INTERNSHIP REPORT

INTEGRATION OF A MODULAR AND INTUITIVE SOFTWARE PLATFORM FOR NEXT GENERATION HAND PROSTHESIS

PETER WESTENBERG

Author	*Peter Westenberg*
Student number	*140457*
Date	*30/10/13*
Version	*1.3*
Faculty	*Robotics and mechatronics*

UNIVERSITEIT TWENTE.

ABSTRACT

Partial amputation of the arm is a life-altering event. The human hand performs several vital actions such as complex motor functions and nonverbal communication. The hand also provides sensory information about the environment. As such, amputation leads to major disabilities in performing everyday actions.

After amputation several options for rehabilitation are available. The three most common devices are: cosmetic prostheses, body powered prostheses and Myo-electric prostheses. The Myopro project uses the Myo-electrical system because it has the most potential for further development.

The Myo-electrical prostheses measure the activity of the remaining muscle tissue in the arm of the user. These signals will be converted into control signals for the actuators in the prosthesis.

Currently available Myo-electric prostheses have limited functionality and lack any form of feedback. In addition, the absence of an intuitive control system is a huge disadvantage for users and 70% stop using their prosthesis in the long term.

To overcome these shortcomings, the goal of this internship assignment is to improve, develop and integrate the high-level (grasp selection) and low-level (finger control) control systems into one modular software platform. In this way the modularity, user-friendliness, intuitive-use and accessibility of the new test system can be improved. This report describes the results of this assignment.

TITLE PAGE

CLIENT

Mr Rd. S. Misra

Postal and physical address
University of Twente Carré
Hallenweg 23
7522NH, Enschede
Room: Carré 3.609

PROJECT NAME

Project group Myopro wp4 control

CONTACT PERSON UNIVERSITY OF TWENTE

Mr Peerdeman
Tel: (0)53 489 2814
Email: B.Peerdeman@utwente.nl

INTERNSHIP COORDINATOR

Mr Eng. H. Van Der Meij
Tel: (0)53-4871473
Email: j.a.m.vandermij@saxion.nl

PROJECT GROUP

Main project group: *Myopro*
Sub-project: *Myopro wp4 control*
Academy: *Robotics and mechatronics (RAM)*

Postal and physical address:
Hallenweg 23
7522NH, Enschede
Room: Carré 3.530

Contact details
Tel: (0)53 489 2814
Email: B.Peerdeman@utwente.nl

PROJECT STRUCTURE

Mr Dr S. Misra
Project leader
S.misra@utwente.nl

Mr MSc B. Peerdeman
Supervisor
B.peerdeman@utwente.nl

Mr P. Westenberg
Project member
PeterWestenberg@hotmail.com

ABBREVIATIONS

Term	Definition
Saxion	Saxion University of Applied Sciences
UT	University of Twente
RAM	Robotics and mechatronics Research group university of Twente (formally Control Engineering)
BME	Biomedical Engineering
CTIT	Centre for Telematics and Information Technology
MIRA	Institute for biomedical technology and technical medicine
EMG	Electromyography
sEMG	Surface electromyography
Biotac	Syntouch Biotac
National Instrument Board	National Instruments ELVIS DAQ device, National Instruments Corporation, Austin, TX, USA
Matlab	The Math Works, Inc., Natick, MA, USA

Table 1

GLOSSARY

Term	Description
Grasptype	The name for a pre-programmed shape of the hand
Preshaping	Shaping the fingers of the hand in a pre-programmed and automatical way
Neutral state	State of the system in which the system waits for input from the user
Preshaping state	State of the system in which the system preshapes the hand
Grasping state	State of the system in which the user is able to open or close the hand
Transition state	If the user switches from preshape during the preshaping state, the system enters a transition state.

Table 2

VERSION CONTROL

Version	Description
0.1	Chapters
0.2	Preface and introduction added
0.3	Myopro chapter added to the report
0.4	Look and feel
0.5	Added chapters abstract, preface, company, assignment, requirement, functional design and made improvements
0.6	System architecture added
0.7	Implementation
0.8	Validation and conclusions recommendations etc. are added
0.9	Tests implemented
1.0	feedback
1.1	Updates and appendix
1.2	Review update
1.3	Final version

Table 3

1 CONTENTS

1 PREFACE

This internship report is the result of a five month internship at the University of Twente at the RAM research group within the Myopro project, executed by Peter Westenberg, student of mechatronics at the Saxion University of Applied Sciences.

In the third year of the mechatronics program a five month internship is a compulsory part of the program. Conventionally a HBO Bachelor degree student performs this internship at a commercially established company. The purpose is to gain experience with a commercial company and to learn how to apply knowledge to a practical and challenging assignment.

However, I wanted to perform the internship at the University of Twente due to my interest in following the BME master program. This gave me the opportunity to see whether I could handle academic working and thinking, and to combine research and development. Through this combination it was possible to gain new knowledge and experiences.

Besides this, the main motivation was my ambition to work on the dividing line between mechatronics and medical robotics, preferably in the research field of next generation prosthesis. For this reason this internship was a great opportunity to contribute to this interesting field of research and eventually to help people to live a better life.

Before you start reading the internship report I want to give thanks to some people who made it possible to conduct my internship. Through the help of Mr Van Der Meij it was possible to contact the right people at the University of Twente- he made it possible to find and conduct this internship. Further, I want to thank my supervisor Mr S. Misra and mentor Mr B. Peerdeman for the trust in me, before and during my internship.
Further on I want to give thanks to my mentor Mr B. Peerdeman for all the help when it was needed; you always had the patience and time to help me and to explain everything I wanted to know.

Last but not least I would like to thank all the students for the helpful discussion and informative talks. I want to give special thanks to Marcello Valori PhD student from Italy, for helping me with calculating the homogenous transformation matrices for the kinematics of the Myopro prosthesis.

Only one thing remains to say: Enjoy reading.

2 UNIVERSITY OF TWENTE

The University of Twente is a young and entrepreneurial university that is located between Enscheda and Hengelo. It is a research university that focusses on the development of future technology to stimulate change, renewal and progress in society.

The university has more than 3300 scientists and students 9000 in the year of 2012. They offer 20 Bachelor degree programs and 31 master degree and PhD programs. The programs offer challenging, thematically oriented education projects in which students will be able to discover their own strengths and utilize them.

The university campus houses more than 100 companies and more than 700 successful spin-off companies. The University of Twente also collaborates with companies to answer their questions. These activities have been united through the platform 'kennispark Twente'.

All of the faculties' research initiatives are housed in six institutions on the campus of the university. This internship assignment was performed in the Myopro project. This project is part of the Robotics and Mechatronics research group (RAM) within the Myopro project.

2.1 ROBOTICS AND MECHATRONICS

Robotics and Mechatronics, (formerly Control Engineering) embedded in the CTIT and MIRA institutes, has been popular amongst students as a group to perform BSc- or MSc-project. The research group deals with the practical application of modern systems and control methods. The focus is on robotics, as a specific class of mechatronic systems which typically require a multidisciplinary system approach.

The research group is application oriented. The main goal is to investigate the applicability of modern systems and control methods to practical situations in the area of robotics.

In the field of robotics the research topics are:

- Inspection robotics.
- Medical robotics (assistance to surgeons).
- Service robotics (street cleaning, service to people).

In the field of science and engineering the research topics are:

- Modelling and simulation of physical systems.
- Intelligent control.
- Robotic actuators.
- Embedded control systems.

3 MYOPRO PROJECT

An amputation or partial amputation of the arm is a life changing event. This leads to major challenges in performing daily life activities. The human hand performs several important functions such as the execution of complex motoric functions, plays an important role in communication with the user's environment (non-verbal communication)[1] and provides important sensory feedback from the user's environment.

To minimize the loss of functionality and improve quality of life, several different reconstruction methods are available. The three most commonly used devices for upper-extremity amputee are:

1. The cosmetic prosthesis.
2. Body powered prosthesis.
3. Myo-electric prosthesis.

Cosmetic prosthesis

The first option is a cosmetic prosthesis. This is a passive prosthesis. This means it is impossible to move the different components of the prosthesis. Therefore it possesses limited functionality in performing daily life activities. The advantage of these prostheses is that they closely resemble the human hand.

Body endorsed prosthesis

The body endorsed prosthesis makes it possible to perform some daily activities. These prostheses are also highly durable and give some kind of feedback to the user. As such, they enable the user to develop some kind of feeling with the prosthetic system, to a limited extent. However, they do not resemble the human hand as closely as a cosmetic prosthesis. There is also a high risk of developing back injuries when this device is used on a long term basis..

Figure 1

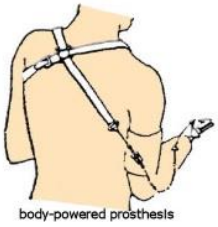


Figure 2

Myo-electric prosthesis

Finally there are Myo-electric prostheses. In these prostheses, the electrical activity of the muscles is measured with electrodes attached to the skin of the user's arm and processed by a classifier. These signals from the EMG classifiers are converted into control signals for the actuators in the hand or arm.

Currently the use of Myo-electrical prosthesis is low, mainly due to the lack of functionality. These systems have few selectable grasp types (only extension, flexion) and are limited in selectivity control. There is also an absence of sensory feedback and lack of intuitive and natural control. Further, the contraction of functional muscle tissue is a problem, since muscle training can only begin after fitting the prosthesis.

Although this method has the highest potential for further development, approximately 70% of the users stop using their prosthetic device on a long term basis. This is because of the above mentioned shortcomings in currently available Myo-electrical prostheses.

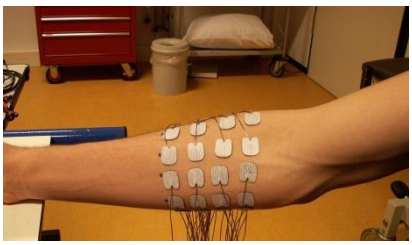


Figure 3

Figure 4

3.1 PROJECT MOTIVATION AND GOALS

Due to the shortcomings mentioned above, practical and long term use is not a possibility. Nevertheless, this method has the highest potential for further development. To enable **practical** and **long-term use,** the amputee needs prosthesis with a *natural*, *intuitive*, *fast* and *non-fatiguing* command interface.

This method in combination with a training setup that uses a virtual reality environment enables the patient to immediately start with selective muscle training. This is an entirely new concept and forms the basis of a functional prosthetic system that could partially restore the lost hand function. The goals of the Myopro are to:

- Develop a natural, intuitive, fast and non-fatiguing command interface. Using a non-invasive measurement method (multichannel surface electromyography).
- Improve the control of a Myo-electric arm-prosthesis by increasing the number of degrees of freedom using multichannel surface electromyography.
    - Improving the hardware and mechanical design of the hand
    - Improving the control systems of the hand.
- Develop a natural and intuitive feedback mechanism.
- Develop a virtual reality training program to enable targeted early-phase rehabilitation.

3.2 PROJECT STRUCTURE AND ACTIVITIES

The Myopro project is part of the Robotics and Mechatronics research group at the University of Twente. It is also a member of a consortium for commercially companies and knowledge institutes. These companies have extensive knowledge and experience in the field of health and technology.

To implement the defined goals of the Myopro project, the research is divided in to seven work packages and several sub-projects. Each of these projects has a specific development goal for the realisation of the next generation's prototype. A very short description is given below, for further information see http://www.myopro.nl.

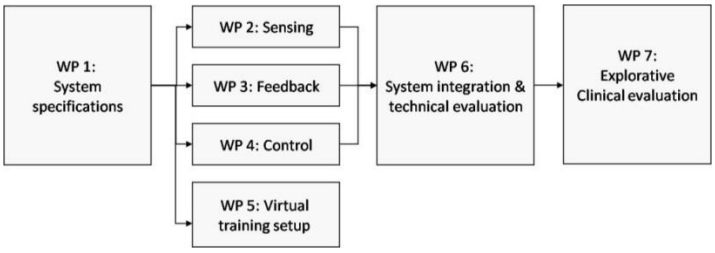


Figure 5

- **WP2 : Sensing**
    - This project group aims to develop a classifying algorithm that 'senses' the intention from the user with the muscle activity of the residual limb.
- **WP3 : Feedback**
    - This project group aims to develop a feedback mechanism to give the user more information than only visual feedback concerning the status of the prosthesis. The goal is to improve the intuitiveness of the prosthesis.
- **WP4 : Control & mechanics (my project group)**
    - This project group aims to develop a control algorithm that converts the input from the user into motor movements. By using a new concept of interaction control strategies, "intended behaviour" instead of "intended movement". Thereafter, this group aims to improve the mechanical hand design.
- **WP5 : virtual training setup**
    - This group aims to develop an early phase training environment to shorten the training time with the prosthesis and minimize the degradation of the residual muscle tissue.

3.3 HARDWARE DESIGN / TEST SETUP

To develop a hardware design for the prosthetic hand, with many degrees of freedom and good user control, previous research was conducted by Bart Peerdeman[2]. This research led to a hardware design based on controlling multiple degrees of freedom with a single actuator, called under actuation. In this way it is possible to achieve many finger motions with a minimal number of actuators.

The under actuation is attained by attaching tendons around pulleys which have been implemented in the fingers. The tendon of every finger has been connected to one common tendon that is in turn connected to the actuator. To form the fingers in a specific shape, specific joint locks need to be locked. The two figures below show several finger shapes and the hardware design[3] of the test system picture from this paper can be seen below[4].

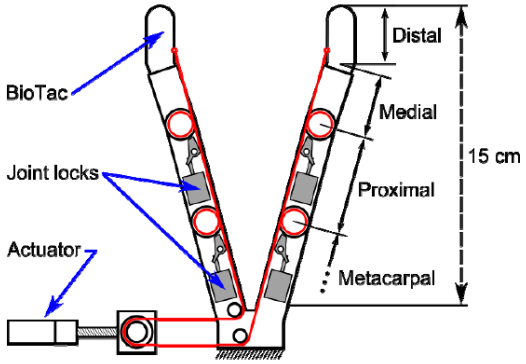


Figure 6

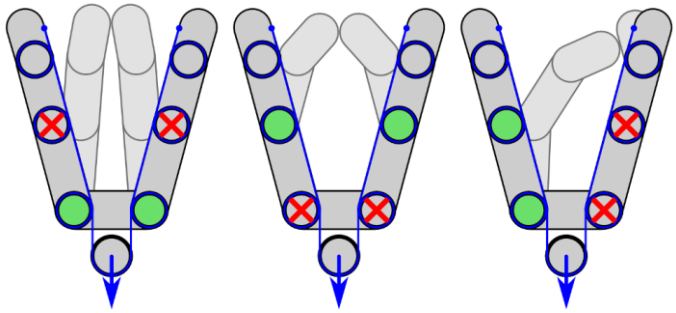


Figure 7

To control the behaviour of the system it is necessary to know the interaction with the environment and the position of each joint. A hall sensor is attached to the end of each joint to measure the angle, in this way it is possible to calculate the position of each joint. Secondly a Biotac sensor system has been attached to the end of each finger. This sensor provides two important input signals for the system. Firstly, it determines the magnitude and direction of the force applied on the surface of the Biotac with several pressure sensitive electrodes, and secondly it is able to detect high frequency vibrations on the surface with a common pressure sensor.

This new hardware design led to a two-fingered prototype that will be used during the internship. The figure below shows a detail picture with all important components. In previous research the hardware has been attached to the computer but with the current software it is not possible to communicate with all the hardware within one software platform.

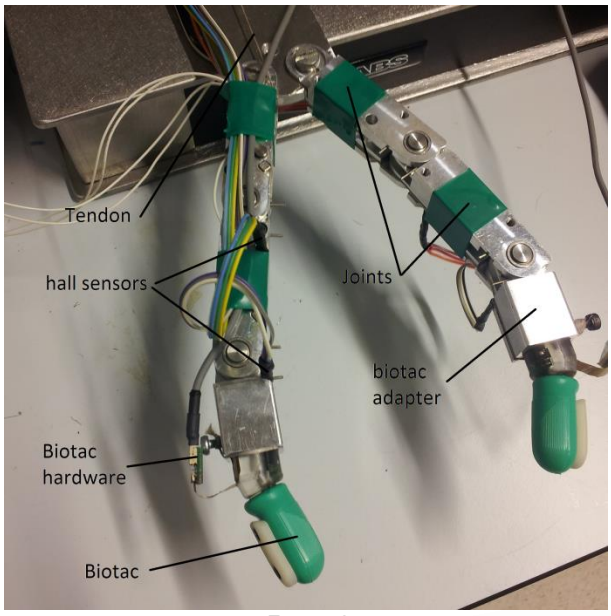


Figure 8

3.3.1 CONTROL

Alongside to the development of a two- fingered prototype, research has been performed in the field of motion control[2] and EMG sensing[1]. This led to a grasp control system for the prototype. The current control system is divided into two layers, a high level grasp control system and low level automated finger control system, as seen below.

Figure 9

- The high-level controller determines the automatic behaviour of the prosthesis based on several user input signals.
- Low-level control determines the forces the actuator should apply to move the fingers to the desired end position.

3.3.2 HIGH LEVEL CONTROL

The input for the high level control system is data from the EMG classifier. These data presents what the user intends to do with the prosthesis. The current model of the controller can be seen in the picture below. The dashed arrows are automatic transitions; the others can be influenced by the user. Implementing and improving this model will be done within this internship project.

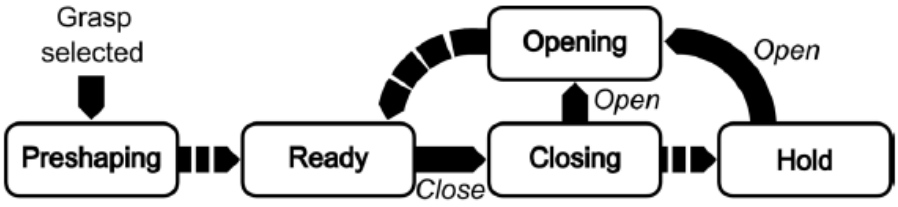


Figure 10

Most of the grasping of an object is an automated process through the limitations of the EMG classifier. The system must operate with a limited number of input signals. However, this problem may be solved by combining automatic behaviour and user controlled execution of certain tasks.

To begin grasping an object, the user has to shape the hand in a pre-programmed shape. To execute a grasp, a grasp type must be selected by the user. After this is selected, the prosthesis automatically shapes the hand into a pre-programmed position of the hand, called preshape. When preshaping is done it is possible for the user to open and close the hand until the user wants to change the preshape.

Examples of preshape can be seen in the pictures below. How the hardware is designed to achieve this kind of behaviour is described in the test system chapter.

Figure 11 lateral grasptype

Figure 12 cylindrical grasptype

Figure 13tripod grasptype

3.3.3 *LOW LEVEL CONTROL*

To enable the system to interact with its environment, a robust controller is necessary. Preliminary research tested several different controllers and implemented them on the UB hand[2] The described controls are: impedance control, admittance control and intrinsically passive control[2,3] see picture below.

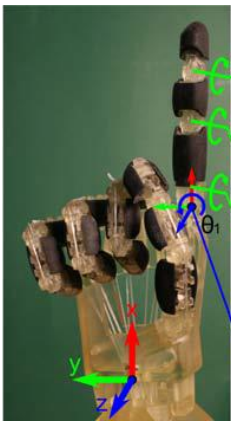


Figure 14

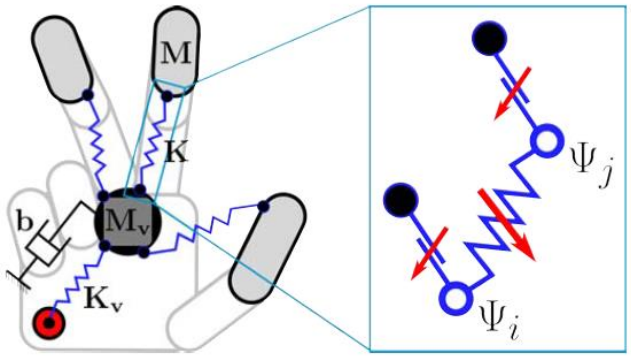


Figure 15

3.3.4 *FUTURE WORK*

Integration of the control systems and state machine as well the feedback and EMG sensing subsystems, is the main priority for future development. Further, once the five fingered prototype is completed the control systems will be adapted to accommodate its increased functionality.

3.4 INTERNSHIP ASSIGNMENT

In summary, the goal of the internship is to develop and improve the high level controller and the framework for the low level controller (to be developed) and develop an integrated software platform for the current and new five-fingered prototype with user intuitiveness and modularity. In this way it is possible to deliver an enhanced intuitive experience and improved functionality and an easily adaptable and accessible system. The focus points of the project are:

- Implement an integrated software platform to enable modular, plug and play data connections.
- Implement the high level controller to improve intuitiveness and enhance functionality.
- Implement the low level controller framework to allow the implementation of the controller.
- Validate the system, check if the implemented system complies with the compulsory requirements.

As described earlier, the current prostheses have limited control and lack intuitive control. Therefore an important goal for the assignment is to develop a new high level grasp controller that offers enhanced intuitive experience and improved functionality to the user. The requirements concerning the intuitive and functionality can be found in the requirements chapter. In addition, the framework for the low level controller has to be implemented; this means there must be a function which can access all the necessary inputs for the controller. By doing this, it should be easy to implement this controller into the software platform.

In addition, constructing a software platform for the existing two-fingered prototype and new five-fingered prototypes is necessarybecause the current setup consists of several separated system. Through this, communication between the systems and the simultaneous processing of data is not possible. Furthermore the implementation of new software or hardware components proves to be difficult.

By developing an integrated software platform, the test setup should become easily accessible (plug and play) and adaptable. This means it is easy to change the software or hardware configuration. Furthermore integrated means data exchange and simultaneous processing is possible.

Validation of the system is the last goal within the internship. The system is reviewed to assess whether it meets the requirements.

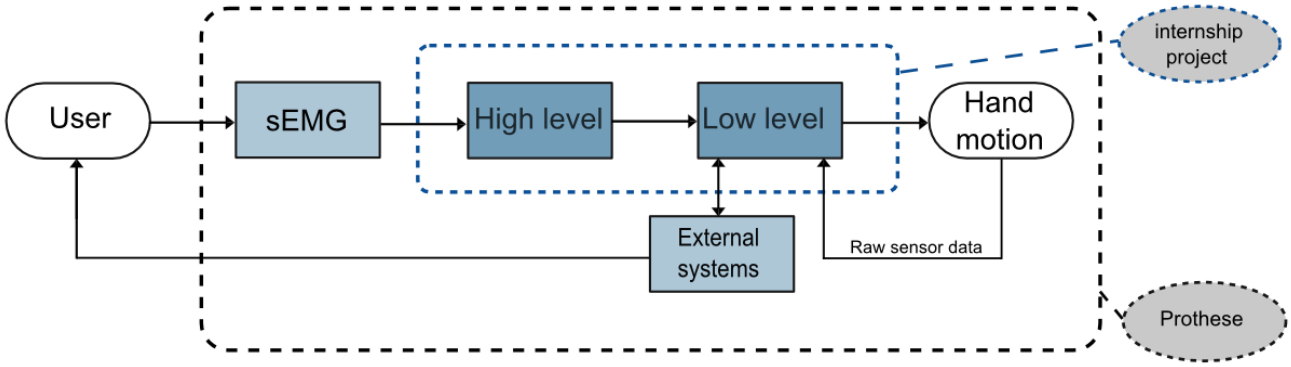


Figure 16

3.5 PROJECT BOUNDARIES

3.5.1 ENVIRONMENTAL

The goal of the internship project is to develop an integrated software platform and implement the possibility to integrate other Myopro software functions. This should provide a natural, non-fatiguing and fast command interface. The trainee did not develop other subsystems such as the EMG, slip detection, feedback mechanism and the hardware and electrical design of the test platform. A working two-fingered test setup was provided to the trainee to use during the internship assignment for testing purposes.

The next recourses should be available during the internship period:

- A complete and working test setup which can be controlled by the computer
- The appropriated software which makes it possible to communicate with the hardware components of the system

3.5.2 DEADLINES

In the table below the important dates within this internships time schedule are listed.

What	Datum
Project start-up	01-09-2012
Project plan completed	18-09-2012
Literature research	03-10-2012
Defining requirements	19-10-2012
Functional design	31-10-2012
Software platform	15-01-2012
Internship report	25-01-2012
Presentation	25-01-2012
End date project	01-02-2012

Table 4

3.5.3 TIME SCHEDULE

The format of the schedule is very large, therefore it will be delivered as a separated document in Microsoft project.

3.5.4 WORKING HOURS

The normal working hours have been listed below. Although it is possible to work outside these hours and there is a possibility to work at home, through a remote desktop connection.

Day	Begin time	end time
Monday	9:00	17:30
Tuesday	9:00	17:30
Wednesday	9:00	17:30
Thursday	9:00	17:30
Friday	9:00	17:30

Table 5

3.5.5 *ACTIVITIES*

During the internship project several activities have been conducted. The goal of these activities was to develop a specific sub-product. The table below presents the activities. For the most part these documents will be included in the internship report and not submitted as a separated document

Activity	**product**
Project start-up	Project plan
Meetings	Minutes, progress presentations
Literature research	Background information
Defining requirements	Requirements document
Designing High level and low software architecture	Functional design
Preparing/updating the test setup	A working/ functional test system
Gaining extra knowledge (kinematical calculations, PID controllers)	Input information for improving the software controllers
Developed and implemented the software	Integrated software platform within Matlab environment
Testing the software	Test and analysis report
Finishing project	Internship report, presentation

Table 6

3.6 QUALITY

Technical documents, software and hardware system extensions will be constructed during this internship project. The way the quality is ensured is explained in the section below.

3.6.1 *GENERAL*

To guarantee the quality of the products during this internship project a weekly process meeting was held with the supervisor B. Peerdeman. Feedback on the completed activity and sub-products was given during this meeting. In addition the activities of the next week were discussed and new decision were made. In this way the supervisor had good insight into the progress and quality.

3.6.2 *DOCUMENTS*

All documents were produced with the same layout, using primarily Microsoft office products. Drawings could be made with Solidworks or Inkscape. All the documents were checked by B. Peerdeman and had to meet the requirements of the V-model, the supervisor and Saxion. In addition, it is important that the documents have a professional look and feel and have a distinctive structure. In this way they should be accessible and understandable for other people.

3.6.3 *SOFTWARE*

To ensure the quality of the software the V-model method is used. Besides this, tests have been conducted to ensure correct functionality of every (sub) function. To make the software understandable for the next students, several points have to be described in the software:

- Declare the start and end of a particularly function with a distinctive name.
- What the functionality of the function is.
- The event table, if possible.
- Logical, self-explanatory names for variables and functions

3.7 MEETINGS

Mr B. Peerdeman is the supervisor of the trainee Peter Westenberg. It was possible for the trainee to ask questions on a daily basis. However, a weekly progress meeting was held to discuss new products, activities and progress. The template used in these meetings can be found in appendix 3.After the meeting these minutes were sent to the supervisor. In this way it was possible to track all changes, progress, deadlines and decisions and it will be available to the next students as backlog.

In addition, a group meeting was held with the overall project leader every Monday. Once every two or three months a progress report was delivered by the trainee during a group meeting.

In addition two compulsory visits with the supervisors of Saxion were conducted. One at the begin of the internship, to provide an introduction to the assignment and the second to finish up the internship.

Who	What	Frequency
Mr. B. Peerdeman	Ad hoc contact for questions	daily
Mr. B. Peerdeman	Weekly progress meeting	one week
Mr. Dr. S. Misra	Progress meeting	2 weeks
Mr. Dr. S. Misra	Presenting results, progress and problems	3 months
Mr. V.D. Meij	Internship meeting for Saxion	2.5 months

Table 7

4 SYSTEM DEFINITION

4.1 REQUIREMENTS

Before the software platform was developed the requirements had to be determined in consultation with Bart Peerdeman. These requirements have been divided in several categories and will be explained in the sections below. Only the requirements that are compulsory to the internship assignment will be explained all other requirements have been left out.

4.2 SOFTWARE PLATFORM

The requirements involving the architecture, functions or functionality of the software platform have been stated below. Some requirements will be explained in more detail.

4.2.1 GENERAL

Currently all the software is distributed over separated systems, which are not able to communicate with each other. These systems have to be redesigned to implement an integrated software program (platform). The general requirements for this have been listed below.

Description	Value		Unit
The software platform must work in a Matlab environment	Yes	=	-
Realization of a software program in which the software functions are able to communicate with each other (integrated platform)	Yes	=	-
All input and output signals within the software of the platform should be easily accessible*	Yes	=	-
It must be possible to integrate the new five fingered prototype in an easy way	Yes	=	-
All the different functions should be able to communicate with each other	Yes	=	-
The software platform must be modular	Yes	=	-
The software platform should be divided in a user controlled and automatic controlled part	Yes	=	-

*Table 8 *In this context easy means: no need for programming new software for accessing the compulsory data*

4.2.2 MODULARITY

Modularity means how easy the software platform is adaptable to new situations through the implementation of a new prototype configuration or new software functions from other Myopro projects. The modularity of the system is determined by several points.

- Is it easy to change the prototype configuration or implement new software functions?
- If new software systems have to be implemented, is there any need to program new software?
- How much time is involved developing this?

Within this project it must be possible to integrate the new five-fingered prototype in an easy and quick way for testing purposes. In addition, other software functions which will be developed within the Myopro project should be implementable in the software platform with minimal or no need for programming and without changing the overall functionality of the software platform. This results in several requirements that have been listed below.

Description	Value		Unit
It must be possible to easily adjust the software platform for changes in the configuration of the prototype setup (number of finger, sensors, channels etc.)	Yes	=	-
It must be possible to implement the new five-fingered prototype	Yes	=	-
it must be possible to easily implement and change software systems without changing the overall functionality of the software platform	Yes	=	-
The framework for yet to develop functions should be constructed for easy implementation (including sEMG, feedback mechanism and low level controller)	Yes	=	-
All input and output signals between functions should be easily accessible in the main function of the software platform	Yes	=	-

Table 9

4.2.3 *STRUCTURE*

Beside the modularity requirements which have influence on the software architecture of the platform, there is a hard requirement for dividing the system in to a user influenced (high level) component and an automatic behaviour (low level) component.

- The high-level controller determines the automatic behaviour of the prosthesis based on several user input signals.
- Low-level control determines the forces the actuator should apply to move the fingers to the desired end position.

Figure 17

4.3 HIGH LEVEL CONTROLLER

As described above the high-level controller determines the automatic behaviour of the prosthesis based on input from the user. The requirements of this part of the system have been listed below

4.3.1 *GENERAL*

The general requirements of the high level controller can be seen below

Description	Value		Unit
The controller must be able to handle all the conditioned EMG data input	Yes	=	-
The state machine must be easily to modified or replaced	Yes	=	-
The system must become more intuitive	Yes	=	-
The controller must prevent a false true situation	Yes	=	-

Table 10

4.3.2 *INTUITIVE*

What makes the prosthetic system intuitive? This question is very important within the Myopro project and internship assignment because the intuitive level of the system determines whether the system will be used on a long term basis by the user. Although this requirement is very important, due to the subjective nature of the requirement it is difficult to describe these as 'hard requirements'. However, in this section an attempt is made to provide a measurable definition within the reference frame of the internship assignment.

When the user is able to control the system in an easy and non-fatiguing manner, it seems the system 'feels' what the user wants, and reacts to this in a reasonable way. In this case the user has a natural experience and wants to use the system, even on a long term basis. If this is the case the system can be called intuitive.

To control the system in a non-fatiguing way, the number of steps to execute a specific task (like grasping a bottle) should be as low as possible. In this case this implies the number of actions the user must perform before it is possible to grasp an object (called preshaping) or switch to another preshape. As the number of steps decreases the more intuitive the system becomes

The intuitiveness of the system is also influenced by the execution speed. When the system reacts too slowly the user does not have a natural experience. When this happens it is particularly difficult and exhausting to use. For this reason this is a very important requirement. The system must quickly react to input signals from the user and must execute preshaping as fast as possible.
Literature defines the threshold for reaction time to input signals from the user as 300 milliseconds, this means time from input signal from user to reaction from the system (fingers start moving). For preshaping of the hand a threshold value of 500 milliseconds is found.

Even if all the points mentioned above work perfectly, errors in the input data from the user would spoil the natural experience. This is caused through the EMG classifier which interprets the user data in an incorrect way. Therefore, it is necessary to prevent a false true situation developing where the system enters the wrong system state, for example the system moves suddenly to another preshape.

Secondly, when a faulty state is selected it is not possible to directly switch to the intended state. In the current system the user has to wait until the system has completed the execution of the task and start all over again. It has to be possible to switch states during the execution of a task and without the need to start all over again. In this way the system should react more naturally to changes from the user. This leads to several design goals and requirements, listed below.

Design goals

- To select a state or grasp type, the number of user actions must be as low as possible
- To switch from one grasp type to another grasp type the number of steps must be as low as possible
- Preventing false true situations.

Requirements

Description	Value		Unit
The maximum delay time from input data to reaction from the system must be	300	<=	Ms
Preshaping must be completed within	500	<=	Ms
Develop a filter for false true situations through faulty input data from the EMG	Yes	=	-
Switching time between different system states must be as short as possible	Yes	=	-
Implement the possibility to switch from state during preshaping without the need to start all over with preshaping	Yes	=	-

Table 11

Data input

The state machine must be able to process all the possible input signals and combinations thereof. Therefore the system has to react in a reasonable way (e.g. system should operate by following the requirements and should be human friendly) and the system should know what the user wants to do with as few input signals as possible-

Grasptype and transitions

It must be possible to add, remove or change grasptype and transitions types very easily. In this way the system should be easy to use and adapt.

4.3.3 LOW LEVEL CONTROL

Low-level control determines the forces required by the actuator to move the fingers to the desired end position. The motion control has not been developed during my internship. However, for the implementation of the new software platform it is necessary to implement the framework for those controllers. Through this all input and output signals will present on the software platform. In this way it should be easy to implement these controllers in the future

Description	Value		unit
Implement framework for the low level controller	Yes	=	-
All input and output signals have to be present in the software platform	Yes	=	-

Table 12

4.4 FUNCTIONAL DESIGN

Before the system architecture will be developed it is necessary to obtain a clear picture of what kind of activity the user would like to perform with the system and what the functionality of the new system should be to meet the described requirements and goals. Once this is defined it is possible to develop the functional design for the new test system.

Preferably the user wants to have all the functionality of a human hand in his prosthesis. Unfortunately this is not possible with currently available technology. On these grounds it was decided to offer the functionality which enables the user to once again execute daily life activities. As such the prosthetic system will not be designed for sporting activities or for the performance of hard physical work.

4.4.1 ACTION ANALYSIS

To offer the right functionality to perform daily activities and meet the requirements, it is necessary to determine the actions and processes of a user with a human hand. The investigated user cases can be seen below.

- Grasping or picking up a cup.
- Opening a bottle of water.
- Grasping for a key, key card or pencil.

Based on these empirical observations, the user cases show great similarity in the actions and process that took place. For this reason only the first user case will be explained in more detail. This should be sufficient to obtain a clear picture of the actions and processes and make it be possible to define the functional design for the new test system.

1. The user does nothing.
2. Then user wants to pick up the cup and several different actions occur.
    a. First the user will focus his attention on the cup and has visual feedback about the position and shape of the cup
    b. The user gives signals to the muscles in the arm and starts to move his arm towards the cup.
    c. In the meantime the user will automatically shape his hand in the best shape to pick up the cup (preshaping).
3. If the user makes a mistake several other actions occur
    a. The user looks at what happens and where the objects are. After this the user reacts in the best possible way.
    b. If the user grasps incorrectly the user will retreat his arm and change the shape of the hand.
    c. Then the user tries to grasp for the cup again.
    d. If this does not help he has to clean the mess.
4. When the hand reaches the cup the user gets sensory and visual feedback about the position, forces, slip, temperature etc.
5. Then the user grasps the cup and does this until the user feels that the cup is not sliding away.
6. Now the user could bring the cup to his mouth. In the meantime forces on the cup are adjusted automatically by the user body to hold enough grip on the cup.

4.4.2 SYSTEM STATES

Based on the analysis above it is possible to differentiate several distinct functions for the new functional design.

First the user does nothing, and it is assumed that the hand also does nothing. This *'I do nothing'* activity can be seen as the first function of the system. In this state the system waits for input from the user to execute a task. This state is called the neutral state

The second function that can be defined is the shaping of the hand during grasping for the cup. This function controls the automatic shaping of the hand in a specific pre-programmed shape called 'grasptype'. This state will be called the preshaping state.

The third function is the interaction with the environment, in this case the cup. This function enables the user to open and close the hand. In this way it is possible to firmly hold the cup. This is called the grasping state. Although this is not a part of the assignment, the feedback mechanism provides feedback about the forces exerted on the cup and the slip detection prevents slipping away of the cup, like a human hand.

Then there is the situation where the user or system makes a fault. In this situation the user tries to adjust the shape of the hand and position of the arm, and grasp the cup again. With a human hand this is easily accomplished – a person just has to think how to change the shape of the hand, observe the position of the objects and try again (I would say that this is 'unconscious?).

Unfortunately this behaviour is not implemented in the current system. The user has to wait until preshaping is completed and then select the new preshape and wait again until the system is finished with preshaping. This behaviour is not user friendly and very tiring for the user.

With the implementation of a so called transition state it should be possible to switch from preshape even if the current preshape is not complete. This allows the user to quickly adjust to changes and give the user a more intuitive and natural experience with the system.
In the case where the system makes a fault and a false true situation occurs it is now possible for the user to quickly adjust the preshape of the hand.

4.4.3 CONTROL

Determining who has the control in certain states is the last step for completely defining the functional design. With the current technology it is not possible to control every finger independently. As described in the requirements it was decided to divide the system in two parts, namely a user controlled part and automatic behaviour part. So shaping every finger of the hand, e.g. setting the joint locks in the right configurations and to the desired end position, will be done automatically. The other functions can be controlled by the user within the limits of the selected preshape.

Table 13 shows a short summery of the defined system states and who is controlling this state.

System state	Control	Function
Neutral	User	In this state the system waits for input of the user to do something
Grasping	User	In this state the user can open and close the unlocked fingers of the hand according to a pre-programmed shape of the hand.
Preshaping	System	This state controls the automatic shaping of the hand according to a pre-programmed shape
Transition	System	This state Allows the user to switch from preshape during preshaping

Table 13

4.4.4 RELATIONS

Now that all the systems states have been defined it is important to define the relationship between the different states in a functional way. The technical details will be explained later.

Neutral state:
When the system is booted it will start in this state. The system waits until the user gives the first command to do something. In this state the user has three options

1. The user does nothing so the system will wait for input.
1. The user wants to shape the hand in a certain pre-programmed shape (preshaping) and the system goes to preshaping state.
2. The system is in a certain preshape and the user wants to open and close the hand. In this case the system goes to the grasping state.

Preshaping state:
The user wants to grasp an object. Before this the system has to shape the hand in a certain pre-programmed shape. In this state the user has four options.

1. The user waits until the system is ready with preshaping.
2. If the user has selected the wrong grasptype while the system is busy with preshaping the system goes to the transition state and shape the hand for the new preshape.
3. When the system has completed preshaping and the user wants to open or close the hand. In this case the system goes to the grasping state.
4. When the user is done with preshaping and for example has to wait for a while to pick up the cup. In this case the system goes to the neutral state.

Transition state:
If the system is busy with preshaping the hand and the user wants to change the preshape the system moves to the transition state. In this case the user has one option: in the current system the user has to wait until the system completes the transition from the old preshape to the new preshape.

Grasping state:
When the user has completed preshaping and wants to open or close the hand to grasp an object, the system goes to the grasping state. In this state the user has only two options, namely:

1. The user can stay in the grasping state and open or close the hand.
2. The user wants to change the preshape of the hand and the system goes to the preshaping state.

In the table below a visual representation of the relations can be seen. The vertical row shows the beginning state and in the horizontal row shows the new state. Where the x is filled in it is a valid transition, all other transitions are not possible with the current design.

	Neutral	**Preshaping**	**Grasping**	**Transition**
Neutral	**x**	**x**	**x**	
Preshaping	**x**	**x**	**x**	**x**
Grasping			**x**	
Transition		**x**		**x**

Table 14

These transitions are not possible for several reasons. It is for security against false true situations, comprehensibility of the system for the user and to make the system more intuitive.

Now all the system states and relations have been defined it is possible to determine the final functional design. This can be seen in the picture below with all compulsory parts.

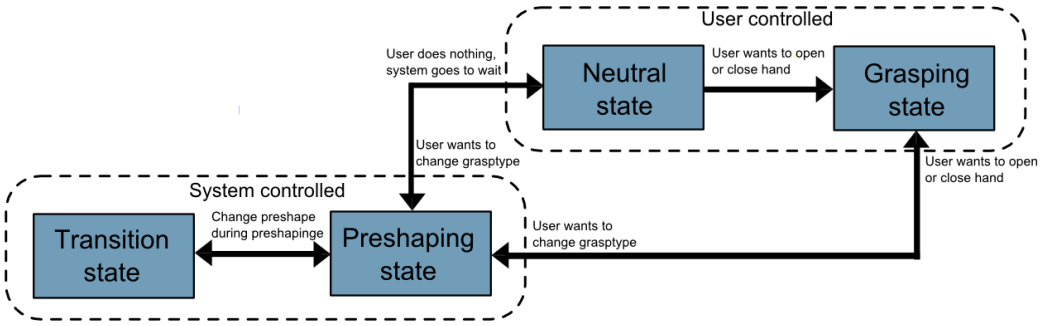


Figure 18

5 SYSTEM ARCHITECTURE

5.1 SYSTEM OVERVIEW

Based on the requirements and functional design it was possible to develop a global system architecture. This can be seen in the picture below, the blue parts were developed within my internship assignment. However, the implementation of the low level controllers will be done in further projects.

The state divided in a high level and low level controller to meet the requirement as described above. A total overview of the system architecture can be found in appendix one

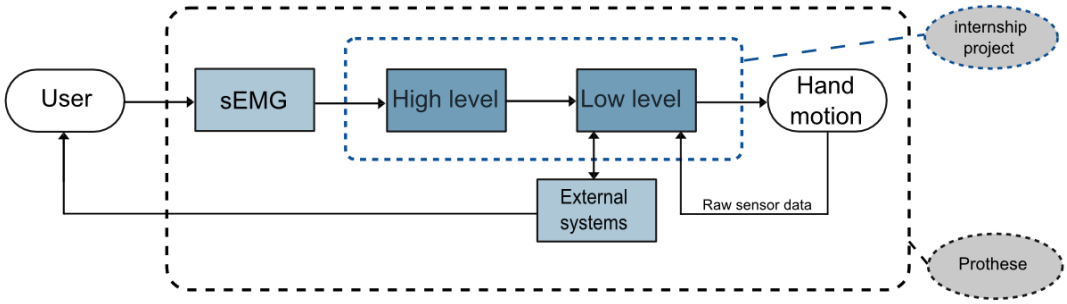


Figure 19

5.1.1 USER

The user is the person who uses the prostheses. Through muscle tissue activities the user gives signals to the system and receives feedback from the feedback mechanism which will be located in the external systems block (not part of the assignment).

Input	Output
• Feedback signals about the exerted force on an object (not part of the assignment	• Muscle tissue signals for the EMG classifier

Table 15

5.1.2 EMG CLASSIFIER

This function translates the muscle tissue activity into digital information. This data is collected by electrodes attached to the user's skin and conditioned with an EMG classifier. The development of this function is not part of my assignment. Currently this is developed by another Myopro project group.

However, the specifications for the output signals of the EMG classifier are known. Only the signals which have influence on my assignment will be explained.

Input	Output
• Electrical signals with information about the muscle activity	• Digital EMG data output

Table 16

5.1.3 HIGH LEVEL

The high level controller processes the user input and produces set points for the low level controller. The tasks of the high level controller have been listed below

- Processing and filtering the data from the EMG classifier, and preventing false true situations.
- Making the EMG data usable for the system.
- Processing variables
- Determining the state of the system (state machine).
- Generating set points for low level controller

Input	Output
• Digital EMG data • Information about the position of the fingers	• Set points for the low level controller

Table 17

5.1.4 LOW LEVEL

The low level controller performs the tasks which cannot be influenced by the user and make all the data accessible for the external systems. The tasks have been listed below:

- Controlling the movements of the hand to the desired end positions during preshaping and grasping.
    - During preshaping this means controlling the joint locks and speed with a controller.
    - During grasping this means controlling the forces exerted on object by the actuators, speed and direction of the movement
- Making all the available data accessible for the external systems.
- Processing all the sensory data in such a way that it is useable for the system.

Input	Output
• Set points from the high level controller • Raw sensor data	• Output data for the hardware controller of the motor and joint locks • Position data of the fingers for the high level controller • All available data for the use in external systems

Table 18

5.1.5 HAND MOTION

This block represents the hardware controllers, sensors and hand design of the hand.

Input	Output

• Controller information	• Sensory information
	  ○ Motor
	  ○ Biotac
	  ○ Hall sensors

Table 19

5.1.6 EXTERNAL SYSTEMS

This block represents the systems which will be made by other Myopro project such as the slip detection and feedback mechanism.

Input	Output
• All available data	• To be determined

Table 20

5.2 HIGH LEVEL CONTROLLER

The figure below shows the decomposition of the high level controller. The functions and interfaces are explained below.

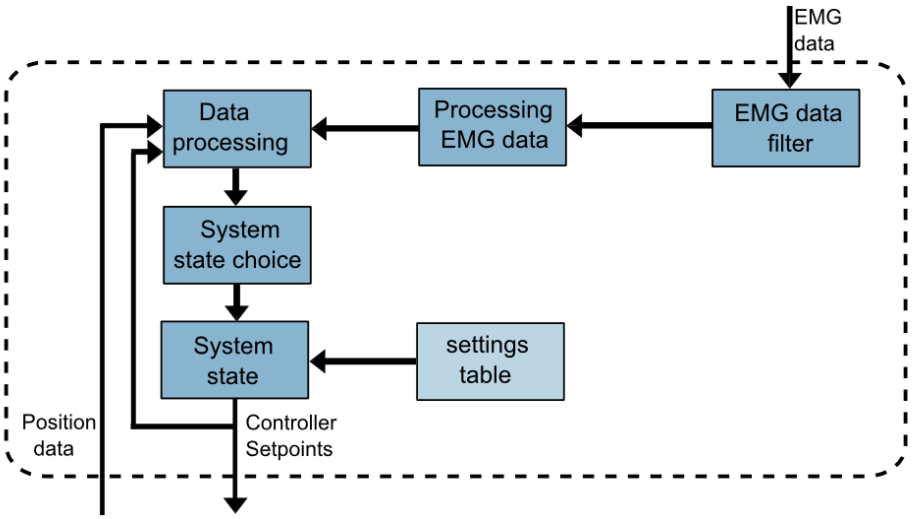


Figure 20

5.2.1 EMG DATA FILTER

The goal of this function is to prevent false true situations. Through inconsistency in the conditioned EMG data it is necessary to filter this data on false true situations. In this way it is possible to generate reliable input data for the system.

Input	**Output**
• Parameter 3: this signal the user selects a grasptype • Parameter 4 : this signal allows the user to give an open or close signal for the hand	• Same data but filtered

Table 21

5.2.2 PROCESSING USER INPUT

This function translates the EMG input into useable variables for the system. Essentially this is an API block to connect the EMG classifier to the system. However, this block is currently empty and all processing is done by data processing.

Input	**Output**
• Filtered EMG data	• Grasptype • Grasptype_change • Open/close • Intensity

Table 22

5.2.3 *DATA PROCESSING*

Based on all the input data, from the low level controller (finger position information), EMG data (users intention) and feedback on the current state of the system this function updates and calculates all required variables

By implementing the processing of the variables in one function block several improvements can be implemented.

1. The modularity of the system is increased. If the processing behaviour of a variable has to be changed, you only have to search in one place and not several different places. In addition all other function become easily to adapt because they only process the input of the processed variables
2. The software becomes easy to read for other people.
3. All variables are processed in the same place at the same time. In this way the variable update process is synchronized it is not possible incorrectly update a variable and cause dangerous behaviour. Besides this, the busses which communicate with the external system cannot give old or incorrect information.

In the chapter below only the most important variables and the processing of these variables have been explained. For all the other variables see the documentation in the software. In Table 23 all the input signals that will be processed have been listed. All needed variables to make the processing possible have been listed in Table 24.

Processed variables input

Name	Function
System state	Determines the current and new state of the system
System_state_change	Detects whether the system state is changed
Trans_state	Determines the state of transition
Trans_done	Determines if the system is done with transition
Pre_extension	Determines if all fingers have been in extension
Pre_state	Determines the state of preshaping
Pre_done	Determines if preshaping is completed
Grasptype	Selected pre-programmed grasptype the user
Grasptype_change	Detects whether the grasptype is changed
Open_close	Detects if the user gives an open/neutral/close signal
Signal_intensity	Determines the intensity of the signal for the grasping state
Grasping_opened	Determines if the user has given an open signal in the grasping state

Table 23

Secondary input

Name	Function
joint pos simple	This variable provides information about the position of every joint.
Preshape_settings	This array contains the desired position for every joint of the prosthetic hand
Pre_control settings	This array contains the joint lock and motor settings for preshaping the prosthetic hand
Extension_settings	This array tells at which angle the system is in extension. Through this variable it is possible to easily adapt the extension angle.
Pre extension	This variable tells the system of all joints have been in extension. This means all the joint locks are unlocked, after this it is possible to preshape the hand
Pre_jointlock_error	With this array it is possible to count the number of times the system has not locked the joints correctly. And it is possible to say how many times the system may retry before an error signal is given.

Table 24

5.2.3.1 Input & output variables

In this chapter all the values of the important variables will be explained.

System state

This variable is an integer and contains the system state information. The values can be seen in Table 25.

Value	Meaning
1	Neutral
2	Pre-shaping
3	Grasping
4	Transition
5	Error

Table 25

System state change

This variable is a Boolean and shows if there is a change in the system state. The values can be inTable 26

Value	Meaning
0	No system state change
1	There is a system state change

Table 26

Trans state

This variable should contain similar information as the pre state variable although this is currently not implemented

Trans done

This variable is a Boolean and shows if the system is busy with making a transition from the current preshape to the new preshape, this process occurs within the transition state.

value	meaning
0	Busy with transition
1	Done with transition

Table 27

Grasptype

This variable shows which grasptype the user has selected. This can be as may integers as you want but within my assignment the following values are possible.

value	meaning
1	Lateral grasp
2	Cylinder grasp

Table 28

Grasping opened

This variable is a Boolean and show if the user has given an open signal during the grasping state. This is an extra safety mechanism to prevent a unexpected change of preshape, through a false true situation in the EMG data.

Value	Meaning
0	No open signal is given
1	Open signal is given

Table 29

Grasptype_change

This variable is a Boolean and shows if the user changed the grasptype. The possible values can be seen below

Value	Meaning
0	No Grasptype change
1	Grasptype change

Table 30

Pre extension

This variable is a Boolean and shows if all the fingers have been in fully extension. This is very important because all joint locks have to be unlocked before starting to preshape.

Value	Meaning
0	Not extended
1	Extended

Table 31

Pre state

If the system is in the preshaping state, the user wants to shape the prosthetic hand in one of the pre-programmed shapes. Through the hardware configuration of the test system a specific path has to be followed to preshape the hand this variable declares in which state preshaping is

Value	Meaning
0	Go to extension, this is necessary in order to unlock all the joint locks
1	Turn on the joint locks and motor according to the settings table. In this way it is possible to lock the right configuration.
2	Turn off the joint locks. Through the hardware configuration it is possible to turn the joint locks off and keep the joints locked
3	Error situation. In this state the motor and joint locks are turned off. (safety)

Table 32

Pre done

This variable is a Boolean and shows whether the system is done with preshaping, thus if the hand is in the desired end position.

Value	Meaning
0	Busy preshaping
1	Done with preshaping

Table 33

Open close

This variable is an integer and shows whether the user wants to open, close or do not want to move the hand.

Value	Meaning
0	Neutral
1	Open
2	Closing

Table 34

Signal intensity

This is currently not implemented

5.2.3.2 Processing of the variables

In this chapter the processing of the important variables will be explained.

System state

This function stores the old value and updates the variable with the new value

System state change

This function stores the old value and updates the variable with the new value and checks if a system state change occurs based on the table below.

Old function state == new function state	System state change
Yes	No
No	Yes

Table 35

Trans done

Only a basic definition of this variable has been implemented in the software due to time constrains. Basically when the system is in the transition state it waits until the system is in the desired position (preshape settings are used). In this way it is possible to manipulate the system for testing purposes. The current definition can be seen in the table below.

Joint_pos_simple == preshape setting	Current Trans_done	New Trans_done
no	0	0
Yes	0	1
Don't matter	1	1

Table 36

Grasptype

This function stores the old value and updates the variable with new data. Currently this function converts the simulated EMG data into data for the system. This must become a separated function, although it is currently implemented in the processing function.

Type	EMG value	System value
Lateral grasp	1	1
Cylinder	2	2

Table 37

Grasping opened

This function stores the old data and updates the variable with new data. This function checks if the user has given a open signal when the system is in grasping state. This is a safety mechanism to prevent undesirable changes in the preshape though false true data from the EMG classifier.

System_state	Open/close	Current Grasping opened	New Grasping opened
2	Open	0	1
2	Don't matter	1	1
~2	Don't matter	Don't matter	0

Table 38

Grasptype change

This functions stores the old data and updates the variable with the new data this function also checks if a grasptype change occurs based on the table below.

Old grasptype == new grasptype	Grasptype change
Yes	No
No	Yes

Table 39

Pre extension (preshaping state)

This function stores the old value and updates the variable with the new value. In addition this function check if all fingers and thumb (depending on the current configuration) have been in fully extension. Through the hardware design it is necessary to do this, when this is true the system can start with preshaping or transitions. See the table below for the conditions

System state	Joint_pos_simple == extension settings	Current Pre_extension	New Pre_extension
2	Don't matter	Don't matter	0
2	No	0	0
2	Yes	0	1
~2	Don't matter	Don't matter	0

Table 40

Pre state

This function stores the old value and updates the variable with the new value. Based on the table below this function controls the preshaping state of the hand (as explained before). This function will only be checked in the preshaping state.

System_state	Pre_done	Pre_extension	Current Pre_state	New Pre_state
2	0	0	0	0
2	0	1	0	1
2	0	1	1	2
2	1	Don't matter	Don't matter	0
~2	Don't matter	Don't matter	Don't matter	0
All	Not	Defined	Situation	3

Table 41

Open/close

This function stores the old data and updates the variable with the new data. This function converts the open/close signals from the simulated EMG classifier into a usable variable for the rest of the system. In the further this function should be moved to the processing user input function.

Type	EMG value	System value
Open	-1	2
neutral	0	0
close	1	1

Table 42

Pre done & joint lock checking

This function stores the old value and updates the status of the variable. In addition this function checks several conditions to see if the system is done with preshaping.

- The functions check if the system has unlocked al the joint locks (pre extension)
- Checks all individual joints (joint pos simple), are in the desired position (pre settings)
- How many times a faulty situation has occurred, though not properly locking of the joints.

Retries < maximum	Preshape_settings = joint pos simple	Pre_extension	Current pre state	Current pre done	New pre done
yes	no	0	Don't matter	0	0
yes	no	1	Don't matter	0	0
Yes	Yes	0	Don't matter	0	0
Yes	Yes	1	2	0	1
Yes	Yes	1	0 or 1	0	0
No	Don't matter	Don't matter	Don't matter	0	error

Table 43

Intensity signal

This is currently not implemented. This function should process the intensity data from the EMG classifier and convert this to speed and force signals for the low level controller.

Trans state

Currently not implemented, due to time constraints.

5.2.4 SETTINGS TABLE

This block contains all the settings tables for all the system variables and states and will be explained below.

Input	Output
• None	• preshape settings

5.2.5 STATE MACHINE

Based on input data from the processing block a decision is made in which state the system should be. All details about the functional behaviour of the system can be found in the functional design chapter. The table from the functional design chapter is shown below, Just to clarify what the system states are and who is controlling them.

System state	Control	Function
Neutral	User	In this state the system waits for input of the user to do something
Grasping	User	In this state the user can open and close the unlocked fingers of the hand according to a pre-programmed shape of the hand.
Preshaping	System	This state controls the automatic shaping of the hand according to a pre-programmed shapes
Transition	System	Allows the user to switch from preshape during preshaping

Table 44

Based on the functional design the state machine has been formulated in a technical manner. This figure with all the technical conditions can be seen below. To meet the requirements, improve the intuitiveness and feasibility of the software several new (technical conditions) have been implemented in the state machine.

- An extra system state has been implemented (error state). When the system detects a fault in any way which cannot be solved the system goes to the error state and stops the motion of the prosthesis for safety reasons.
- Secondly the neutral state has been implemented in such a way that's possible to switch back to preshaping in a very fast way
- Further on a false true security have been implemented:
    - In the grasping state. The user has to give an open/close signal and a grasptype change to go to the preshaping state.
    - When the transition from the neutral state to the grasping state takes place it is not possible to go back to the neutral state. In this way unwanted behaviour is avoided.

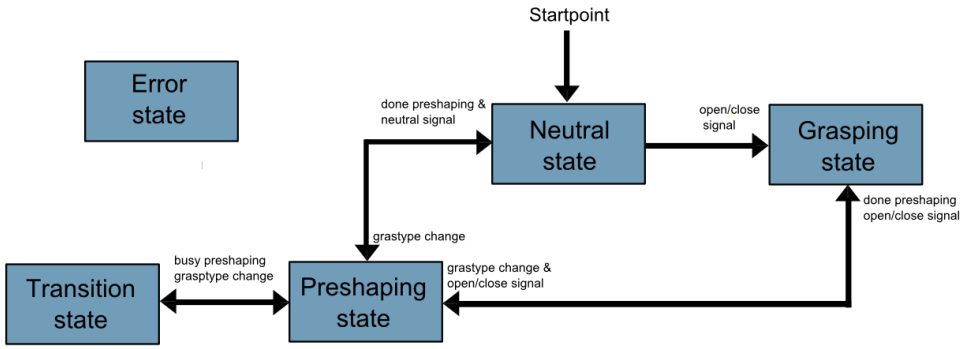


Figure 21

Now the state machine is defined it is possible to construct the technical choice table for all conditions and system states. Because the size of this table is very large it can be found in appendix 4

To make the choice for the correct system state several input signals are needed.
A detailed explanation of every variable and its processing can be found in the data processing chapter.

Input	Output
• System_state • Trans_done • Grasping_opened • Grasptype_change • Pre_done • Open_close	• System_state

Table 45

5.2.6 SYSTEM STATE

Based on the decision of the state machine this block executes one of the defined system states. Basically these states only pass on data from the setting tables based on data from the processing block. Because of this reason the functions are easy to read/modify. A description of inputs outputs and the sub state machines can be found in the chapters below. Details about the involved variables can be found in the functional design, processing and state machine chapters.

Preshaping

This state provides set points for low level controller. These set points contain information about the joint locks and actuators. In addition the preshaping state is divided in several stages that have been explained in the processing chapter. Inputs/outputs can be seen in the table below.

Input	Output
• pre_state • pre_setting_table • pre control settings	• jointlock setpoints • motor controller setpoints

Transition

This state controls the transition pattern from 1 preshape to a new preshape when preshaping hasn't completed. The idea is to us a settings table with settings for the most effective transition path for every possible preshape combination. In addition this process will be divided in several steps just like the preshaping state. But the framework is implemented currently.

Input	Output
• trans_state • trans_settings (not implemented)	• jointlock set points • motor controller set points

Neutral

This state is a wait and hold function, it waits for input from the user to perform a certain action. In addition it sends data to turn off the motor and joint locks for safety purposes.

Input	Output
• none	• jointlock set points • motor controller set points

Grasping

This state provides the low level controller with the necessary data from the user. Based on these signals the user is able to manipulate the motion speed and movement direction. In addition it is possible to control exerted force on an object. The framework for this state is implemented but further development is necessary.

Input	Output
• open/close • intensity signal (not implemented)	• direction signal • speed and force signal

Error

This state handles errors. This means if something occurs what the system couldn't solve or process this state should prevent dangerous behaviour. The current system only goes to the error state when the system has an unknown dataset for the state machine. This should be extended with in the future. All other errors give only an error message on the screen.

Input	Output
• none	• none

5.3 LOW LEVEL

This chapter describes the functions which have been located in the low level controller.

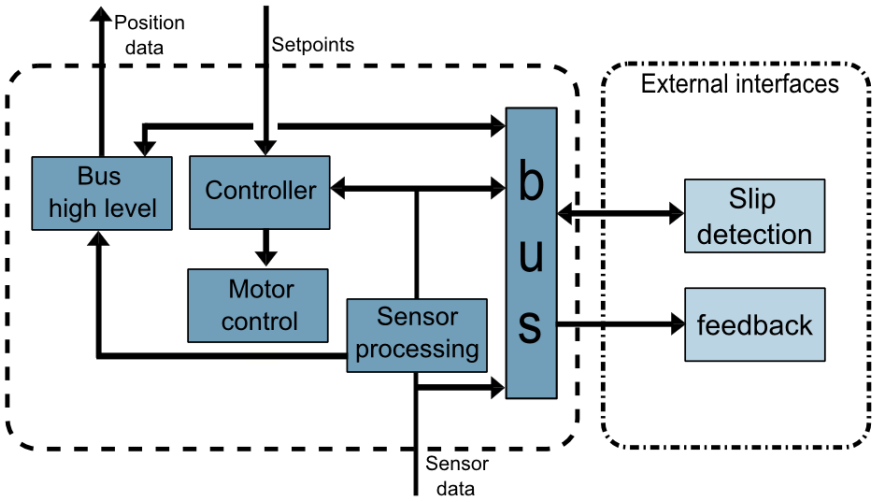


Figure 22

5.3.1 CONTROLLER

This function controls the motion pattern for every individual finger. The current system consists of 2 separated controllers:

1. An position controller that controllers the motion during preshaping, transition, error and neutral state
2. A second controller that controls the motions during the grasping state.

There is chosen to do this because of several reasons. During preshaping it is assumed there is no interaction with the environment and precision movement is not necessary. Because of this it is possible to implement a simple position controller. The second controller is necessary when the user controls the system to grasp a object. In this state a controller with high precision and which can handle disturbance signals cause by contact with objects. Because of this reason a controller like an impedance controller is necessary. In addition it is possible to manipulate the force in an easy way

Implementation of the second controller is not part of the internship. Only the framework and the kinematical calculations and visualization of this function will be realized. The kinematics will be explained in the implementation chapter. All the input and output signals have been listed below.

Input	Output

• preshaping/neutral/transition/error
  ○ desired end position of fingers
• Grasping
  ○ Current positions of the joints
  ○ direction signal
  ○ intensity signal
	• Motor control signals
	  ○ Motor speed
	  ○ Motor direction
	  ○ Motor force
	• Jointlock signals

Table 46

5.3.2 MOTOR CONTROL

This function translates output data from the controller function into data for the hardware controllers such as the actuators and joint locks and sends this data to the correct channels.

Input	Output
• Motor setpoints • Joinlock setpoints • Configurations settings	• Motor controller signals • Jointlock controller signals

Table 47

5.3.3 *PROCESSING SENSOR DATA*

This function converts raw sensor data into data which the system understands. The datasheets contains the exact layout for the output signals for all the sensors.

Input	Output
• Sensory data from the Biotac • Sensory data from the hall sensors • data from the motor controller	• Actuator force • Actuator direction • Angle data of every joint • Force magnitude and direction • High frequency spectrum of contact surface

Table 48

5.3.4 BUS

This function makes the information from the low level controller available to the high level controller and external systems. Below the two busses will be explained

Output external systems

Development of software and hardware functions is currently not finished. Because of this not all inputs and outputs have been defined. In this way all system variables have been made available to this function.

Output high level

This function processes the raw hall sensor data into discrete angle values (0, 1 and 2). for the high level controller. Because these sensors provide inaccurate output there is chosen to use discrete values. In addition if the system was restarted the output values where different for the same angles. To find an average output value for specific angles several angle measurement test have been conducted. The left chart shows average angle versus voltage output and the right picture shows the used setup to measure this.

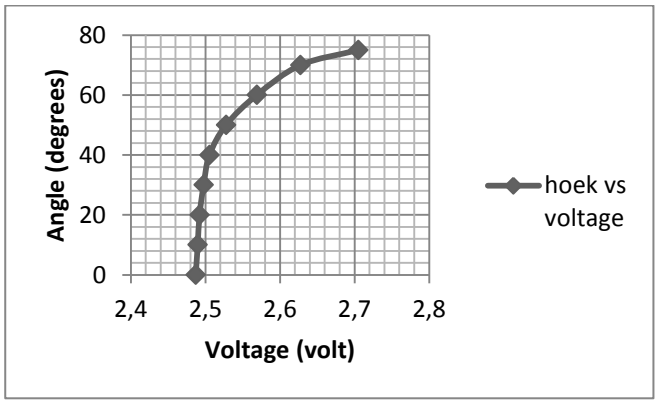


chart 1

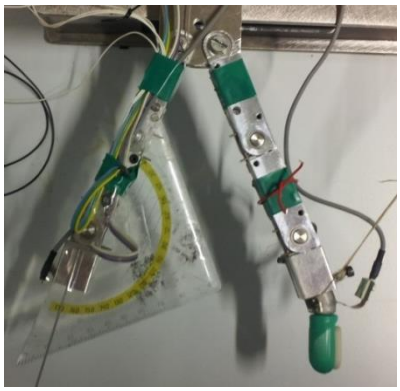


Figure 23

Based on these tests set points have been determined for the sensors. But with a configuration array it was possible to compensate for the different output values. Although this method is not desirable it was possible to get an idea what the position of the finger was. This was enough for the position controller but delivered very poor results for the kinematical calculations which will be used in the low level controller. Future research should include improving these hardware and software systems, to realize reliable angle output data. The used angle values can be seen in the table below.

Position	Discrete -value	Angle (degrees)
Extension	0	0
In between	1	~40
Flexion	2	~70

Table 49

The table below shows the inputs and outputs of this block.

Input	Output
• Hall_calibration • Configuration settings	• Jointlock_pos_simple

Table 50

5.4 IMPLEMENTATIONS

Before developing the software several implementation decisions have been made to improve the overall functionality and performance of the system and to comply with the requirements. These decisions will be explained in this chapter.

5.4.1 TEST SYSTEM INTEGRATION

A 2 fingered test system was provided to use during the internship. Several hardware and mainly software changes had to be made. Because the test setup consisted of separated parts could not communicate with each other and were not able to process data from the various sensors at the same time.

To meet the requirements, all hardware/software components had to be integrated into 1 software environment which is connected to a Matlab environment. An image of the test system indicating relevant subsystems can be seen in the figure below. With the implementation of this system it is now possible to communicate with all hardware controllers in an easy to use way.

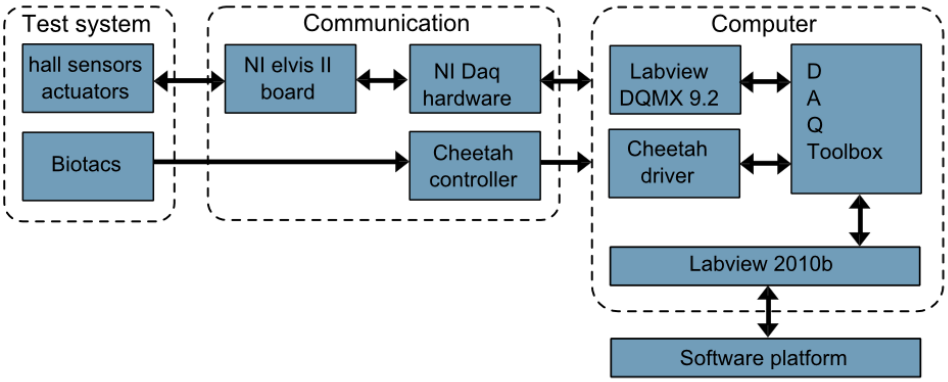


Figure 24

The actuation systems (DC motor and solenoids) and the sensing systems (Hall sensors, BioTacs) are connected to a National Instruments ELVIS DAQ device (National Instruments Corporation, Austin, TX, USA) and cheetah controller. These controllers are connected to the computer and controlled within a Matlab environment. (The MathWorks, Inc., Natick, MA, USA).

5.4.2 CONFIGURATION

With the implementation of more than 30 hardware configuration options the system has been made modular (adaptable to different test system configurations). In this way it is possible to change the configuration of the test system without any reprogramming. all variables have been listed below with a short description. The process to adapt to a new configuration is partially automated but this should be further developed future research.

Configuration settings	Type	Description
Fingers installed	Integer	How many fingers have been installed
Thumb_installed	Array	Is a thumb installed
Joinlocks_installed	Integer	How many joint locks have been installed
Jointlocks_per_finger	Array	How many joint locks have been installed in every individual finger
Hallsensors_installed	Integer	How many hall sensor have been installed
Hallsensor_per_finger	Array	How many hall sensors have been installed in every individual finger
Motors_installed	Integer	How many actuators have been installed
Finger_configuration	Array	Which fingers have been installed
Joinlock_configuration	Array	Where every joint lock have been installed in every finger
Hallsensor_configuration	Array	Where every hall sensor have been installed in every finger

Table 51

Controller settings	Type	Description
Ai_max_channel	Integer	How many analog input channels are available
Ao_max_channel	Integer	How many analog ouput channels are available
Dio_max_channel	Integer	How many digital channels are available
Ai_samplerate	Integer	The samplerate of the national instrument board for analog input
Ai_sampleratetrigger	Integer	The samplerate trigger of the national instrument board for analog input
Ao_samplerate	Integer	The samplerate of the national instrument board for analog output
Ai_channel		Contains information about which sensor on which channel have been installed
Ai_channel_name		The physical name of the analog input channels
Dio_channel		Contains information about which sensor on which channel have been installed
Dio_channel_name		The physical name of the digital output channels
Dio_channel_direction		Determines if the channel is a input or output channel
Dio_motor_channel		Contains information about which sensor on which channel have been installed
Dio_motor_direction		Determines if the channel is a input or output channel
Dio_motor_channel_name		The physical name of the digital output channels for the actuators
Ao_motor_channel		Contains information about which actuators have been connected to a specific channel
Ao_motor_channel_name		The physical name of the analog output channels for the actuators

Table 52

5.4.3 SOFTWARE ARCHITECTURE

Also the software is constructed in a modular way. All main and sub function are written as physically separated function. These functions communicate with one main program which controls the executions and update sequence. Because of this reason it is easy to adjust the systems behaviour just by dragging the functions to another place. In addition the software becomes easy to read as seen in the picture below.

```
while(1)
%//////////////////////////////////high level//////
emg_processing()
high_processing();

choice();
       switch system_vars(1,1)
            case 1
                 neutral();
            case 2
                preshaping();
            case 3
                grasping();
            case 4
                transition();
            case 5
                error();
            otherwise
       end

       databus_high_level();
%//////////////////////////////////high level//////
```

Figure 25

```
%//////////////////////////////////low level////////
        if system_vars(1,1) ~= 2
            interaction_control();
        else
            direct_control();
        end

        get_sensor_information();
        processing_data();

        databus_low_level();
%//////////////////////////////////low level////////
end
```

Figure 26

5.4.4 FUNCTION IMPLEMENTATION

Because the system is written in a modular way, there are a lot of setting options. Because of this all functions have been written in a generic way. This means all functions will automatically adapt when the settings change.

5.4.5 KINEMATICS

Because the low level controller uses an interaction controller or an equivalent to this it's necessary to calculate the end effector of every finger. Because the framework for this low level controller was a part of my internship all kinematics have been calculated and implemented in the software. In addition a sub function is added to view the hand motions in real-time.

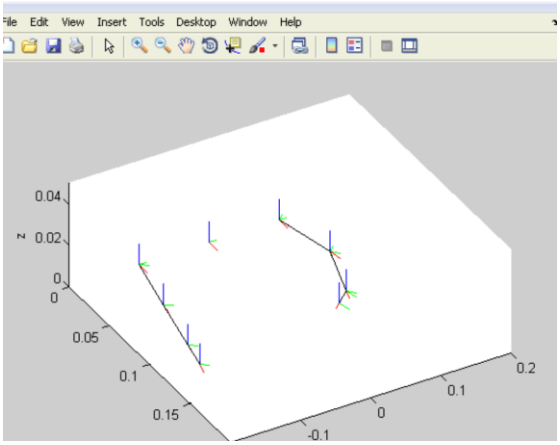


Figure 27

Currently a new design for the hand is under development because of this reason several dimension settings have been implemented in the software. In this way it's easy to adjust this function to the new design

Variable	Type	Description
Finger_dim	Floating	Contains the dimensions of every finger
Finger_reference	Floating	Describes the position of the reference point
display	Integer	Tells if a real-time plot should be shown or not
Display_view	Integer	Contains the angles at which the display is view

Table 53

5.4.6 HALL SENSORS

The current angle measurement of the joints is done with hall sensors. Also for these sensor a few settings have been made to quickly adjust the system in an easy way. Because there was lot of noise on the output data from the hall sensor a calibration table is added (values are different for every sensor). in addition a setting table is added to control what the desired end position for the fingers are in a specific preshape. Besides this new preshapes can be added in a very easy way.

Variable	Value	Description
extension_setting	Array	this array provides the threshold information about when the joints are in extension or flexion
Hall_calibration	Array	This array provides the threshold for every hall sensor in the test system
Pre_control_settings	Array	This array provides the desired end position of the fingers for every defined preshape

Table 54

5.4.7 CONTROL SYSTEM

At last there are settings to adjust the how the motors and joint locks should behave during preshaping the hand. This is done to simplify and shorten the time to adjust the systems behaviour.

Variable	Value	Description
Motor_control	Array	Provides information for the motor controller for every preshape
Joint_control	Array	Provides the information about what kind of joint locks have to be activated
Pre_time_delay	Array	Provides the necessary delay time to preshape the hand and turn of the joint locks and motors

Table 55

6 VALIDATION

6.1 EXPERIMENTAL SETUP

As explained in the current system chapter the experimental setup consists of the two-finger prototype, this system is used to conduct the experiments

6.2 TEST PROTOCOL

Several tests have been conducted to determine the performance of the new software system in various system states and actions. All tests have been conducted five times and the speed measurements have been done with the tic toc function in Matlab.

1. The first test was to determine the cycle time when the system waits on input signals from the user to do something.
2. The second test was performed to determine if the system responds correctly to the input signals and secondly how fast could the system preshape the hand and go to the preshaping state. When the test was started all the fingers were fully extended.

Further a validation of the most important requirements will be done to verify if the new test system meets these will be tested if they m the system meets the requirements that have been described earlier.

- System performance
- Modularity
- Integration
- Intuitive

6.3 RESULTS

6.3.1 SPEED TEST 1

The program was started and executed in the neutral state for 100 program cycles. The results in the table below show the average values of the 100 cycles; in the last column the average processing time of all functions and cycle time over the 5 test can be seen. As shown retrieving and processing data from the hall sensors and controlling the motors takes up most of the time cycle time.

The kinematics step shows 0 seconds, because this is a trivial implementation to visualize the motion of the fingers in real-time this function is disabled. Fluctuations in the cycle time can be caused through the tic toc function of Matlab. This should be evaluated more closely in the future. But overall the speed requirement of 300ms has been met!

		High processing	Choice	High level control	Low level control	Get hall data	Low high bus	Kinematics	Cycle time
Test	Cycles	t1	t2	t3	t4	t5	t6	t7	t8
1	100	0,00	0,00	0,00	4,80	6,40	4,90	0,00	16,30
2	100	0,00	0,00	0,00	4,80	6,40	4,90	0,00	16,30
3	100	0,01	0,02	0,02	4,89	6,54	4,90	0,00	16,39
4	100	0,01	0,02	0,02	4,85	6,40	4,86	0,00	16,16
5	100	0,01	0,02	0,02	4,89	6,30	4,90	0,00	16,15
Average value		**0,00**	**0,00**	**0,00**	**0,05**	**0,06**	**0,05**	**0,00**	**0,16**

Table 56

6.3.2 SPEED TEST 2

During the second test the system was booted and simulated user input was given. This data instructed the system to preshape the hand into the tripod preshape and go to the grasping state. The expected variable pattern should be similar to the description below.

Variable	State one	State two	State three
System state	Neutral	Preshaping	Grasping
Pre done	Done	Not done	Done
Pre state	1	1 → 2	1
Grasp change	Yes	No	No

Table 57

The figure below shows the important variables which have influence on preshaping the hand. A short description of all the variables can be seen in the table below. Extended information could be found in the system architecture chapter.

Variable	Description
System state	Contains information about the state of the system
Pre done	Tells if the system is done with preshaping 1 = done
Pre state	Tells in what state preshaping currently is
Grasp change	Tells the system if there is an change in grasptype

Table 58

The pattern which occurred was very similar to the expected pattern and validates that the system works according to the specifications. But further testing should be done to verify the correct behaviour of the state machine.

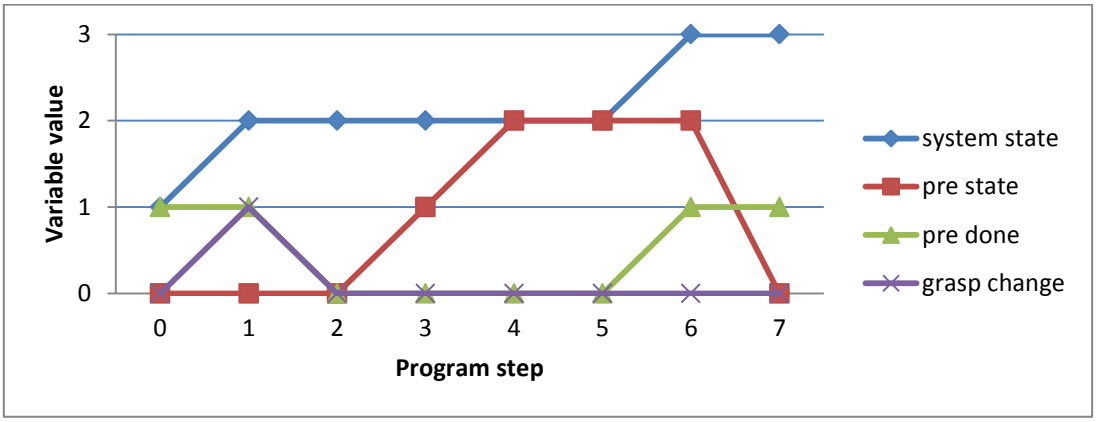


Chart 2

The figure below shows the position of every joint when the desired position is reached. Due to a lot of noise on the sensors there is chosen to use fixed values 0 is extended one is roughly 40 degrees and 2 are roughly 70 degrees. Further testing should be done to determine the performance of the position control.

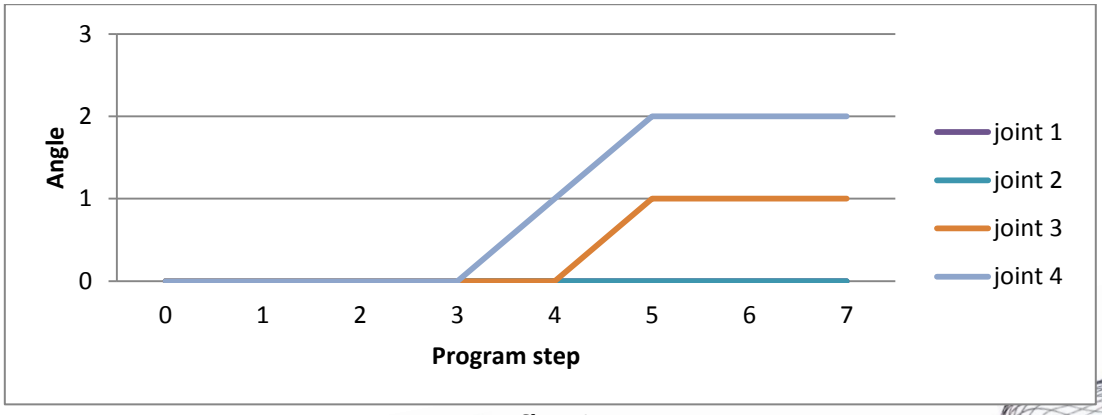


Chart 3

Further on the speed was measured during these tests. The figure below shows the average processing time of the functions for all the five tests. As stated before most of the time goes to retrieving sensor data and controlling the actuator. With the current configuration preshaping to the tripod shape takes 8,01 seconds. All other shapes have not been tested jet due to time constrains.

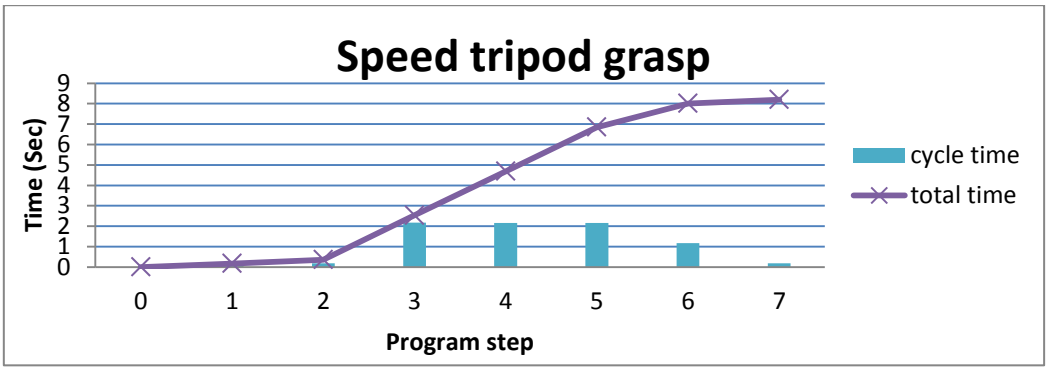


chart 4

6.4 REQUIREMENTS

6.4.1 SYSTEM PERFORMANCE

As seen in the second speed test the system react as predicted to the simulated input signals from the user and selects the correct system state. Some rough tests have been conducted to see how the system reacts on all kind of user signals. These tests where very promising but to verify the correct functionality of the state machine further testing should be conducted.

6.4.2 MODULARITY

Several different ports and configurations have been chosen and tested if they worked according to the specification and this seems to work. Besides this the system is very flexible (as described in the implementation chapter), there are a lot of possibility to adjust the test systems to different configurations

The Modular setup of software enables easy adaptation of the systems behaviour/configuration without any need of reprogramming functions and it is possible to easily implement new functions.
So this requirement should be met.

6.4.3 INTEGRATION

The separated systems have been integrated into one software platform. With the new software it is possible to communicate with all the sensors, control the actuators and exchange information between functions, thus this requirement is met.

6.4.4 INTUITIVE

Within this internship the intuitive requirement is defined in a specific way. The definition for this can be found in the requirements chapter. All the different factors will be explained below but have not been tested.

First factor was the needed steps to execute of specific task. With the new system this simplified, the user controls the system with a few variables and the amount of steps is reduced. Because the needed steps are reduced it should be easier to control the system.

Besides this the partially implementation of the transition state makes it possible to switch from preshape without waiting to finish preshape. This is a big improvement compared to the old situation. Through this the system should behave more natural. But the False true filter for the EMG data is not implemented jet and this should be done in future research.

The third factor is the executions speed. The reaction time of the system must be less than 300 milliseconds the current system has a maximum reaction time of 160 milliseconds.

7 CONCLUSION

The implementation of the improved state machine allows the user to execute a wide range of activities and provides a more natural experience through implementation of several new functions.

The new software platform led to an integrated test system which can process all data and controls the system in an effective way. Besides this the system speed is increased and through several implementations the software is highly modular. Therefore it is very easy to modify the test system to different hardware controllers, hardware configurations and behaviour settings.

Even dough not all functions and requirements have been met, a very good foundation has been developed for future research. In addition the new system is a huge improvement compared to the old system.

7.1 FUTURE WORK

Not all requirements have been met and some functions have to be implemented in the framework. This should improve the functionality and performance of the system drastically. Besides this a new angle measurement method should be develop to reduce the hysteresis and noise on the angle measurements sensors. To determine the performance further validation of the developed system should be done.

8 RECOMMENDATIONS

To improve the functionality and performance of the system several points should be improved during future research.

In the first place it is time consuming work to fill in and check all the settings. Besides this the current system assumes all the settings have been filled in correctly. This could be improved by implementing a graphical interface with an integrity check. In addition it is possible to partially automate the process which will save a lot of time.

Not all functions, low level controller, error state. Etc. has been implemented currently. This drastically limits the functionality of the system so this is a very important point to improve. In addition integration of the external systems functions should be tested.

Besides this a new angle measurement system should be implemented in the hardware because the current system just is not working as it should be. When this system is updated the high level and low level controller could perform optimal.
A possible solution could be the use of small encoders in the joints or strain gauges on the joints. the encoders can measure the position precisely but are relatively large. The strain gauges can do it with the same precision but can be difficult to attach to the joint. Another possibility is to use a linear encoder in the base of the finger and attach a rope with a spring on the end and measure the difference in force to calculate the angle. See the picture below for clarification.

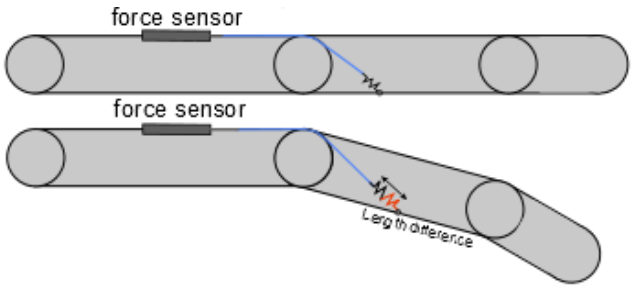


Figure 28

9 References

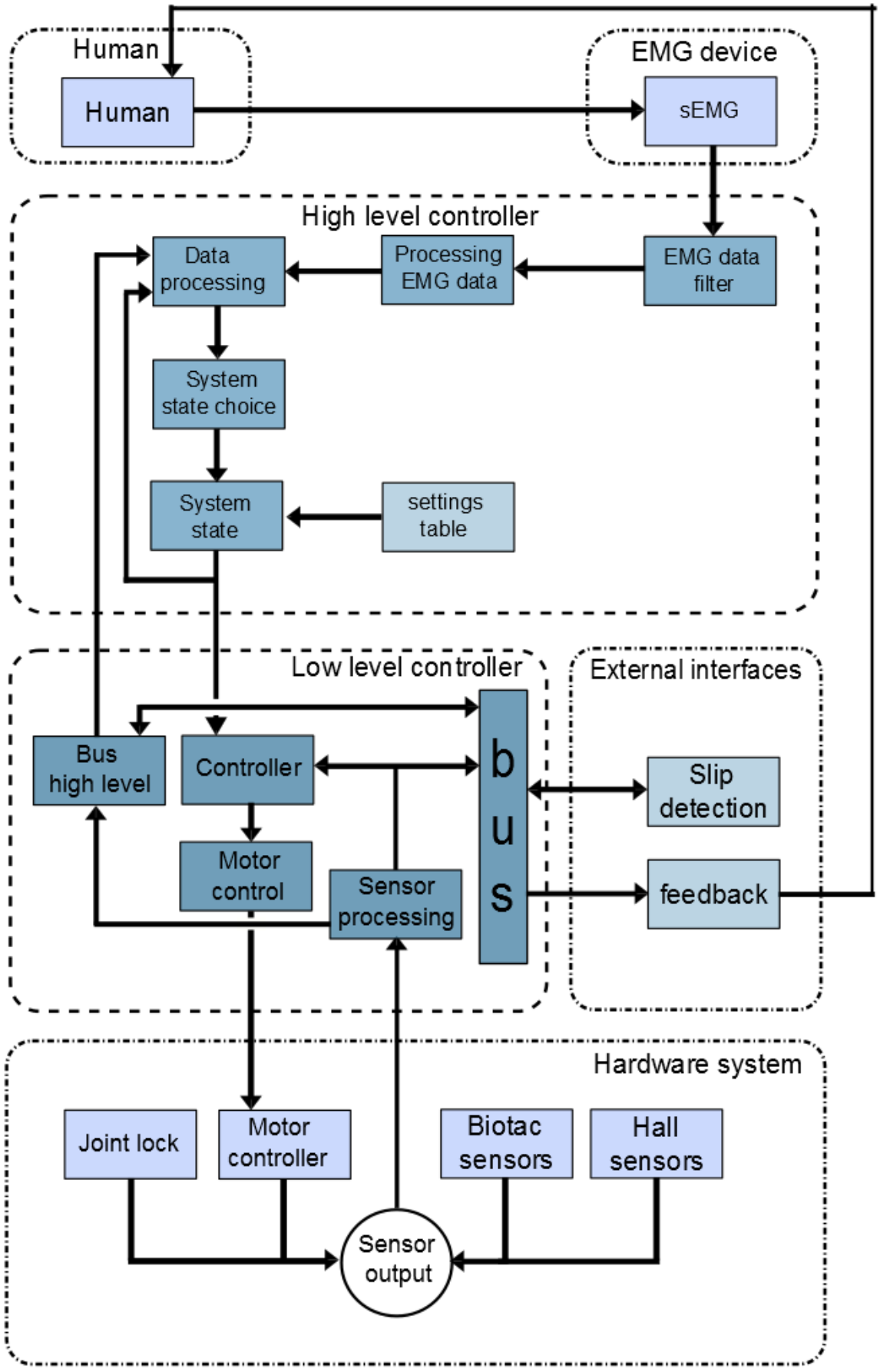
1. Peerdeman, B.; Boerey, D.; Kallenbergy, L.; Stramigioli, S.; Misra, S. A Biomechanical Model for the Development of Myoelectric Hand Prosthesis Control Systems. *Engineering in Medicine and Biology Society EMBC 2010 Annual International Conference of the IEEE* **2010**, *2010*, 519–523.

2. Peerdeman, B.; Fabrizi, U.; Palli, G.; Melchiorri, C.; Stramigioli, S.; Misra, S. Development of prosthesis grasp control systems on a robotic testbed. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics BioRob*; IEEE, 2012; pp. 1110–1115.

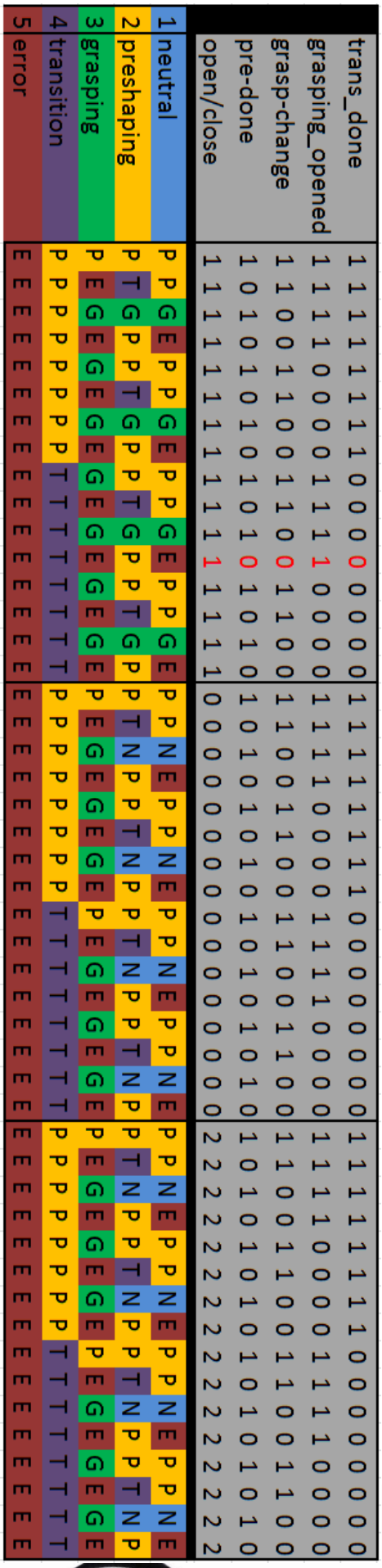
3. Peerdeman, B.; Pieterse, G. J.; Stramigioli, S.; Rietman, H.; Hekman, E.; Brouwer, D.; Misra, S. Design of joint locks for underactuated fingers. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics BioRob*; IEEE, 2012; pp. 488–493.

4. Peerdeman, B.; Stramigioli, S.; Hekman, E. E. G.; Brouwer, D. M.; Misra, S. Development of Underactuated Prosthetic Fingers with Joint Locking and Electromyographic Control. *Mechanical Engineering Research* **2013**, *3*, 130–142.

APPENDIX 1


Human
Human
EMG device
sEMG
High level controller
Data processing
Processing EMG data
EMG data filter
System state choice
System state
settings table
Low level controller
External interfaces
Bus high level
Controller
b
u
s
Slip detection
Motor control
Sensor processing
feedback
Hardware system
Joint lock
Motor controller
Biotac sensors
Hall sensors
Sensor output

APPENDIX 2


trans_done
grasping_opened
grasp-change
pre-done
open/close
1 neutral
2 preshaping
3 grasping
4 transition
5 error

APPENDIX 3

AGENDA PROJECT MEETING WEEK 47

topic:	Progress meeting Myopro WP4 project		
date:	20-11-2012	time:	10:00
place:	Enschede		
chairman:	Bart Peerdeman	secretary:	Peter Westenberg
present:	Bart Peerdeman, Peter Westenberg		

1. **Opening**

2. **Discussing minutes of the last meeting.**

3. **Received documents, announcements and questions.**
   -

4. **Progress**

5. **Other discussion points**
   -
   -
   -

6. **Summary of decisions**

7. **Summary of action points**

8. **Survey around the table**

9. **Ending the meeting**

MINUTES MEETING WEEK 47 (NOTES)

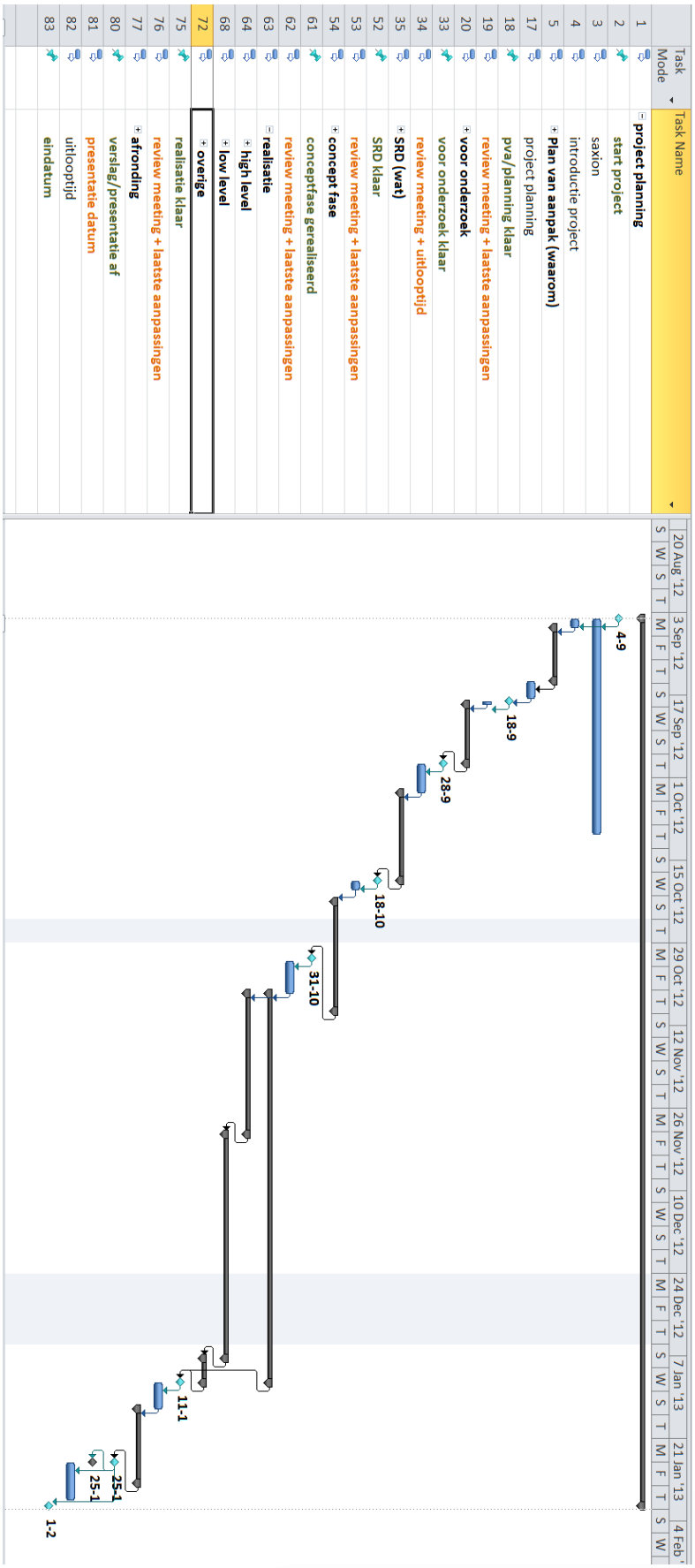
Date	:	20-11-12
Present	:	Bart Peerdeman, Peter Westenberg

ACTION POINTS

Action points	Person(s)	deadline

DECISIONS / ANNOUNCEMENTS

APPENDIX 4


Task Mode
Task Name
1 project planning
2 start project
3 saxion
4 introductie project
5 Plan van aanpak (waarom)
17 project planning
18 pva/planning klaar
19 review meeting + laatste aanpassingen
20 voor onderzoek
33 voor onderzoek klaar
34 review meeting + uitlooptijd
35 SRD (wat)
52 SRD klaar
53 review meeting + laatste aanpassingen
54 concept fase
61 conceptfase gerealiseerd
62 review meeting + laatste aanpassingen
63 realisatie
64 high level
68 low level
72 overige
75 realisatie klaar
76 review meeting + laatste aanpassingen
77 afronding
80 verslag/presentatie af
81 presentatie datum
82 uitlooptijd
83 eindatum
20 Aug '12
3 Sep '12
17 Sep '12
1 Oct '12
15 Oct '12
29 Oct '12
12 Nov '12
26 Nov '12
10 Dec '12
24 Dec '12
7 Jan '13
21 Jan '13
4 Feb '13
4-9
18-9
28-9
18-10
31-10
11-1
25-1
25-1
1-2