On design and realisation of a telemanipulation demonstration setup

Arnold Hofstede

February 17, 2017

Summary

Within the i-Botics innovation center, founded by TNO and the University of Twente, tele-robotics is one of the research lines. One of the goals is to transfer knowledge on tele-robotics to the business community. Tele-robotics is used to perform manipulation on an external location by the operator. In this project a tele-manipulation setup will be designed and realised as a tool for knowledge transition.

The whole system, including a control base, will consist of a platform, robotic arm and gripper that are controlled by a joystick and haptic device from a remote computer. However, only the tele-manipulation platform, consisting of a movable platform and robotic arm, is realised in this project. The implementation of the gripper is not done since it was not available during the project.

The design and realisation of the setup consist of multiple parts: mechanical, electrical and software. These parts combined form the complete system. To design the system, System Engineering tools are used to keep track of the interfaces between the components.

First it has to be determined if the system can be realised, to do this a feasibility study has been done. In this study it is determined if the power supply is capable of supplying a sufficient power for the system, if the platform is capable of carrying the weight of the payload and if the system will be sufficiently stable when the arm is mounted to the frame.

Thereafter, it is determined whether the system can indeed be realised and the mechanical and electrical parts are designed and realised. These parts consist of a frame, mounting adapter for the arm, a mounting plate for the electrical components, a computer and a battery that supplies power to the components.

In the project ROS is used as a middleware to handle communication between the different components of the system. The software that is used to control the platform using a joystick is implemented using newly written code and reusing software components already available in the ROS community. The available software is used to interface the joystick and the platform, whereas the newly written code converts the output of the joystick interface software into the format of the inputs of the platform interface software.

To achieve a high level of reusability of the written software, Component-Based Software Design methodology is used. This methodology promotes the separation of concerns. This separations of concerns is applied in the design of the new software component.

The realised system is tested on a few subjects. These subjects are: capability of the platform to bear a high payload, the power supply, software and movement. It is concluded that the system works as designed, except for sideways movement the platform. Also the realised software component and the reused components work correctly, accordingly to the expectations.

To improve the sideways moving capabilities of the platform a controller that corrects the angular and forward movement errors should be implemented. Additional sensors are required for this since the included odometry sensors on the platform are not accurate enough due to slipping of the wheels in which rotary encoders are build.

To also implement the software of the system outside the ROS framework new software, using the separation of concerns principle, for interfacing the movable platform and the joystick have to be realised.

Samenvatting

Binnen het i-Botics innovatiecentrum, opgericht door TNO en University of Twente, is telerobotica een van de onderzoeksdoelen. Een van de doelen is informatie over telerobotica over te dragen aan het bedrijfsleven. Telerobotica wordt gebruikt om een persoon een handeling te laten verichten op externe locaties. In dit project wordt er een telemanipulatie demonstratie opstelling ontworpen en gerealiseerd als een middel om deze kennis over te dragen.

Het hele systeem, inclusief een besturingsconsole, bestaat uit een platform, robotarm en een hand die respectievelijk aangestuurd worden door een joystick en een haptische controller. In dit project is alleen tele-manipulatie platform, bestaande uit een rijdbaar platform en robot arm, gerealiseerd.

Het ontwerpen en realiseren van de opstelling bestaat uit verschillende onderdelen: er is een mechanisch deel, een elektrisch deel en er moet software gemaakt worden om het geheel aan te sturen. Deze onderdelen gecombineerd resulteert in the complete systeem. Om het geheel in een overzichtelijke manier te ontwerpen wordt er gebruik gemaakt van System Engineering.

Om zeker te weten dat het geheel daadwerkelijk gerealiseerd kan worden wordt er een haalbaarheid studie gedaan. In deze studie is bepaald of het mogelijk is om het systeem van stroom te voorzien doormiddel van een batterij, en als dit mogelijks aan welke specificaties deze batterij moet voldoen. Ook is het bepaald of het platform in staat is om het gewicht te dragen dat op het platform geplaatst wordt. Als laatste wordt onderzocht of het systeem stabiel genoeg is als de robotarm aan de zijkant bevestigd wordt.

Nadat het bepaald is dat het systeem daadwerkelijk gemaakt kan worden, worden de mechanische en elektrische onderdelen ontworpen en gemaakt. Deze onderdelen bestaan uit een frame, adapter voor de arm, een plaat om alle elektronica op te bevestigen en een batterij die het geheel van stroom voorziet.

In het project is ROS gebruikt als middleware om de communicatie tussen de componenten in het systeem te voorzien. De software die gebruikt is om het platform te besturen door middel van een joystick gekoppeld aan een externe computer bestaat deels uit hergebruikte software vanuit de ROS-community en deels uit nieuw geschreven software. De hergebruikte software is gebruikt om te communiceren met de joystick en het platform . De nieuw geschreven software is in staat om de gegenereerde data van de joystick software om te zetten naar het ingangs formaat van de platform software.

The available software is used to interface the joystick and the platform, whereas the newly written code converts the output of the joystick interface software into the format of the inputs of the platform interface software.

Om een hoog niveau van herbruikbaarheid van de geschreven software te bereiken is de 'Component Based Software Design' methodologie gebruikt. Deze methodologie maakt gebruik van het zogenaamde 'seperation of concerns'. Hierin zijn de functionele delen van de software onafhankelijk van het gebruikte framewerk.

Het gerealiseerde systeem is op een aantal punten getest: het kunnen dragen van de zware lading door het platform, de stroomvoorziening, de software en het bewegen van het platform.

Het is geconcludeerd dat het systeem werkt zoals ontworpen, met uitzondering van het zijwaarts bewegen van het platform. Ook de geschreven software en de hergebruikte software dat geïmplementeerd is werkt correct.

Om het zijwaarts bewegen van het platform te verbeteren zou er een controller geïmplementeerd moeten worden die ervoor zorgt dat als het commando 'zijwaarts bewegen' gegeven

wordt het systeem niet roteert of voorwaarts beweegt. Aangezien de odometry sensoren op het platform niet in staat zijn om goede verplaatsingsinformatie te geven door het slippen van de wielen zullen deze vervangen moeten worden door een sensor die niet afhankelijk is van de rotatie van de wielen.

Ook zal de rest van de software hergeschreven moeten worden om het platform te kunnen gebruiken in combinatie met andere framewerken. Hierbij zou dan gebruik gemaakt moeten worden van het 'seperation of concerns' principe.

Contents

1 Introduction

1.1 Context

This project is done in the scope of i-Botics. i-Botics is an open innovation center for research and development in interaction robotics. The research center has been founded by TNO and University of Twente to develop knowledge and technology for value adding robotic solutions.

One of the research lines of the project is tele-robotics. Tele-robotics is used to do remote operation that is steered by an operator. This separates the operater from the workspace and enables him to do work at places where he is not able to perform the work himself, due to for example safety or travel time.

In this project, a part of the telemanipulation demonstration setup will be realized as a first step to transfer knowledge about telemanipulation towards the business community. The setup will consist of a robotic arm on a moveable platform which can be operated by using a haptic device and joystick, respectively. In Figure 1.1 a sketch is included of a complete telemanipulation setup.

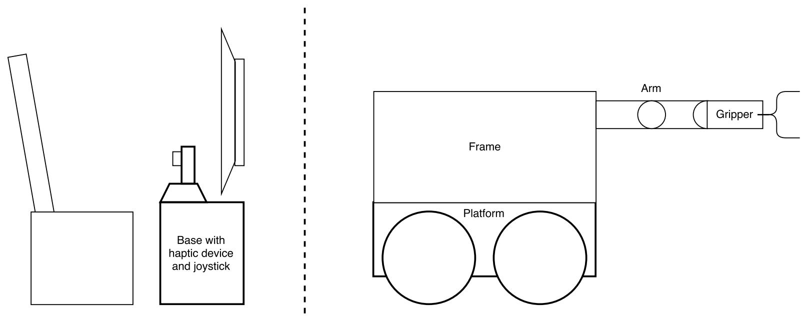

Figure 1.1: Sketch of the to be realised telemanipulation system.

1.2 Project goals

The first goal of the project is to realise a physical moveable system with an arm and gripper that can carry all components, power all components and is able to communicate with an external computer. The second goal is to implement some of the software functionality: the platform must be controlled by a joystick connected to an external computer. If time permits more software functionality, such as the software for the arm and haptic device, can be implemented.

1.3 Approach

To make sure that the system can be realised using the available components a feasibility study will be done first. When this study is done, the systems frame will be designed and realised. After all the components are connected, the software parts will be implemented. Initially, available code in combination with newly written code will be used to run the system.

The software will be realised and implemented using a component-based software design methodology which results in a high level reusability of the individual components. However, this is not a main goal of the project and therefore, only the new software parts will be realised using this methodology. In this project the implementation of the code will be done in ROS.

In Figure A.1 all the task that have to be completed to realise the whole system are put in a graph, using dependencies as in a PERT chart. Since not each task can be completed each of the tasks is given a priority based on the MoSCoW method that is described in section 2.4.2.

1.4 Report organisation

The report outline is as follows:

- Chapter 2 gives background information on the used components, ROS framework, Component Based Software Design and System Engineering tools that are used.
- Chapter 3 is a feasibility study which concludes if the system is feasible.
- Chapter 4 discusses the design of the system.
- Chapter 5 discusses the realisation of the system.
- Chapter 6 evaluates the realised system.
- Chapter 7 concludes the report with conclusions and recommendations.

2 Background

2.1 Components

In this section the components of the complete system are discussed. These components are: Segway RMP 50 omni, KUKA light weigth robot 4+, Right Hand Robotics ReFlex TakkTile, force dimension Omega.7 and Logitech Extreme 3D pro. For each component a brief description, relevant specifications, communication means and power requirements are given.

2.1.1 Segway RMP 50 omni

The Segway RMP 50 omni is an omni-directional platform, consisting of 2 normal Segway's, that has an embedded control processor in each of the 2 Segway's that independently control the motors each axle. The platform uses mecanum wheels, named after the company at which they were invented, that have rollers attached to its circumference at an angle of 45 degrees. By adjusting the speed of each wheel the platform can drive forwards, sideways or rotate. In Figure 2.1 the direction of movement of the platform at different wheel velocities is given.

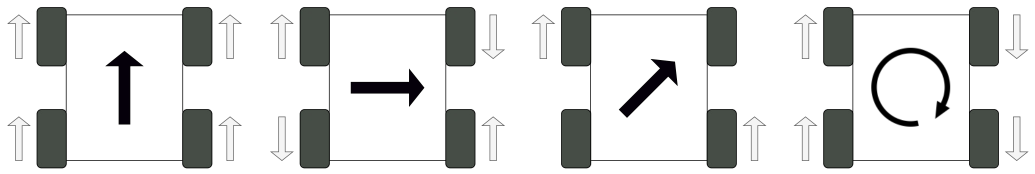

Figure 2.1: Movement of segway RMP 50 omni due to the mecanum wheels

The individual platforms are connected by an extruded aluminium structure, that can be used for equipment mounting. The platform supports a payload of 68 kg. In Figure 2.2 the RMP 50 omni is shown.

Communication with the platform can be done in three ways: CAN, USB via FTDI's D2XX Library and Serial. In the platform a battery is included, which is used to power the propulsion system. This battery is capable of letting the platform drive 5 km or stand in standby mode for 8 hours. Also an additional 145Wh battery is build in to power auxiliary devices. This auxiliary power supply is limited to 150 W at 24V or 48V (Segway, 2011).

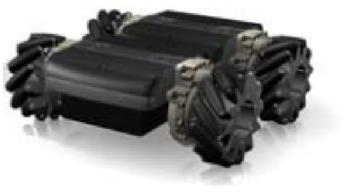

Figure 2.2: The Segway RMP 50 omni

2.1.2 KUKA Light Weight Robot 4+

The KUKA Light Weight Robot 4+, shown in Figure 2.3, is a robotic arm with torque sensors integrated in the joints. This makes this arm capable of sending a force-feedback value. The KUKA LWR 4+ is controlled by its control box, the KR C2 lr, that can be operated by a remote control pad. The arm is capable of lifting a 7 kg payload and has a range of 790 mm (KUKA, 2012b).

The arm can also be operated from an external computer using the Fast Research Interface (FRI) that allows the computer to communicate with the control box over Ethernet using the UDP protocol. The arm receives its power from the control box which has a 230 V AC power input rated at 1.1 kVA. The arm includes build-in cables, both power and Ethernet, to be used to power

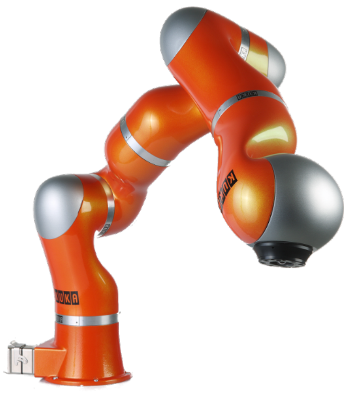

Figure 2.3: The KUKA LWR 4+

and to communicate with an auxiliary device without need of external wiring (KUKA, 2012a).

2.1.3 RightHand Robotics ReFlex TakkTile

The Right Hand Robotics ReFlex TakkTile (RightHand robotics, 2017), shown in Figure 2.4, is a gripper that has 4 degrees of freedom, three fingers and one preshape that rotates 2 of the fingers. All the three fingers have compliant joints and have 9 tactile sensors on each finger.

The gripper can communicate with a host computer over Ethernet and the creators of the gripper have made a ROS package for easy integration in a ROS system.

The weight of this gripper is 800 g and it must be powered by a 12 V power supply. When stalling the hand could draw up to 5 A, but for light testing a power supply that supplies 1.5 A is sufficient.

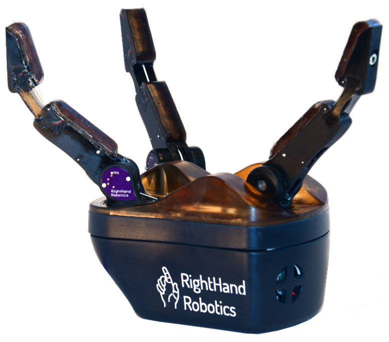
Figure 2.4: The Right Hand Robotics ReFlex TakkTile

2.1.4 Force dimension omega.7

The force dimension omega.7 (force dimension, 2017), shown in Figure 2.5, is a 7 degrees of freedom haptic device which can interface translation, rotation and grasping and is capable of giving haptic feedback on the translational axes. The device has a cylindrical translational workspace with a diameter of 160 mm and a depth of 110 mm, a rotational workspace of 240° around x, 140° around y and 180° around z, and a grasping workspace of 25 mm. The feedback on the translation axes can be up to 12 N and the grasping feedback up to 8 N.

The device communicates over USB 2.0 with a host computer and has a refresh rate of up to 4 kHz. A 230 V AC power supply is included to power the device.

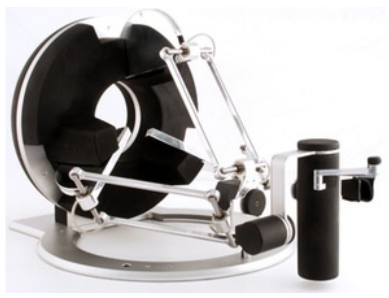
Figure 2.5: The force dimension omega.7

2.1.5 Joystick

The used joystick is a Logitech Extreme 3D pro, shown in Figure 2.6, which has 2 linear degrees of freedom and one angular degree of freedom. Additionally there are 16 buttons that can be used. Communication and powering the device is done over an USB interface.

Figure 2.6: The Logitech Extreme 3D pro

2.2 ROS

ROS (Robotic Operating System) is a distributed framework of processes (nodes). The communication between the nodes is done by an anonymous publish/subscribe mechanism. These nodes can be distributed over different computers in the same network and can communicate using TCP and UDP.

Within the ROS community, for a lot of components there already exists code. This code can be reused within other projects. This reusable code is available as packages.

2.2.1 ROS concepts

In this section the used ROS concepts are briefly introduced. Also an example of a simple system is included.

Master

The ROS Master enables individual ROS nodes to locate each other. This is done once when starting up the nodes. When the nodes have located each other they communicate peer-to-peer.

Nodes

Nodes are the processes that perform the computation within ROS. Most systems using ROS consist of multiple nodes which each perform a task. The use of nodes reduces code complexity since each node has limited functionality and can be seen from the rest of the system as a black box which performs computation and has known inputs and outputs.

Messages

Nodes communicate with each other by passing messages. A message is a data structure. A message description can be made to define custom data structures for messages sent in ROS.

Topics

The messages are passed by publishing it to a given topic, to which another node can subscribe to receive this data. Each node may publish and subscribe to multiple topics.

In Figure 2.7 an example of a simple system is visualised where a node communicates with an external device and another node processes the data of the external device. When the nodes are started, they register to the master. `Node_1` states that it will publish to topic`/unprocessed_data` and `Node_2` states that it will subscribe to it. After registration `Node_1` and sends the `/unprocessed_data` message directly to `Node_2`.

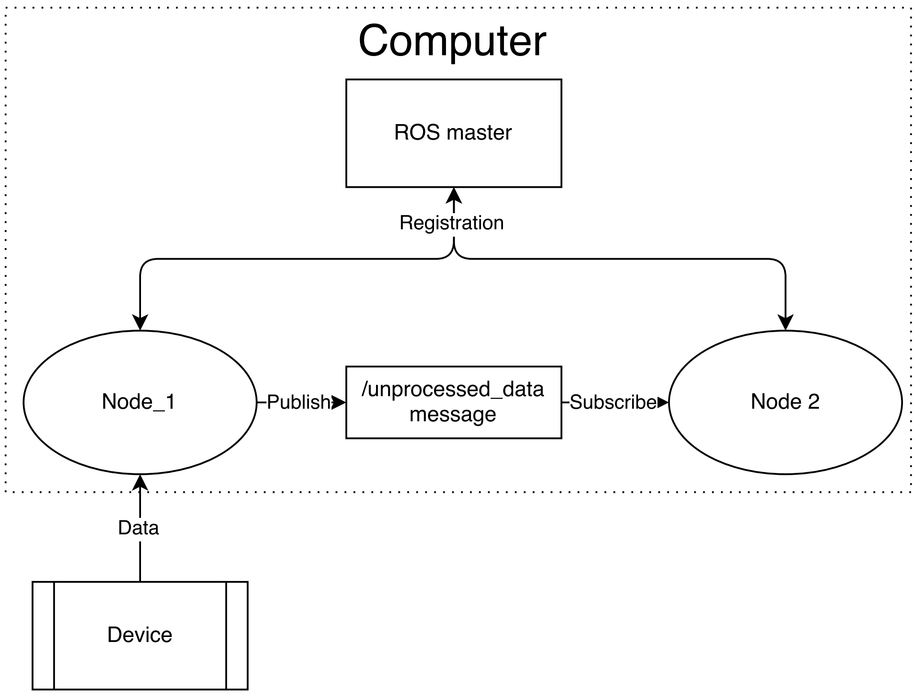

Figure 2.7: Simple system consisting of 2 nodes

2.2.2 ROS packages

Software in ROS is organized in packages. A packages is the smallest individual thing that can be build in ROS. It contains everything to make a useful module. They can be used to implement functionality of a certain component in another (ROS) system. A package can consist of the files and folders in Figure 2.8. Not all of these folders are required to make an useful module.

To build and run the package only `CMakeLists.txt` and a C++ file in src or a Python script in scripts is required, depending on the choise of programming language. However, others might not be able to install the package since they might be missing package dependencies that are on the computer of the developer, but not on the computer of the person that reuses the software. When releasing a package to the the ROS community these dependencies have to be declared in `packages.xml`.

If extra files are used in a package they should be stored in the correct folder. Configuration files should stored in the config folder, headers and library files in the include folder, launch files in the launch folder, custom message definitions in the msg folder, service definitions in the srv folder and action definitions in the action folder.

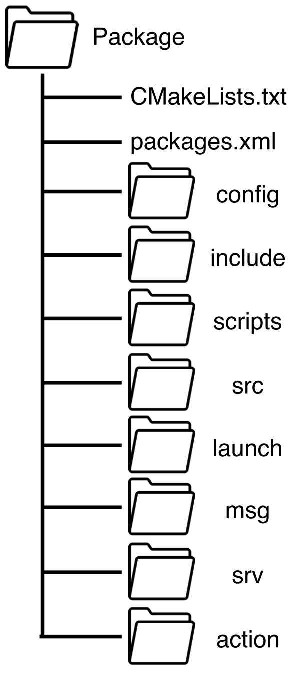

Figure 2.8: ROS package file system

2.3 Component Based Software Design

To achieve a high level of code reusability, Component-Based Software Design Methodology (CBSD) can be used. CBSD promotes the separation of concerns which makes it possible to reuse the computational functions of the code, even when using a different framework than ROS. The 5c "meta-model" by Bruyninckx et al. (2013) helps to separate the framework from functional code. These 5 concerns are: Computation, communication, coordination, configuration and composition.

When making new software components, the framework dependent and independent code should be separated to achieve a high level of reusability. When these codes are separated, the functional code can also be used in combination with other frameworks.

2.4 System engineeringing tools

2.4.1 n^2 diagram

The N^2 diagram is a square N x N matrix with the functions (subsystems) on its diagonal (Bonnema et al., 2016). It is used to inventory all interfaces between functions and to monitor these interfaces during the development of the system. The outputs of the functions are on the rows and the inputs are in the columns. An example is displayed in Table 2.1.

Function 1	output of 1 input of 2	
	Function 2	output of 2 input of 3
output of 3 input of 1		**Function 3**

Table 2.1: Example of a n^2 diagram

2.4.2 MoSCoW

To prioritize requirements or tasks of a project, the MoSCoW-method can be used. MoSCoW stands for must haves, should haves, could haves and will not haves, which each determine the priority of a task. Without the must have tasks finished the project will fail. The should haves are not necessary for delivering the project, but are the most critical after the must have tasks. The could have requirements are desirable, but not necessary for the project. The will not have requirements are either least critical or not yet possible realise.

2.4.3 PERT chart

PERT, Program Evaluation and Review Technique, can be used in project management. It is designed to represent the tasks involved in completing a project. Each task in a PERT chart has a predecessor task, successor task or both. To these tasks an optimistic, pessimistic and most likely time can be given. Using this, the expected time and the change of failing the deadline can be calculated.

PERT can be used in conjunction with the critical path method. The longest time of succession tasks will be the project time. However, this is only the case if parallel tasks are performed by multiple people.

3 Feasibility study

In this chapter a feasibility study is done to determine if the system can be made with the components described in section 2.1 and which additional components are needed to complete the system.

Firstly, it is determined if it is feasible to use a battery to power the components. Reason for this is the high power consumption of the KUKA controller. Secondly it is determined if the Segway is capable of handling the payload which will be installed on it since the battery, controller, arm and frame are each heavy components. Lastly it is calculated if the arm can be mounted sideways to the platform, because due to its length and weight it might be unstable.

3.1 Power supply

The purpose of the to be designed setup is to transfer knowledge about telemanipulation using demonstrations. It is assumed that these demonstrations will take less then 60 minutes. Therefore, the battery on the platform must have a capacity such that the system can be used for 60 minutes. The RMP 50 omni itself has an integrated battery with sufficient capacity; it can drive 1 hour and has a standby time of 8 hours (Segway, 2011). Therefore, it is not included in the calculations for battery capacity.

The maximum power consumption of the KUKA LWR 4+ is 1100 W (KUKA, 2012a). However, average power consumption will be well below this. The average power consumption cannot be measured yet due to the fact that the code for controlling the KUKA arm is not yet written and specifications of the average power consumption of the KUKA LWR 4+ are not available.

The KUKA LBR 3, predecessor of the LWR 4+ is using 150 W during normal operation (Loughlin et al., 2007). However, this power consumption of 150 W does not include the power consumption of the controller. To be able to choose a suitable battery, the average power consumption of the LWR 4+ and controller is assumed to be 300 to 400 W.

Due to the extensive safety features in the controller it is decided to use a DC to AC inverter instead of replacing the whole power supply with DC to DC converters at the cost of lower efficiency. This safety system consists of a circuit that uses an backup battery inside the controller for a controlled shut down of the system during a power failure. Since the controller will be powered using a battery, these power failures might occur. Also a regular shut down of the controller is done using this safety battery: the power has to be cut from the controller and then the controller shuts down.

At an expected efficiency of 90% of the inverter, the power consumption, at the battery, of the controller and arm would be 340 to 450 W. At 12 V, the average current draw from the battery will be between 28 A and 38 A.

An Intel mini PC, consisting of a core i3 processor, 4GB ram and a 128GB ssd, is used as platform computer. Under load, its power consumption will be between 20 and 30 W.

The ReFlex TakkTile hand has a stall current of 5A at 12 V, however this is a worst case scenario. The manufacturer itself states that the device can be used with a 12V 1.5A power supply. Therefore, a power consumption of 20 Watt is used for calculations.

To be able to use the demonstration setup for 1 hour, a battery capacity of circa 500 Wh is required. This can be realised with a 12 V battery with a capacity of 40 Ah or a 24V battery with a capacity of 20 Ah.

The battery capacity of lead-acid, gel and AGM batteries are effected by Peukert's law, that expresses the capacity at different discharge currents, which is:

$$t = H\left(\frac{C}{I * H}\right)^{k} \tag{3.1}$$

in this equation, t is the time in hours until the battery is empty, H the number of hours on which capacity C is based, I the current which is drawn and k is the Peukert exponent. Equation 3.1 can be rearranged to calculate the required battery capacity per type of battery:

$$C = I * H * \sqrt[k]{\frac{t}{H}} \tag{3.2}$$

Based on output voltage, availability, discharge current and safety, a selection of 4 battery types that are candidates as the power supply of the system is made. These 4 battery types are: Lead-acid, AGM (Absorbent glass mat), gel and LiFePO4. From these batteries, AGM and LiFePO4 have the best deep cycle lifetime.

For the calculations of the required capacity per battery using equation 3.2 the following variables are used: I = 40 A, H = 20 hours and t = 1 hour. In Table 3.1 result of this calculations are written. For AGM batteries the Peukert exponent ranges between 1.05 and 1.15; for gel batteries the exponent ranges between 1.1 and 1.25 and for flooded batteries the exponent ranges between 1.2 and 1.6. The capacity of Lithium batteries is not affected by the current that is drawn.

Type	**Peukert's exponent**	**Capacity**	**Capacity at 40A**	**Weight**	**Costs**
Lead-acid	1.4	94 Ah	40 Ah	28 kg	€80-€150
Gel	1.2	65 Ah	40 Ah	20 kg	€100-€150
AGM	1.1	55 Ah	40 Ah	17 kg	€160-€220
LiFePO4	-	40 Ah	40 Ah	6 kg	€500-€600

Table 3.1: Capacity calculations using peukert's exponents

At a capacity of 40 Ah on 12 V the battery is capable of delivering 480 Wh. Since the average power consumption is assumed to be between 360 and 460 Watt the battery life will be between 63 and 80 minutes, which is sufficient.

From Table 3.1, it can be concluded that the LiFePO4 is the best battery to be used, however the are expensive and not well available at the required capacity. The definitive battery type depends on whether it fits in the mass budget or not. If an AGM battery fits within the mass budget it will be used since it has excellent deep cycle behaviour, whereas gel and lead-acid batteries with similar deep cycle behaviour are only slightly cheaper than AGM batteries.

3.2 Payload

The maximum specified payload that can be mounted on the RMP 50 omni is 68 kg. However, a developer of the platform noted that it can be used without problems up to 100 kg. If the payload on the platform is too high, the rollers in the mecanum wheels will bend. If the mass of the system is too much when the system is realised, new wheels with a higher capacity should be applied. Because all the components together are heavy, a mass budget is made. The mass of most of the components is given by their manufactures, but the mass of the frame and arm adapter plate must be estimated.

To estimate the frame mass, the specifications of Boikon 40 series aluminium profiles are used. These have a mass of 1.55 kg/m. Assuming that 4 x 0.8m, 4 x 0.4 m and 4 x 0.5m is required this results in a mass of around 11 kg. Also an aluminium adapter plate for the LWR 4+ is included which will approximately have the following dimensions: 400mm x 500mm x 10 mm. Using these dimensions and the density of aluminium, 2.7 g/cm^3, the plate will have a mass of 5.6 kg.

Component	**Weight (lead acid)**	**Weight (Gel)**	**Weight (AGM)**	**Weight (LiFePO4)**
KUKA LWR 4+	16	16	16	16
KUKA KR C2 lr	34	34	34	34
Battery	28	20	17	6
Inverter	5	5	5	5
PC	1	1	1	1
Frame	11	11	11	11
Adapter plate	5	5	5	5
Gripper	1	1	1	1
Payload	6	6	6	6
Total	**107 kg**	**99 kg**	**96 kg**	**85 kg**

Table 3.2: Mass budgets for the different battery types

The total weight of the system mounted upon the RMP 50 omni will be between 85 kg and 103 kg depending on the used battery. From Table 3.2 can be concluded that the LiFePO4, AGM and Gel batteries all fit within the weight budget.

3.3 System stability

In this section, it is calculated if the platform will be stable if the robotic arm is mounted to the front side of the frame on the platform. In this calculation the arm is mounted 15 cm right of the front axle. The position of all components is relative to the front axle. This calculation is done for the worst case scenario in which the maximum payload of 7 kg is used. In Figure 3.1 is this position of the masses visualised.

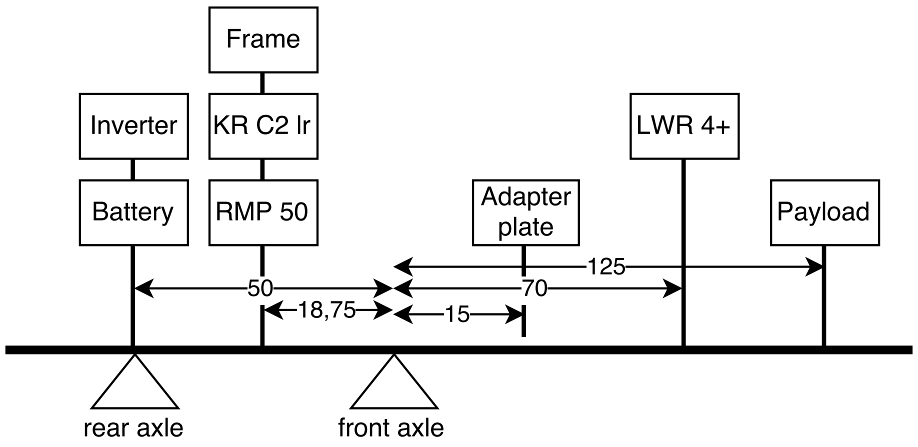

Figure 3.1: The partes balancing on front axis, with distances in cm

In Table 3.3 the moment calculations for each component are done. The balance calculation shows that the system will be stable even in the worst case scenario with the maximum payload positioned at the end of the arms range. Therefore the arm can be mounted horizontally at the front of the frame.

Item	**Weight**	**Position**	**Momente left**	**Item**	**Weight**	**Position**	**Moment right**
RMP 50	60 kg	0.1875 m	110 Nm	KUKA lwr 4+	16 kg	0.70 m	110 Nm
KR C2 lr	34	0.1875 m	62.5 Nm	Adapter	5.6 kg	0.15 m	8 Nm
Battery	17 kg	0.50 m	83 Nm	Gripper	1 kg	1.25 m	12 Nm
Inverter	4 kg	0.50 m	20 Nm	Payload	6 kg	1.25 m	73.5 Nm
Frame	11 kg	0.1875 m	20 Nm				
		Total:	**295.5 Nm**			**Total:**	**203.5 Nm**

Table 3.3: Stability calculation

3.4 Conclusion

From the previous sections, it can be concluded that the system is feasible regarding the power supply, payload and stability. Since in each calculation the worst case numbers are used, this conclusion is usable in all situations. Based on weight, availability, costs and deep cycle performance, an AGM battery will be used. This means that the weight of the system while lifting the maximum payload will be 96 kg.

4 Design

The design is divided in 3 sections: Mechanical, Electrical and Software. In each section, it is stated which parts must be made, what design choices are made and how the components are realised.

4.1 Mechanical system

There are three mechanical parts that must be designed to realise the demonstration setup. The first part is a frame in which the electronics can be housed and to which the KUKA LWR 4+ can be mounted, the second part is a adapter plate that makes it possible to mount the KUKA LWR 4+ to the frame and the last part is a mounting adapter for between the arm and the gripper. The first 2 parts are discussed in the subsections below. The mounting adapter of the gripper will not be designed and realised due to the fact that it is not available during the project.

4.1.1 Frame

For construction of the frame, 40 mm x 40 mm aluminium profiles are used. The frame will be connected to the RMP 50 omni using a structure which is already on the platform. A render of the designed frame placed on the platform and including the location of some of the larger components is in Figure 4.1. The exact location of the electrical components can be seen in Figure 4.2. More detailed drawings of the frame, including dimensions, are shown in Appendix B.

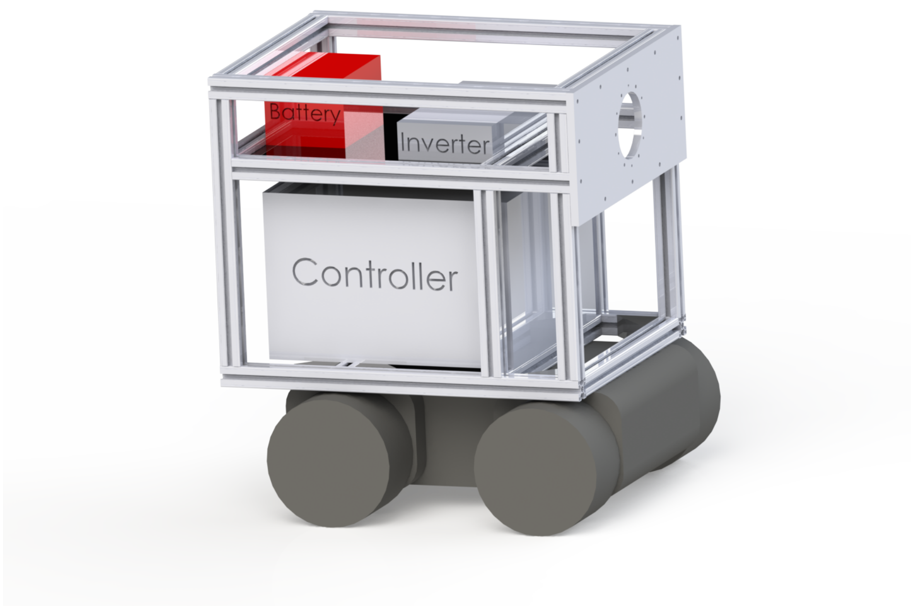

Figure 4.1: Render of the designed frame with some of the large components

The frame is designed such that the KUKA LWR 4+ will be mounted on a height of 710mm. Mounted at this height the system has a working range from the ground up to 1500mm. The working envelope of the system is displayed in grey in Figure 4.3.

Since the electrical components cannot be placed directly upon the controller a plate will be putt between the square aluminium structure in the middle of the frame. The position of the components on the plate can be seen in Figure 4.2, whereas the position of the plate including the larger components can be seen in Figure 4.1.

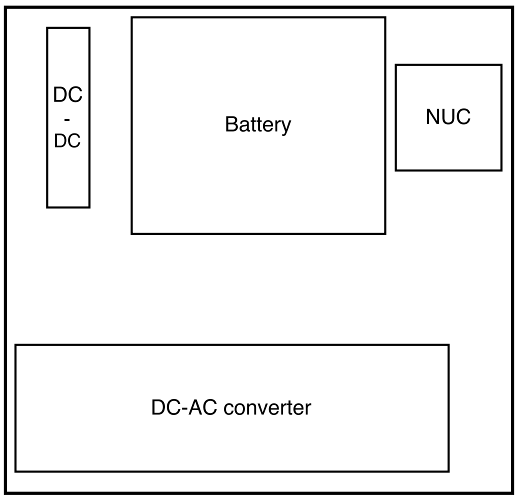

Figure 4.2: Intended position of the electrical components

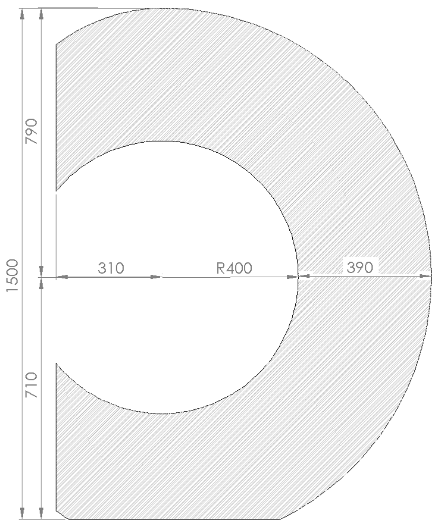

Figure 4.3: Range envelope of the KUKA arm mounted to the platform

4.1.2 Adapter plate

To mount the arm to the frame an adapter plate must also be made. This adapter will be realised of 565x250x15 mm aluminium plate. The width of this plate is equal to the width of the frame. To fasten the plate to the frame 8 mm holes, for M8 bolts, are included in the design. The position and dimensions of the holes are retrieved from the assembly manual of the Kuka LWR 4+ (KUKA, 2012b). In this plate, there are holes to fasten the plate on the frame. A render of the designed adapter plate is in Figure 4.4. More detailed drawings, that include dimensions, can be found in Appendix B.

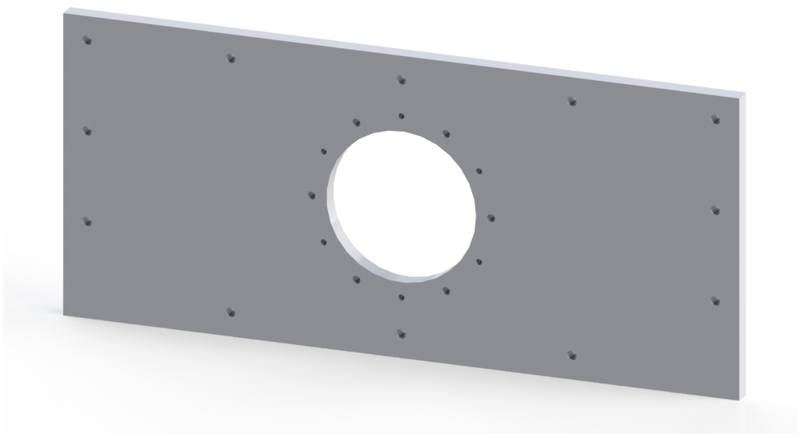

Figure 4.4: Render of the designed adapter plate

4.2 Electrical system

The electrical system consists of two parts, power and data. In Figure 4.5 an overview of the electrical system is given. In this figure all the power and data connections are drawn, each type having another colour. In subsections 4.2.1 and 4.2.2 the design choices are discussed.

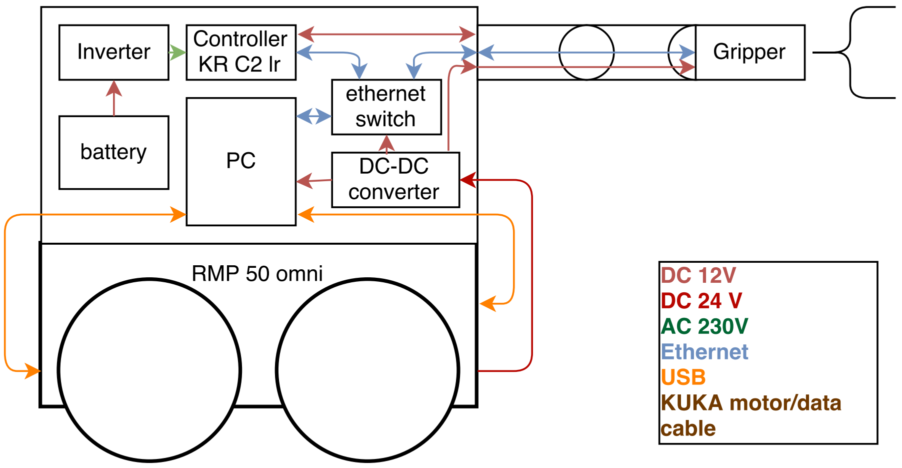

Figure 4.5: Overview of the electrical systems

4.2.1 Power

The components on the platform for which a power supply is needed are:

- Intel NUC
- Ethernet switch
- Right Hand Robotics ReFlex TakkTile
- Kuka KR C2 lr controller with the LWR 4+ arm

The Intel NUC, Ethernet switch and Right Hand Robotics ReFlex TakkTile all require a 12V power supply, while the Kuka controller requires a 230 VAC power supply.

Since the Segway RMP 50 omni has an integrated battery for auxiliary devices that is not used to power the motors inside the platform, it is chosen to use this battery to power the platform computer and other 12V devices. This battery has a capacity of 145 Wh and can supply a maximum power of 150 W which is enough to power the computer, gripper and Ethernet switch. Using this battery has only benefits: the battery life of the whole system will increase, the current draw on the AGM battery is reduced and the 12 V components are not affected by the ripple that might be caused by the inverter.

The 24V output voltage of the battery in the Segway will be reduced to 12V using a DC-DC converter. In the feasibility study is assumed that the computer and gripper together us a maximum of 60 W. In later stages, additional components might be added to the system. Therefore the DC-DC converter must be able to supply more than 60 W. The converter that will be used is the *DCDC-USB-200* that can supply a constant power of 150 W. This leaves room for additional components with a combined continuous consumption of 90 W.

All these power in and outputs are represented in N^2 Table 4.1. As already mentioned in Section 2.4.1, the outputs of a component are in the rows and the inputs of a component are in the columns.

12V AGM accu		12V DC 40 A					
	Segway auxillary battery		24V DC 6 A				
		Inverter					230V AC 2 A
			DC-DC converter	12V DC 3 A	12V DC 0.2 A	12V DC 1.5 A	
				PC			
					Ethernet switch		
						Gripper	
							Controller

Table 4.1: Power in and outputs of the components

To power the controller it is chosen to use a 1000W pure sine DC to AC inverter in combination with a 12V 55Ah AGM battery. A pure sine inverter is chosen since this variant is capable of powering devices with a switch mode power supply. The inverter that will be used is the *Mean Well TS-1000*. This inverter has some useful functions: when the battery is almost empty and the voltages reaches 11.3V it gives a battery low alarm. At 10.5V it shuts down to prevent the battery from over discharging.

In the feasibility study in Chapter 3 it was calculated that the system should be used for 60 minutes using a battery with this capacity. However, there was assumed that the computer and gripper were going to use the same battery as the arm. Since those components will now use the internal batteries of the Segway RMP 50 omni the battery lifetime will increase. Reducing the drawn current also reduces the effects of Peukert's law.

In Section 3.1 it was calculated that the battery life of the system would be between 63 and 80 minutes. Since there are less components that require power from the battery that powers the controller the battery life will increase. The average power consumption will now reduce from 360-460 W to 300-400 W. This means that the battery life of the controller will now be between and 72 and 96 minute

4.2.2 Data

On the platform 4 data connections will be realised: an Ethernet connection to the controller, an Ethernet connection to the gripper and two USB connections to the RMP 50 omni, one to the front axle and one to the rear axle. To be able to communicate a network is used to which both the base computer and platform computer are connected by wifi. In a later stage of the project, where the base station is used, a wired connection between the network and the base computer will be made. The gripper is connected using the internal Ethernet cable in the KUKA LWR 4+.

Table 4.2 is a n^2 diagram that shows how the connections between the components are realised. It can be seen in this table how the different parts; base, network and platform are separated from each other.

Gripper			Ethernet				
	Arm		Ethernet				
		Platform	USB 2x				
Ethernet	Ethernet	USB 2x	**Platform Pc**	Wifi			
			Wifi	**Network**	Wifi		
				Wif	**Base PC**	USB	
					USB	**Haptic Device**	
					USB		**Joystick**

Table 4.2: Data interfaces between components

The USB cables connecting the platform and the computer will be directly connected to the computer since it has enough ports. There is only one Ethernet port available on the computer. To solve this an Ethernet switch will be included on the platform.

4.3 Sofware

The software of the telemanipulation system is divided in 5 main software components: software for the platform, arm, gripper, haptic device and joystick. Some of these software components will be located on a base computer and some on the platform computer. The division of software on these two computers can be seen in Figure 4.6. Also the contents of the messages is given in this figure.

In this project the software to move the platform while using a joystick on a different computer will be made. Some of this software is already available in the ROS community and will be used initially. The parts that will be reused are the software packages to interface the Segway RMP 50 omni platform and the joystick.

Since the Segway RMP 50 omni consists of two regular Segway RMP 50 platforms it requires two inputs. Therefore two Segway RMP nodes must be set up. To convert the output message of the joystick node into two input messages for the Segway RMP nodes new node will be made.

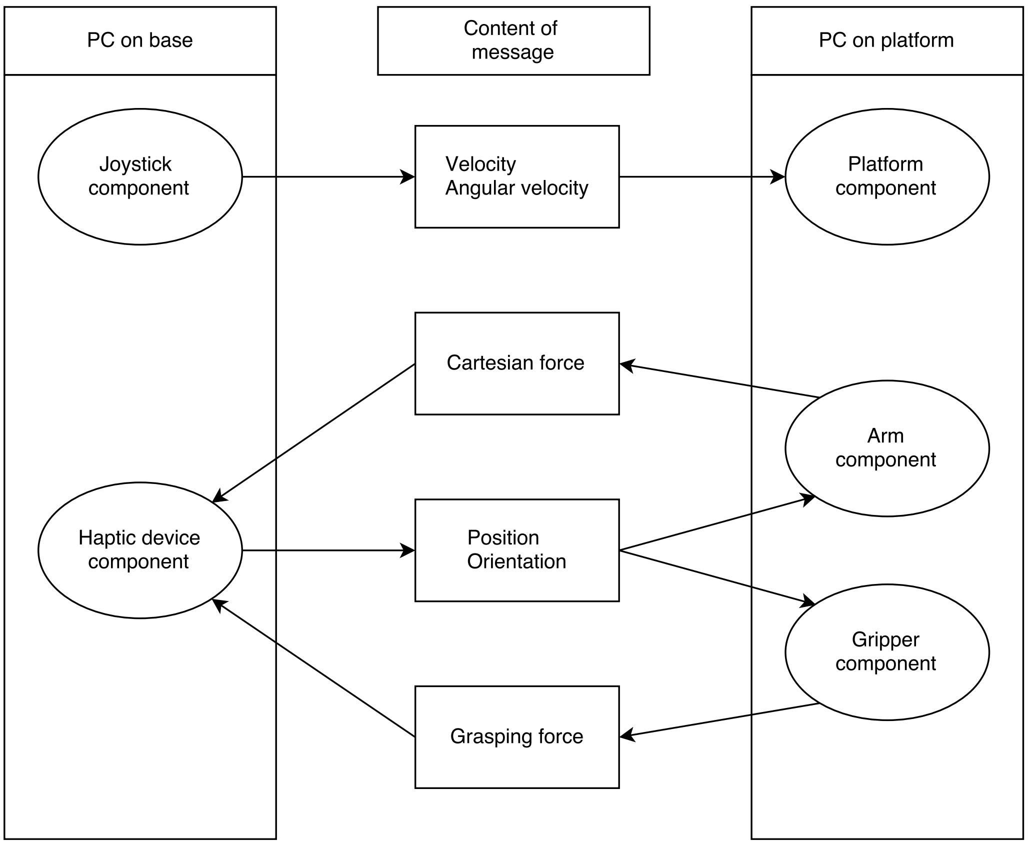

Figure 4.6: Conceptional overview of the software

To achieve a high level of reusability, seperation of concerns can be applied to software. For this the "5Cs" defined in the context of the BRICS Component Model are used (Bruyninckx et al., 2013). This separation of concerns will be applied in the software component that will be made to convert the messages from the joystick node to the two Segway RMP nodes. In the list below is stated how each concern will be separated:

- **Computation:** the computational elements of the software will be located in a framework independent file which contains only functional code. In the ROS packages this will be located in the `src` folder
- **Communication:** the communication between the software components will be handled by ROS. The file which handles the communication and calls the functional code will be located in the `src` folder.
- **Coordination:** The coordination of the system will be done in ROS using .launch files.
- **Configuration:** The configuration of the components will be done by using the parameters stored in a YAML file. In the ROS package these are located in the `config` folder.
- **Composition:** Composition couples the components such that they can be optimally reused. It handles the interaction the interaction between the 4 other concerns. In case of the software that will be realised in this project, composition is done by ROS.

5 Realisation

In this chapter the realisation and implementation of the designed components in chapter 4 are reported. Also changes that are made during the realisation are discussed.

5.1 Mechanical system

The designed frame in Section 4.1 is build and can be seen in Figure 5.1 and 5.2. To secure the controller in the frame some additional corner pars are used. In the plate, holes are made to fasten the electrical components to the system.

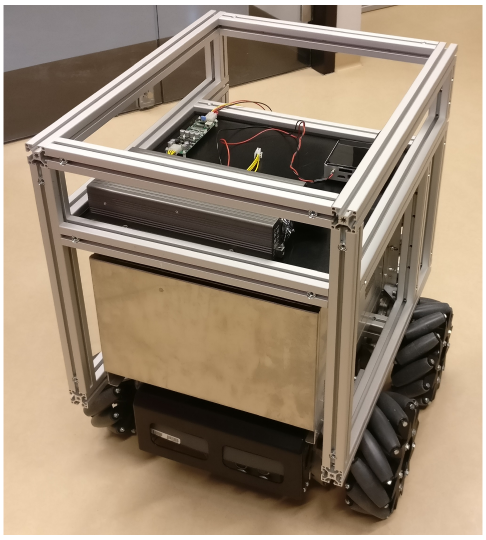
Figure 5.1: Rear view of the frame

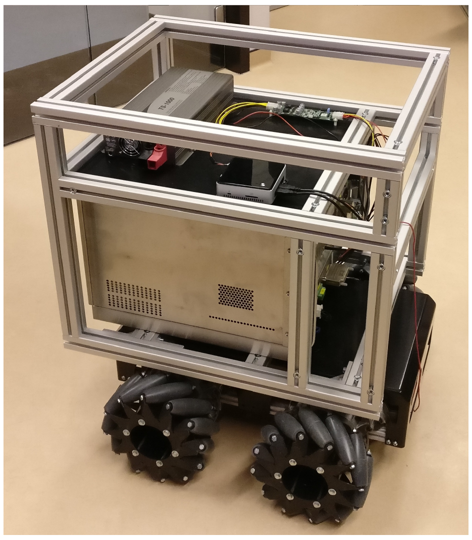
Figure 5.2: Side view of the frame

To mount the adapter plate, additional holes are drilled in the front of the frame are drilled. These holes are drilled in the same pattern as the holes in the adapter plate. The realised adapter mounted to the frame is shown in Figure 5.3

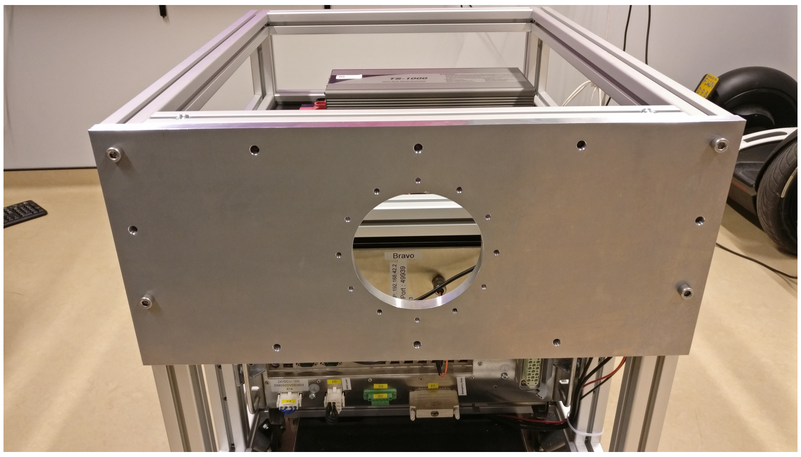
Figure 5.3: The adapter mounted on the frame

Since the mechanical parts of the system are now made, the arm can be mounted to the frame. A picture of the final system including the Kuka arm is Figure 5.4.

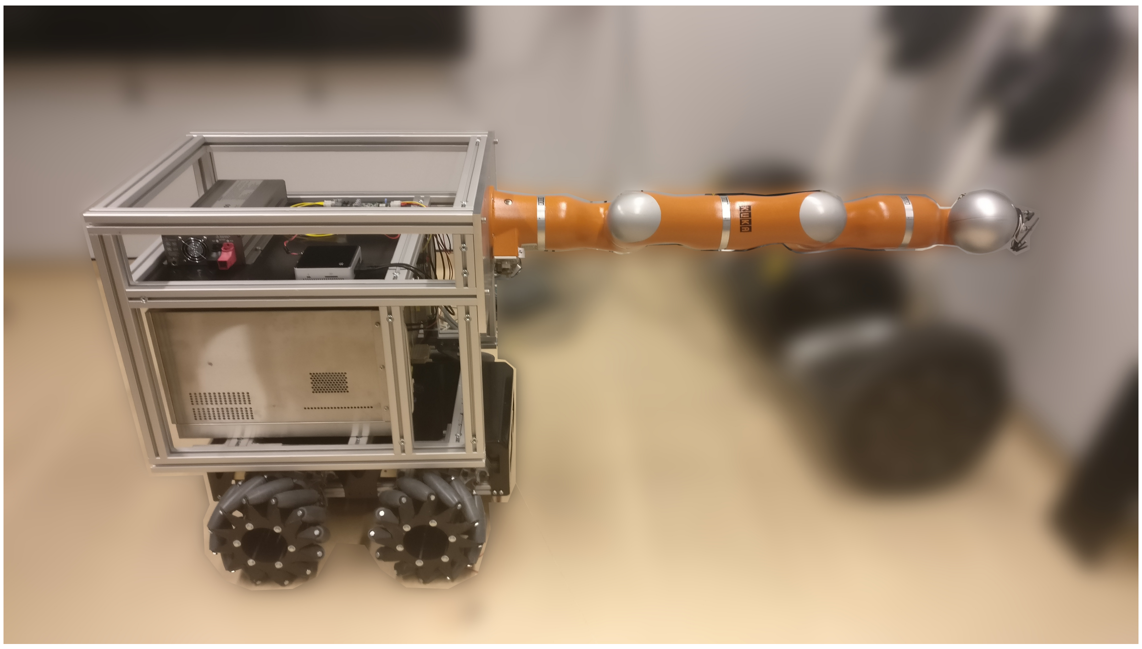

Figure 5.4: Picture of the realised system

5.2 Electrical system

5.2.1 Power

All the power connections, except for the the Ethernet switches power connection, are realised as they are described in Section 4.2.1.

5.2.2 Data

Most data connections are made such as they were design in Section 4.2.2. However, due to delayed delivery times of the Ethernet switch the Ethernet connection for the gripper is not yet realised

5.3 Sofware

Using the ROS function rqt_graph, a graph with all the nodes and topics is made. This graph is in Figure 5.5. In Appendix C is described how to install and run the software.

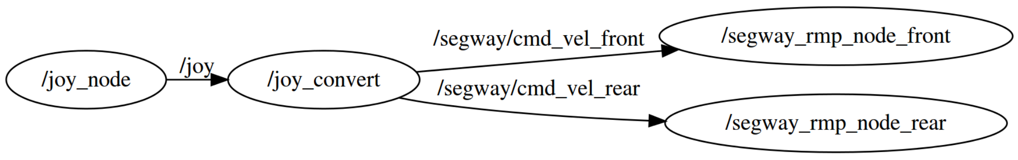

Figure 5.5: Overview of the nodes and topic

Platform computer

On the platform computer only the software to communicate with the Segway RMP 50 omni is located. To communicate with the the Segway RMP 50 omni the available package `segway_RMP` is used. In the launch file which starts the nodes on the platform, `segway_rmp_node` is started twice, one for the front and one for the rear. The software is configured to communicate with the Segway RMP platforms using serial.

Base computer

The ROS Master, a joystick interface node and a message conversion node are located on the base computer. The communication with the joystick is done using the available package `joy`. The coversion is done by the newly created node: `convert_joy`.

5.3.1 Joy_convert

The `joy_convert` node has been realised and converts the `Joy` message of the joystick into 2 `Twist` messages. A `Twist` message consists of a linear velocity and an angular velocity component. The message consists of two vectors with 3 components:

```
[linear.x, linear.y, linear.z]
[angular.x, angular.y, angular.z]
```

This node has be realised using the separation of concerns and implemented in ROS as described in Section 4.3. In Figure 5.6 the file structure of the package which contains the node is given. The realized node also serves as an use case for implementing the separation of concerns and is realised using the theory in Chapter 4.3.

The `joy` message consists of an array where the first 3 values are: translation in y direction, tranlation in x direction and rotation around z. These values will be mentioned later as `joy.y`, `joy.x` and `joy.z` The other values in this array are states of the buttons on the joystick.

The controllers in the front and rear axis each require a velocity set-point that consists of a `Twist` message. In this `Twist` message only the `linear.x` and `angular.z` component are used by the platform axes.

The front and rear axes of the Segway RMP 50 omni each require their own `Twist` message. In Figure 2.1 can be seen what each wheel requires to move in different directions and rotations. For forward movement they both require only `linear.x`, for sideward movement they each require an opposite value `angular.z`, to move diagonally they require both a combination of `linear.x` and `angular.z` and to rotate both the axes require the same `angular.z`.

Using the information above the following equation can be made:

```
 front.linear.x = joy.x
 front.angular.z = joy.y + joy.z
 rear.linear.x = joy.x
rear.angular.z = joy.y - joy.z
```

The functional code is written in a separate C++ file. The functions in this file are called by the main program that handles the communication. The computational code can be used within every C++ program and is not dependable on a framework such as ROS.

To also show how the configuration of software is handled some parameters are added in the software. With these parameters the linear and angular speed can be adjusted in the configuration file. These parameters are located in the config folder in `config.YAML`.

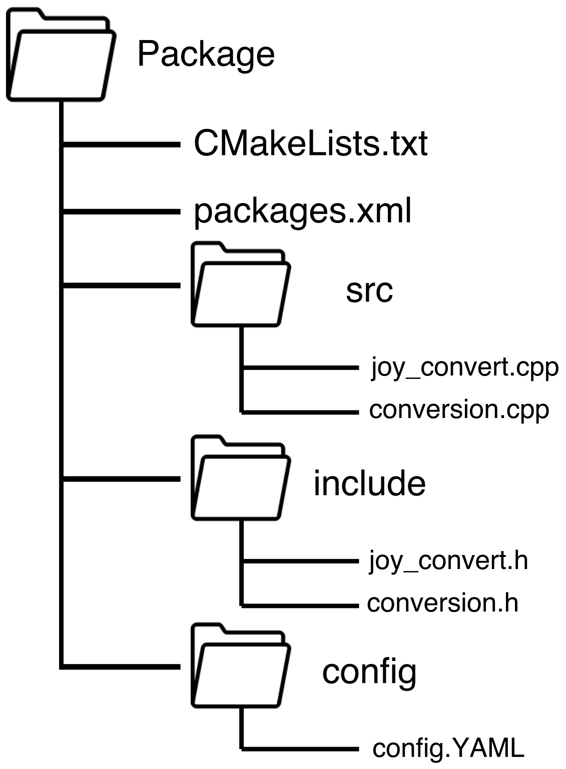

Figure 5.6: File structure of the realised package

6 Evaluation

After the realisation of the system some experiments are done to validate the design and to discover why parts are behaving different than designed. It is evaluated if the platform is indeed capable of handling the payload, if the battery and inverter able to supply sufficient power for the controller and arm, and if the software parts are working correctly.

After some initial tests on controlling the Segway RMP 50 omni it was observed that while strafing also forward movement and rotation occurred. This can be caused two ways: the wheels in the Segway do not reach the given velocity setpoint or the wheels do not all have equal traction. Therefore, some experiments are done to determine what the cause of bad strafing performance is.

6.1 Payload handling

Since the specified maximum payload of the Segway RMP 50 omni is 68 kg and the payload in this project is a lot higher, circa 100 kg, it is evaluated if the platform is able to handle this weight. A Segway engineer mentioned that his personal opinion was that 100 kg should not cause any problems and that the mecanum wheels are the weakest point of the system. If these mecanum wheel bend the payload would be to much.

The deflection of the rollers is evaluated by visual inspection. In Figure 6.1 are some photographs of the rollers under load. In these images the curvature of the roller is marked with a red line. In these images can be seen that the convex shape of the rollers is still intact. Although the bolts seem bend this is not the case. This is due to the perspective in which the photograph is taken.

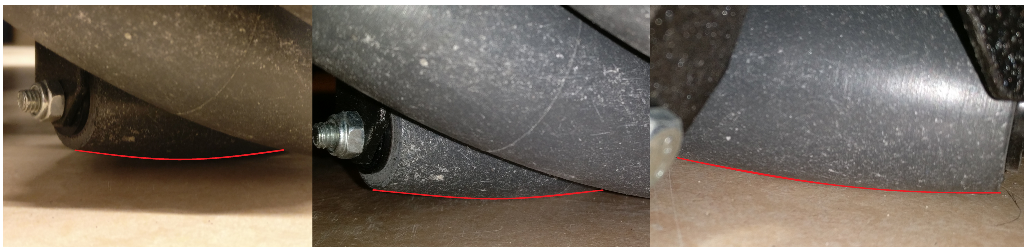

Figure 6.1: The curvature of the rollers under load.

6.2 Electrical

The power supply of the controller and arm, consisting of a battery and inverter, is not tested in combination with the controller and arm itself. This is not done because the software on the controller gave error messages when booting up and it can not be used till this is fixed. These error messages are not easy to solve since they do not appear in the "KUKA fault listing manual" and also the answer of the technical service was inconclusive. After further investigation it was concluded that the problem was in the safety circuit of the controller since several safety errors occurred. However, due to lack of time the problem has not been solved and the power supply is not tested completely.

The DC-AC inverter is tested on two fronts. Firstly it is tested what signal the inverter gives when the voltage comes below 11.3V to warn that the battery is almost empty since it was not specified how this warning is given. This appeared to be a "beep" sound. Secondly it was tested if the inverted was indeed capable of powering devices with a "switch mode power supply" which is in most computers. This was done by using a computer with an AC adapter connected to the inverter. The inverter is indeed suitable to power this kind of devices.

The controller, without the arm, is tested using the 230V AC powersupply on the platform. During this test the current draw on the battery is measured to be able to make a better estimate of the power consumption of the controller and arm combined.

The current is measured using a DC clamp meter. During the start up of the controller the current spikes up to 20 A, but after a while the current becomes constant at around 5 A. This means that at an output voltage of 12 V the power consumption of the controller, before the inverter, is 60 Watt. However, this is measured during idle and while not powering the safety board. Assuming that the power consumption of just the LWR 4+ arm is more or less equal to the power consumption of the LBR 3 and including some margin for the safety board the total power consumption will be circa 250 watt.

Now that there is a more accurate estimation of the power consumption also a more accurate estimation of the battery life can be made. Assuming that the available capacity of the battery due to Peukerts's law is 480 Wh the arm can be used for almost 2 hours.

6.3 Software

The software installed on the base and platform pc works. The `joy` message is correctly translated in to two `Twist` messages that are send to the platform. However, due to the heavy and not perfectly centered load the platform does not move sideways as it should. The user of the platform can manually compensate for the rotation that is introduced when moving sideways, but this is not a perfect solution.

At this moment, the software must be started manually on the platform, but this is not practical since there is not a keyboard, mouse and screen on the platform. For communication the Eduroam network has been used, but the platform and base computer do not have a static IP. This requires the user to edit the .bashrc file before the setup can be used. To solve this some options are available: register the computers at the ICT service desk to receive a static IP or use a different network.

6.3.1 Data throughput

To decide where to locate the `joy_convert` node, the used bandwidth of the `joy` message is compared with the bandwidth of the two `Twist` messages. Depending on whcih of these messages require the least bandtwidth it is chosen on which computer to locate the node.

The used bandwidth is measured using the ROS command: `rostopic_bw`. Firstly the data throughput of a `Twist` message is measured. When all the values are zero the `Twist` message has uses on average 2.4 kBps and when all the values are maximum it uses 6.0 kBps. For two `Twist` messages this is 4.8 kBps and 12.0 kBps.

Secondly the throughput of the joy message is measured. When all the values are zero it is on average 4.8 kBps and when all the values are maximum it is 12.0 kBps

From this can be concluded that it does not matter on which computer the `joy_convert` node is located. If 12.0 kBps would be to high one `Twist` message can be used to send the velocity to the platform on which it then must be separated into the two `Twist` messages.

6.4 Moving sideways

To examine whether the forward linear and angular movement while moving sideways is caused by the inability of the Segway to reach the desired wheel rotation velocity or not an experiment has been done. In this experiment it is examined if the wheels on one side rotate in opposite direction in equal speed. This is measured using a camera and a marker on each of the wheels, Figure 6.2. After each rotation of the front wheel the phase difference with the rear wheel is measured.

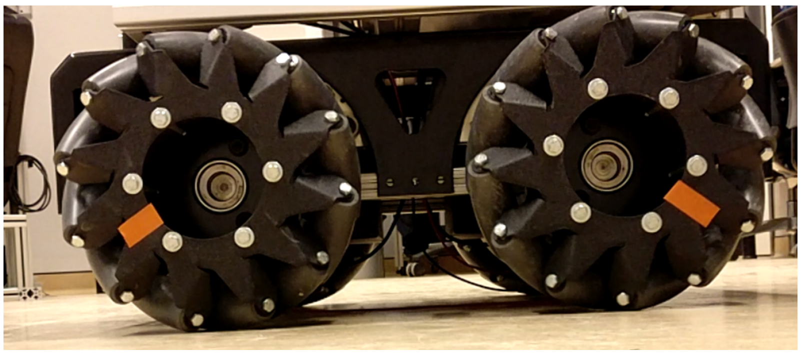

Figure 6.2: The markers used to measure the phase difference between the wheels.

The experiment has been done multiple times, but due to the limited space available each measurement has only a few rotations. The platform is controlled during the measurement using constant velocity setpoint of 0.5 m/s in the y direction, this value is send using a ROS command in the commandline and not by the joystick. The results of the measurements that have been done are written in Table 6.1. Although there are only a few rotations measured per measurement it can be seen that the difference in phase is caused in the first rotation.

If not reaching the desired wheel velocity would be the cause of not moving sideways correct the phase difference would increase of decrease over time. This is not the case; all the difference is made when the wheels are getting up to speed. Since the phase difference becomes constant after the first rotation it can be concluded that the incorrect sideways movement is not caused by not achieving the wheel velocity setpoint.

Since the platform moves also forward while it should move sideways and most of the weight rests on the front axle it is most likely that the incorrect weight distribution, and therefore not equal traction of the wheels, is causing the deviations while moving sideways. However, this hypothesis still needs to be verified.

Measurment 1			**Measurement 2**	
Rotation	Difference		Rotation	Difference
1	15		1	10
2	15		2	10
			3	10

Table 6.1: Measurements of the wheel rotation

7 Conclusion and Recommendations

7.1 Conclusion

A stated in the introduction the two goals of the project were to realise the platform side of the tele-manipulation and to implement the first software parts. Both of these goals were achieved, however the electrical system on the platform could not be fully tested.

The realised platform is capable of bearing all components and has a power supply for all components. Due to the heavy and not perfectly centered payload the platform is not capable of strafing in a straight line without compensation of the user.

The batteries in the Segway RMP 50 omni are capable to drive the platform and power the computer for at least the required 60 minutes. The battery that supplies power to the controller could not be tested due to failure of the controller

For controlling the platform with a joystick from an external computer software has been implemented. This software consists of reused packages and a newly made part which is created with a focus on reusability. In this project the software has been implemented using ROS, but in future projects the created code could be used in combination with other frameworks.

The newly created software package successfully converts the output data of the joystick into the input data of the Segway RMP platforms. This software package is created using the separation of concerns design principle.

7.2 Recommendations

In this section recommendations are done for further improvement of the realised system.

The capability of the battery to power the controller has not been tested. Therefore, the controller of the arm must be repaired to test its the power source in combination with the arm, since it is now only tested without powering the arm. During this test the power consumption of the arm and controller, and the battery life should be measured.

Since the platform is not able to move sideways linearly and the center of mass changes due to the movement of the arm a controller should be designed that compensates for the rotation. Due to the slipping of the wheels the odometry information of the Segway is not accurate. Therefore additional sensors must be added to the platform if a controller will be implemented.

To be able to use all the components framework-independently, the software to interface the joystick and the Segway RMP 50 omni should be refactored using a component-based software design methodology.

At the moment Eduroam is used as a network for the communication between the computers. The computer does not have a static IP yet, which has the effect that the user has to change setup files after most reboots. Also the software on the platform computer cannot be started without the use of display, keyboard and mouse. These problems can be solved by requesting a static IP form the network administrator, but also by using a network of which the settings can be edited without having to request this. This last option has the benefit that new computers can easily be added to the system without having to wait.

A Tasks

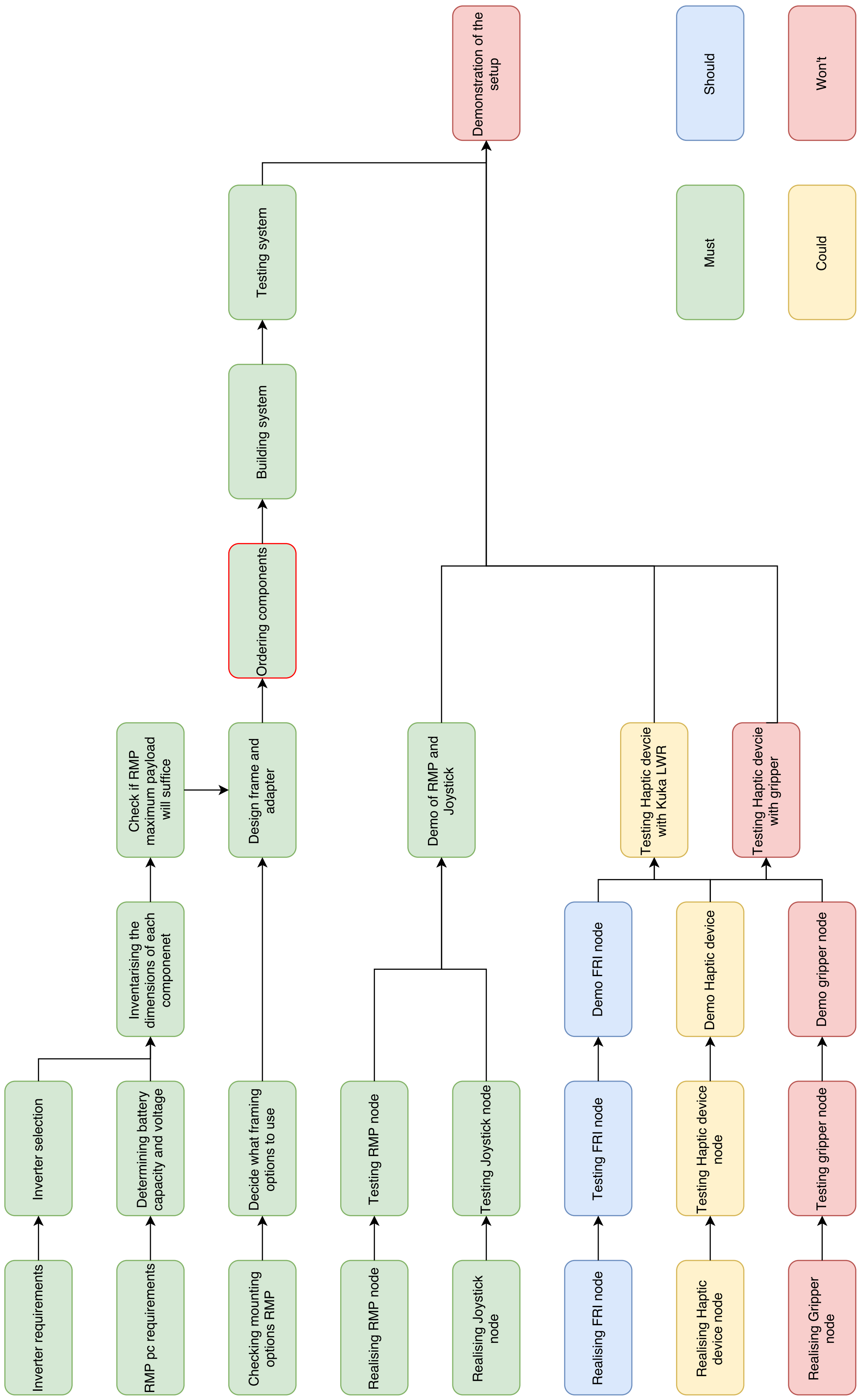

Figure A.1: Task that have to be performed during the project

B Mechanical design

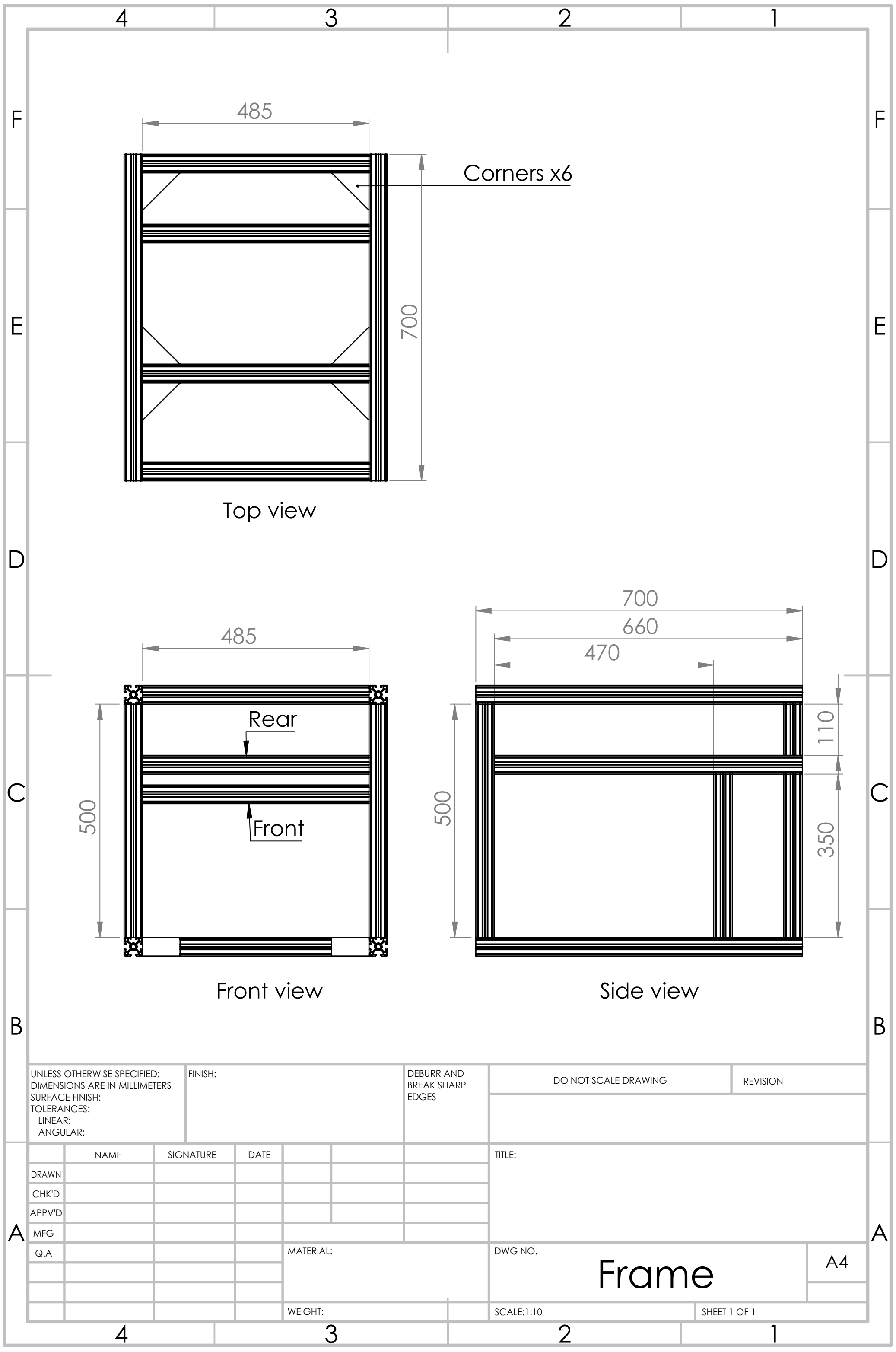

Figure B.1: Detailed drawing of the frame with dimensions

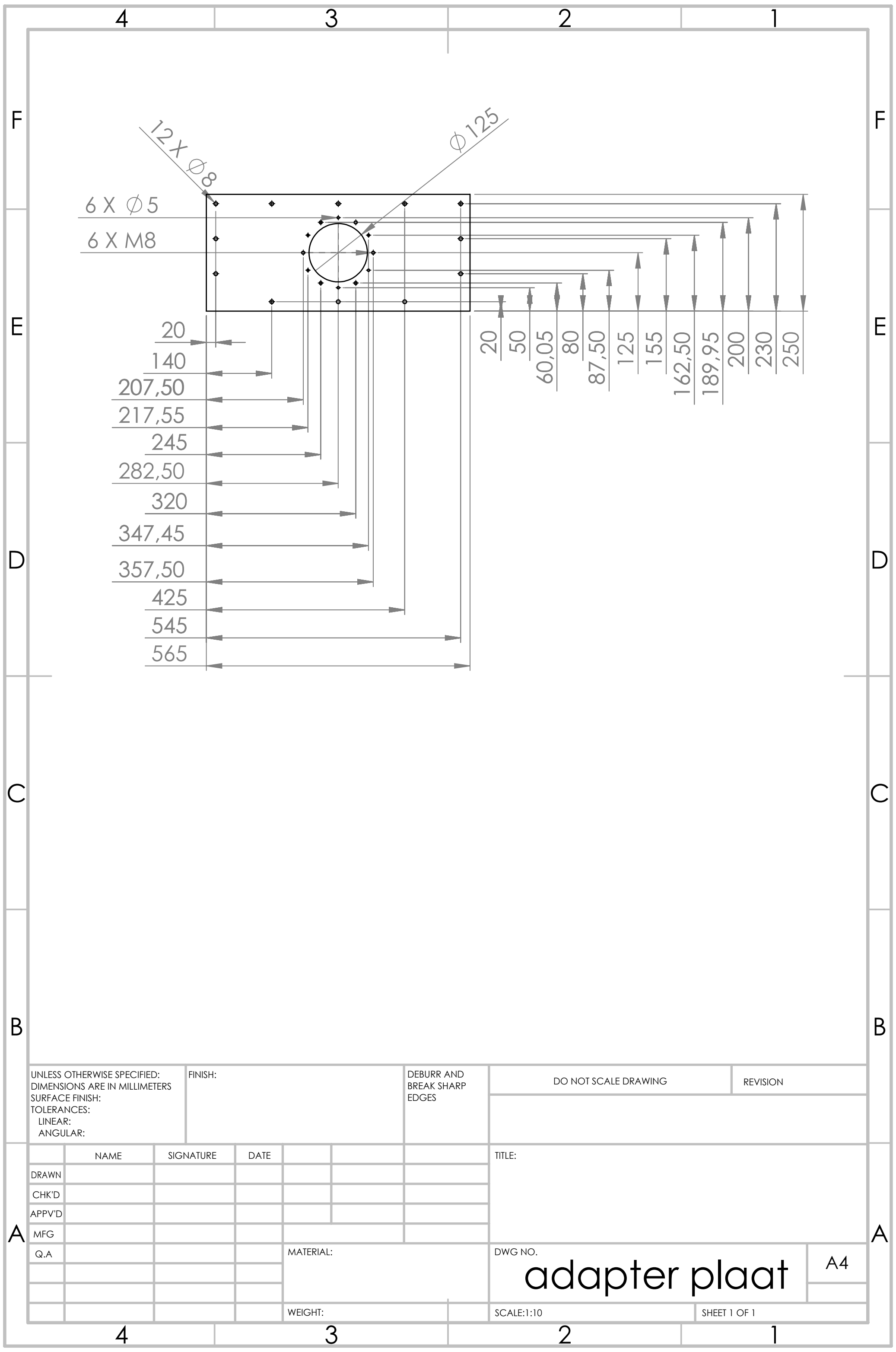

Figure B.2: Detailed drawing of the adapter with dimensions

C Software

C.1 Software on base

It is assumed that the computer has Ubuntu 14.04 with ROS Indigo installed.

Install the following packages in the commandline:

```
sudo apt-get ros-indigo-joy
```

Install the 'conversion_joy' package as follows:
Create a catkin workspace.
Copy the folder 'conversion_joy' into the catkin workspace's src
Run `catkin_make`

Modify /.bashrc:

```
sudo gedit /.bashrc
```

Paste the following under the code:

```
export ROS_MASTER_URI=http://IP_OF_BASE:11311
export ROS_IP=http://IP_OF_BASE
```

Copy the file `base.launch` to the base computer:

C.2 Install software on platform

It is assumed that the computer has Ubuntu 14.04 with ROS Indigo installed.

Install the following packages in the commandline:

```
sudo apt-get ros-indigo-serial
sudo apt-get ros-indigo-libsegwayrmp
sudo apt-get ros-indigo-segway-rmp
```

Install ftd2xx drivers from: `http://www.ftdichip.com/Drivers/D2XX.htm`

Create a file '99-ftdi.rules'

```
touch 99-ftdi.rules
```

Paste the following in the file '99-ftdi.rules':

```
SUBSYSTEM=="usb usb_device", ATTRS{idVendor}=="0403", ATTRS{id
Product}=="e729", GROUP="dialout", MODE="0660"
```

Move the file: `sudo cp 99-ftdi.rules  /etc/udev/rules.d/`

Modify /.bashrc:

```
sudo gedit /.bashrc
```

Paste the following under the code:

```
export ROS_MASTER_URI=http://IP_OF_BASE:11311
export ROS_IP=http://IP_OF_PLATFORM
```

Copy the file `platform.launch` to the platfrom computer.

C.3 Run the software

On the base computer

Run the roscore in a terminal:
```
ros_core
```
Launch the launch file in a new terminal:
```
source ~/YOUR_CATKIN_WS/devel/setup.bash
roslaunch YOUR_DIRECTROY/base.launch
```

On the platform computer

Launch the launch file in a new terminal:
```
source ~/.bashrc
roslaunch YOUR_DIRECTROY/platform.launch
```

Bibliography

Bonnema, G. M., K. T. Veenvliet and J. F. Broenink (2016), *Systems design and engineering: facilitating multidisciplinary development projects*, CRC Press.

Bruyninckx, H., M. Klotzbücher, N. Hochgeschwender, G. Kraetzschmar, L. Gherardi and D. Brugali (2013), The BRICS component model: a model-based development paradigm for complex robotics software systems, in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, pp. 1758–1764.

force dimension (2017), omega.7 specification sheet.
http://www.forcedimension.com/downloads/specs/specsheet-omega.7.pdf

KUKA (2012a), KR C2 lr specifications.
https://wiki.ram.ewi.utwente.nl/images/ramwiki/7/79/KRC2lrSpec.pdf

KUKA (2012b), LWR 4+ assembly manual.
https://wiki.ram.ewi.utwente.nl/images/ramwiki/c/c4/LWR4plusAssembly.pdf

Loughlin, C., A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck and G. Hirzinger (2007), The DLR lightweight robot: design and control concepts for robots in human environments, **vol. 34**, no.5, pp. 376–385.

RightHand robotics (2017), Reflex documentation.
http://docs.righthandrobotics.com/doc:reflex

Segway (2011), Segway RMP user manual.
https://wiki.ram.ewi.utwente.nl/images/ramwiki/d/d5/Segway_Robotic_Platform_UserManual_v1.pdf