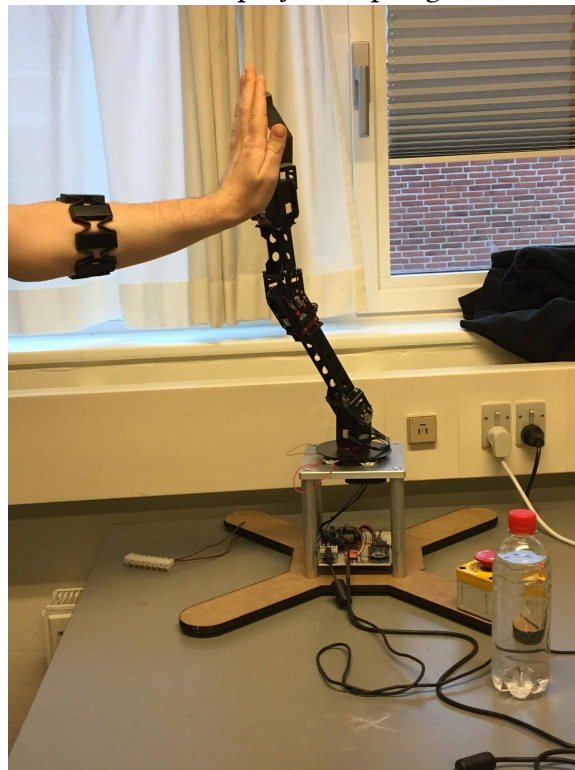


Robotics

Manipulating the surroundings

3. Semester project - Spring 2016



Daniil Avdejev, Kasper F.S. Sørensen, Kasper Tandrup,
Laura Montesdeoca Fenoy, Simon Kaihøj,
and Torben Nielsen

**Sub-theme:**

Design of a control system for a patient with very high level amputation

Project Period:

P3, Autumn 2016

Date: 02/09/2016 - 21/12/2016

Group number:

369

Participants:

Daniil Avdejev

Kasper Sørensen

Kasper Tandrup

Laura Montesdeoca Fenoy

Simon Kaihøj

Torben Moesgaard Nielsen

Supervisors:

Jakob Lund Dideriksen

Karl Damkjær Hansen

Pages: 64

3rd semester

The technical scientific faculty

School of information and communication technologies

Fredrik Bajers Vej 7B

9220 Aalborg East

webinfo@es.aau.dk

Abstract:

This project revolves around creating a Myoelectric prosthetic that can improve motor skills of people with a high level of upper limb amputation. The case being used is Anders, a 33 years old male who had his arm amputated and had an operation done to allow implementation of a prosthesis. The Case goal is to pick up a water bottle and lift it up to Anders mouth in the most fast and effective way. During this project, a prototype capable of lifting the bottle was designed. The Prototype was capable of doing the movement using custom gestures. However the performance time was too high. The prototype is not user-friendly leading to certain failings.

Names and signatures:

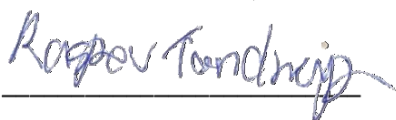
Daniil Avdejev – 20155436



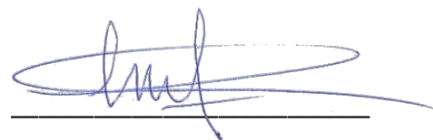
Kasper Faaborg Siegenfeldt Sørensen –
20144205



Kasper Tandrup – 20153397



Laura Montesdeoca Fenoy – 20153630



Torben Moesgaard Nielsen – 20154124



Simon Peter Heide Kaihøj – 20144060



Contents

1	Foreword	5
2	Introduction	6
3	Initial problem statement	7
4	Problem analysis	8
5	Requirements	19
6	Final Problem statement	26
7	Solution	27
8	Testing	52
9	Further development	55
10	Discussion	57
11	Conclusion	58
	Bibliography	59
A	Code - Github	62
B	Code - Obtainment	63

1 Foreword

This report was written by project group 369 on bachelor education of robotics (3rd Semester) at Aalborg University in the period from 2/9-2016 to 21/12-2016. The main theme of this project was “manipulating the surroundings” with the chosen sub-theme being “Design of a control system for a patient with very high level amputation”. This project’s main focus is giving some mobility to an amputee. This was achieved by setting up a case with a fictional person, having problems lifting a bottle. All references being used in this report are placed after claims or facts, and figures were credited to their authors or were remade by the group. The group would like to thank the people who helped, or gave advise during the project.

2 Introduction

A prosthesis as replacement for a lost limb have been around for centuries, but it was just until late 58' that the first myoelectric prosthesis was created [1], and is now at a stage where an amputee can regain a lot of the lost motor functions. This project will look into the task of lifting and drinking from a water bottle with the use of a Myo electric armband and a robotic arm to simulate a prosthesis. The initial problem statement was based on the purpose of an upper-limb prosthesis, to add extra capabilities to the use. The problem analysis will start it's focus on researching what prostheses are and what the challenges it faces currently. To have solvable problem a case was created, this would be the basis for creating the requirements. Having the requirements done, could then lead into the solution. After analyzing the problem a final problem statement was formed trying to fix a certain case specific issue. The Prototype made during this was be using custom gestures, to make the robotic arm move.

3 Initial problem statement

How can we design a robotic solution that using EMG improves the motor skills for people with a high level of upper limb amputation?

4 Problem analysis

The focus of this project is the development of a robotic solution for amputees with a high level of upper-limb amputation. Amputation is a procedure that affects a significant part of the population. According to the Advanced Amputee Solutions: "There are more than a million annual limb amputations globally - one every 30 seconds-" and out of those, 71% are lower limb amputations. [2]

But even when upper limb amputations are not as common, for those who have one, finding a replacement is an arduous task due to the fact that just in a human hand, there are 27 degrees of freedom.[3] These allow performances of complex movements with fine motor skills. It is important that upper-limb prostheses are capable of emulating at least some of that.

For the people who have had amputations, prostheses have been made to allow some level of increase in functionality, these prostheses come in different shapes and sizes and with different purposes and functionality. During this problem analysis the different types of upper-limb prostheses currently available are presented, along with the primary problems they are facing regarding future improvements.

4.1 Upper-limb prostheses

This section aims to explain the demands and challenges of upper-limb prostheses, but in order to understand where these come from, there should be a basic understanding about prosthesis functionality. A prosthesis is a device, either external or implanted, that substitutes for or supplements a missing or defective part of the body.[4]

These prostheses can vary in functionality depending on what goal the prosthesis is supposed to fulfill, or the level of amputation the user has. In this chapter the two main categories of upper-limb prostheses will be looked at:

Passive prostheses also known as a cosmetic prostheses, is a device meant to emulate the visuals of the lost arm/hand, without adding functionality, they tend to be light-weight and not cumbersome for the user. The main purpose of these prostheses is to have the appearance of the lost appendage and they are often inexpensive.[5]

Active prostheses are prostheses meant to give back some level of functionality and therefore need an adequate end terminal, which is the part of the prosthesis meant to be the hand. This is often the most important part of the prosthesis, since this is the part that allows the handling of objects. The end terminal can adopt several shapes depending the task it will serve, being designed as a hand with fingers for more rudimentary assignments, or in other specific shapes for various tasks, such as a special gripping tool or even a hook. [5]

There are two main types of active prostheses that are covered in this chapter. Body powered and Myoelectric prostheses.

Body powered prostheses are those that move based on certain muscle movement of the user. This is accomplished through harnesses attached to the user's body. Which result in simple and cheap solutions, however they can be uncomfortable to wear and sometimes even painful.

Myoelectric prostheses enable movement based on electrical charges during muscle movement. This is accomplished using sensors placed on the user's skin. This type of prostheses allows for the widest range of motion, however these are often the heaviest weighted of the solutions, and due to the electronics the cost is not as affordable.

There are also other types of upper-limb prostheses not mentioned in this project. For example: sports specific upper-limb prostheses which will not be taken into consideration.[5] Having analyzed the different kind of prostheses most available depending on their functionality, the demands they are supposed to meet should be covered to gain a better understanding of the challenges they are facing.

4.2 Demands for upper-limb prostheses

The primary demand for upper-limb prostheses is to mimic the functionality of the hand or arm. This could be in appearance or in dexterity. [6] Lower-limb prostheses tend to have sim-

ple demands, to walk and stand. As opposed to upper-limb prostheses which must have the capability to perform a wide variety of movements. The upper limbs are complicated body parts when designing a robot, for example the wrist and shoulder can rotate, the fingers can move independently of one another, and the gripping of any object must be done with varying degrees of strength.

A prosthetic arm/hand must be capable of emulating at least some of these characteristics to be effective and function properly. Unfortunately, a perfect substitute for a lost upper-limb has not been found. Besides the demand for complex motion previously mentioned, there are other demands that are important for a successfully integrated prosthesis.

One of the major problems with upper-limb prostheses is how comfortable the user feels with them. This issue arises due to the fact that the prosthesis will be attached to the user's body, therefore it is important for the prosthesis to be comfortable. According to a study on costumer design priorities, this is one of the key areas for improvement, because most prostheses, when in contact with the residual limb, cause some discomfort and sometimes pain.[7] Another important aspect is sensory feedback. Sensory feedback is the responds a person get when interacting with the environment. This feature is not present when using a prosthesis[8]. This lack of feedback is a barrier which makes it harder for the user to relate to the prosthetic limb, because they have no information on how much pressure they are applying, and only have visual input to determine whether or not they are touching or gripping something.

Whereas people with lower-limb prostheses identify more with it since during the movement, the leg vibrates and pushes back from the prostheses impacting with the ground while walking. This lets the user feel more immersed with the new leg as a natural part of the body. This feature is not as present in upper-limb prostheses since these vibrations do not transfer to the rest of the body in the same way. This can lead to the user feeling that the prosthesis is just a foreign object attached to the body instead of thinking of it as an extension of the body. As a consequence, many upper-limb amputees feel uncomfortable with the prosthesis, and most describe removing the prosthesis as a feeling of relief.

Lastly, the weight of the prosthesis is a significant factor. Upper-limb prostheses are attached to the body by different means, which will add weight to the user's arm/shoulder whenever the prostheses is on. This can add some extra weight to the back when lifting, plus the weight of the object which is lifted. This is not only a matter of discomfort, but also a health

issue, because having a human 5 kg arm which is by nature incorporated with the body, as to have a 5 kg robotic arm which is attached to the shoulder. Therefore it is necessary to make sure the prosthesis is light to prevent as much as possible of these disadvantages and avoid major discomfort.[6]

The main priority, based on user demands, is functionality. This means that it must be capable of helping the user regain some of their lost capabilities, and at the same time not cause discomfort. It must be able to fulfill this while being as light-weight as possible. Sensory feedback is a feature that can help the user identify more with the prosthesis.

Having itemized the demands for prostheses, a analyze of what are the challenges with upper-limb prostheses that are delimiting them from meeting these demands.

4.3 Challenges for upper-limb prostheses

A study by Elaine A. Biddiss and Tom T. Chau on the rejection rates of upper-limb prostheses over the last 25 years, concluded that the rejection rate is approximately 1 in 5 users. However, the article stated that the surveys, on which the study was predominately based, came from rehabilitation centers; it would not be representative of those who did not expend their time there. Therefore it was possible that the rejection rate could be even higher, the precise number is hard to specify[9] whenever a user rejects a prosthesis, it means that the prosthesis is either not fulfilling the requirements or it is uncomfortable, and therefore not adequate to be used.

The study stated that only 22% of rejection is due to prosthesis performance or discomfort. It also states that if the person has one healthy arm, almost 90 % of everyday actions can be done one-handed. According to the same study, the following are the main challenges upper-limb prostheses are having are. [8]

- Stump prosthesis interface
- Control and recover function
- Sensory feedback

Stump prosthesis interface: As mentioned before, the prosthesis will have to be attached to the body. One method to do it is via sockets, which are an intermediate between the residual

limb and the prosthesis itself. However, these are only an adequate solution for some cases, and with transhumeral amputees (above elbow amputees), where the stump is short, the design of the sockets is usually problematic because in order to properly fit, it will block certain movements of the shoulder joint. They are also known to be uncomfortable for the user to wear. Therefore, for that kind of amputation a different method called osseointegration is used. This process involves a surgery which will place a metal rod into the bone to allow the attachment of the prosthesis. Osseointegration solves the problems stated above when using the socket but at the same time, since it requires surgery, the rehabilitation period is longer. [10]

Control and recover function: The main limitation is based on the information transfer of data from the user to the prosthesis(control) and back(sensory feedback). The success of the prosthesis is the functionality. So if the delay between the user sending a signal and the prosthesis reacting is too long, the prosthesis cannot serve its purpose.

Sensory feedback: Providing sensory feedback for the users has been a challenge for a long time. However the primary benefits of this are not as obvious as first assumed. The use of visual cues, and in the case of Myoelectric prosthetic limbs the ability of the muscle to use the prosthetic limb. The advantages of having sensory feedback implemented in the system is an advantage of embodiment for the user. This is something that makes the prostheses easier to have as part of the user and not as a foreign object. This could be done non-invasively through vibrations or through implanted nerve electrodes.

As presented above, there are several issues with the upper-limb prostheses of current years. Varying from the method used to attach them to the user's body, the degree of discomfort or the range of motion they allow. The prosthetic limb must also be responsive to the user's movements and process the information fast to make sure that the functionality of the prostheses comes as close to a human arm as possible. Lastly, sensory feedback is an important feature to allow the user have full embodiment with the prostheses. With this as a baseline information about prosthetic limbs, the case for this project is formulated.

4.4 The case

Throughout this project, a MYO armband, which read EMG signals, and a Crustcrawler, which is a robotic manipulator used to simulate a prosthetic arm; will constitute the hardware for the

robotic solution to this problem.

4.4.1 The Myo gesture control armband

The Myo armband is an armband capable of measuring information based on different sensors. Those sensors are:

- Surface EMG sensors
- Nine axis IMU

The IMU can measure a three axis accelerometer, three axis magnetometer and three axis gyroscope.[11] The armband comes with certain pre-programmed gestures. The MYO can then transmit the data it receives through blue tooth where installed software can recognize the gestures to allow control of computers, phones or other technical devices. The Myo can also be programmed for more specific purposes, such as programming your own gestures or stream its sensor data. This is done through Myo's own software development kit Myo SDK. Myo SDK allows work with the Myo more internally. It consists of such things as Events and a Hub. The Hub is an entry point in SDK. It provides a way to connect to one or more myo armbands (Myo Connect application serves as a central hub). Events are data provided to the application. Usually events are divided in 3 parts: spatial, gestural and auxiliary events. Spatial events generally appear regularly, for example, whenever Myo armband updates its internal measurement of orientation. Gestural events happen more rarely (irregularly), for example, whenever Myo detects gesture change from the user. Auxiliary events occur infrequently and relate to Myo, for example, being connected or disconnected. Myo SDK is needed for manipulation of Myo armband. Most of the possibilities to access data from Myo armband consist of using Myo SDK. [12] [13]

When using pre-made technology such as the Myo in the project it has certain advantages and disadvantages. An advantage of the Myo armband is that it is easy to place on for the user and calibrate. This product is commercially available so it is made with ease of access in mind. A disadvantage is the lack of flexibility. The armband can only be placed on an arm and not on the chest, limiting the potential areas where the Myo could be placed on the amputee, when talking about transhumeral amputees the stump left to work on can be short so this is a delimiting factor that must be taken into account when producing a solution. Since the

Myo measures EMG and an important feature in the problem statement is EMG, the following section analyzes what EMG is.

4.4.2 EMG

EMG is electrical activity of Motor neurons transmitting electrical signals that cause muscles to contract and EMG signals can be translated into graphs, sounds or numerical values that a specialist can interpret. EMG is usually detected by either electrodes mounted on the skin or intramuscular electrodes. Additionally size of electrodes, distance between sources and detection points (positioning) is relevant. In addition, to remove technical interference, for example, from power-lines, and to compensate spatial low-pass filtering effect of the tissue separating sources and electrodes (aka “blurring”), surface signals are detected as a linear combination of the recorded signals at more than one electrode. Also such operations can be recorded monopolarly.[14] There are two types of EMG - surface and intramuscular. The general understanding is that surface EMG also known as sEMG represents harmless way of gathering EMG data and as for intramuscular EMG, process involves insertion of monopolar (or concentric) needle electrode into muscle tissue. This project was using surface electromyography.

The raw EMG signal comes in unprocessed form that can be seen in figure 4.1, the raw EMG cannot be used for the purpose of Myoelectric Prosthetics, therefore some analysis and processing of the signal is necessary. Furthermore because surface EMG is used to read the signal instead of the invasive method of inserting needles in the muscle, also creates some problems that take additional processing of the signal.

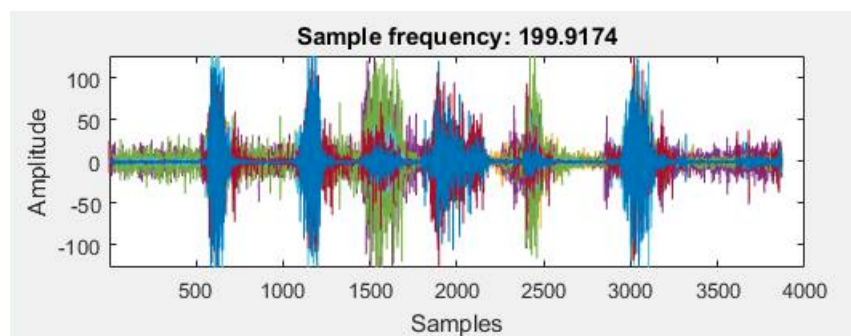


Figure 4.1: Graph of raw EMG data

One of the problems that can be faced is from using surface EMG; the problem lies with

what is called cross talk, which are signals detected by the surface electrodes from neighboring muscles. This problem and the problem of ECG which is signals picked up from the heart can be solved using sophisticated algorithms to reduce cross talk. Other than dealing with cross talk, the signal will have to be rectified which is the process of converting all negative amplitudes into positive amplitudes, this has the effect that amplitude methods can be applied to the signal. Another process to use on the EMG signal is the processing of smoothing. Smoothing is applied because of random EMG bursts that cannot be reproduced a second time around to a precise shape. The non-reproducible shapes are minimized by applying smoothing algorithms that outline the mean trend of the signal. Two algorithms that can be used for this purpose is moving average (MA) and Root Mean Square (RMS). RMS is the preferred recommendation for smoothing. An example of EMG signal processed through Root Mean Square can be seen in figure 4.2. Another form for processing is digital filtering, digital filtering can further fix the problem that smoothing fixes if it is necessary. Another signal process is Amplitude Normalization, amplitude normalization is used because the amplitude can be influenced by detection condition (cross talk, changes in the geometry between muscle belly and electrode site), the detection condition can be very different depending on a number of conditions, but one solution to overcome this problem is the normalization to a reference value, which is also called the maximum voluntary contraction (MVC) value. The whole point of it is to “calibrate the microvolts value to a unique calibration unit with physiological relevance, the percent of maximum innervation capacity”. [15]

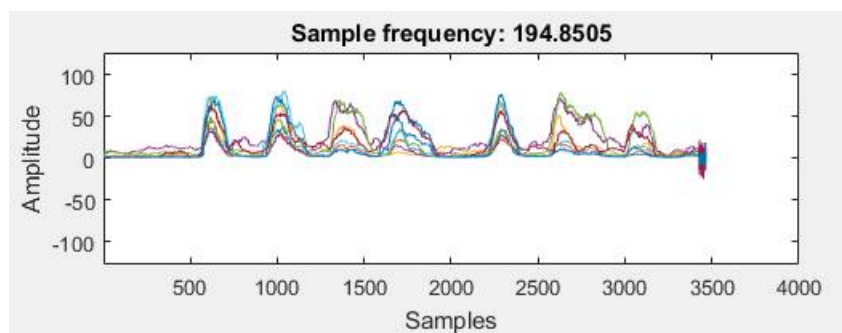


Figure 4.2: Graph of processed EMG

4.4.3 The Crustcrawler

The Crustcrawler is a composite robotic arm made from different servos. These servos are connected with some metal bar and it is equipped with a dual gripper.

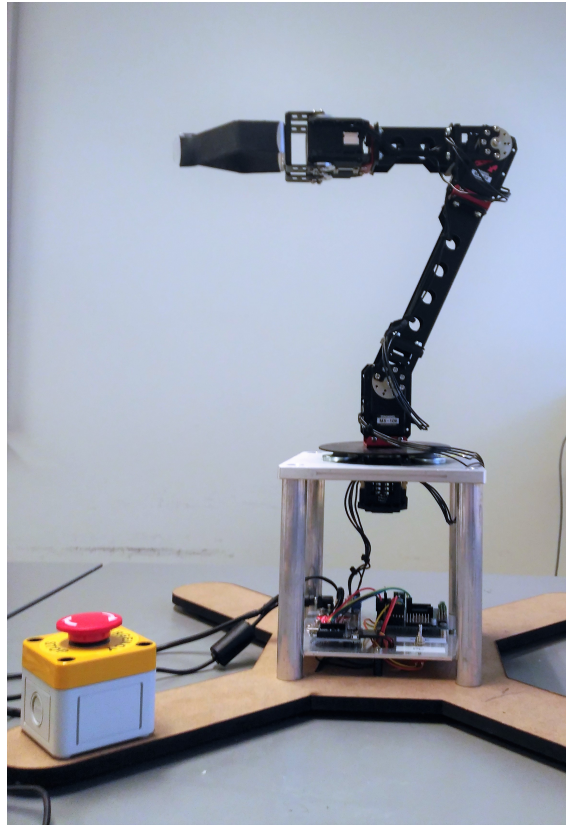


Figure 4.3: The Crustcrawler

These are connected via electric boards that allow for connection through USB to a computer. The hardware will be explained below.

- The UartSBee is the first link between a computer and the actuator, meaning a USB to Serial adapter. UartSBee is compact and able to communicate with TTL/CMOS Serial devices, and program for Arduino and other boards alike. [16]
- The Breakout board for the RS-485 transceiver IC is able to convert the data received from the Uart in the form of a Serial stream. This board is a half-duplex transceiver and can only send data one way at a time.[16]

- The Arbotix-M can control and operate much like an Arduino. This is where the Actuator is able to use the written code to operate, it receives a series of data and uses it to control the servos.[16]
- The six port AX/MX power hub is where the entire system gets power from, with six bioloid port hubs and have a SMPS adapter.[16]
- The servos are the muscles of the entire project. Each servo is able to track the speed with which it is moving, temperature, shaft position, voltage and load capacity. All of the sensor can be handled by management and position control by the servo's built-in micro-controller.[16]

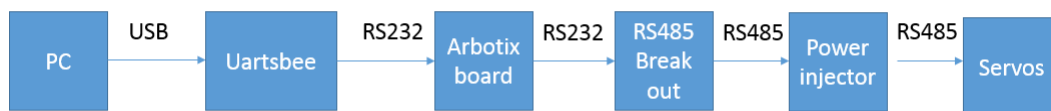


Figure 4.4: Connection with the Arbotix board

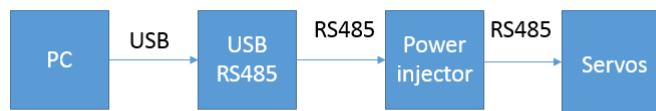


Figure 4.5: Connection without the Arbotix board

4.4.4 The Case - Anders

Anders is a 33 years old male who due to a car accident, had his arm amputated. The amputation was done at the right humerus leaving 9cm of the bone. Anders is 180 cm tall and weighs 100 kg. Anders has had an operation done to allow for a osseointegrated prosthetic to be implemented on him.

Anders has trouble when it comes to eating, the fact that he only has one arm/hand to grab various objects during a meal is something that limits him, meaning it takes more time.

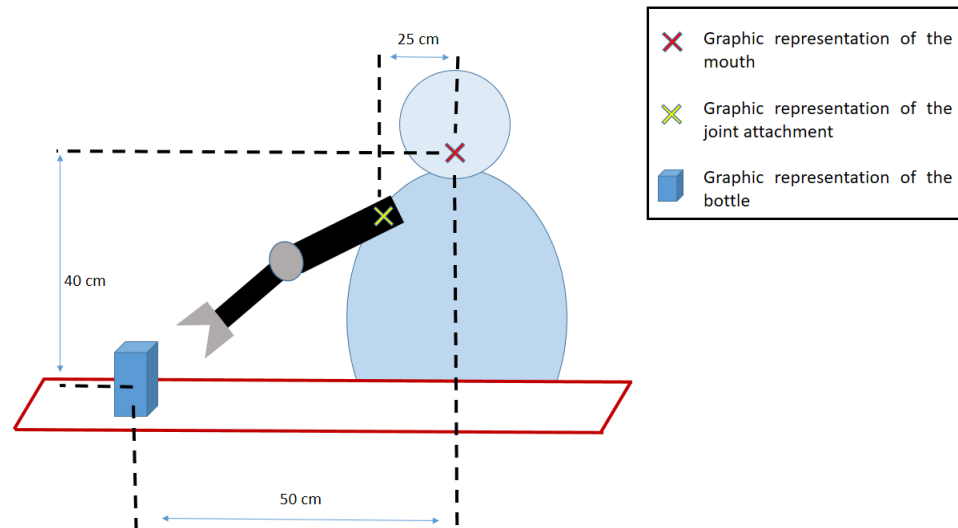


Figure 4.6: case scenario

In this case the goal is to pick up a water bottle and bring it up to Anders mouth. The water bottle is 50 cm from the center of Anders body, there is 40 cm from the table to Ander's mouth and the prosthesis start 25 cm from the center of his body.

The lift must be done in the most effective and fast way possible, the accuracy is also important to avoid spilling.

5 Requirements

5.1 Purpose

The purpose of the product requirement is to illustrate the requirements for the solution made in the P3 project of Robotics 2016. All the requirements have been taken from the problem analysis, analyzing the initial problem - "How can we design a robotic solution, that, using EMG, improves the motor skills for people with a high level upper-limb amputation?"

Index for product requirements:

- Purpose
- General description
- overall requirements
- Performance requirements
- External requirements

5.2 General description

The product will need to allow Anders to grab the water bottle at a 45.54 cm distance from the shoulder and lift it up to the mouth at 40 cm above the table. This must be done in a fluent motion and needs to avoid spilling.

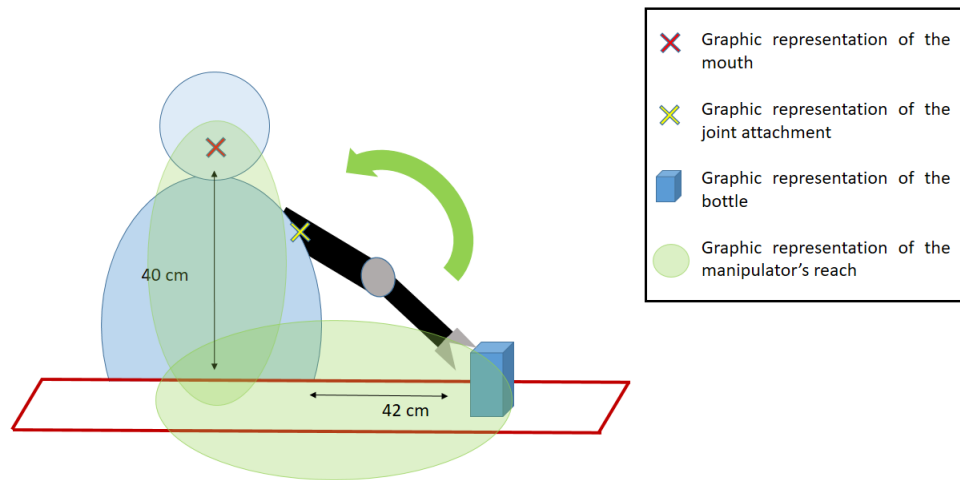


Figure 5.1: This is a representation of the robotic manipulator's reach viewed from the front

The figure 5.1 illustrates the concrete case of Anders and shows the reachable work space of the prosthetic limb. In order to do so, the robotic manipulator will move towards the bottle as shown in the figure and once it has picked it up it will bend on its elbow joint over 90 degrees to relocate it close to the mouth.

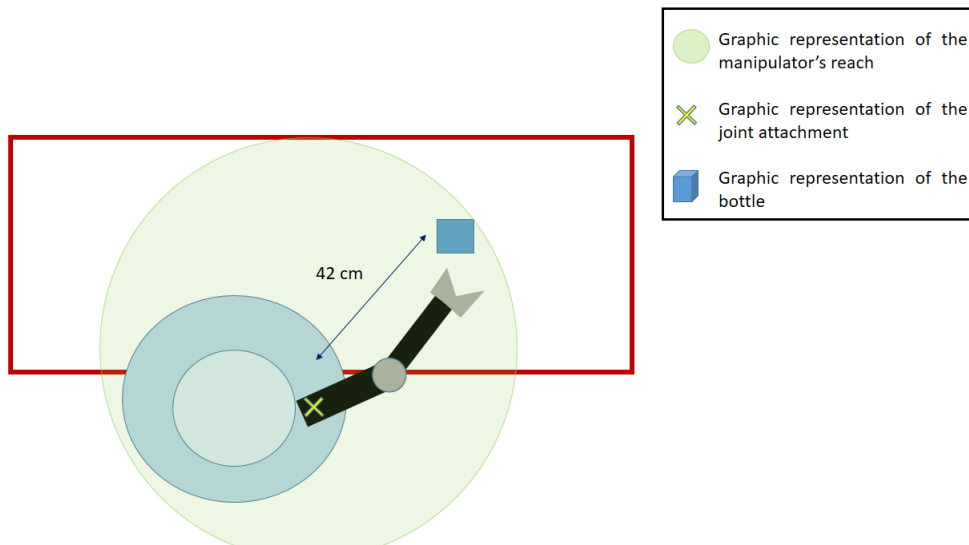


Figure 5.2: This is a representation of the robotic manipulator's reach viewed from an upper position

5.3 Requirement

5.3.1 Overall requirements

The overall requirements are requirements that are not quantifiable, meaning that they can not be tested with numbers as some of the requirements later on can.

- Must help regain some level of capability (see in the case).

The idea behind an EMG prosthesis is to be able to regain some sort of capability, furthermore Anders have trouble when it comes to eating so regaining some capability would be helpful for him.

- Should not limit shoulder movement.

By not limiting shoulder movement the prosthesis would both be more comfortable and regain more functionalities for Anders.

- Have to use EMG to move the robotic arm.

The Myo armband has been given to the group, which is why EMG is a requirement for solving the problem.

- Has to be mounted using osseointegration.

Because of the case, where we have a person with osseointegration, this has to be one of the requirements for the project.

- Must allow for some level of sensory feedback.

When the product allows for some sort of sensory feedback it helps with regaining some functionalities that have been lost.

- Must not be uncomfortable to the costumer.

To reduce the chance of the user rejecting the prosthesis it must not be uncomfortable to wear for the user.

5.3.2 Performance Requirements

The performance requirements are both static numerical and dynamic numerical. They include things as a minimum requirement for lift, reach and speed. The performance requirements are as follows:

- Has to be able to have a reach of 46 cm in both directions.

In order to set this requirement, some experimental test were made. The experiment consisted of measuring the distance at which different people would feel comfortable place a bottle after drinking from it. Since the case this manipulator is based on is designed for a male, most of our samples correspond to four healthy males of varying ages and physical constitution. In addition, one female was also recorded.

Reach samples		
Subject	Samples recorded	Average
<i>Male</i> ₁	35 cm 32 cm 34 cm	33.67 cm
<i>Male</i> ₂	47 cm 49 cm 51 cm	49 cm
<i>Male</i> ₃	66 cm 72 cm 64 cm	67.33 cm
<i>Male</i> ₄	38 cm 39 cm 43 cm	40 cm
<i>Female</i> ₁	37 cm 40 cm 36 cm	37.67 cm

Figure 5.3: Shows the reach for each individual

The average of all the people combined is 45.54 cm. However, for the specific case which the prototype is going to be tested on, due to the length of the manipulator, it was agreed to set the distance of 42 cm in both directions to be the reach in the work space in order to pick up the bottle.

- Minimum carry weight of 600 grams.

With a minimum carry weight of 600 grams it will be able to carry a half liter water bottle.

- Vertical reach of 40 cm.

With this requirement he would be able to reach his mouth with the bottle so he can be able to drink out of it.

- Must be able to perform the specific task in under 30 seconds.

If it takes more than 30 seconds to move a water bottle to your mouth it means it would take too long and using the other hand would then be more appropriate.

5.3.3 External interfaces

- Should not weigh more than 5.2 kilos

Because the person in the case weighs 100 kg the manipulator should not exceed that by more than 10%. This will allow to reduce the rejection rate of the prosthesis because of it being too heavy and therefore uncomfortable.

5.4 Success criteria

Success criteria		
Requirements	Criteria	How to Test
EMG signal received	Receiving EMG signals from the Myo armband	A user wears the Myo armband and a computer receives a signal
EMG signal read/translate	Get a signal that the robot can use	Interpreting the signal into code
EMG cooperation	Use EMG to move the robotic arm	Using the Myo armband to move the robotic arm to desired destinations
Horizontal reach	Have a reach of 25 cm in both directions	Pick up a water bottle from both directions
Vertical reach	Have a reach of 40 cm	Lift up a water bottle 40 cm
Lift	Able to lift 600 grams	Carry a full water bottle to simulate drinking
Performance time	Start and finish drinking from a bottle within 30 seconds	Timing a full cycle time of the task
Sensory feedback	The product gives the user some sensory feedback	The Myo armband gives feedback when performing a task
Weight of manipulator	Cannot weight more than 5.2 kilos	The arm is weighted to ensure it does not exceed the weight
Comfortable	The product is not uncomfortable to the costumer	The product does not hurt or strain the user when used
Mount to a person	Can be mounted using osseointegration	Not going to test this
Regain capabilities	Help to regain some level of capability	Lift a water bottle from a table to mouth to simulate drinking
Shoulder limitations	Not limiting shoulder movement	Not going to test this

Figure 5.4: Success criteria

6 Final Problem statement

How can we produce a Myoelectric Prosthetic that can perform the case specific movement of lifting a bottle to the users mouth?

7 Solution

7.1 Delimitation

There are some of the product requirements that, because of limitations during prototype development, will not be focused on. These requirements will be elaborated on in this section.

- The Hardware used will be the Myo armband and the Crustcrawler robotic arm.

This means that certain requirements will not be met for this prototype, based on the fact that the prototype itself is not going to be a prosthesis. These requirements are:

- Must be attached through osseointegration.
- Should not limit shoulder movement.
- Must allow for some level of sensory feedback.
- Must not be uncomfortable for the costumer

Since the robotic arm will not be mounted on the person, attachment through osseointegration will not be explored for this prototype. The limitation of shoulder movement is due to the socket in certain transhumeral amputees, limiting movement of the shoulder joint. This can not be tested in this project, due to the prototype not being a prosthetic limb. Therefore the requirement about not limiting shoulder movement is delimited. While the Myo armband has the capability of vibrating, enabling some form of sensory feedback. Sensory feedback is more about identifying with the prosthesis, and since this prototype will not be attached, it can not be tested and will be delimited because of that. It can not be tested in this project if it is going to be uncomfortable for the costumer, since the prototype is not a prosthesis.

7.2 Prototype description

The prototype is a robotic arm controlled by 8 EMG sensors placed on the arm of the user. The sensors are in the Myo armband. The robotic arm will be capable of moving with three degrees of freedom allowing it to reach the goal of picking up a bottle and lifting it up to the user's mouth.

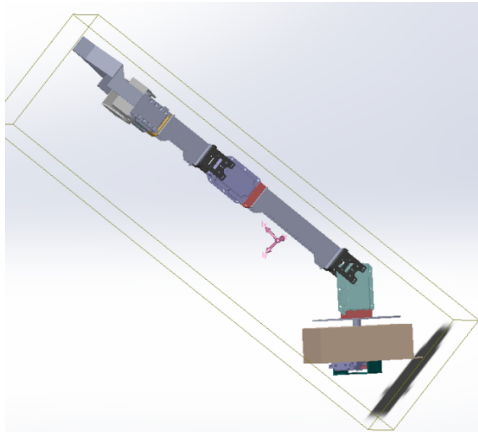


Figure 7.1: simulation of the prosthetic arm

The CrustCrawler is placed on the table and will move between two predetermined points. The first set point is the position of the bottle and the other is the mouth of the user. The software that is used is a gesture recognition software called myoGestureMasterA B. During the project code written to allow control of the robot arm via the Myo armband. The Robotic arm moves based on velocity control changing the velocity of the robots movement. The prosthetic arm will move in a predetermined plane of trajectories, from the position of the bottle to the users mouth, when one of the gestures is used.

7.3 Kinematics

Kinematics is a field which analyzes the geometrical and time based properties of motion but it disregards the forces that cause it. When evaluating the kinematics of a manipulator, the main focus is figuring out the position and orientation of the joints and the end-effector. There are two different ways of calculating the kinematics: forward and inverse.

7.3.1 Forward Kinematics

Forward kinematics is a process which enables the calculation of the position of the end-effector of a manipulator based on specified values for the joint parameters. Where joints are defined as connections which allow motion between two or more links, and links are rigid bodies which have the potential of being attached to one another. Since the manipulator is formed with only three joints and two links, this process can be either calculated with trigonometry or by using the conventional Denavit-Hartenberg (DH) parameters. The Denavit-Hartenberg parameters are a standard method to calculate the homogeneous transformation matrix for a manipulator, it consists of a set of 4 different parameters which combined simplify the process of finding the kinematics. Those parameters consist of two rotational and two displacement parameters: The rotational ones are called θ_i and α_i . Defining θ_i as the rotation around z_{i-1} (Where i refers to the joint which is currently being evaluated and the -1 symbolizes the previous joint) which means that the value of θ_i is the joint variable if the $i-1$ joint is revolute; and α_i is the rotation between z_i and z_{i-1} around x_i . The other two parameters are d_i , which refers to the displacement between the two frames along the z_{i-1} and a_i which measures the displacement along x_i .

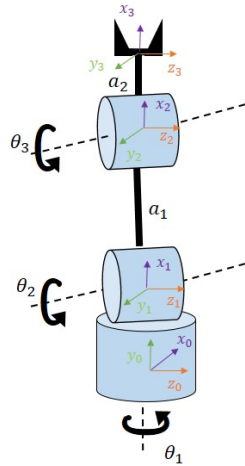


Figure 7.2: Coordinate frame assignment of the CrustCrawler in the starting position

Based on 7.2 the DH-parameters of the manipulator were obtained and shown in the following table:

i	θ_i	α_i	d_i	a_i
1	θ_1	$\frac{-\pi}{2}$	0	0
2	θ_2	0	0	215
3	θ_3	0	0	175

Table 7.1: Denavit-Hartenberg Parameters for the Crust Crawler

The distances a_i and d_i are given in millimeters while the degrees in θ_i and α_i are given in radians.

The following step is to insert the DH-Parameters into the homogeneous transformation matrix as showed with the following equation:

$$T_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_i \\ \sin(\theta_i) \cos(\alpha_{i-1}) & \cos(\theta_i) \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i) \sin(\alpha_{i-1}) & \cos(\theta_i) \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.1)$$

The matrix must be filled with the input from one reference frame to the immediate next one, therefore there is a homogeneous matrix T_1^0 , which is from frame 0 to frame 1; another one T_2^1 which goes from frame 1 to frame 2, and T_3^2 which goes from frame 2 to frame 3.

$$T_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.2)$$

$$T_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 215 \\ 0 & 0 & 1 & 0 \\ -\sin(\theta_2) & -\cos(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.3)$$

$$T_3^2 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 175 \\ \sin(\theta_3) & \cos(\theta_3) & -0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.4)$$

But in order to get the homogeneous transformation matrix for the forward kinematics all the individual matrices stated before must be combined as showed in the following equation:

$$T_3^0 = T_1^0 T_2^1 T_3^2 \quad (7.5)$$

Where T represents the transformation matrix for the different coordinate systems and the result of all of them is the transformation matrix describing the end-effector position related to the base.

$$T_3^0 = \begin{bmatrix} \cos(\theta_1) \cos(\theta_2) \cos(\theta_3) - \cos(\theta_1) \sin(\theta_2) \cos(\theta_3) & -\cos(\theta_1) \cos(\theta_2) \sin(\theta_3) - \cos(\theta_1) \sin(\theta_2) \cos(\theta_3) & -\sin(\theta_1) & 175 \cos(\theta_1) \cos(\theta_2) + 215 \cos(\theta_1) \\ \sin(\theta_1) \cos(\theta_2) \cos(\theta_3) - \sin(\theta_1) \sin(\theta_2) \cos(\theta_3) & -\sin(\theta_1) \cos(\theta_2) \sin(\theta_3) - \sin(\theta_1) \sin(\theta_2) \cos(\theta_3) & \cos(\theta_1) & 175 \sin(\theta_1) \cos(\theta_2) + 215 \sin(\theta_1) \\ -\sin(\theta_2) \cos(\theta_3) - \cos(\theta_2) \sin(\theta_3) & \sin(\theta_2) \sin(\theta_3) - \cos(\theta_2) \cos(\theta_3) & 0 & -175 \sin(\theta_2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.6)$$

The equation 7.6 is the homogeneous transformation matrix showing the position of the end-effector related to the base of the CrustCrawler.

7.3.2 Inverse Kinematics

Inverse kinematics is the process of calculating the joint coordinates needed for an end-effector to reach a Cartesian pose. Since the CrustCrawler is a three joint robot the inverse kinematics can be calculated through trigonometry. First θ_1 is calculated and a graphical representation of θ_1 can be seen in figure 7.3. There it is shown that θ_1 is a rotational joint which determines the orientation of the manipulator.

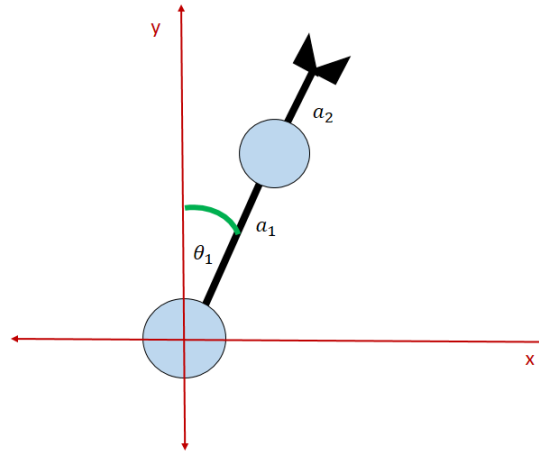


Figure 7.3: Top view of inverse kinematics representation

The value of θ_1 is calculated according to the equation (7.7), where an arctan function of y and x is used. This process will result in two solutions of equal value and opposite sign. Where x and y are the coordinates in Cartesian space of the end-effector.

$$\theta_1 = \arctan 2(y, x) \quad (7.7)$$

In figure 7.4 a graphical representation of θ_2 and θ_3 can be seen. This figure shows θ_2 and θ_3 which correspond to the last two servos before the gripper.

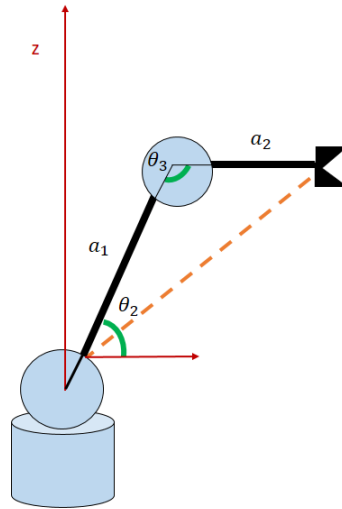


Figure 7.4: Inverse kinematics representation

Using an arccos function θ_3 can be calculated, which is the joint just before the gripper. In order to do that, the length L_1 has to be found which is done by taking the square root of x, y and z squared.

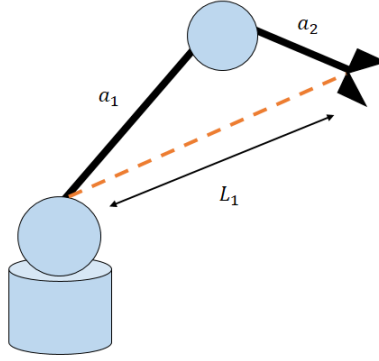


Figure 7.5: Inverse kinematics representation

The method is shown in equation(7.8) which is illustrated in figure 7.5, is simple trigonometry.

$$L_1 = \sqrt{x^2 + y^2 + z^2} \quad (7.8)$$

After L_1 has been found θ_3 is calculated by using the law of cosine. The formula for this can be seen in equation (7.9).

$$\theta_3 = \arccos\left(\frac{a_1^2 + a_2^2 - L_1^2}{2 \cdot a_1 \cdot a_2}\right) \quad (7.9)$$

With both θ_1 and θ_3 found only θ_2 needs to be found to have the inverse kinematics. θ_2 can be seen in figure 7.4 to be the angle by the base. The first thing to do is to find the length L_2 which can be seen in figure 7.6.

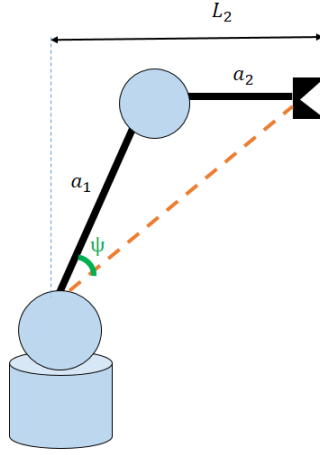


Figure 7.6: Inverse kinematics representation

And the equation to find L_2 can be seen in (7.10).

$$L_2 = \sqrt{x^2 + y^2} \quad (7.10)$$

With L_2 calculated θ_2 can be calculated as an arctan function of two variables plus ψ , which is shown below in equation (7.11).

$$\theta_2 = \arctan 2(z, L_2) + \psi \quad (7.11)$$

ψ can be calculated using the law of cosine which is shown below in equation (7.12)

$$\psi = \arccos\left(\frac{a_1^2 + L_1^2 - a_2^2}{2 \cdot a_1 \cdot L_1}\right) \quad (7.12)$$

With these calculations, it is concluded that the inverse kinematics is defined by:

$$\theta_1 = \arctan 2(y, x) \quad (7.13)$$

$$\theta_2 = \arctan 2(z, L_2) + \psi \quad (7.14)$$

$$\theta_3 = \arccos\left(\frac{a_1^2 + a_2^2 - L_1^2}{2 \cdot a_1 \cdot a_2}\right) \quad (7.15)$$

Which when expanded results as:

$$\theta_1 = \arctan 2(y, x) \quad (7.16)$$

$$\theta_2 = \arctan 2(z, \sqrt{x^2 + y^2}) + \arccos\left(\frac{a_1^2 + (\sqrt{x^2 + y^2 + z^2})^2 - a_2^2}{2 \cdot a_1 \cdot \sqrt{x^2 + y^2 + z^2}}\right) \quad (7.17)$$

$$\theta_3 = \arccos\left(\frac{a_1^2 + a_2^2 - (\sqrt{x^2 + y^2 + z^2})^2}{2 \cdot a_1 \cdot a_2}\right) \quad (7.18)$$

And finally when adding the specific parameters:

$$\theta_1 = \arctan 2(y, x) \quad (7.19)$$

$$\theta_2 = \arctan 2(z, \sqrt{x^2 + y^2}) + \arccos\left(\frac{15600 + x^2 + y^2 + z^2}{430 \cdot \sqrt{x^2 + y^2 + z^2}}\right) \quad (7.20)$$

$$\theta_3 = \arccos\left(\frac{76850 - (x^2 + y^2 + z^2)}{75250}\right) \quad (7.21)$$

7.4 Dynamics

Dynamics is a field which focuses its study around the forces required to cause motion. When analyzing the dynamic behavior of a robotic mechanism we are evaluating "the time rate of change of the robot configuration in relation to the joint torques exerted by the actuators".[17] In order to obtain the equations of motion, there are two main approaches: the Newton-Euler formulation, which describes dynamic systems in terms of force and momentum; and the Lagrangian formulation, where the dynamics are described in terms of work and energy using generalized coordinates. [17] For this specific case, the Lagrangian formulation is a more effective and systematic approach and therefore is the one which is used.

The first step before starting with the dynamic formulation, is to analyze the statics of the manipulator. This operation is used in order to determine the equations for balance of forces and torques for each of the links. Since the dynamics are calculated using the Lagrangian instead of the Newton-Euler formulation, there is no need to formulate the equations for balance, but other aspects analyzed on the statics will be used.

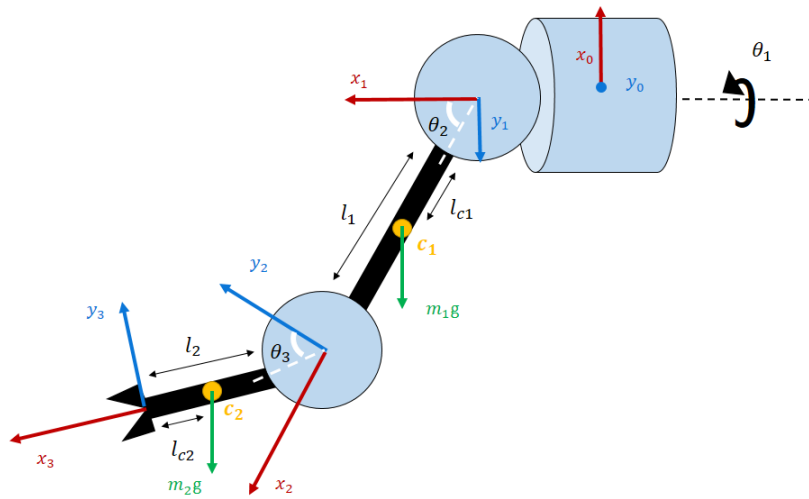


Figure 7.7: Statics representation of the manipulator

Figure 7.7 illustrates the statics of the manipulator where a yellow circle can be seen on each link (c_1 and c_2) representing the centers of mass. Due to the semi-hollow composition of the links, the most accurate way to calculate them is with a CAD model.

<i>Link₁</i>	<i>Link₂</i>
<p>The center of mass and the moments of inertia are output in the coordinate Mass = 197.21 grams</p> <p>Volume = 197213.37 cubic millimeters</p> <p>Surface area = 81597.94 square millimeters</p> <p>Center of mass: (millimeters) X = 20.13 Y = -268.25 Z = 0.71</p> <p>Principal axes of inertia and principal moments of inertia: (grams * square m Taken at the center of mass. lx = (0.00, 1.00, -0.06) Px = 68046.03 ly = (-0.99, 0.00, 0.15) Py = 2127360.13 lz = (0.15, 0.06, 0.99) Pz = 2133093.35</p> <p>Moments of inertia: (grams * square millimeters) Taken at the center of mass and aligned with the output coordinate system. Lxx = 2127453.67 Lxy = -9173.91 Lxz = -359.10 Lyx = -9173.91 Lyy = 74417.67 Lyz = -114161.14 Lzx = -359.10 Lzy = -114161.14 Lzz = 2126628.18</p> <p>Moments of inertia: (grams * square millimeters) Taken at the output coordinate system. lxx = 16319013.96 lxy = -1073943.09 lxz = 2465.80 lyx = -1073943.09 lyy = 154405.98 lyz = -151812.11 lzx = 2465.80 lzy = -151812.11 lzz = 16397977.00</p>	<p>The center of mass and the moments of inertia are output in the coordinate Mass = 122.31 grams</p> <p>Volume = 122312.48 cubic millimeters</p> <p>Surface area = 74910.16 square millimeters</p> <p>Center of mass: (millimeters) X = 18.70 Y = -10.04 Z = -0.73</p> <p>Principal axes of inertia and principal moments of inertia: (grams * square m Taken at the center of mass. lx = (-0.01, 1.00, -0.05) Px = 70654.11 ly = (-1.00, -0.01, 0.00) Py = 152704.93 lz = (0.00, 0.05, 1.00) Pz = 191185.10</p> <p>Moments of inertia: (grams * square millimeters) Taken at the center of mass and aligned with the output coordinate system. Lxx = 152699.97 Lxy = -651.65 Lxz = -47.33 Lyx = -651.65 Lyy = 70965.10 Lyz = -6063.83 Lzx = -47.33 Lzy = -6063.83 Lzz = 190879.08</p> <p>Moments of inertia: (grams * square millimeters) Taken at the output coordinate system. lxx = 165085.99 lxy = -23609.39 lxz = -1708.65 lyx = -23609.39 lyy = 113805.10 lyz = -5172.19 lzx = -1708.65 lzy = -5172.19 lzz = 245976.06</p>

Figure 7.8: Parameters obtained with the CAD model

Once the variables have been calculated, we proceed with the Lagrangian formulation. This formulation derives the equations of motion for the robot from a scalar called the Lagrangian 7.22, which is the difference between the kinetic and potential energy of a mechanical system.

$$\mathcal{L}(\theta, \dot{\theta}) = k(\theta, \dot{\theta}) - u(\theta) \quad (7.22)$$

And thus, the equations of motion for the manipulator are given by

$$\tau = \frac{d}{dt} \left(\frac{\delta \mathcal{L}}{\delta \dot{\theta}} \right) - \frac{\delta \mathcal{L}}{\delta \theta} \quad (7.23)$$

where τ is the nx1 vector of actuator torques. This equation is more useful in the form of

$$\tau = \frac{d}{dt} \left(\frac{\delta k}{\delta \dot{\theta}} \right) - \frac{\delta k}{\delta \theta} + \frac{\delta u}{\delta \theta} \quad (7.24)$$

In order to calculate τ it is required to find both, the kinetic and potential energies. The expression for the kinetic energy of the i th link of the manipulator can be expressed as

$$k_i = \frac{1}{2} (m_{ci}^T) v_{ci} + \frac{1}{2} ({}^i w_i^T) ({}^i I_i) ({}^i w_i) \quad (7.25)$$

where the first term is the kinetic energy due to linear velocity of the link's center of mass and the second term is the kinetic energy due to angular velocity of the link. The v_{ci} and ${}^i w_i^T$ in 7.25 are functions of θ and $\dot{\theta}$, so we see that the kinetic energy of a manipulator can be described by a scalar formula as a function of joint position and velocity, $k(\theta, \dot{\theta})$. Which can be rewritten as

$$k(\theta, \dot{\theta}) = \frac{1}{2} \dot{\theta}^T M(\theta) \dot{\theta} \quad (7.26)$$

where $M(\theta)$ is the n x n manipulator mass matrix which was calculated along with the center of mass with the cAD model as shown in the figure 7.8. The potential energy can be defined as

$$u_i = -m_i ({}^0 g^T) ({}^0 P_{Ci} + u_{ref_i}) \quad (7.27)$$

where ${}^0 g$ is the gravity 3 x 1 vector, ${}^0 P_{Ci}$ is the vector locating the center of mass of the i th link, and u_{ref_i} is a constant chosen so that the minimum value of u_i is zero. Because the ${}^0 P_{Ci}$ in 7.27 are functions of θ , the potential energy of a manipulator can be described by a scalar formula as a function of joint position, $u(\theta)$.

Using the formula 7.25, we substitute for $link_1$ and $link_2$.

$$k_1 = \frac{1}{2} m_1 l_{c1}^2 \dot{\theta}_2^2 + \frac{1}{2} I_{zz1} \dot{\theta}_2^2 \quad (7.28)$$

$$k_2 = \frac{1}{2} m_2 l_{c2}^2 (\dot{\theta}_2^2 + \dot{\theta}_3^2) + \frac{1}{2} I_{zz2} (\dot{\theta}_2^2 + \dot{\theta}_3^2) \quad (7.29)$$

Where m_i represents the mass of the link, l_{ci} is the distance of the center of mass from the joint-axis and I_{zz2} is one of the inertia tensors. And therefore, when combining both kinetic energies the following equation is obtained.

$$k(\theta, \dot{\theta}) = \frac{1}{2}(m_1 l_{c1}^2 \dot{\theta}_2^2 + m_2 l_{c2}^2 (\dot{\theta}_2^2 + \dot{\theta}_3^2) + I_{zz1} \dot{\theta}_2^2 + 2I_{zz2} (\dot{\theta}_2^2 + \dot{\theta}_3^2)) \quad (7.30)$$

The following step is to calculate the potential energy of each link by substituting in formula 7.27

$$u_1 = m_1 l_{c1} g \sin(\theta_2) + m_1 l_{c1} g \quad (7.31)$$

$$u_2 = m_2 l_{c2} g \sin(\theta_2 + \theta_3) + m_2 l_{c2} g \quad (7.32)$$

Which when combined results in

$$u(\theta) = (m_1 l_{c1} \sin(\theta_2) + m_2 l_{c2} \sin(\theta_2 + \theta_3)) m_1 l_{c1} + m_2 l_{c2} g \quad (7.33)$$

Lastly, the partial derivatives required to complete equation 7.24 are computed

$$\frac{\delta k}{\delta \dot{\theta}} = \begin{bmatrix} m_1 l_{c1}^2 \dot{\theta}_2 + I_{zz1} \dot{\theta}_2 \\ m_2 l_{c2}^2 (\dot{\theta}_2 + \dot{\theta}_3) + I_{zz2} (\dot{\theta}_2 + \dot{\theta}_3) \end{bmatrix} \quad (7.34)$$

$$\frac{\delta k}{\delta \theta} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (7.35)$$

$$\frac{\delta u}{\delta \theta} = \begin{bmatrix} m_1 l_{c1} g \cos(\theta_2) \\ m_2 l_{c2} g \cos(\theta_2 + \theta_3) \end{bmatrix} \quad (7.36)$$

Which results on τ , the equations of motion of the manipulator.

$$\tau = \begin{bmatrix} m_1 l_{c1}^2 \dot{\theta}_2 + I_{zz1} \dot{\theta}_2 \\ m_2 l_{c2}^2 (\dot{\theta}_2 + \dot{\theta}_3) + I_{zz2} (\dot{\theta}_2 + \dot{\theta}_3) \end{bmatrix} + \begin{bmatrix} m_1 l_{c1} g \cos(\theta_2) \\ m_2 l_{c2} g \cos((\theta_2 + \theta_3)) \end{bmatrix} \quad (7.37)$$

Where m_1 and m_2 are measured in *grams*, I_{zz1} and I_{zz2} are measured in *grams*milimeters²*, l_{c1} and l_{c2} are measure in *milimeters* and g is measured in $\frac{\text{milimeters}}{\text{seconds}^2}$

$$\tau = \begin{bmatrix} (368.5 \begin{bmatrix} 20.13 \\ -268.25 \\ 0.71 \end{bmatrix})^2 + 2126628.18\dot{\theta}_2 + (368.5 \begin{bmatrix} 20.13 \\ -268.25 \\ 0.71 \end{bmatrix} 9810) \cos(\theta_2) \\ (226.8 \begin{bmatrix} 18.70 \\ -10.04 \\ -0.73 \end{bmatrix})^2 + 190879.08(\dot{\theta}_2 + \dot{\theta}_3) + (226.8 \begin{bmatrix} 18.70 \\ -10.04 \\ -0.73 \end{bmatrix} 9810) \cos(\theta_2 + \theta_3) \end{bmatrix} \quad (7.38)$$

The above equation showcases the τ for the manipulator using the Lagrangian with the specific parameters for the case.

7.5 Programming

7.5.1 PID

PID controller also known as proportional-integral-derivative controller is a control system that computes an actuator output by calculating the proportional, integral and derivative responses and adding them up. A PID controller always has a feedback mechanism that constantly calculate an error value called $e(t)$ which is the difference between a set point and a process variable.

The proportional term in a PID controller can be described as $K_p e(t)$ which is the proportional gain multiplied with the current error signal where the result will be the output signal while the integral term can be described as $K_i \int_0^t e(t) dt$ where K_i is the integral gain, t is the instantaneous time and $e(t)$ is again the error signal. When the integral term is added to the proportional term it will accelerate the process towards setpoint and it also has the benefit of eliminating the residual steady.state error that can occur when only proportional controller is used. The derivative term is the change of the error signal multiplied with the derivative gain and can be described as $K_d \frac{d}{dt} e(t)$ where K_d is the derivative gain and as with all the others $e(t)$ is the error signal. The derivative term will have the effect of slowing the rate of change for the output, and this is an effect that will be most noticeable closer to the setpoint. All the

terms can be set up in a combined algorithm for a PID controller which can be seen in equation (7.39).

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{d}{dt} e(t) \quad (7.39)$$

PID controller is used in the servos for the Crustcrawler and a diagram of the PID controller can be seen in figure 7.9. From the diagram it can be seen that a desired position is put through the P gain, I gain and the D gain where after they are added together then put through the motor model followed by which then gives the output signal. The line going back is the feedback of the output going back to calculate the error value.

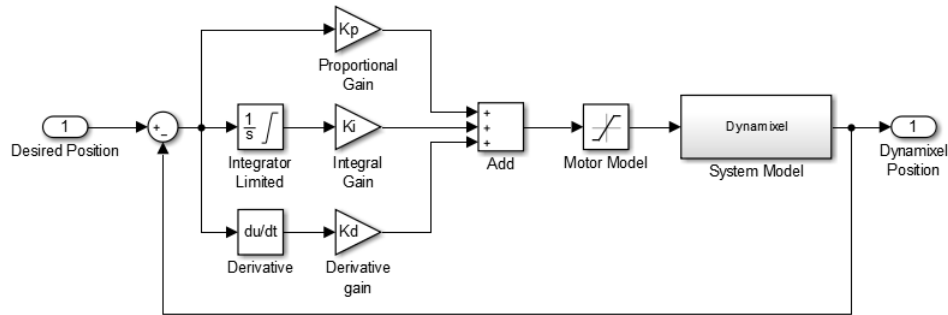


Figure 7.9: Diagram of the PID controller in the Dynamixel MX servos[18]

The following is specific for all the dynamixel mx servos used with the CrustCrawler. Gain values from the servos are between 0254 because there is 1 byte for storage of the values. the proportional gain is divided by eight because it has an initial value of 32 which then brings it down to four. The integral gain is multiplied by $\frac{1000}{2048}$, the reason they divide by 2048 is because that is half a rotation. The derivative gain is multiplied by $\frac{4}{1000}$ [18].

Custom gesture software

During this project, it was decided to use custom gesture software, which allows creation of gestures similarly to what is originally available in the Myo(wave left, wave right etc.). This was done to accomplish intuitive and customizable manipulation of the arm since the projects theme is correlating with humans.

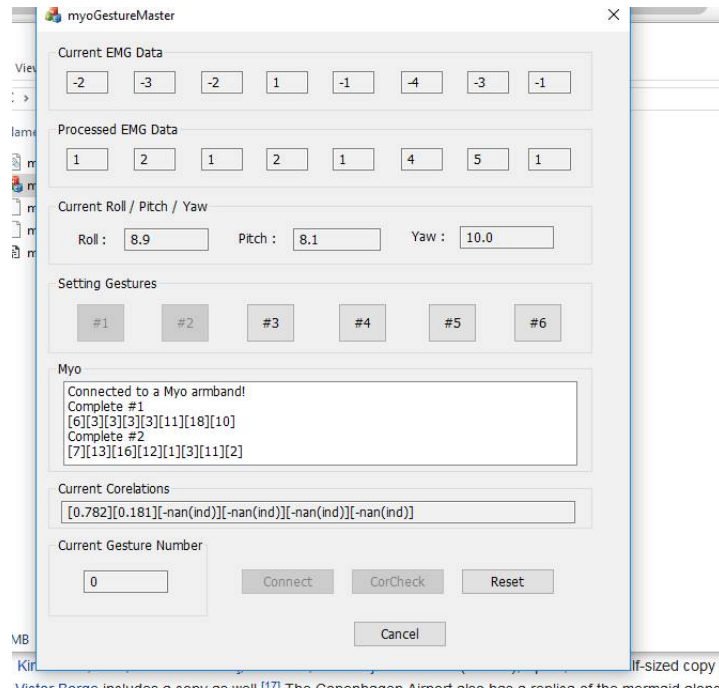


Figure 7.10: Gesture Program

As of seen in the figure 7.10, program acquires raw EMG data from the 8 EMG sensors implemented in the Myo Armband (seen in first row - 'Current EMG data') and processes it in the next row. In addition it gets Roll/Pitch/Yaw data using internal sensors of the Myo. In the next rows of the program we can see actual interface for creating gestures. Currently you can see that there are 2 created gestures and four more available (blackened out rectangles are gestures that are already in the memory; amount of setup gestures can always be changed). Next part of the interface, provides information that was output during gesture setup. Additionally we get correlations and gesture number output to track down which gesture is being used currently during this session. The method of making the program work includes using covariance measure, by comparing absolute correlations (biggest correlation and smallest correlation compared to each other) given by the program.

7.5.2 Covariance and Correlation

Covariance and correlation are calculated in the code to calculate correlation. Correlation is used for recognizing gestures in the way that it looks at how much a performed gesture cor-

respond to a recorded gesture. Covariance and correlation both belong under probability and statistics math.

Covariance measure the joint variability of two or more random variables that are called X and Y . The equation for covariance is defined as equation (7.40), where $\mathbf{E}(X)$ and $\mathbf{E}(Y)$ is the expected value of X and Y , and X and Y are the sets of random variables. As mentioned covariance measure the the joint variability, which means that covariance is equal to zero then the variables are uncorrelated and if the covariance is any other number than zero then they correlate in some way.[19]

$$cov(X, Y) = \mathbf{E}([X - \mathbf{E}(X)][Y - \mathbf{E}(Y)]) \quad (7.40)$$

in covariance when $X = Y$ then the covariance will be equal the standard deviation of one variable squared which can be seen in equation (7.41).

$$cov(X, X) = var(X) = \sigma_X^2 \quad (7.41)$$

The equation that defines correlation can be seen in equation (7.42) where the covariance is divided by the standard deviation of X and Y where the standard deviation is denoted by σ . By calculating equation (7.42) the result will be the correlation coefficient.

$$cor(X, Y) = \frac{cov(X, Y)}{\sigma(X)\sigma(Y)} \quad (7.42)$$

In the code covariance and correlation is calculated in the class `SignalProcessing` in the function `getCorr`. The calculation of covariance in the code can be seen in figure 7.11 where the result of the covariance named `child`. It can be seen in the code that the first thing done is calculating the sum for the EMG signal for then afterwards calculating the mean for it. In the end the mean is subtracted from each value and put into two arrays called `standard` and `comp`. In the end the vectors are multiplied and the result is the covariance.

The calculation of the correlation coefficient in the code can be seen in figure 7.12 where `child` is as mentioned before the covariance and `parent` is the standard deviation. For the correlation coefficient to be calculated first the standard deviation need to be found. As seen in equation (7.41) that if the variables are the same in the covariance then it is equal the standard deviation squared, which means that to calculate the standard deviation in the code the two

arrays from covariance are squared. Afterwards they are multiplied together for then taking the square root which then gives the standard deviation. In the end equation (7.42) is used in the code to calculate the correlation coefficient.

```
// Get the sum
for (int i = 0; i < 8; i++)
{
    sum_s = sum_s + standard_tmp[i];
    sum_c = sum_c + comp_tmp[i];
}

// find the average
mean_s = (double)sum_s / 8;
mean_c = (double)sum_c / 8;

// Subtract the average for each value.
for (int i = 0; i < 8; i++)
{
    standard[i] = (double)standard_tmp[i] - mean_s;
    comp[i] = (double)comp_tmp[i] - mean_c;
}

// The numerator is multiplied by the value corresponding to each vector. (Covariance)
for (int i = 0; i < 8; i++)
{
    child = child + (standard[i] * comp[i]);
}
```

Figure 7.11: Computing covariance

```
// Find the sum of the squares of each vector in the denominator.
for (int i = 0; i < 8; i++)
{
    parent_tmp_1 = parent_tmp_1 + ((comp[i])*(comp[i]));
}

for (int i = 0; i < 8; i++)
{
    parent_tmp_2 = parent_tmp_2 + ((standard[i])*(standard[i]));
}

// Multiply the sum of squares of each vector.
parent_tmp = parent_tmp_1 * parent_tmp_2;

// Put the root on the value.
parent = sqrt(parent_tmp);

// Divide the numerator and denominator to get the correlation coefficient. A value between -1 and 1.
// close to 0 means no relation, close to 1 same shape, close to -1 shape.
corr = ((double)child / (double)parent);
test_double[input_i] = corr;
return corr;
```

Figure 7.12: Computing correlation

7.5.3 Code description

As mentioned before in section 7.5.1 a custom gesture program was used to create and recognize gestures to control the robot. This section will go more into depth with the code of the program.

The section will also go into the code written for the robot.

The first thing in the myoGestureMasterDlg.cpp is a class called DataCollector. In this class, the EMG data from the Myo armband is collected, whereafter it is processed. The way that the EMG data is collected from the Myo, is with the function onEmgData. Every time an EMG value is available from the Myo, it is collected and put into emgSamples, which can be seen in figure 7.13. onEmgData is run when hub.run is activated in the code telling the Myo that the program is ready to receive data.

```
void onEmgData(myo::Myo* myo, uint64_t timestamp, const int8_t* emg)
{
    for (int i = 0; i < 8; i++) {
        emgSamples[i] = emg[i];
        emgTmp[i] = (int)emgSamples[i];
        originTmp[i] = (int)emgSamples[i];
    }
}
```

Figure 7.13: emg extraction from myo

The data is processed in two different functions. One, setUserStandard, is used to record the gestures for later recognition, the other upDataEmg processes the data for it to be compared against the gestures recorded in setUserStandard. setUserStandard is called whenever one of the "Setting Gestures" buttons is called. upDataEmg is called from WorkerThread2, where it is used to get the correlation coefficient.

In the class SignalProcessing, the correlation coefficient is calculated in the function getCorr, using the process explained in section 7.5.2. getCorr is, like upDataEmg, called from WorkerThread2.

There are five worker threads in the program, such as WorkerThread2. They handle background tasks that the user should not wait on for the program to continue. This could be such things as recalculations, background printing and so on[20]. The first WorkerThread makes sure that the raw emg, processed emg and roll pitch yaw is updated in the user interface. WorkerThread2 shows the current correlation and determines which gesture is being produced at the moment. WorkerThread3 writes the number of the current gesture in the user interface. Thread number 1, 2 and 5 are activated when the button connect is pressed while 3 and 4 are activated when the button check corr is pressed.

The Workerthread4 function 7.14 is the link between the EMG signals and the robot. This functionality is written into an existing function, that handles updating the display of cor-

relation values. This location was chosen, because the delay when waiting for a response from the servos would not delay the execution of any important code. The function is called when the “CorCheck” button is pressed. It keeps running as a separate thread until either the “everyThreadFlag” is set to false, by pressing the “Cancel” button, or the program is terminated. The function starts out by declaring variables needed for the updating of the correlation values.

The function is passed a pointer to the interface object, and it casts this to a pointer of the correct type. It then creates a robot object connected on the COM4 port at a baudrate of 1 Mbaud. Because the software is compiled with unicode or multibyte character sets enabled, an L is needed before the string to signify that COM4 is a string of long characters. If the program was to be run on another computer, the COM port number would need to be changed to reflect the name assigned to the connection to the robot. After that, currentvalue, lastservo, and gripper are declared and initialized, to values roughly corresponding to the robot standing straight up. This was done such that the robot can be put in this position before the program starts to control the robot, to avoid sudden large movements when the program takes control. TorqueEnable() sends a command to the servos to enable torque, such that the servos cannot be moved freely. The sensitivity values are used to set the sensitivity to the different servos. Changing this will change the sensitivity only with regards to the selection of what value to integrate on, and will not change the speed. With all the values set to 1, none of the sensitivities are being changed.

```

UINT WorkerThread4(LPVOID lParam)
{
    CmyoGestureMasterDlg* dlg = (CmyoGestureMasterDlg*)lParam;
    CString tmp;
    robot crusty(L"COM4", 1000000);
    double currentvalue = 0.8f, lastservo = 0, gripper = 0;
    crusty.torqueEnable(_ALL_SERVOS);
    double sensitivity[6] = { 1, 1, 1, 1, 1, 1 };

    while (EveryThreadFlag)
    {
        tmp.Format(_T("[%0.31f][%0.31f][%0.31f][%0.31f][%0.31f][%0.31f]"), corr[0], corr[1], corr[2], corr[3],
            corr[4], corr[5]);
        dlg->c_cors.SetWindowTextW(tmp);

        int largest = -1; //find out what gesture is most active
        double temp = -1;
        for (int i = 0; i < 6; i++)
            if (corr[i] * sensitivity[i] > temp)
            {
                temp = corr[i] * sensitivity[i];
                largest = i;
            }

        if (isnormal(corr[0]) && isnormal(corr[1])) //integrate the result
        {
            if (largest == 0 || largest == 1)
                currentvalue = currentvalue + (corr[0] - corr[1]) / 500;

            if (largest == 2 || largest == 3)
                lastservo = lastservo + (corr[2] - corr[3]) / 500;

            if (largest == 4 || largest == 5)
                gripper = gripper + (corr[4] - corr[5]) / 200;
        }

        crusty.setpos(1, scaleservo(currentvalue, 1066, 2042)); //values determined experimentally
        crusty.setpos(2, limitservo(scaleservo(currentvalue, 3175, 1647), 800)); //
        crusty.setpos(3, limitservo(scaleservo(currentvalue, 1849, 2469) + scaleservo(lastservo, 0, 2000), 800)
            ); //
        crusty.setpos(4, scaleservo(gripper, 2048, 4096));
        crusty.setpos(5, scaleservo(gripper, 2048, 0));

        Sleep(5);
    }
}

```

Figure 7.14: WorkerThread4

The while loop runs until the “cancel” button or the “reset” button is pressed. The first two lines in the loop handle the printing of the correlation values to the interface. The next section finds the gesture with the highest correlation value. The isnormal() function takes a floating point number and returns true if the number is not infinitely large or small, and the variable contains an actual number (as opposed to “NaN” or “not a number.”) The correlation value will occasionally be NaN if the user relaxes their arm, but all the correlation values are NaN at the same time.

The setpos() function sets the goal position in the corresponding servo to the given value. The scaleservo()7.15 function interpolates between two integers based on a floating point value.

It does not limit the result to the given values. `Scaleservo(0, x, y)` would return `x`, while `scaleservo(1, x, y)` would return `y`. `limitservo(7.15)` limits a value to the given value as minimum, and 4096 - that value as maximum. This function is used to limit some of the servos to positions actually reachable by the given servos.

```
int scaleservo(double current, int min, int max)
{
    int fullscale = max - min;
    float result = current * (double)fullscale + min;
    return (int)result;
}

int limitservo(int value, int min)
{
    if (value < min)
        return min;
    if (value > 4096 - min)
        return 4096 - min;
    return value;
}
```

Figure 7.15: Scaleservo and limitservo functions

The function `servo1()`, `servo2()`, and `servo3()` calculate the inverse kinematics, as explained in 7.3.2. To see this in action, see the “Linear” branch on github.

Robot control

A C++ library was made to control Dynamixel AX/MX servos, using a USB-RS485-WE to connect the servos to a computer. It uses a slightly modified version of a serial library[21] (modified to be able to set the baudrate). The robot library also contains a set of defines from the “DYNAMIXEL Library”[22], made for Arduino. The library implements the ability to send and receive arbitrary data to and from the memory of the servos, and has some additional functions for sending data to specific addresses, such as “torque enable” and “goal position.”

```

void robot::setpos(unsigned int ID, int pos)
{
    char arr[9];
    arr[0]= (char)255;    //start message with 0xFFFF
    arr[1]= (char)255;
    arr[2]= (char)ID;     //servo ID
    arr[3]= (char)5;      //length
    arr[4]= (char)AX_WRITE_DATA; //command (write)
    arr[5]= (char)AX_GOAL_POSITION_L; //Address (goal pos)
    arr[6]= pos % 256; //lower byte of position
    arr[7]= pos / 256; //upper byte
    int sum=0;
    for (int i = 2; i < 8; i++)
        sum = sum + arr[i];
    arr[8]= ~(sum % 256);
    serialobj->WriteData(arr, 9);
    if (ID != 254)
        status_msg_reciever();
}

```

Figure 7.16: Function from robot.cpp

This is the function that tells the servo to go to a certain position. First, an array for storing the message to be sent to the servo is created. Since the exact length of the message is known, that can be used to create an array of the exact size needed. All messages sent to the servos need to start with 0xFFFF [23] so the servos start listening. Next is the ID of the servo, that is provided by the calling function. The length is the number of bytes left of the message, and for a write data that writes two bytes, it will always be 5. If the position is what needs to be set, the data needs to be written to address 30 and 31. These are defined as AX_GOAL_POSITION_L for the lower byte and AX_GOAL_POSITION_H for the upper byte. Since they are consecutive, it is possible to write to both of them by writing two bytes starting at the lower address. Then the lower and upper byte of the new goal position is copied to the message. Testing if the input position is a valid value is not done, since this might change in a future version of the servos, similar to how it changed from 1024 position in AX servos to 4096 positions in the MX servos. The datasheet for the AX12 servo defines the checksum as:

“The computation method for the ‘Check Sum’ is as the following. Check Sum = (ID + Length + Instruction + Parameter1 + ... Parameter N) If the calculated value is larger than 255, the lower byte is defined as the checksum value. ~ represents the NOT logic operation.” [23]

serialobj is a pointer to the serial object that is connected to the robot. The WriteData(char *buffer, int nbChar) function sends the first nbChar bytes from buffer to the serial port, with

the baudrate from the constructor (default is 1 000 000), byte size of 8 bits, one stop bit, and no parity. `Status_msg_reciever()` handles the response from the servos, but only needs to be called if the id is not the broadcast ID (254).

8 Testing

Testing the prototype will be done by having two persons doing the lifting of the water bottle 5 times and calculating the average. The persons doing the test will be chosen based on experience with the prototype. One of the individuals will have experience working with the prototype the other will have little experience. This will show the user friendliness of the prototype. The experiment is done with a full bottle and there will be measured how long it will take to pick up the bottle, and lift it up to the mouth. the distance will be measured based on the requirements.

8.1 Results

The results of the test showed:

Results person 1: experienced		
Test	Time	Success/not success
1	46 Seconds	Success
2	25.5 Seconds	Success
3	37 Seconds	Not success
4	33 Seconds	Not Success
5	25.8 Seconds	Success
Average: 33.43 Seconds		

Figure 8.1: Result for person one

Results person 2: inexperienced		
Test	Time	Success/not success
1	31 Seconds	Not success
2	29 Seconds	Not success
3	47 Seconds	Success
4	33 Seconds	Not Success
5	28 Seconds	Not success
Average: 47 Seconds		

Figure 8.2: Result for person two

The average was calculated based on all the successful attempts. the reasons for the non-successful attempts were that the gripper would overload causing the gripper to not work, or that the bottle would slip during the lift. The experience shows the lift was successful 60 % of the time for the experienced user and only 20 % of the time for the inexperienced user.

The distance from the base of the robotic arm was 49 cm and the lift was done 40 cm's up. The testing shows that there is a big gap between the experienced user and the inexperienced users success rate. This shows that there is an issue with user friendliness and something that should be addressed in future prototypes. The success rate of only 60 % even for the experienced user is not enough for a user who plans on using the product daily. Therefore the consistency of the prototype needs to be addressed.

The performance time failed during the test it was not capable of having an average lift time of 30 seconds or less however during testing it was shown that the lift can be performed in under 30 seconds. This means that the prototypes performance time must be lowered for it to be applied in real life scenarios.

The issue with the gripper overloading is something that must be corrected, since it caused the prototype to fail in 40% of cases for the experienced user or 80% for the inexperienced. This is something that must be fixed since if the prototype fails in close to halve of the lifts at best, it is useless at least halve of the time.

The prototype could lift the filled bottle of water and had no issue with it. The prototype was shown to have mayor issues completing the lift, there are certain errors that occur at a

regular basis that necessitates a complete reset of the setup for another lift to be performed. The prototype showed that it is capable of performing a lift and performing the action, however it is inconsistent and needs to be easier to use. Currently the performance time is too high for the average lift and must be lowered. The problem with overloading the gripper must be either fully eliminated or minimized to ensure consistency.

Success criteria			
Requirements	Criteria	How to Test	Pass/fail
EMG signal received	Receiving EMG signals from the Myo armband	A user wears the Myo armband and a computer receives a signal	pass
EMG signal read/translate	Get a signal that the robot can use	Interpreting the signal into code	pass
EMG cooperation	Use EMG to move the robotic arm	Using the Myo armband to move the robotic arm to desired destinations	pass
Horizontal reach	Have a reach of 45.45 cm in both directions	Pick up a water bottle from both directions	pass
Vertical reach	Have a reach of 40 cm	Lift up a water bottle 40 cm	pass
Lift	Able to lift 600 grams	Carry a full water bottle to simulate drinking	pass
Performance time	Start and finish drinking from a bottle within 30 seconds	Timing a full cycle time of the task	fail
Regain capabilities	Help to regain some level of capability	Lift a water bottle from a table to mouth to simulate drinking	pass

Figure 8.3: Success criteria for testing

9 Further development

If further development were to continue in the future, some areas would be improved such as the delimitations and others implemented. These were not implemented due to time limitations and resources. This sections will go in depth with some of these problems and where improvements that was decided not to be included at the time.

- **Free movement** - right now, the robot moves from a fixed position to a fixed position, where only the velocity varies. In the future, this would be corrected to free movement that would be controlled by pre-made gestures controlled by the Myo armband, this would still include velocity control. This would eliminate the user having to worry about the placement of a bottle and the users own position and possible reach problems.
- **Degree of freedom** - the manipulator does only resemble a human arm to some degree, it lacks the possibility to rotate the distal end of the radius around the ulna when the hand and forearm rotate. This would make the pouring experience for the user resemble the natural movement more, and hence give the user a better relationship with the prostheses. Additional adding “fingers” to the prostheses would give a better grip on a bottle, and again therefor give a more natural flow to the movement.
- **Performance time** - During testing the performance time of the prototype was to high. This is something that limits the prototypes effectiveness due to it being too slow to be effective in real life application. This can be fixed by turning up the speed of the prototype.
- **The gripper overloading** - The gripper overloaded if the user was not careful, this lead to many test not being successful especially for the inexperienced user. If this were fixed the success rate could be increased.

- **Safety** - Some precaution that ensures the user will not get hurt wearing the prostheses. This will include coding that makes sure movement within certain parameters is impossible. The arm does not produce a lot of force, but with a knife or fork attached it will be able to do some damage. This will be prevented by implementing some safety barriers into the code.
- **Mount to a person** - right now, the arm is not able to be attached to a person via osseointegration due to several factors. One of the main factors being the need to be connected to a power outlet. In the future, some solution with an internal power source must be integrated, to give the sense of freedom for the user. An interlocking system for the prosthesis and an amputee must be designed. This limitation was mostly due to resource restrictions.
- **Aesthetics** - Because this is a prosthesis and the user is supposed to wear it around in daytime, some coating on top of the servos must be made, most likely made of latex giving the look and feeling of human skin. This would enhance the experience for the use, and give a greater connection and not alienate the prostheses.

Some of these further developments were possible for the group to make, if more time was given, others would require funding from a company with an interest in amputees and prosthesis. The group will in the future, refine and look into solving some of the minor problems that occurred.

10 Discussion

The prototype, made in this project, was based with a particular case in mind, meaning that this prototype is only going to be capable of moving, within a certain trajectories. This means that any modifications to the environment would cause the prototype to not being capable of solving the current problem. This could be solved by instead of moving in a certain plane, the robotic arm could move within all possible positions. The precision of the data transmitted to the robot is only going to be as effective as the method it is collected by. This prototype uses surface EMG sensors in the Myo, to measure EMG. This could be more effective if intramuscular had been used. The robot has three degrees of freedom this limits the movement of the robot making the simulation of an actual prosthesis more difficult. The robotic arm is not made to be fitted on an amputee, nor does the members of this have the required medical experience to fit a prosthesis on a user. the testing of the prototype would be more realistic if it had been fitted on a user.

From the tests performed it can be seen that there are a few problems that would need to be solved. The first problem that occurred was the problem of overloading in the gripper which would cause the bottle to be dropped. Another problem with the prototype was the fact that when lifting a full water bottle it would not move smoothly to its destination. At some points it was not possible for the test subjects to grab the bottle and lift in under 30 seconds, although a significant difference could be seen between the experienced test subject and inexperienced test subject showing that improvement is possible.

11 Conclusion

A Myoelectric prosthesis is a prosthesis that is meant to substitute a lost limb. The prosthesis is controlled through electromyography (EMG). Compared to other types of upper-limb prosthesis, this kind are one of the most functional because it restores lost capabilities the user would not have with other prosthesis. However, they present some disadvantages as being heavier and more expensive. Upper-limb prostheses are currently seeing improvements. However, the prostheses on the market at the moment have a similar performance in efficiency as the ones made ten years ago. The aim of this project is to use the Myo armband and the CrustCrawler robotic arm to simulate the case of lifting a water bottle to the users mouth. The prototype made for this project is the combined result of the two pieces of hardware available and the software to connect the hardware. The program for the prototype is a combination between a pre-made program called myoGestureMaster which can be found in appendix A, correspondence with the creator can be seen in appendixB. The program is capable of saving gestures made by the user, and afterwards, by using covariance and correlations it can calculate the similarity between the saved gestures and the gestures measured by EMG while its being used. The second part of the program takes the data gathered in the previous step and translates it into control of the CrustCCrawler robotic arm. During testing it is concluded that the prototype is capable of lifting a 600 gram water bottle to the users mouth. However the performance time was slow and the success rate needs to be increased for real life application. It Is shown that a Myoelectric prosthesis can be designed to assist a person with upper-limb amputation.

Bibliography

- [1] Wikipedia. Prosthesis, 10th of december 2016. URL <https://en.wikipedia.org/wiki/Prosthesis>.
- [2] Advanced amputee solutions LLC. Amputee statistics you ought to know, 2012. URL <http://advancedamputees.com/amputee-statistics-you-ought-know>.
- [3] George ElKoura and Karan Singh. Handrix: Animating the human hand. *Department of Computer Science, University of Toronto, Toronto, Canada*, accessed 11. of december 2016. URL <http://www.dgp.toronto.edu/~gelkoura/noback/scapaper03.pdf>.
- [4] dictionary.com. Prostheses, accessed: 19/11-2016. URL <http://www.dictionary.com/browse/prosthetic>.
- [5] biomed.brown.edu. Upper extremity prosthetics, accessed: 20/11-2016. URL http://biomed.brown.edu/Courses/BI108/BI108_2003.Groups/AthleticProsthetics/UpperLimb.htm.
- [6] MD Douglas G. Smith. Introduction to upper-limb prosthetics: Part 1. April 2007 accessed: 20/11-2016. URL <http://www.amputee-coalition.org/resources/upper-limb-prosthetics-part-1/>.
- [7] Dorcas Beaton Elaine Biddiss and Tom Chau. Consumer design priorities for upper limb prosthetics, disability and rehabilitation: Assistive technology, 2:6,346-357,. published 09/06-2009. URL <http://dx.doi.org/10.1080/17483100701714733>.

- [8] Dario Farina and Sebastian Amsüss. Reflections on the present and future of upper limb prostheses. *Expert Review of Medical Devices*, 13:4, 321-324, 2016. URL <http://dx.doi.org/10.1586/17434440.2016.1159511>.
- [9] ELAINE A. BIDDISS and TOM T. CHAU. Upper limb prosthesis use and abandonment: A survey of the last 25 years. *Prosthetics and Orthotics International*, September 2007. URL https://www.researchgate.net/profile/ElaineBiddiss/publication/5867605_Upper_limb_prosthesis_use_and_abandonment_A_survey_of_the_last_25_years/links/568a8b0d08ae1e63f1fbd39f.pdf.
- [10] Kerstin Caine-Winterberger Stewe Jönsson and Rickard Brånemark. Osseointegration amputation prostheses on the upper limbs: methods, prosthetics and rehabilitation. *Prosthetics and Orthotics International*, April 2011. URL <http://poi.sagepub.com/content/35/2/190.full.pdf+html>.
- [11] Myo. What's included?, Accessed: 21/11-2016. URL <https://www.myo.com/techspecs>.
- [12] Thalmic Labs. Myo developer - sdk source, 2013-2016. URL <https://developer.thalmic.com/downloads>.
- [13] Thalmic Labs. Myo developer - sdk description, 2013-2016. URL <https://developer.thalmic.com/docs/api-reference/platform/the-sdk.html>.
- [14] Dario Farina Roberto Merletti. Surface electromyography: Physiology, engineering and applications, page 38-39. accessed 19. of december 2016.
- [15] Peter Konrad. The abc of emg, a practical introduction to kinesiological electromyography, march 2006.
- [16] Inc CrustCrawler. Crustcrawler information, 2002 - 2016. URL <http://www.crustcrawler.com/>.
- [17] H. Harry Asada. Introduction to robotics. *Massachusetts Institute of Technology*, accessed 22. of novemer 2016.

- [18] Robotis. Robotis e-manuel: Mx-28t / mx28r, 12th of september 2016. URL http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28.htm#Actuator_Address_1A.
- [19] Eric W. Weisstein. Covariance. from mathworld—a wolfram web resource, 20. of December 2016. URL <http://mathworld.wolfram.com/Covariance.html>.
- [20] Microsoft. Multithreading: Creating worker threads, 20. of December 2016. URL <https://msdn.microsoft.com/en-us/library/69644x60.aspx>.
- [21] playground.arduino.cc. Arduino c++ for windoes, accessed: 20th of december 2016. URL <http://playground.arduino.cc/Interfacing/CPPWindows>.
- [22] Michael E. Ferguson. Dynamixel library, 2008-2011 accessed: 20th of december 2016. URL <https://github.com/vanadiumlabs/arbotix/blob/master/libraries/Bioloid/ax12.h>.
- [23] ROBOTIS. Dynamixel ax-12 user's manual, 14th of june 2006. URL [www.trossenrobotics.com/images/productdownloads/AX-12\(English\).pdf](http://www.trossenrobotics.com/images/productdownloads/AX-12(English).pdf).

A Code - Github

`https://github.com/skaihoj/myogesturep3`

For access to the code, email `skaiha14@student.aau.dk` with your GitHub account name.

The main project is `myoGestureMaster/myoGestureMaster.sln`

The robot library is located in `myo-sdk-win-0.9.0/src/robot.cpp`

B Code - Obtainment

This following is the corresponding between the creator of the Myogesturemaster and group 369.

From: 권혁민 [guraosdl1030@naver.com]
Sent: 26 November 2016 13:58
To: Daniils Avdejevs
Subject: RE: HA: Myo code.

대용량 첨부파일 1개(119MB) 대용량 첨부 파일은 30일간 보관 / 100회까지 다운로드 가능
PC저장 네이버 클라우드
myoGestureMaster.zip 119MB
다운로드 기간 : 2016/11/26 2016/12/26
Hi! I'm hyukmin send myo code!

I found out the code you first require. So I send you.

Best regards!

-----Original Message-----

From: "Daniils Avdejevs" <davdej15@student.aau.dk>
To: "권혁민" <guraosdl1030@naver.com>;
Cc:
Sent: 2016-11-24 (목) 18:30:49
Subject: HA: Myo code.

Thank you very much for the code.

Going to look into it as soon as I can and ask questions if I have them.

Best regards,

Daniils Avdejevs

От : 권혁민 [guraosdl1030@naver.com]

Отправлено : 23 ноября 2016 г. 2:59

Кому : Daniils Avdejevs

Тема : Myo code.

대용량 첨부파일 1개(71MB) 대용량 첨부 파일은 30일간 보관 / 100회까지 다운로드 가능
PC저장 네이버 클라우드 MyoEmgData1.zip 71MB
다운로드 기간 : 2016/11/23 2016/12/23
Hi, I'm Hyukmin.

I attached two file (code and ppt). PPT is simple explanation.

The core principle is a correlation analysis.

For example, I store EMG data for five hand motions. While the program is running, getting the correlation value between stored EMG data and current EMG data.

But EMG from MYO rocked rapidly, I calculate the average of n EMG data.

In the program, the numbers of top is averages.

If you need any explanation, contact me.

p.s / The code is disorderly. sorry. :)

Thanks.

With kind regards.