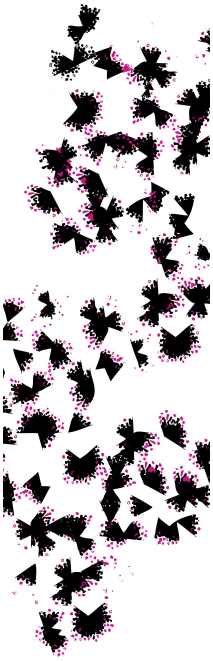


The implementation of collision avoiding haptic feedback in the pedal-based control of a robotic platform

S.D. (Sierd) Meijer

BSc Report

Committee:
Dr. Ir. D. Dresscher
Dr.ir. E.C. Dertien

August 2018

028RAM2018
Robotics and Mechatronics
EE-Math-CS
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

UNIVERSITY OF TWENTE.

MIRA CTIT
BIOMEDICAL TECHNOLOGY AND TECHNICAL MEDICINE

Summary

i-Botics aims to improve the knowledge and technology concerning telerobotics and exoskeletons. The overarching project of this thesis is about telerobotics, which is a combination of telepresence and teleoperation. This thesis focusses on the improvement of the teleoperation. Teleoperation is about the control of a robot over long distances.

This thesis is about the implementation of haptic feedback in the pedal based control control of a robot. The haptic feedback is supposed to indicate the location of obstacles in the path of motion, reducing the required mental effort to operate the robot. The main goal is divided in three sub goals.
1. A research on different haptic feedback implementations;
2. The design and implementation of a haptic feedback capable interface for the pedals;
3. The implementation of a connection between the pedals' interface and the robot.

In the analysis an initial plan for the design and implementation phase is set up. First, three kinds of haptic feedback are researched and compared based on their implementation in the system in question. Second, an overview of the existing hardware is made. Third, a controller with a variable spring constant for the pedals is designed. And last, the software architecture and functionality for the whole system is designed.

In the design and implementation chapter, the final implementation is described. The chapter is divided in three sections: electrical, digital and mechanical. Any alterations or additions with respect to the analysis are mention here.

After the final implementation, the system is tested and evaluated. The testing is divided into two parts. The first part tests the pedals' sensors and the controller, and the second part tests the haptic feedback implementation. All implemented parts are tested individually.

The testing showed that the controller is usable, based on the input from the sensors. The controller is also capable of maintaining a relatively stable state while altering the spring constant. The controller does show unexpected behaviour caused by the delay in the controller design. The pedals can operate a virtual robot based on their relative position. The haptic feedback implementation is also tested in a virtual environment and works as expected.

The tests have shown that all the technical requirements of the assignment have been met. Whether the implementation of haptic feedback decreases the required workload cannot be said because no user tests have been conducted.

Samenvatting

i-Botics richt zich op het vergroten van de kennis en verbeteren van de technologie op het gebied van telerobotica and exoskeletten. Het overkoepelende project van deze thesis houdt zich bezig met telerobotica, wat bestaat uit tele-aanwezigheid en tele-operatie. Deze thesis focust op het verbeteren van tele-operatie: het besturen van een robot over lange afstanden.

Deze thesis beschrijft de implementatie van haptische feedback in een pedaal-gestuurde robot. De haptische feedback moet de bestuurder informeren over de locatie en afstand van obstakels op het pad van de robot. De implementatie zou de vereiste mentale inspanning voor het besturen van de robot af moeten nemen.
Het hoofddoel bestaat uit drie subdoelen:
1. Het vinden van een geschikte implementatie van haptische feedback;
2. Het ontwerpen en implementeren van een, haptische feedback capabele, interface voor de pedalen;
3. Het implementeren van een verbinding tussen de interface van de pedalen en de robot.

In de analyse is een initieel plan voor de ontwerp- en implementatie-fase opgesteld. Eest zijn drie verschillende soorten haptische feedback onderzocht en vergeleken op basis van hun implementatie in de pedalen. Vervolgens is alle bestaande hardware in kaart gebracht. Als derde is een controller met een variabele veerconstante ontworpen. En als laatste is de software architectuur en functionaliteit voor het gehele systeem ontworpen.

In het hoofstuk Ontwerp en Implementatie is de uiteindelijke implementatie besproken. Dit hoofdstuk is verdeeld in drie delen: elektrisch, digitaal en mechanisch. Ook de afwijkingen en aanvullingen ten opzichte van de analyse worden hier besproken.

Na de voltooide implementatie is het systeem getest en geëvalueerd. De tests zijn opgedeeld in twee delen. Het eerste deel test de sensoren en de controller van de pedalen. Het tweede deel test de implementatie van de haptische feedback. Alle geïmplementeerde onderdelen zijn individueel getest.

Uit de testen bleek dat, op basis van de sensor waarden, de controller functioneel was. Ook bleef de controller relatief stabiel wanneer de veerconstante veranderde van waarde. De controller vertoonde soms onverwachts gedrag vanwege de vertraging in het ontwerp. De pedalen kunnen een virtuele robot aansturen op basis van de relative positie. De haptische feedback is ook getest in een virtuele omgeving en gedroeg zich als verwacht.

De testen laten zien dat alle technische vereisten van de opdracht zijn vervuld. Of de implementatie van haptische feedback de werklast verlaagd kan niet worden gezegd omdat er geen gebruikerstesten zijn uitgevoerd.

Contents

1 Introduction

1.1 Context

This assignment is commissioned by i-Botics, a collaboration between TNO and University of Twente, and is part of an overarching project aimed at the innovation of telerobotics. Telerobotics is concerned with the remote control of semi-autonomous robots, in which telepresence and teleoperation is combined. The robot that is used is shown in figure 1.1a and consists of two RMP omni 50 systems that form the base. Attached to the robot is a KUKA LWR 4+ arm with at the end a ReFlex TakkTile gripper. Figure 1.1b shows the Leo Universal Cockpit that is used to operate the robot and is capable of providing the operator video, audio and haptic feedback. The goal of this assignment consists of two parts. The first part is creating an interface between the pedals located in the cockpit and the control of the robot, and the second part is providing the operator with haptic feedback via the pedals.

1.2 Project goal

The goal of this project is to reduce the mental effort required for the control of the robot because during the operation the operator will be mostly focussed on the control of the arm and less so on the control of the robot. Therefore the use of the pedals should be intuitive. This intuitiveness will also be attempted by providing the operator information on the location and distance to obstacles detected by the robot via haptic feedback in the pedals. The haptic feedback should be of a guiding nature to decrease the operators required attention, while not restricting any movability of the platform.

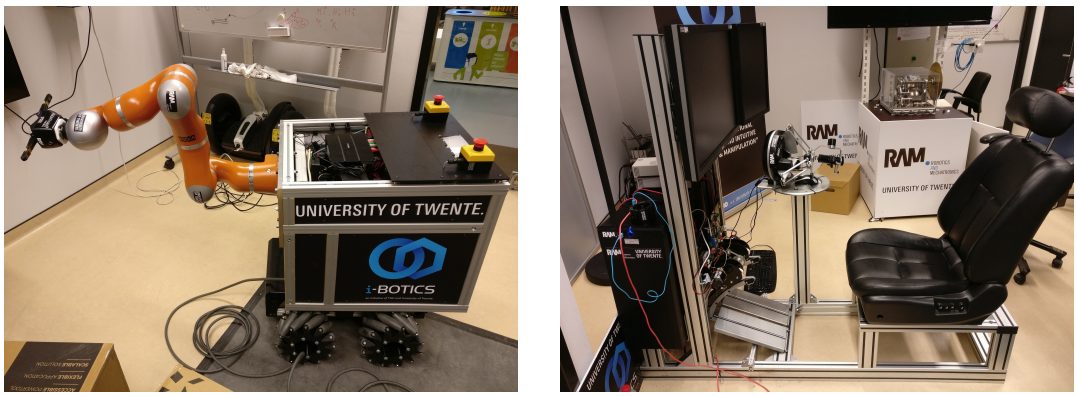
1.3 Method

First, the pedals are equipped with an encoder, accelerometer and motor and need to be connected to an embedded computer running a control system for the operation of the pedals. A connection needs to be established between the embedded computer and ROS (Robotic Operating System), a middleware which is used for the control of the robot. After the pedals are connected to ROS, an algorithm will be implemented for controlling the motion of the robot which is based on the sensors located in the pedal. When full control of the robot is established another algorithm has to be designed for the implementation of haptic feedback. The haptic feedback will be based on the distance and relative location of an obstacle detected by the robot, and the velocity of the robot.

1.4 Report outline

To reach the goal the project, four aspects are researched in (chapter 2) the analysis. First, different implementations of haptic feedback are compared and evaluated for their applicability. Second, an overview of the existing hardware and their characteristics is made. Third, an interface for the pedals and the control system are researched. And last, a software architecture is designed for the control of the platform and the implementation of

haptic feedback. After the analysis, (chapter 3) the design & implementation is described, which is divided in a hardware and a software part. When the system is fully operational it will be tested based on the project goals set out in the introduction in chapter 4, testing & results. Then, a conclusion (chapter 5) is written based on the findings during the assignment and after, recommendations (chapter 6) will be made for future work.



(a) Robotic platform (b) Leo Universal Cockpit

Figure 1.1: Setup robotics platform

2 Analysis

In the analysis four different aspects of the research question are elaborated upon. First, different kinds of haptic feedback implementations are discussed and how they translate into bilateral pedal control. Second, an overview of the existing hardware is made, whether they will be used and if so, the characteristics they have. Third, the interfacing of the pedals is discussed for both the hardware and the software. And last, the use of ROS is discussed in combination with the results of the hardware and software chosen in the previous part. A simplified overview of the end system is given in figure 2.1.

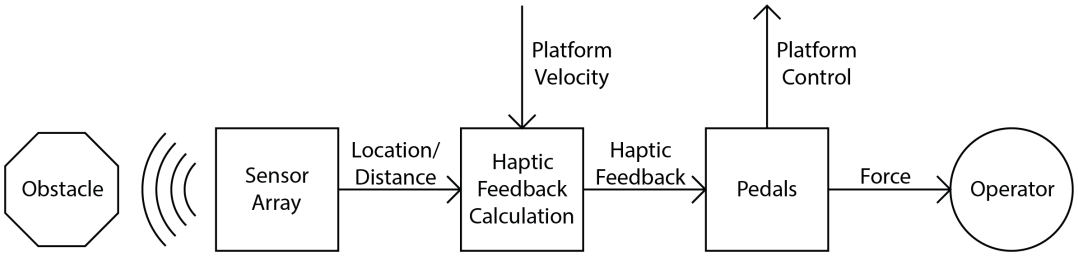


Figure 2.1: Simplified system overview

2.1 Comparing haptic feedback implementations

There are multiple variations on the implementation of haptic feedback based on obstacle detection. In this section three different implementations are compared and evaluated whether they could be effective in pedal based control. All three implementations have a trade-off between an increase in physical effort and a decrease in mental effort for the operation of the platform. The implementations are force offset, stiffness feedback and force-stiffness feedback.

2.1.1 Force offset

Force offset [1] gives feedback by increasing the position error of the controller based on the distance to an obstacle. This can be compared to a spring of which a supposedly static connection changes in position. Figure 2.2 (1) shows a spring attached to a wall with rest length x_{rest} and spring constant K_s. Based on Hooke's law shown in formula 2.1 the exerted force by the spring is 0 because the displacement of the spring is equal to the rest length.

$$F_{spring} = K_s \cdot (x_{rest} - x) \tag{2.1}$$

Figure 2.2 (2) shows the same spring with a force F_{p1} applied to point A which increases the displacement δx. When force offset is applied, it alters the location of, in this case, the wall based on the obstacle distance in order to increase the displacement of the spring, resulting in a force exerted by the spring. Figure 2.2 (3) shows the increased force F_{p2} that

is required to keep point A in place when an offset is implemented. This force is exerted until it reaches its new rest position.

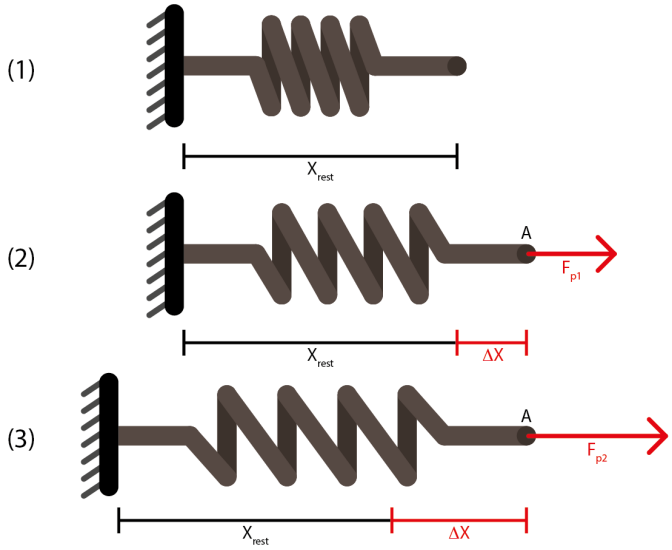


Figure 2.2: Force offset example (1) Spring at rest length. (2) Spring extended by force F_{p1}. (3) Spring influenced by force offset, increasing Δx and force F_{p2} to keep point A in place

Implementing force offset can inform the operator of the obstacle distance and location without input from the operator, but increases the required physical effort while operating. [2] and [1], [3] show that a force offset improves the collision avoidance when implemented with a correct feedback loop gain by shifting the origin positition of the control device so that a collision is avoided if the operator does not give input. A characteristic of force offset feedback is that the offset is also present if no force is exerted by the operator (passive feedback) [1], which means that the way op operating is changed. This can be useful if the controlled platform can move independently, for example an UAV drifting because of wind because the platform will correct itself without input. The distance to an obstacle is inversely proportional to the shift of the origin position giving a good representation of distance when there is no input forc. However, after the offset is compensated the spring constant does not change, and the steering looses its indication of distance. Also important is that the offset needs to be compensated, whenever the operator wants to go straight while in the vicinity of an obstacle a force is required, which increases the required mental effort of the operator instead of decreasing it as shown by [3], [1], [4]. The goal of the implementation of haptic feedback is to assist the operator with maneuvering the platform which should decrease the required mental effort.

Implementing a force offset in the pedals requires the operator to be aware of the pedal positions before applying force to operate the platform, because the offset could have altered the origin position. This increases the required mental effort. First of all, the operation of the pedals may not allow for the implementation of a force offset. Most pedal

based controls have the origin position of the pedal in its most upright position (e.g. car pedals), in this case a force offset would be impossible, however the control of the pedals in relation to the platform is not definitive yet. Second, the force offset is also present when there is no input force, this was proven to be useful in case of a drifting UAV. However, when implemented in a platform that is unlikely to move because of the friction with the ground, it can lead to the platform moving itself without the input of an operator. This exceeds the role of guiding the operator in controlling the platform. Because the force offset requires the operator to actively counter the force offset and alters the controls, it is unsuited to be implemented in the pedal based control.

2.1.2 Stiffness feedback

Stiffness feedback [1] alters the spring constant of the controller based on the distance to an obstacle. Controllers that return to their origin position when there is no input force have a physical or virtual spring to do so. Force applied to the virtual spring needs to be larger than the retracting force that the spring exerts in order to extend the spring. Stiffness feedback changes the required force to extend the spring by altering the spring constant as shown by [1], this is K_s in formula 2.1. Figure 2.3 (1) again shows a spring at rest length. Figure 2.3 (2) shows a spring with a displacement Δx because of a pulling force F_{p1}. In figure 2.3 (3) K_s is increased which results in a larger F_{p2} in order to keep point A in place.

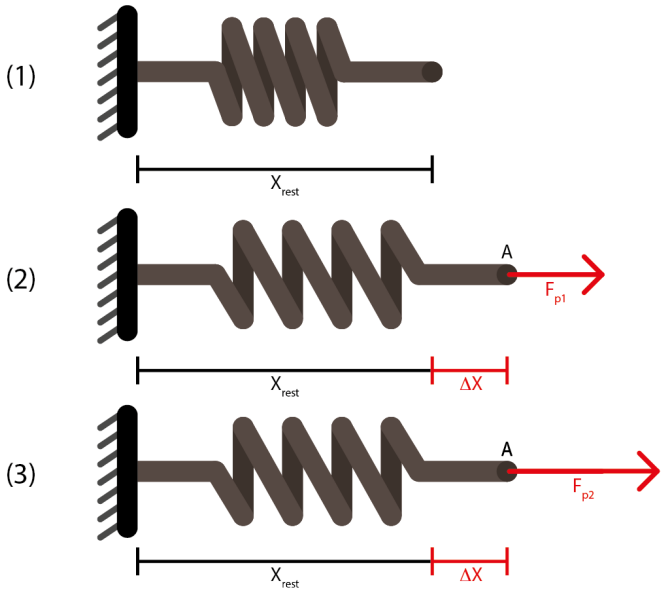


Figure 2.3: Stiffness feedback example (1) Spring at rest length. (2) Spring extended by force F_{p1}. (3) Spring influenced by increased spring constant K_s, increasing F_{p2} to keep point A in place.

In contrast to the force offset, the stiffness feedback does not indicate the distance and

location of an obstacle in rest position, but provides this info while force is exerted on the pedal. The closer an obstacle gets, the more force the pedal will exert on the operator to get to its origin position. This increases the required physical effort, but less than the force offset because there is no constant force required to keep the pedal in its place. One characteristic is that the pedal does not provide any information when there is no force applied to the pedals like the force offset does. A second characteristic is that, because the spring constant is proportional to the distance and the pedal position, there is a relatively small feedback force when the operator applies a small force. [1], [3] discuss that this relatively small force could be too small to perceive, which defeats its purpose. [1] do show that in case of a joystick controlled UAV, stiffness feedback results in better performance and reduced required mental effort compared to an offset force. Based on the requirements of the system discussed in the introduction, it is beneficial that the pedal does not provide a feedback force when the operator does not apply force because it does not alter the control of the system. The proportional relation between obstacle distance and increased spring constant can be changed into, for example, an exponential function that provides larger force variations when the obstacle is at minimum range.

Applying the stiffness feedback on the pedal based control is a viable option because of its guiding nature. That stiffness feedback does not provide any feedback when there is no force applied, is an advantage because it decreases the required mental effort of the operator. The stiffness feedback does not alter the control, only increases the required physical effort, and thus does not exceed its guiding purpose, making it an option for testing.

2.1.3 Force-stiffness feedback

Force offset and stiffness feedback both have their advantages and disadvantages and can be combined into force-stiffness feedback [3]. Force-stiffness feedback combines the advantages of both force offset and stiffness feedback by increasing the force to return to a shifted origin position, based on the distance to an object [2]. [3] explains it as adding force feedback to compensate the small feedback force that is generated by stiffness feedback when there is a small displacement in the pedal's position. This is done by adding force directly with an offset force and indirectly by creating a larger distance to the origin, increasing the stiffness feedback force. Equation 2.2 is an adaptation of Hooke's law including force offset and stiffness feedback.

$$F_{spring} = K_s \cdot ((x_{rest} + x_{offset}) - x) \tag{2.2}$$

Where F_{spring} is the output force, K_s the varying spring constant, x_{rest} the origin position, x_{offset} the shift in origin position and x the current position.
Figure 2.5 (1) shows a spring attached to a wall with rest length x_{rest} and spring constant K_s in rest position, in figure 2.5 (2) a force F_{p1} is applied and the spring extends with displacement Δx. Figure 2.5 (3) is a repetition of the result of stiffness feedback. Figure 2.5 (4) shows the combination of force offset and stiffness feedback which results in an even larger required force F_{p3} to keep point A in place. Based on the location of an obstacle, the origin position of the controller is shifted, increasing the position error. Then,

the increases position error is multiplied by an increased spring constant, resulting in an exponential increase in force output. This is visualised in figure 2.4.

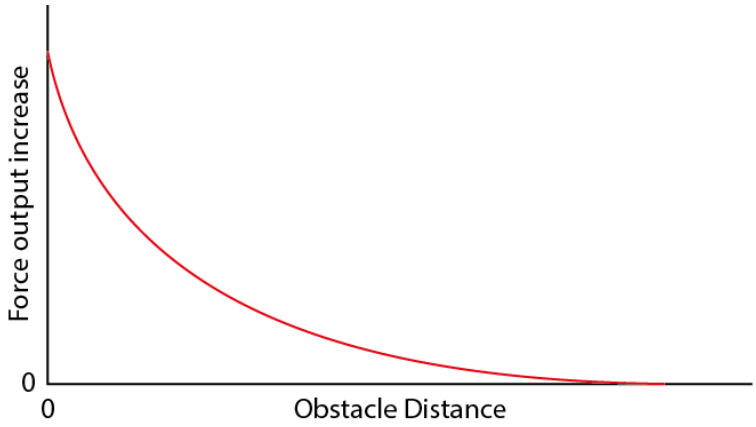


Figure 2.4: Relation between the increase in output force and the distance to an obstacle, using force-stiffness feedback

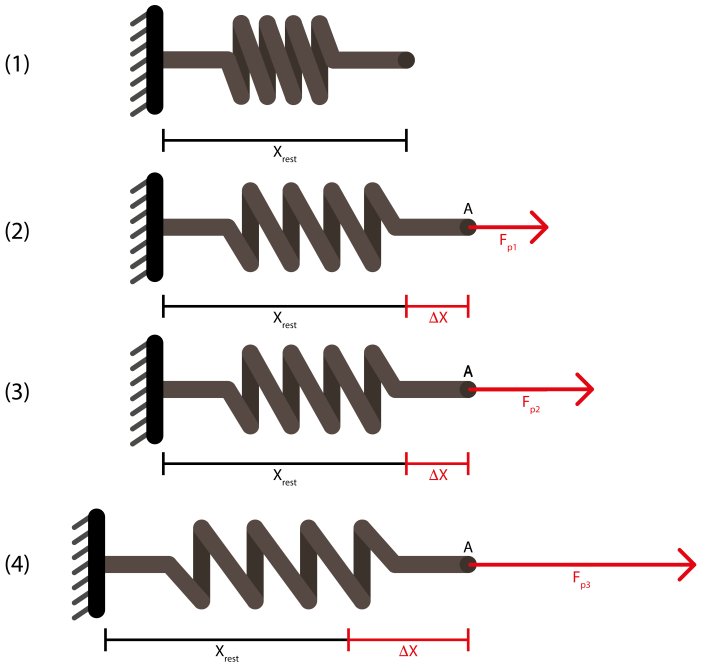


Figure 2.5: Force-stiffness feedback example (1) Spring at rest length. (2) Spring extended by force F_{p1}. (3) Spring influenced by increased spring constant K_s, increasing F_{p2} to keep point A in place. (4) Combination of force offset and stiffness feedback, increasing F_p3 because of an increased Δx and K_s.

[3] conducted an experiment where subjects where asked to fly an UAV from way point to way point as fast as possible, the results shows that only force offset feedback significantly decreased the amount of collisions, however, adding stiffness feedback decreased the amount of collisions even more. Nevertheless, the same disadvantage of creating an offset with the force feedback is still present, but the amount of offset is decreased because of the added stiffness feedback [3]. Depending on which properties of different implementations prove to be more important, a trade off will have to be made between stiffness and force-stiffness feedback.

Force-stiffness feedback is a potent option and can be made even more interesting when altered. [3] discusses changing the origin, and basing the spring displacement on the new origin, however, this would still create the unwanted pedal offset discussed earlier. A variation on the implementation could be to virtually displace the origin of the pedal and only apply the stiffness based on the displacement when force is applied to the pedal. Because both the distance to the origin and the spring constant are increased, the output force based on the distance will be an exponential function. Because the combination of force offset and stiffness feedback is practically an exponential spring, a non-linear spring can be used. This will give the same result without the drawback of the offset and is therefore a viable option.

2.1.4 Conclusion

All three implementations have different characteristics which are compared with respect to the goal of the system as stated in the introduction. The following subjects are compared: distance indication, increase in effort, guidance and reactive feedback. Distance indication means how well the operator is able to interpret the distance to an obstacle based on the feedback of the pedals. An increase in mental effort is created when more force is required to operate the system in comparison to a system without feedback. The system is supposed to guide the operator based on his input and the surroundings of the robot platform and not apply any restrictions or alter the way of operation, which is reflected in guidance. Reactive feedback means how well the system reacts to the input of the user. Table 1 shows the comparison between the four implementations and four subjects.

	Distance indication	**Increase in physical effort**	**Guidance**	**Reactive feedback**
Force offset	-	- -	- -	- -
Stiffness feedback	+	-	++	++
Force-stiffness feedback	++	- -	+	+
Non-linear stiffness feedback	++	-	++	++

Table 1: Comparison table for the three different feedback implementations

Force offset is not a suited implementation based on the requirements of the haptic pedals. The distance indication is clear when no force is applied by the operator, however, it is likely to get lost when force is applied. Preventing the system from moving on its own because of the offset is physically intensive and also alters the control of the system which exceeds the goal. The feedback is also not based on the input of the operator because the increase in force is based on the displacement created by the offset.

Stiffness feedback is a viable option to test based on the requirements. The distance indication is good when a large force is applied by the operator, but not when this force is small. Because the displacement is multiplied by the spring constant which increases, the required input force can increase rapidly leading to an increase in physical effort which is not ideal. The increase in required force is only present when applying force and thus guides the operator in avoiding obstacles.

Force-stiffness feedback is very similar to stiffness feedback in terms of characteristics, however the drawbacks of the force offset remain. Because of the offset, force-stiffness is not a viable option. The non-linear stiffness feedback on the other hand takes the advantages of force-stiffness feedback and removes the disadvantage of the offset, making it an equally good or better option than the other three in every aspect. Based on this finding non-linear stiffness feedback will be used during the implementation phase.

2.2 Hardware

The pedals and interface used for driving the robot platform and receiving force feedback are made by Martijn de Roo [5]. For now only one pedal is available. It consists of a pedal frame and profile, drive train, motor, accelerometer and rotary optical incremental encoder. Figure 2.6 gives an schematic overview of how the hardware is connected.

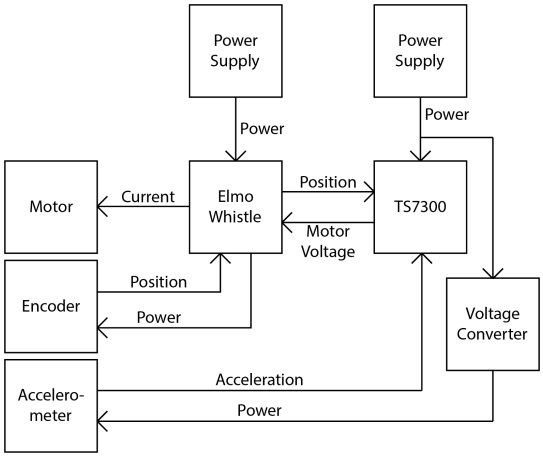


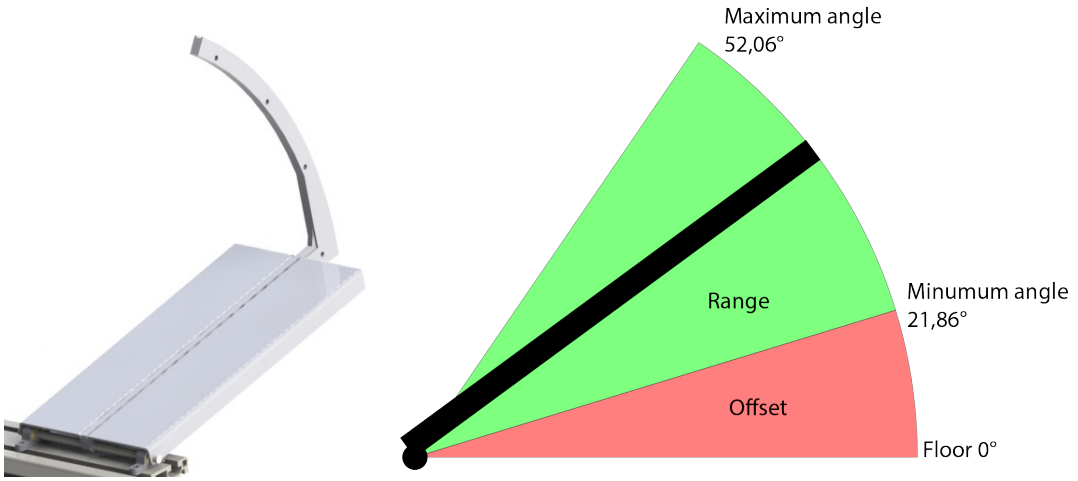
Figure 2.6: Schematic overview on the old hardware layout

2.2.1 Pedal arm & profile

Both the pedal arm and profile are made out of aluminium and consist of five parts, two symmetric side panels and a centre part flanked by plastic, the side parts are folded from a flat sheet of aluminium. The side parts are made out of 3mm thick aluminium and the centre part out of 5mm. Between the sides and centre parts two strips of plastic are added to widen the middle section to accommodate the timing belt. A render of the pedal is shown in figure 2.7a. The dimensions of one foot plate are 334mm x 150mm x 16mm and support up to a size 47 foot. The base of the pedal is mounted to the LUC via a BOIKON profile [6]. It has a rotational movement range of 30,2 degrees with an offset of 21,86 degrees, which is shown in figure 2.7b.

2.2.2 Motor & drive train

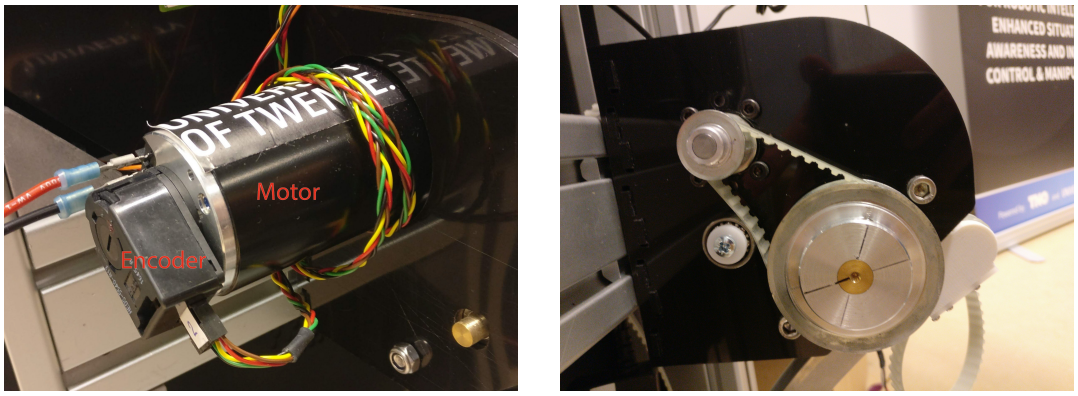
The motor attached to the pedals is a Maxon RE 50 [7] and is shown in figure 2.8a. The motor is able to transfer force to the pedal with a timing belt as shown in figure 2.8b. The transfer ratio between the motor and the pedal is 3:1, which makes the motor capable of exerting 127N on the pedal [5]. The pedal is connected to the pulley axle with another timing belt that is connected underneath the pedal and at the top of the arm.



(a) Render of the pedal [5]

(b) Movement range of the pedal

Figure 2.7: Pedal for operating the robot



(a) The motor and encoder

(b) The pulley system

Figure 2.8: Pedal hardware

2.2.3 Encoder

The pedal uses an encoder to keep track of the position of the motor which provides the position of the pedal. It is mounted on the shaft of the motor as shown in figure 2.8a. The used encoder is HEDS 5540 from Avago Technologies [8]. It is a three channel optical incremental encoder delivering a two channel square wave in quadrature and the third is an index channel that delivers a pulse for every full rotation, enabling the tracking of absolute position. Because the pedals can be manually moved, the encoder count will be reset upon restart while the pedals are in a predefined position to ensure a correct calibration. The pedal can move a maximum of 30,2 degrees, which results in a resolution of 1,85E-3 degrees per step of the encoder.

2.2.4 Accelerometer

The accelerometer that is used is a triple axis accelerometer Breakout MMA7260Q [9]. It is placed underneath the pedal as close to the axle as possible, as shown in figure 2.9, to minimise the amount of vibrations. Because of the way the accelerometer is mounted to the pedal, the accelerometer will only give output over one axis because it is perpendicular to the pedal and moving in a circular motion. The output will not only be used for the acceleration, but will also be integrated to velocity, and if proven to be necessary, integrated again into position.

The Breakout has four sensitivity modes: 1.5g, 2g, 4g and 6g. By pressing the pedal with altering forces and with the accelerometer connected to an Arduino, the maximum acceleration was measured in order to choose an appropriate sensitivity mode. The output of the Breakout in 1,5g mode is shown in figure 2.10 and 2g in figure 2.11 using an Arduino & Processing sketch for making graphs [10]. The spikes in both figures can be ignored because they were formed by the Arduino hitting the ground while moving the pedal. 1,5g mode gave results with enough margin to keep accurately measuring the acceleration. This mode outputs 800mV/g with variance between 740 and 860 mV/g [9], however, exact calibration is required during the implementation. The accelerometer also measures gravity, when an axis is perpendicular to the ground it will give a 1g output in the respective axis, this needs to be compensated for an accurate acceleration measurement.

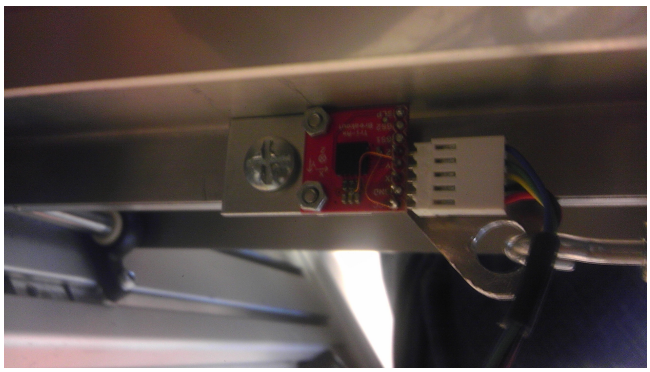


Figure 2.9: Accelerometer placement under the pedal

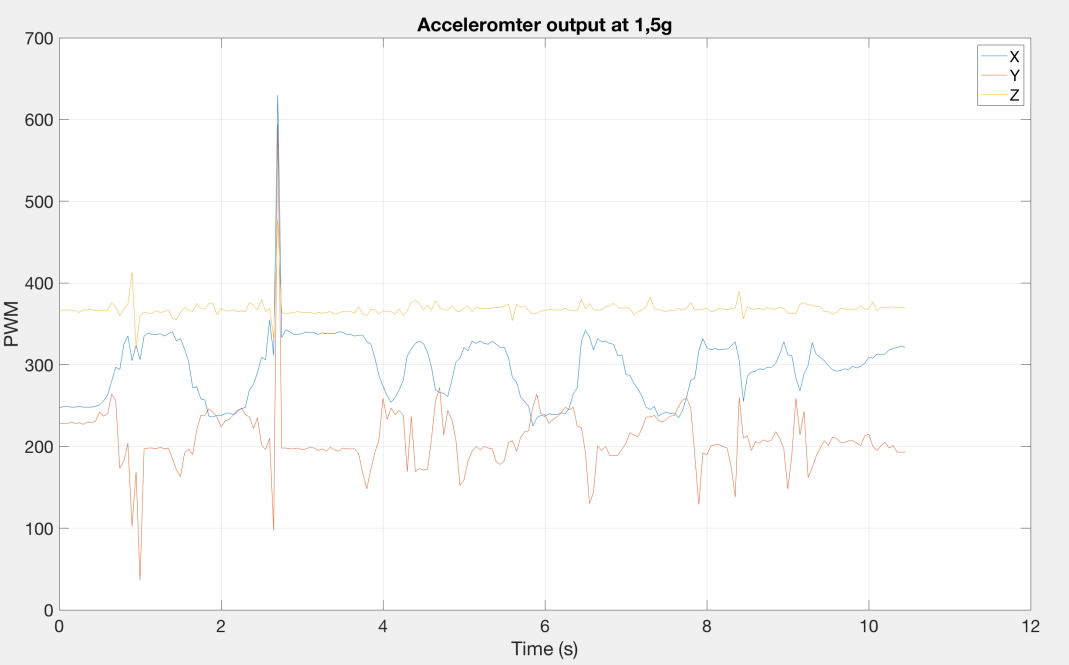


Figure 2.10: Accelerometer output at 1,5g sensitivity

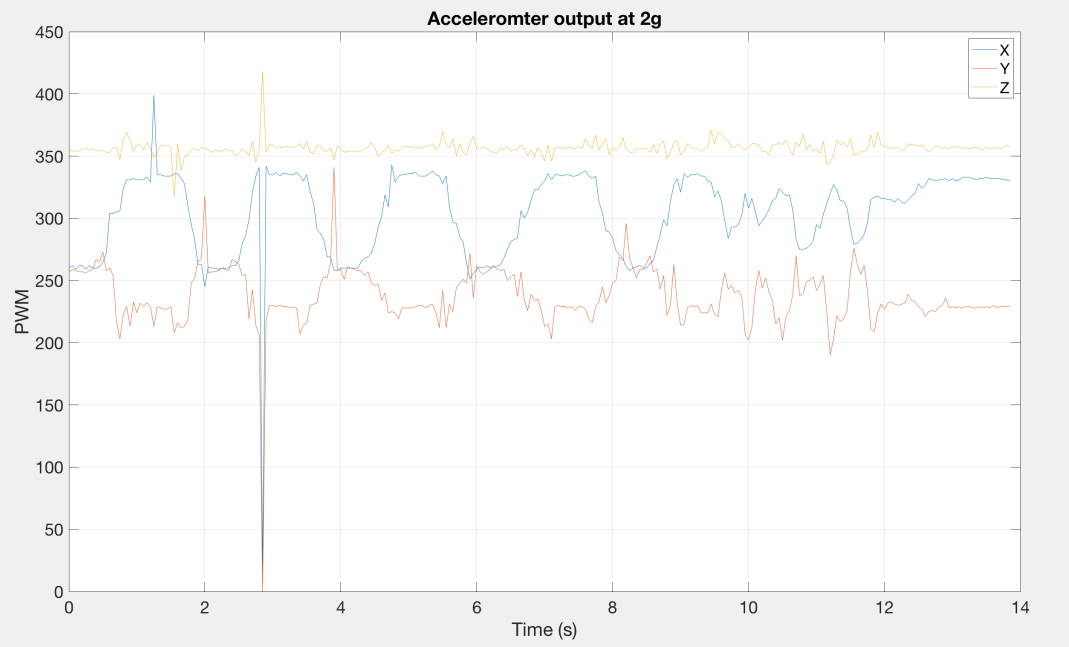


Figure 2.11: Accelerometer output at 2,0g sensitivity

2.2.5 Elmo Whistle & power supply

The Elmo Whistle is a motor controller which regulates the current going to the motors. On top of the Elmo Whistle is an Elmo Whistle board as designed by G. te Riet o/g Scholten which provides the required I/O for operation. The Elmo will be used in voltage mode where it takes a voltage between two set-points and translates it into a current to the motor. The position of the pedal is controlled by controlling the voltage. Also connected to the Elmo is a power supply which is used to drive the motors. The power supply in question is the EA-PS 548-05T [11], capable of delivering between 43V and 58V and 5,2A with a 78% efficiency.

2.2.6 TS7300, YS9700 & TS-XDIO

The existing interface is made with a TS7300 micro controller [12] running 20-sim 4C software [13]. This TS7300 is used as proof of concept, but the board is no longer supported by RaM and therefore needs to be replaced. However, the wiring schematic will be copied because the pedal hardware will stay the same. To connect the accelerometer to the TS7300 and the Elmo whistle (motor controller) [14] to the motor, a TS9700 (AD/DA converter) [15] is placed on top. The encoder is connected to the TS7300 via a TS-XDIO (extended digital IO shield). How the TS7300 and its add-ons will be replaced will be discussed in 2.2.7.

2.2.7 Embedded computer replacement

The pedals need to be connected to an embedded computer in order to be controlled and connected to a network. Two options are compared. The first option is an Arduino, because of prior experience no learning is required, enabling more time allocation for development. Second, a RaMstix is a possibiltiy because it is resourceful and it is developed by RaM and therefore well supported. There are three main requirements for the embedded system to operate correctly:
1. Containing all required I/O
2. Sufficient processing capabilities
3. A connection to ROS (see section 2.4)

Both embedded systems provide sufficient I/O to connect to all sensors and actuators and will not be compared on this characteristic.

2.2.7.1 Arduino

The Arduino is chosen as an option because of the time restriction for this research. The advantage of programming with Arduino is that it will not take time to learn because of previous experience, however the disadvantage is that Arduino is a relatively limited board in comparison to the RaMstix in terms of processing power and dedicated I/O. For the testing an Arduino Uno is used due to availability. It is connected directly to a laptop during testing.

Connection to ROS

ROS and Arduino can easily be connected with the rosserial_arduino [16] package. The package provides a ROS communication protocol that works over Arduino's universal asynchronous receiver-transmitter (UART) which makes the Arduino a ROS node which can directly publish and subscribe to ROS messages [17]. Also, the IDE from Arduino can be used for developing the node, which makes programming the Arduino relatively easy.

Processing capability

A significant limitation of the Arduino is the processor speed in combination with the lack of dedicated encoder inputs. The Arduino uses interrupt pins to add to or subtract from the step count, taking up processing time. A test has been done in order to see whether the Arduino could keep up with counting the steps of the encoder. The pedal was pulled all the way up before starting the counting on the Arduino. During the test the pedal was quickly pushed all the way down and slowly pulled back up again seven times. The results can be seen in figure 2.12. The figure clearly shows that the Arduino counted less steps when the pedal was going down (decreasing the step count) in comparison with going up, meaning it missed steps when the pedal is going too fast. Each position of the pedal should correspond to a unique step count, however missing steps alters this unique step count making the Arduino unusable for position tracking and thus for the implementation.

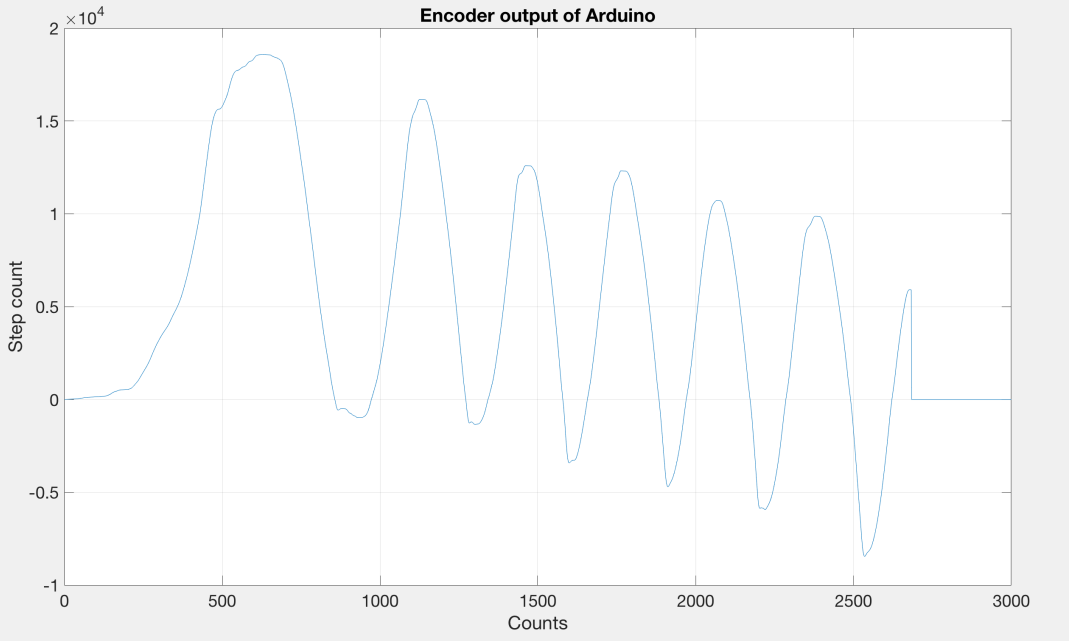


Figure 2.12: Encoder output when connected to an Arduino Uno

2.2.7.2 RaMstix

The RaMstix is more a powerful and versatile board than the Arduino, giving it the preference with respect to future development. The board contains an Overo module with an ARM processor, FPGA and many I/O options. The FPGA and Overo module are able to communicate via a General Purpose Memory Controller. The Overo module runs Linux, and because it is connected to the FPGA, C programs are able to communicate with the I/O components.

Connection to ROS

The RaMstix is able to run Linux which in turn can run ROS and thus multiple ROS nodes. Because Linux on the RaMstix is able to communicate to the I/O, ROS can do the same. Because the board is directly connected to ROS the pedals can be controlled without the need of the PC already present in the cockpit, this is an advantage because any nodes required to run on the master side will not have to share processor resources with nodes not relevant to the pedals.

Processing capability

The RaMstix has a significant advantage over the Arduino in processing capability. It contains a processor that is approximately 63 times faster and it has dedicated encoder inputs. This means that the encoder values are stored in a register and can be requested at any point in time. This is also true for the DAC and ADC values used for the accelerometers and Elmos. Therefore the full processor capability is available for the processing of the signals.

2.2.7.3 Conclusion

In the end the RaMstix has been chosen because of the significantly larger capabilities of the system. A comparison table is shown in table 2. Because the Arduino was not able to keep up with one incremental encoder it is not resourceful enough for the use of this project. The RaMstix will be used as embedded computer, limiting time for current development, but enabling future development.

Table 2: Embedded computer comparison

Device	**Easy of Use**	**Ros Connection**	**Processing Capabilities**
Arduino Uno	+	+	- -
RaMstix	-	+	+

2.3 Pedal controller

In order to control the position and damping of the pedals, a controller is designed using the accelerometer and encoder data as input. Both sensors' output first needs to be converted to SI-units and possibly integrated or differentiated depending on the controller that is used. The pedal controller will also include a variable spring constant for the implementation of haptic feedback, this is discussed in section 2.4.2. Note that the controller is in the rotational domain.

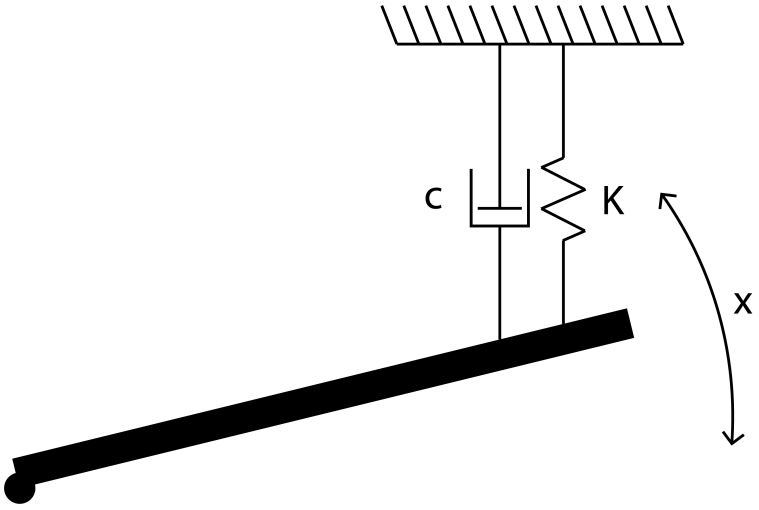


Figure 2.13: A schematic overview of the spring and damper connected to the pedal, both the spring and damper are in rotational domain

2.3.1 Harmonic oscillator

The pedals will be controlled to behave like a damped harmonic oscillator. A schematic drawing of the components is shown in figure 2.13. All motion for the pedals is in rotational domain. To return the pedal to the origin position, a virtual spring will be used based on the displacement of the pedal with respect to the origin position. This is described by Hooke's law as shown in equation 2.3.

$$F_s = -K\theta_p \tag{2.3}$$

Where F_s is the exerted force, K is the spring constant and θ_p the displacement of the spring. The constant has a minus sign because the exerted force is in the opposite direction of the displacement of the spring. To prevent the pedal from overshooting its target position, a virtual damper acting as friction is added to the controller based on the velocity of the pedal and a damping coefficient (equation 2.4).

$$F_d = -D\omega_p \tag{2.4}$$

Where F_d is the frictional force, D the damping coefficient and ω_p the velocity. The combination of the two above equations make the base of the controller and gives the following equation.

$$F = -K\theta_p - D\omega_p \tag{2.5}$$

With the controller equation defined, the damping ratio can be tuned to the desired value with the following equation.

$$\zeta = \frac{D}{2\sqrt{IK}} \tag{2.6}$$

Where ζ is the damping ratio, D the damping coefficient, I the inertia of the pedal and K the spring constant. The damping ratio will be tuned to be critically damped, this will return the pedal to its origin position as fast as possible while preventing it from oscillating. By rewriting the damping ratio equation as a function of D with a damping ratio of 1 as shown in equation 2.7, equation 2.5 can be rewritten as equation 2.8.

$$D = 2\sqrt{IK} \tag{2.7}$$

$$F = -K\theta_p - 2\omega_p\sqrt{IK} \tag{2.8}$$

As described in section 2.1, the haptic feedback implementation will alter the spring constant. However, the damping constant in equation 2.8 is now dependent on the spring constant, maintaining a constant damping ratio of 1.

2.3.2 Position

The encoder data will be used to measure the position, compensate gravity in the accelerometer and differentiate the position into the velocity. Because the pedal has a limited movement range, there is a finite number of steps divided over the movement range. If the starting position of the pedal is a set position, any number of steps will correspond to a unique pedal position. When the position is requested by the driver, the step count will be converted to a position in radians. The measured resolution of the encoder was 1,85E-3 degrees per step, this converts into 3,23E-5 radians per step. The offset of the pedals is 21,86 degrees which has to be added to the position. The resulting angle will also be used to compensate for the gravity in the accelerometer data, this is discussed in 2.3.3.

2.3.3 Acceleration

The accelerometer is used to measure the acceleration of the pedal and integrate it to estimate the velocity. The output of the accelerometer also includes gravity. Depending on the orientation, an accelerometer axis outputs between -1 to 1g when pointing down or up respectively. Because the used accelerometer axis is perpendicular to the pedal, the angle of the pedal is the same as the angle between the accelerometer axis and the gravity as

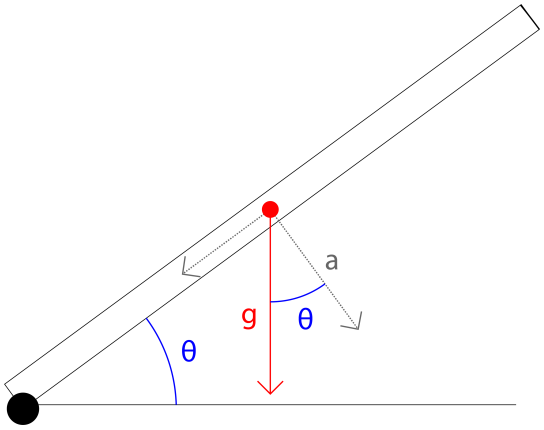


Figure 2.14: The magnitude of a can be calculated by multiplying g with the cosine of θ

shown in figure 2.14. By taking the cosine of θ and multiplying it by g ($9,81 ms^{-2}$) and then subtracting this from the measured acceleration, will result in the acceleration without the gravity component. This is tested by measuring both the accelerometer and encoder at the same time when attached to the pedal. The pedal is moved up and down twice. The measured pedal angle is used to calculate the gravity working on the accelerometer. The calculated gravity is subtracted from the measured acceleration, this should result in the acceleration of the pedal without gravity. The output is shown in figure 2.15. (1) shows the acceleration signal before and after the calculated gravity is subtracted (uncompensated and compensated respectively). (2) shows the angle of the pedal and the calculated magnitude of the gravity working on the accelerometer based on the pedal angle. If working correctly, the compensated acceleration in (1) should be 0 when the pedal angle is not changing. This is the case when the pedal is in its lowest position (T ≈ 0s - T ≈ 0,5s), but not in the highest (T ≈ 4,5s - T ≈ 5,5s). Looking at (2) it shows that the compensation is working as expected, a higher angle results in a lower compensated gravity. The error in the compensated acceleration could be causes by inaccurate calibration of both the encoder and accelerometer, this will have to be calibrated during the implementation phase.

2.3.4 Velocity

The velocity is calculated using the encoder and accelerometer output, the acceleration will be integrated and the position differentiated. There are three problems that need to be solved, the first is getting rid of the noise created by differentiating the position by means of a low-pass filter. The second is the drift that occurs when integrating the acceleration, this can be removed with a high-pass filter. Because of the low-pass filter it is possible that high frequencies in the differentiated velocity are not registered. This will be compensated by using the low-pass filtered integrated acceleration. The last problem is that both velocities need to be combined using a form of sensor fusion to get a usable signal.

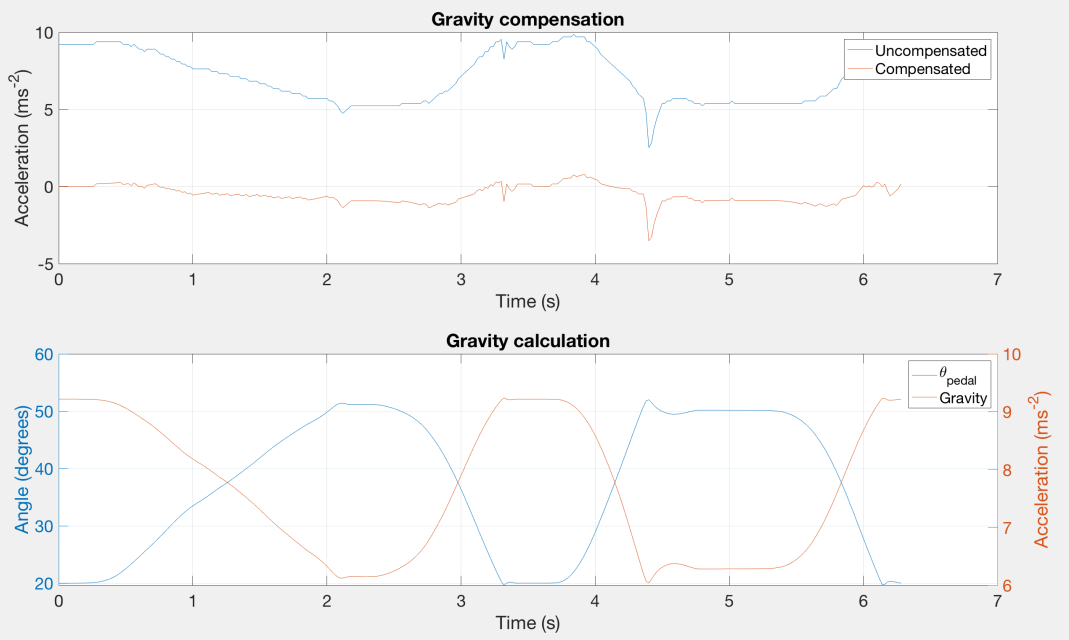


Figure 2.15: TTB (1) Acceleration signal before and after gravity compensation based on the pedal angle (2) Calculated gravity component based on the angle of the pedal

2.3.4.1 Differentiating position

The position signal from the encoder proves to be very suited to differentiate into velocity. Sample data is collected by moving the pedal up and down with the accelerometer attached as shown in figure 2.9 and the encoder connected to the motor, a sample frequency of 50hz was used. The pedal angle will be differentiated and then filtered with a low-pass filter to remove the expected high frequency noise. For the low-pass filter a second order Butterworth filter with a 10Hz cutoff frequency is used. The differentiation and filtering are done using Matlab.

Figure 2.16 (1) shows the angle of the pedal over time. The position signal is a clean signal without noise which is wanted because the differentiation of the signal will introduce noise, making the frequencies more separable. Figure 2.16 (2) shows both the unfiltered and filtered velocity from the differentiation with their respective frequency spectra in figure 2.18. Figure 2.17 shows a zoomed view on the first peak in figure 2.16 (2). As expected, the differentiation introduces noise in the velocity signal. However, the low-pass filter removes this noise, with as tradeoff a delay in the signal. The frequency spectrum of the filtered velocity shows a decrease of magnitude after 10Hz, indicating that the filter works as expected.

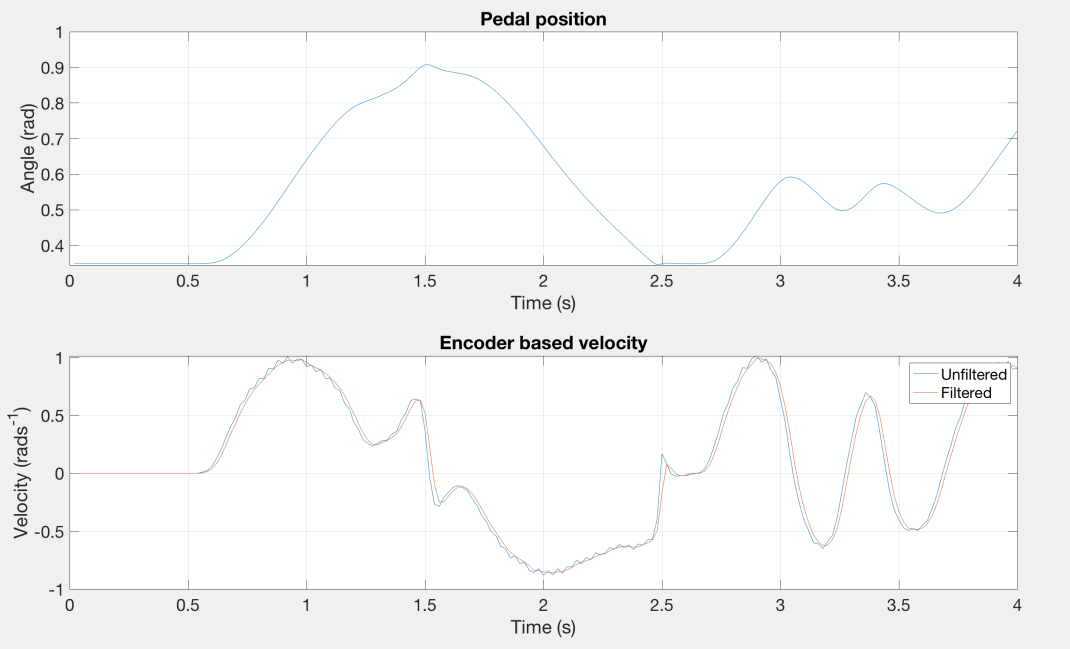


Figure 2.16: TTB (1) Encoder output converted into the angle of the pedal over time (2) The unfiltered and filtered outcome of integrating the position to angular velocity

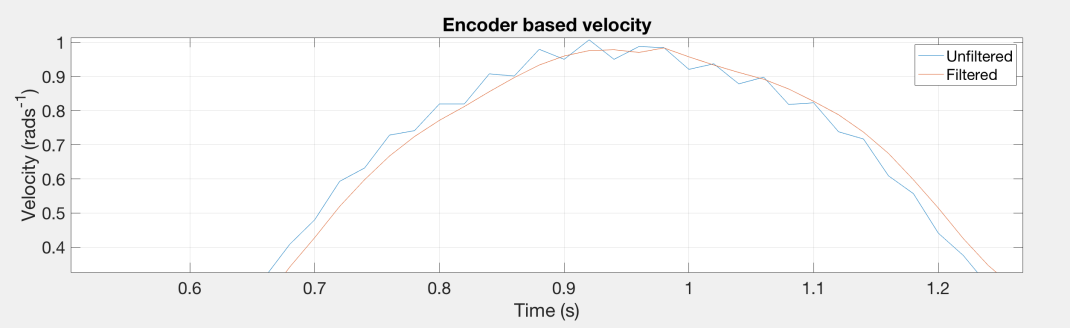


Figure 2.17: A zoomed in view of the unfiltered and filtered velocity

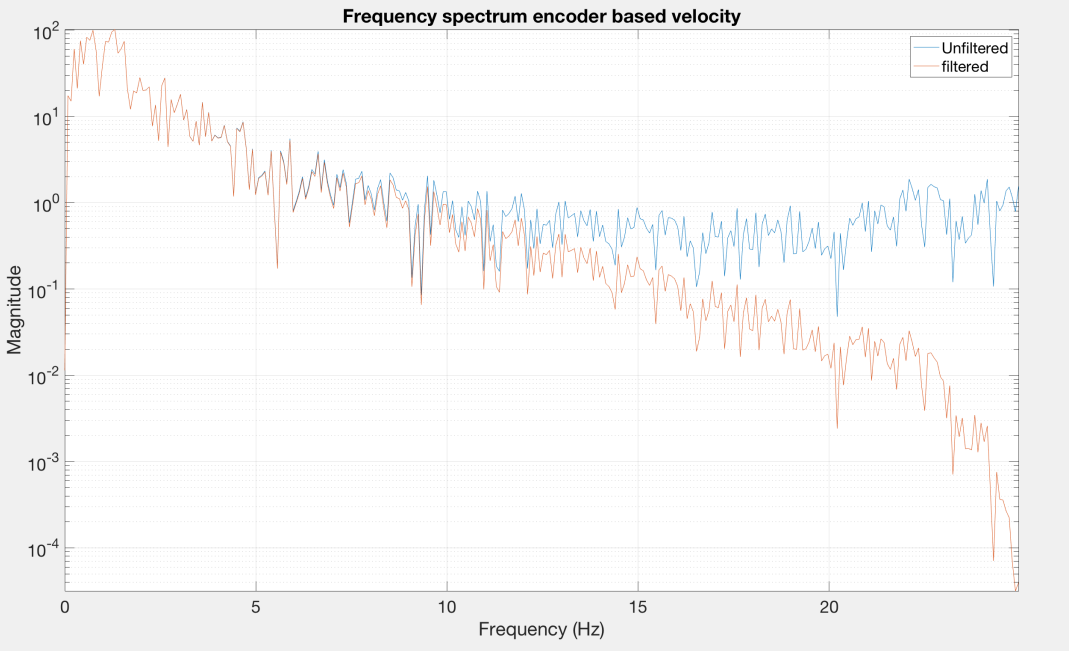


Figure 2.18: Frequency spectra of the unfiltered and filtered encoder based velocity

2.3.4.2 Integrating acceleration

The acceleration signal from the accelerometer is likely to contain an offset and high frequencies. This will cause the integrated signal to drift. The implementation of gravity compensation will get rid of a significant part of the offset, but not all. Using a high-pass filter, the remaining offset can be filtered out to create a usable velocity signal. The same sample data as in section 2.3.4.1 is used for testing. After integration a second order 2,5Hz high-pass Butterworth filter is applied to reduce the offset. Both the integration and filter are applied with Matlab.

The compensated acceleration signal is shown in figure 2.19 (1), the unfiltered and filtered velocity after integration are shown in (2). Both velocities are plotted on their own y-axis because of the significant range difference. The frequency spectra corresponding to the velocities are shown in figure 2.20. In the first 0,5s of the acceleration signal, the limiting resolution of the Arduino's ADC is visible, this will not be as much of a problem with the RaMstix because it has a 128 times larger resolution. The acceleration signal does not contain a lot of noise and is therefore usable. The unfiltered velocity signal contains a large amount of drift as expected, the signal is supposed to oscillate around the x-axis. After applying the high-pass filter, the signal oscillates around the x-axis as required. Because the low frequencies are filtered out, the velocity range is not as large as from the encoder, however, the peaks do not contain high frequency noise. The filter's effect is visible in the frequency spectrum where the 0Hz offset is almost completely removed. The delay introduced by the filter is not as clear, it can be seen in figure 2.19 (1) and (2) at 1,5s

where the peak in (2) appears later in time than in (1).

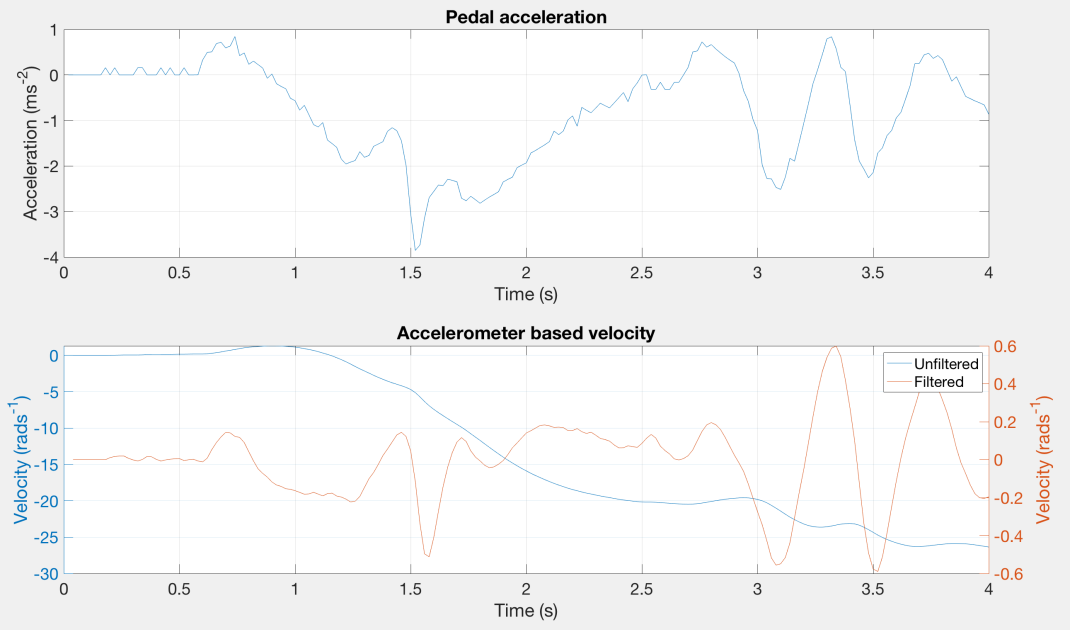


Figure 2.19: TTB (1) Gravity compensated accelerometer output (2) Unfiltered and filtered velocity from integrating acceleration

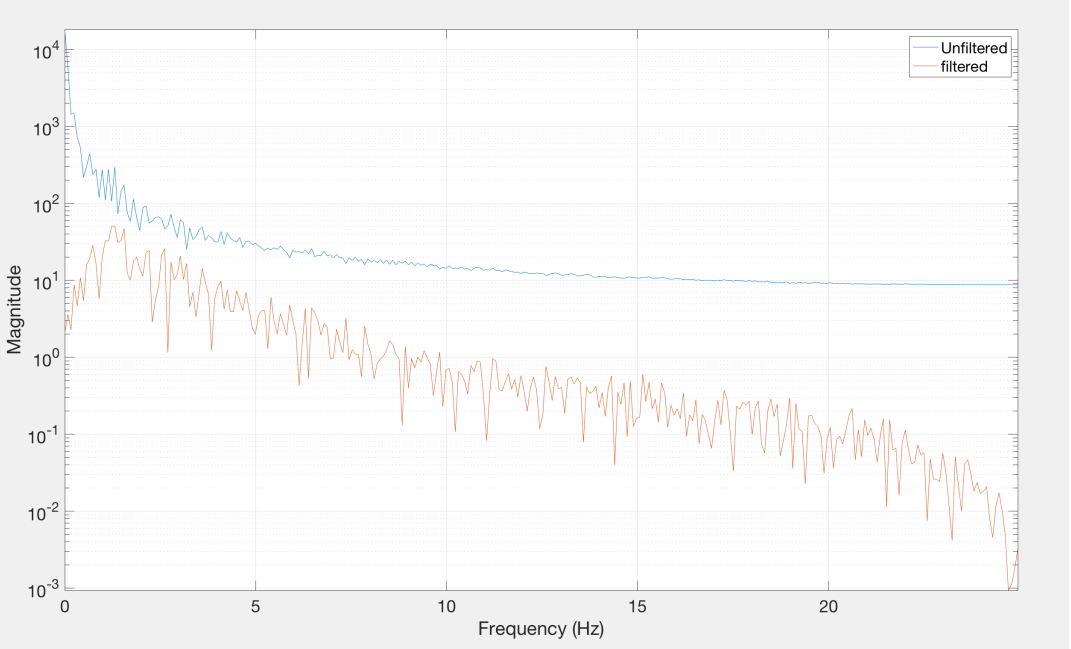


Figure 2.20: TTB (1) Frequency response unfiltered accelerometer output (2) Frequency response velocity based on integrated unfiltered accelerometer output

2.3.4.3 Sensor fusion

Both velocity signals from the accelerometer and encoder need to be filtered with a high-pass and low-pass filter respectively, which can be used to fuse the signals. Matching the cutoff frequencies of both filters makes it possible to sum the signals. Figure 2.21 shows the diagram of the summation of both signals after a first order filter. By rewriting the transfer function, it is proven that the combined signals equal the velocity. This is shown in equation 2.9 to 2.13.

$$v(s) = v(s) \cdot \frac{1}{s+1} + v(s) \cdot \frac{s}{s+1} \tag{2.9}$$

$$v(s) = \frac{v(s)}{s+1} + \frac{sv(s)}{s+1} \tag{2.10}$$

$$v(s) = \frac{v(s) + sv(s)}{s+1} \tag{2.11}$$

$$v(s) = \frac{(s+1)v(s)}{s+1} \tag{2.12}$$

$$v(s) = v(s) \tag{2.13}$$

This will be true for any order filter as long as both filters are equal in order and cutoff frequency.

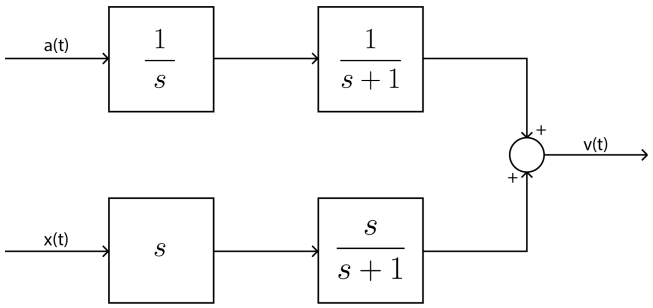


Figure 2.21: Diagram for summing the velocity values after integration/differentiation and filtering

2.3.4.4 Infinite Impulse Response filter

For the digital filtering of both velocity signals an Infinite Impulse Response (IIR) filter will be used. Either a Finite Impulse Response (FIR) or IIR filter can be used, FIR filters have as an advantage that the produces phase lag is linear and therefore predictable compared to IIR filters which have a less predictable phase lag, however, the phase lag is shorter. Because the phase lag is shorter in IIR filters they are more suitable for real-time applications and therefore they are chosen. During the design phase multiple IIR filters will be designed and tested with as goal an accurate velocity estimation using lower frequencies from the encoder and higher frequencies from the accelerometer. Another important aspect of the filters that needs to be tested is the delay, because when too large, the feedback could be noticeably delayed, reducing the effect of the feedback. The estimated velocity will have a delay of:

$$Delay = \frac{N}{f_s} \tag{2.14}$$

Where N is the order of the filter and f_s the operating frequency. The filters also introduce a phase shift in the signal. Figure 2.22 and 2.23 show the phase shift of a high-pass and low-pass filter respectively. Both have a 10Hz (0,4 normalized frequency) cutoff frequency at a sample frequency of 50Hz. The low-pass filter introduces a 0 to -90° phase shift up to the cutoff frequency, and the high-pass filter introduces a 90 to 0° phase shift after the cutoff frequency.

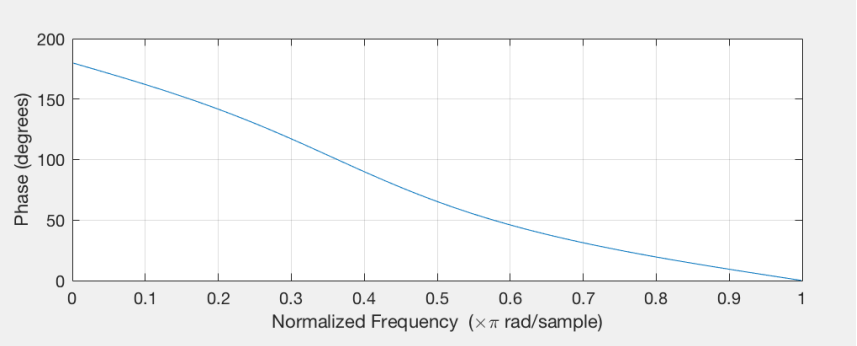


Figure 2.22: Phase shift for a second order high-pass Butterworth filter with a 10Hz (0,4 normalized frequency) cutoff frequency at a 50Hz sample frequency

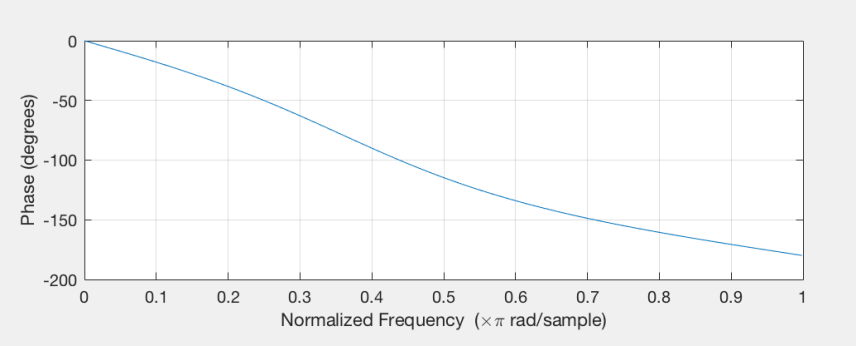


Figure 2.23: Phase shift for a second order low-pass Butterworth filter with a 10Hz (0,4 normalized frequency) cutoff frequency at a 50Hz sample frequency

2.4 Software architecture

The software architecture is built up of four different parts: the pedal driver, controller, feedback calculation and interpretation. The communication between the cockpit and the robotic platform is setup using ROS (Robot Operating System). ROS provides a framework for writing robot software. The framework allows for writing code in a modular approach which will be used for the separation of the four parts. A simplified schematic of the interaction between the four nodes (software module) is shown in figure 2.24. This section will elaborate on the function and the in- and output of each node.

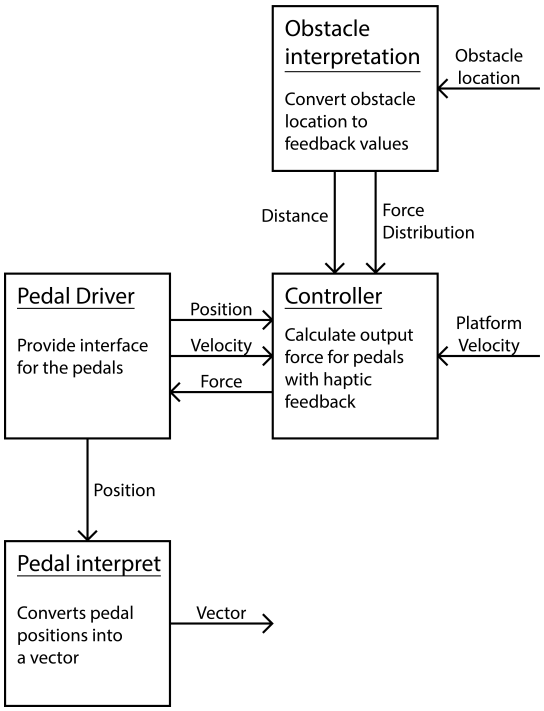


Figure 2.24: Overview of the different nodes in ROS to operate the pedals

2.4.1 Pedal driver

The pedal driver is directly connected to the I/O of the RaMstix and will publish the sensor data and subscribe to the force signal that will be send to the motor. This is a separate ROS node so that when the hardware of the pedals changes, only this driver node has to be changed in order to keep the system working. The driver node will convert the sensor values into SI units and the actuator values into voltage values. Also the gravity compensation of the acceleration and differentiation, integration and filtering for estimating the velocity is done in this node.

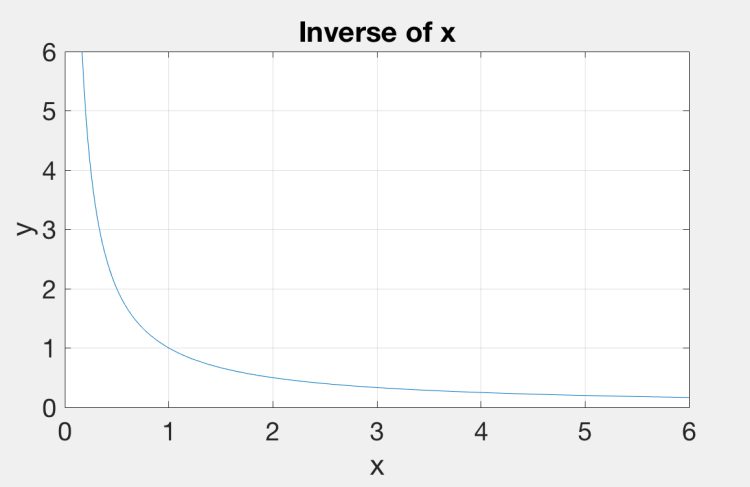


Figure 2.25: Graph for the formula $y = \frac{1}{x}$

2.4.2 Pedal controller

The controller node is the node that calculates the force that the pedals will exert onto the user. How this node calculates the force is described in section 2.3.1. This node requires the position and velocity of the pedal for control without haptic feedback. With haptic feedback, the spring constant in the controller is dependent on the distance and location of an obstacle, and the platform's velocity. Because the increase in spring constant will be calculated in this node, those variables are therefore required as input as well.

To implement the haptic feedback in the controller, the spring constant is made a function instead of a constant. First a base spring constant is required for the operation of the pedal without haptic feedback. Then, the increase in spring constant will be determined by the distance to an obstacle. As determined in section 2.1.4, the haptic feedback based on the distance to an obstacle should act like a non-linear spring. More specific, the haptic feedback should become exponentially larger when the platform closes in on an obstacle. The curve should range from maximum spring constant increase at minimal detection range to no spring constant increase at maximum detection range. This behaviour is similar to the function $y = \frac{1}{x}$ as shown in figure 2.25, where y becomes larger when x decreases. Therefore the increase in spring constant will be a variation of $y = \frac{1}{x}$, tuned to the characteristics of the system. The general equation for this is given in equation 2.15

$$K_l = \frac{m}{l_o + n} + p \tag{2.15}$$

Where K_l is the increase in spring constant based on the distance to an obstacle, l_o the distance to the obstacle and m, n and p variables for determining the curve based on the characteristics of the obstacle detection and the maximum spring increase.

Then, to indicate the location of the obstacle, the increase in spring constant is scaled for each pedal based on an algorithm described in section 2.4.3.2. The distribution for each pedal is between 0 and 1, which the increase is multiplied by. Adding to equation 2.15 gives the following equation:

$$K_i = K_l \cdot U \tag{2.16}$$

Where K_i is the increase in spring constant based on the distance and location of an obstacle and U the distribution for the corresponding pedal.

Last, the platform's velocity is added to the spring increase equation. The faster the platform goes, the larger the spring constant should be. However, the feedback should also be noticeable while moving slowly. To accommodate for both requirements, the platform's velocity is converted to a 0,5 to 1 scale which the spring constant increase is multiplied by. The equation for the conversion is:

$$V = \frac{0,5v_r}{v_{max}} + 0,5 \tag{2.17}$$

Where V is the converted velocity, v_r the platform's current velocity and v_{max} the platforms maximum velocity. Combining equation 2.17 with 2.16 results in the following equation.

$$K_i = K_l \cdot U \cdot V \tag{2.18}$$

Adding the final equation for the increase in spring constant to the controller equation (equation 2.8 in section 2.3.1) results in the final controller equation as shown in equation 2.19.

$$F = -(K_b + K_i)\theta_p - 2\omega_p\sqrt{I(K_b + K_i)} \tag{2.19}$$

Where F is the output force, K_b the base spring constant, K_i the spring constant increase, θ_p the pedal position, ω_p the pedal velocity and I the pedal's inertia.

2.4.3 Obstacle interpretation

The obstacle interpretation node outputs the distance to, and force distribution based on the relative angle of the obstacle. The node receives two vectors containing the location of the obstacle with respect to the orientation of the robotic platform. The vectors are parallel and perpendicular to the platform, shown in figure 2.26 as X_o and Y_o respectively. The vectors are assumed to have their origin in the centre of the robotic platform.

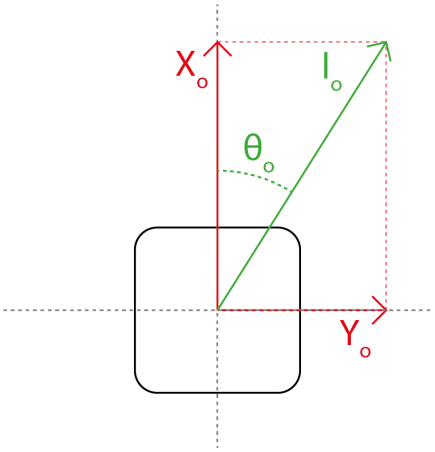


Figure 2.26: Calculation of the obstacle distance based on the input vectors

2.4.3.1 Obstacle distance

To determine the distance to the obstacle, Pythagoras' theorem can be used. The magnitude of the combined vector can be calculated as shown in equation 2.20.

$$l_o = \sqrt{X_o^2 + Y_o^2} \tag{2.20}$$

Where l_o is the distance to the obstacle, X_o the vector's component in parallel direction and Y_o the vector's component in perpendicular direction. The magnitude is then passed to the controller node as the distance to the obstacle.

2.4.3.2 Obstacle location

For the force distribution, the angle of the vector with respect to the orientation of the robotic platform is used and can be calculated using basic trigonometry.

$$\theta_o = \arctan \frac{X_o}{Y_o} \tag{2.21}$$

Where θ_o is the angle to the obstacle in radians with respect to the direction of the robotic platform. The angle is then converted to a distribution between the two pedals. Figure 2.27 shows how the conversion of the angle to the obstacle and the distribution between the two pedals will be done. The combined distribution is always 1, resulting in the red diamond shape. The distribution for any vector is based on the point of intersection between the vector's angle θ_o and the distribution diamond, point A in figure 2.27. The distribution for one pedal is between 0 and 1, based on the angle of the obstacle. Each $0{,}5\pi$ quarter alters the distribution with at most 0,5 per pedal based on the quarter. Equation 2.22 - 2.25 give the distribution of the left pedal for each quarter.

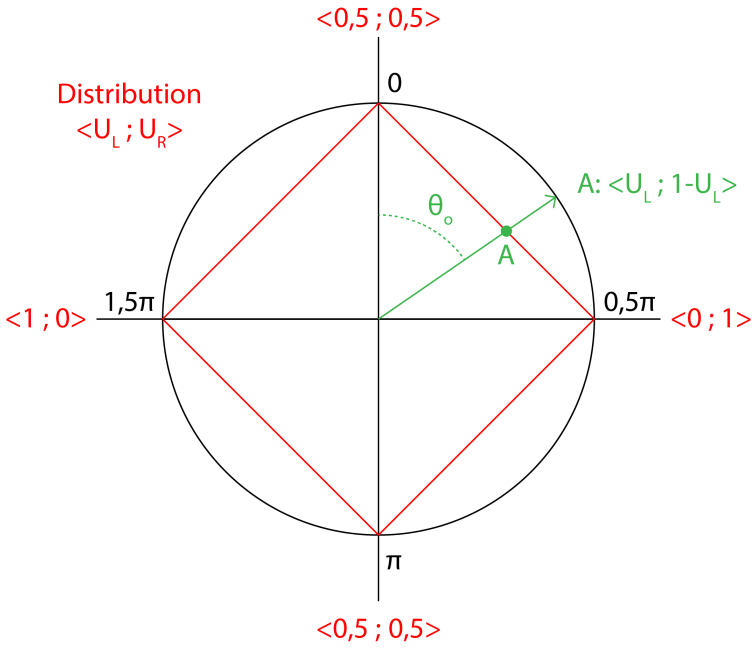


Figure 2.27: Calculation of the pedal force distribution based on the obstacle angle

Between 0 and 0,5π:

$$U_L = \frac{0,5 \cdot \theta_o}{0,5\pi} \tag{2.22}$$

Between 0,5 and π:

$$U_L = \frac{0,5 \cdot (\theta_o)}{0,5\pi} \tag{2.23}$$

Between π and 1,5π:

$$U_L = 0,5 + \frac{0,5 \cdot (0,5\pi - |\theta_o|)}{0,5\pi} \tag{2.24}$$

Between 1,5π and 2π:

$$U_L = 1 - \frac{0,5 \cdot (0,5\pi - \theta_o)}{0,5\pi} \tag{2.25}$$

Because the combined distribution is always 1, the distribution for the right pedal is:

$$U_R = 1 - U_L \tag{2.26}$$

2.4.4 Pedal interpretation

In order to drive the robotic platform, the position of both pedals has to be converted into a twist with two degrees of freedom. A twist expresses velocity in free space broken into its linear and angular parts. Because the robotic platform is assumed to be differentially driven and moving over a plane, only one linear and one angular component is used. There are four extremes in the possible motion of the robotic platform: forwards

and backwards at maximum velocity and clockwise and counterclockwise rotation on the spot. In order to achieve all four motions with the two pedals, the pedal position is interpreted as shown in figure 2.28. Each pedal controls its corresponding side of the robotic platform. The pedal controller is supposed to have its origin position in the centre of the pedal range, allowing for motion in two directions. Pushing the pedal downwards results in a forward motion on the robotic platform, and pulling the pedal up results in a backwards motion. Using two pedals, all four extremes of motion can be achieved.

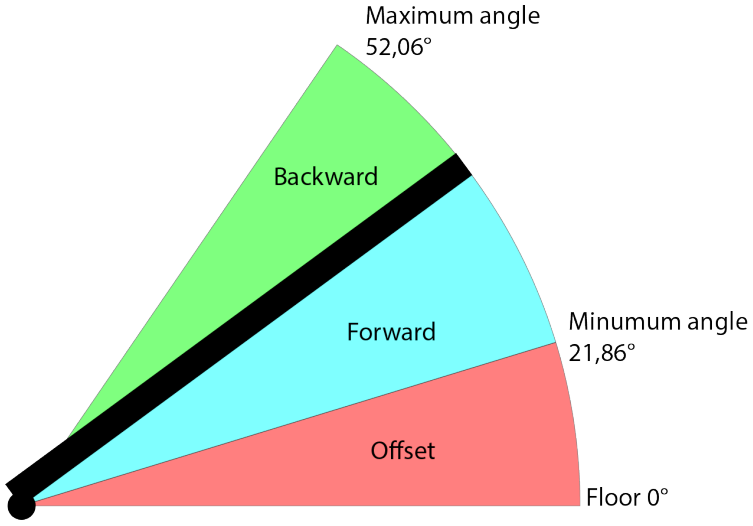


Figure 2.28: Pedal range and corresponding direction

For the conversion of the pedal positions to a twist, the pedal range is first converted to a displacement of -50% to 50% with 0 being the origin position. Using this range, all motions are based on the ratio between the two pedals. Moving forwards at maximum velocity, both pedals are at 50% summing up to 100% with a 0% difference. For a clockwise rotation on the spot the left pedal is at 50% and the right at -50%, making the sum 0% and the difference 100%. Table 3 gives the pedal values for all four extreme motions and the corresponding sum and difference. The sum and difference are then used for the twist's linear and rotational speed. Using the predefined maximum velocity for the wheels multiplied by the sum and difference for the linear and rotational velocity respectively, to construct a twist.

Table 3: Extreme motions, corresponding pedal positions and resulting twist

Motion extremes	**Pedal values <%Left, %Right>**	**Sum**	**Difference**	**Twist <Linear, Rotational>**
Forwards	<50, 50>	100	0	<max, 0>
Backwards	<-50, -50>	-100	0	<-max, 0>
Rotate clockwise	<50, -50>	0	100	<0, max>
Rotate counterclockwise	<-50, 50>	0	-100	<0, -max>

3 Design & implementation

In this design and implementation section the design based on the plan in the analysis will be discussed. First, the RaMstix setup will be discussed in section 3.1. All connection diagrams, hardware components and specifications are documented in this section. Second, the software architecture for the system is written and divided in parts called nodes in section 3.2. Each node is described in terms of input/output and all the functions it fulfils. And finally, in section 3.3 the mechanical alterations are discussed. The final implementation is shown in figure 3.1.

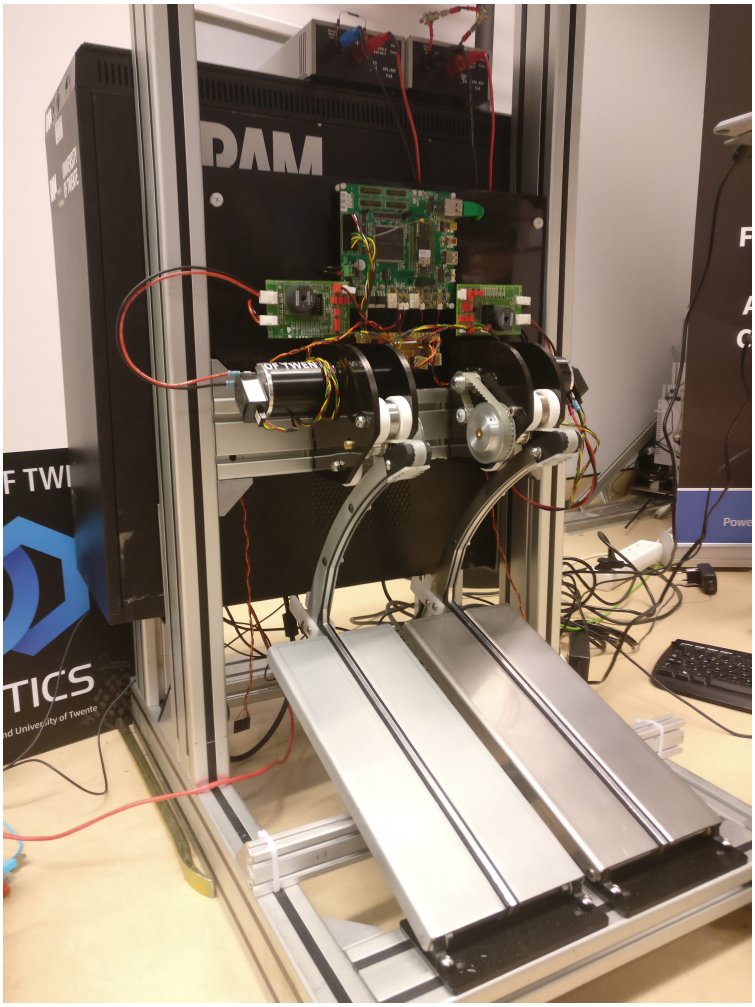


Figure 3.1: Overview image of the physical implementation

3.1 Electrical

Because the embedded computer (TS-7300) is replaced by a RaMstix, all connections will need to be redone. In this section all the new hardware connections will be described and illustrated. Illustrations will be based on figure 3.2 and 3.3. Figure 3.3 is a connection board for SV10, X5 and X6 of the RaMstix. K3 and K4 on the connection board can be used for extended signal processing, this is not used and are therefore directly connected.

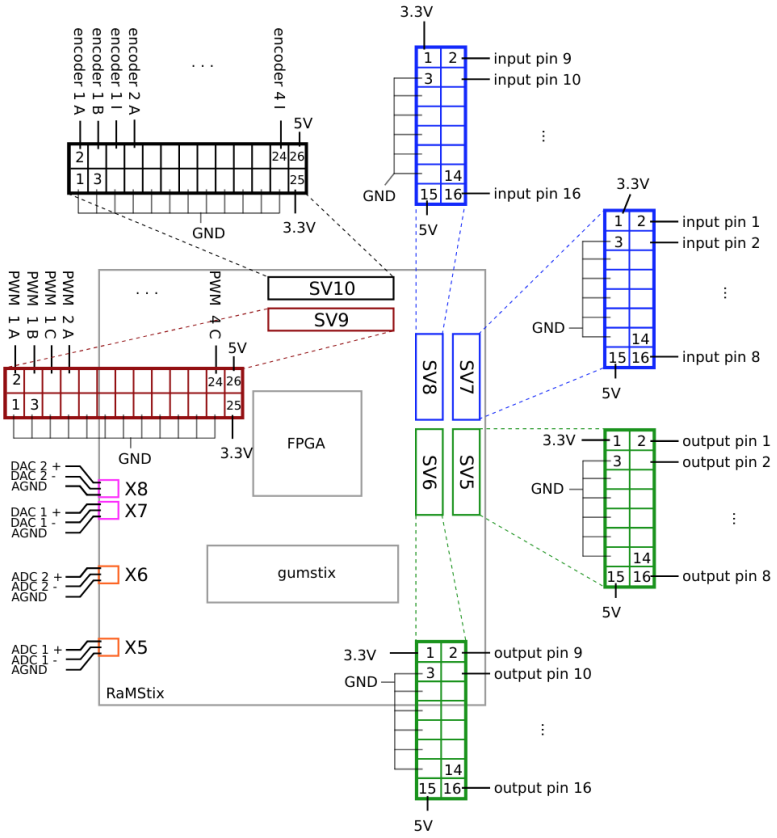


Figure 3.2: Schematic pin overview of the RaMstix [18]

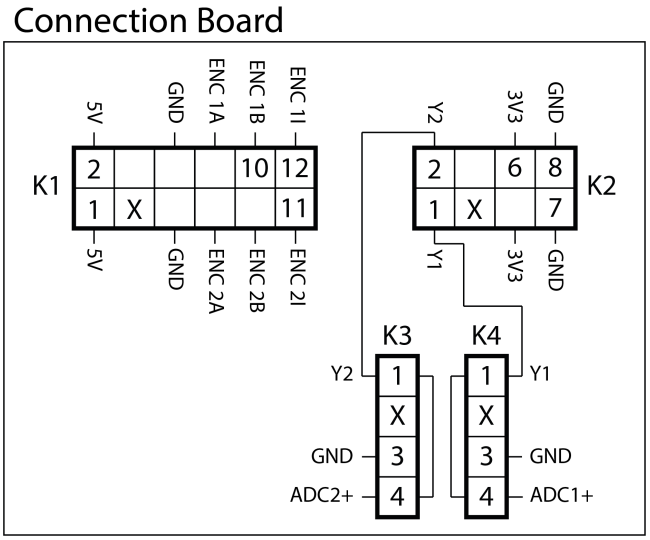


Figure 3.3: Schematic pin overview of the connection board to the RaMstix

3.1.1 Encoder

The encoders will be connected to the RaMstix's encoder inputs via the connection board. SV10 in figure 3.2 has connections for up to four encoder inputs, one 5V output and ground connections. The first two encoder inputs have been wired to K1 on the connection board, including the 5V output and ground. The two used encoders are wired to encoder input 1 & 2 as shown in figure 3.4. The index pin does not have to be connected since the encoder register will be reset every reboot as explained in section 2.2.3, but will be connected for possible future implementation.

3.1.2 Accelerometer

The accelerometers attached to the bottom of the pedals are connected to the connection board as shown in figure 3.5. Because the accelerometers output an analog signal, the connection board is wired to the ADCs of the RaMstix (X5 and X6 in figure 3.2). Only the Y-axis of the accelerometer is required and therefore connected. The GS1 and GS2 on the accelerometers are for selecting the sensitivity, leaving both disconnected gives a range of ±1,5g because of an internal pulldown, which is the desired sensitivity. The SLP pin can be used to put the accelerometers to sleep, however this is not required and is therefore directly connected to 3,3V. The negative input of the ADC is connected to the ground because the accelerometer outputs a positive voltage between 0 and 3,3V.

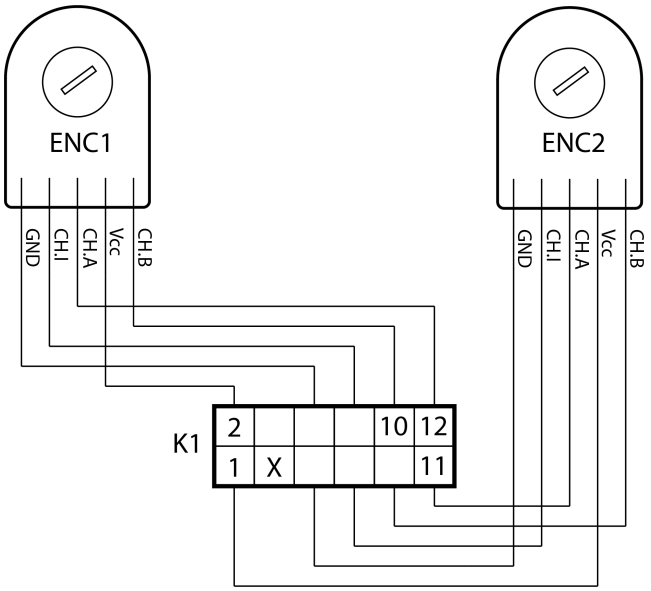


Figure 3.4: Schematic wiring of the encoders to the connection board

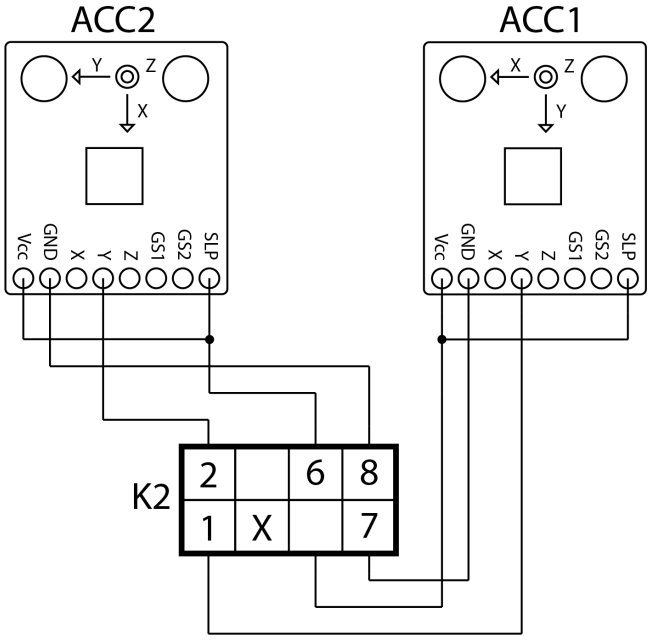


Figure 3.5: Schematic wiring of the accelerometers to the connection board

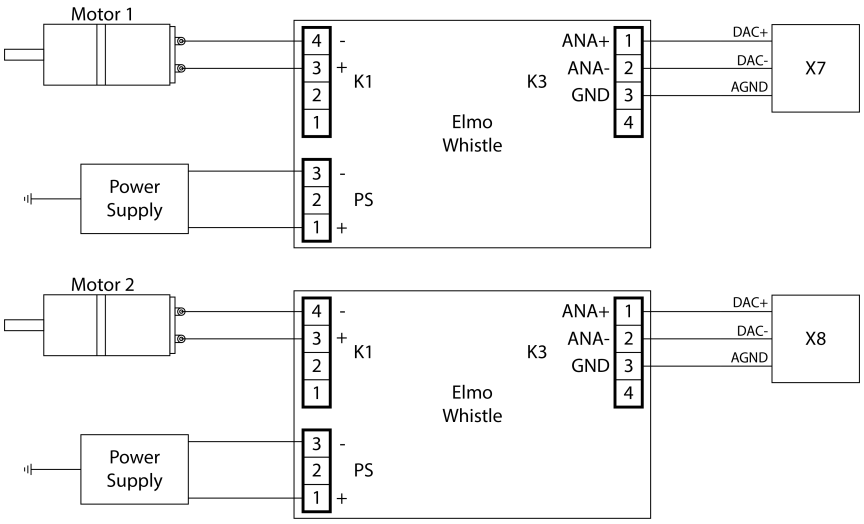


Figure 3.6: Schematic wiring of the motors connected via an Elmo Whistle to the RaMstix, the Elmo is connected to a power supply to drive the motors

3.1.3 Motor, Elmo Whistle and power supply

The motors are controlled by an Elmo Whistle each, which regulate the current going to the motors. A schematic drawing of the wiring is shown in figure 3.6. The Elmos are directly connected to the RaMstix. The power used to drive the motors is supplied by two power supplies directly connected to the Elmos. The set-point for the output current of the Elmo is regulated with an analog voltage supplied by the RaMstix. The RaMstix is able to output a voltage of ±2,5V from the DAC. The motors have a maximum continuous current of 4,58A, which in combination with the DAC voltage translates into a 1,832A/V control ratio, both positive and negative.

3.2 Software

As stated in section 2.4 of the analysis, ROS will be used as framework for the software. This gives the possibility to divide the software into separate nodes which will be described in this section. First an explanation of the usage of ROS will be given. Second, the driver node will be elaborated upon, forming the bridge between the pedals and the robotic platform. Third, the controller node where an algorithm for the output force of the motors based on the pedal's position, velocity and feedback is implemented will be discussed. Fourth, the algorithm for the distance to an obstacle and distribution of feedback is discussed. And last, a node for the conversion of the pedal positions to a twist is discussed. A schematic overview of the communication between nodes is shown in figure 3.7. A change from the analysis chapter is that the pedal driver node is separate from the RaMstix.

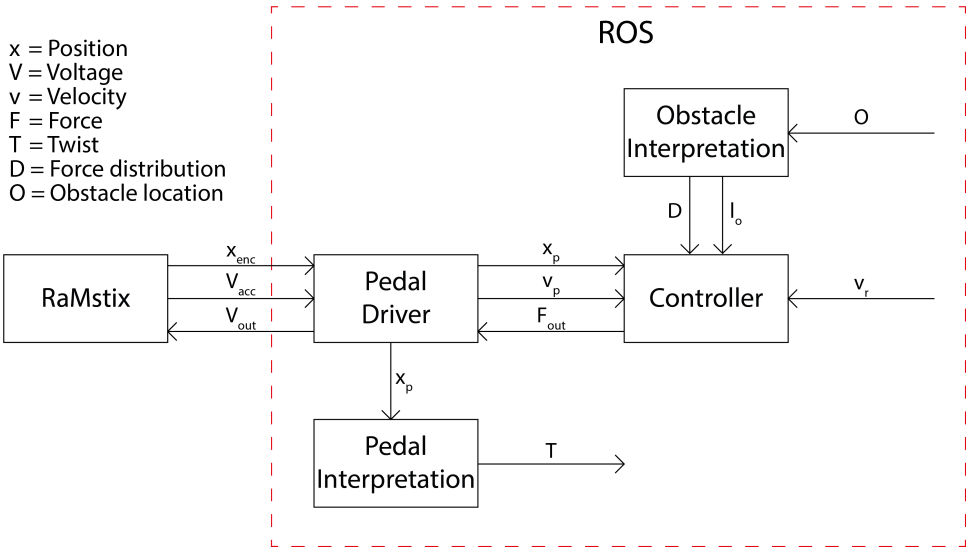


Figure 3.7: Schematic overview of the software architecture including the signal flows

3.2.1 ROS usage

For the communication between different nodes, ROS uses topics on which messages are send. Messages can be standard or custom made and can be communicated in different ways. For this software architecture two communication protocols will be used: asynchronous and synchronous. A synchronous communication consists of one client and one server. The client sends a request to the server and waits for a reply. The server will only send a message whenever it receives a request. A visual representation is shown in figure 3.8. In an asynchronous communication there is a topic to which can be published or subscribed without the need of a request. Multiple publishers can publish to the topic and multiple subscribers can subscribe to the topic at the same time. A visual representation with one publisher and multiple subscribers is shown in figure 3.9.

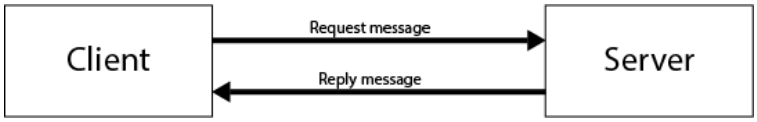


Figure 3.8: Schematic overview of a synchronous communication

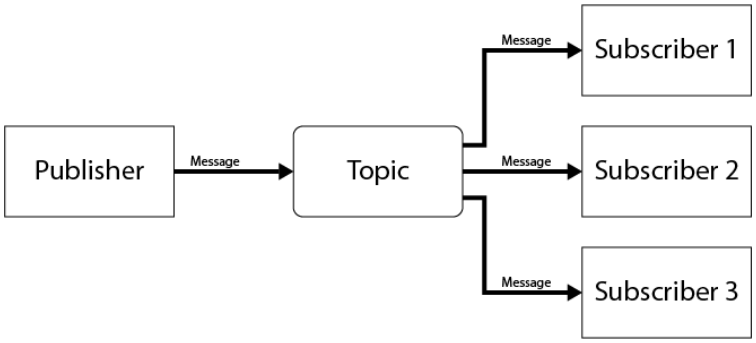


Figure 3.9: Schematic overview of a asynchornous communication with one publisher and three subscribers

3.2.2 Pedal driver

The pedal driver node is a node that requests the sensor values from the RaMstix and converts them into SI units which can be used by the other ROS nodes. The node also estimates the velocity of the pedal based on the position and acceleration. The node will have three main functions which are described in this section: requesting sensor values, converting sensor values and estimating velocity. A schematic overview of the in- and output of the pedal driver node is shown in figure 3.10.

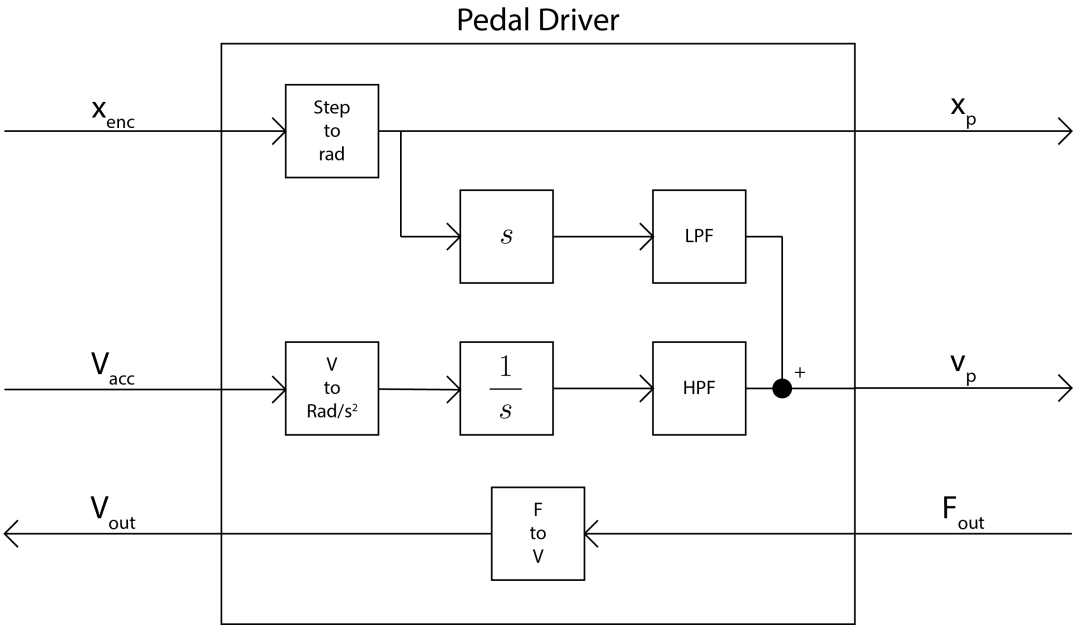


Figure 3.10: Schematic overview of the in- and output of the pedal driver node connected to the RaMstix

3.2.2.1 Requesting sensor values

For the communication between the RaMstix and the pedal driver node, the ZeroMQ [19] library has been used. ZeroMQ provides the same synchronous communication as with ROS. First, the received output force from the controller node is mapped to the DAC's output range. Then, the pedal driver node sends a request to the RaMstix, containing the output voltage values of the controller. The RaMstix then sets the DACs to the received voltages and retrieves the sensor values from the registers. The encoder counts and accelerometer voltages are send back as the reply message to the pedal driver node. A synchronous communication is used in order to set the frequency at which the RaMstix runs from the pedal driver node.

3.2.2.2 Standardising sensor values

In the pedal driver node the received sensor values from the RaMstix are converted into SI units. First the encoder steps are converted to radians using the 3,23E-5 $rad/step$ and the 21,86° (3,82E-1 rad) offset, determined in section 2.3.2, as shown in equation 3.1.

$$\theta_p = 3,82E-1 + x_{enc} \cdot 3,23E-5 \tag{3.1}$$

Where θ_p is the pedal angle in radians and x_{enc} the pedal position in steps.

Next, the accelerometer voltage is converted to $rads^{-2}$. The accelerometer has an offset voltage which is subtracted from the measured voltage. Then the voltage is divided by the sensitivity of the accelerometer and multiplied by the gravitational force ($9,81ms^{-2}$), resulting in the acceleration in ms^{-2}.

$$a_{mes} = \frac{(V_{in} - V_{off}) \cdot g}{S_{acc}} \tag{3.2}$$

Where a_{mes} is the measured acceleration in ms^{-2}, V_{in} the measured voltage, V_{off} the offset voltage of the accelerometer, g the gravitational force and S_{acc} the accelerometer's sensitivity.

Then the signal is compensated for gravity as discussed in section 2.3.3 of the analysis, resulting in equation 3.3.

$$a_{comp} = \frac{(V_{in} - V_{off}) \cdot g}{S_{acc}} - \cos(\theta_p) \cdot g \tag{3.3}$$

Where a_{comp} is the compensated acceleration in ms^{-2} and θ_p the position of the pedal in radians.

For the conversion from ms^2 to $rads^{-2}$, equation 3.4 is used.

$$v = r\omega \tag{3.4}$$

Where v is the linear velocity, r the radius and ω the angular velocity.

Combining equation 3.3 and 3.4 result in equation 3.5 for the calculation of the rotational acceleration of the pedal.

$$a_p = \frac{\frac{V_{in}-V_{off}}{S_{acc}} \cdot g - \cos(\theta_p) \cdot g}{r_{acc}} \tag{3.5}$$

Where a_p is the pedal acceleration in $rads^{-2}$ and r_{acc} is the radius of the motion of the accelerometer in m.

3.2.2.3 Calculating velocity

Using the pedal position and acceleration to estimate velocity, they need to be differentiated and integrated respectively. For differentiating the position the difference quotient is used as shown in formula 3.6.

$$\omega(t) \approx \frac{\theta_p(t_k) - \theta_p(t_{k-1})}{\Delta t_k} \tag{3.6}$$

Where $\omega(t)$ is the angular velocity, θ_p(t) the pedal's position, t the time and k the current measurement.

For the integration of the acceleration the trapezoidal rule is used as shown in formula 3.7.

$$\omega(t) \approx \sum_{k=1}^{N} \frac{a(t_k) + a(t_{k-1})}{2} \Delta t_k \tag{3.7}$$

Where $\omega(t)$ is the angular velocity, a(t) the pedal's angular acceleration, t the time and k the current measurement.

After both velocities are acquired, they are filtered to prevent noise and allow sensor fusion as described in section 2.3.4.3 of the analysis. Two cut-off frequency matched second-order Butterworth filters are used, one being a low-pass and the other a high-pass filter. Both are IIR filters using the general transfer function as shown in equation 3.8, with coefficients generated by Matlab.

$$\omega_f[t_k] = \frac{1}{b_0}\left(\sum_{i=0}^{P} c_i \omega_u[t_{k-i}] - \sum_{j=1}^{Q} b_j \omega_f[t_{k-j}]\right) \tag{3.8}$$

Where:
- ω_f is the filtered angular velocity
- ω_u is the unfiltered angular velocity
- b_i are the feedback filter coefficients
- c_i are the feedforward filter coefficients
- Q is the feedback filter order
- P is the feedforward filter order

Both filters have a 10Hz cutoff frequency at a 200Hz sample frequency. The frequency response and phase shift are shown in figure 3.11 and 3.12 for the low-pass and high-pass filter respectively. After both velocity signals are filtered, they are summed and send to the controller node.

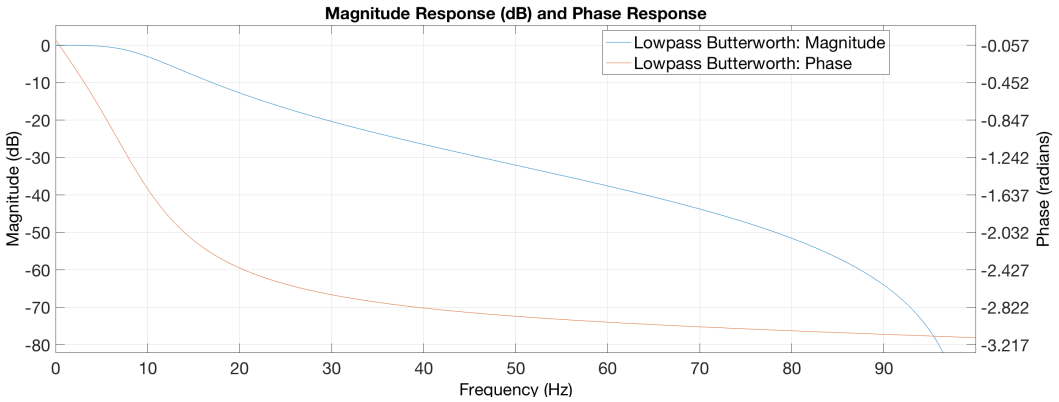


Figure 3.11: Plot of the frequency response and phase lag of a second order low-pass Butterworth filter with a cutoff frequency of 10Hz and a sampling rate of 200Hz

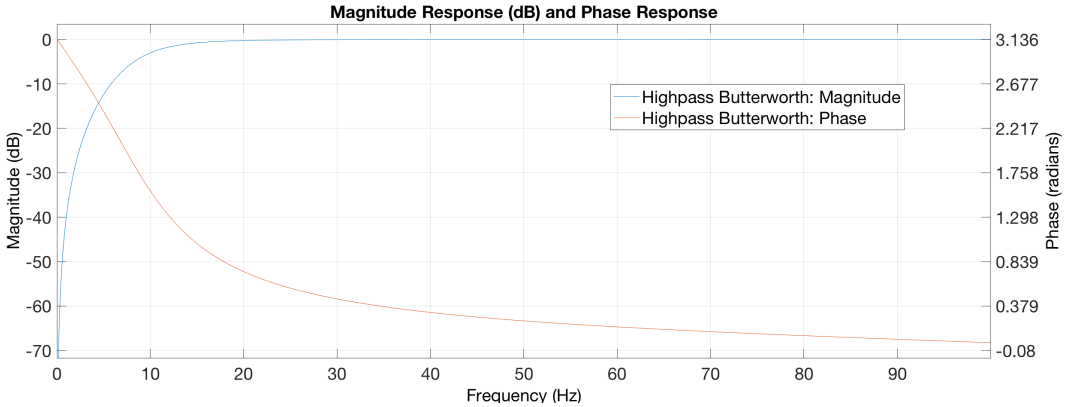


Figure 3.12: Plot of the frequency response and phase lag of a second order high-pass Butterworth filter with a cutoff frequency of 10Hz and a sampling rate of 200Hz

3.2.3 Controller

The controller node determines the output force applied to the pedal based on multiple inputs. It receives the pedals' position and velocity from the pedal driver node. The controller node also subscribes to the message of the obstacle interpretation node and the robot's velocity, which it uses to alter the spring and damper constants, creating the haptic feedback effect. A schematic overview of the in- and output and signal flow of the node is shown in figure 3.13.

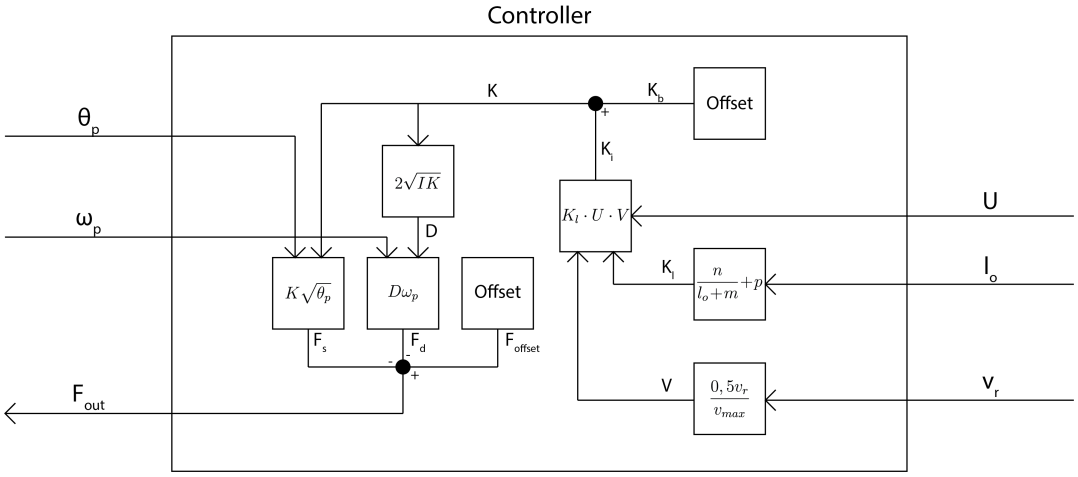


Figure 3.13: Schematic overview of the in- and output of the spring-damper node

First, a base spring-damper controller based on section 2.3.1 of the analysis is implemented. The base controller operates without haptic feedback ($K_i = 0$) and has a minimum spring constant K_b to return the pedal to its origin position. First, an offset force is added to the output force to counter the gravity as result of the weight of the pedal. Adding a force offset to equation 2.19 of the analysis with $K_i = 0$ results in equation 3.9.

$$F_{out} = -K_b\theta_p - 2\omega\sqrt{IK_b} + F_{off} \tag{3.9}$$

Where F_{out} is the output force of the controller, F_{off} the offset force, K_b is the base spring constant, ω the rotational velocity and I the pedal's inertia.

Without haptic feedback, the pedal should exert a large enough force to not feel loose, but small enough to not be perceived as feedback. To achieve a tight feeling, the square root of the displacement is taken because of the square root's shape as shown in figure 3.14. A small displacement results in a relatively large output force, preventing accidental movement when a foot is rested on the pedal. Altering equation 3.9 gives equation 3.10, forming the base spring-damper algorithm for the controller.

$$F_{out} = -K_b\sqrt{\theta_p} - 2\omega_p\sqrt{IK_b} + F_{off} \tag{3.10}$$

When $K_i \neq 0$, the spring constant is a function of the distance to an obstacle, force distribution and velocity of the robot. The distance to an obstacle and the robot velocity are converted to K_i and V, using equation 2.15 and 2.17 of the analysis respectively as shown in figure 3.13. Then, K_l, U and V are multiplied resulting in K_i. K_i and K_b are then summed, resulting in K.Including K_i in equation 3.10 results in equation 3.11 as the transfer function of the controller node.

$$F_{out} = -(K_b + K_i)\sqrt{\theta_p} - 2\omega\sqrt{I(K_b + K_i)} + F_{off} \tag{3.11}$$

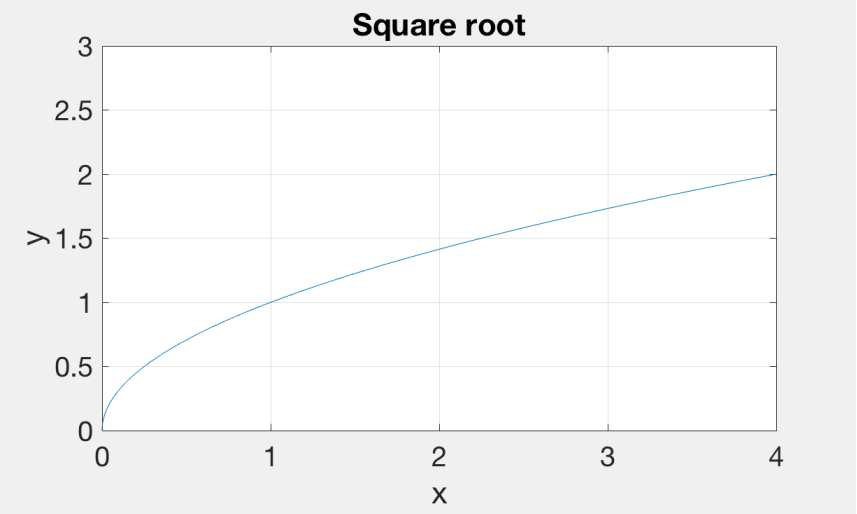


Figure 3.14: Graph for the formula $y = \sqrt{x}$

3.2.4 Obstacle interpretation

The obstacle interpretation node is subscribed to the obstacle location message and publishes a message containing the force distribution for the pedals and the distance to the obstacle. The signal flows of the node are shown in figure 3.15.

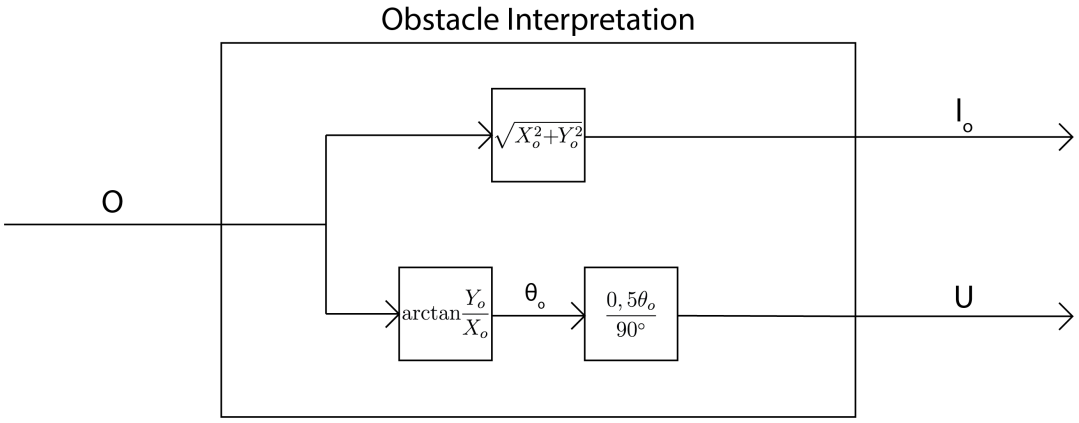


Figure 3.15: Schematic overview of the signal flows of the obstacle interpretation node

The obstacle location message contains two vectors, a parallel X_o and a perpendicular Y_o with respect to the robot direction, as described in section 2.4.3. Using equation 3.12, the distance to the obstacle l_o is calculated.

$$l_o = \sqrt{X_o^2 + Y_o^2} \tag{3.12}$$

For the force distribution, first the angle to the obstacle is calculated using equation 3.13.

$$\theta_o = \arctan \frac{X_o}{Y_o} \tag{3.13}$$

Where θ_o is the relative angle to the obstacle in radians. Then, equation 3.14 to 3.17 are used to determine the force distribution on a 0 to 1 scale for the left pedal.
θ_o between 0 and 0,5π:

$$U_L = \frac{0,5 \cdot \theta_o}{0,5\pi} \tag{3.14}$$

θ_o between 0,5 and π:

$$U_L = \frac{0,5 \cdot (\theta_o)}{0,5\pi} \tag{3.15}$$

θ_o between π and 1,5π:

$$U_L = 0,5 + \frac{0,5 \cdot (0,5\pi - |\theta_o|)}{0,5\pi} \tag{3.16}$$

θ_o between 1,5π and 2π:

$$U_L = 1 - \frac{0,5 \cdot (0,5\pi - \theta_o)}{0,5\pi} \tag{3.17}$$

Where U_L is the distribution for the left pedal. The equation for the distribution for the right pedal is shown in equation 3.18.

$$U_R = 1 - U_L \tag{3.18}$$

3.2.5 Pedal interpretation

The pedal interpretation node contains the algorithm for converting the pedal position into a twist as described in section 2.4.4 of the analysis. Because the robotic platform is assumed to move with two degrees of freedom, the parallel linear velocity and the angular velocity for yaw are used of the twist. This node subscribes to the message from the pedal driver node containing the pedal positions, and publishes the twist to the control node of the robot. A schematic overview of the signal flows of the node is shown in figure 3.16.
As described in section 2.4.4 of the analysis, the pedal positions are first converted to the displacement in percentage with respect to the origin. The equation for the conversion is shown in equation 3.19.

$$P = \frac{100 \cdot (\theta_p - \theta_{min})}{\theta_{max} - \theta_{min}} - 50 \tag{3.19}$$

Where P is the displacement percentage between ±50% for one pedal, θ_p the received pedal position and θ_{max} and θ_{min} the pedal position's extremes in radians.
Then, using either the sum or the difference between both displacement percentages, the linear and angular velocities are calculated respectively. The formula for the linear movement is shown in formula 3.20.

$$v_l = \frac{v_{max} \cdot (P_L + P_R)}{100} \tag{3.20}$$

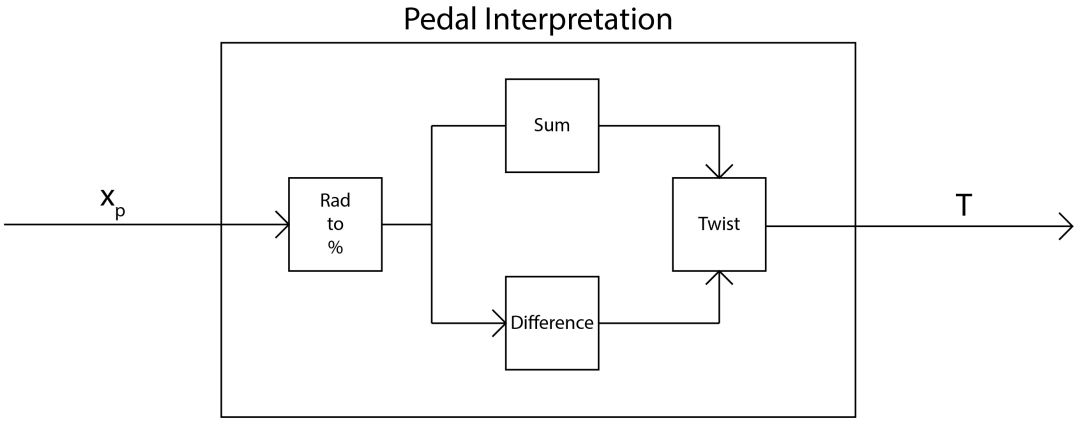


Figure 3.16: Schematic overview of the in- and output of the pedal interpretation node

Where v_l is the the linear velocity, v_{max} the maximum velocity of the robotic platform and P_L & P_R the displacement percentages of the pedals. The formula for the angular velocity is shown in formula 3.21.

$$v_r = \frac{v_{max} \cdot (P_L - P_R)}{100} \tag{3.21}$$

Where v_r is the angular velocity.

3.3 Mechanical

The pedal is designed to only be pushed in one direction by the operator, however, as discussed in section 2.4.4 of the analysis, the pedal is required to be pushed down and pulled up. The possibility to pull the pedal up enables the operator tho move either side of the robotic platform in two directions. To enable upward motion some sort of attachment system on the pedal has to be added. Designing a definitive attachment system will not be done due to time constraints, a temporary solution will be used which provides the same functionality as the definitive attachment is supposed to provide. The operator's feet need to be secured in two places, at the heel and the instep of the foot. The placement of the heel will be equal for all foot sizes, a small lip preventing the foot from sliding off the pedal suffices. For the instep of the foot a different mechanism is required. Because all foot sizes are different, an adjustable strap is implemented which secures the foot to the pedal.

4 Testing & results

After the design & implementation phase, all the different parts of the system are tested to be able to evaluate its performance. All tests that involve the pedals will be performed using the right pedal. The reason being that the left pedal does not contain an accelerometer. Also all tests are conducted with the pedal controller node, and thus the RaMstix, running at a 200Hz frequency. The test phase contains two parts: the first one is about testing the sensor signal processing, PD controller and controlling the robot, and the second part is about testing the feedback implementation.

4.1 Pedal tests

In the first part of the testing phase the sensor data acquisition, processing and implementation is tested.

4.1.1 Pedal angle

First, the accuracy of the pedal angle measurement is determined. The pedal angle is used for determining the pedal's displacement, velocity and acceleration compensation, and therefore the fundament of the whole system. By determining the precision of the pedal angle measurement, other components can be validated as well.

The measured angle of the pedal is compared to the measured angle in section 2.2.1 of the analysis chapter. This test will provide the precision of the $rad/step$ conversion. To determine the precision of the measurement, the pedal is placed in its minimum and maximum angle. The output values are compared against the physically measured angle. The accuracy is presented in percentage based on formula 4.1.

$$Q = \frac{100 \cdot \theta_{exp}}{\theta_{exp} + (\mid \theta_{exp} - \theta_{mes} \mid)} \tag{4.1}$$

Where Q is the precision in percentage, θ_p is the pedal position in radians, θ_{exp} the expected angle and θ_{mes} the measured angle. It is expected that the output angle will at least have a precision 98,74% of the expected angle. This is based on the encoder calibration and the encountered error in the step count. The encoder is calibrated at 3,2309E-5 radians per step and during testing a maximum error of ±150 steps was encountered.

Table 4 shows the results of the test. It shows that the accuracy of both the minimum and maximum pedal angle output are within 0,2% of the expected angle output. This provides a sufficiently good base to continue testing with.

Table 4: Results pedal angle test

Position	Expected (radians)	Measured (radians)	Delta	Accuracy (%)
Min	3,80830843E-1	3,81528974E-1	6,98131E-4	99,82
Max	9,07920277E-1	9,08618409E-1	7,98132E-4	99,91

4.1.2 Acceleration and gravity compensation

The measured acceleration is used for estimating the velocity signal's higher frequencies, to determine the usability of the measured signal the gravity compensation and noise in the signal are examined. The acceleration before and after the compensation are compared to determine the effectiveness of the compensation. The compensated signal will also shown errors in the calibration of the accelerometer. Last, the compensated acceleration signal is examined in frequency domain to evaluate the noise.

For the testing of both the uncompensated and compensated acceleration signal, test data is acquired by saving the output of the pedal angle, uncompensated acceleration and compensated acceleration over multiple pedal angles. The pedal is placed in an angle and then held in position to remove any movement acceleration. Both acceleration signals are plotted against the pedal angle. To examine the noise, the compensated acceleration signal is plotted in frequency domain.

The hypothesis is, that without pedal motion the uncompensated signal outputs the magnitude of gravity working on the accelerometer. Increasing the pedal angle is expected to result in a decrease in output. For the compensated acceleration, it is expected that the acceleration is 0 for every pedal angle. The angle measurement of the encoder is evaluated sufficiently good, therefore the gravity compensation is expected to be reliable as well. During motion, both accelerations are expected to behave similar, excluding the offset in the uncompensated acceleration. Going from one pedal angle to the next should show a spike in both acceleration signals. If the accelerometer is not correctly calibrated, the compensated acceleration signal will either have an offset or not be horizontal.

The pedal angle over time during the test is shown in figure 4.1. The test results for both acceleration signals against the pedal angle are shown in a scatterplot in figure 4.2. The upper orange signal is the compensated signal, the lower blue signal is the acceleration signal. The vertical lines in both acceleration signals indicate the angle at which the pedal was held still. The uncompensated gravity output is negative because of the orientation of the accelerometer. Looking at the uncompensated acceleration without pedal motion, the gravity working on the accelerometer creates a large offset. The gravity decreases while increasing the pedal angle as expected. The compensated acceleration signal is centred slightly above 0, indicating that the gravity compensation is working as expected, but that the accelerometer's offset voltage is not perfectly calibrated. For both signals, there is a large, but consistent deviation whenever the pedal is not moving.

During motion, in figure 4.2 shown in the areas between the vertical lines, the deviation for both signals is approximately equal to the deviation without motion. During the first motion between 0,37 and 0,55rad, both signals have a deviation larger than the constant deviation. However, for the other three moments of motion, the deviation does not exceed the deviation without motion. This means that, if the acceleration is not large enough, it is drowned out by the deviation.

The frequency spectrum of the compensated acceleration signal is shown in figure 4.3. It shows a large 0Hz magnitude, confirming that the signal still has an offset. It is also visible that the signal contains lower frequencies with a relatively high magnitude, indicating that the acceleration signal does contain the motion of the pedal. It does not show an outstanding peak in the higher frequencies, indicating that the noise is present in a broad frequency range.

Based on the observation that the accelerometer output is consistent, but including noise, the output voltage of the RaMstix was tested by connecting the DAC directly to the ADC and subtracting the mean offset of the signal. The output is shown in figure 4.4 and its corresponding frequency spectrum in figure 4.5. The frequency spectrum shows that the noise in the output voltage is over the full range of frequencies, but with a very small magnitude and therefore unlikely to cause the noise in the acceleration signal. Due to time constraints, the cause of the noise has not been found.

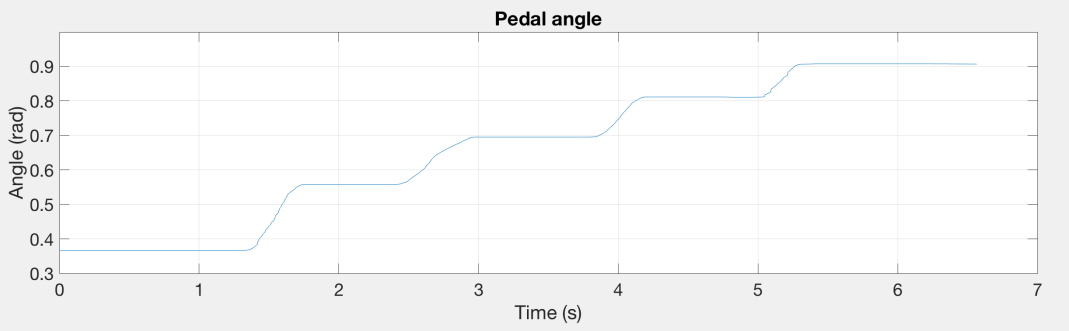


Figure 4.1: Angle of the pedal during the accelerometer test

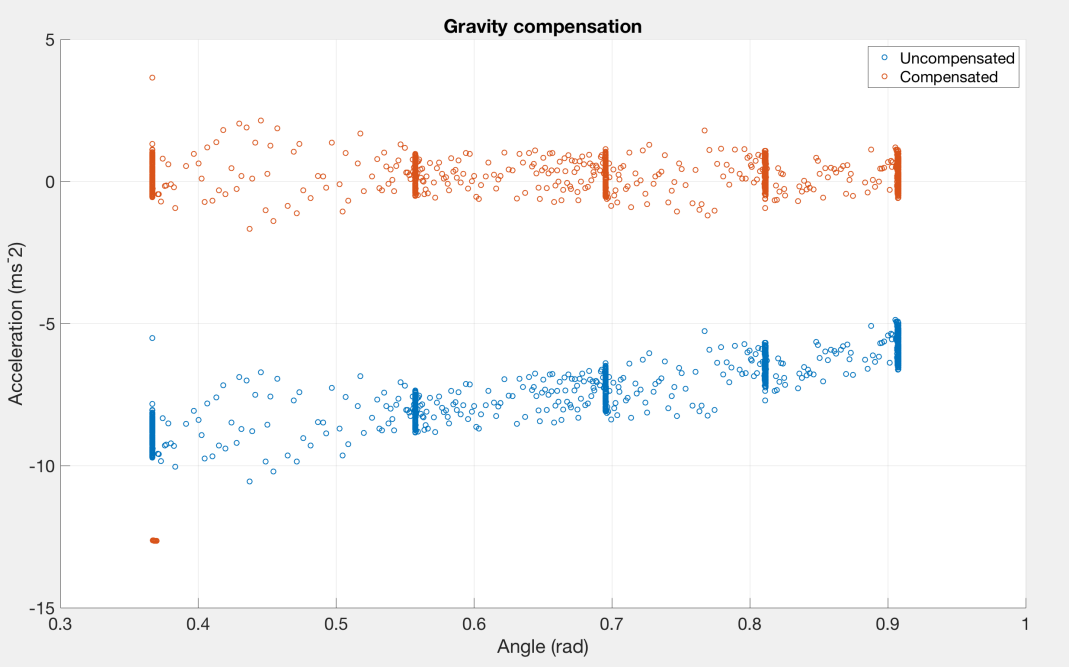


Figure 4.2: The uncompensated and compensated acceleration signal against the pedal angle

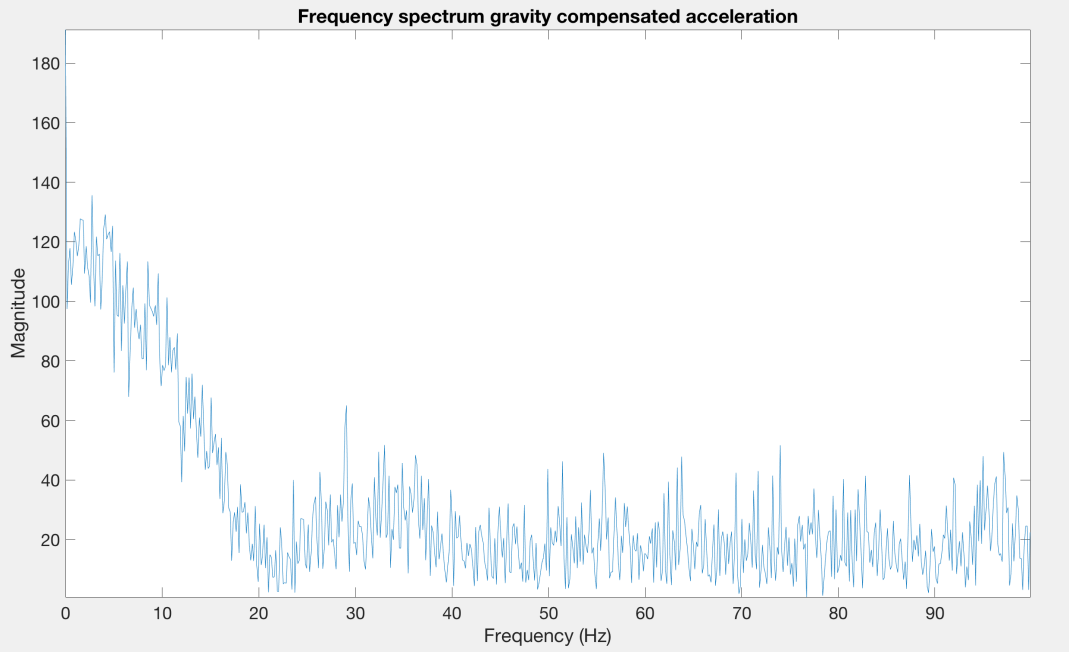


Figure 4.3: Output of the gravity compensated acceleration in frequency domain

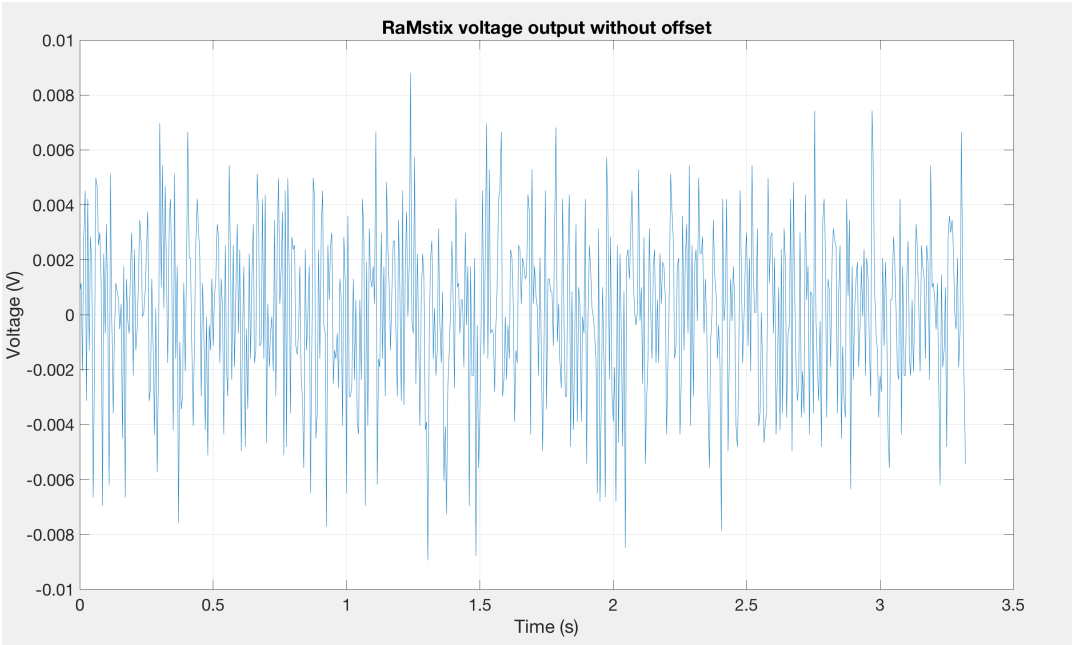


Figure 4.4: Output of the RaMstix DAC directly to the ADC

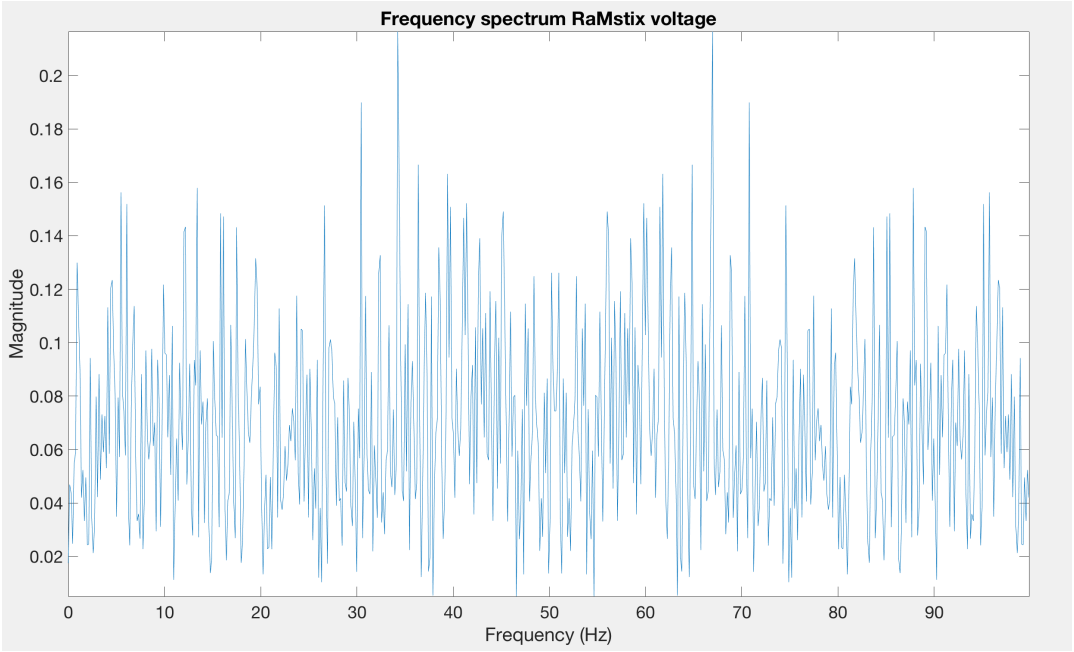


Figure 4.5: Output of the RaMstix DAC directly to the ADC in frequency domain

4.1.3 Sensor Fusion

The goal of this test is to assess the added value of using sensor fusion for the velocity estimation of the pedal. This is determined by the results of the differentiation of the position, integration of the acceleration and low- and high-pass filters. The output values of time, pedal angle, differentiation, integration, LPF, HPF and the combined velocity are recorded during random pedal movement. Then the output signals are compared both in time and frequency domain. The same set of values will be used for this entirety of this section. The output of the pedal angle over time is shown in figure 4.6, and in frequency domain in 4.7. For the frequency spectrum notice that the range of frequencies displayed is only 0 to 10Hz, also the 0Hz offset is vertically cutoff die to the relatively large magnitude.

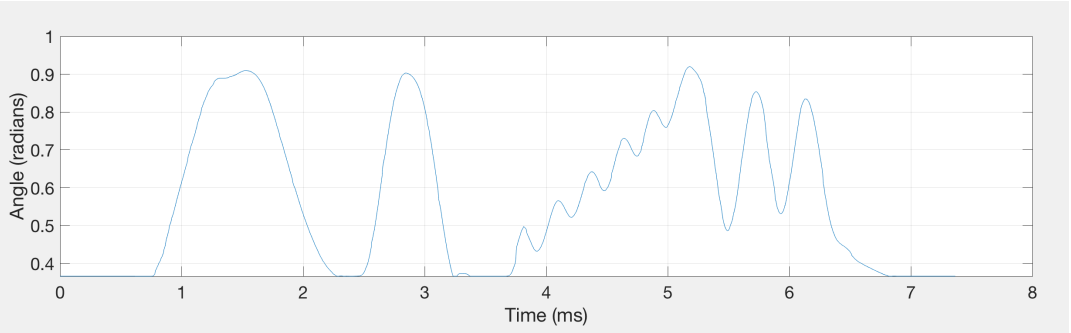


Figure 4.6: Output of the pedal position in radians

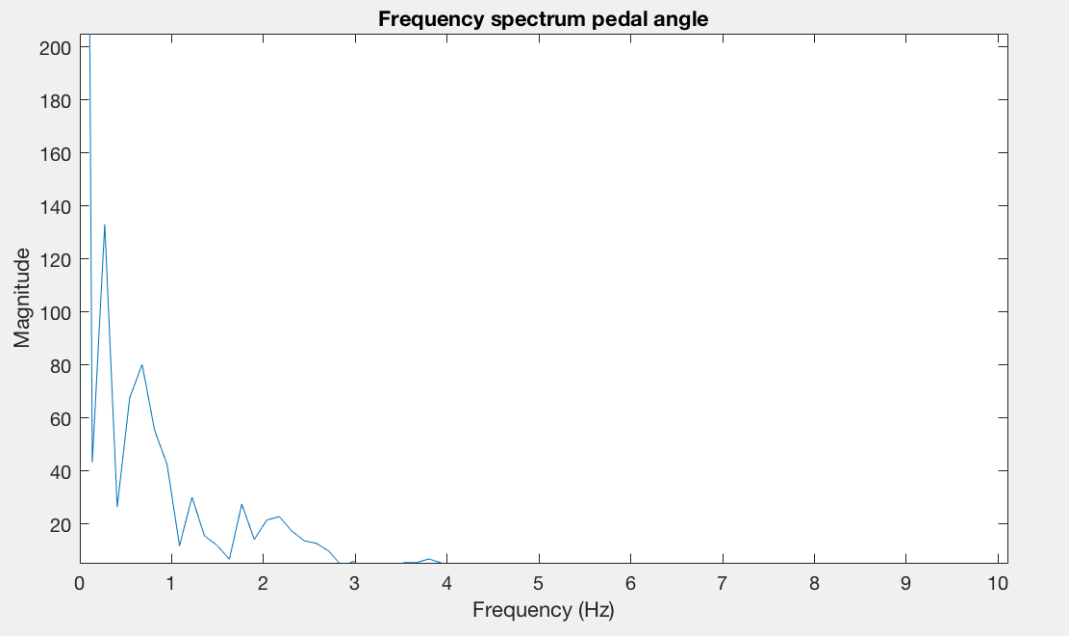


Figure 4.7: Output of the pedal position in frequency domain

4.1.3.1 Differentiation & low-pass filter

Differentiating the pedal position into velocity amplifies the high frequency noise in the signal, therefore the goal of this test is to see whether the differentiation produces a usable signal and if the low-pass filter improves it. Based on the findings in section 2.3.4.1 of the analysis, it is expected that in the peaks of the velocity signal high frequency noise will be present. Which in turn will be filtered out by applying the low-pass filter. The filtered velocity is expected to have a 10ms delay based on the second order filter and the 200Hz sampling frequency.

The output of the differentiation of the pedal position and the low-pass filtered velocity are shown in figure 4.8, a smaller range is displayed for visual purposes. The frequency spectra of both signals are shown in figure 4.9. As expected the unfiltered velocity in time domain shows high frequency noise in the peaks of the signal. This is also visible in the frequency spectrum between 10 and 100Hz as the magnitude is relatively consistent. The noise is significantly lower than the 0 to 10Hz range and therefore most likely does not contain relevant info. Figure 4.9 (2) clearly shows the filter's effect in the 15 to 100Hz range, where the magnitude is significantly reduced. This also shows in the time domain signal as a smoother signal with lower extremes and almost no noise in the peaks, without loosing much definition of the signal. The filter does introduce the expected delay as can be seen in the time domain signal where the filtered signal lags behind.

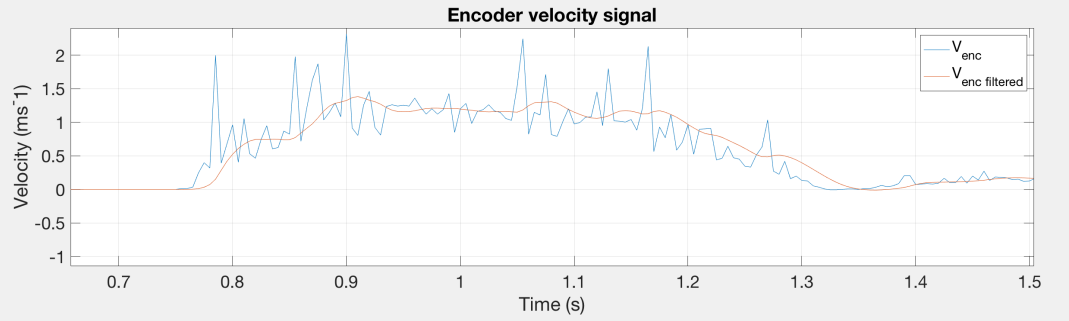


Figure 4.8: Output of the pedal velocity based on the encoder position

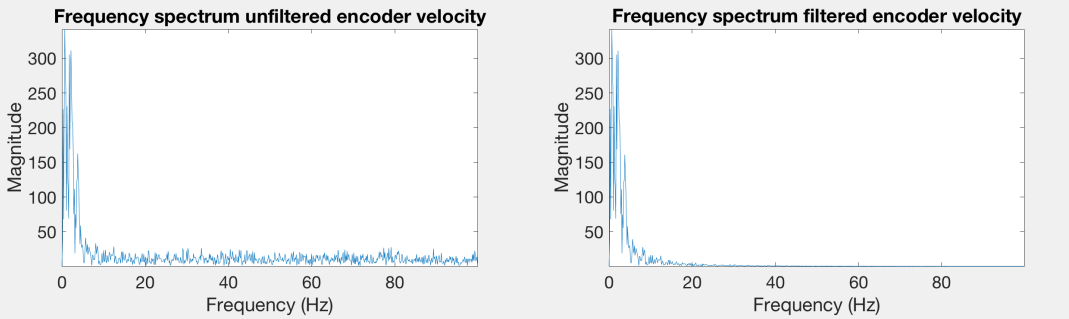


Figure 4.9: Frequency spectrum of the pedal velocity based on the encoder position (LTR)(1) Unfiltered (2) Filtered

4.1.3.2 Integration & high-pass filter

Integrating the pedal acceleration into velocity is likely to cause drift in the signal, making the goal of this test to determine the usability of the velocity signal before and after the high-pass filter. Section 2.3.4.2 of the analysis shows how the integration of the acceleration is likely to drift based on the offset error of the accelerometer. Combined with the outcome of the accelerometer test in section 4.1.2 where it shows that the voltage input of the accelerometer is not constant, the offset error will exist and therefore drift will occur. Also the same high frequency noise, seen in section 4.1.2, is expected. The high-pass filter is expected to recenter the velocity signal around the x-axis by removing the 0Hz offset frequency. Also the same 10ms delay is expected based on the second order filter and the 200Hz sample frequency.

Figure 4.10 shows the unfiltered and filtered integrated acceleration signals and their corresponding frequency spectra are shown in figure 4.11 (1) and (2) respectively. Note that figure 4.10 shows both signals with their own y-axis. Also, in figure 4.11 the y-axis is on a logarithmic scale. The expected drift in the integrated signal is visible in figure 4.10, it corresponds to the first section of the pedal movement where the pedal is stationary. Because of the offset in the gravity compensated acceleration, discovered in section 4.1.2,

the integrated signal starts drifting. This behaviour is also visible in figure 4.11 (1) where the magnitude of 0Hz is relatively high. The effect of the high-pass filter is visible in figure 4.10 as well, the filtered signal is centred around the x-axis proving that the 0Hz offset is removed from the signal. This is also visible in figure 4.11 (2) where the 0Hz frequency is lower than in (1). An unexpected result of filtering the system is, that the higher frequencies are also reduced in magnitude. The complete frequency range in (2) is lower than in (1). Whether the high-pass filter also introduces a delay is difficult to see in figure 4.10. The filtered velocity signal seems to contain relevant info between 0 and approximately 40Hz. However, this could also be due to noise as discovered in section 4.1.2 because the filtered encoder velocity did not seem to contain relevant info above 10Hz. In combination with the 10Hz high-pass filter, it is possible that the accelerometer output adds noise to the velocity estimation instead of a useful signal. This will become more clear by testing the combined velocity.

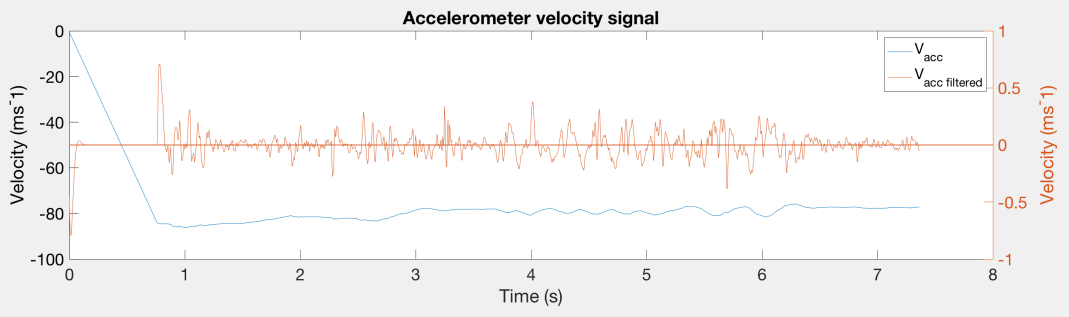


Figure 4.10: Output of the pedal velocity based on the acceleration

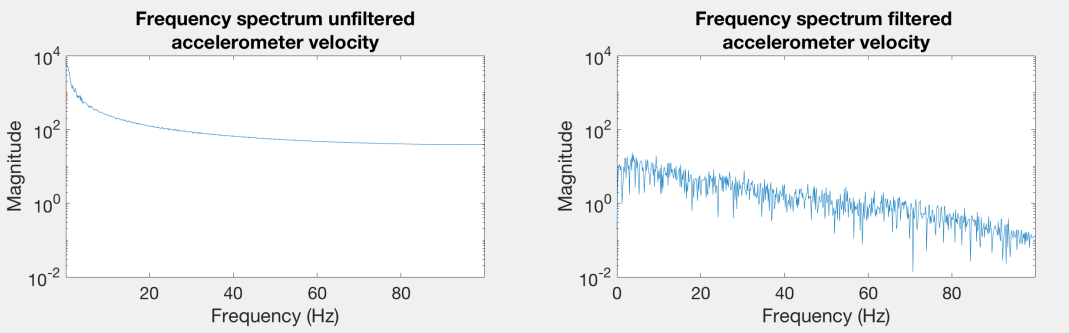


Figure 4.11: Frequency spectrum of the pedal velocity based on the acceleration (LTR)(1) Unfiltered (2) Filtered

4.1.3.3 Combined velocity signals

The goal of this test is to see whether the combined velocity gives better results than either differentiation or integration alone. The output signal is compared to the position signal, the differentiated and integrated velocities and their frequency spectra. Based on the outcomes of section 4.1.3.1 and 4.1.3.2, the combined velocity signal is expected to result in a less usable signal than the filtered differentiated velocity alone. The integrated velocity after filtering does not contain many relevant frequencies which are required to estimate the actual velocity, instead it is likely to add noise to the differentiated velocity, therefore making the combined velocity less usable.

Figure 4.12 and 4.13 show the combined velocity output in time and frequency domain respectively. Figure 4.12 is a part of the full signal and also shows the differentiated and integrated velocity signals. Comparing the combined velocity to the position signal, the combined velocity signal contains more high frequencies in the peaks and also introduces higher frequencies in the transition from peak to peak which are not visible in the position signal. The frequency spectrum in figure 4.13 also shows that above 10Hz there are no frequencies with significant magnitude and these are mostly added by the integrated velocity. Based on the results for the three different velocities, all three signals contain relevant information. However, the differentiated velocity seems to contain less noise than the combined velocity signal. It is expected that the functionality of the controller will benefit from using the differentiated velocity instead of the combined velocity.

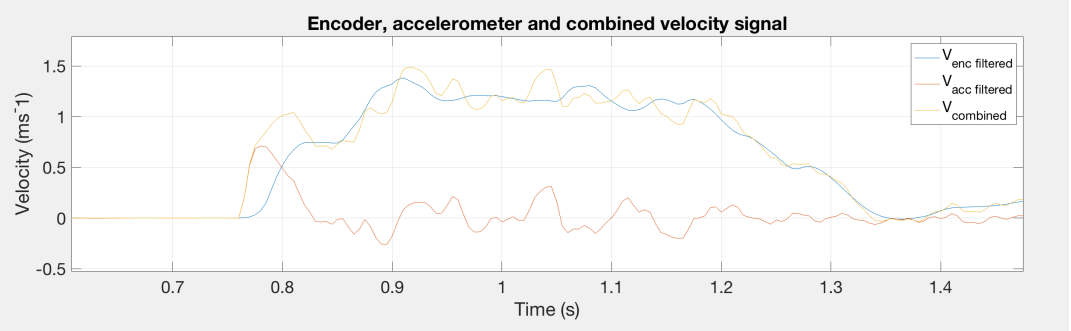


Figure 4.12: Output of the pedal velocity from differentiation, integration and the combined signal

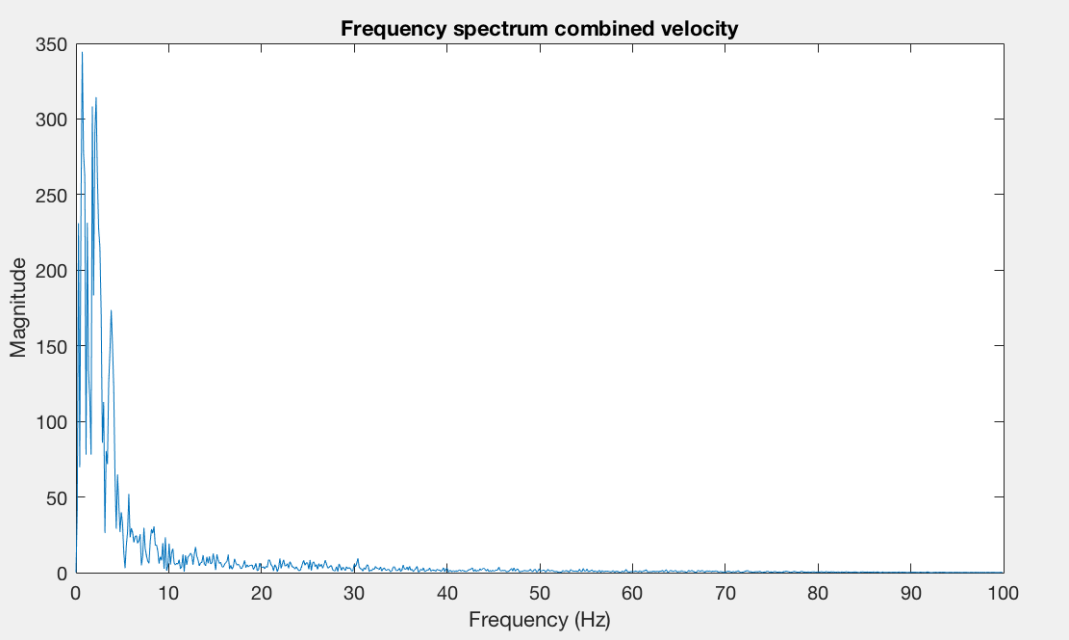


Figure 4.13: Frequency spectrum of the pedal velocity based both the position and acceleration

4.1.4 Controller

The controller is designed such that the spring and damping constants are varied to provide haptic feedback while the damping ratio stays constant. The goals of this test are to determine the stability and consistency of the controller and the performance difference between the differentiated and combined velocity. First, the differentiated velocity is used. For the testing of the stability of the controller, the pedal is maximally displaced and then released while measuring the velocity and position of the pedal. This is done twice in both directions of the pedal. The test is repeated for six different spring constant values to determine the consistency of the controller with varying spring constant. The spring constant values are $K_1 = 6$, $K_2 = 8$ ... $K_6 = 16$, where K_1 is no feedback and K_6 maximum feedback. Then, for the combined velocity the test is repeated. A plot will be generated for each spring constant per velocity.

The original controller takes the square root of the displacement of the pedal, due to stability issues, the linear displacement is used instead. The used transfer function of the controller is shown in equation 4.2.

$$F_{out} = -(K_b + K_i)\theta_p - 2\omega\sqrt{I(K_b + K_i)} + F_{off} \tag{4.2}$$

4.1.4.1 Differentiated velocity

Because the findings in section 4.1.3.3 showed that the differentiated velocity results in a more usable signal than the combined velocity, the controller test is performed with the differentiated velocity. The damper constant is based on the spring constant in such a way that the damping ratio is constant, which should result in a similar pedal motion for any spring constant. Figure 4.14 shows the four different damping ratio states, where ζ is the damping ratio.

- $\zeta > 1$ Overdamped
- $\zeta = 1$ Critically damped
- $\zeta < 1$ Underdamped
- $\zeta = 0$ Undamped

$\zeta = 1$ is the wanted result at which the controller is critically damped. Because the controller does not contain an integrator it is possible that a steady state error appears when the pedal returns to its origin position.

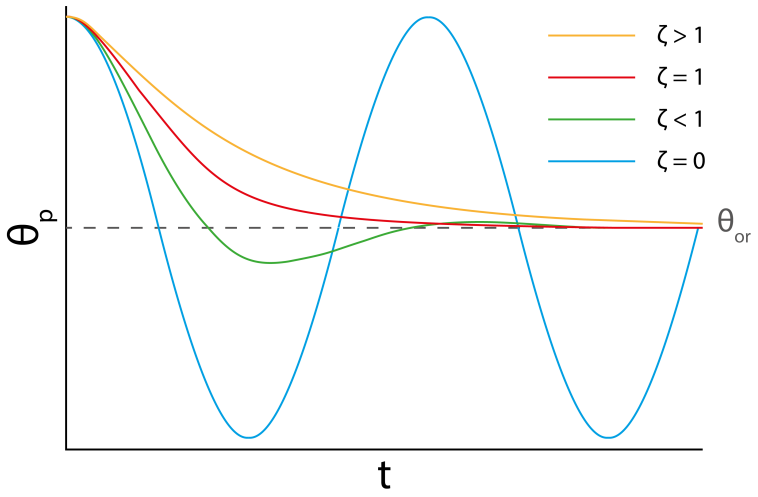


Figure 4.14: Visualisation of four damping ratio situations

Figure 4.15 and 4.16 show the results of the six different spring constants. In each graph the pedal angle θ_p deviates from its origin θ_{or} four times. After the deviation the pedal is released, at which the controller returns the pedal to its origin position. For K_1 to K_3 the controller shows a stable controller with a slight overshoot, however, for K_4 to K_6 the lack of higher frequencies in the velocity signal is visible. All three figures in figure 4.16 show the pedal oscillating when returning to its origin position. The frequency at which the pedal needs to be damped is higher than the cutoff frequency of the low-pass filter and is therefore unable to damp the pedal in time creating the oscillations.

All six graphs show a clear steady state error, the error is the margin between the horizontal parts of the pedal position (blue line) and the origin position (magenta line). As expected, the lack of an integrator in the controller does not correct the steady state error over time. Table 5 shows the steady state errors for all four motions with all six spring constants. The average steady state error decreases when the spring constant is increased, apart from K_5 to K_6. This can be explained with the increased spring force, for a constant displacement.

In figure 4.16 (3) at T ≈ 6,8s the pedal position oscillates in a unexpected way. Before the pedal passes the origin position it switches direction like an oscillation, which happens three times. A possible explanation is that the combination of the low-pass filter and the delay in the system resulted in an unstable controller.

Table 5: Steady state errors for K_1 to K_6 with differentiated velocity

Spring	**Deviation (rad)**				
Constant	**1**	**2**	**3**	**4**	**Average**
K_1	2,18E-02	3,97E-02	5,09E-02	9,53E-02	5,19E-02
K_2	4,08E-02	4,14E-02	3,42E-02	3,68E-02	3,83E-02
K_3	3,50E-02	3,35E-02	1,86E-02	3,09E-02	2,95E-02
K_4	1,37E-02	2,30E-02	2,52E-02	7,32E-03	1,73E-02
K_5	8,92E-03	8,52E-03	7,72E-03	9,42E-03	8,65E-03
K_6	1,33E-02	8,32E-03	7,92E-03	1,79E-02	1,19E-02
				Total	2,63E-02

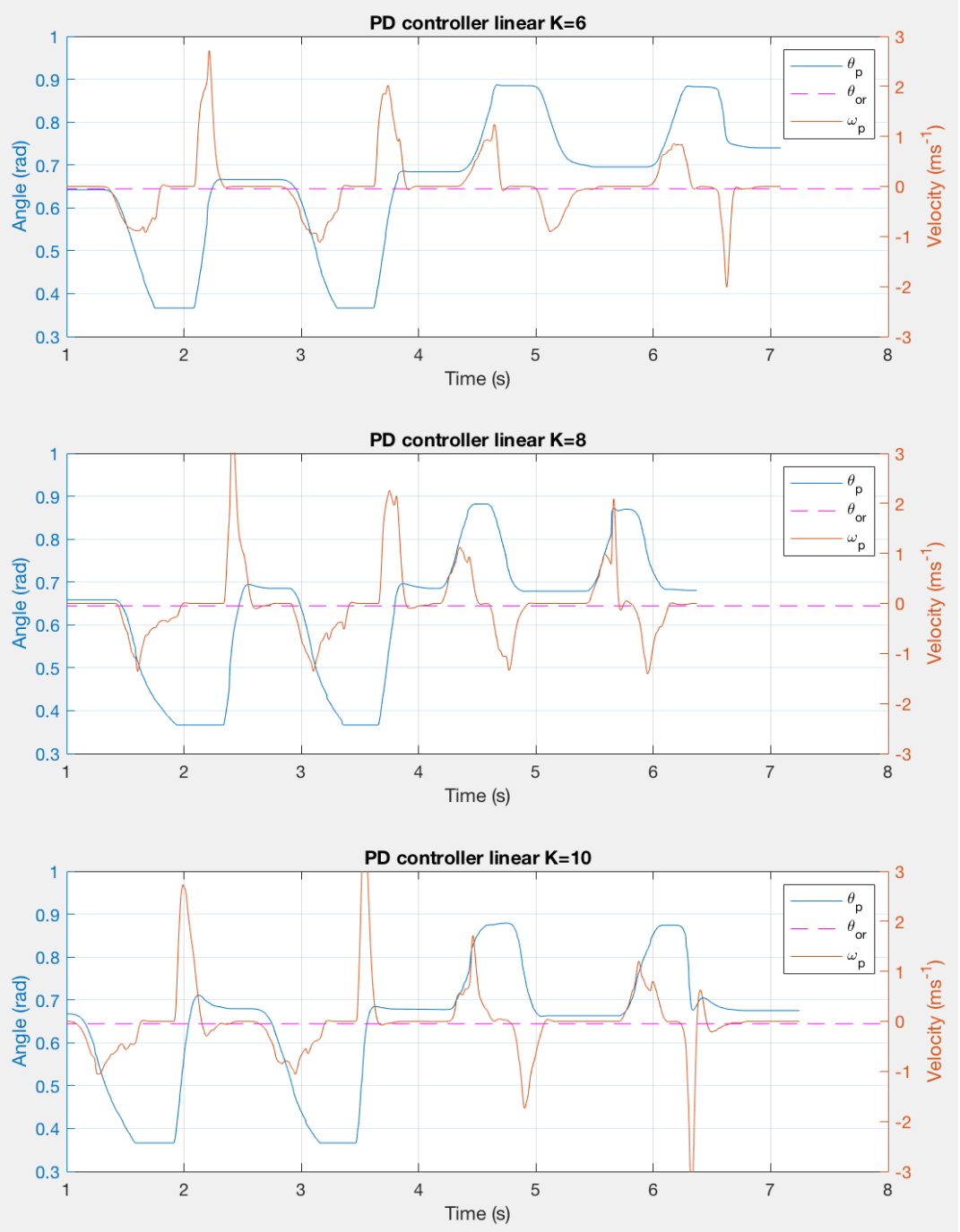


Figure 4.15: Output of the pedal position and differentiated velocity with linear displacement and (TTB:)(1) K = 6 (2) K = 8 (3) K = 10

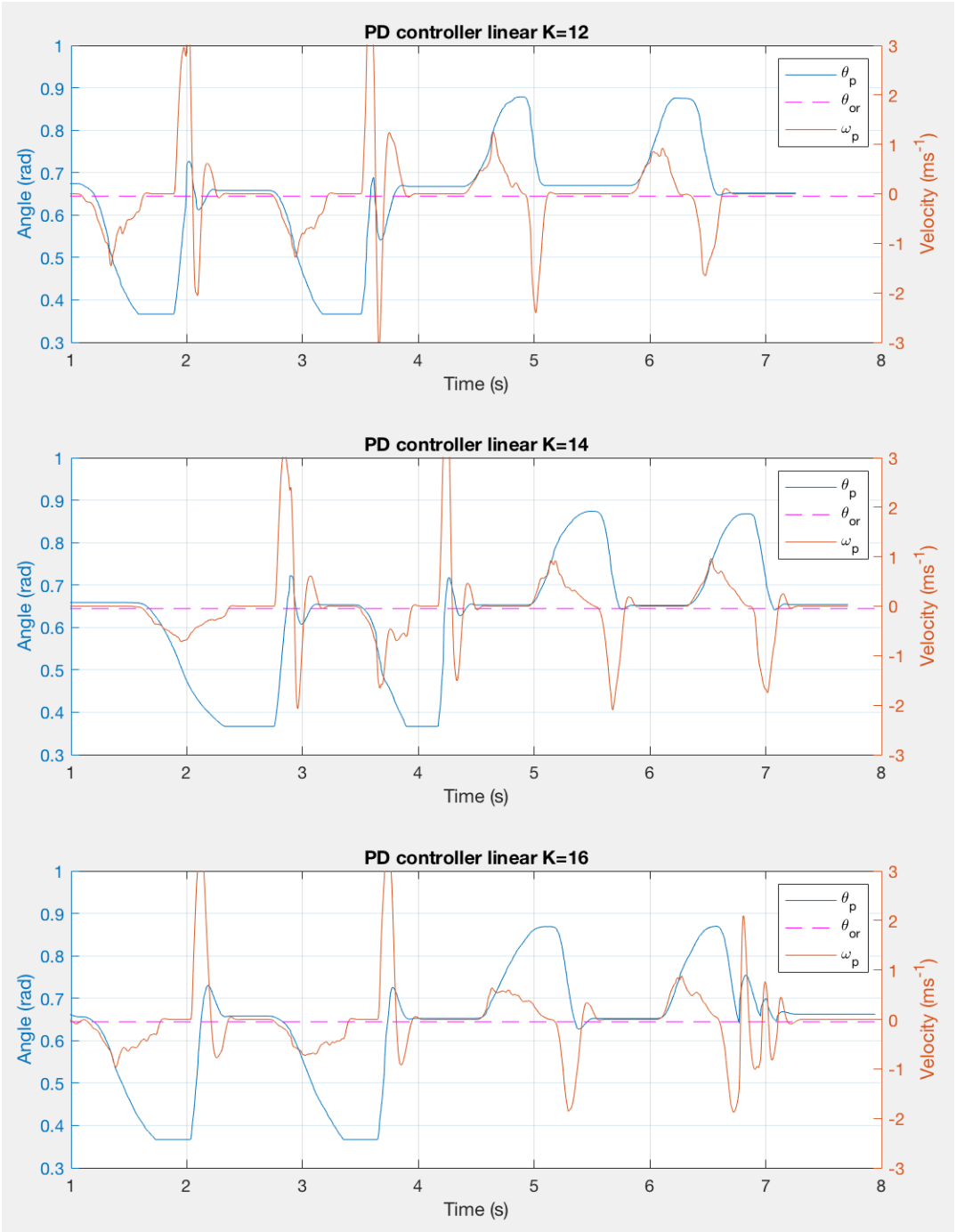


Figure 4.16: Output of the pedal position and differentiated velocity with linear displacement and (TTB:)(1) K = 12 (2) K = 14 (3) K = 16

4.1.4.2 Combined velocity

Second, the controller is tested based on the combined velocity signal. Based on section 4.1.3 it is expected that this velocity signal will be adequate to be used for the damping of the pedal, however the noise could give the pedal a faltering feeling. And again, because the controller does not contain an integrator it is possible that a steady state error appears when the pedal returns to its origin position.

Figure 4.18 and 4.19 show the results of the test for the six different spring constants. Figure 4.18 (1) does not show any oscillation, however, the higher the spring constant becomes, the more underdamped the controller becomes. The increasingly underdamped state causes the controller to overshoot the origin position more. The difference can be seen between (1) at T ≈ 3,7, (2) at T ≈ 3,5 and (3) at T ≈ 3,5. However, the underdamped state does not create a full oscillation. The decrease in damping ratio could be caused by the system becoming more active due to discretisation. When discretising a spring, the stored force is always lower than the restored force. Figure 4.17 shows the discretisation of the displacement of a spring, the dashed line indicates the moment the stored force is released. When storing energy in the spring, the discretised signal is always equal or lower then the continuous signal. During the release of the force the discretised signal is always equal or higher then the continuous signal. Using Hooke's law this can written down as formula 4.4.

$$\sum Kx(t) < \sum Kx(k) \tag{4.3}$$

$$\sum F(t) < \sum F(k) \tag{4.4}$$

Where $F(t)$ is the continuous and $F(k)$ the discrete force. When increasing the spring constant, the discretisation error increases as well, resulting in a more active, and thus less damped system. The difference in stored and restored force causes the controller to overshoot the origin position.

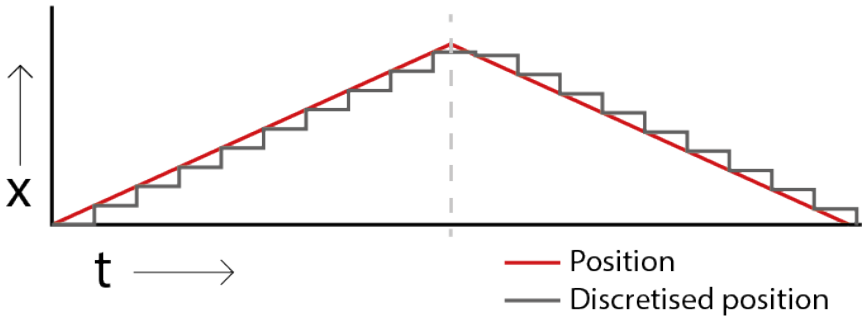


Figure 4.17: Discretising a spring generates energy because the stored force is always lower then the restored force

Table 6 shows the steady state error for all four motions and all six spring constants. The average steady state error decreases when increasing the spring constant which can be explained by the increased spring force for a constant displacement.

Table 6: Steady state errors for K_1 to K_6 with combined velocity

Spring	**Deviation (rad)**				
Constant	**1**	**2**	**3**	**4**	**Average**
K_1	2,42E-02	2,21E-02	3,66E-02	5,23E-02	3,63E-02
K_2	3,45E-02	3,14E-02	2,20E-02	2,02E-02	2,70E-02
K_3	2,11E-02	3,16E-02	5,52E-03	1,09E-02	1,73E-02
K_4	3,05E-02	1,23E-02	6,72E-03	1,72E-03	1,28E-02
K_5	2,56E-02	2,58E-02	8,24E-04	1,61E-02	1,71E-02
K_6	2,01E-02	2,00E-02	2,32E-03	2,32E-03	1,12E-02
				Total	2,03E-02

At T ≈ 6,3 in figure 4.19 (2) a large spike in both the position and velocity is present. This was caused by a short freeze in the program, overshooting its origin and then correcting the position again when the program continued.

Based on the difference between the differentiated and combined velocities, it can be said that the pedal's velocity does contain frequencies above 10Hz. The oscillations on pedal position with the differentiated velocity shows that the pedal motion is not damped fast enough because of the lacking frequency range. The added integrated velocity does improve the controllers performance in this case. However, the differentiated velocity should be tested with a larger frequency range to determine the performance difference between the differentiated and combined velocity. This has not been done due to time constraints. The improved controller performance of the combined velocity does contradict the claim in section 4.1.2 that the acceleration signal's noise drowns out the relevant signal. The combined velocity controller is capable of controlling the position of the pedal with respect to an origin position and stay relatively stable for a range of spring constants.

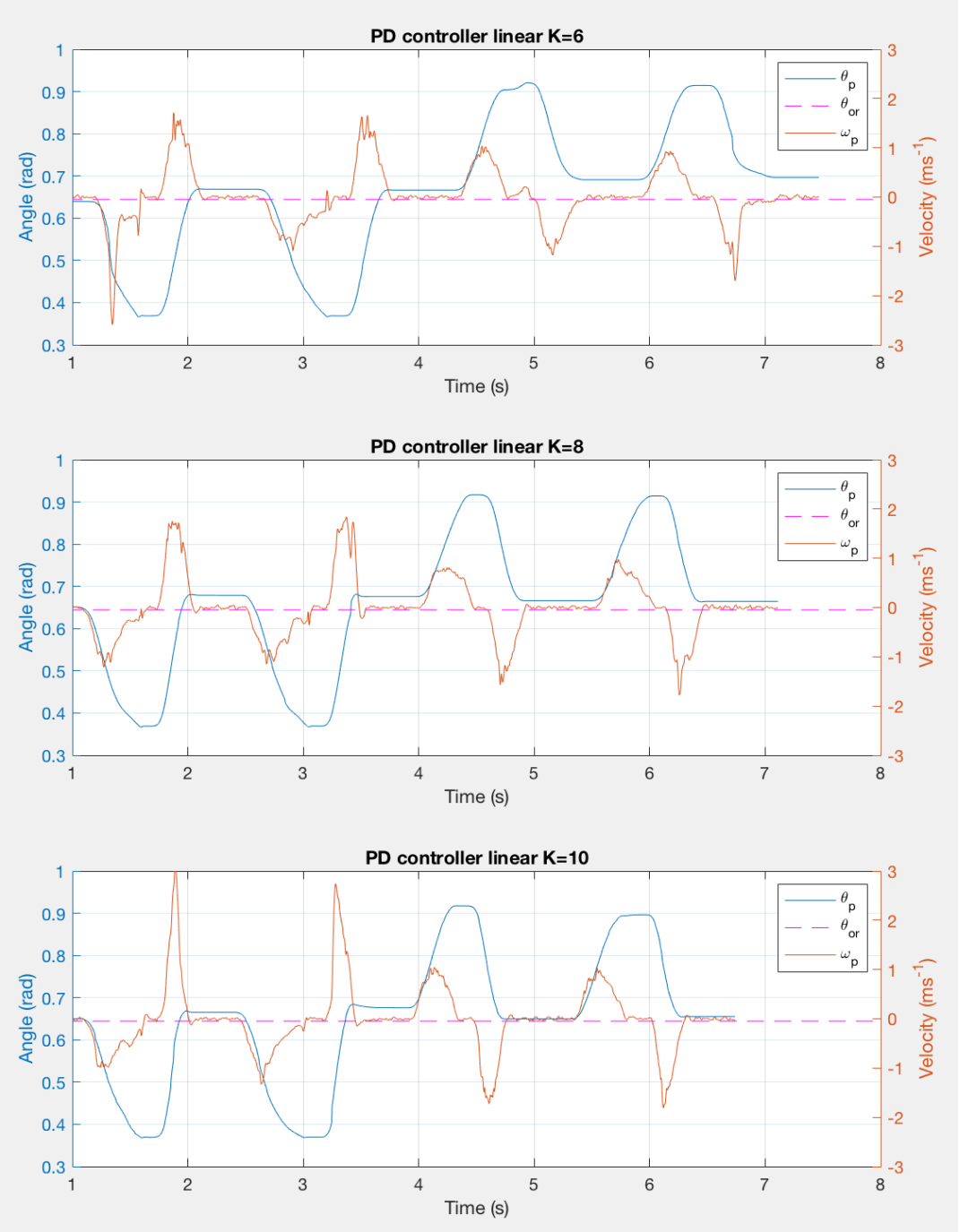


Figure 4.18: Output of the pedal position and combined velocity with linear displacement and (TTB:)(1) K = 6 (2) K = 8 (3) K = 10

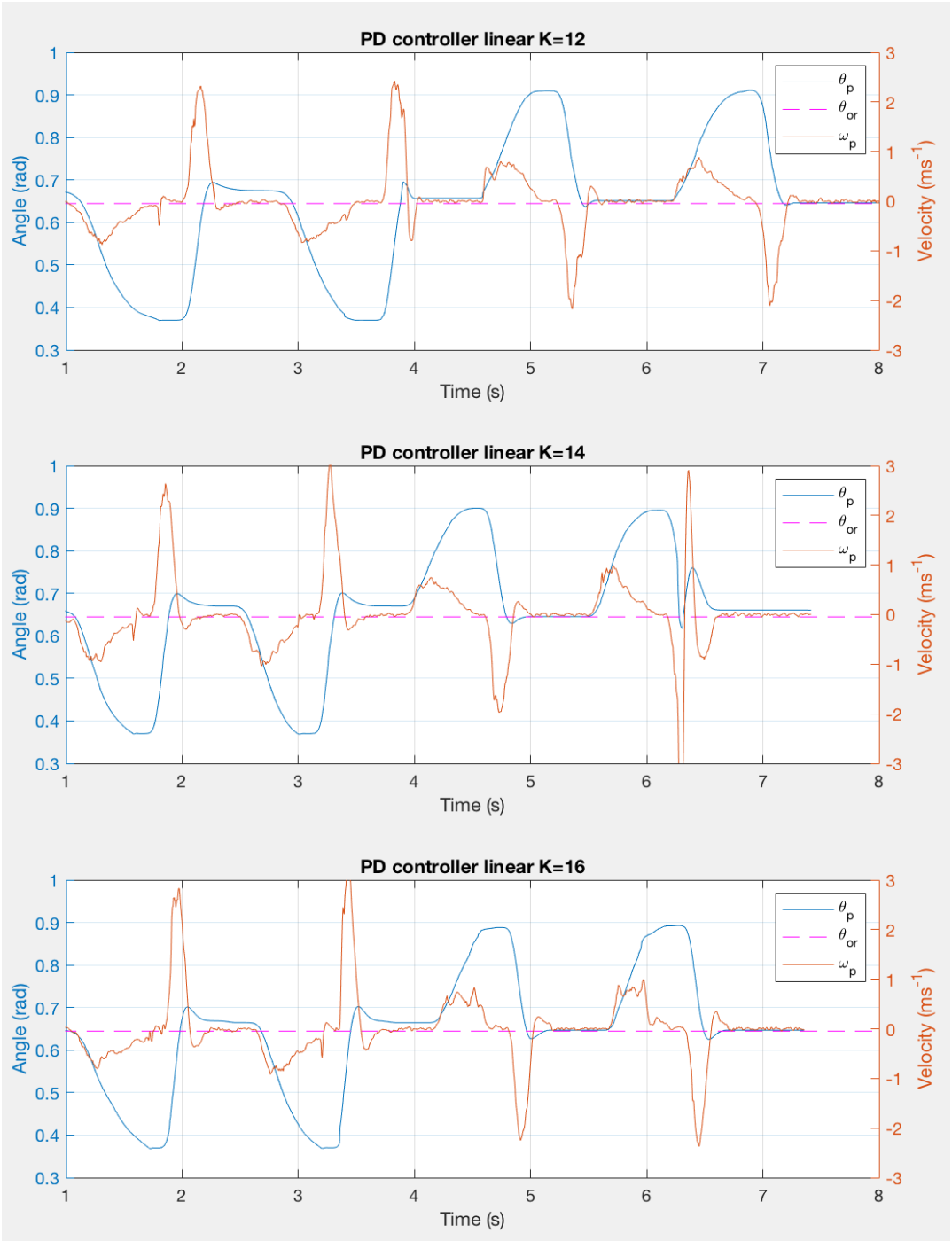


Figure 4.19: Output of the pedal position and combined velocity with linear displacement and (TTB:)(1) K = 12 (2) K = 14 (3) K = 16

4.1.5 Controlling the robot

The goal of this test is to determine whether the pedals are capable of controlling the robot in the range of motion that was stated in section 2.4.4 of the analysis. This is tested by measuring the output vector that is calculated based on the pedal positions (both pedals) and putting the pedals in their four unique extreme combinations. The software is set at a maximum translational velocity of 0,5ms^{-1} and rotational velocity of 2,0$rads^{-1}$. The translational and rotational velocity will be plotted against time and on the secondary y-axis the pedal positions will be plotted. The expected outcomes are shown in table 7, the pedal positions correspond to table 3 of the analysis. In the plot the switching from one pedal position to the next should result in a linear alteration in the output vector.

Table 7: Pedal displacement with respect to the origin in percentage

Position	**Pedal position**		**Expected outcome**
#	Left (%)	Right (%)	<linear, angular>
1	50	50	<0,5; 0>
2	-50	-50	<-0,5; 0>
3	50	-50	<0; 2,0>
4	-50	50	<0; -2,0>

The linear and angular output velocities are shown in figure 4.20. On the left axis (blue) either the linear or angular velocity is shown and on the right axis (orange) the corresponding pedal positions are shown. Both the linear and angular velocity output are as expected, all entries of table 7 are shown in the plots as yellow areas with their corresponding position number.

4.1.6 Evaluation

Based on the test in section 4.1.1 to 4.1.5, the controller and the control of the robot are evaluated. The controller is able to keep a stable state using the sensor values and the variable spring and damper constants. Even though the integrated velocity contains noise generated by accelerometer, the controller's performance improved using the combined velocity. This shows that with a 10Hz cutoff frequency for both filters, using sensor fusion provides a more accurate velocity estimation than the differentiated velocity alone. Without haptic feedback, the controller is able to stay close to critically damped using the measured position and estimated velocity. Increasing the damper constant decreased the damping ratio, but not significant enough to cause an oscillation. The controller is able to adjust the damping constant based on the spring constant, maintaining the stable state. The output force generated by the controller returns the pedal to its origin position, including a clear steady state error due to the lack of an integrator. Using the linear displacement instead of taking the square root resulted in a too low output force to remove the steady state error. Apart from the unexpected behaviour caused by discretisation and software delays, the controller satisfies the needs for this assignment.

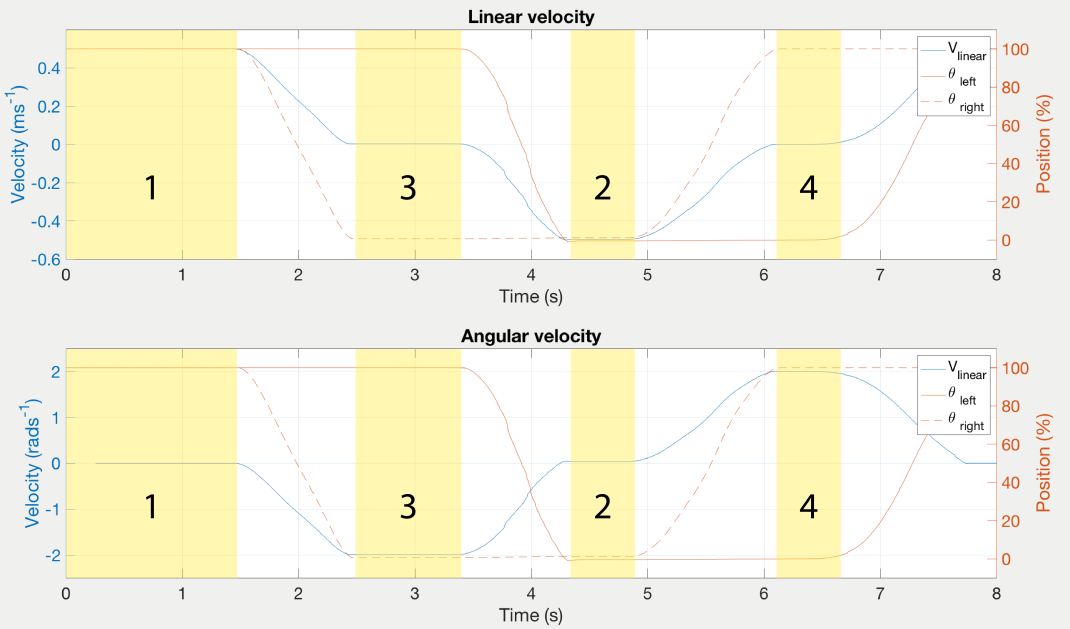


Figure 4.20: Velocity output over the movement range of the pedals with the numbered yellow areas indicating extremes based on table 7 (TTB)(1) Linear (2) Angular

The algorithm for controlling the robot works as expected. The implementation of a two-way motion in the pedal allows the operator to control the rotation and velocity of the robot. Using this interaction design, the robot is capable of rotating around its vertical axle. A limitation of the control design is that the robot cannot rotate while moving at maximum velocity. The implementation is tested on a virtual robot, however, the robotic platform for which the LUC is used, uses the same control interface.

4.2 Simulations

In this part of the test phase the implementation of haptic feedback is tested. The haptic feedback implementation is tested in terms of force distribution, force-distance relation and force-velocity relation. For these tests the virtual environment Gazebo is used in combination with Turtlebot3's Burger robot. The Burger robot is a differentially driven platform containing a 360° laser scanner on top providing the obstacle detection. The virtual environment allows for the placement of cubes which are used as obstacles.

4.2.1 Force distribution

In this test the functionality of the distribution of the increased spring force over the pedals is determined. The goal of the test is to see if the measured force distribution corresponds to the implementation discussed in section 3.2.4 of the analysis. The test setup is shown in figure 4.21, where x is the distance to the obstacle. During the test distance x is constant and the robot is rotated counterclockwise on the spot for one rotation. The angle to the obstacle is based on figure 2.27 in the analysis. All other variables are made constant except for the force distribution during this test. The expected outcome is shown in figure 4.22, which is based on the before mentioned implementation. The time is based on the speed of rotation, but it does not alter the outcome of the test and is therefore irrelevant.

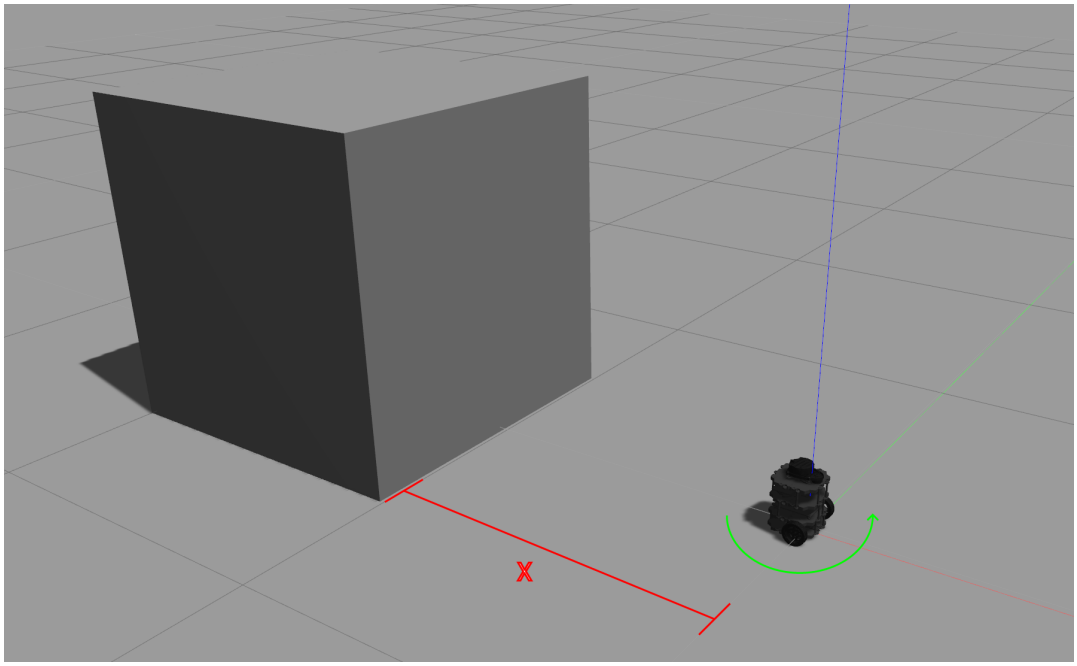


Figure 4.21: Test setup for measuring the force distribution

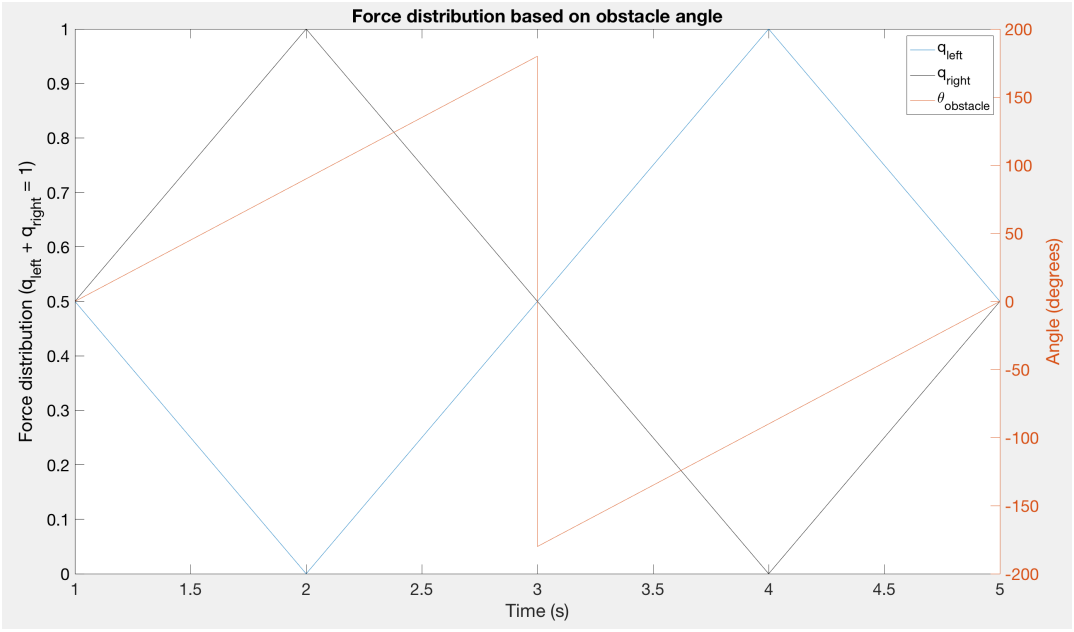


Figure 4.22: Plot based on the expected output values for the force distribution test

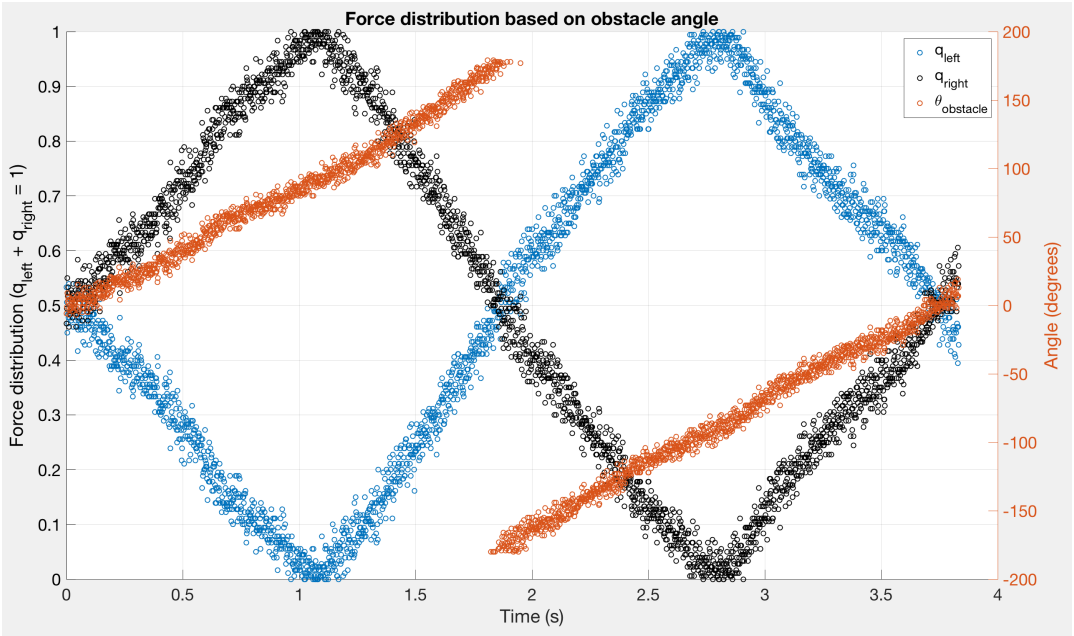


Figure 4.23: Force distribution of the pedals based on the angle to the obstacle

The results of the test are shown in figure 4.23, where on the left y-axis the force distribution is shown and on the right y-axis the angle to the obstacle. There is a lot of high frequency noise in all three signals which is caused by the simulated laser scanner, but does not significantly alter the outcome. Comparing figure 4.23 to 4.22 shows that the outcome is exactly as expected, only including the high frequency noise.

4.2.2 Force-distance relation

The shape of the correlation curve between the distance to an obstacle and the increase in the spring constant (section 2.4.2 of the analysis) is tested to see whether it behaves as expected. This is tested by placing an obstacle 4 meters away from the robot, as shown in figure 4.24, and moving the robot in a straight line with a constant velocity towards the obstacle. Both the force distribution and robot velocity dependency are set to 1, making the distance to the obstacle the only variable. The increase of the spring constant is plotted against the distance to the obstacle. If working correctly the resulting plot should look like figure 4.25, where the magenta lines indicate the limits of the laser scanner of the robot (0,12 - 3,5m).

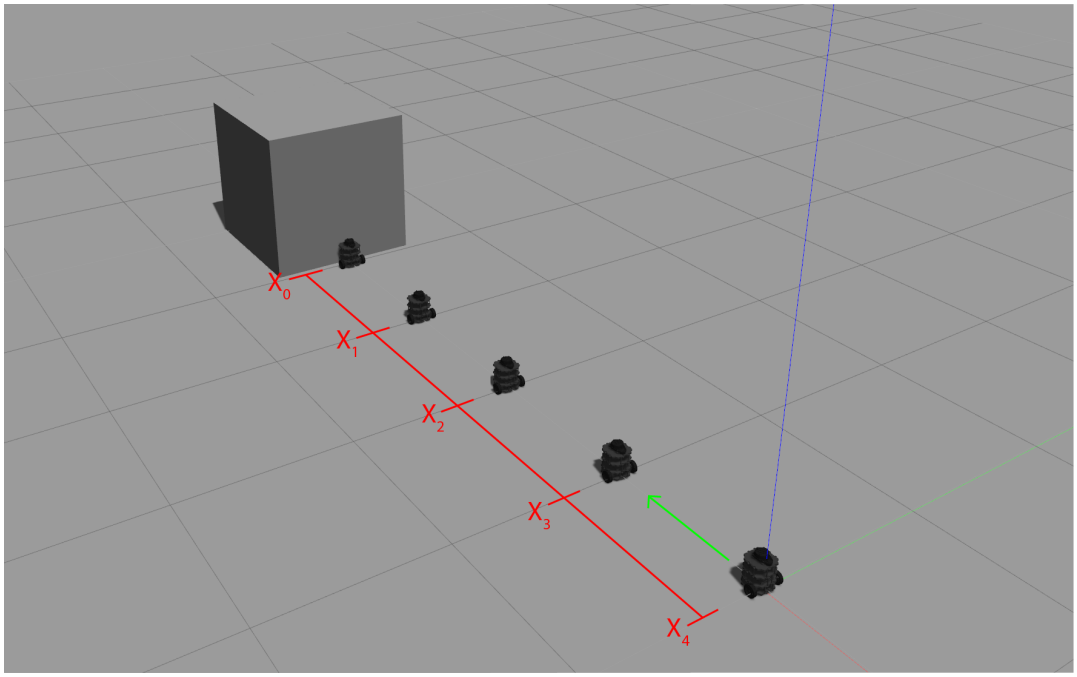


Figure 4.24: Test setup for measuring the force-distance relation

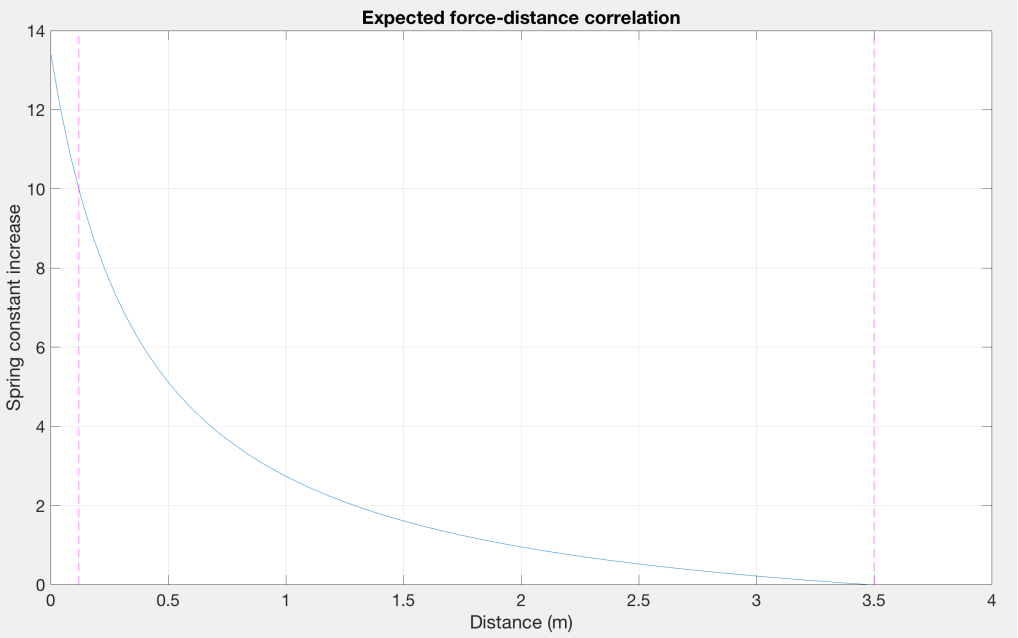


Figure 4.25: Expected correlation curve between the distance to an obstacle and the increase in spring constant

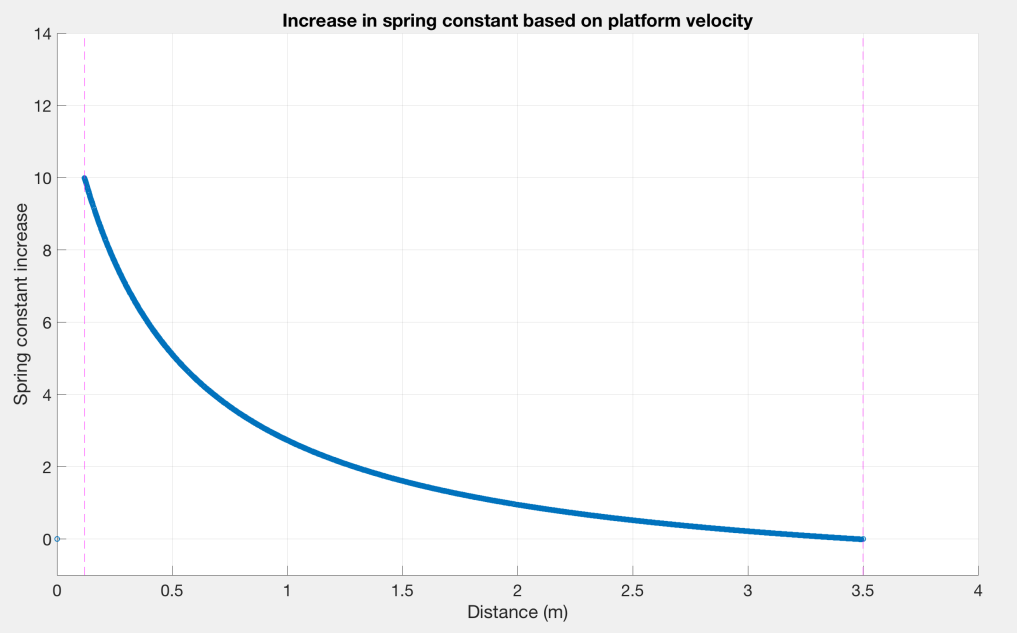


Figure 4.26: Measured correlation curve between the distance to an obstacle and the increase in spring constant

Figure 4.26 shows the measured increase in spring constant based on the distance to an obstacle. Compared to figure 4.25 there are two differences, the first being the lack of continuation beyond the laser scanner's range and a measurement at 0m distance. When no obstacle is detected, the laser scanner outputs a distance of 0, which causes the 0m measurement. Another difference is, that when the obstacle is closer than 0,12m (the minimal detection range) the laser scanner outputs a distance of ±0,6m. This means that when the robot is about to hit an obstacle, the haptic feedback approximately halves. Apart from these anomalies, the outcome is exactly as expected.

4.2.3 Force-velocity relation

The influence of the robot's velocity on the increase in the spring constant is determined using the same test setup as in section 4.2.2. The controller uses the received velocity of the robot, however, for this test the velocity used for calculating the spring increase is manually entered, this keeps the velocity constant. The same calculation is done twice more at the same time for a total of three different velocities, and thus three different spring increase calculations. This is done so that the test setup for each velocity is identical and independent from the actual robot's velocity. If working correctly the robot's velocity scales the spring constant increase between 0,5 and 1 where 0,5 is during standstill and 1 at maximum velocity. The three tested velocities are $V_1 = 0ms^{-1}$, $V_2 = 0{,}25ms^{-1}$ and $V_3 = 0{,}5ms^{-1}$. The expected outcomes are shown in figure 4.27 where the three velocities and their respective curves are indicated and the vertical magenta lines are the range limits of the laser scanner of the robot.

Figure 4.28 shows the measured increase in spring constant based on the distance to an obstacle for the three predetermined velocities. Again, the same anomalies as in section 4.2.2 can be seen. Also the range error when passing the laser scanner's minimal range is present. Apart from these anomalies the measured outcomes are exactly as expected.

4.2.4 Evaluation

The goal was to implement haptic feedback in the pedals, based on the distance and location of an obstacle and the velocity of the robot. Based on the three test in section 4.2.1 to 4.2.3, this implementation is evaluated. The obstacle distance based alteration uses the non-linear stiffness implementation and increases the spring constant. The expected increased in spring constant was as expected, within the range limitations of the obstacle detection. The obstacle location and robot velocity based alteration are both scalers of the increased spring constant. Both variables altered the spring constant increase as expected. The controller, including the variable spring constant, works as intended. The haptic feedback is calculated using a standardised ROS message, making future communication with a sensor array possible. A limitation of the system is that maximum haptic feedback can only be present if the robot is moving sideways at maximum velocity, which is not possible with the (assumed) differentially driven robot. Due to the lack of user tests, nothing can be said about the effect of implementing the haptic feedback.

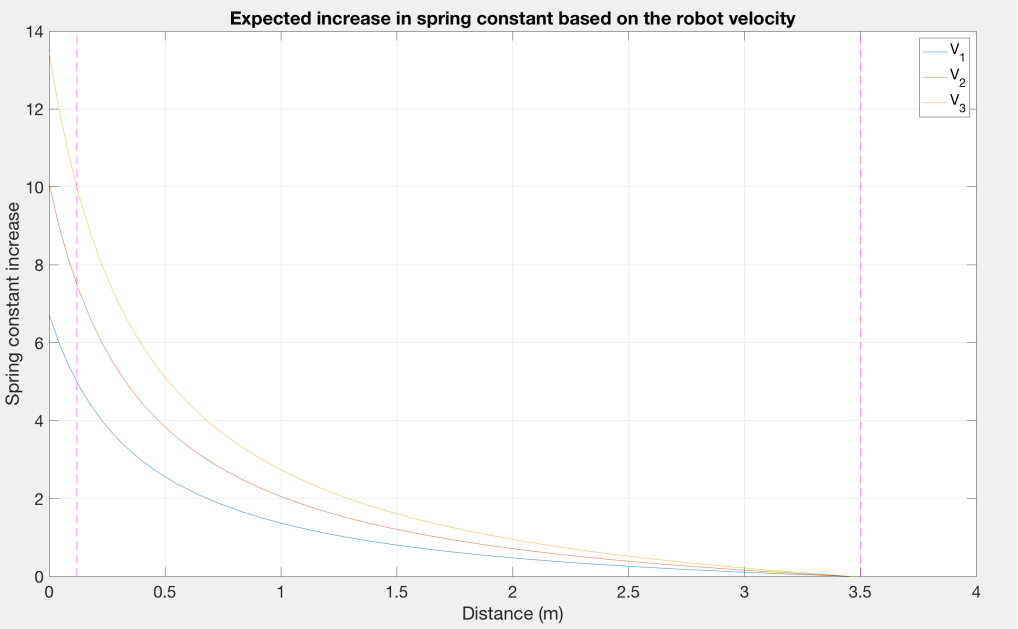


Figure 4.27: Expected correlation curve between the distance to an obstacle and the increase in spring constant altered by the platform velocity where $V_1 = 0 ms^{-1}$, $V_2 = 0{,}25 ms^{-1}$ and $V_3 = 0{,}5 ms^{-1}$

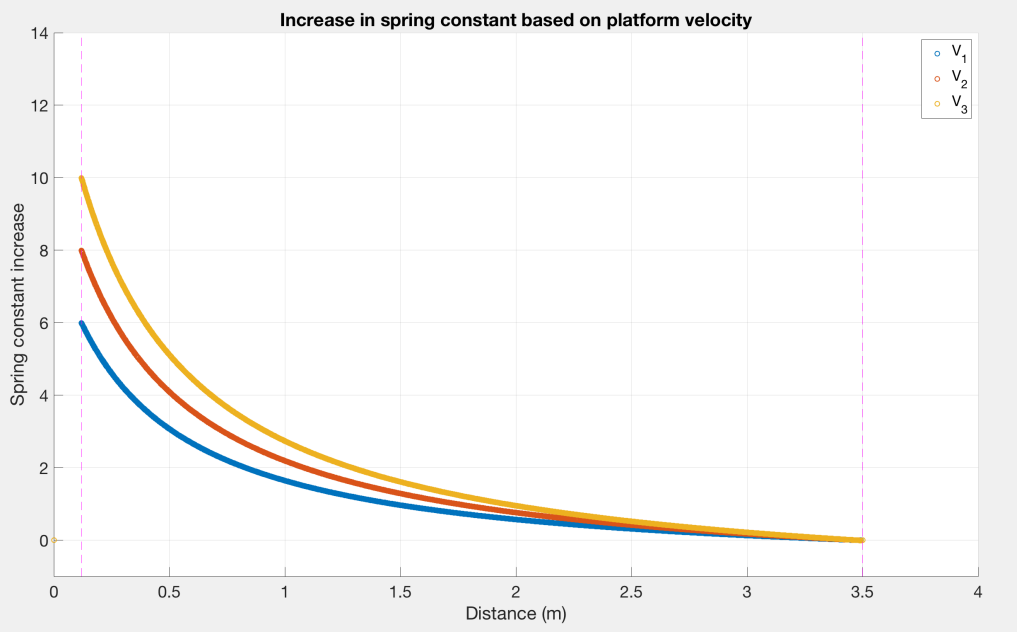


Figure 4.28: Measured correlation curve between the distance to an obstacle and the increase in spring constant altered by the platform velocity where $V_1 = 0 ms^{-1}$, $V_2 = 0{,}25 ms^{-1}$ and $V_3 = 0{,}5 ms^{-1}$

5 Conclusion

The goal of this assignment was to implement obstacle based haptic feedback in the pedals of the LUC to assist the operator with controlling the robotic platform. Besides the main goal, also a physical and digital interface for the pedals and an interaction design for controlling the robotic platform were required.

A comparison between multiple haptic feedback implementations showed that non-linear stiffness feedback is the most suitable implementation. The physical interface for the pedals is made using a RaMstix, for which a digital is interface is implemented as well. Using ZeroMQ a connection is made between the digital pedal interface and ROS. In ROS a modular software architecture is set up for the pedal controller, control of the robot and the implementation of haptic feedback. Using Gazebo for a virtual environment, a virtual robot is used for testing the functionality of the complete system.

The measured position and estimated velocity, used for the controller, were sufficient for a functional controller. The estimated velocity can be improved by altering the matched cutoff frequency of both filters and performing tests accordingly. Without haptic feedback the controller is capable of maintaining a nearly critically damped state, with the exception of software delays.

The controller uses a variable spring and damper constant to maintain a stable state while providing haptic feedback to the operator. Testing showed that increasing the spring constant decreased the damping ratio, however, not enough to make the controller unstable. This decrease is most likely caused by to the effect of discretising a real-time system.

The magnitude of the haptic feedback is dependent on the distance and location of an obstacle, and the velocity of the robot. The distance to an obstacle determines the non-linear increase in output force, while the obstacle's location and robot's velocity scale the increase accordingly. All three implementations worked as expected and were only limited by the characteristics of the obstacle detection of the robot.

The interaction design allows for full control of the differentially driven robot. However, the pedals cannot be pulled up with their current design, preventing backwards motion of the robot. This should be adjusted in future work.

Because no user tests have been conducted, no claims can be made about the effectiveness of the implementation of the haptic feedback. This will be required for further development and improvement of this system.

All technical project goals stated in section 1.3 of the introduction are met, a fully operational prototype is developed during the course of this assignment. Therefore the main goal of this assignment has been met. Recommendations for future work are discussed in chapter 6.

6 Recommendations

During the assignment multiple shortcomings of the system have been exposed. In this section multiple improvements for the system are recommended.

6.1 Software delay

The controller for the pedals is implemented in the ROS framework, this means that the sensor values pass a network connection and the driver node before arriving at the controller. The same connection is used for providing the output force of the controller to the motors. This architecture introduces a significant delay between the measured sensor values, and the calculated output force, resulting in a less stable controller. In order to decrease the delay between measurement and actuation, the controller software should run on the RaMstix. This removes the network delay completely and increases the responsiveness of the system.

The RaMstix has the option to be programmed as a 'real-time' system. This means that the system can receive data, process them and return the results sufficiently fast to affect the environment at that time [20]. It also means that the application can guarantee the response time for a given frequency. If the controller is running on the RaMstix, this could mean that the time between measurements and actuation is small enough to neglect the discretisation effect, or at least reduce it.

6.2 Controller

The controller of the pedals does not make use of an integrator and therefore has a steady state error as shown in section 4.1.4 of the testing & results chapter. The original idea was to take the square root of the displacement of the pedal and multiply it by the spring constant. This was supposed to result in a decreased steady state error and stiffer feeling pedal. Figure 6.1 shows the output of the pedal position based on a controller with the square root of the displacement and maximum feedback force. The steady state error is decreased and the stiffness increased, however, the pedal became unstable. Because the output force of the spring was altered, the virtual damper should be tuned to account for the change. This was not done due to time constraints, but if implemented, it could improve the pedals' precision and feeling.

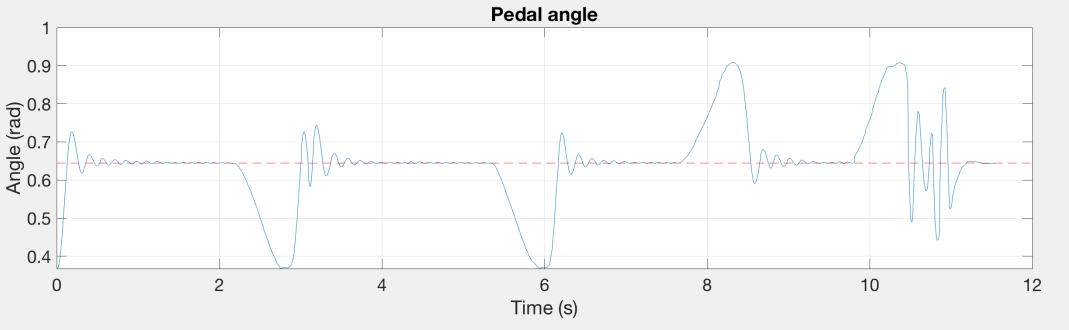


Figure 6.1: Pedal position output from the PD controller test with the square root of the displacement and maximum feedback force, the red dotted line represents the origin position

Another improvement to the controller could be made by implementing a gravity compensating force offset. Currently the gravity working on the pedal is countered by a static force offset added to the output of the controller. Using the weight of the pedal and the angle calculated by the encoder, the gravity working on the pedal can be calculated and countered accordingly.

6.3 Pedal construction

The pedals implemented in the LUC are designed for a one way motion and therefore not suited for the implemented interaction design. Changing the mechanical construction by placing the pivot point in the centre of the pedal, would allow the operator to go backwards without actively pulling the pedal up. Pushing the toes down would move the robot forward, while pushing the heel down would move the robot backwards as shown in figure 6.2. This would reduce the workload required to operate the pedals.

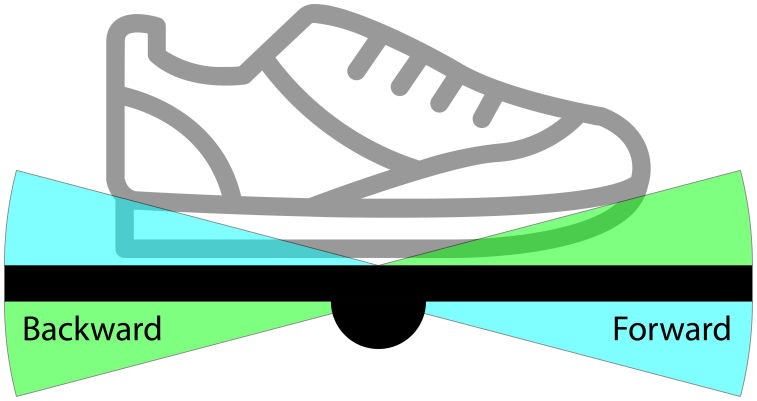


Figure 6.2: Pedal with pivot point at the centre for easier control

References

[1] T. M. Lam, M. Mulder, and M. M. van Paassen, "Haptic Feedback for UAV Teleoperation - Force offset and spring load modification," in *2006 IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1618–1623, IEEE, oct 2006.

[2] D. A. Abbink and M. Mulder, "Exploring the Dimensions of Haptic Feedback Support in Manual Control," *Journal of Computing and Information Science in Engineering*, vol. 9, no. 1, p. 011006, 2009.

[3] T. M. Lam, M. Mulder, and M. M. Van Paassen, "Haptic interface in UAV teleoperation using force-stiffness feedback," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 835–840, 2009.

[4] F. Huang, R. B. Gillespie, and A. Kuo, "Haptic feedback and human performance in a dynamic task," in *10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pp. 24–31, 2002.

[5] M. Roo, "LUC: Haptic pedal." 2012.

[6] Boikon. `https://www.boikon.nl/`. [Online; accessed 29-April-2018].

[7] Maxon. `https://www.maxonmotor.com/maxon/view/product/370354`. [Online; accessed 29-April-2018].

[8] Infineon. `https://www.infineon.com/dgdl/Infineon-Encoder_HEDS-5540-A14-AP-v01_00-EN.pdf?fileId=5546d46147a9c2e40147d3d593970357`. [Online; accessed 29-April-2018].

[9] Sparkfun. `https://www.sparkfun.com/datasheets/Accelerometers/MMA7260Q-Rev1.pdf`. [Online; accessed 29-April-2018].

[10] E. Dertien. `http://wiki.edwindertien.nl/doku.php?id=education:sensors:tools`. [Online; accessed 29-April-2018].

[11] RS. `https://nl.rs-online.com/web/p/desktop-power-supply/5388995/`. [Online; accessed 29-April-2018].

[12] Embeddedarm. `https://www.embeddedarm.com/Manuals/ts-7300-datasheet.pdf`. [Online; accessed 29-April-2018].

[13] 20Sim. `http://www.20sim4c.com/`. [Online; accessed 29-April-2018].

[14] Elmomc. `http://www.elmomc.com/products/whistle-digital-servo-drive-main.htm`. [Online; accessed 29-April-2018].

[15] Embeddedarm. `https://wiki.embeddedarm.com/wiki/TS-9700`. [Online; accessed 29-April-2018].

[16] ROS. http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup. [Online; accessed 29-April-2018].

[17] ROS. http://wiki.ros.org/msg. [Online; accessed 29-April-2018].

[18] RaMstix. https://www.ram.ewi.utwente.nl/ECSSoftware/RaMstix/docs/pin_overview.html. [Online; accessed 13-06-2018].

[19] ZeroMQ. https://http://zeromq.org/. [Online; accessed 02-08-2018].

[20] J. Martin, *Programming Real-time Computer Systems*. Prentice Hall, 1965.