Safe movement of robot in a domestic environment

Roman Technologies

Bram van der Veen	4004345
Hans Gaiser	4007891
Wilson Ko	4005686
Ingmar Jager	1511890
Sjors de Wit	1510568
Jelle ten Kate	1510010

21/10/2011

Contents

1 Preface

This report is written as part of the second assignment of the Minor Robotics WB-MI-168-11 at the Technical University of Delft. For this assignment, we were provided a mobile platform that we had to adjust for our application. This assignment lasted for four weeks and is the sequel of the first

We would like to thank our coach J. Broekens and the staff of the Minor for their help and advise during the assignment.

Roman Technologies
Bram van der Veen,
Hans Gaiser,
Ingmar Jager,
Jelle ten Kate,
Sjors de Wit,
Wilson Ko

Delft Technical University,
Delft,
The Netherlands,

2 Introduction

This report will discuss the mobile base of the service robot that we are developing. A service-robot design is developed based on the social and functional qualities a robot is supposed to have. Kinetic models that predict slip and toppling are made based on the robot design. These kinetic models were then implemented in the robot using the programming language ROS to limit and avoid slip and respectively toppling situations. We will also look at some safety measures in the robot design: Obstacle avoidance is done by proximity sensors and a Kinect is used for vision. If those fail the las resort is a bumper that will provide feedback and shock damping through sensors in the frame.

3 Problem statement

It is essential for a service robot that helps elderly at home or care homes to be mobile. Mobile in this context means to be able to move freely across a domestic environment either remote controlled or autonomously. For people to accept a service robot to be moving around freely, general acceptance is another matter of importance. This is the reason that this report also discusses the social interaction and design of the service robot. This document focuses on the safe and social movement of a robot, where the robot will be initially tele-operated.

The prototype robot we are developing will focus on movement in a domestic environment. This is the environment where we expect the most interaction. It needs to operate safely in this environment. This is important because the robot should never harm the people it works with, especially since these people are likely to be elderly people. As so, the robot needs to drive safely in order not to damage the environment. However, it is equally important that the robot itself is safe from the environment, since the robot should not damage itself to operate reliably.

In summary, the problems needed to overcome are:

- Make the robot acceptable for people;
- Make the robot safe from the environment and vice versa;

The robot that we will be developing will be tele-operated. This allows us to focus on the problems to be overcome, instead of automation of the movement.

4 Approach

In order to solve the problems stated in chapter 3, we need to solve the two main problems. Firstly a research is done on what increases the acceptance of a robot, this subsection will investigate the market and our target group. Lastly the movement, safety features and collision prevention for our robot is discussed.

4.1 Acceptance of a robot

This part of the report reviews the motivation for our robot. Why are we building this robot? Why this market?

Market

The amount of elderly is increasing in the Netherlands. This is causing the markets of home care and residential health care to experience considerable growth (Gezondheid en zorg in cijfers, CBS, 2009). Home care is even experiencing extra growth because of the trend of elderly staying longer at home. [1]. This growth, combined with a job offer that cannot keep up with demand and a decrease in government-funding supported the decision for a home care target group. Other public facilities that were considered were rejected because of mechanical or programming constraints or simply because a robot could not make a likely significant difference with the tasks available.

Needs

The need of the clients can be defined as the need for help in their everyday life and/or mobility to provide independence. This was based off the research performed at a Florence health professional conference where professionals were asked for specific tasks that they fulfilled and tasks a servicer-obot could fulfill. More information about the research we performed at this location can be found in report 1. Tasks were supporting and often domestic in nature. Picking up objects, providing meals and opening doors are examples of such tasks. These tasks are in present day performed by a) if possible, the client b) a home care employee or c) a helping dog of type ADL. ADL[2]dogs support clients with their everyday life and/or mobility and reduce the attention needed from a home care employee that is more expensive and less accessible. This improves the independence of the client.[3]

Business opportunities

Several business opportunities other than general assistent (named AUXILI9) were explored. We chose for our general assistent business opportunity (AUXILI9) because of the following reasons.

1. We had limited functionality possibilities. The robot should be technically feasible (determined by team experts), provide a challenge for the team and would preferably use the components provided by the minor. AUWHL did not use those components and made some experts more or less redundant. AUXILI9 fitted the description.

[1] Verpleeg- en verzorgingshuizen per gemeenten 2009, Deuning CM (RIVM), 2009

[2] Algemene Dagelijkse Levensverichtingen

[3] Kosteneffectiviteit van hulphonden, M. Diepenhorst, T.Weijnen & F van Free, 2011, research performed for College van zorgverzekeringen

2. This businessopportunity could be used at more locations. MODULOS was limited to larger centers because it could only function properly using a number of modules that would have to be purchased. AUXILI9 can be used for small groups or even individuals. This is convenient when the trend towards homecare is taken into account.

3. We had mechanical constraints (e.g. the care-robot module of MODULOS could not perform all tasks we wanted it too, because of the limited strength of the mechanical hand). AUXILI9 only needs to fulfill tasks similar to an ADL dog, so strength is less of an issue. [4] Based on this data it was decided to build a servicerobot that will fulfill general domestic tasks (AUXILI9). Tasks were based on the tasks performed by an ADL dog. The ADL dog was mentioned during the Florence research and provided inspiration for this concept.

Figure 4.1: representation of AUXILI9 (left), AUWHL(middle), MODULOS(right). More information about each business opportunity can be found in report 1

Business opportunity - market

ADL dogs provide a limited amount of tasks and with those tasks decrease the amount of attention needed by clients from a home care employee. A limited number of dogs is provided each year, but a growth of distribution is expected. [5] This expectations are related to the increasing popularity and awareness of dogs but also to the earlier mentioned trend towards homecare. The ADL dog is a product that fulfills this needs and the growth figures support the thought that more products are needed.

The entire market for the home care can be defined as customers, but only a small portion of these customers actually purchase a product (e.g. ADL dog) to fulfill the need. It is believed that a similar offset as ADL dogs can be realized, assuming that similar needs are fulfilled. This would make an offset of ca. 60 robots each year. This is far less than the market potential (the total number of home care clients is much higher) of all home care clients, but not all of these clients are in such a need that they would actually purchase such a product. We do believe that the demand will rise. This rise is also seen in other assisting products (such as an robotic arm) than the ADL

[4] report 1

[5] Kosteneffectiviteit van hulphonden, M. Diepenhorst, T.Weijnen & F van Free, 2011, research performed for College van zorgverzekeringen

dog, that can be related to the robot.[6]

Business opportunity - competitors

A porters analysis was performed to analyze the competitive position of the service robot. The full analysis can be found in assignment 1. In this analysis the largest threat was the threat of new entrants on the market: if our servicerobot turns out to be a success, more may be developed to fulfill the need that AUXILI9 will be fulfilling. The high threat of new arrivals is caused by amongst others accessible software (ROS) and distribution channels. Software cannot be harder to access and it is also difficult to make the distribution channels more accessible as there are no dealers except for the zorginkopers. By making close relationships to them new entrants may be prevented, but differentiating our product may be a better method to avoid new entrants. A well known or recognisable product has a better competitive position than new products and this competition may scare off potential new arrivals.

Figure 4.2: Visual of Porters analysis of competitors, centered on servicerobot business opportunity

The main contemporary competitors for AUXILI0 are the home care employee and the ADL dog. A quick comparison of the ADL dog and home care employee with AUXILI9 is presented in Fig 4.3 The robotic hands of RTD, Exact Dynamics, Kersten Revalidatietechniek and Focal Revalidatietechniek are not seen as direct competitors since they provide different functions. (They provide more specific support for hand, arm and fingers issues.)[7]

Concluding about the service-robot requirements based on market and competition research

The service-robot we will be developing should thus be: a visually (from other potential servicerobots, potential new arrivals) and functionally (from humans and ADL dogs) differentiating product to create a strong competitive position.

Differentiating

To functionally differentiate from humans we want to provide a much more costefficient and accessible method of homecare. Though robots cannot handle full responsibility (Florence research confirmed that people also do not like the idea of robots carring responsibility), they can be present around the clock: replacing a large part of the work a human performs. An ADL dog functions

[6]Overheveling van honden en Robotmanipulatoren, L. Plas, B. Noordhuizen &, F. van Vree, 2008, researched performed for College van zorgverzekeringen

[7]Overheveling van honden en Robotmanipulatoren, L. Plas, B. Noordhuizen &, F. van Vree, 2008, researched performed for College van zorgverzekeringen

ADL Dog	Human
+ living being that can give personal attention + typically can perform around 80 tasks to aid client, but can be trained to perform more. + dogs can effectively support the same client for 6-10 years, 24/7. + ADL dogs provide a cost-effective strategy to provide more attention to a client. - unsuitable for people that have allergies - unsuitable for people that do not like animals - requires maintanance (costs!) - requires time for training (also for person) - accessibility is becoming less because insurance will likely not cover ADL dogs from 2012. (Haagse PVDA wil behoud hulphonden in zorgpakket, Omroep West, 2011) Currently ADL dogs are often not fully covered.	+ living being that can give personal attention + employee can perform more tasks than the client can and can fully support the client. + employee can carry responsibility and can check up on client. - health care employees have limited accessibility because of the number of clients. - health care employees are expensive

Figure 4.3: Brainstorm and Florence-based comparison of ADL dogs and home care employees. + indicates pros of the product, - indicates cons of the product.

the same way.[8]

We can functionally differentiate from the ADL dog by providing more electronic or digital options that only a robot can provide. We have opted for communication, since home care is also using communication at distance (seen at Florence) and this would allow the robot to function as a warning system in case of danger (or e.g. falling). This would also provide a more accessible means of contact to others as the phone will come to you. The robot should also have social functionality in order for it to interact properly as dogs and humans do, this is vital for the acceptance of the robots [9]. Differentiating visually is implemented in the design drawing part of this report.

Demands and wishes

A program of demands and wishes was made in assignment 1. This list was based on the functions a helper dog would fulfill. This list is revamped in appendix 6.1, presenting a renewed list of functions our product is required to perform. The function of retrieving has been elaborated in more detail. This is done because this function will be the focus of the prototype. Other functions are mentioned to provide a picture of the total product.

The motivation and a draft of demands to build a specific robot have now been determined. The next part will focus on the actual design of the robot. This will be a simplified step-by-step showcase of how the robot was designed.

[8]Kosteneffectiviteit van hulphonden, M. Diepenhorst, T.Weijnen & F van Free, 2011, research performed for College van zorgverzekeringen

[9]The Influence of Social Presence on Acceptance of a Companion Robot by Older People, Marcel Heerink, Ben Krse, Vanessa Evers & Bob Wielinga, 2009

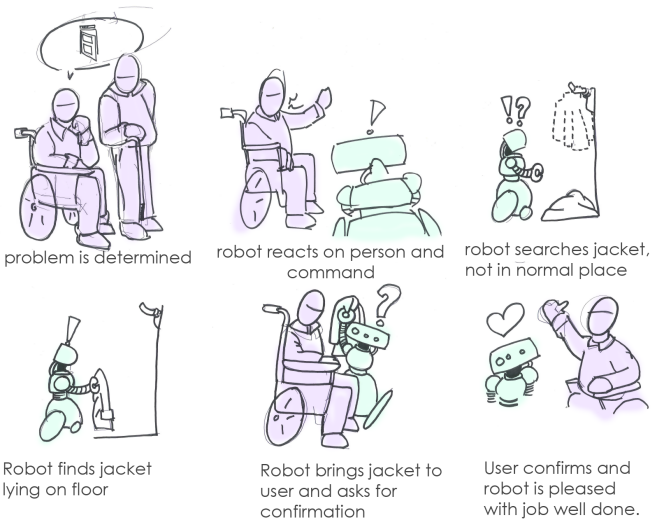


Figure 4.4: Use scenario. This shows how the robot can interact socially by being confused (when something is not there), confused (when not sure if it is the right product) and pleased (when the client confirms that it is the right product). It also shows the task for our prototype: the prototype should be able to retrieve a product and give it to the correct person. (Even though there are two persons there, the robot has to know which one has to get the jacket.)

4.2 Conceptual design

The next part of the design process is the concept design. The main functions of the robot were set in a morfological chart. Several options/parts were placed to fulfill those functions and three combinations were chosen to evaluate.

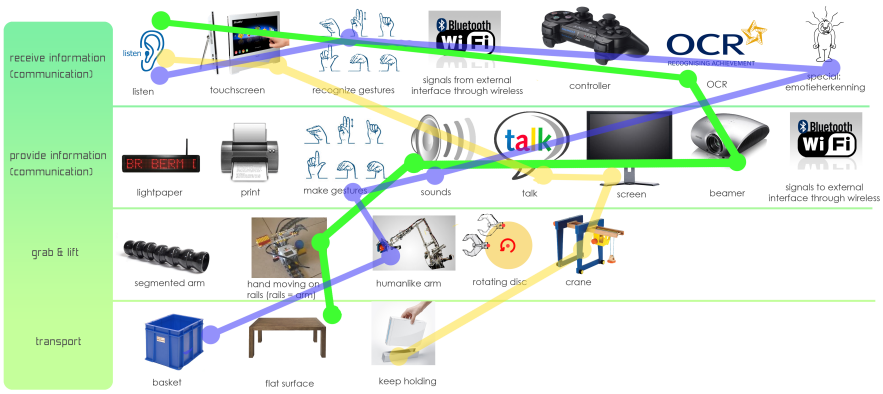


Figure 4.5: Morfological chart. Lines illustrate thre three combinations of features that can result in a producttype. Appendix 6.4 eloborates on this subject.

Fig 4.6 Shows the visualisation a visualisation of the combinations of the morfological chart. The Hammond concept was continued. This concept is represented by the blue lines in the morfological chart. It is a servicerobot that could communicate and react using emotion. This concept uses communication by sounds instead of language. This allowed for a general way of reacting (e.g. confirming, confused, displeased, neutral), that could easily react on a large amount of situations.

It was also assumed that the robot would be more likeable this way, since it seems it is more emotive than the preprogrammed language speaking robot. The robotic arm provides a versatile manipulator that does not purtrude to the backside (such as the rail concept of Clarkson). It can grasp further than the container-inspired manipulator (hanging from a rope) that is presented in the May concept. More sketches were made on the Hammond concept to investigate other compositions.

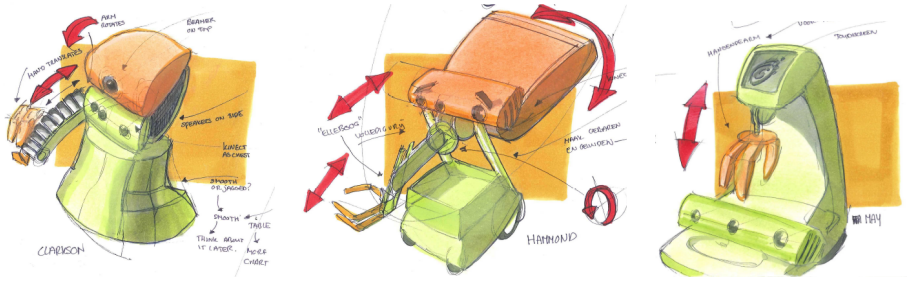


Figure 4.6: . Visualisation of morfological chart combinations. (Clarkson(green lines), Hammond(blue lines), May(yellow lines). Arrows visualize movement directions for parts. These arrows were made to index the possibilities of the robots as sketched, they do not show the actual movement limitations of the concept.

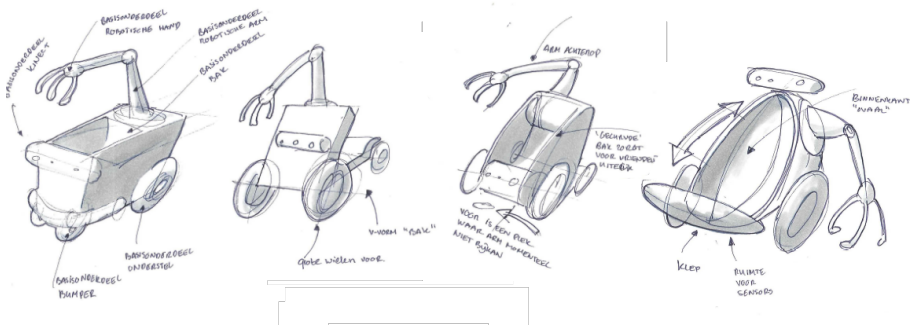


Figure 4.7: Conceptual design sketching, based on function research. Appendix 4 contains more sketches. The sketches illustrate the variations that can be made while still using the same parts that were chosen in the morfological chart.

The first part of the conceptual design is now finished. The second part will focus on form-integration and emotion. An analysis of robotdesign was performed to create a stereotype for a robot. We will differentiate from this reference to create a recognisable and unique product, but we will keep the robot somewhat similar to the stereotype to prevent designing an estranging product. An estranging product would prevent the desired interaction. Social functionalities are implemented to encourage usage and interacting with the robot, these are discussed following the discussion of the stereotype.

The stereotype robot can be described as a anthromorphic white-gray-black colored robot that has geometric jagged shapes because of function and organic smooth shapes if possible. (A part of the body is a block, because the engine part beneath it is a block). Colors specify important (e.g. moving) parts of the robot and can be seen as usecues. If the white color is not dominant, the robot is probably designed for children or has a more specific application. This specific application would make it less of a robot and more of e.g. a vacuum cleaner. ASIMO is a robot that is quite similar to the stereotype robot. The general shape that is present in most robots is humanoid, making them anthromorphic. The supporting analysis can be found in Appendix 3

Figure 4.8: ASIMO and PR2, showing recognisable elements of stereotype robot

Special attention has been paid to the social functionalities of the robot. The robot will not initiate any movement (indication that this was important was found at Florence) that result in the robot coming into the personal space (1m from robot to person). In a hallway this will result in interaction rules as described in Embodied social interaction for robots by H I Christensen & E Pacchierotti: the robot will try to avoid people when moving towards eachother, moving to the right if it is not certain (e.g. by clutter) that an individual can be avoided. The robot will then wait till the individual has moved past the robot and will then resume movement. A simple face (by inserting eyebrows) is also implemented in the design to improve the ability to communicate with people [10]. This face combined with body language (turning towards the communication partner) will allow the robot to confirm and maintain contact. By looking at the individual, the robot will encourage the person to interact [11]. The last part of the social part of the robot will be an emotive system. Our robot communicates using not language, but moods and emotions. The moods we selected are: neutral, pleased, confused/uncertain, displeased.These moods are expressed using sounds, movement of eyebrows and colors. A combination was pursued because our target group may suffer from limited speech or hearing. We did not pursue realistic human aspects of this robot, as this will probably make people feel uncomfortable. This is called the uncanny valley effect.[12], chapter 6.6 in Appendix A will explain this phenomenon briefly.

Final design

The final conceptual design is a social service robot. It can perform a multitude of tasks as described in the Program of demands and wishes. It can grab and manipulate objects using its arm and store those objects on the flap that is located on the lower part of the torso. This flap has to be opened to serve as a box for transport. The screen on the top part of its chest is a removable touchscreen that serves as a back-up input method, for when users are unfamiliar with the voice-commands or if there are problems with the voice-recognition. The bumper at the front and back protects the robot and its environment from harm. This will be discussed in more elaborate in chapter 4.3. It interacts using four emotional states or moods that change the color of its head (currently yellow) and the setting of the eyebrows. A sound will be made to accompany these mood changes to provide maximum input for the user. The robot will always try to point towards the person

[10]Interaction with Mobile Robots in Public places by S Thrun, J Schulte and C Rosenberg, 2000

[11]the role of expressiveness and attention in human-robot interaction by A. Bruce, I Nourbakhsh and R Simmons, 2002

[12]The Uncanny Valley by M. Mori, 1970

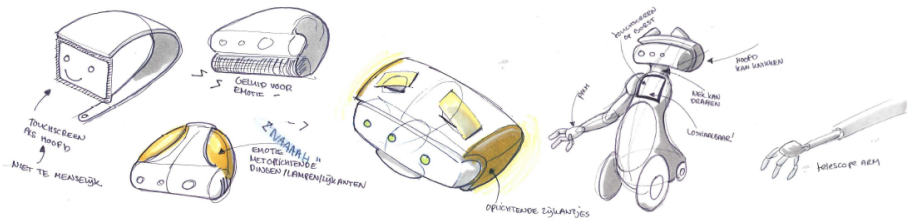


Figure 4.9: Conceptual design sketching, based on additional input from stereotype design and social aspects. Appendix 4 contains more sketches. Sketches are shown to illustrate the variations that can be thought of when trying to communicate, and when trying to integrate this into a complete robot.

communicating to stimulate interaction, but will keep its distance when moving towards each other.

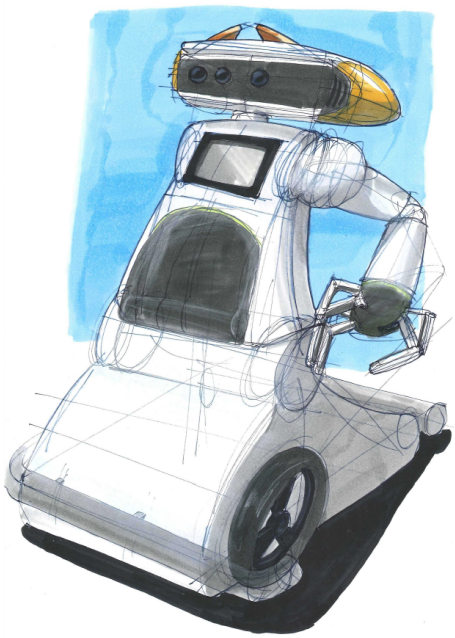


Figure 4.10: Final Conceptual design

Our prototype

From the final conceptual design we will only implement a limited amount of functions in our prototype. This is because of time and knowledge constraints: we are not able to do everything (even things we do not know of) within a limited amount of time.

1. The general design. We want to show how our robot should function, its design is a crucial part of its functioning.
2. The basic grabbing functions. The mood functions (neutral, pleased, confused/uncertain, displeased) Fig 4.4 (User scenario) shows the functions that are prototype should be able to perform. A more specific list is presented:
    a Understand command // (not by speech recog, but by controller)
    b Determine where is object // (Preprogrammed knowledge e.g. apples are always in

kitchen)

c Move towards object and open door // (Only push open door)

d Recognize object // (1 object from three different objects)

e Grab object // (grab correct object)

f Return object to correct person // (Return to person, recognize person by face recog or simplified elements, then present object to person)

g React doubtful, then react pleased if person is pleased OR react sad if person is not pleased.

4.3 Safety from the environment

The second problem we are trying to solve is how to make a mobile base move safely in a domestic environment. This part of the chapter will discuss how the robot currently moves, how we made it move safely, how it detects possible obstacles and how it keeps itself safe from the environment.

4.3.1 Mobile platform

A front wheel driven mobile platform from the minor was provided to use for the robot's movements. This base uses two motors to control each of the two front wheels and one motor to control the counter weight of the (to be built) arm.

Figure 4.11: Mobile base equipped with self built bumper

4.3.2 Teleoperation

To test the bases acceleration, braking and safety features (which will be discussed in chapter 4.3) we need the robot to move. Since the mobile base is not yet autonomous, we need some alternative way to move the base. A PS3 Controller is used to control the robot. There are two different modes to steer the robot, game control mode (as the controls are the same as in racing games) and remote control mode, see 4.12 and 4.13 .

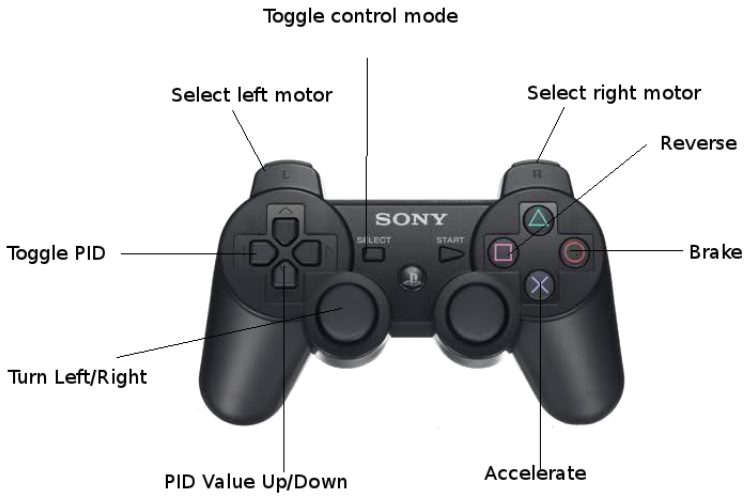


Figure 4.12: Button mapping of remote control mode

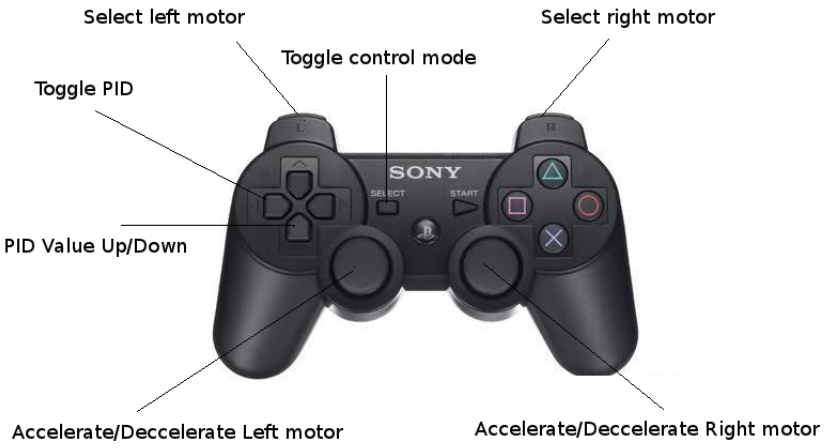


Figure 4.13: Button mapping of remote control mode

4.3.3 Safe maneuvering

To avoid the robot harming itself and its environment, it is important to consider the stability, ccornering and breaking specifics of the robot. If the robot suddenly brakes or takes very sharp corners, it will tend to topple. Toppling is the most dangerous situation that can occur. In that situation the full weight of the robot may fall over a person. This is absolutely unacceptable behaviour and should be avoided at all costs. Another potentially dangerous situation is unadequate braking, causing the robot to crash into its surrounding environment. To avoid this, the grip of the wheels on the floor has to be maximized. The wheels should always be in contact with the floor with enough force to avoid slipping.

The first part of this paragraph will discuss these two key topics in preventive safety: avoiding toppling and maximizing grip. In combination with the safety measures as described in chapter 4.4, this should make the robot capable of preventing collisions. However, if for some reason a collision cannot be avoided, it is desired to minimize impact. Minimizing collision impact wil be discussed in chapter 4.3.

4.3.3.1 Toppling and slip requirements

Combining the grip and toppling requirements results in a set of constraints on the movement of the robot. The goal is to implement these constraints into the robot. This can be done by defining a limit to the acceleration of the robot, together with a maximum angular velocity.

The allowable acceleration is wished to be as high as possible, as this will ensure the robot is able to stop quickly in case of emergency. The maximum angular velocity is also desired to be high, as it enables the robot to quickly avoid collisions. However, the robot should meet both requirements while turning. If the robot already needs all of its grip for staying on its curved path, no grip is left for braking. This means there is a tradeoff between the maximum allowed turning speed and the maximum acceleration: when quick acceleration is wished, fast turning cannot be allowed.

For toppling this tradeoff is not important, as the robot will topple either to the side or to the front, but never over one of its corners.[13] This is illustrated in figure 4.14.

To investigate the correlation between the maximum acceleration α and the maximum angular speed ω, the toppling and slip formulas[14] have been rewritten to be only dependent on α and ω. A matlab script has been created to automatially solve this system. The script tries to increase α and ω until one of the constraints fails.[15] Repeating this for a different height of the center of mass[16] gives a good overview of the relation between α and ω.

Figure 4.15 clearly shows the relation between α and ω. The lowest graph consists of mostly straight lines. This graph represents the maneuvering limits for a mass center at two meters height. This is not realistic, but shows how this height influences the limits. The straight lines mean that the toppling limit has reached, thus no faster acceleration or angular speed is allowed. The very circular graph is the situation for a mass center at height 0. That graph is circular because the

[13] This is because the distance from the center of mass is always closer to one of its sides than to its corners

[14] The toppling and slip formulas are formulas **[REF to first formula]** to **[REF to last formula]** in the appendix

[15] For example, one of the reaction forces is calculated to be negative, indicating the robot would have toppled

[16] See section 7.1.1 for more information about the center of mass

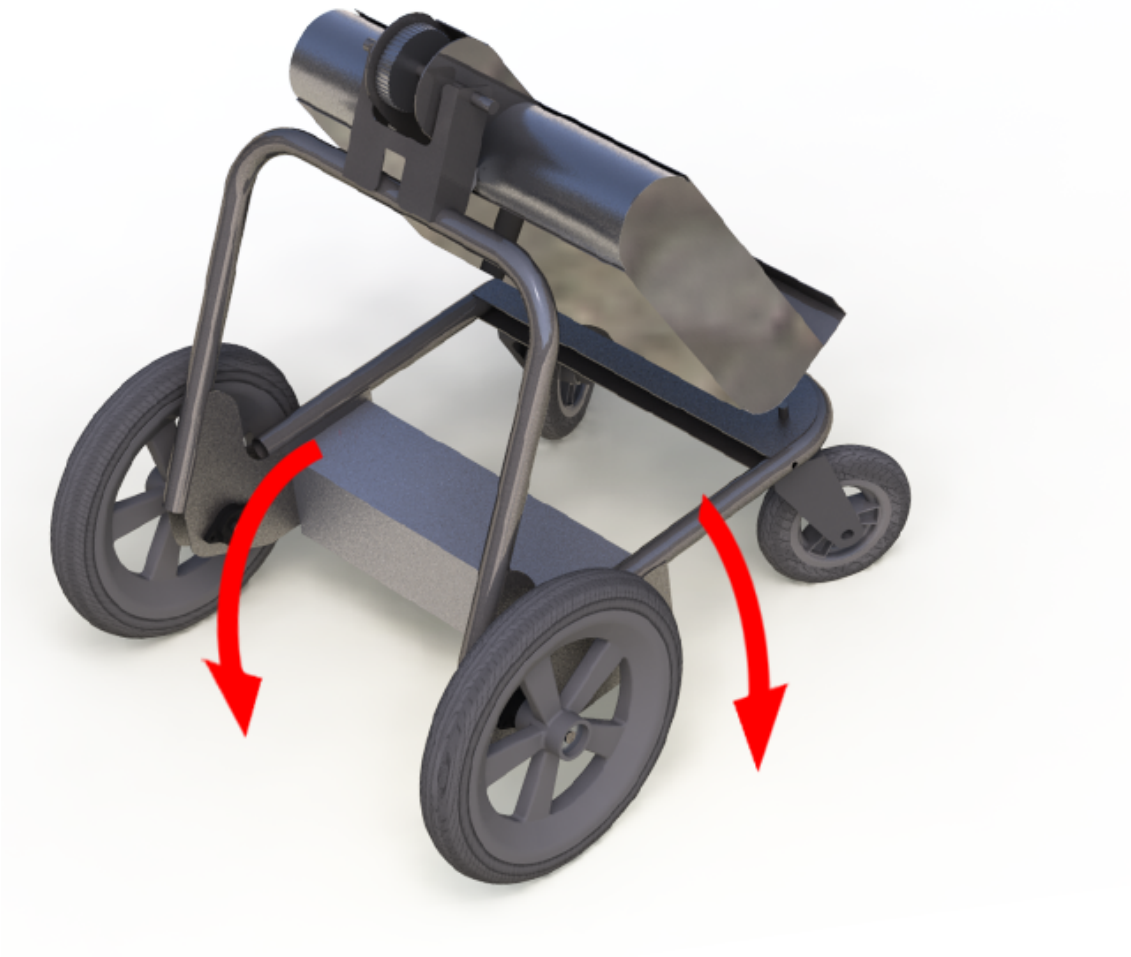


Figure 4.14: The robot tends to topple to the sides or to the front, but never over the corner

total friction is the square root of the linear and angular friction. All lines in between show a tradeoff between slip and toppling.

4.3.4 Friction measurements

As described in section 7.1.2, the maximum wheel friction is needed to calculate if the robot has enough grip. This friction was measured by slowly pulling the robot sideways. The required

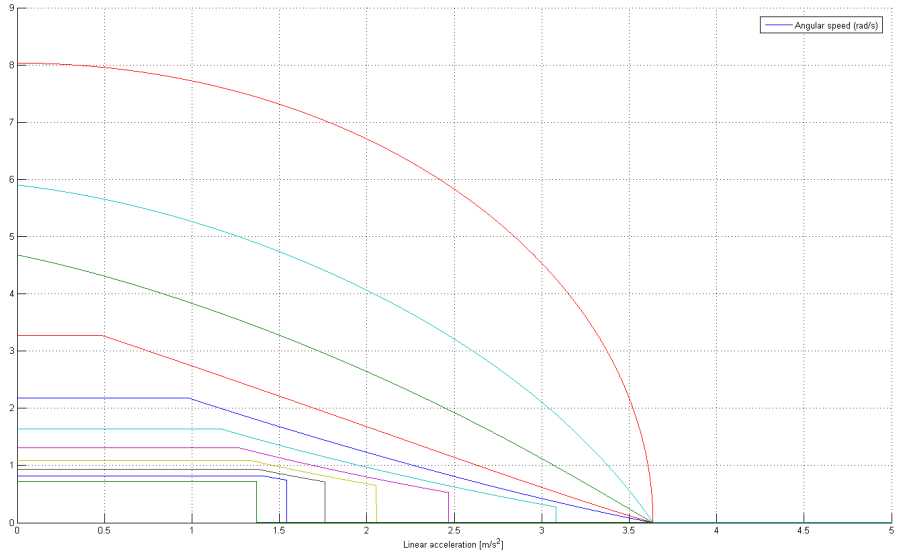


Figure 4.15: Maximum angular velocity versus maximum acceleration for different positions of the mass center

force was measured repeatedly on different surfaces using a newton-meter. [17] The lowest measured value was 44N. This lowest measured friction was taken since the least friction is the worst-case scenario.[18] The measured friction mentioned before is the total friction. As the robot was in equilibrium (e.g. not accelerating or moving), the friction is assumed to be evenly distributed between the front wheels. The back wheels are assumed to operate frictionless, as they can move freely to any orientation. This means they do not contribute in keeping the robot on track.

Apart from the sliding friction, it is also important to know the rolling resistance. This is the force that is required to move the robot with constant speed. This was determined in the same way as the sliding friction described above: slowly pulling the vehicle and noting the required force when it starts to move. In this case the maximum value was taken, as a higher friction may lead to extra deceleration. These measurements resulted in a maximum rolling resistance of 12.3N.

4.4 Detection

The robot should be able to drive around autonomously while avoiding obstacles. To do this it is required for the robot to have a sense of the surrounding objects. This chapter will discuss what kind of sensors we selected and why these were chosen.

[17] a newton meter essentially is a spring with a known force constant, used to measure force

[18] The least friction is the worst-case scenario because this leads to the least grip

4.4.1 Sensor selection

To choose our sensors, we decided on a list of requirements. These requirements are listed below.

- The sensor should work within 60cm. Any objects beyond this distance can be covered by the Kinect.
- The total costs should fall within our budget.
- The sensors should have a beam width of at least 30°. This ensures that we need a reasonable amount(considering our limited budget) of sensors to cover the entire base.

Because of our positive experience with the UltraSonic Parallax Ping sensor used in our gripper (details can be found in report 1) we decided to go with ultrasonic sensors again. The prices for these ultrasonic sensors vary quite a bit, from €31,89 to €12,49 each[19]. Because we needed at least six sensors to cover the base, considering a beam width of 30°, the lower price was very attractive. This cheap sensor, the one that costs €12,49, fits every requirement we had set. This sensor is called SRF02.

According to the datasheet[20], these sensors can detect objects from 16 centimeters at an angle of 40° to both sides. However, testing showed that the beam width neared an angle of only $30 - 40°$ in total. This forced us to put the sensors closer together than initially planned, resulting in the layout as shown in figure 4.16. This layout ensures coverage of the whole front end of the mobile base. On the rear side the left and right sensors are aimed in the direction of the swiveling of the castors to prevent hitting objects while making turns. The sides of the robot are not covered. Future testing has to point out if it is necessary to place additional sensors on the sides.

Details on the connection of these sensors can be found in appendix 10.1.

[19]Sensor reseller: http://www.antratek.nl/Ultrasonic-sensors.html

[20]Datasheet: http://www.robot-electronics.co.uk/htm/srf02tech.htm

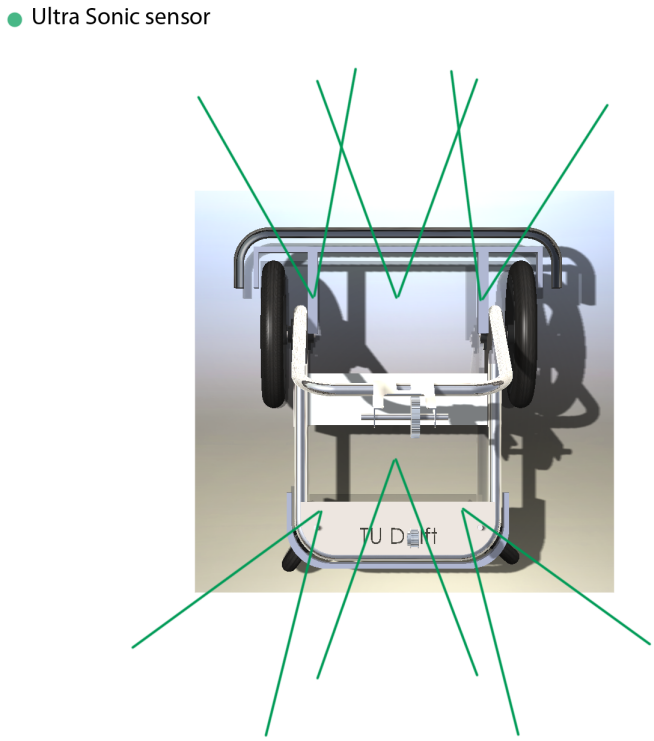


Figure 4.16: Ultra sonic sensor beams on the mobile base

4.4.2 Vision

These ultrasonic sensors alone cannot detect everything. We picked them because of their higher beam width, but they still cannot cover the entire robot. These ultrasonic sensors were placed so that they cover the lower part of the robot, but since it is unfeasable to have these sensors all over the robot, we choose to install a vision system. This was a pretty straightforward decision, since we already needed a visual system for object detection at a later stadium (where the robot would drive completely autonomously). The Microsoft Kinect[21] provides not only a regular camera image, but it also provides a point cloud[22]. This point cloud is often used in object detection. There is a library developed purely around this type of data, the Point Cloud Library[23](PCL). The PCL provides us with a set of tools we would have to develop ourselfs otherwise. Besides this advantage, experience with the Kinect was already available to us. A final advantage of the Kinect over other camera systems, even ones that produce point clouds, is that ROS has a package specially for this Kinect, making it easy to implement the kinect into the system.

To detect obstacles, the Kinect has to differentiate the ground from obstacles. Otherwise it will

[21]http://en.wikipedia.org/wiki/Kinect, consulted on 21 october 2011

[22]http://en.wikipedia.org/wiki/Point_cloud, consulted on 21 october 2011

[23]http://pointclouds.org/

recognize the ground as an object. To accomplish this goal we used the previously mentioned Point Cloud Library. First of all, only those points are selected that the robot would come in contact with if it would move forward in a straight line. These points are colored blue in figure 4.17.

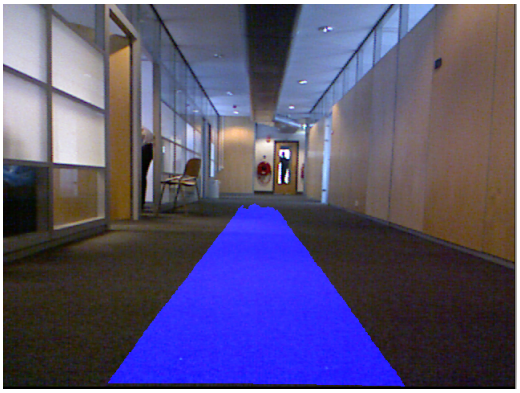


Figure 4.17: Shows contact points if robot would move in a straight line ahead

Then an algorithm of the Point Cloud library is used to generate what is called a least square[24] plane with these points. This plane is used as an estimation of the ground the robot is moving along. To detect objects that are not a part of the ground, points are tested against this plane to see if their distance is above a certain tolerance level. If this is the case, the point is considered obstacle, as can be seen in figure 4.18 with the color red.

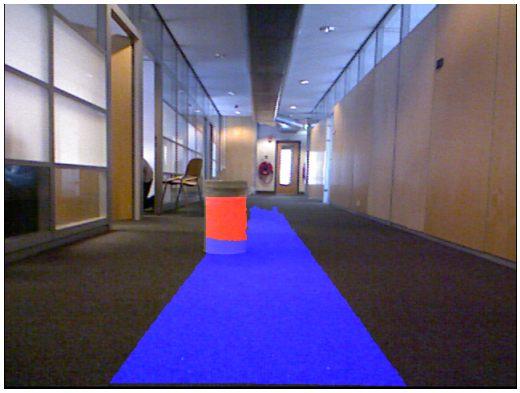


Figure 4.18: Shows contact points if robot would move in a straight line ahead with an obstacle

These red points can be grouped together by distance to eachother, forming actual obstacles. This way small errors of just a few points can be filtered out. The remaining obstacles are checked for their distance to the robot and it can then be determined whether the robot can continue its path, slow down, divert or even perform an emergency stop. More details about this visual obstacle detection can be found in appendix 10.2.

[24] http://en.wikipedia.org/wiki/Least_squares, consulted on 21 october 2011

4.4.3 Bumper

When the obstacle avoidance described in 4.4 fails we need some kind of backup. To counter any unexpected actions we designed a bumper. The bumper will absorb forces caused by an impact as well as detect where the impact happened. To design a good bumper, we have to analyse in which directions the robot can manoeuvre.

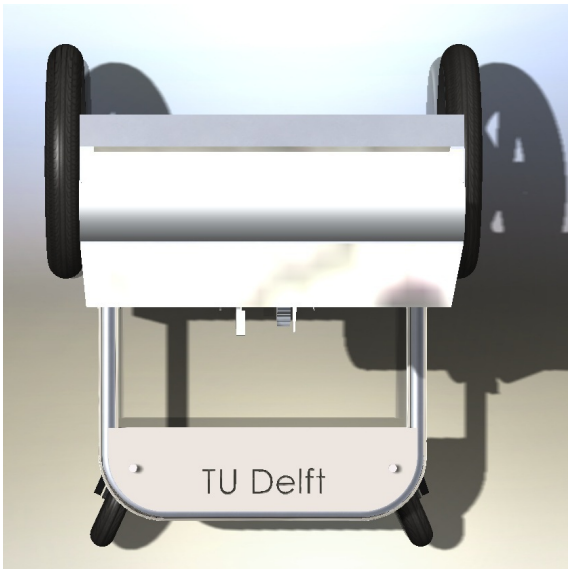


Figure 4.19: The robot without bumpers

First, the force that the bumper has to absorb has to be determined. In a worst case scenario the robot will hit a human. The bumper is designed so, that when the robot hits a human leg, it doesnt hurt the person. A definition of the maximum allowable pain limit of a person is called the pain pressure threshold (PPT). We used for the maximum PPT, 150 kPa. This is based on the literature overview of tests[25]. In this literature overview 150 kPa is the maximum PPT on a human head. The head is more vulnerable than a leg, so we used this PPT for safety features. If the bumper hits a human leg the contact area is proximally 30 cm2. With the contact area and the maximum PPT we can determine the design constraint, which shows the static contact force:

$$F_{StaticContact} \leq 150[kPa] * 30[cm^2]$$

So the static contact force is 450 N. With this force we calculated the thickness of the material which absorbs the impact. We assumed that the total weight of the robot is 30 kg. The maximum speed is based on the specifications of the DC-motors[26]. This is the no load speed, but the real maximum speed is lower. We measured this, which is 1.1 m/s.

[25]Jensen K, Andersen HO, Olesen J, Lindblom U (1986) Pressurepain threshold in human temporal region: evaluation of a new pressure algometer. Pain25(3):313-323

[26]Bhler Motor, DC Gear motor 1.61.050.XXX

$$0.5 \cdot m \cdot v^2 = F \cdot s$$
$$s = (0.5 \cdot 30[kg] \cdot 1.1^2[m/s])/450[N]$$
$$s = 0.04033[m]$$

The bumper should be able to be displaced 40.33 mm(and bounce back). To absorb the forces on the bumper, we had to decide which material to use. Some examples of applicable materials are: foam, rubber, insulation material and Styrofoam. We also had to see which materials are well available for us. So we made the decision to choose insulation material. It is a material which absorbs an impact really good and it is well available for us. We used to layers of insulation material, so the total thickness can be used to absorb the contact force at maximum speed.

4.4.4 Bumper design

We decided to make the bumpers as compact as possible. This means that it is designed as small as possible and as easy as possible to build. This causes a dilemma for the front bumper. We first designed a bumper which absorbed the forces from the side and from the front. But the bumper became too complex and too large. The bumper existed of too many parts, which were difficult to build and the robot could not fit easily trough a doorway, because the robot width became 70 cm and the width of a doorway is 93 cm. For safety features we decided to maintain a width of 60 cm.

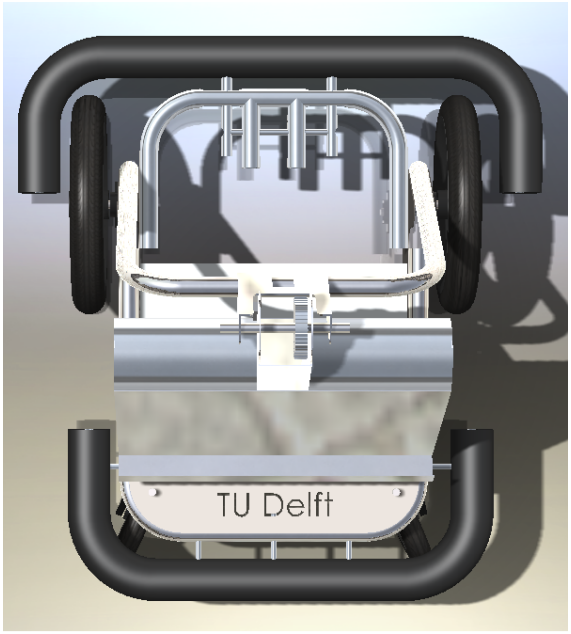
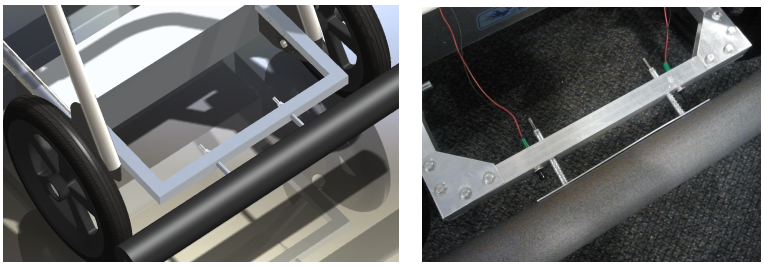


Figure 4.20: The robot without bumpers

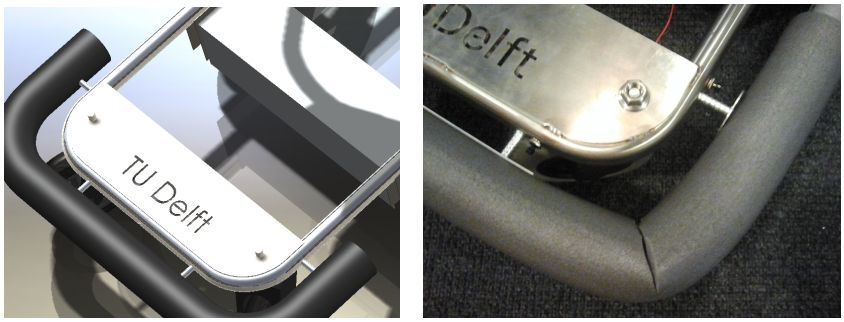
So we designed a much simpler bumper, which only absorbed the forces of the front. To protect the wheels from object sideways, there will be mounted a cap on the final robot. This is the final design of the front bumper:



(a) drawing of the front bumper (b) photo of the front bumper

Figure 4.21: The front bumper

The bumper at the back is designed in such a way that it will absorb the forces from the back and from the side. Also, six buttons are placed in the bumper. Those are able to detect the location of impact. Technical aspects of the front- and back bumper and button schematics are found in chapter 11



(a) drawing of the back bumper (b) photo of the back bumper

Figure 4.22: The back bumper

5 Conclusion

From the research in this report we can conclude that we can make a robot acceptable by implementing social communication and appearance features. Dynamic features are also important, as the robot makes several movements (e.g. for making eye contact) in order to socially succeed. Dynamics are not limited to social acting as the robot should be able to perform, in a safe way for itself and the environment, all of its intended functionalities. Software steered kinetic models are needed to provide such moment, limiting speed or corner-speed to prevent toppling and slip, while preserving a certain amount of acceleration. A sensor system can supplement these models to account for walls and objects that are in vicinity of the robots. However even these models cannot account for everything. As such it is necessary to build safety measures such as bumpers in a robot. We can conclude that many disciplines are needed to build a robot that can safely act in a environment with humans and that safety is of great importance in this environment.

6 Appendix A

6.1 Program of demands and wishes

This is the program of demands and wishes for our product. Elements that are necessary for the prototype are explained more elaborate.

1. Interaction with person
    a. Robot is commanded using speech or direct input
    b. Robot follows commands from owner
        i. Fetch object[1] for person[2]
            1. Robot autonomously finds object
            2. Robot autonomously picks up object
                a. Object can be picked up from surfaces ranging from heights of 0m (floor) to 1.572m (elbow height P99 male 60y+[3])
                b. Object can exert a maximum of 20N downwards (Appendix 2).
            3. Robot autonomously carries object back to designated person
            (person can be person who asked for object, but not per se)
            OR place object on designated place.
        ii. Follow person
        iii. Open/close door
        iv. Open/close window
        v. Open/close drawer
        vi. Turn on/off lights
        vii. Turn on/off media such as: tv, radio, computer.
        viii. Go to location
        ix. Stay and wait
    c. Robot can react to person
        i. Robot will signal for enlightening if speech/command was not understood (emotive, social)
        ii. Robot will respond to emergency situations by asking person and alerting professionals if necessary[4]
            1. Person falling
            2. Person lying on floor not moving and responding
        iii. Robot seeks eye-sight when communicating (social)
        iv. Robot will face person when communicating (social)
        v. Robot can communicate using colors, speech and/or 'body movements'.
    d. Robot can provide means of communicating with others
    e. Robot will not initiate movement that will result in a distance from robot to person that is lower than 1m.

2. Environment
    a. Robot can move at speeds up to 1.46m/s.[5]
        i. Robot will act according to interaction rules as described in 'Embodiment design'
        ii. Pain caused by robot should stay under the PPT of a head in a colission.
        iii. Robot should minimize slip
        iv. Robot does not topple
        v. Robot brakes with the max distance between person and robot to be 0.3m
    b. Robot can move in a domestic environment[6]
        i. Robot can move over carpet, laminate
        ii. Robot can move over thresholds
        iii. Robot can move (push, pull or lift) objects if neccessary to move
        iv. Robot can move through doors (assumed width is 90 cm)
    c. Robot can move outside of the domestic environment.[7]
        i. Robot can move over pavement
        ii. Robot can move over kerb
        iii. Robot can move over grass
    d. Robot can limited maintain environment
        i. Robot can close windows/doors when owner is not present (if not ASKED NOT to)
        ii. Robot can maintain plants (if ASKED to)

3. Independence
    a. The robot should be able to recharge itself
        i. The robot should be able to recharge itself through general household sockets.
    b. The robot should signal when 'damaged'
        i. The robot should communicate damage to owner
        ii. Robot can ask for maintenance from supplier WHEN ALLOWED.

[1] Objects may be personal and unique (such as a jacket), general (a tennis ball) or somewhere in between (an apple, that can vary in shape).
[2] Persons should be recognized if met before.
[3] DINED data, 'Dutch adult, dined 2004, female and males 60+', consulted on 18 okt 2011
[4] Professionals may be homecare or 112. This is done when owner does not respond or asks for this. This is an aspect of the robot that involves legal issues. These have not been explored.
[5] Maximum comfortable walking speed for fourty-year old men (Oxford Comfortable and maximum walking speed of adults aged 20—79 years: reference values and determinants, Richard W. Bohannon, 1997). Ensures that robots can follow their owner (that are of later age and decreased walking speed).
[6] It can be expected that the owner will make their home suitable for the size of the domestic robot. Stairs will not be within the scope of this robot, since there are already parts given by the minor that cannot cross stairs.
[7] Robot should also be able to move outside if needed by their owner.

The ViP (Report 1) included the design demands to have the robot make the environment feel more safe and sustainable, while interacting in a personal familial way. The social and emotive functions were implemented to provide a personal way of interacting. The arlarm-setting and general control of the robot are elements that should make the environment more safe. The self-sustaining functions (recharge itself, close windows ...) were built to make the environment more sustainable.

6.2 Force of gripper

Forces in the thumb are 20N + 18N = 38 N and the forces in finger one and finger two are total 38 N. So the total force on an object is 76 N. The force from the object on the gripper depends on the friction coefficient between the object and the gripper. To determine the maxiumum force (and weight) that the gripper can hold the coefficient of friction has to be used. We assumed that the coefficient of friction will go up to 0.5. So the gripper can hold up to 38 N. This is presents a safety feature, because it limits the hand from exerting its maximum force that would limit the life-time of the product. Data was derived from experiments performed in report 1.

6.3 Shape Analysis

The shape analysis is made using the elements shape, texture, composition and space. These elements together have been described to provide a stereotype robot. A collage was made to provide input for the analysis. A sample of consumer robots has been used. These robots interact with humans and make them suitable for the analysis because our own design also interacts with humans.

Figure 6.1: Collage of existing robots for shape analysis

Description of shape

There are two types of shapes present in the robots in the collage. Geometric shapes are dominant, but one can tell that effort is being made to introduce organic shapes into the robot. Luna demonstrates this: the face is geometric since it has to house a screen and the main body is more organic - since it does not possess 'screen' limitations. The robot always seem to be an analogy of something else: be it akin to a human, a dog or a lawnmower.

Description of texture

All robots in the collage are made of metal or synthetic material. Robots with the white hoods have a smooth surface. The robots without hoods can be described as rough. This may have been done intentionally to make the product seem experimental and 'tweakable'. This may make them more attractive for hobbyists since it is not a 'finished perfect product'. We are working towards an finished product so we should aim for a smooth texture.

Description of composition

The robots often have a bigger lower part that is used for stability. This part makes the robots more stable, but it also make it also look more stable. Robots are generally wider than their real-life counterpart (e.g. human).

Description of space

The dominant type uses curves and closed shapes. Closed shapes means that there are no openings present in the shape. The volume is generally limited. There are no big blobs of volume present in the robots, except for the lower body part. This is often to account for wheels and other parts.

Fictional robots

Fictional robots are discussed because of their 'personality' that is present in fiction. This is an aspect that is desirable. Some fictional robots are shown to see what robots 'could look like' and 'can do' if not limited by current technology. Fictional robots present a different design than their real counterparts. They have more color and show more emotion. These robots seem to get more personality because of these extrovert colors - supporting their active role in fiction.

6.4 Morfological Chart

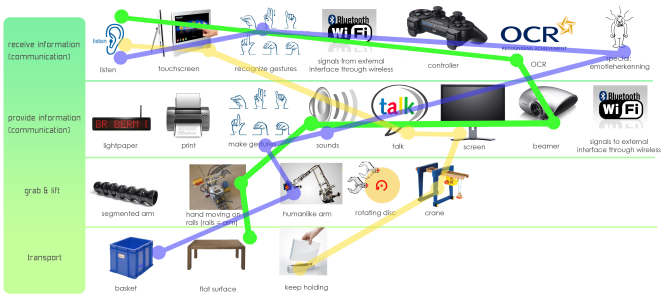


Figure 6.2: Morfological chart

The morfological chart is a method to easily and systematically generate ideas. The abstract functions formulated on the left of the screen are based on the PvE and means to find components for the robot. The lines in our chart represent combinations that are considered compatible or interesting to explore. Visual representations can be found in the report. These combinations can then be evaluated for further development. [1]

[1] (http://www.wikid.eu/index.php/Morphological_chart, consulted on 21 oktober 2011)

6.5 Conceptual Sketch

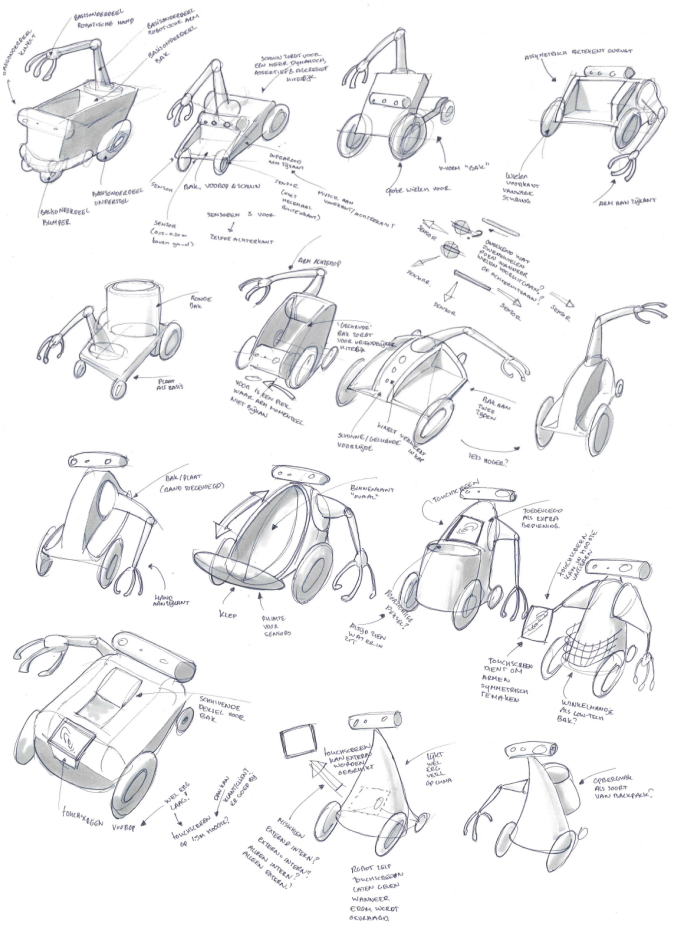


Figure 6.3: Sketching of general robot form and functionalities. A touchscreen was added in the end to provide a back-up interaction if the user cannot provide proper speech input.

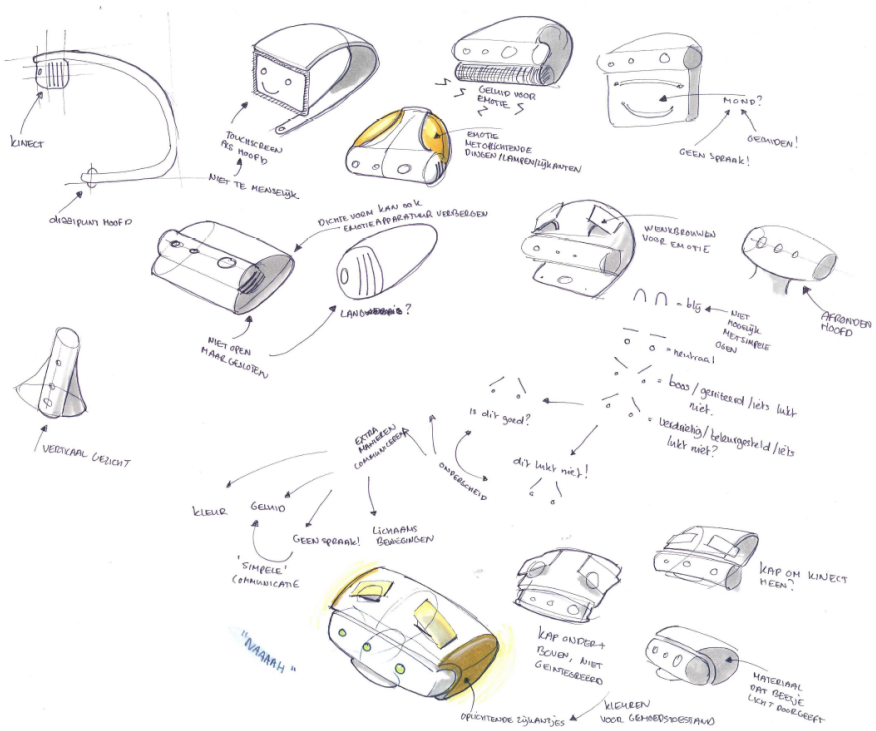


Figure 6.4: Sketching of head of robot. The focus of this part is on interaction. Variations show the different ways of communication.

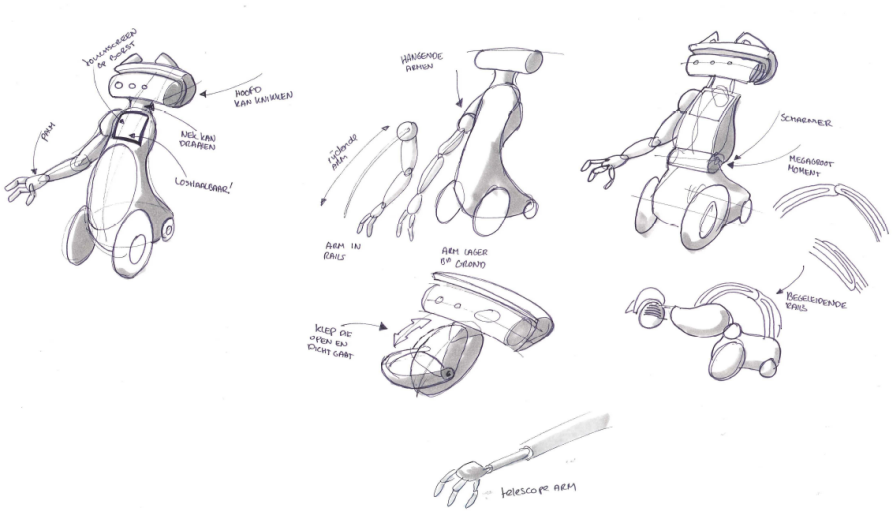


Figure 6.5: Sketching of combined features.

6.6 Uncanny Valley

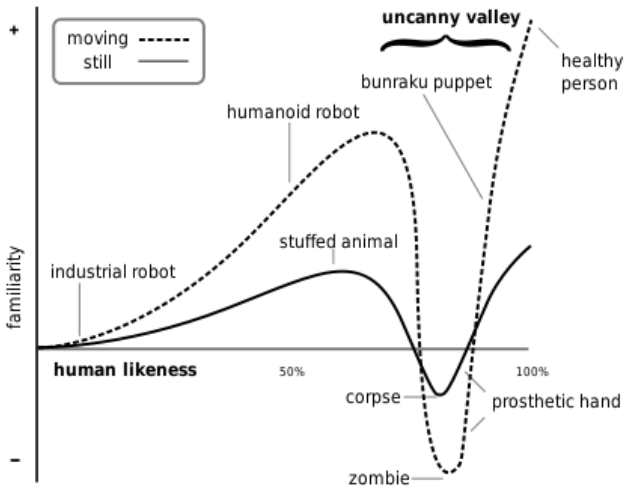


Figure 6.6: The Uncanny Valley figure by M. Mori. This shows that we should limit the human likeness.

The uncanny valley is a gap in familiarity caused by human likeness. In essence it expresses that a product should not be too humanlike, because a just not human product looks very unfamiliar and uncomforting. (A corpse illustrates those feelings, as a corpse is almost human, but not completely since the human is deceased.)

7 Appendix B

7.1 Maneuvering calculations

7.1.1 Avoiding toppling

There are two main causes that will result in toppling: quick acceleration (for example sudden breaking) and taking tight corners too fast. Both cases are actually quite similar for the calculations.

First thing that is needed is the location of the mass center. As the robot is still in development, the location is kept variable. The robot and its mass center are shown in figure 7.1.

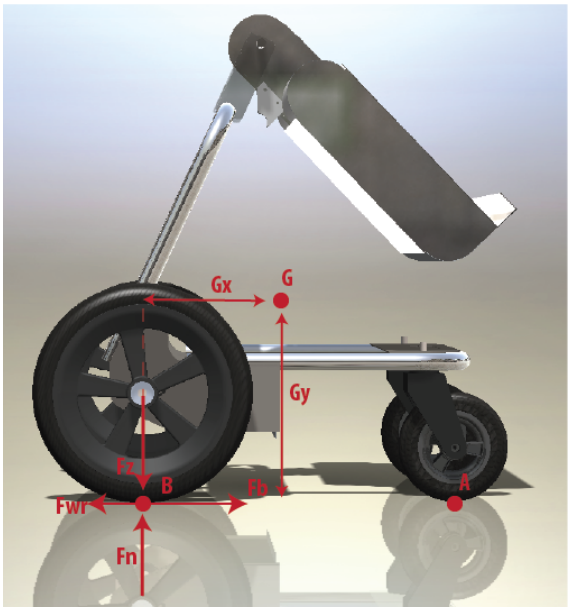


Figure 7.1: Mass center and static forces acting on the robot

To ensure the robot will not topple, it should not accelerate too fast. Figure 7.2 shows the driving acceleration as well as the gravitational acceleration acting on the robot. If the momentum these forces create about the wheel are in equilibrium, the robot is about to topple. This means that any acceleration faster than the equilibrium acceleration should not be allowed. This maximum acceleration is calculated as follows:

$$(m \cdot g \cdot x) - (m \cdot a \cdot y) = 0$$

$$a_{Limit} = \frac{g * x}{y} = g * \frac{x}{y}$$

$$a_{Limit} = a_{robot} + a_{friction}$$

$$a_{robot} = a_{friction} - a_{Limit}$$

Depending on the direction of acceleration (acceleration or decceleration), the friction force is positive or negative. This means that the maximum tolerable acceleration is lower while deccelerating because the friction adds decceleration.

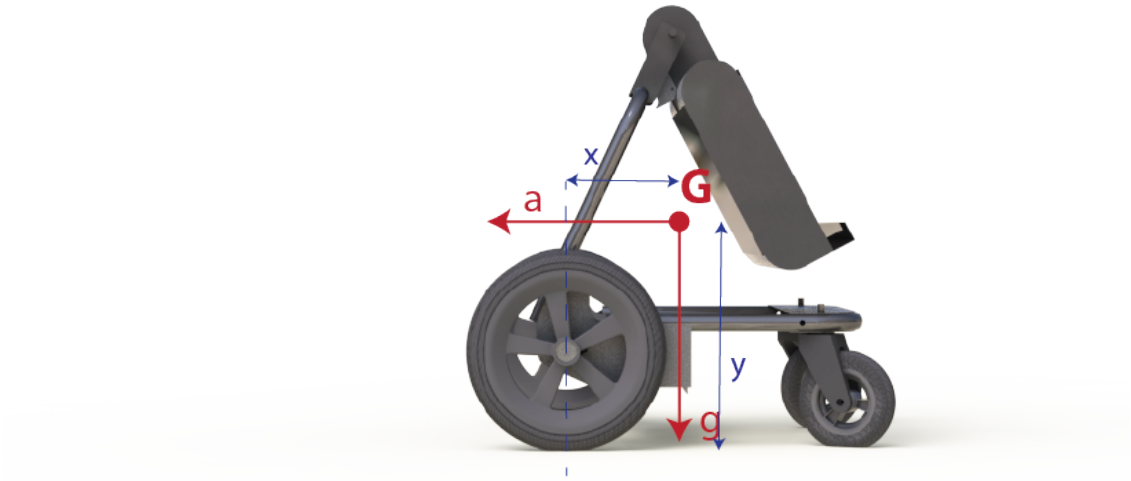


Figure 7.2: Acceleration and gravity in equilibrium: side view

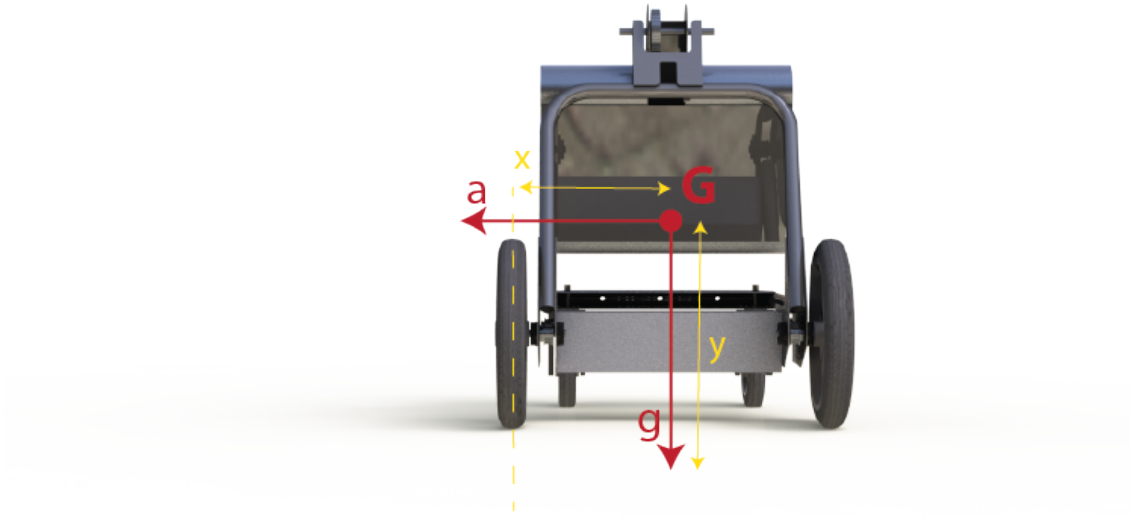


Figure 7.3: Acceleration and gravity in equilibrium: front view

Apart from quick acceleration and deceleration, tight cornering could also lead to toppling. This is because cornering introduces another acceleration which is perpendicular to the driving direction. This situation is shown in figure 7.3. The calculations for this situation are the same as for the side view. The angulular speed of the robot is directly related to the calculated side acceleration. This fact is used in section 4.3.3.1 to generate the overall safety requirements.

The distance of the mass center to the front (or to the side) of the robot is always shorter than the distance to one of its corners (e.g. the front-left wheel) of the robot. This ensures that if the robot topples, it will either topple to the side or to the front, but never over just one of its corners. This means that if the two conditions in the equations above are met, the robot will not topple.

7.1.2 Maximizing grip

To ensure the breaking path is as short as possible, grip of the wheels has to be maximized. In general there is more grip without slipping, as the dynamic friction coefficient is lower than the static coefficient. This was verified by noticing stick-slip behavior[1] while pushing the robot to the side. Apart from a longer breaking path, slip is also undesirable because the robot is not fully in control, nor is it aware of its position in that situation.

Avoiding slip is done by calculating the required friction force between the front wheels of the robot [2] and the floor. The acceleration should not cause this required friction to be larger than the feasable friction. This maximum allowed friction was measured as described in section 4.3.4.

Similar to the toppling calculations, a frictional force is required for linear acceleration as well as to support the robot when cornering. Both frictional forces are displayed in figure 7.4.

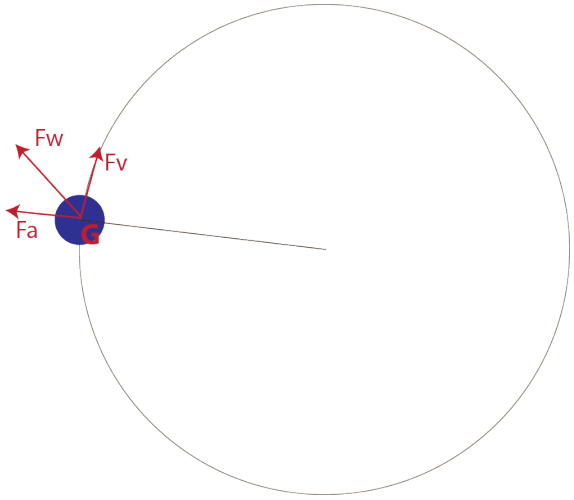


Figure 7.4: Frictional forces acting on the robot

[1] Stick-slip is the tendency for sliding motion to be intermittent rather than smooth, see http://webphysics.davidson.edu/faculty/dmb/PY430/Friction/stick-slip.html

[2] The back wheels are assumed to be frictionless, since they can move freely in any direction: they cannot counteract slip

The frictional forces directly follow from the accelerations, as can be seen in the formulas below.

$$F_v = m * a_v$$

$$F_a = m * a_a$$

Both frictional forces are vectors that are perpendicular to each other. The total required friction force is thus the vector sum of the two components. This total friction force should not exceed the measured friction force: $f_{w,total} = \sqrt{f_a^2 + f_v^2}, f_{w,total} < f_{w,allowed}$.

8 Appendix C

8.1 Software

This chapter describes the architecture and implementation model of the software. As usual in robotics, software is essential to control the robot. In order to make the robot move, there needs to be software that controls the engines of the wheels. On the other hand, software is also needed to enable the user to control the robot with a joystick or some other interface. To create this software it is first needed to make a design, which is divided in two parts: the software architecture and the implementation model. Usually, these two designs are preceeded by requirements engineering and use case modelling, which has already been done in the previous report [1] and in chapter 4 .

In computer science, the software architecture describes what subsystems are present and what their functions are. Also, the architecture describes how these subsystems interact with each other and how they are connected. This architecture does not necessarily focus on software, but it also describes the hardware subsystems since hardware has a very close relation to software [2] . Furthermore, the architecture shows the mapping of software on the available hardware (so which part of the software corresponds with which part of the hardware).

The implementation model provides a design for the realisation of the software architecture. To do so, a node chart is used to visualize the node constellation in ROS in figure 8.2 . This model shows how the nodes interrelate with each other and on what subsystem the nodes belong.

8.2 Software architecture

The whole system can be distinguished in multiple subsystems: a computer system that acts as a user interface, a computer system that acts as interface to the hardware, the robot (the mobile base and its sensors) and a PS3 Controller, for user input. In this case, the first computer system acts as a client and the latter acts as a server in the client-server architecture [3] . In other words, the whole system is based on a client-server model with only one client; the server receives the commands from the client to control the hardware. Figure 8.1 shows the connections between the subsystems.

8.2.1 Communication between server and client

In general, there needs to be a network connection between the server and client so they can communicate with each other. In this assignment the systems communicate with each other through a self-created network, with one of the systems as access point [4] . The systems know each other's IP addressess and names so they can address and message each other. Appendix D 9 shows how to set up a network connection between the systems. Once this network is set up correctly, the ROS

[1] See assignment 1 of the Robotics Minor WB-Mi-168-11

[2] http://wiki.answers.com/Q/Relationship_between_hardware_and_software_of_computer_system

[3] http://en.wikipedia.org/wiki/Client%E2%80%93server_model

[4] http://en.wikipedia.org/wiki/Wireless_access_point

nodes will automatically communicate with each other, even when they are executed on different systems (this is a useful feature in ROS).

Figure 8.1: Connection and interaction between the subsystems

8.2.2 PS3 Controller

As in the previous assignment, the user input is provided by a PS3 Controller and is used to control the robot's movements (for more details, see chapter 4.3.2). The controller communicates with the client subsystem through bluetooth, see appendix D 7 to see how to connect the PS3 Controller to the client subsystem. The PS3 Controller was chosen as input device because it is easy to use due to the buttons on the device, which are easy to distinguish from each other.

Another reason is that the PS3 Controller can also give feedback in the form of vibrations. This is used for object detection.

8.2.3 Client subsystem

The client system serves as the interface of the main system. This means that the client system listens to user input and gives feedback coming from the whole system to the user. As mentioned above, the user input is provided by the PS3 Controller.

The client displays as feedback the images taken from the camera at the server side of the system. Other forms of feedback are a map of the environment, which is made on the fly when the robot moves around and the vibrations of the PS3 Controller, which will be activated when the robot nears an obstacle in front of him. In summary, the client subsystem has the following tasks:

- Listening to user input device;
- Sending, based on input, control commands to the server;
- Listening to feedback from the server and display them, where feedback is also known as:
  - Camera images taken from the camera for teleoperation (see chapter 4.3.2);
  - Distances measured by the other sensors, so the client can let the PS3 Controller vibrate if needed (a form of physical feedback);

8.2.4 Server subsystem

The server subsystem receives commands from the client subsystems and executes them. If the commands return feedback information, the server sends the feedback to the client so that it can be displayed to the user. As described earlier, an example is that the server takes pictures from a camera and sends them over to the client.

Most of the commands the server receives are to control the engines of the robot or to give information from the sensors and camera. In other words, most of the commands require the server to access the hardware of the robot, either to control them or to read their measured values. That is why the server is directly connected to the robot and its sensors, the connection is through a serial port (USB). In short, the server has the following tasks:

- Listening to the client subsystem;
- Controlling the robot, based on the commands send over by the client;
- Sending information for feedback to the client, where information is read from the sensors attached to the robot 4.4 ;

8.2.5 The robot

The robot will execute the commands it gets from the server subsystem, which on its turn executes the commands from the client subsystem. The robot consists of the mobile base provided for the second assignment and the sensors needed to perform object detection. In a later stadium of the minor, the robot will include the gripper from assignment 1 and an arm, which will be build for the next assignment.

8.3 Implementation model

This section describes the design of the software structure in nodes, see figure 8.2. This design is implemented in ROS and will be expanded in a later stadium of the minor. The code can be found at [5] .

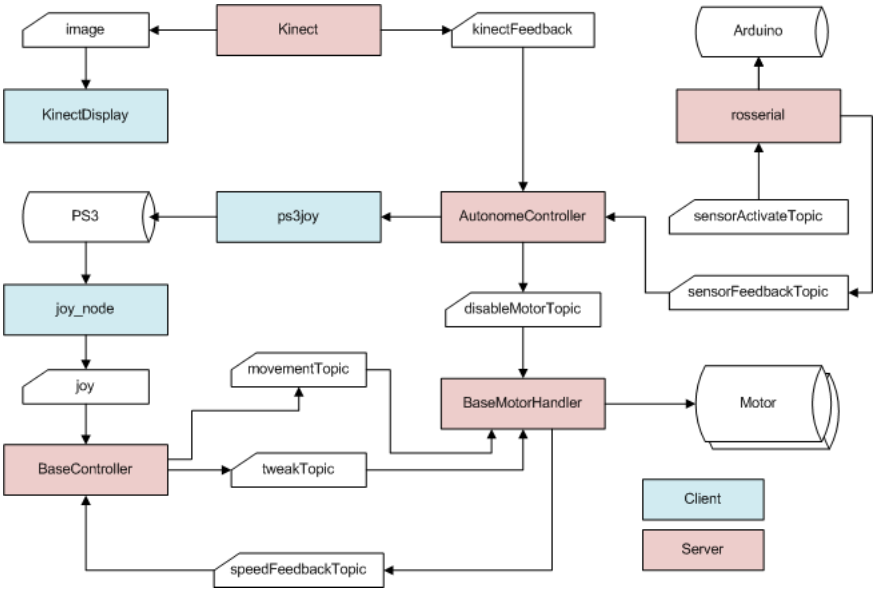


Figure 8.2: Node structure in ROS

The design distinguishes two kinds of nodes, nodes that run on the client and nodes that run on the server. In general, nodes that provide the user interface will run on the client system, while the other nodes to control the hardware will run on the server machine. The red nodes run on the client and the blue nodes run on the server. See below for a more specific explanation of the nodes and their functions.

[5] http://code.google.com/p/roman-technologies

8.3.1 Client nodes

The client nodes are as follows:

- ps3joy, connects the PS3 Controller to the client system and enables the PS3 Controller to rumble;
- joy_node, listens to PS3 buttons and publishes them over the joy topic for further processing;
- KinectDisplay, listens to messages sent over the image topic, which contain information about the captured images by the Kinect node. This node displays the images it receives to the user;

8.3.2 Server nodes

The following nodes will run on the server:

- BaseController, listens to messages sent by joy_node and acts depending on the pressed button. This is the main node of the design in the sense that most of the control happens here. It sends commands to the BaseMotorHandler to control the motors and sends tweak messages to tune the PID values of the selected motor, see 4.3.2 . Also, the node listens to the actual speed of the motors to calculate the angular speed of the robot, see 4.3 ;
- AutonomeController, reads the measured values from the sensors and the Kinect over the sensorFeedBackTopic and the kinectFeedback topic. It sends messages when the robot nears objects in front or behind it within 100 cm. This information could be fed back to the user as a warning, but it is mainly used to make the robot stop before it collide with the objects;
- BaseMotorHandler, this node listens to the commands sent by the BaseController and steers the motors. It calculates the velocities of both wheels and drives them. Also, this node stops the motors when receiving a message from AutonomeController;
- Kinect, reads the measured data from the Kinect camera and publishes data about the images them so the client can display them with the KinectDisplay;
- rosserial, provides the interface between the arduino device and the other nodes. In this case, it publishes the measured data from the ultrasone sensors which are plugged in the arduino;

9 Appendix D

9.1 Network

In this tutorial, it is assumed that the client runs Ubuntu 11.04 or lower and has a compatible copy of ROS installed. The server should run the server edition of Ubuntu 11.04 or lower, also with a compatible copy of ROS installed. We chose to install the server edition because it is more lightweight than the client version.

Also, this tutorial uses a Sony VPCEB2M1E laptop as client and a HP Elitebook 8530p as server, with the client acting as access point for broadcasting the network. Usually the broadcasting is done by the server, but since the HP Elitebook 8530p does not support this, the client is used as access point. The files needed for this tutorial can be downloaded at:

http://code.google.com/p/roman-technologies/source/browse/#svn%2Ftrunk%2Fnetwork

9.2 Network setup

The steps for the client system to set up the network are as follows [1] [2] :

Step 1 Download hostapd from http://linuxwireless.org/en/users/Documentation/hostapd and follow the instructions to install it. Do **not** set up hostapd.conf as the instructions say.

Step 2 Download hostapd.conf from svn and copy it to the directory /etc/hostapd/.

Step 3 Edit /etc/hosts, add the line "192.168.0.2 ${SERVER_HOSTNAME}".

Step 4 Edit ~/.bashrc and add the line "export ROS_MASTER_URI=http://${SERVER_HOSTNAME}:11311"

With this, the client system is now able to broadcast a network and act as an access point. As for the server system, only the file /etc/hosts needs to be adjusted. Add the line "192.168.0.2 ${CLIENT_HOSTNAME}" .

9.3 Network connection

To connect to the network, it is needed to download accesspoint.sh from svn to the client machine. The steps are as follows [3] :

- Go to the directory where accesspoint.sh is located
- Run "chmod 755 ./accesspoint.sh"
- Run "./accesspoint.sh"

[1] Note that sudo needs to be used for some of the commands.

[2] Replace ${SERVER_HOSTNAME} by the hostname of your server, which can be found by typing "hostname" (without quotes) at the command line of the server system. Do the same for ${CLIENT_HOSTNAME} for the client's hostname.

[3] It is important to follow the client steps first, before following the server steps in connecting to the network.

For the server machine it is needed to download the file connect.sh

- Go to the directory where connect.sh is located
- Run “chmod 755 ./connect.sh”
- Run “./connect.sh”

To check whether the connection succeeded, run ping ${SERVER_HOSTNAME} at the client machine and run ping ${CLIENT_HOSTNAME} at the server machine. If this succeeds, the connection has been set up properly.

10 Appendix E

10.1 SRF02 Sensor Connection

All six sensors are connected on an I2C[1] bus as a slave device. Therefore they all need a different address[2] (0xE0, 0xE2, 0xE4, 0xE6, 0xE8, 0xEA). An Arduino Uno is serving as an I2C host triggering the sensors and reading the distances. Figure 10.1 shows how one of the sensors is connected to the Arduino. The remaining five sensors are connected in parallel with the first one.

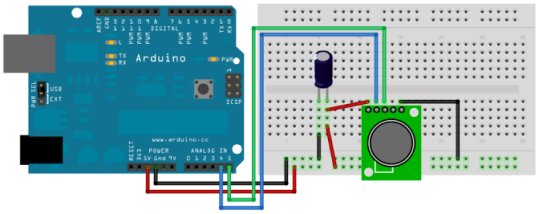


Figure 10.1: Connection of one SRF02 on a I2C bus

Ultra sonic sensors next to each other cannot operate simultaneously due to the risk of interference. A sensor is triggered in the following way:

First the address of the sensor that has to do a measurement is selected. Then the sensor is asked to return the detected range in centimeters by writing 0x51 to its control register. Subsequently, a delay of 65 microseconds is needed to give the sound time to return to the sensor. When the sensor has completed the ranging, the value is written in his register and can be accessed from address 0x02.

10.2 Kinect

The fitting of the plane on points from a point cloud is done using the Point Cloud Library (PCL). From this PCL we use the NormalEstimation class and from this class the computePointNormal[3] function. This function takes a PointCloud¡pcl::PointXYZ¿ object (a 2D matrix of points in a 3D space) and indices of points in this cloud that need to be used. It outputs plane parameters in the form of :

$$ax + by + cz + d = 0$$

where x, y and z are point values. To determine whether a point belongs to the ground or to an obstacle, its distance to the calculated plane is checked. This distance is calculated using the

[1] http://www.i2c-bus.org/ Consulted at 21-11-2011

[2] http://www.robot-electronics.co.uk/htm/srf02techI2C.htm Consuted at 21-11-2011

[3] http://docs.pointclouds.org/trunk/classpcl_1_1_normal_estimation.html, consulted on 18 october 2011

following formula[4]:

$$D = \frac{ax_0 + by_0 + cz_0 + d}{\sqrt{a^2 + b^2 + c^2}}$$

If this distance is above a certain tolerance, it is considered obstacle. This tolerance is set at 0.05, which is in meters, so 5cm. This distance is chosen through empirical testing.

[4]http://mathworld.wolfram.com/Point-PlaneDistance.html, consulted on 18 october 2011

11 Appendix F

11.1 Technical aspect of the front bumper

To mount the front bumper we designed an extra frame part, which we could mount on the existing frame of the mobile platform.

The front bumper is made of an aluminum tube with 2 layers of insulation material around it. In the aluminum tube there are two slots with two spacers with springs around it. The springs are used so that the bumper can be impressed. And the slots ensure that the bumper can slide front and back ways. When a spring is fully impressed a switch is activated so that the wheels are deactivated. The impact will be absorbed by the springs and by the insulation material. The spring can be 10 mm impressed and the insulation material can be 2 times 18 mm impressed. So the total impression is 46 mm, which is larger than the design requirements, 40.33 mm. So the robot will not heard a person.

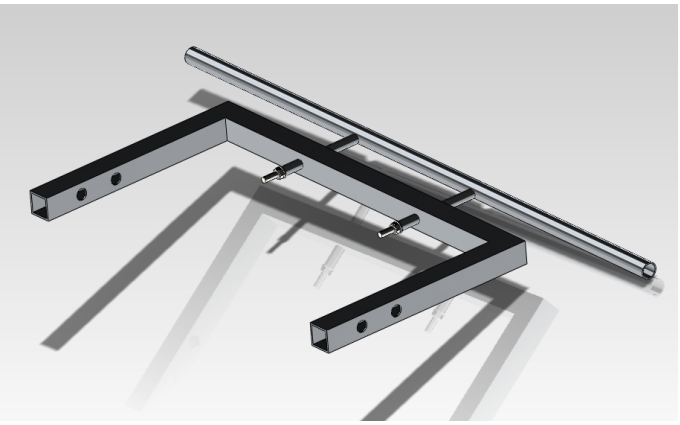


Figure 11.1: Technical drawing of the front bumper

11.2 Technical aspect of the back bumper

The back bumper is also made of an aluminum tube with 2 layers of insulation material around it. In the aluminum tube of the back bumper are four slots with four spacers with springs around it. The impression of the springs and the insulation has the same working principle as the front bumper. The difference between the front bumper is that the back bumper can absorb an impact from the back but also from the side. This is possible by the 2 slots in the side and 2 slots in the back of the aluminum tube. When the bumper is pressed backwards the bumper will slide in the 2 slots from the side. And when the bumper is pressed sideways the bumper will slide in the 2 slots in the back. By this design the bumper can move in the hole 2D surface with boundaries of the distance of the spacers.

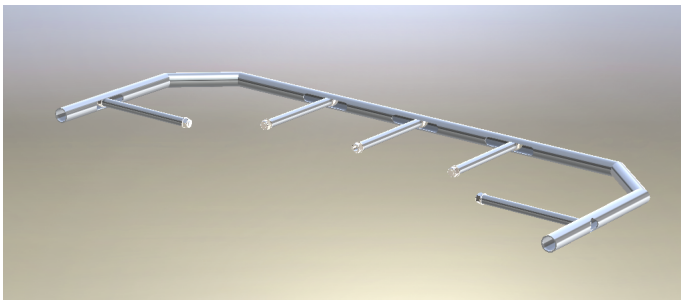


Figure 11.2: Technical drawing of the back bumper

11.3 Button schematics

A total of six buttons are placed on the bumper. Together they can distinguish four different locations of impact. Front, rear, rear right and rear left. The buttons are connected to the Arduino Uno that is also used to read the other sensors. Figure 11.3 shows how the buttons are connected to the Arduino.

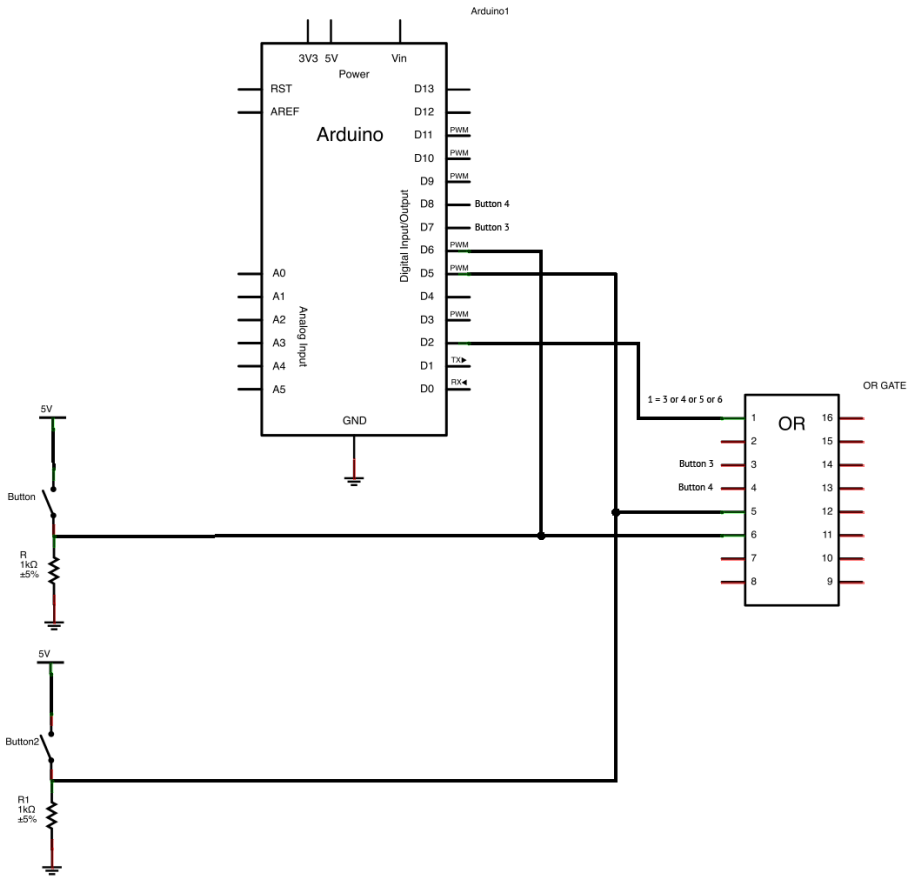


Figure 11.3: Button connection

Each button is connected to the ground with a 1 kΩ pull-down resistor. All button outputs are connected to a logic OR-gate as well as directly to an Arduino digital input. The output of the OR-gate is then connected to an external interrupt pin on the Arduino. When the interrupt pin is pulled high, a subroutine is initiated on the Arduino which reads out the button states. Then, the Arduino is able to report the location of impact to ROS.