on the Arduino. Repeated sections of code shortened for brevity.

```
  Serial.print(force_sensor[0]);
  Serial.print(",");
  // [...]
  Serial.print(force_sensor[5]);
  Serial.print(",");

  Serial.print(motor_position_sensor[0]);
  Serial.print(",");
  // [...]
  Serial.print(motor_position_sensor[5]);
  Serial.print(",");

  Serial.print(Ref[row][0]);
  Serial.print(",");
  // [...]
  Serial.println(Ref[row][5]);
```

Listing C.3: Code for transmitting sensor-data via serial communication

```
def serial_read_sensors_matlab(rec_time=10,
```

```
        path="../ArduinoSensorData/SensorOutput.mat",
        port='/dev/cu.usbmodem14101'):
    """ Reads and records sensor data from arduino and stores it in a mat-file."""
    final_sensor_matrix = list()
    arduino = serial.Serial(port, baudrate=115200, timeout=1)
    arduino.flushInput()

    # Recieve whole matrix
    t_end = time.time() + rec_time
    while time.time() < t_end:
        try:
            count = time.perf_counter()
            data = arduino.readline().decode()[:-2].split(',')
            sensor_values = [float(num) for num in data]
            sensor_values.insert(0, count)
            print('RCVD: {}'.format(sensor_values))
            final_sensor_matrix.append(sensor_values)
        except:
            pass
    data = dict(); time_list = list(); force = list(); position = list()
    ref = list()

    # Loop through each row in matrix
    for row in final_sensor_matrix:
        if len(row) == 19:
            # Time
            time_list.append(row[0])
            # Force
            values = [row[i] for i in range(1,7)]
            force.append(values)
            # Position
            values = [row[i] for i in range(7,13)]
            position.append(values)
            # Reference
            values = [row[i] for i in range(13,19)]
            ref.append(values)

    data["sensor_time"] = time_list
    data["sensor_force"] = force
    data["sensor_position"] = position
    data["ref"] = ref
    mat4py.savemat(path, data)
    arduino.close()
    return
```

Listing C.4: Python script for reading parsing and storing sensor-data in mat file-format

The `mat4py` python package needs to be installed with `pip3` package manager.

Appendix D

Wiring Diagram

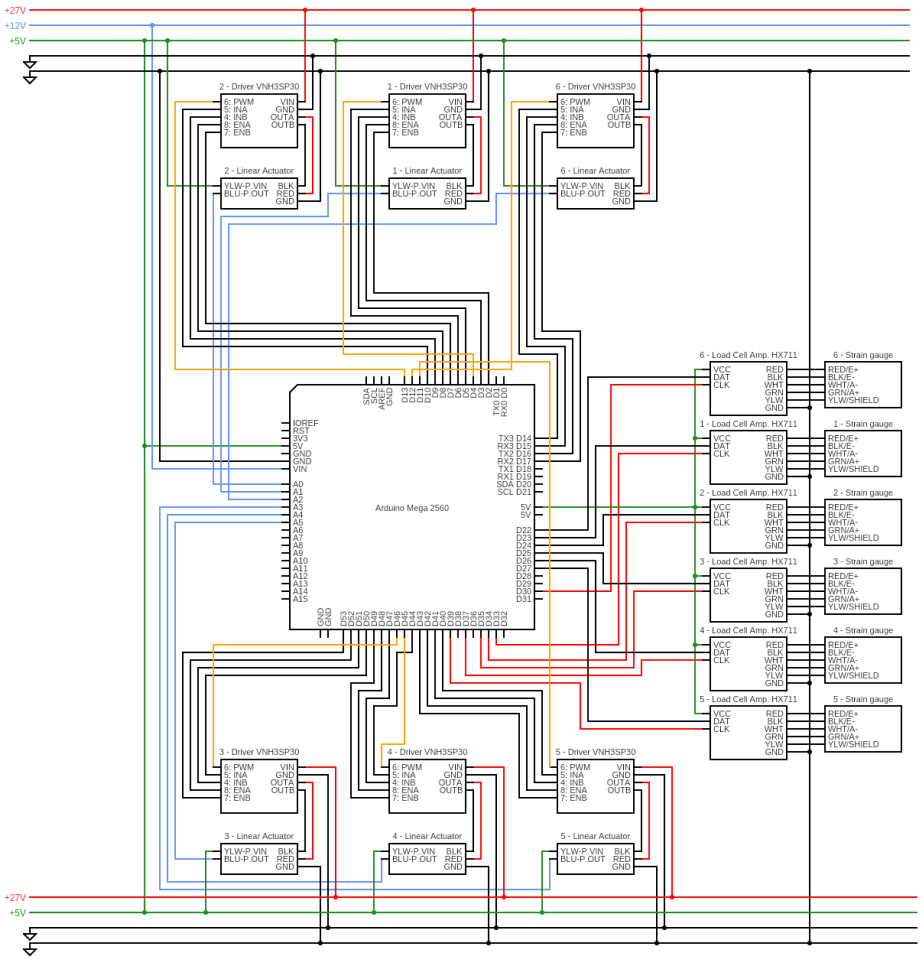
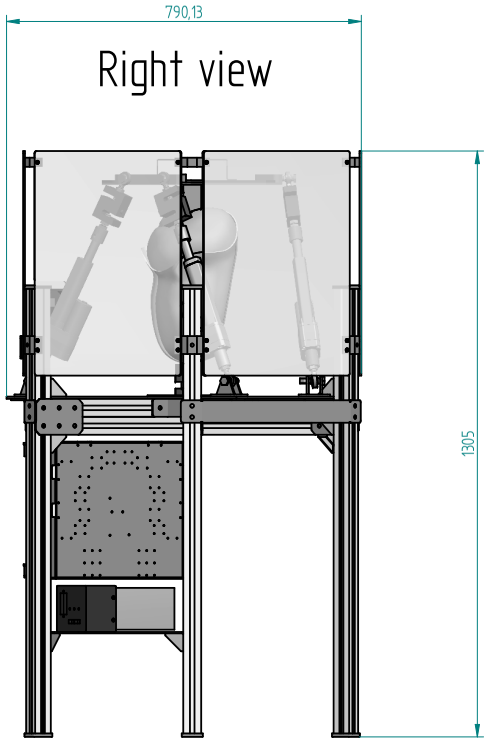
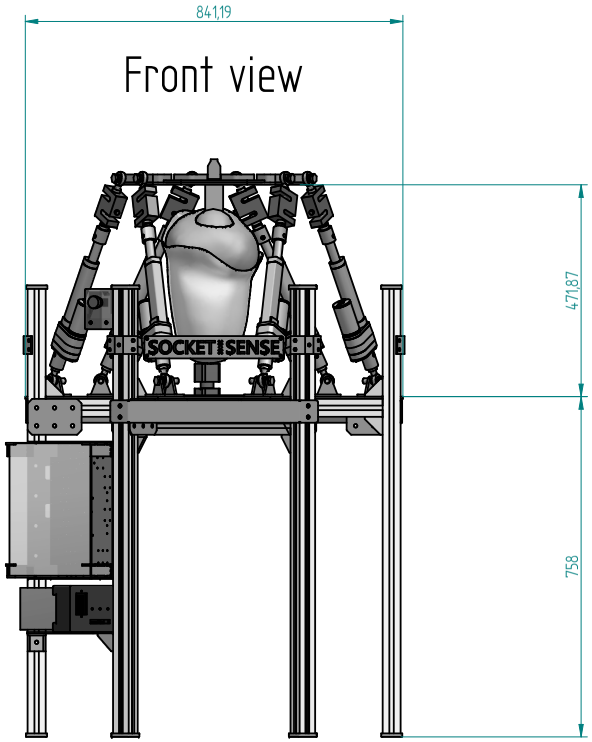


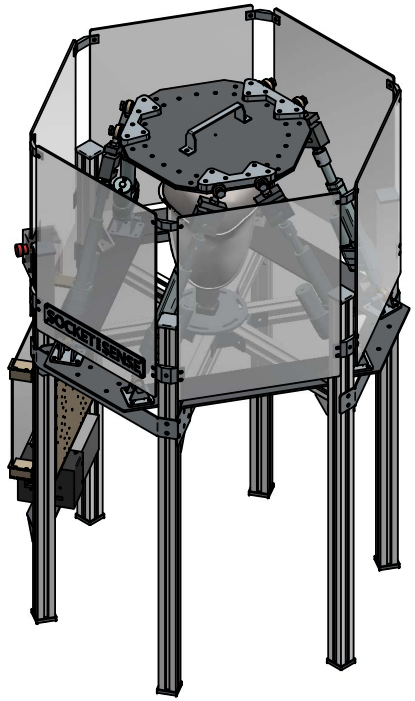
Figure D.1: Wiring Diagram.

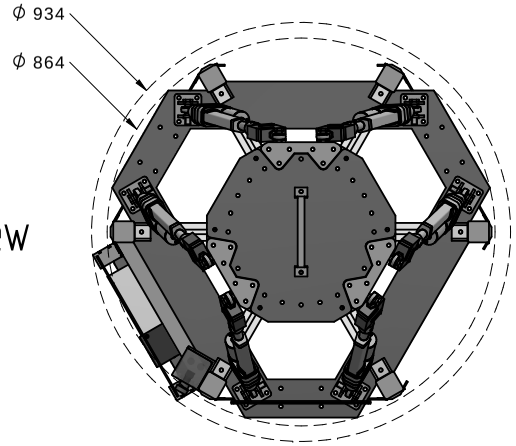
Appendix E

CAD Draft









Detaljnr	Antal	Benämning	Material	Ämne / Dimension	Anmärkning

Skapad av	Godkänd av	Material	Skala	Ersätter	Ersatt av
SS_HK19/20			1:8		

MF2059	SocketSense_Testrig	Datum: 2020-12-13 Ritningsnr