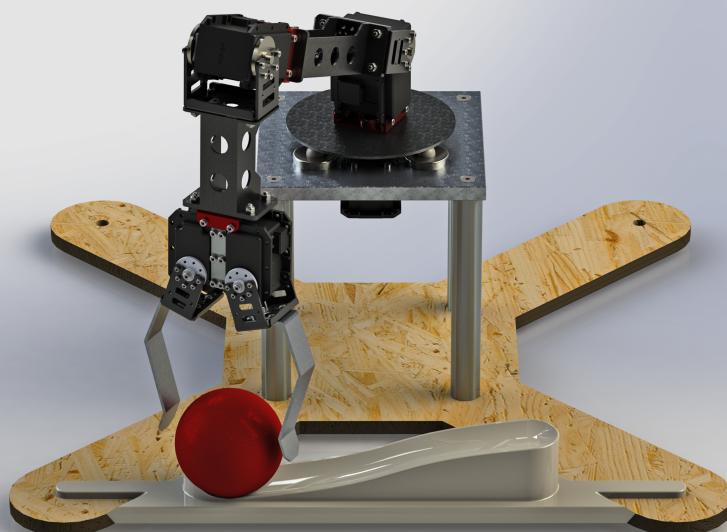

Aiding upper limb rehabilitation of hemiparetic patients using a robotic manipulator controlled by surface electromyographic signals.

Manipulating the surroundings

Project Report
Group 366



Aalborg University
Robotics



AALBORG UNIVERSITY STUDENT REPORT

Title:

Aiding upper limb rehabilitation of hemiparetic patients using a robotic manipulator controlled by surface electromyographic signals

Theme:

Manipulating the surroundings

Semester:

3rd semester of robotics.

Project Period:

Fall semester 2016

Project Group:

366

Group Members:

Casper Mariager

Daniel Fischer

Hjalte Nielsen

Jakob Kristiansen

Jesper Frendrup

Marius Hensel

Supervisor:

Per Printz Madsen

ppm@es.aau.dk

Erika Geraldina Spaich

espaich@hst.aau.dk

Report:

80 pages

Submitted:

December 21st

Department of
Electronic Systems
Robotics
Fredrik Bajers Vej 7B
DK-9000 Aalborg
www.es.aau.dk

Abstract:

Strokes are one of the leading causes of disability in the entire world, often causing hemiparesis. In order to address this problem and improve the quality of life for hemiparetic patients with a partially automated solution, a fundamental analysis of the problem was conducted. Based on the analysis a rehabilitation game, for patients with upper extremity disabilities, consisting of a robotic manipulator and a commercially available sensing device is proposed. In order to develop this game, kinematics of the manipulator are calculated and a control system with a PID controller is implemented to control movements of the manipulator in cartesian space. A prototype of the game was built and tested in relation to specific requirements in order to evaluate the product. The prototype fulfilled all the requirements, but it was not without flaws. The final prototype adequately demonstrated the concept of the game even though it never was tested on hemiparetic patients.

CONTENTS

Preface	1
I Prologue	2
1 Introduction	3
1.1 Cause and effects of strokes	3
1.2 Initial problem statement	6
II Analysis	7
2 Rehabilitation of hemiparetic patients	8
2.1 General rehabilitation practises	8
2.2 Robotic rehabilitation	10
3 Electromyography	13
3.1 EMG origin	13
3.2 EMG signal detection	16
4 Hardware analysis	18
4.1 The Myo Armband	18
4.2 The CrustCrawler	22
5 Control Theory	24
5.1 Open-loop systems	24
5.2 Closed-loop systems	24
5.3 PID controller	25

III Development	28
6 Conceptualisation	29
7 Delimitations	30
8 Final problem formulation	32
9 Initial prototype setups	33
9.1 CrustCrawler setup	33
9.2 Myo Armband setup	34
9.3 Kinematics	35
9.4 Data collection	39
10 Control systems	41
10.1 First CrustCrawler control system	41
10.2 Second CrustCrawler control system	46
11 Prototypes	54
11.1 Description and specific requirements	54
11.2 Testing and requirement verification	55
12 Discussion	62
13 Conclusion	65
A Micro-controllers	66
B Testing of the prototype	67
C Linear model of dynamic behaviour	69
D Signal processing and analysis of EMG	71
D.1 Signal processing	71
D.2 Signal Analysis	73
Bibliography	76

PREFACE

This report was written on 3rd semester of the Aalborg University (AAU) Robotics bachelor. It revolves around the topics of control theory- and systems, robotic manipulators, electromyography and effects of cerebrovascular diseases such as hemiparesis. The project is developed according to the problem based learning method of AAU meaning for which reason it is developed as a group project that includes a problem analysis of a socially relevant problem followed by development of a prototype to illustrate a solution to the problem. Thanks to AAU for supplying an electromyographic recording device (Myo Armband) and a robotic manipulator (CrustCrawler). All videos mentioned in the report are also available via:

www.youtube.com/playlist?list=PLHSjUYowigf0Z_ZX-DEjN0gH03MByoYfb

And all code regarding this project is also available on:

www.github.com/hjalte33/myoCC

Aalborg University, December 21, 2016



Casper Mariager
cmaria15@student.aau.dk



Daniel Fischer
dfisch15@student.aau.dk



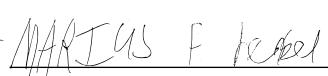
Hjalte Nielsen
hbdb15@student.aau.dk



Jakob Kristiansen
jkris15@student.aau.dk



Jesper Frendrup
jfpe15@student.aau.dk



Marius F. Hensel
mhense15@student.aau.dk

Part I

Prologue

INTRODUCTION

This chapter examines the cause and effects of strokes along with statistics on these topics. The main focus of the project is to create a robotic device that can assist in upper limb rehabilitation.

Stroke is defined as:

“A sudden and lasting impairment of brain function caused by obstruction of or haemorrhage from the cerebral blood vessels.” [1]

Strokes are one of the biggest causes of death and disability. Worldwide it is ranked as the 2nd highest cause of human mortality, and the 3rd highest cause of disability [2]. In 2014, according to *Sundhed.dk*, it was estimated that 12.000 people fall victim to strokes every year in Denmark alone, whereas the Danish National Board of Health estimated an even higher number of 15.000 yearly stroke occurrences in 2015. One third of these are recurrent events and two thirds are first time stroke patients [3].

Estimations show that the number of stroke patients in Denmark will rise in the coming years since the elderly population, where two thirds of all patients are located, is growing in Denmark. This can become a big burden on the health care system since the treatment and rehabilitation of stroke patients is very demanding in terms of working hours and resources in general [4].

For many years the prospects for the future of stroke patients were not regarded with optimism. However, towards the end of the 20th-century going into the beginning of the 21st century, improvements in rehabilitation practises confirmed the possibility of recovering lost functional operability. At this point in time, an increased understanding of the causes of cerebrovascular diseases was also achieved with attention aimed at e.g. smoking and high blood pressure as risk factors. This knowledge sparked an interest in the prevention of strokes. [1]

1.1 Cause and effects of strokes

Stroke occurs as a result of a cerebrovascular disease which is defined as:

"Any disorder of the blood vessels of the brain and its covering membranes (meninges)" [5].

A cerebrovascular disease may involve rupture of diseased blood cells, known as cerebral or subarachnoid haemorrhage (hemorrhagic strokes), or shortage of blood supply to the brain due to cerebral thrombosis or embolism (ischemic strokes) [5]. Known contributing factors to the development of cerebrovascular diseases are high levels of cholesterol in the blood (hypercholesterolaemia), high blood pressure (hypertension) and hormone disruption among others [6]. In addition, cerebrovascular diseases are more inclined to develop as age increases [7].

The clinical syndrome accompanying a sudden and sometimes severe case of cerebrovascular disease is referred to as a cerebrovascular accident (CVA) [8]. As mentioned in the initial definition of stroke, CVA leads to stroke. Strokes give a sudden loss of body functioning because of disturbances in the blood circulation of the brain. Statistics on the cause of strokes in the United Kingdom show that strokes due to thrombosis and embolism account for 85% while strokes caused by haemorrhages make up 15% of all mentioned incidents [1]. Depending on age as well as other possible illnesses and the treatment of those, the mortality rate of stroke patients after five years is between 40-70% [9].

The items in the list below highlights some of the consequences of impaired brain function due to strokes. The list is based on information from *Christiana Care Health System* and *Sundhed.dk*. [10, 11]

- **Speech and language**

After a stroke, problems can occur when trying to talk and it can be difficult to find the right words to say, to name objects correctly, or to comprehend what others are saying. Others may experience difficulties in related skills such as math, reading or writing. This does not mean these skills are lost forever, because many times, these skills can be relearned through therapy. Otherwise alternate ways of communication are formed.

- **Memory**

A patient's short-term memory can be affected by a stroke, and they may not be able to retain what happened five minutes ago. The same goes for the long-term memory as some patients also cannot retrieve memories from the past. Verbal and visual memory can be affected by strokes as well, e.g. in the ways of not being able to name items on a shopping list or recognizing faces.

- **Visuoperceptual skills**

The ability to pay attention to one side of the visual field can be affected, even though the patient does not have any vision problems. The inability to pay attention with one eye (often the left) may cause the patient to bump into

walls or trip over objects in their path. [12]

The neglect of space can sometimes be so severe that patients may deny that a body part even belongs to them or will not use the one side of the body despite no actual loss of physical ability.

This can also lead to difficulty in solving puzzles or drawing and cause problems with reading.

- **Emotional functioning**

After a stroke the patient is at high risk of developing emotional problems such as depression and mood swings, where depression often goes undiagnosed and untreated.

- **Personality changes**

If a patient experiences personality changes they often involve a loss of interest in activities or increased spontaneity resulting in e.g. social inappropriateness. These changes can be treated and often overcome in time. [10]

- **Hemiparesis**

“... paresis or impairment of muscular strength affecting one side of the body.” [13] Hemi- is a prefix which means half or one half, in the word hemiparesis it refers to the fact that only one half of the body is affected and paresis means partially paralysed [14, 15]. Hemiparesis can be caused by different reasons such as traumatic brain injuries or strokes, where the most common cause is strokes [2]. The reason why people lose function of a limb after a stroke is due to the lack of blood supply to a specific part of the brain, which then cause brain cells that were used to control this limb to die [1].

A study of ischemic stroke survivors of people over 65 years showed that 50% of the mentioned group suffered from hemiparesis. This observation was made six months after the occurrence of the strokes [2].

If a person has fallen victim to a stroke, rehabilitation will start as soon as possible, often within two days after the stroke has occurred, and should continue as necessary after release from the hospital. [16] Although some stroke survivors recover quickly, most need some form of long-term rehabilitation, possibly months or years after the occurrence of a stroke. [17]

Rehabilitation is in general a complex and unique process involving experts from a variety of fields within the world of medicine and physiology such as doctors, physiotherapists, occupational therapists and nurses. Generally, these experts will not be present for the majority of the rehabilitation since little time is available for training in direct collaboration with specialists [18].

This project aims to assist the rehabilitation of stroke patients suffering from hemiparesis with a robotic device. One reason for such a device is to provide a tool

for rehabilitation that reduces the need for presence of specialised personnel during training. The device will be controlled by electromyographic (EMG) signals acquired from remaining muscle activity in hemiparetic patients.

Based on this introduction of the problem, the focus of the project is delimited to the following initial problem statement.

1.2 Initial problem statement

How can robotic equipment interact with hemiparetic patients in order to aid the rehabilitation?

Part II

Analysis

REHABILITATION OF HEMIPARETIC PATIENTS

The following chapter will give a definition of rehabilitation and describe how hemiparetic patients can be categorised while providing insight into different rehabilitation methods for hemiparetic patients. The application of robots in rehabilitation will also be investigated.

WHO's definition of rehabilitation:

"Rehabilitation of people with disabilities is a process aimed at enabling them to reach and maintain their optimal physical, sensory, intellectual, psychological and social functional levels. Rehabilitation provides disabled people with the tools they need to attain independence and self-determination." [19]

2.1 General rehabilitation practises

As previously mentioned, one of the common sequelae of stroke is hemiparesis, which causes the patient to lose motoric abilities in one side of the body [20]. The severity of the paresis as well as the side affected can vary from case to case. In spite of this, the methods used for rehabilitation by physiotherapists and occupational therapists very often involve elements of cardio- and or strength training. **Cardio training** involves training continuously or in intervals, where it is recommended to have a target heart rate at 75% of maximum yielding capacity calculated based on Karvonen's Formula [21]. **Strength training** have been a controversial rehabilitation practise in many years due to theories from *Bobath* which concluded negative neurological effects. Later studies suggest this to be untrue and some even have opposite conclusions. The definition of strength training is:

"A form of training with load on individual muscle groups in order to improve the musculature, while the load on the circulatory system is minimal." [22]

A method for upper-limb rehabilitation is **CIMT** (Constraint-Induced Movement Therapy) where the unaffected arm is put into a sling, forcing the patient to use the affected arm [23].

Some hemiparetic patients receive **Functional Electrical Stimulation** (FES) due to the fact that they generally have very little muscle activity left. FES is a method of rehabilitation which aid the contraction of specific muscles in order to make the patient perform specific movements. There are two ways of using FES, either by implanting electrodes or by mounting them on the surface of the muscle.

Another common practise used in rehabilitation of patients with stroke is known as **Activities for Daily Living** (ADL), which is done in collaboration with an occupational therapist and involves daily activities e.g. cooking. ADL can be divided into two subcategories: Personal ADL (PADL) and Instrumental ADL (IADL) where PADL could involve activities relating to personal hygiene and IADL could be chores in the house. It is also recommended that the patient have a leisure activity. An adequate balance is required to execute ADL exercises, which is why some patients receive **balance training**, especially patients who got a deteriorated balance as a consequence of stroke. One way of training the balance is by utilising a balance board.

Rehabilitation exercises may be assisted by **virtual reality** to simulate environments or provide some context to the exercises that are performed. [24, 20]

As the rehabilitation process can vary between individual patients, the goal can also be different in terms of strength, mobility etc.

In a publication regarding guidelines for physiotherapists and occupational therapists, the Health Department of Denmark distinguishes between two levels of recommendations when evaluating rehabilitation practises.

Strong recommendation ↑↑

The health department gives a strong recommendation when the total benefits of the intervention are assessed to be bigger than the cons.

Weak or conditional recommendation ↑

The health department gives a weak or conditional recommendation when it is believed that the benefits of the intervention is marginally greater than the disadvantages, or the available evidence does not rule out a significant advantage of an existing practise, while it is estimated that adverse effects are few or absent.

The evaluation of practises for occupational therapists and physiotherapist is given in the list below, however, CIMT and robot assisted methods are excluded from

these guidelines. This report was released in 2014 meaning that newer evidence that will interfere with current knowledge might have come to light. [24]

- **Cardio training - ↑↑**

This practise is highly recommended for both upper and lower limb rehabilitation. The purpose of cardio training is to better the cardiovascular performance, which is done by putting high load on the respiration and the circuit function. This practise also increases the general health of the patient.

- **ADL - ↑↑**

This practise is highly recommended. It can be used as both a method or a goal.

- **Strength training - ↑**

In order to strengthen the motor function of the patient, it is recommended to have at least 6 weeks of strength training with 3 sessions per week.

- **FES - ↑**

The FES will not be available for people with either completely lost or slightly lost muscle function, only those who have moderate to severe reductions in muscle functionality. FES for use on upper extremities is not recommended.

- **Balance training - ↑**

Balance is an important element in rehabilitation, but the documentation of its importance is limited for which reason it is given a weak recommendation.

- **Virtual reality - ↑**

For rehabilitation of upper extremities with virtual reality, it is recommended that patients only have mild to moderate reduction in muscle function. In the guidelines from the Danish Health Department, virtual reality was not given a recommendation as a practise yet due to the technology being relatively young. However, in the 2016 guidelines for rehabilitation practises of stroke patients by The American Heart Association/American Stroke Association, virtual reality was recommended since evidence showed improved progress in gait rehabilitation. Virtual reality also showed promising results in reference to rehabilitation of upper extremities [20].

2.2 Robotic rehabilitation

Rehabilitation robots have the advantage of allowing the therapist to see progress in numeric values e.g. torque while being able to increase session intensity and duration of rehabilitation sessions which is important in relation to effective training [25]. Furthermore, it has good repeatability which can help the patient make a

specific movement many times in row, where the robot can be passive or actively, supporting the patient [26, 27]. One of the more common used robotic systems is the LOKOMAT, which is an exoskeleton for the lower body that helps the patient walk on a treadmill [28]. Another use of robotic technology in rehabilitation is the ARMin robot which is used for upper limbs and can be seen on figure 2.1. A group of researchers from ETH Zurich in Switzerland did a study with the use of ARMin and compared it to conventional methods with two different groups. The study was conducted over 34 weeks, and evaluation with Fugl-Meyers Assessment (FMA) showed that the patients who were using ARMin gained better arm function, but had lower gains in mean strength. [29]

In order to acquire a different perspective on the categorization of hemiparetic patients compared to the one obtained from literature, a written interview was conducted with Jim Jensen, MSc in rehabilitation.

Jim Jensen states that in the start of the rehabilitation process an assessment of the patient's degree of hemiparesis is conducted. Many different methods are available for this, e.g. testing how well a patient can use the arm for daily tasks such as eating, getting dressed etc. In order to get more instrumental investigations of the patient, methods such as transcranial magnetic stimulation (TMS) are used to evaluate the damage of a stroke [30]. When all the tests are done the patient will often be put into one of the three categories: mild, moderate or severe.

A different categorization, based on FMA, proposes four divisions: slight, moderate, marked and severe. FMA evaluates the following elements: Motor function, sensory function, balance, joint range of motion and joint pain. Each element is



Figure 2.1: ARMin is an exoskeleton for rehabilitation of upper limbs. - ETH Zürich.

given a score which summed will give a total value that will lie in an interval relating to one of the previously mentioned divisions. [31, 32]

As previously mentioned, the use of ARMin showed lower gains in mean strength. The research article suggested that the reason for this is that the ARMin robot is too supportive and does not provide enough strength training [29]. Only 18% of patients manage to retrieve full functionality of the hand and arm after 6 months. Utilising robotic aid it might be able to prolong the 6 months rehabilitation gain

In order to see how big a part intensity played in the rehabilitation process a study was conducted with four different patients, all of which have had a stroke at least twelve months prior to the start of the study. Two of the patients received training 3 hours a week, and the other two received 4 hours a week. Throughout the study there was an indication that intensive training is beneficial for the patient. Another finding in the study related to the movements that should be focused on, where a combination of wrist, elbow and shoulder movement was found to be optimal, due to the fact that many activities of daily living consists of these. In addition, motivation, cooperation and satisfaction can enhance the overall result of rehabilitation. [33] With robotic rehabilitation it is possible to utilise different approaches to rehabilitation, such as passive which means the patient will not use any effort whereas active rehabilitation do require effort [34], assisted rehabilitation [35] can be a middle-way of the two. Lastly, resistance rehabilitation [36] is a form of rehabilitation that requires more effort than active rehabilitation.

In this chapter the understanding of both the conventional and newer methods of rehabilitation was extended. These methods include strength- and cardio training, CIMT, FES and ADL, where cardio training and ADL activities are strongly recommended by the Danish Health Department, and FES is only recommended under certain conditions and not at all for rehabilitation of upper extremities. Furthermore it was explained how the degree of severity of hemiparetic patients can be categorised based on the FMA assessment. Key factors of effective rehabilitation were found to be intensity, duration, motivation, satisfaction and cooperation and it was mentioned how robots can be an effective tool to increase intensity and duration during rehabilitation courses. However, the study of the ARMin also revealed that robotic assistance can result in a small gain of mean strength compared to those who receive conventional rehabilitation. Finally, it was discovered that rehabilitation exercises aimed at upper limbs should incorporate movement of the wrist, elbow and shoulder joints.

ELECTROMYOGRAPHY

The following chapter gives an introduction to EMG signals. This involves a description of signal origin and detection methods in order to give a basic understanding of EMG. The project is focusing on EMG as a method of controlling a robot for which reason nerve signals and body chemistry will only be covered roughly.

Electromyography (EMG) has the following definition:

“Electromyography is the study of muscle function through the inquiry of the electrical signal the muscle emanate” [37].

This project focuses a branch of EMG called kinesiological EMG, which involves an investigation of voluntary neuromuscular activation of muscles such as lifting an arm or moving a leg. Kinesiological EMG can be used in a variety of ways. Within the field of medical research it can be used for gait and posture analysis, athletes might be interested in using it for motion analysis in order to improve sports activities while others might see a use for kinesiological EMG as an evaluation tool when developing products with ergonomic designs. The most interesting use of EMG, in relation to this project, is in the field of physical therapy where it can aid rehabilitation. EMG makes monitoring of muscle activity possible which is a particularly important feature that will be investigated in the remainder of this section and used throughout this project. [38]

3.1 EMG origin

In order to move a body part, muscles will have to contract. The process of muscular contraction starts in one of the upper motor neurons, typically located in the cerebral cortex of the brain. The purpose of the upper motor neurons is to send signals to the lower motor neurons, which are located in either the brainstem or the spinal cord. The lower motor neurons are part of **motor units** which are the smallest functional units in the body that describe neural control of muscle contractions. Specifically, the motor unit consists of a lower motor neuron, including

the dendrites and the axon connected to it as well as the muscle fibers innervated by it, see figure 3.1. The muscle fibers of a motor unit act in unity in the innervation process, meaning that the motor neuron activates all the muscle fibers when a signal is sent along the axon, hence the name motor unit. [38]

When a muscle is not contracted, the muscle fiber membrane (sarcolemma), has a difference in electrical charge on the internal and external side of the membrane due to different concentrations of sodium ions (Na^+) and potassium ions (K^+) on each side. The internal side of the membrane has a negative charge compared to the external side meaning that, when the external charge is subtracted from the internal charge, a voltage of approximately -80 to -90 mV is obtained, which is referred to as the **resting potential** of the muscle fiber membrane. When contracting a muscle, motor neurons of different motor units are excited causing an electrical impulse, called an **action potential**, to travel along the axon towards the muscle fibers of a particular motor unit, given that the motor neuron excitation surpassed a certain threshold. At the motor endplate, located at the end of the axon, this electrical action potential causes neurotransmitters in the form of acetylcholine to react with receptors of the sarcolemma. If a sufficient amount of reactions occur, the diffusion properties of the muscle fiber membranes change to allow sodium ions to enter the membrane essentially causing a depolarization of the membrane potential from the -80/-90 mV mentioned earlier up to as much as 20/30 mV. When a

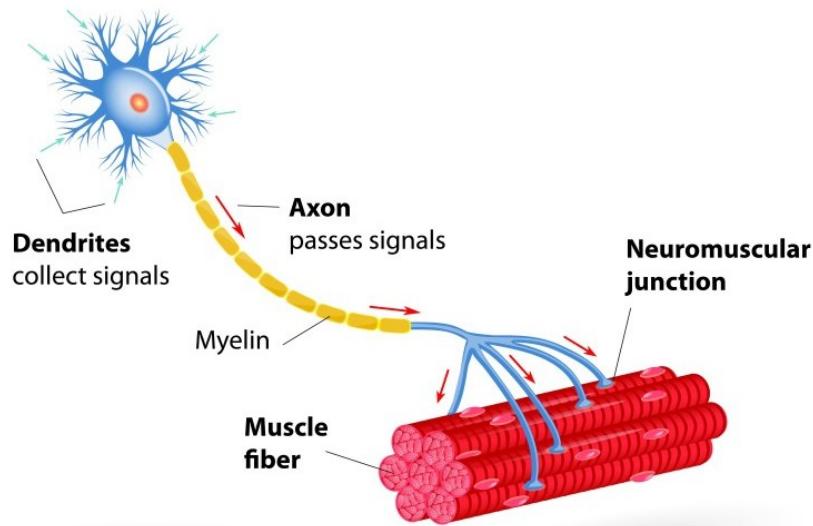


Figure 3.1: Anatomy of a motor unit. The lower motor neuron, illustrated in blue, is stimulated by upper motor neurons through the dendrites. The axon passes the action potentials generated in the lower motor neuron towards the neuromuscular junction where the electrical impulse is transferred to the muscle fibers through a chemical reaction. <http://www.muaythaischolar.com/motor-unit/>

threshold of approximately -50 mV is passed during the depolarization process, an action potential is generated along the muscle fiber membrane, which ultimately causes the muscle fiber to contract. Repolarization of the membrane follows immediately after the action potential has been generated and it is the difference in membrane potential as a result of continuous generation of action potentials that can be measured by electrodes and therefore makes up the basis of EMG. Figure 3.2 shows all mentioned elements of an action potential. The electrodes cannot pick up action potentials of single muscle fibers. In reality they register various **motor unit action potentials** which have been superposed. A motor unit action potential is the summed magnitude of the action potentials from all innervated muscle fibers in one motor unit. [38]

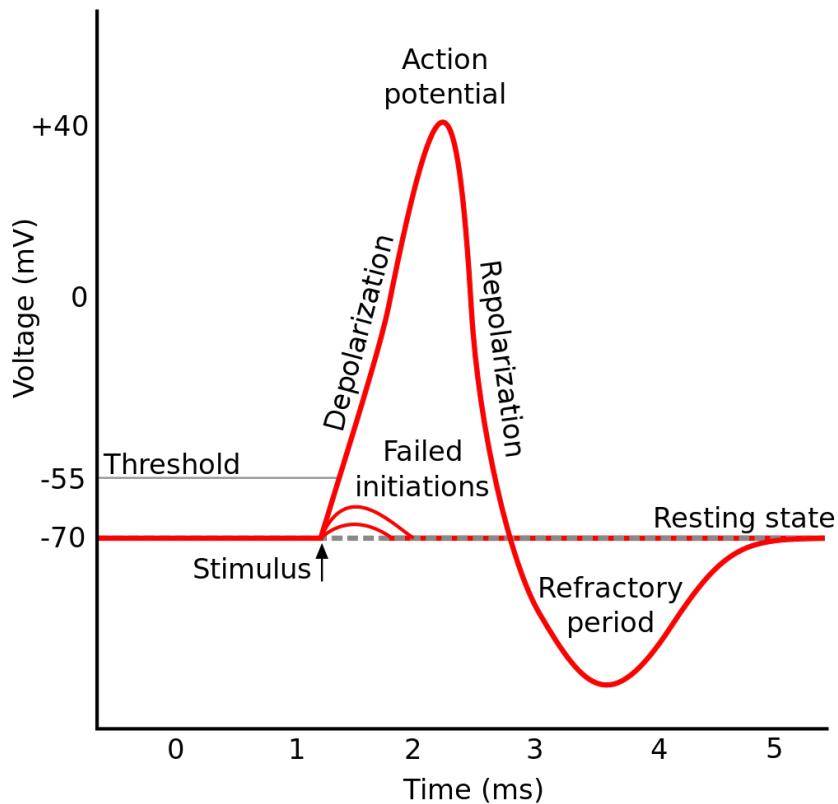


Figure 3.2: An illustration of the different phases of an action potential as it passes a point on a cell membrane. At 1 ms, a signal is received which will generate an action potential if the threshold potential is exceeded. The depolarization is quickly followed by a repolarization reestablishing the resting potential once again. https://commons.wikimedia.org/wiki/File:Action_potential.svg

3.2 EMG signal detection

Two different kinds of electrodes are available for recording of EMG signals, namely invasive and noninvasive electrodes. The invasive electrodes are needle like sensors which are pushed through the skin directly into the muscle to be monitored. This method gives very accurate signals and it is very effective when EMG from deep-seated muscles is of interest, however it is also related to discomfort for the patient. The other type of electrodes, the surface electrodes, are mounted on the skin and have small flat metallic contact points. These metallic surfaces are placed on the skin directly above the targeted muscle. EMG recorded from surface electrodes is referred to as surface EMG (sEMG). Surface electrodes can satisfactorily measure EMG from the most important limb and trunk muscles. At the same time they are easy to handle and the comfort level is greater for which reason they are often used for studies of kinesiological EMG. However, they are more prone to pick up physiological crosstalk from other nearby muscles and they are limited to recording of muscles close to the skin. [38]

The signal characteristics of the EMG varies in both magnitude and frequency depending on the number of recruited motor units during a contraction and the firing frequency of each motor unit action potential respectively. An EMG signal recorded directly from the muscle without any processing or filtering, besides amplification, is referred to as a raw EMG signal. In a raw EMG signal, muscle contractions will show as a dense area of pulsations, while a relaxed muscle will produce the baseline of the signal. The characteristics of the baseline signal as well as the EMG signal in general is affected by several detection conditions. Detection conditions are generally an important field of EMG recording as they can affect the signal outcome in different ways. Mainly, there are five areas to keep in mind when detecting EMG. [38]

1. Tissue characteristics

If EMG is recorded from the surface of the skin, there are several tissue characteristics which might affect the shape of the signal received by the electrodes. The tissue type, thickness of tissue between muscle and electrode as well as tissue temperature are all parameters that can change the characteristics of the EMG signal. For example, the amplitude of the raw EMG signal will decrease as the thickness of subcutaneous fat increases. As these parameters vary between individuals, the comparison of amplitude characteristics of raw EMG signals from multiple people is unreliable. [38]

2. Physiological crosstalk

Physiological crosstalk refers to EMG that is picked up from surrounding muscles to the one that is of actual interest. Up to 10-15% of a given EMG signal can originate from cross talk, but it may not appear at all. [38]

3. Changes in the geometry between muscle belly and electrode site

The electrodes should be placed centrally on top of the muscle belly, when the muscle is fully contracted, in order to avoid distance between the place of signal origin and site of signal detection as creation of this distance would affect the outcome of the EMG signal. [38]

In continuation, preparation of the skin before applying electrodes is important with reference to the quality of the sEMG signal. First step of the preparation involves removement of hair in order to obtain a good adhesion between the electrodes and the skin. Next, the skin should be cleaned: The cleaning is commonly done with either abrasive and conductive cleaning pastes, fine sand paper in combination with alcohol pads or purely with alcohol or water. It is important not to damage the skin during preparation. However, a slight red colour of the skin indicates good skin conditions. The preferred method of skin preparation greatly depends on the use of the EMG. If EMG from static or slow movements is analysed with interest confined to the general trend of the EMG signal, a pure alcohol cleaning is usually sufficient. If, on the other hand, the EMG study involves fast, highly accelerated movements, a very careful preparation of the skin is necessary. [38]

Signal processing is applied to the raw EMG signal in order to compensate for some or all of the previously mentioned detection conditions or generally to ease interpretation and enable the possibility of quantitative analysis on the EMG signals. Examples of different popular signal processing methods is described in appendix D on page 71.

Basic understanding of EMG together with an understanding of detection methods was acquired. In addition it was discovered that the choice of electrode is based on parameters such as muscle depth, ease of handling and comfort and that the shape of raw EMG signals is inconsistent and influenced by detection conditions for which reason signal processing is a prerequisite to most analyses of EMG.

HARDWARE ANALYSIS

In this chapter the main hardware available for this project will be analysed. The Myo Armband and the CrustCrawler are both pieces of equipment which were made available for this project. This chapter aims to find the capabilities and limitations of these two essential hardware components.

4.1 The Myo Armband

The Myo Armband is a sensing device, created by the company Thalmic LabsTM, capable of gathering EMG data. The armband is constructed of eight links connected through two bands of expandable flex. All eight links contain proprietary EMG sensors, see figure 4.2a on the facing page. Five of the links solely contain the EMG sensors making them thinner than the last three as these are housing additional components, see figure 4.1a.

The three thicker links are placed next to each other where the middle link contains two LEDs on the surface i.e. the Thalmic Labs logo LED and a status LED, see

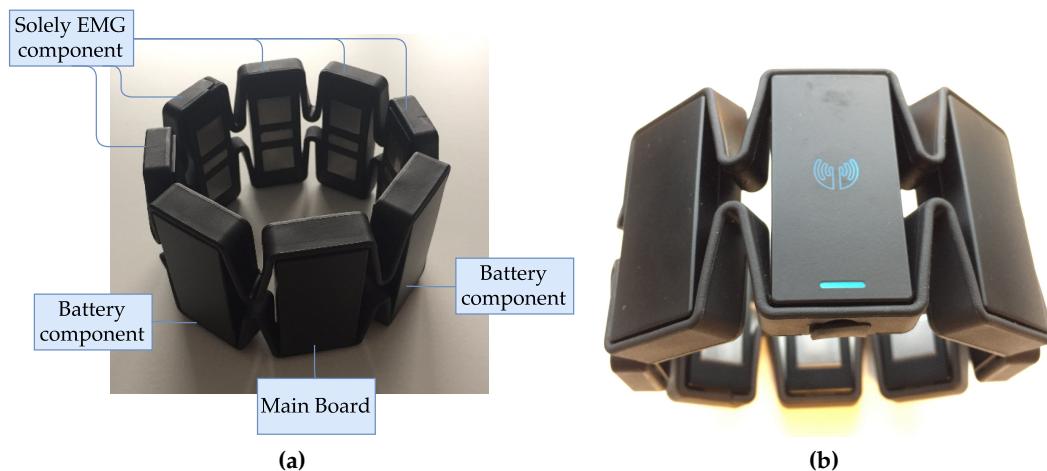


Figure 4.1: (a) The difference in link size is shown in the figure. (b) The two LEDs turned on.

figure 4.1b on the facing page. Furthermore this link contains the main board of the device. On the main board is a Kinetis MK22FN1M0 processor which among other things contains an ARM^(R) CortexTM-M4 120MHz core, a micro USB charging port, a nRF51822 SoC for bluetooth low energy and a MPU 9150 SiP 9-axis motion tracker, see figure 4.2b and 4.2c.

The MPU 9150 SiP contains two chips; a MPU 6050 and a AK8975. The MPU 6050 contains a 3-axis gyroscope and a 3-axis accelerometer whereas the AK8975 contains a 3-axis compass. The last two thicker links each have a 260 mAh lithium ion battery, see figure 4.2d. [39]

The device comes with a micro USB cable, a bluetooth 4.0 dongle and sizing clips, see figure 4.3a on the following page.

Communication between the device and the computer is established via Bluetooth[®] Smart Wireless Technology.

The Myo Armband is designed to be placed on the widest part of the forearm of the user. Figure 4.3b on the next page showcases proper positioning of the armband. From this position the device will collect EMG signals from the skeletal muscles of the forearm through the eight EMG sensors. The Myo Armband comes pre-programmed with the ability to recognise five different gestures, namely *text/fist*, *spread fingers*, *wave in*, *wave out* and *double tap*, see figure 4.4 on the following page for which reason additional signal processing is not a necessity.

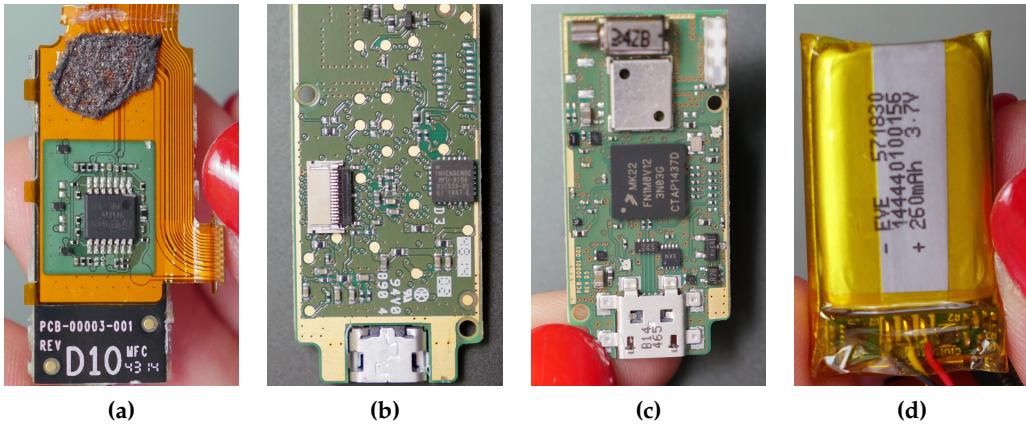


Figure 4.2:

- (a) The proprietary EMG sensors, source: <https://learn.adafruit.com/assets/29293>
- (b) Back view of PCB with MPU 9150, source: <https://learn.adafruit.com/assets/29292>
- (c) Front view of PCB with MK22FN1M0 and nRF51822, source: <https://learn.adafruit.com/assets/29291>
- (d) 260 mAh battery, source: <https://learn.adafruit.com/assets/30337>

EMG data from the eight channels of the Myo Armband was gathered whilst doing a fist gesture to discover the characteristics of the EMG signals captured by the armband, see figure 4.5 on the next page and the attached video named “Myo Armband EMG features testing”.

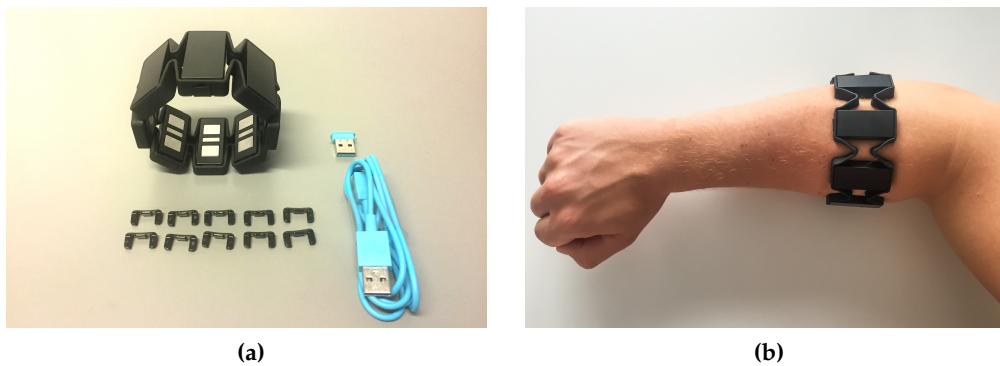


Figure 4.3: (a) Myo Armband, micro USB cable, bluetooth 4.0 dongle and sizing clips (b) Myo Armband placed on a forearm.

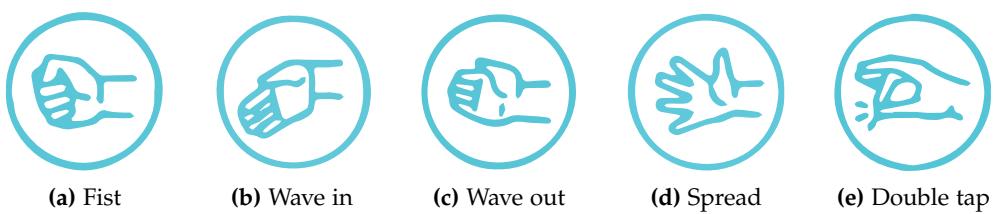


Figure 4.4: Gestures from the accompanying software “Myo Keyboard Mapper”.

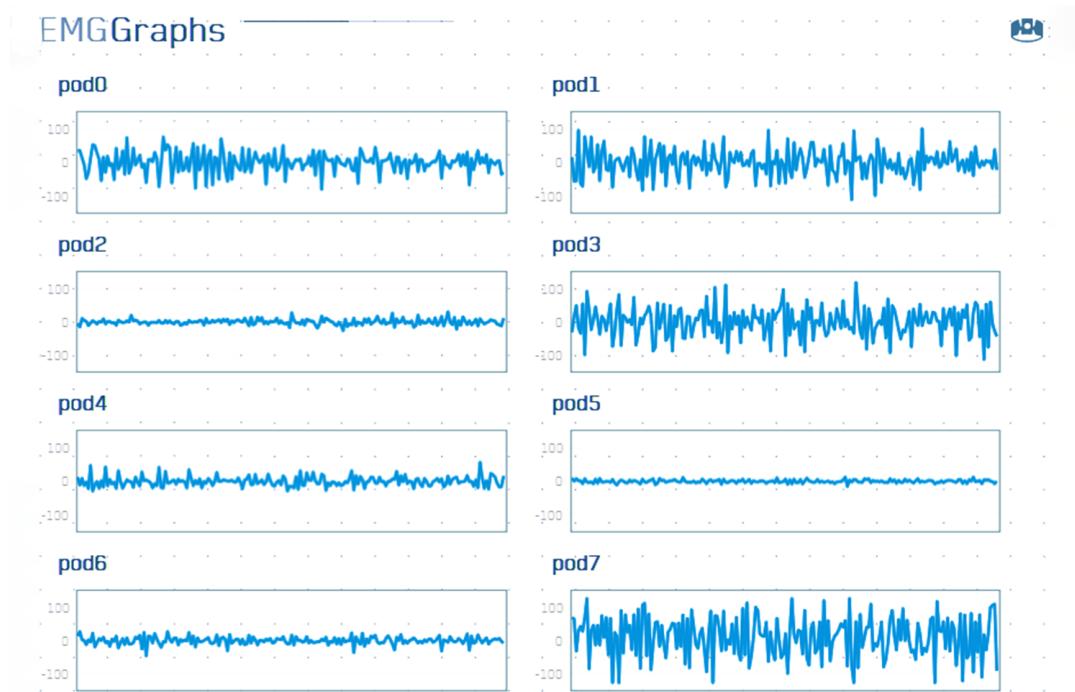


Figure 4.5: EMG recordings of the eight channels while doing a fist gesture. The image is a screenshot of the Myo diagnostic tools on diagnostics.myo.com/, and a video of all 5 gestures can be found on the attached video named “Myo Armband EMG features testing”.

4.2 The CrustCrawler

The CrustCrawler, seen in the figure 4.6, is a 3R manipulator, meaning it consists of three revolute joints, which gives the manipulator five degrees of freedom. The manipulator has a total of five Dynamixel servos from the MX series. Three different models are used: MX-28R, MX-64R and MX-106R.

The first joint, called the base, is one of the MX-64R models. The second joint is the only MX-106R model. The third joint is actuated by an MX-64R servo, and the gripper consists of the two MX-28R servos. [40, 41, 42].

As seen in table 4.1 on the next page all the motors are able to turn 360 degrees, though this is not mechanically possible due to the construction of the **CrustCrawler**. The motors differ in dimension, weight and amount of torque they are able to produce with the MX-28 being the smallest and least torque producing motor. The MX-106 is the biggest servo and has the highest torque of the three. The weight is ranging from 75 - 153 g and torque ranges from 2,3 - 10 N.m. The servos have a resolution of 4096 points pr. revolution. The links connecting the joints are hollow metal beams from the Pro Series arm kits.

Controlling the CrustCrawler

The Dynamixel servos can be controlled directly from a PC, but since the PC have a lot of other tasks it needs to attend, due to the functioning of the operating system and other installed applications, this might result in delays compared to a microcontroller which is another way of controlling the manipulator, a small analysis of an applicable microcontroller can be seen in appendix A. A USB to RS-485 adaptor is also present allowing a PC with a USB port to be connected directly to the servos, through which controlling and pinging the motors of the manipulator is possible. Power is supplied to the CrustCrawler via the power hub (figure A.1c on page 66) which has 6 Molex SPOX 50-37-5043 making it able to deliver the 10-14,8 volts all the components need but also the RS-485 signal [44].

Through analysis of the Myo Armband, all of its components and sensors were discovered which included 8 EMG sensors, a 3-axis gyroscope, 3-axis accelerometer and 3-axis compass. These sensors will be taken into consideration when developing a solution. It was also found that the Myo Armband is able to deduce five different feature extractions. Furthermore, investigation revealed that data communication between the Myo Armband and the computer is via Bluetooth Smart Wireless Technology. The components and capabilities of the CrustCrawler were studied as well, which revealed that it is a 3R manipulator actuated by five Dynamixel servos with torque yielding capacities ranging from 2,3 - 10 Nm. These maximum torques have to be taken into account when designing an application for hemiparetic patients. In addition, an explanation of the interface to the Dynamixel servos showed that serial communication to the servos runs on a RS-485 standard and that they need 10-14,8 volts to operate.



Figure 4.6: CAD drawing of the CrustCrawler.

Table 4.1: The different servo models and their statistics. The data is from: [43]

Model	MX-28R	MX-64R	MX-106R
Stall Torque at Max Voltage	3.1 N.m or 31.6 kg-cm	7.3 N.m or 74.4 kg-cm	10.0 N.m or 101 kg-cm
Operating Voltage	10 ~ 14.8V (Rec. Voltage: 12V)	10 ~ 14.8V (Rec. Voltage: 12V)	10 ~ 14.8V (Rec. Voltage: 12V)
Speed (RPM)	97	78	55
Resolution	4096 points pr. rev. or 0,088°	4096 points pr. rev. or 0,088°	4096 points pr. rev. or 0,088°
Operating angle	360°	360°	360°



CONTROL THEORY

This chapter will look into basic control theory as well as widely used control methods and how they function to get an understanding of how a controller can be implemented on a robotic manipulator.

5.1 Open-loop systems

A control system controls and/or regulates another system also known as a plant. Control systems can either be open- or closed-loop. An open-loop system, shown in figure 5.1, is referring to a system without regulation of the plant.

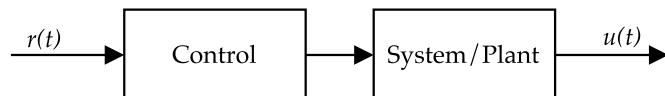


Figure 5.1: Open-loop system with reference signal $r(t)$ and the output $u(t)$.

Many products use an open loop-system, for example a toaster. A toaster does not scan the toast to see if it is burned or not. An open-loop system can be useful because it is a cheap and easy solution, and it can be applied where the accuracy of the output is not essential. [45]

5.2 Closed-loop systems

Closed-loop systems differ from the open-loop system because the plant gives a feedback to the system controller. The feedback involves a measuring of the current value of the output from which the reference signal is subtracted. This will evaluate to an error which is fed to the controller. It is now up to the controller to make this error go to zero. As explained in figure 5.2 on the next page, the error is calculated by

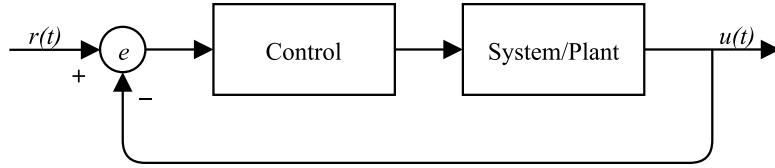


Figure 5.2: Closed-loop system with reference signal $r(t)$ and the output $u(t)$.

$$e(t) = r(t) - u(t) \quad (5.1)$$

where $e(t)$ is the error, $r(t)$ is the reference and $u(t)$ is the output. The closed-loop system is practical for the consumer since it will handle the errors that might occur. Closed-loop systems often require an advanced controller and most importantly sensors to give feedback, and this can result in more expensive solutions, compared to an open-loop.

If a manipulator has to be able to work in an environment alongside or in collaboration with humans, it should have a closed-loop system to ensure the safety of the surroundings. A closed-loop system can help keep the manipulator on the right trajectory, e.g. if a force is pushing the manipulator down, the control system can calculate how much torque has to be applied to the motors in order to stay in position. [45] The control system is evaluated by a number of parameters including: **a)** Overshoot, which is the offset from the goal point in percent of total travel distance. **b)** Rise time, which is how long it takes to move from a specified point in the beginning of the movement to a specified point near the end of the movement. **c)** Settling time, which is how long it takes to stay in a specified margin. **d)** Steady state error, which is the persistent error value when the system has settled.

5.3 PID controller

A Proportional-Integral-Derivative (PID) controller is a part of closed-loop system. It is one of the most commonly used controllers implemented with the purpose of controlling systems e.g. movements of a manipulator in which case the PID controller works by continuously calculating an error value $e(t)$ as the difference between the desired point and the measured point of the manipulator. [46] The advantage of using a PID controller is the ease of adjustment in the different elements, P, I and D respectively, which can be done separately [47]. However the PID gains must be tuned properly to have optimal performance [48]. The three elements of the PID controller each have different functions which can be seen in table 5.1 on the following page and are defined as follows:

- P -** In the proportional controller the error signal is multiplied by a constant, meaning that when a manipulator is moving towards a goal-point the error will become smaller, and the power fed to the motors will decrease proportionally as well. The power fed to the motors will eventually be so low the motor will stop moving and thereby stop going any closer to the set point raising a steady-state error. The Proportional controller operates on the present error.
- I -** The integral controller is used to avoid the steady-state error by providing more power (increasing the torque) to the motors if the error persist, though this usually decreases stability or eliminates it altogether. Described in another way; when the integral increases the manipulator will have an overshoot on its path to the goal-point. The integral controller operates on the accumulated error.
- D -** The differential controller is calculating the rate of change in the error by differentiating. The D-controller is often used to avoid a big accumulated error from the I-controller that otherwise would have caused an overshoot. If the system moves too quickly towards the set point, the D-controller will slow down the system, or if the system moves in the wrong direction e.g. when an overshoot is happening, it will increase the power in the correct direction. The derivative controller operates on the direction of the error. [49]

Table 5.1: Showing the effects of independent tuning of - K_p: Proportional, K_i: Integral and K_d: Derivative. [47]

Closed-loop response	Rise time	Overshoot	Settling time	Steady-state error
Increasing K _p	Decrease	Increase	Small change	Decrease
Increasing K _i	Decrease	Increase	Increase	Eliminate
Increasing K _d	Small change	Decrease	Decrease	No change

Figure 5.3 on the next page shows a block diagram of a PID controller, where the P, I and D elements are illustrated. The system gets a set-point, R(t) also known as a reference, which then is put into the transfer functions. Each element has its own computation, but as seen on the figure all transfer functions will be summed up as:

$$U(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (5.2)$$

Thus concluding the closed-loop feedback system of a PID controller. [47]

General control system theory, closed-loop and open-loop, was investigated in order to acquire knowledge on how to implement a controller on a manipulator. It was discovered that a closed-loop system is needed in order to keep the manipulator on the right trajectory. Furthermore, since PID controllers are commonly used a thorough analysis of it, with the focus on manipulators, was conducted. In the analysis it was found that this controller could be used for optimisation of the individual parts of a control system. The reason for doing a full investigation of the PID, is acquiring the ability to choose the right combination of the three controllers and tune them correctly.

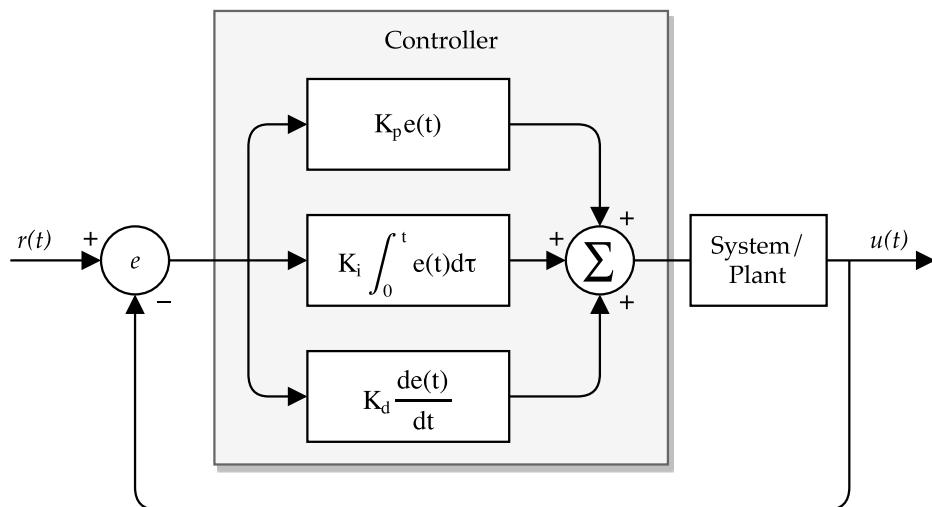


Figure 5.3: A block diagram of a PID controller in a feedback loop.

Part III

Development

CONCEPTUALISATION

This chapter describes and conceptualises the vision for this project.

The vision for this project is a platform that helps hemiparetic patients with rehabilitation through the use of a manipulator interacting with the patients. The platform has four different rehabilitation modes: Passive, assistive, active and resistance, see section 2.2 for a description of these. The idea of this platform is to reach the biggest possible target group i.e. all patients with some degree of hemiparesis. This would be accomplished by development of multiple applications, in the form of games, which would allow a specialist to tailor a specific rehabilitation programme for each patient. If the patient's degree of hemiparesis allows it, the games should involve ADL specific movements. Some patients suffering from severe cases of hemiparesis have very low mobility in the arm, which will restrict the amount of games they are able to take part in and consequently, the patients will have to start with only moving the hand.

Moreover, the system will include a data collection of e.g. EMG recordings from each patient, which will be sent to a server based cloud and stored there. This will allow the system to be used at home while the specialist can follow the patient's progress. The data collection also makes it possible to give a better overview of the strengths and weaknesses of each game, which means they can be reshaped by feedback from users and rehabilitation specialists.

The ultimate goal of this platform is to introduce a product that will induce a motivating, rewarding and fun rehabilitation environment while reducing the need for physical assistance from specialists compared to conventional rehabilitation. Such a product should improve results of rehabilitation by motivating each patient to increase effort in their individual rehabilitation courses, while relieving the workload per patient for the personnel involved in the rehabilitation. It is important to note, the conceptualisation shall be seen as a description of a hypothetical product that will work as a framework of the development goal of this project.

The vision is a motivating rehabilitation platform involving ADL movement games, and data synchronization to a cloud server.

DELIMITATIONS

Following the conceptualisation, this chapter will set the boundaries of the upcoming development of prototypes.

Overview

The general functionality, user characteristics and design constraints of the prototypes will be described to provide a base for the specific requirements of the prototypes. The specific requirements consisting of functional- and performance requirements as well as requirements for external interfaces will be listed in section 11.1 after the prototype has been explained.

Scope

Throughout the development part of this report, a prototype based on the CrustCrawler and the Myo armband will be build in order to demonstrate parts of the robotic system described in chapter 6. The prototype details will be described in chapter 11.

Product functions

The CrustCrawler will execute a sequence of movements initiated by sensor data captured by the Myo Armband. This means that execution of the program on the CrustCrawler will rely on the movements performed by the user of the armband.

User characteristics

The intended users of the prototype developed in this report are hemiparetic patients where the degree of impairment does not prohibit them from generating EMG signals in the impaired arm that can be registered as features by the Myo Armband.

Hardware constraints

The hardware used for development of the prototype is limited to the CrustCrawler and the Myo Armband. The Myo Armband can only record sEMG and its placement is limited to the forearm. However, since it shares many of the benefits of non-invasive electrodes described in section 3.2 i.e. easy handling, comfort and satisfactory EMG recording from limb muscles, it is eligible for use in the development of a prototype. The other sensors of the Myo Armband mentioned in section

4.1 may also be incorporated in the prototype development. Use of the Crust Crawler induces some restrictions on the reach and maximum motor torques as described in section 4.2 as well as strict restrictions of dexterous workspace due to the configuration and the number of joints.

Based on the results of the problem analysis, the vision provided by the conceptualisation and the delimitations thereof, a final problem formulation is reached.

FINAL PROBLEM FORMULATION

How is it possible to make use of the sensors in the Myo Armband and the capabilities of the CrustCrawler in order to develop a prototype that demonstrates parts of the system described in chapter 6?

INITIAL PROTOTYPE SETUPS

This chapter explains how the hardware and software need to be setup before it can be used for prototyping. The chapter also explains the forward and inverse kinematics of the CrustCrawler so trajectories can be calculated. Furthermore, it explains how rehabilitation data can be collected and synced with cloud based software.

9.1 CrustCrawler setup

Motor IDs

Before the CrustCrawler can be used it needs some initial setups. First of all one must assure that the Dynamixel motors have different ID-numbers between 1 and 253 which is the allowed interval. The motors of the CrustCrawler manipulator has been assigned consecutive IDs [1,2,...,5] making it more convenient to access them individually in the code. If all Dynamixel motors need to be accessed at once, a *write command* can be send to ID #254 on which all motors respond. A *read all motors command* is not possible since all motors will try to respond simultaneously.

Joint limits

Since the body of the manipulator is hindering the servos from turning infinitely, software joint limits are needed. The joint angle limits are stored in the Dynamixel motors' internal Read Only Memory (ROM), so they will be remembered even if the motors are turned off. From the analysis of the Dynamixel motors, section 4.2, it is known that the resolution of the motors is 4096 points pr. revolution. The angles are measured by the magnetic encoder inside the motors. The joint limits are set as a maximum counter clockwise and a maximum clockwise direction, being lower numbers starting from 0, and higher numbers going up to 4096 respectively. If the motor is asked to move beyond the limit, the motor will not respond. The reason for these joint limits is to protect the motor from overload when trying to move beyond the mechanical stop.

Home position

The home position of the motors have been defined in a way where the manipulator will be standing straight up as shown in figure 9.1. At this position all the motors should be at their midpoint between the software joint limits.

Without software and hardware limits the motors can do one full rotation between 0 and 4096 points when in *joint mode*. The starting point 0, is fixed and cannot be offset and in *joint mode* the motors cannot move through this point. This could cause problems as the motor joint value would have to jump from 0 to 4096 making the motor spin a full revolution the other way to avoid moving through 0. Therefore the home position motor angle for this CrustCrawler is set to 2048 which is the midpoint between 0 and 4096, putting the limits 0 and 4096 outside reach. Consequently the CrustCrawler should be assembled with the motors firstly set to their midpoint, and then the robot can be assembled standing in its home position as in figure 9.1

Communication

In order to communicate with the motors a USB to RS485 converter is used together with a power hub to supply power to the motors. The computer interprets the USB as a serial port with a serial interface. The bit rate of the communication is at 1Mbps.

In order to send commands over the serial connection, the programming language Python was chosen, but other languages could also have been used such as C++.

Python code and classes

The python code, written to control the CrustCrawler, consists of several libraries, where the main library is the *ccctrl* library (short for CrustCrawler control library). The *ccctrl* library consists of multiple classes that all work together to form a controller. This controller has been created through several iterations over two versions. The two versions are described later in chapter 10.

9.2 Myo Armband setup

Thalmic Labs, the maker of the Myo Armband, has provided a Software Development Kit (SDK) for interfacing with the Myo Armband. The SDK is a library that takes care of the connection between the armband and the Bluetooth dongle, and essentially all the data carried over the Bluetooth connection. The Myo SDK library is accessed through a plain C API (Application Programming Interface), where all the functionality of the library is exposed for other applications.

The software applications of this project are not interacting directly with the C API but through a Python binding. A language binding makes it possible to use

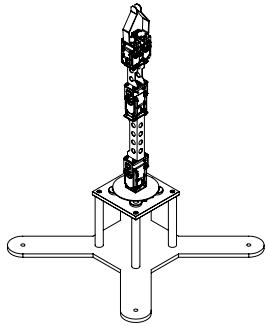
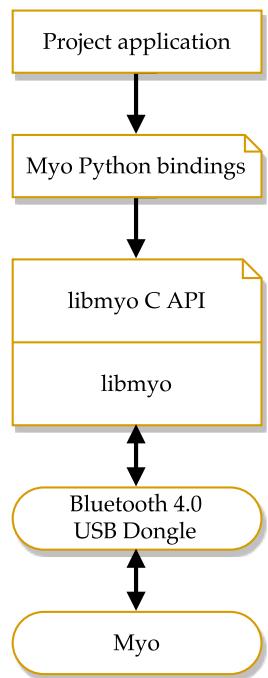


Figure 9.1: Home position.

the Myo SDK in for example a Python application, even though the Myo SDK is written in C. The structure of the Python application, API and SDK library working together can be seen on the margin figure 9.2.

Classes of the Myo SDK library

Figure 9.3 on the following page shows the classes described below. When using the library, an object from the class *Hub* must be created. Only one hub object can be running at a time. The purpose of the hub is to connect to Myo Armband and handle all the data sent between the Myo Armband and the computer. The hub uses an object of the class *DeviceListener*. Objects of this class handles callbacks every time the hub registers an event such as a hand gesture or orientation. The callbacks can for example execute commands for the CrustCrawler at a certain event. A special type of device listener called *Feed* takes all the data from the Myo Armband and makes the most recent data available on demand. This is especially useful when older queued up data is not of importance, and only the live data feed is wanted. The *Feed* inherits from the *DeviceListener* class and when a callback occurs it saves the most recent data as its attributes.



9.3 Kinematics

Forward Kinematics

The kinematics are calculated for the purpose of controlling the CrustCrawler in cartesian space. The way forward kinematics are calculated is by knowing the Denavit–Hartenberg parameters (DH parameters) for the CrustCrawler, which can be found in table 9.1. Given angles for each of the motors in the manipulator, the orientation and the placement in Cartesian space can be calculated.

Table 9.1: Denavit hartenberg parameters for the CrustCrawler are in mm and radians.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$\pi/2$	0	244,16	$\theta_1 + \pi$
2	0	217,16	0	$\theta_2 + \pi/2$
3	0	269,80	0	θ_3

By placing the DH parameters in the standard equation 9.1, for each of the links, and multiplying the acquired matrices of each link with each other will give an equation where the unknown variables, in this case θ_1 θ_2 θ_3 can be inserted in 9.2 where x = equation 9.3, y = equation 9.4 and z = equation 9.5 (d_0 , a_1 and a_2 are

Figure 9.2: Myo SDK library.

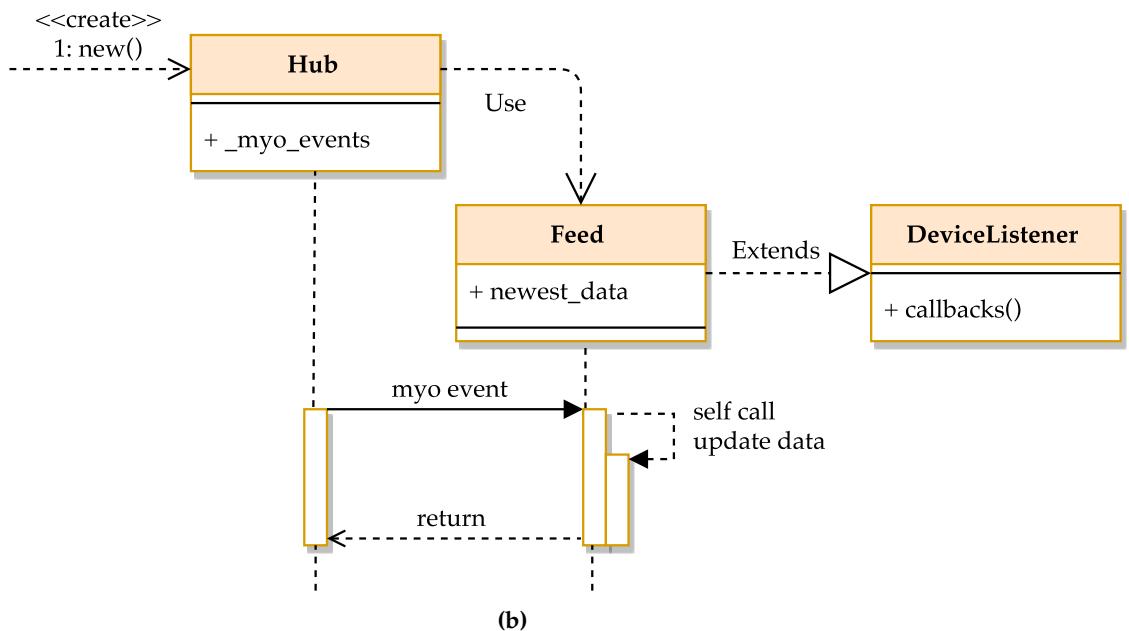
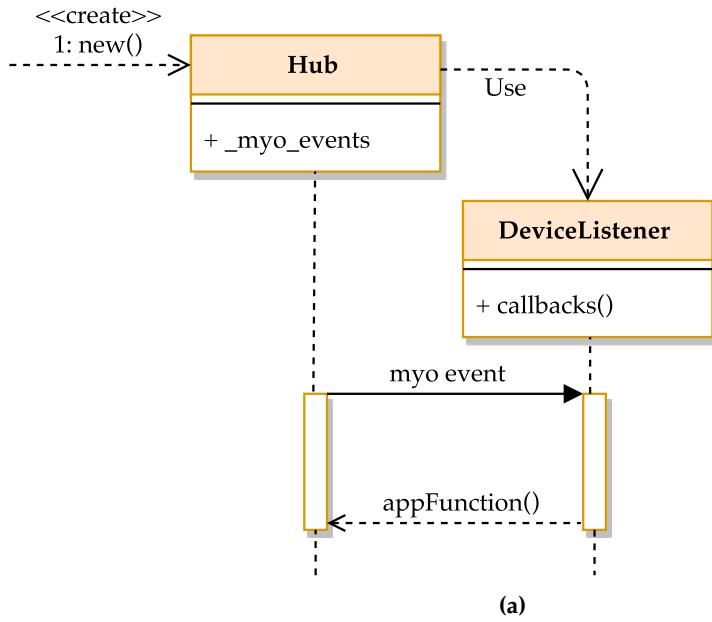


Figure 9.3: The UML diagram (a) shows the use of the class **DeviceListener**. The UML diagram (b) shows the use of the class **Feed**.

the d and a values from table 9.1). These equations will be used in the Python code to calculate the forward kinematics used when controlling the CrustCrawler.

$${}_{i-1}^i T = \begin{bmatrix} \cos \theta_i & -\sin \theta_i & 0 & a_{i-1} \\ \sin \theta_i \cos \alpha_{i-1} & \cos \theta_i \cos \alpha_{i-1} & -\sin \alpha_{i-1} & -\sin \alpha_{i-1} d_i \\ \sin \theta_i \sin \alpha_{i-1} & \cos \theta_i \sin \alpha_{i-1} & \cos \alpha_{i-1} & \cos \alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9.1)$$

$$T = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (9.2)$$

$$x = a_2 \cdot (\cos(\theta_0) \cdot \sin(\theta_2) \cdot \sin(\theta_1 + \pi/2) - \cos(\theta_0) \cdot \cos(\theta_2) \cdot \cos(\theta_1 + \pi/2)) - a_1 \cdot \cos(\theta_0) \cdot \cos(\theta_1 + \pi/2) \quad (9.3)$$

$$y = -a_2 \cdot (\cos(\theta_2) \cdot \cos(\theta_1 + \pi/2) \cdot \sin(\theta_0) - \sin(\theta_0) \cdot \sin(\theta_2) \cdot \sin(\theta_1 + \pi/2)) - a_1 \cdot \cos(\theta_1 + \pi/2) \cdot \sin(\theta_0) \quad (9.4)$$

$$z = d_+ a_2 \cdot (\cos(\theta_2) \cdot \sin(\theta_1 + \pi/2) + \cos(\theta_1 + \pi/2) \cdot \sin(\theta_2)) + a_1 \cdot \sin(\theta_1 + \pi/2) \quad (9.5)$$

Inverse Kinematics

Inverse kinematics of a manipulator determines the joint angles of a certain point in space.

Calculation of the inverse kinematics of the CrustCrawler starts of by determining the vector \vec{P} pointing from origin to the end-effector, which has the coordinates: x_0, y_0, z_0 , as seen in figure 9.4 on the next page. The length $||\vec{P}||$ is then calculated with the Pythagorean theorem as seen in equation 9.7:

$$\vec{P} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \quad (9.6)$$

$$||\vec{P}|| = \sqrt{x_0^2 + y_0^2 + z_0^2} \quad (9.7)$$

With the length of the vector \vec{P} the angle θ_2 can be calculated with the law of cosines as seen in equation 9.8,

$$\theta_2 = 180 - \cos^{-1} \left(\frac{d_1^2 + d_2^2 - \|\vec{P}\|^2}{2 \cdot d_1 \cdot d_2} \right) \quad (9.8)$$

where d_1 and d_2 are the link lengths. Note the subtraction from 180, the reason for this is that the angle calculated is actually angle III and θ_2 is measured from the desired zero-position-angle illustrated on figure 9.4 as the dashed line continuing from d_1 .

θ_1 is calculated as the angle between the z -axis and the link d_1 (illustrated on the figure). θ_1 can be calculated as 90 degrees minus angle I and II. This can be seen in equation 9.9,

$$\theta_1 = 90 - \tan^{-1} \left(\frac{\vec{z}}{\sqrt{\vec{x}^2 + \vec{y}^2}} \right) - \cos^{-1} \left(\frac{d_1^2 + \|\vec{P}\|^2 - d_2^2}{2 \cdot d_1 \cdot \|\vec{P}\|} \right) \quad (9.9)$$

where angle I is calculated as the inverse tangent function, and angle II is calculated with the law of cosines.

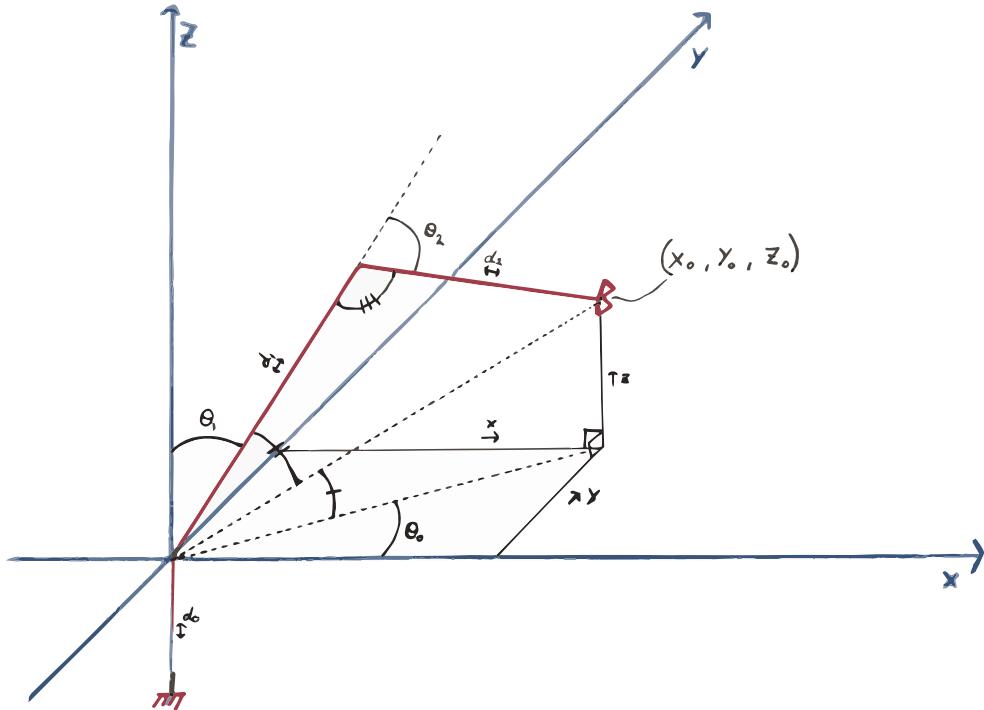


Figure 9.4: Coordinate system containing the CrustCrawler.

Calculation of θ_0 is done with the inverse tangent to y over x in 9.10.

$$\theta_0 = \tan^{-1} \left(\frac{y}{x} \right) \quad (9.10)$$

This concludes the calculations of the inverse kinematics and with a given x, y and z coordinate the 3 angles θ_1, θ_2 and θ_3 can be calculated using formulae 9.8, 9.9, 9.10.

9.4 Data collection

All prototypes proposed in this project will collect data in the cloud service Google Drive. However if any of these prototypes are to be used in the rehabilitation process another service would be required for the data storage. This is due to performance restrictions and the fact that personal information must be handled in a way where the user does not feel any discomfort by sharing the information. Therefore, the data need to be secured with encryption only allowing certain authorities to have access to it. The idea of storing the data of patients is to allow the therapists to see the progress of each individual, as well as being able to get a statistical overview of all patients. Furthermore, it will serve the purpose of giving specialists quantitative data that can be used to weigh the effectiveness of a game, or specifically a motion.

The data is collected from the Myo Armband which, as mentioned in section 4.1, has built-in EMG sensors which can output EMG data. In the prototype this will be stored in a CSV (comma separated values) file, which then will be synced with Google Drive. Through a script written in Google's own scripting language a spreadsheet will collect the data from the file and continuously graph the incoming data. A flowchart of this process can be seen in figure 9.5 on the following page.

This chapter concluded that the Dynamixel motors of the CrustCrawler should be given consecutive ID numbers and they should be set at a home position of 2048 while the CrustCrawler is assembled in an upright position. An explanation of the relation between fundamental libraries and classes within was given. Essentials needed to develop prototypes, e.g. kinematics and the data collection, were calculated and programmed.

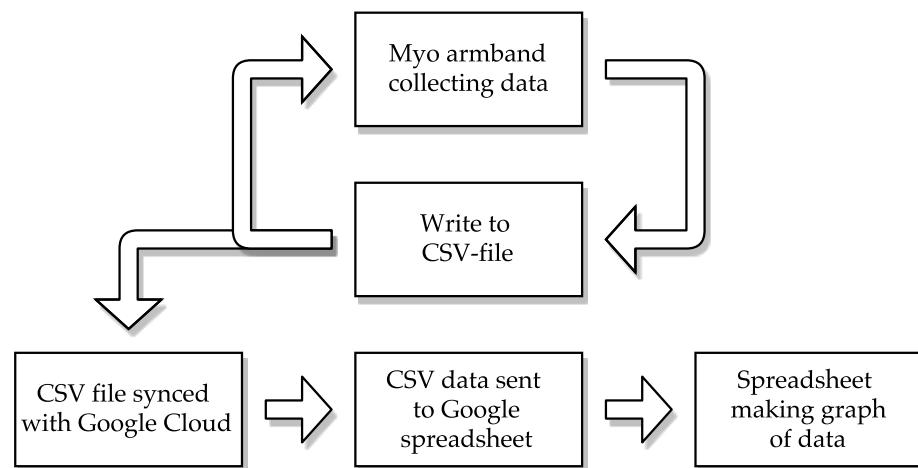


Figure 9.5: Flowchart of how the EMG data is send to the cloud.

CONTROL SYSTEMS

This chapter examines the possibility of developing a controller following the general control theory of chapter 5. The developed control system is important when designing the prototype of the concept since controlled movements will be required. Validation of the developed control systems will be carried out through testing.

10.1 First CrustCrawler control system

The first attempt at a controller for the CrustCrawler, implements the kinematics from section 9.3 and control theory from chapter 5. The first controller has the ability to perform both Point To Point (PTP) and linear movements.

The controller consists of two classes namely the *dynamixel_mx* and *robot* class, see figure 10.1 on the next page to get an overview of the classes. The *dynamixel_mx* class is controlling the low-level serial communication with the dynamixel motors such as reading and writing data. The *robot* class inherits from the *dynamixel_mx* class, and is controlling movements in Cartesian space i.e. PTP and linear movements. The *robot* class contains the forward and inverse kinematics which is needed to perform the PTP and linear movements.

PTP controller algorithm

The PTP algorithm is quite simple since it only performs inverse kinematics of the goal point once and sends the joint angles to the motors. The motors of the CrustCrawler will then go to those angles with a desired joint velocity. The PTP algorithm can be seen on figure 10.2 on page 43.

Linear controller algorithm

Creating a linear movement in Cartesian space is much more demanding when it comes to both complexity and computational power. The flowchart-figure 10.3 on page 45 explains the first attempt at making a controller for linear movements.

Looking at the flowchart, the algorithm is divided into three parts. First, the algorithm needs two inputs which is a goal point, marking the desired goal of the movement, and an end-effector velocity. Secondly, a number of initial calculations,

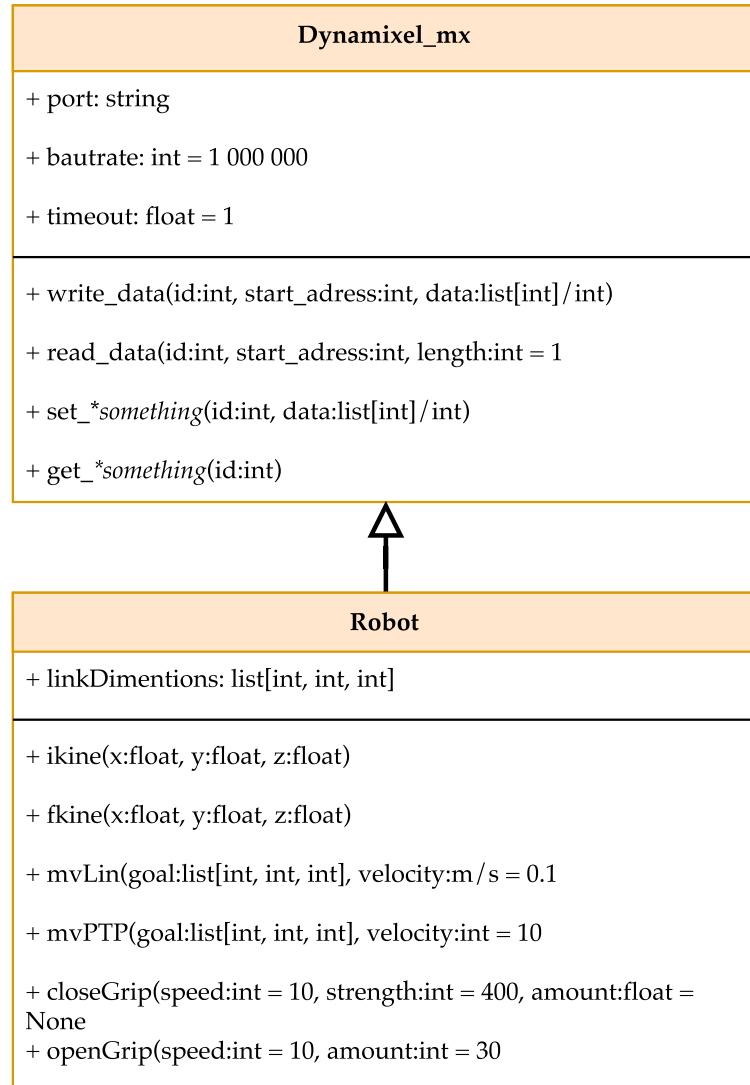


Figure 10.1: Class diagram of the class that controls the Dynamixel motors (the dynamixel_mx class) and the CrustCrawler movements in cartesian space (the Robot class).

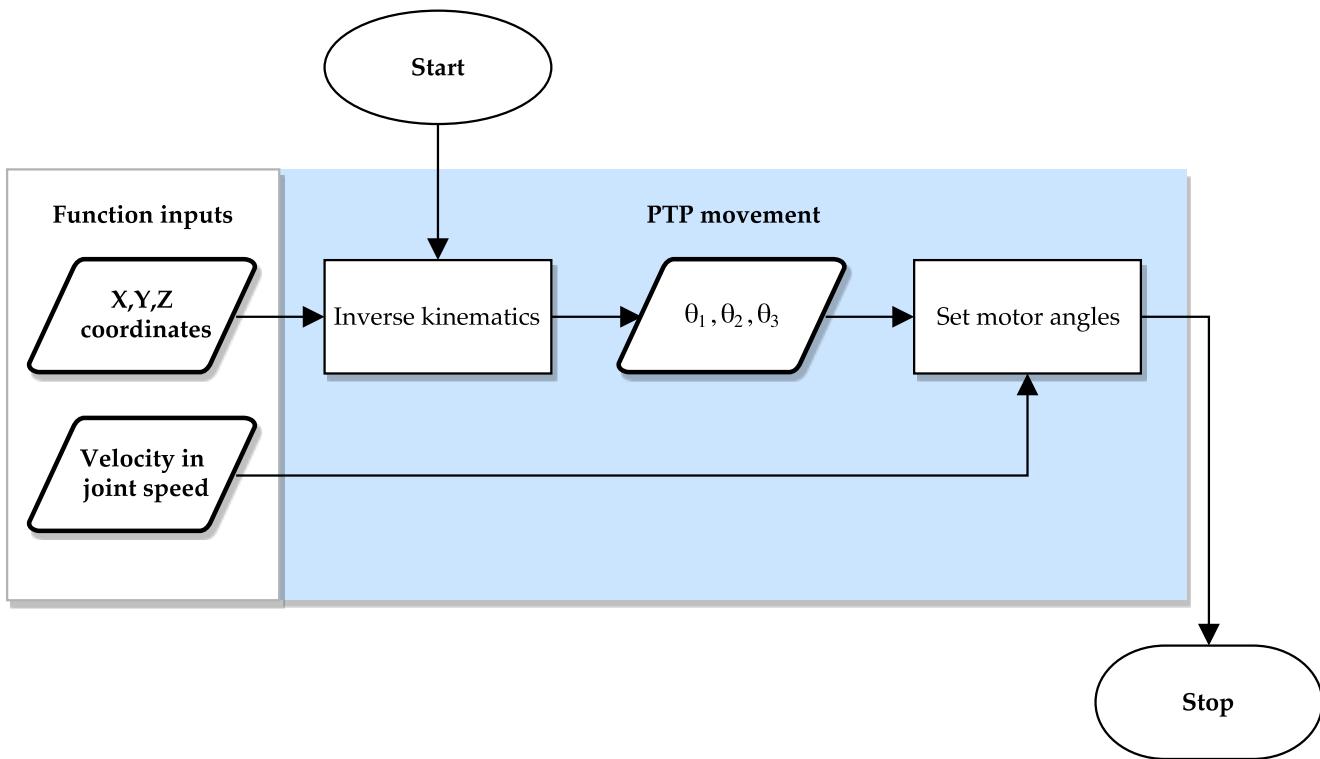


Figure 10.2: A flowchart of how the CrustCrawler controller performs the Point To Point (PTP) movement.

showcased in the pink area of the flowchart, are made starting with a calculation of forward kinematics to reveal the initial point in space of the CrustCrawler's end-effector, referred to as the start point. With a known start- and goal point, it is possible to calculate the vector from start point to goal point, displayed as the trajectory vector in the flowchart. The length of the trajectory vector is calculated, as this value is a prerequisite to the calculation of execution time i.e. time available for the linear movement given the velocity that was inputted.

Next, the execution of linear movement enters a loop where points along the trajectory vector are calculated based on the fraction of execution time that has passed. This loop is encapsulated in the blue block of the flowchart. Specifically, the new trajectory points result from a multiplication of the trajectory vector and the time fraction which is added to the start point.

$$P_{Traj} = P_{Start} + \frac{t}{t_{Available}} \cdot \vec{u}_{Traj}$$

Once the cartesian coordinates of the next trajectory point are obtained, the corresponding angles to that point are deduced through inverse kinematics. This

process runs until the value of the time fraction passes one, since this marks the end of the available execution time.

As showed in the flowchart, the angles of a given trajectory point is delivered to the CrustCrawler after each calculation, and then it is up to the embedded PID controllers in the Dynamixel motors to make sure that these angles are achieved. The angular velocities the motors use are controlled using a separate PID controller class written in python, see section 5.3 for reference on PID.

When the angles of a trajectory point has been set, the velocity PID controller calculates a positional error for each servo ($\theta_{desired} - \theta_{actual}$) after which it will compensate for a possible error. The error occurs when the motors are expected to be at a certain angle at a certain time ($\theta_{desired}$), but have yet to reach that angle (θ_{actual}). The error will be passed to the PID controller where operations on the error will result in a boost in velocity. Specifically, the PID controller processes the errors with the following parameters which were determined by trial and error: $K_p = 800$, $K_i = 100$ and $K_d = 10$. By inputting these parameters into the equation,

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (10.1)$$

which is the same equation as 5.2 from section 5.3, the following equation for the response of the motors is obtained:

$$u(t) = 800e(t) + 100 \int_0^t e(\tau) d\tau + 10 \frac{de(t)}{dt} \quad (10.2)$$

Testing and evaluation

One significant flaw of the above mentioned control algorithm for linear movements is the jagged motion it generates due to the vast amount of intermediate points the servo motors are commanded to travel to. Once the servo motors obtain the desired angle of each intermediate point on the trajectory vector, movement will pause shortly before continuing to the next desired value thereby creating this inconsistent and jagged pattern of movement. The pause in motion happens due to the fact that the desired angle positions have no corresponding velocity information to follow. Without this, the embedded PID controllers will simply make the Dynamixel servos come to a complete stop. The velocity PID controller will make sure that the motor will get there rapidly in order to make the error go towards zero. Basically, this linear controller has two PID controllers working on the same task i.e. measuring the angle and controlling the speed.

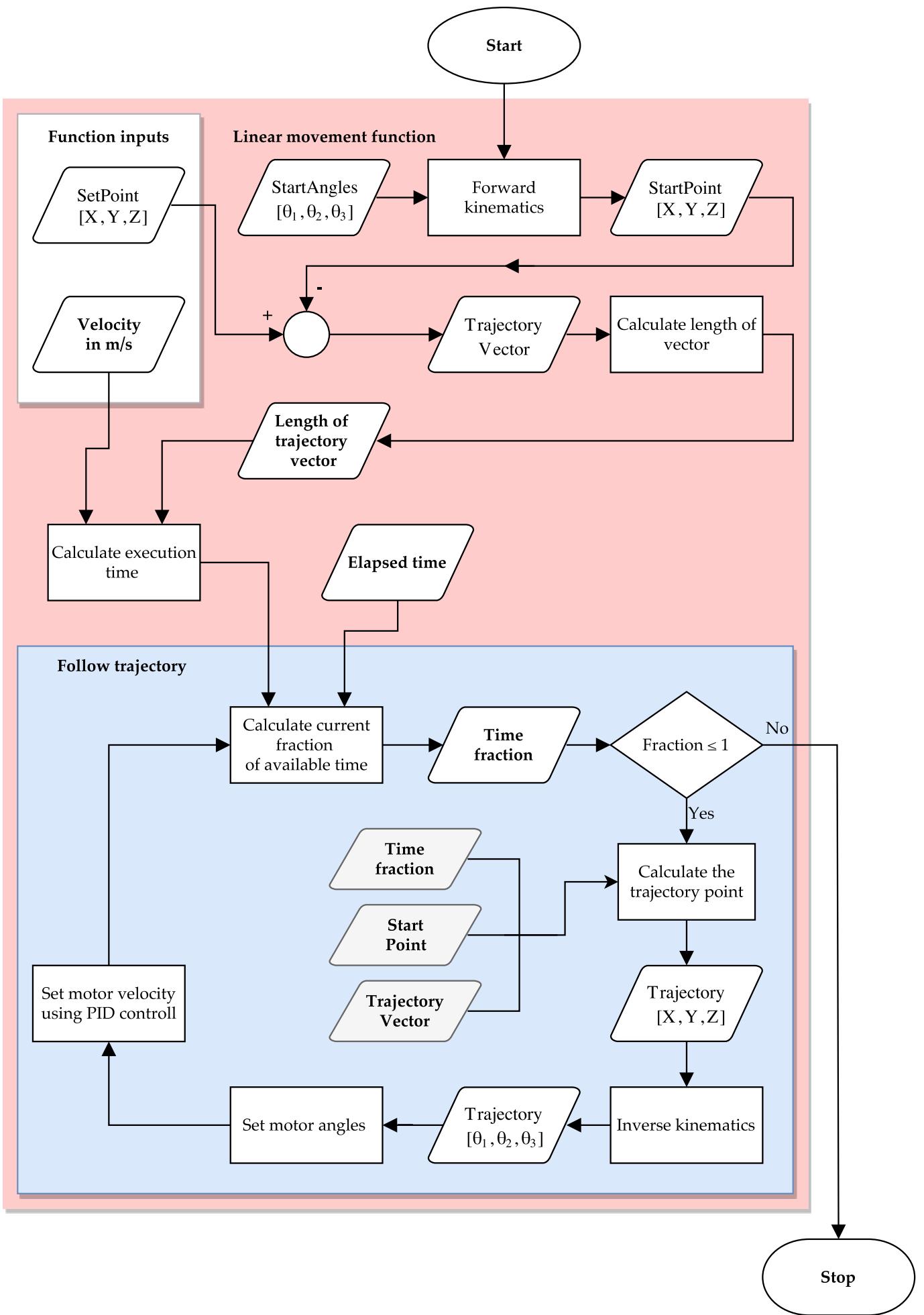
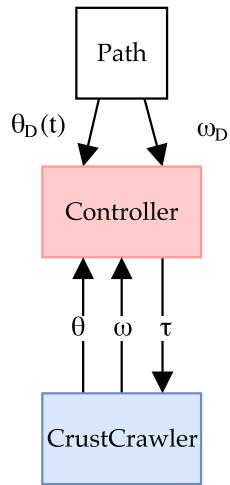


Figure 10.3: Flowchart of how the CrustCrawler controller performs a linear movement.

10.2 Second CrustCrawler control system

A new control system was developed as an attempt to correct the flaws of the first one. The movement types of the CrustCrawler remain the same after implementation of this new control system, i.e. it should still be able to perform PTP as well as linear movements.



This controller was built completely from scratch due to the flaws of the first one being at a fundamental level which could not be corrected using the same system.

Firstly, the new controller turns off the embedded PID controllers of the Dynamixel motors. The angle and joint velocities are now solely taken care of by the new controller. Instead of sending goal angles to the motors, the new controller turns the motors by sending torque values. The controller then reads the angles and the angular velocities of all the motors as feedback allowing the controller to constantly adjust the torques. The calculation of desired angles and angular velocities is carried out by a trajectory planner that is able to pass this data to the controller. The overall structure of the new controller can be seen in figure 10.4

Trajectory planner

The trajectory planner and the controller are separate entities, which makes future expansions to system possible. One can imagine having many different trajectory planners instead of just the PTP and linear, e.g. circular and other complex curves and paths, but this project only focuses on the execution of linear movements. As seen on the small diagram on figure 10.4, the control of a linear movement starts with the trajectory planner which outputs angular positions and velocities for the CrustCrawler's three joints (the last two servos control the gripper). This also marks the first difference between the first and second version of the control system since the first system did not calculate any desired velocities on the path towards the next point on the trajectory.

The initial setup, before the individual desired trajectory points can be calculated, is almost the same as the original trajectory planner for linear movements seen in figure 10.3 from the previous section.

A trajectory vector \vec{u} in Cartesian space is created between the start point P_{Start} and the goal point P_{Goal} by subtracting the goal point from the start point. $\vec{u} = P_{Goal} - P_{Start}$.

The length of the vector is calculated with $\|\vec{u}\| = \sqrt{u_x^2 + u_y^2 + u_z^2}$, and with the length of the vector and the desired end-effector velocity a total move time can be calculated as $t_{Total} = \frac{\|\vec{u}\|}{v}$ where v is the end-effector velocity.

Figure 10.4:
New Controller.

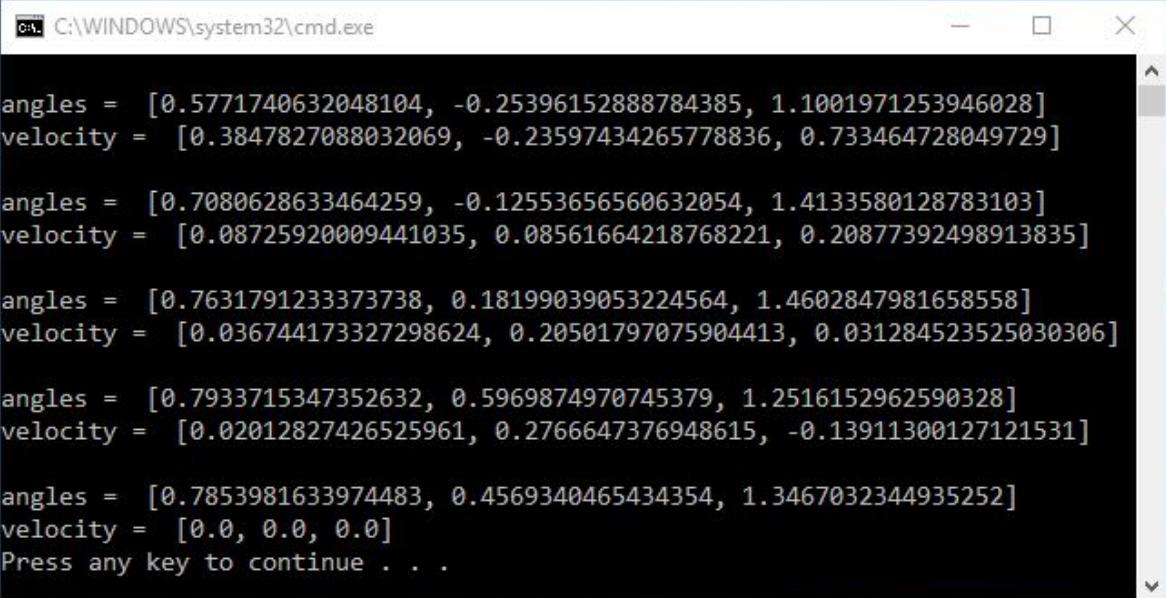
Moving along the vector \vec{u} is now a matter of adding fractions of the vector to the start point according to the following equation:

$$P_{Traj}(t) = P_{Start} + \frac{t}{t_{Total}} \cdot \vec{u}$$

Afterwards, the angles are found by using inverse kinematics on the trajectory point.

A new addition to the revised control system is the input of a sample speed which controls the rate of samples, where each sample refers to one calculation of angular position and velocity for each joint on the CrustCrawler. Once a sample has been calculated and sent to the robot, execution of the trajectory planner pauses until the inputted sample speed has elapsed. After this, a new sample is calculated and passed to the controller for evaluation, given that there is still time left for execution of the movement. The terminal print, on figure 10.5, is the raw output of the trajectory planner for an arbitrary linear movement, which is the data the controller would receive.

The desired angular velocities of the joints are calculated over two steps. First the joint angles corresponding to the desired current position of the CrustCrawler is



```

C:\WINDOWS\system32\cmd.exe

angles = [0.5771740632048104, -0.25396152888784385, 1.1001971253946028]
velocity = [0.3847827088032069, -0.23597434265778836, 0.733464728049729]

angles = [0.7080628633464259, -0.12553656560632054, 1.4133580128783103]
velocity = [0.08725920009441035, 0.08561664218768221, 0.20877392498913835]

angles = [0.7631791233373738, 0.18199039053224564, 1.4602847981658558]
velocity = [0.036744173327298624, 0.20501797075904413, 0.031284523525030306]

angles = [0.7933715347352632, 0.5969874970745379, 1.2516152962590328]
velocity = [0.02012827426525961, 0.2766647376948615, -0.13911300127121531]

angles = [0.7853981633974483, 0.4569340465434354, 1.3467032344935252]
velocity = [0.0, 0.0, 0.0]
Press any key to continue . . .

```

Figure 10.5: This snippet shows the last five packets of angles and velocities for the three main joints of the CrustCrawler. The first four packets were delivered at a sample speed of 1,5 seconds, meaning that 1,5 seconds has passed between each calculation of the values in these packets. This is too slow to be useful but it serves the purpose of illustration. The last packet output is just the goal point sent after the time available for the movements has expired. This explains velocity of 0,0 and ensures the robot is at the desired location.

subtracted from the joint angles corresponding to the next desired position of the CrustCrawler. Then the difference in angular position is divided by the sample speed which give an angular velocity in radians per second for each joint towards the next point on the path.

$$v_{ang} = \frac{P_{Next}(t) - P_{Traj}(t)}{SampleSpeed}$$

When the execution time has passed one final sample containing the angle positions corresponding to the goal point and a velocity of 0,0 rad/s for each joint will be delivered to the controller to mark the end of the movement, see figure 10.5.

Controller

The controller is built according to object oriented programming practises, and it consist of five classes. A full class diagram can be found in figure 10.7 on page 50.

The *Robot* class is an abstract class that contains exactly one object of the low-level *Dynamixel_mx* class, and many objects from the *Motor* class. The purpose of the *Robot* class is to run a background process that constantly writes torque values and reads joint angles and joint velocities. It receives the desired angles and joint velocities from the trajectory planner, and makes sure the CrustCrawler acts accordingly.

There can only be one instance of the *Dynamixel_mx* because only one serial communication can be established at a time. Having the *Dynamixel_mx* object in the *Robot* class and passing it as an instance to the *Motor* class solves the issue with multiple motors trying to access the serial communication. The background process ran by the *Robot* class then controls when and which of the motors can read and write.

The *Motor* class was made to keep track of all the relevant motor data e.g. joint limits, motor ID and desired angle and velocity. The most important elements of

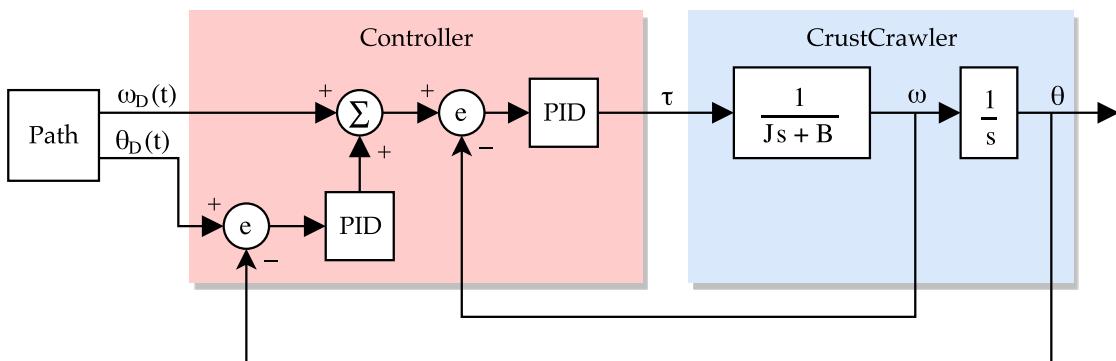


Figure 10.6: This flowchart shows the connection between the two PID controllers in the *Motor* class. $\omega_D(t)$ is the desired angle velocity in radians per second, and $\theta_D(t)$ is the desired angle in radians.

the *Motor* class are the two PID controllers that processes the error values of the angles and joint velocities. The *Motor* class is passive and only when the *Robot* class runs the *update()* method on a motor the following few steps execute: It uses the *Dynamixel_mx* instance to read data, update PID objects and write a torque value.

Looking at figure 10.6 on the preceding page it can be seen how the two PID controllers are connected. One PID controller measures the angle error with the purpose of adjusting the joint velocity in case the manipulator has an offset compared to the desired trajectory position. The other PID controller converts the angular velocity to a torque signal, and makes sure the motor has the desired velocity.

The blue box in figure 10.6 on the facing page is referring to the CrustCrawler. It is given the transfer function $\frac{1}{Js+B}$ which is a simplified model for the response of the CrustCrawler. It might be very different in practise, but there is nothing that can be changed about the response of the CrustCrawler, the only possible change are the two PID boxes on the red controller side. A deduction of the transfer function can be found in appendix C. This concludes how the second controller is constructed.

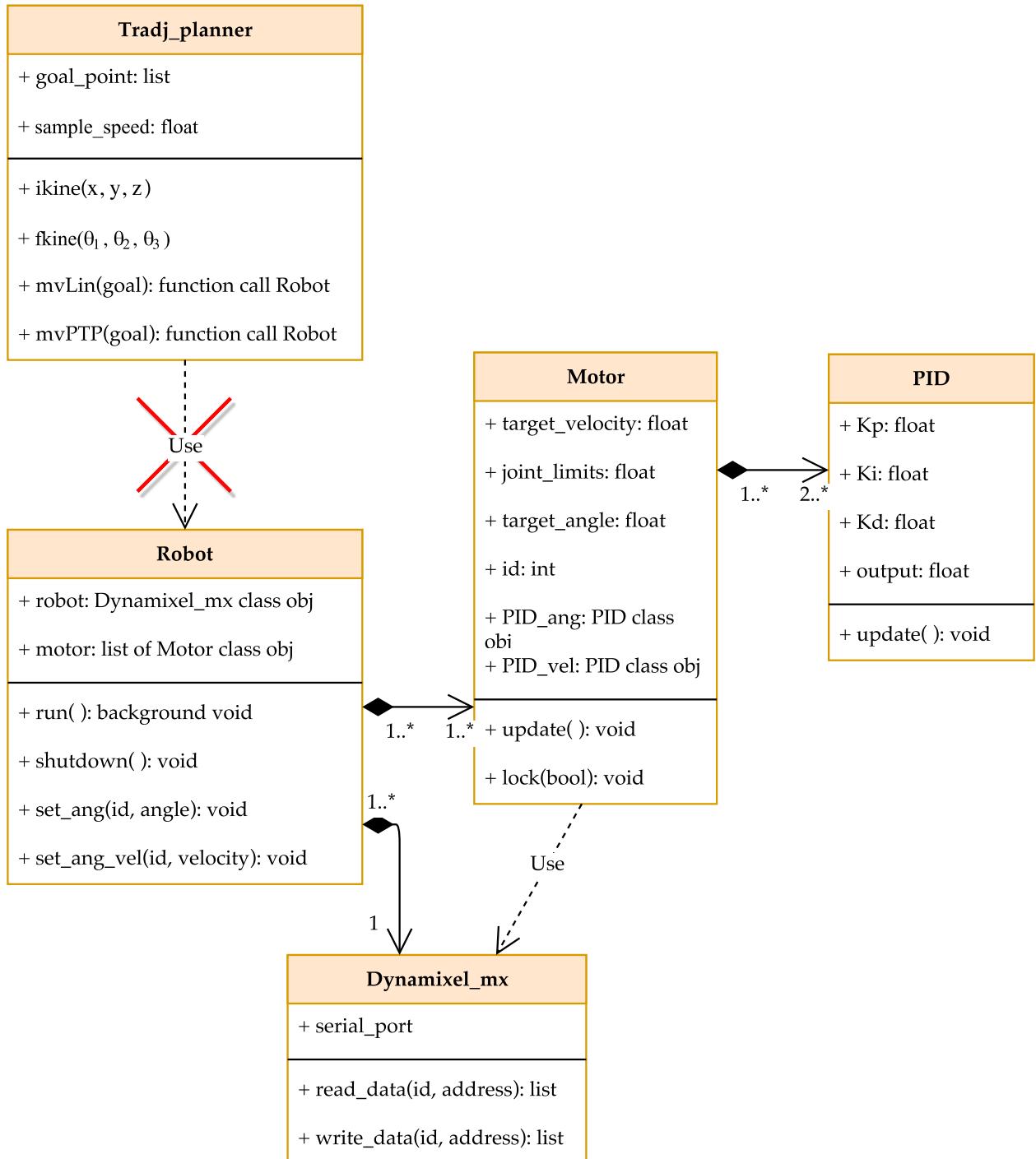


Figure 10.7: Class diagram of the elements of the controller for the CrustCrawler. The red cross is to illustrate a missing implementation of trajectory planner. Reason for missing implementation is described in body text of paragraph 10.2 where the controller is tested and evaluated.

Testing and evaluation

Testing of the controller serves the purpose of getting reasonable parameter values for the two PID controllers. When testing it was quickly realised that the controller sampled too slow to give a satisfactory result. The sample rate was approximately 6,7 Hz which means a pure proportional controller would either raise a steady state error of several degrees, or it would make an overshoot and begin oscillating.

The code was investigated, and it was found that the cause of the slow sample speed was the *Dynamixel_mx* class being slow at handling the reading and writing of data from and to the motors. A read and write speed of 0,015 seconds was measured, meaning that updating a single motor would take 0,03 s and adding that up for all five motors it takes 0,15 seconds to update all motors resulting in a sample speed of 6,7 Hz. It is currently not known whether the slow serial communication is due to the controller being written in a high level language as Python, or if it is due to inefficient programming. Even if the slow execution speed is due to inefficient programming, the likelihood of getting satisfactory sampling speeds with Python after code improvements is considered too low to proceed the development. This is also the reason why the connection between the trajectory planner and the controller was never implemented and the reason for the red cross on the class diagram on figure 10.7 on the preceding page. Thus, it is highly suggested to rewrite the whole controller to a compiled language such as C++.

On the other hand, Python provides benefits as a programming language when it comes to the speed of developing and testing, in the way that it does not need to be compiled. With the slow, but functioning controller, a couple of tests were performed in order to test the abilities of the controller. The first tests were simply aimed at getting some proper PID values for the controller of the manipulator. The test involved movement from an arbitrary position to a fixed goal point as fast as



Figure 10.8: Picture (a) shows the test set-up for the control system without weight and picture (b) shows the same test setup but with 350 g of extra weight added to the blue bag.

possible with an acceptable steady state error and overshoot. With a combination of trial and error and the knowledge about PID controllers from chapter 5 the values 80, 1,7 and 0,41 were found to be a satisfying tuning for K_p , K_i and K_d respectively, for the angle controller. The velocity in the goal point of all tests below has been zero so the velocity controller had the values 1, 0 and 0 for K_p , K_i and K_d respectively.

The second test involved setting a goal point and having the CrustCrawler move to that point (see figure 10.8 on the preceding page). After the CrustCrawler settled on the goal point, extra weight of approximately 350 g was added to the end-effector to see how well the controller would restore the position. The CrustCrawler was also pushed out of position by hand and its ability to return to the goal position was observed.

In order to get the highest possible sample rate the two motors for the gripper were excluded during the second testing. This allowed the sample rate to reach $\frac{1}{0,03 \cdot 3} = 11,1 \text{Hz}$. Since the sample rate changed, new values for the PID controller were evaluated and this time each motor of the three active joints got their own set of values for the PID controllers that can be seen in table 10.1.

The table lists the values that gave the best response without observable overshoot or oscillation. The test can be found on the attached video named "response test - second controller". The test shows how the controller responds quickly to the extra added weight, but gains a big error. The error decreases slowly due to the integral part of the controller. By increasing the integral control parameter it would restore much quicker, but also make a much bigger overshoot when the bottle is removed again towards the end of the video. After the weight has been removed, the robot returns slowly to the goal point again.

The conclusion of the test is that the controller responded as expected, but much slower than what should be considered necessary to implement the control sys-

Table 10.1: This table shows the settings of the PID controllers for the first three joint of the CrustCrawler

	Angle controller [K_p , K_i , K_d]	Velocity controller [K_p , K_i , K_d]
Joint 1	[120, 2, 0]	[1, 0, 0]
Joint 2	[800, 50, 5]	[1, 0, 0]
Joint 3	[300, 50, 5]	[1, 0, 0]

tem. With a much faster sample speed the controller should be able to respond so quickly that it would not move at all when adding weight or pushing it by hand.

This chapter gave a description of the two control systems developed for the CrustCrawler. The first controller caused jagged linear movements for which reason a second controller was developed, which solved this flaw. However, a low sampling time resulted in non satisfying performance and therefore it cannot be considered as an applicable control system in the development of prototypes. For this reason, the first controller will be used in the following development phase.

PROTOTYPES

In this chapter the prototypes made will be explained, the requirements for the prototype will be listed and the prototype will be tested to see if the requirements are met.

11.1 Description and specific requirements

The prototype in this report will illustrate a rehabilitation application based on the active approach that was described in section 2.2. Consequently, the patient will have to be actively involved in the rehabilitation exercise. The goal is to create a pick and place game, stationed in either the home of the patient or in rehabilitation centres and hospitals. The game involves moving a ball from a start point to an end point with help of the CrustCrawler. When the ball is dropped at the end point, it should return to the starting point by a slant, thereby creating a repeating cycle.

Specific requirements are established in order to provide an understanding of the development goal of the project while guiding the development by limiting the range of viable product designs. Product development based on well defined requirements will assist in the process of reaching a solution that is performing as desired. After development, the requirements can be used to evaluate the solution. The items listed under external interfaces are requirements to the surrounding environment of the system which, when fulfilled, will ensure that the system is working. The functional requirements refer to the layout of the system and the tasks conducted by it while performance requirements will list criteria related to the performance of the robot system in measurable terms.

1. External interfaces
 - (a) The system requires a power supply delivering 10-14,8V as mentioned in section 4.2. It must be possible to get power from common sockets allowing the patient to use the system in their homes.
 - (b) Section 4.2 explained that the Dynamixel servos use the RS-485 protocol for serial communication, for which reason this protocol must be used.

2. Functional requirements

- (a) The patient should be able to start the game through a graphical user interface. This is important to reduce complexity.
- (b) The patient must repeatedly interact with the manipulator through feature extractions performed by the Myo Armband during the rehabilitation exercise.
- (c) The game shall involve wrist, elbow and shoulder movements since section 2.2 revealed that these are recommended in rehabilitation exercises.
- (d) The manipulator of the game must be able to grasp, hold and release a ball.
- (e) The locations of ball pick-up and ball placement must lie within the operating area shown in figure 11.1 on the next page. This assures visibility of the manipulator's movements for the patient during rehabilitation.
- (f) The patient must have the ability to terminate the game cycle.
- (g) Conventional skin preparation methods for sEMG recordings, mentioned in section 3.2, should not be required for the gesture recognition to work. This is to avoid further restrictions of the user characteristics.

3. Performance requirements

- (a) The end-effector movement must be initiated within one second of muscle activation by the patient to ensure meaningful and rewarding feedback in the sense that the patient will be able to see a relation between muscle activation and manipulator movement.
- (b) The application must inform the patient nonverbally when a gesture is to be performed, and keep reminding the patient every second if not performed. This will serve as an alternative way of communication as it is known from section 1.1 that hemiparetic patients may have trouble understanding verbal communication.
- (c) A delay of at least one second between required actions from the patient must be incorporated.
- (d) The repeatability of the manipulator should be efficient enough to place the ball at the correct end point with a success rate of 100%.

11.2 Testing and requirement verification

Description of the prototype

Prototype **Version 1** of the game was constructed in such a way that the patient have to grasp and release the ball at specific points in a cycle. The manipulator

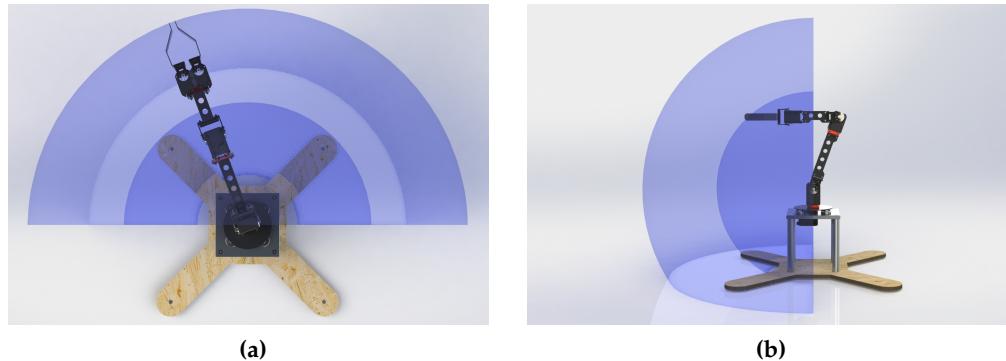


Figure 11.1: (a) The figure is showing the operating area from the top perspective where the CrustCrawler is reached as far out as possible. (b) The CrustCrawler operating area shown from the side, in which the balls pick-up and placement location must lie.

moves between a grasp point (see figure 11.2a) and a release point (see figure 11.2b). When the manipulator reaches the grasp point, the patient will be informed to make a gesture by two short vibrations from the Myo Armband. If no gesture is made by the patient, reminder vibrations will be sent every second until the desired gesture is made. The patient has to make a *fist* gesture to make the gripper grasp the ball. The ball will be moved to the release point where the patient is informed again by two short vibrations to make a gesture. The gesture to be made in the release point is a *spread* gesture. Again, if no *spread* gesture is made vibrations will be sent every second in order to remind the patient to perform the gesture. When the spread gesture is made the ball will be dropped into a slope which transfers the ball back to the grasp point. A flowchart of the game can be found on figure 11.3 on the facing page

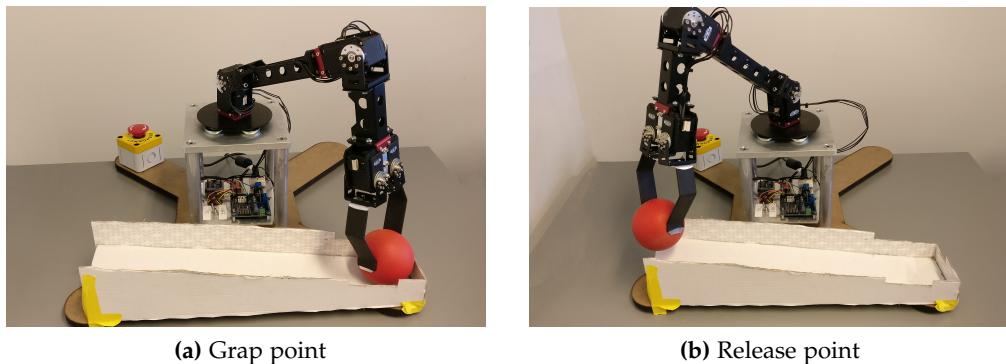


Figure 11.2: The grab and release points in **Version 1** of the prototype.

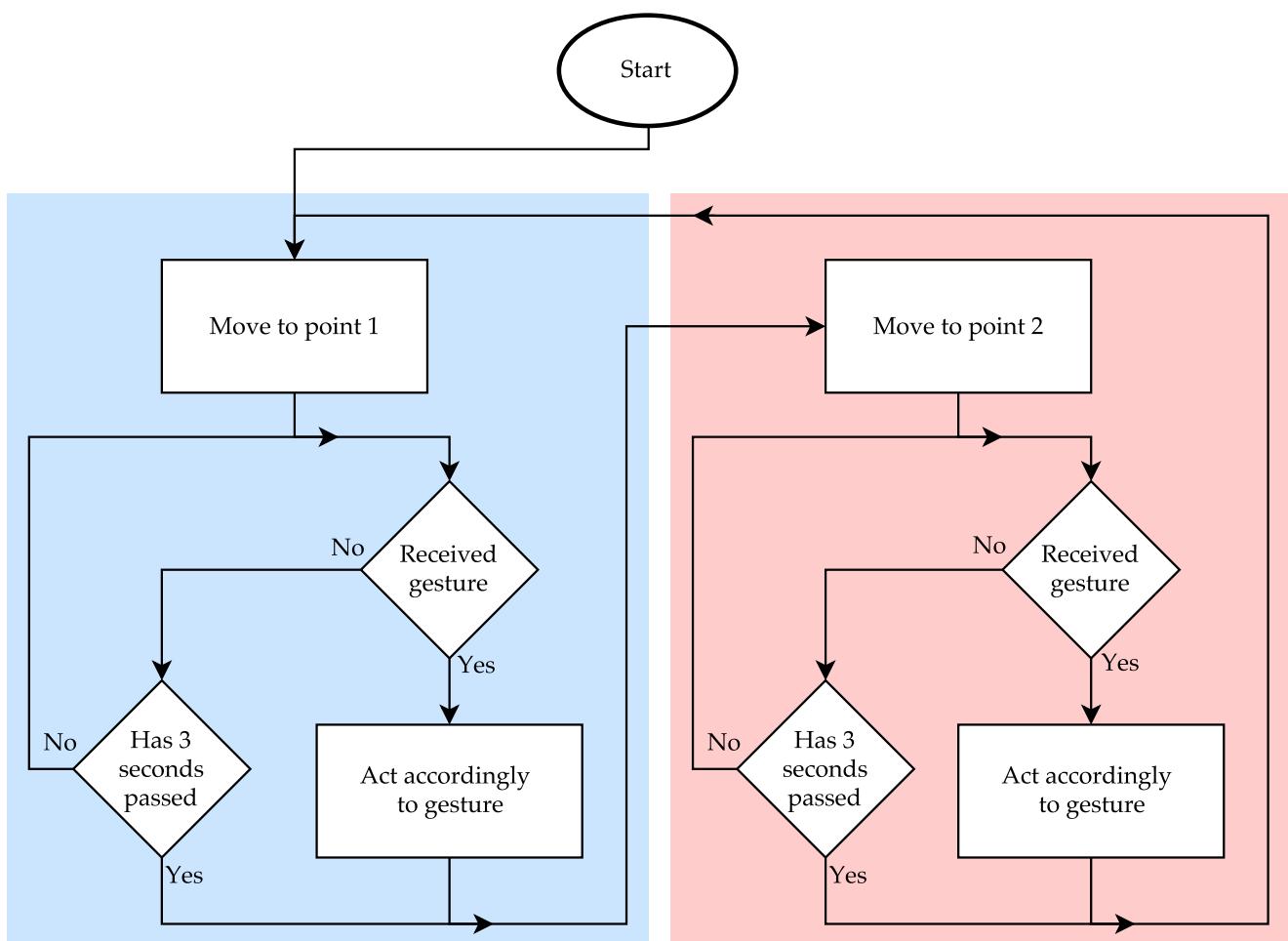


Figure 11.3: Flowchart showing the process of the game flow in prototype version 1

Prototype **Version 2** focused on getting more wrist movement implemented as part of the game. Similarly to **Version 1** the manipulator moves between two points grasping and releasing the ball. Now halfway between these points, the manipulator stops in a via point and waits for a feature to be performed by the patient. The patient needs to wave in the direction the manipulator is heading by making use of the *wave in* and *wave out* gestures. The program can be seen in the video “Version 2”. A Graphical User Interface (GUI) was also made, as seen in figure 11.4, to help the user starting and stopping the game.

The previous two prototypes lacked elbow and shoulder movements which were listed in requirement 2c. In order to implement these movements, prototype **Version 3** utilises the built-in accelerometer in the Myo Armband. The accelerometer provides *proper acceleration* along the three cartesian axes X, Y and Z. Looking at one of these axis the program can measure acceleration of the armband. When launching the program it records a starting acceleration which later on in the game cycle is compared to a current acceleration. The difference between the two will determine if the manipulator should move or not. This triggering difference can be set prior to the start of the program thereby enabling the specialists to customise the triggering difference based on the severeness of the patient’s hemiparesis. The terminal window of the game will inform the patient on the movements and gestures which should be performed during the game. The movement pattern of the manipulator is the same as in the two prior prototypes but now it is divided into additional steps. The linear movements from and to grasp- and release points will now be initiated by a difference in accelerometer values. The whole cycle can be seen in the video titled *Version 3*. The problem of using the accelerometer is that the proper accelerations will change according to the orientation of the armband. The movements can therefore be triggered by moving your arm in many directions. Another problem is that a certain difference in acceleration must be achieved which might not be possible for some patients.

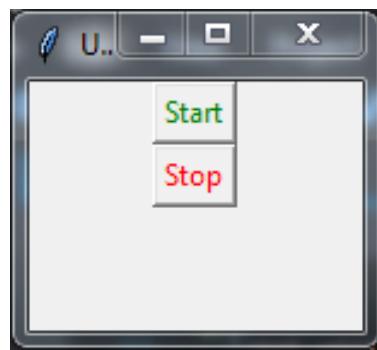


Figure 11.4: Graphical User Interface.

Version 4 of the prototype accounted for the accelerometer pitfalls by utilising the built-in IMU of the Myo Armband described in section 4.1 on page 18. The IMU is used to get the orientation of the armband and thereby determine which way the arm is pointing. This is implemented in the same way as the accelerometer was implemented in the cycle in **Version 3** but now the patient has to explicitly point up for a upwards movement of the manipulator and down for a downwards movement. The triggering difference in acceleration can still be customised to the patients.

In order for this version to be implemented in all environments a reference frame must be made. Therefore, the hand must be placed on a table when the start button of the GUI is pressed, see figure 11.5a, the reason being that the IMU initially is based on Myo's own coordinate system, which will vary depending on the placement of the arm. This is done by taking the initial values from when the user presses the start button and then conjugate these values. The frame is then created by multiplying the centre with the incoming orientations from the IMU, which is represented by a vector of four quaternions(x, y, z, w) which are converted into Euler-angles(roll-pitch-yaw). In this version only the extrinsic rotation around the x-axis is of interest in order to see if the user is lifting or lowering the hand. Even though this works satisfactorily there are still some downfalls of doing it this way because only one value is operated upon, meaning that the arm in some positions might trigger the lifting and or lowering of the manipulator. To avoid this, all parameters must be taken into the equation, i.e. the rotations around all axes.

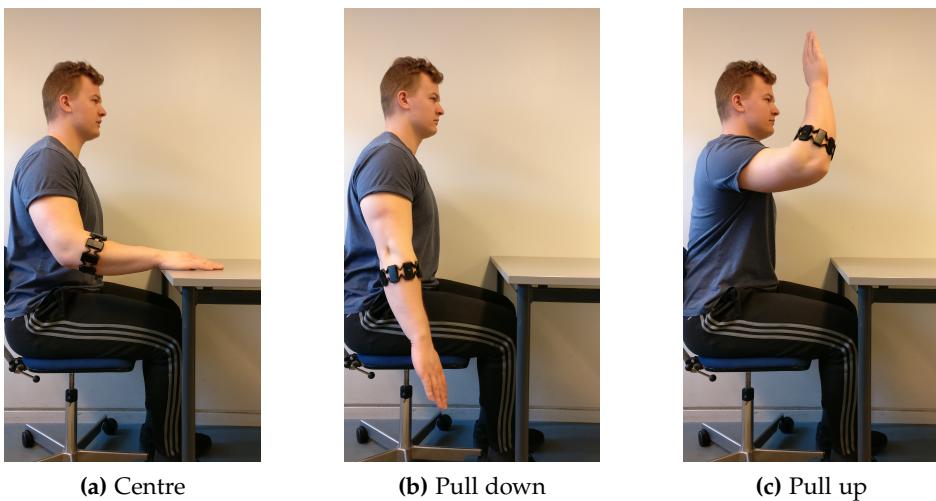


Figure 11.5: All arm positions for IMU

Moreover a new feature in the GUI was made in order to set a “difficulty” of the game in terms of the necessary degrees of rotation around the x-axis before it

would trigger. The GUI can be seen on figure 11.6. On the left there is no input, which is why the start button cannot be pressed. On the right it is ready if a valid value is entered, i.e. a float. On figure 11.5 (b) and (c) the positions needed to trigger the “pull up” and “pull down” with a value set on 1.3 can be seen. With a substantial difficulty, the pull-up will also require shoulder movement as seen on figure 11.5c on the previous page. If a patient lacks too much mobility in the shoulder, the value can be lowered whereafter the triggering will require barely any shoulder movement, thereby changing the focus to elbow mobility, and enable patients with a high degree of hemiparesis to play the game.

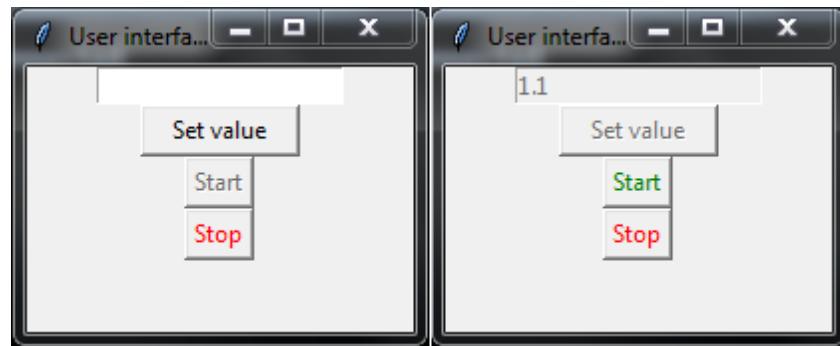


Figure 11.6: GUI - the left is before the value is entered, the right is after.

Verification of the requirements

Version 4 is the final version of the prototype made in this project. Thus, the requirements will be verified based on the outcome of this prototype version.

1. External interfaces
 - (a) The system gets power from the 230V socket which an adaptor converts into 12V for the system. For this reason the system can easily be used in any home.
 - (b) RS485 is used as it is mandatory.
 - (c) The Dynamixel servos received the required voltage.
2. Verification of functional requirements
 - (a) The prototype manages to fulfil this requirement through a GUI with both start and stop buttons.
 - (b) The patient will interact with the manipulator eight times each cycle. Four of these interactions require feature extractions recognized by the Myo Armband.

- (c) The game invokes wrist movement from the features *wave in* and *wave out*. The elbow and shoulder movements are involved by forcing the patient to point their arm up and down by use of quaternions.
- (d) The manipulator is able to grasp, hold and release a ball. The ball has a diameter of 70 mm.
- (e) The manipulator moves within the set workspace.
- (f) The patient can terminate the game via the GUI.
- (g) None of the six persons from the testing had any skin preparation done, therefore it can be concluded that this requirement is fulfilled. Test can be found in appendix B on page 67.

3. Verification of performance requirements

- (a) The end-effector and the manipulator reacts within one second of muscle activity from the patient. Results of this test can be found in appendix B on page 67.
- (b) Nonverbal information to the patient is provided by the vibrations from the Myo armband. These are delivered within the required time span of two seconds and the requirement is therefore fulfilled.
- (c) The time between actions required by the patient ranges between 1,3 and 4,4 seconds. Test results can be found in appendix B on page 67.
- (d) The cycle was seen to be repeatable within the requirement. Test can be found in appendix B on page 68.

Through iterative optimisation of the prototype, a final version managed to comply with all given requirements.

DISCUSSION

The EMG sensor

The Myo Armband has some limitations, like the placement of the armband and the lack of raw EMG signals. These constraints limit the development process, since the prototype must use the already predetermined feature extractions. An EMG recording device sending raw signals would allow for customised feature extractions through signal processing. This opens up the possibilities of enhancing the project with new features, like applying focus towards fine motor skills, or bigger movements of the arm. The other device would have to be of the same quality as the Myo Armband, which currently is the only armband on the market sufficient enough to be used for this task. Due to the fact there was no testing on actual hemiparetic patients, there is no knowledge of whether the Myo Armband will be able to record any EMG data from the patients, which could cause problems with feature extractions. Therefore, if the project is to be developed any further the Myo Armband will need testing on patients suffering from hemiparesis. The same simplicity seen in the Myo Armband is needed if this product is to be used in the homes of the patients. This is because if the setup of the system is either too complex or the time it takes to do the setup is too long it might discourage the patient from using the system. The device must record surface EMG, and the idea of using an armband seems ideal for the vision of the project.

Future development

Choosing a manipulator able to use the method of *teach by showing* would allow the specialist e.g. physiotherapist to easily design new games for the patient or maybe alter already existing games to fit the patient's needs better. Instead of games, programmed with the *teach by showing* method, the manipulator could be helping with ADLs in the home. This requires the manipulator to be mobile enough to be moved around without any struggle e.g. if wheels were put on and it could be controlled.

It should also be ensured that the robot used, complies with safety standard ISO/TS 15066:2016 about collaborative robots. Another standard, currently under development, would be the IEC/NP 80601-2-78 about "basic safety and essential performance of medical robots for rehabilitation, compensation or alleviation of

disease, injury or disability". The latter would be the most fitting for a system like the one developed in this project.

Moreover, a social game environment could be a way to motivate patients to dedicate more time towards the rehabilitation, because of the aspect of playing with or against others. Combining this with a competitive aspect of wanting to better yourself and beat others have potential as a major motivational factor. All patients might not find the same game entertaining and/or motivating, so by including various games there might be something for the individual patient.

Micro-controller and embedded software

The final product could be equipped with a micro-controller responsible for managing the control system of the robot. The micro-controller would have embedded software only focusing on the control and thereby cutting of unnecessary processes. There will then have to be another piece of hardware handling the storage of games, data communication and visual connection to the patient i.e. an easy and intuitive GUI. This could be a tablet with pre-written software from the manufacturer, or simply the patient's own tablet or PC. The required software will depend on either of these choices. An arrangement from the manufacturer's side can with embedded software be build into the tablet, customising it to handle the specific tasks. Designing a program to run on the patients own devices will require some initial setups which could be avoided the other way around. Either way which platform is used for handling the code, the system will need to be connected to the Internet, this can either be an existing Wi-Fi in the home or through a mobile network, which is needed in order to achieve the conceptualisation. When using an electronic product in an unknown environment there could be electrical noise that could interfere with the system which needs to be investigated.

Manipulator choice

Since the final product equipment is not settled there will be given an estimated guess of what the final cost will be. The estimate is 200.000 danish crowns as a max cost. The manipulator used as replacement of the current will need to be a collaborative robot, which means it complies with danish laws and regulation of this. Moreover, the manipulator will need a certain range and payload, which is the weight of an arm, in order to achieve the active rehabilitation as mentioned in chapter 6, moreover the size of the robot will need to be limited if it is to be used in the home. This is both in terms of the system size as a whole but also the portability of it. The robot which will be used as a reference point in this discussion is the Universal Robots(UR) 10, which has a payload on 10kg. The average human arm weighs 5-6 percent of the total body weight depending on gender according to Plagenhoef et al, 1983. This manipulator also has 6 DoF and the reachable workspace is 1300mm, the weight of the UR 10 is 28.9kg and the control box weighs 17kg which means it can be moved to the home without needing any

bigger equipment, but the patient might have problems moving it around in the home. The most important reason for choosing the UR as a reference is due to its collaborative technology, which allow the manipulator to be used with the patient standing next to it without worrying about hurting themselves, others nearby and the surroundings. This estimate is given with the cost of UR 10 in mind, which is around 195.000 danish crowns. The other equipment used is the Myo Armband, which cost around 1500 danish crowns. This is a high end estimate, if there is not use for the robot to have a payload of 10kg the cost will go down. The estimated price is only for the hardware used for the final product, man hours etc. is not included.

Restrictions and ethical problems

The prototype can not yet be tested on hemiparetic patients in order to test the efficiency, motivational factor nor whether the game is fun or not. This is because the ethical council in Denmark needs to give the project an approval before any testing on patients can be done. If the project are to be further developed this needs to be done in order to optimise both the current game, and also for the development of new games in collaboration with experts, which can then be tested on patients. A large ethical problem when working with patients is if the games do not give any progress it might be demotivating and deterring for any further rehabilitation, which means before any larger scale testing on patients the system must be validated. As previously mentioned in section 9.4 the sensitive personal data will need to be encrypted, this is due to the danish law about sensitive personal data, which also means that the data only can be seen by experts.

The possibilities of data collection

The idea of data collection and the vision behind have already been presented in this report. Equipping the patient with an armband with an accelerometer on the prohibited arm will help determine progress in daily activities. Combining the EMG data with accelerometer data will give a large data set for specialist to examine and thereby customised the rehabilitation to be patient specific.

Hemiparetic patients are affected throughout the entire side of the body, so incorporating readings from other muscles than the forearm have a high potential. Reading data from the biceps and triceps and utilising these into the program could aid in rehabilitation a larger area of the body all at once.

In theory all skeletal muscles has the ability to be incorporate into this product, but surface muscles would be of higher priority to avoid use of invasive electrodes.

CONCLUSION

This project examined the possibility of improving the rehabilitation process of upper extremities of hemiparetic patients with the use of robot technology. In order to do this an analysis of the problem was conducted, which served the purpose of acquiring necessary contextual knowledge to the problem of the project. Following the analysis, the goal of the project was depicted as a rehabilitation game. A conceptualisation, describing a hypothetical goal product was conducted, which served as a framework for the development in collaboration with the following delimitation.

A prototype was developed to demonstrate certain aspects of the conceptualisation. After iterative development and testing, a final prototype using the EMG sensors and the IMU of the Myo Armband was produced. A combination of these sensors ensures movements of both wrist, elbow and shoulder. The prototype presented an active rehabilitation game, which utilises the end-effector of the CrustCrawler to grasp a ball. The ball is then moved and released at another point following actions performed by the patient playing the game. The movements of the CrustCrawler during the game are either point to point or linear and these are regulated by a designed control system making use of a PID controller. It was concluded that the control system behind the rehabilitation game was not functioning optimally but still, it performed sufficiently enough for development of the prototype.

A second control system was developed in order to address the problems of the first control system, which succeeded. However, new problems arose with this control system due to a slow sampling rate of maximum 11,1 Hz, and for this reason the second control system was never implemented in the rehabilitation game and the first control system remained the solution.

The final prototype was developed to create a fun and motivating rehabilitation application, but conclusion on whether this was accomplished would require testing on hemiparetic patients which was not conducted.

MICRO-CONTROLLERS

A way of controlling the dynamixel servos is by use of an Arbotix-M microcontroller. [50] The Arbotix-M (figure A.1a) is a AVR microcontroller with a 16 Mhz ATMEGA644p processor and 28 input ports (20 digital and 8 analog). Programs can be uploaded to the Arbotix-M board through an Arduino IDE version 1.0.6. An advantage of using the Arbotix-M board is that it is dedicated to only one task, namely the control of the servos. The RS-485 breakout (figure A.1b) is used to convert the UART serial stream to RS-485 in order to connect the Dynamixel servos to the Arbotix-M board. [51] A UartSBee V4 which can be (figure A.1d) used as a USB to serial adapter, but can be utilised for wireless control of the robot. [52]

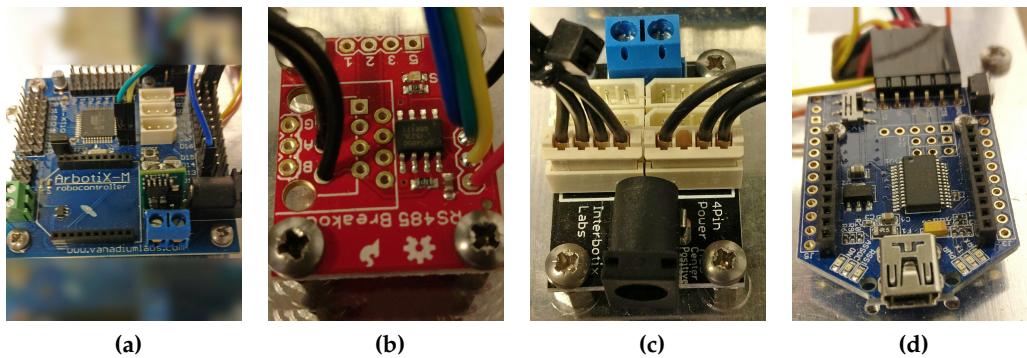


Figure A.1: (a) Arbotix-M board. (b) RS-485 board. (c) Power hub. (d) UartSBee V4.

TESTING OF THE PROTOTYPE

Skin preparation test

The Myo Armband's ability to register features from patients without any conventional skin preparation was tested by looking at the features generated from 6 different test persons. Each test person was asked to perform all the five gestures(?? on page ??). All of the 6 test person was able to perform the gestures. This concludes the device is able to be used without any skin preparation.

Reaction test

The time between a feature from the patient and a action from the manipulator was recorded for 10 features. The results below are in seconds:

1. Wave out - 0,048
2. Pull down - 0,040
3. Fist - 0,047
4. Pull up - 0,044
5. Wave in - 0,040
6. Pull down - 0,041
7. Spread - 0,041
8. Pull up - 0,039

All the test data showed a reaction time below 0,048 seconds.

Range test

The game went through three cycles and the time for each movement to be completed was recorded in seconds:

1. Wave out - 1,379; 1,381; 1,381
2. Pull down - 2,920; 2,931; 2,921

3. Fist - 0,689; 0,342; 0,695
4. Pull up - 4,530; 4,569; 4,485
5. Wave in - 1,521; 1,515; 1,528
6. Pull down - 3,024; 3,013; 3,033
7. Spread - 0,531; 0,641; 0,563
8. Pull up - 3,119; 3,141; 3,099

At all these steps an action is required from the patient. The time between all these movements were recorded, in order to see if any of the times are faster than one seconds. Due to a requirement which states that there must be a wait time of at least one second between each movement, the velocity of fist and spread must be altered in order to cope with this. Therefore, a second test was conducted with the end-effector velocities changed:

1. Fist - 1,111; 1,119; 1,088
2. Spread - 1,027; 1,024; 1,031

The prototype now complies with the mentioned requirement, because the speed of the end-effector was lowered.

Reliability test

The total cycle was ran through 30 times in order to check reliability of the designed program and cycle. The test resulted in 30/30 complete cycles.

LINEAR MODEL OF DYNAMIC BEHAVIOUR

In the following, a linear model of the dynamic behaviour of a one DoF manipulator representing the first link of the CrustCrawler will be deduced. The manipulator in question is illustrated on figure C.1 on the following page. To do this, Newton's 2nd law for rotations is utilised.

$$\sum \tau = I\alpha \quad (\text{C.1})$$

When this equation is applied to the manipulator of figure C.1 it obtains the following form,

$$\tau(t) - B\omega(t) = J\dot{\omega}(t) \quad (\text{C.2})$$

where τ is the torque provided by the servo motor and $B\omega$ is a resisting torque which is dependant on the angular velocity, ω . However, for this example it is presumed to be constant due to simplification. Torque occurring due to gravitational force can be calculated as $\cos(\theta) \times \frac{L}{2} \times mg$, but since it is a linear model that is being derived, the torque due to gravity is considered as a constant, meaning it can be interpreted as a part of B in equation C.2. J is the moment of inertia and $\dot{\omega}$ is the angular acceleration of the manipulator expressed as the time derivative of angular velocity.

An application of the Laplace transform yields the following equation,

$$\tau - B\omega(s) = J \cdot s \cdot \omega(s)$$

which can be solved for $\omega(s)$

$$\omega(s) = \frac{1}{(Js + B)} \cdot \tau(s)$$

Finally, J is replaced with the moment of inertia for a unified rod where the axis of rotation is located at the end of the rod. Again, this is a simplified expression of the moment of inertia for the links of the CrustCrawler as these cannot be considered as unified rods.

$$\omega = \frac{1}{\left(\frac{mL^2}{3}s + B\right)} \cdot \tau$$

This equation gives a relationship between motor torque and angular velocity which enables calculations of torque given a angular velocity or vice versa. This is useful in control systems like the one of section 10.2, but since this model is a linear approximation of the actual behaviour of the CrustCrawler which, in addition, only applies to a single link, it was not implemented in the control system.

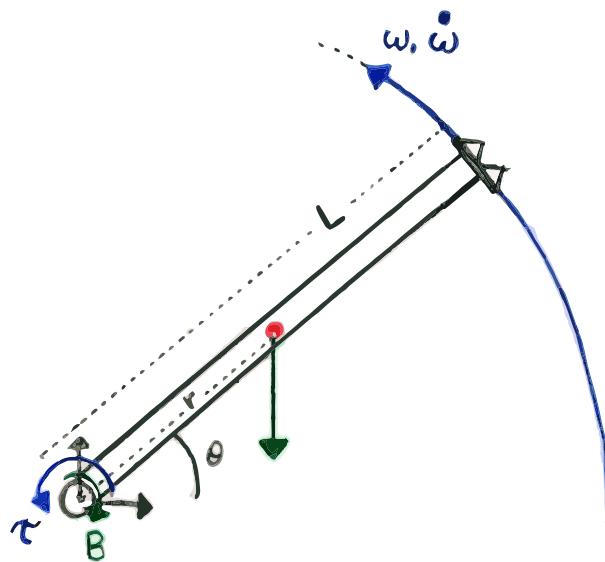


Figure C.1: A one DoF manipulator representing the first link of the CrustCrawler. The figure shows all acting forces on the link, which includes τ , a torque produced by the servo motor, B , a resisting torque due to e.g. friction and gravitational force, mg , which is illustrated by the green arrow emanating from the center of mass. r is the distance from the axis of rotation to the center of mass while L is the length of the link, and ω is angular velocity while $\dot{\omega}$ is angular acceleration.

SIGNAL PROCESSING AND ANALYSIS OF EMG

The chapter describing signal processing and analysis of EMG signals, was throughout the process of writing the report moved to appendix. This is due to the fact that the EMG signals received from the Myo Armband were challenging to deduct and the fact that the five feature extractions was chosen for use in the prototype. The chapter is of importance when reevaluating what optimisation could be done. An optimal solution would use a EMG recording device enabling us to process and analyse the raw EMG signals.

D.1 Signal processing

This sections introduces five possible ways to process EMG signals.

Full wave rectification

A raw EMG signal consist of both negative and positive amplitudes. A full wave rectification takes all negative amplitudes and converts them into positive amplitudes. This allows for more convenient reading of the signals, along with the possibility for improved amplitude analysis involving minimum and maximum peaks. The result of a full wave rectification can be seen on figure D.1 on the next page. [38]

Smoothing

When looking at raw EMG signals the same specific signal would not appear twice. This is due to the fact that the amount of available motor units changes slightly with each contraction and the fact that the interference between motor unit action potentials vary. The peak amplitudes of raw EMG signals differ which troubles comparison between signals generated by the same movement. To avoid this problem smoothing algorithms can be applied to the signals; either raw EMG signals or already rectified signals can be smoothed. The smoothing algorithms cuts off the height of the peaks making the signals more similar and comparable. The smoothing algorithm *Root Mean Square* accomplishes smoothing of the signals by taking

the square root and returning the mean of the signals. Figure D.2 on the facing page showcases a root mean square smoothed EMG signal. [38]

Digital filtering

After taking the raw EMG signal through rectification and smoothing additional filtering is generally not recommended for kinesiological studies. This is due to the fact that too much filtering alters the EMG signals too much and makes the readings inaccurate. However, it is possible to apply either low pass or high pass filters if a specific frequency needs to be removed from the signal e.g 50-60 hz. [38]

Amplitude normalization

The amplitude of EMG signals tend to be varying greatly between subjects but also from the same subject. For example a subject's EMG signal originating from the same muscle is not given to have the same amplitude on different days. Processing with amplitude normalisation compensates for this. Normalising a signal and finding the maximum value of electrical activity (100%) from a specific muscle enables the ability to investigate percentages of the maximum muscle activation created by specific muscles. It is then possible to analyse the percentages of muscles activity needed to perform a specific task e.g. grabbing and lifting a coffee cup. Processing signals by normalizing their amplitudes makes it possible to conduct quantitative statistical measures of EMG signals. Comparing one subject to another is possible if a reference signal is conducted through amplitude normalization methods.[38]

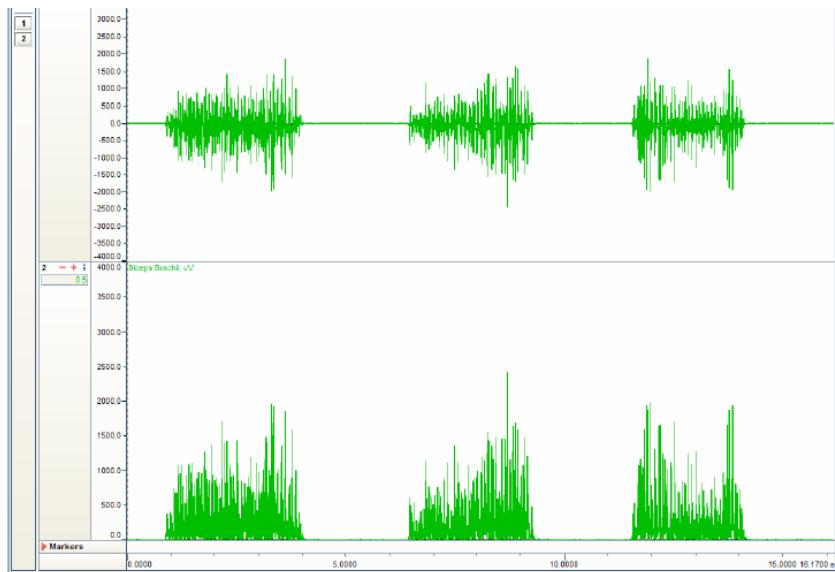


Figure D.1: Top half shows the raw EMG signal, bottom half shows the same EMG signal now rectified. [38]

ECG reduction

Electrocardiography (ECG) signals stemming from the heart have a center frequency of 80 Hz. The ECG signals can travel through tissue and affect EMG recordings from muscles close to the heart e.g chest muscles and shoulder muscles, which will affect the interpretation of the EMG negatively. Luckily, the ECG has very distinctive characteristics making it easy to point out from a an EMG recording. Afterwards it can be eliminated by means of ECG reduction. [38]

The figure D.3 on the next page shows how the ECG affected EMG recordings can look before and after an ECG reduction algorithm is ran on the signals.[38]

D.2 Signal Analysis

EMG signals can be analysed with different purposes. This section describes various analysis methods with the purpose of obtaining knowledge of common applications of EMG signals. Different analyses require different types of signal processing. [38]

Determination of whether the muscle is active or not

Determining whether a muscle is active or not is a basic analysis of the EMG signals that can be done already to the unprocessed raw EMG signals. The analysis can help determine if the muscle in question is generating EMG signals. This analysis will produce a result in nominal value e.g. yes or no. [38]

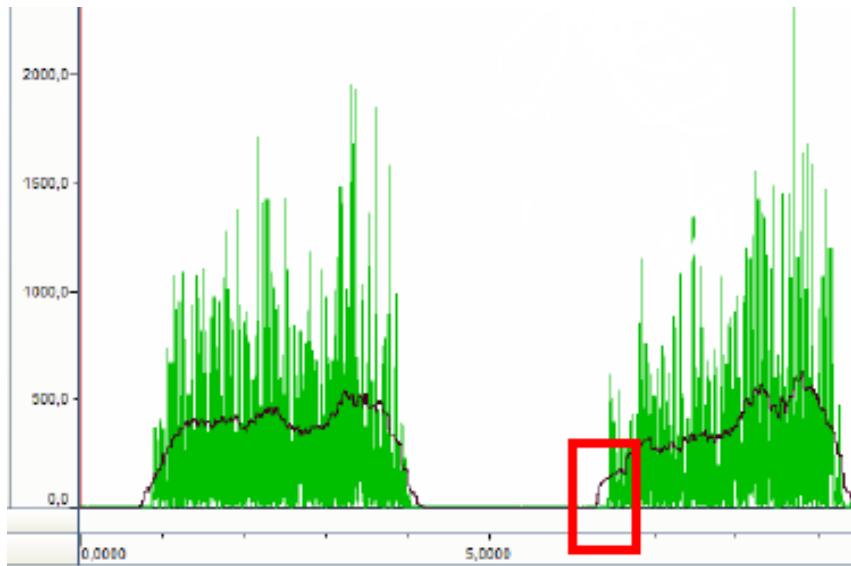


Figure D.2: Root Mean Square smoothing on a rectified EMG signal from the biceps brachii muscle. [38]

Is the muscle more or less active?

The question of more or less muscle activity is answered by a comparative test between two or more EMG measurements. It can be a valuable method in e.g. ranking muscle activity from different contractions of the same muscle or comparing the EMG from the same muscle, but on the left and right side of the body respectively. [38]

When is the muscle on/off?

A muscle can be referred to as *on* if the electrical signal obtained from the muscle exceeds a certain threshold. With this knowledge it is also possible to register the points in time where different muscles are active during a movement, e.g. a gait cycle. This is known as the timing characteristics of muscles. Determination of timing characteristics can be used to analyse the reaction time of muscles or the activation order of muscles. Since the timing characteristics are identified based on a threshold, the credibility is heavily reliant on the definition of the threshold.[38]

Muscle activity level

Determining the specific muscle activity level is done by finding the percentage of EMG compared to reference maximum EMG obtained under static conditions. This results in a measurement on the metric scale i.e. a number, which opens for

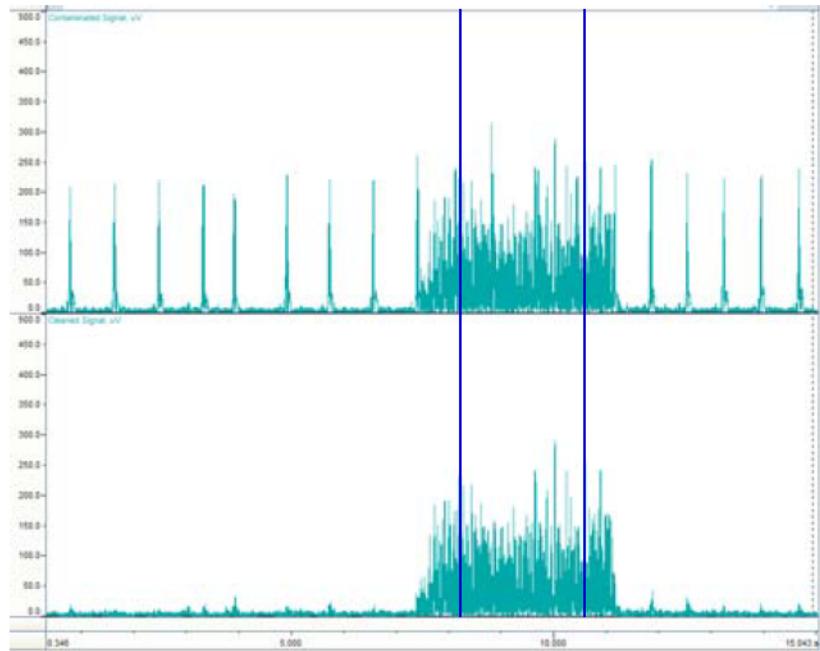


Figure D.3: Top half shows EMG signals before ECG reduction is done and bottom half shows the EMG recordings after ECG reduction. [38]

possible research questions e.g. what percentages muscle activation is required for the forearm muscles when lifting a cup from a table? [38]

Muscle fatigue

Muscle fatigue is commonly investigated through analysis of EMG recordings from static contractions with a well defined angle position/muscle length and a constant load on the muscle. Recruitment of motor units during the contraction will cause the amplitude of the EMG signal to increase while the frequency will see a left shift towards lower frequencies due to a decrease in the velocity of the action potentials among other reasons. Investigation of muscle fatigue is useful for identification of weak muscles as well as determination of the effectiveness of hypertrophy training as muscle fatigue is an important aspect of muscular growth.[38]

BIBLIOGRAPHY

- [1] Colin Blakemore and Sheila Jennett. *The Oxford Companion to the Body*. Oxford University Press, 2001. ISBN: 9780198524038. DOI: 10.1093/acref/9780198524038.001.0001.
- [2] M. Z. Koubeissi; A. Alshekhlee; P. Mehndiratta. *Seizures in Cerebrovascular Disorders: A Clinical Guide*. Springer New York, 2015. ISBN: 978-1-4939-2558-2. DOI: 10.1007/978-1-4939-2559-9.
- [3] Esben Meulengracht Flachs et al. *Sygdomsbyrden i Danmark*. Tech. rep. Sundhedsstyrelsen, Oct. 2015. URL: <https://sundhedsstyrelsen.dk/da/sygdom-og-behandling/~/media/00C6825B11BD46F9B064536C6E7DFBA0.ashx>.
- [4] Christina Rostrup Kruuse. *Apopleksi, genopträning*. Web. June 2015. URL: <https://www.sundhed.dk/borger/patienthaandbogen/hjerte-og-blodkar/sygdomme/apopleksi/apopleksi-genoptraening/>.
- [5] Elizabeth Martin. *Concise Medical Dictionary*. 9th ed. Oxford University Press, 2016. ISBN: 9780199687817. DOI: 10.1093/acref/9780199687817.001.0001.
- [6] Gyldendals Røde Ordbøger. *Atherosclerosis*. Web. URL: <https://ordbog.gyldental.dk/#/pages/result/daklin/atherosclerosis/expert>.
- [7] Jens Kastrup. *Åreforkalkning*. Web. May 2016. URL: <https://www.sundhed.dk/borger/patienthaandbogen/hjerte-og-blodkar/om-hjerte-og-blodaarer/aareforkalkning/>.
- [8] Elizabeth A. Martin and Tanya A. McFerran. *A Dictionary of Nursing*. 6th ed. Oxford University Press, 2016. ISBN: 9780199666379. DOI: 10.1093/acref/9780199666379.001.0001.
- [9] Christina Rostrup Kruuse. *Apopleksi, -Blodprop eller blødning i hjernen*. Web. Aug. 2014. URL: <https://www.sundhed.dk/borger/patienthaandbogen/hjerte-og-blodkar/sygdomme/apopleksi/apopleksi-blodprop-eller-bloedning-i-hjernen/>.
- [10] Christiana Care Health System. *Long-term Effects of Stroke*. URL: <http://www.christianacare.org/howstrokecanaffectionyourlife>.
- [11] Christina Rostrup Kruuse. *Forstyrrelser i kropsopfattelsen*. Web. June 2015. URL: <https://www.sundhed.dk/borger/patienthaandbogen/hjerte-og-blodkar/sygdomme/apopleksi/forstyrrelser-i-kropsopfattelsen/>.

- [12] Robert Teasell and Norhayati Hussein. "Clinical Consequences of Stroke". Evidence-Based Review of Stroke Rehabilitation. Sept. 2016. URL: <http://www.ebrsr.com/evidence-review/2-clinical-consequences-stroke>.
- [13] Oxford English Dictionary. *Hemiparesis*. URL: <http://www.oed.com/view/Entry/85797?redirectedFrom=Hemiparesis#eid1707884>.
- [14] Gyldendals Røde Ordbøger. *Paresis*. URL: <https://ordbog.gyldental.dk/#/pages/result/daklin/paresis/expert>.
- [15] Oxford English Dictionary. *Hemi*. URL: <http://www.oed.com/view/Entry/85797?redirectedFrom=hemi#eid>.
- [16] National Stroke Association. *Rehabilitation Therapy after a Stroke*. URL: <http://www.stroke.org/we-can-help/stroke-survivors/just-experienced-stroke/rehab>.
- [17] Mayo Clinic Staff. *Stroke rehabilitation: What to expect as you recover*. June 2014. URL: <http://www.mayoclinic.org/diseases-conditions/stroke/in-depth/stroke-rehabilitation/art-20045172?pg=1>.
- [18] Christina Rostrup Kruuse. *Apopleksi, rehabilitering*. Web. Apr. 2015. URL: <https://www.sundhed.dk/borger/patienthaandbogen/hjerte-og-blodkar/sygdomme/apopleksi/apopleksi-rehabilitering/>.
- [19] World Health Organization. *Rehabilitation - Definition*. URL: <http://www.who.int/topics/rehabilitation/en/>.
- [20] AHA/ASA. *Guidelines for Adult Stroke Rehabilitation and Recovery - A Guideline for Healthcare Professionals From the American Heart Association/American Stroke Association*. Tech. rep. American Heart Association, Inc., May 2016. DOI: 10.1161/STR.0000000000000098. URL: <http://stroke.ahajournals.org/content/47/6/e98>.
- [21] Michael Kent. *The Oxford Dictionary of Sports Science & Medicine*. Oxford University Press, 2006. URL: <http://www.oxfordreference.com/view/10.1093/acref/9780198568506.001.0001/acref-9780198568506-e-3771>.
- [22] Kåre Severinsen. "Fysisk træning af patienter med kroniske følger efter apopleksi". In: (2008).
- [23] Linda Sundekilde. *Effekten af CIMT/mCIMT på hemiparese i overekstremiteten hos børn og unge 0-15 år med svær erhvervet hjerneskade*. Web. Jan. 2014. URL: <https://fysio.dk/fafo/faglige-anbefalinger/critically-appraised-topics/effekten-af-cimtmcmmt-pa-hemiparese-i-overekstremiteten-hos-born-og-unge-0-15-ar-med-svar-erhvervet-hjerneskade/>.

- [24] Sundhedsstyrelsen. *National klinisk retningslinje for fysioterapi og ergoterapi til Voksne med nedsat funktionsevne som følge af erhvervet hjerneskade, herunder apopleksi*. Tech. rep. Sundhedsstyrelsen, Nov. 2014. URL: <https://sundhedsstyrelsen.dk/da/udgivelser/2014/~/media/F3A5AAE7542049FE8854C25109E40D1C.ashx>.
- [25] Marco Guidali et al. "A robotic system to train activities of daily living in a virtual environment". In: *Medical & Biological Engineering & Computing* (2011) 49 (Oct. 2011), pp. 1213–1223. doi: 10.1007/s11517-011-0809-0.
- [26] David J. Reinkensmeyer, Brian D. Schmit, and William Z. Rymer. "Assessment of Active and Passive Restraint During Guided Reaching After Chronic Brain Injury". In: *Annals of Biomedical Engineering* 27 (1999), pp. 805–814. doi: 10.1114/1.233.
- [27] C.T Freeman. "Newton-method based iterative learning control for robot-assisted rehabilitation using FES". In: (May 2014).
- [28] LOKOMAT - FUNCTIONAL ROBOTIC GAIT THERAPY. URL: <https://www.hocoma.com/world/en/products/lokomat/>.
- [29] Verena Klamroth-Marganska et al. "Three-dimensional, task-specific robot therapy of the arm after stroke: a multicentre, parallel-group randomised trial". In: *The Lancet Neurology* (Dec. 2013), pp. 159–166. doi: [http://dx.doi.org/10.1016/S1474-4422\(13\)70305-3](http://dx.doi.org/10.1016/S1474-4422(13)70305-3). URL: http://ac.cdn.com/S1474442213703053/1-s2.0-S1474442213703053-main.pdf?_tid=047ff9cc-abf9-11e6-ad11-00000aab0f6c&acdnat=1479299849_3965ce5f212b845d511a04612f5b823f.
- [30] Groppa S et al. "A practical guide to diagnostic transcranial magnetic stimulation: Report of an IFCN committee". In: *Clinical Neurophysiology* (May 2012). doi: 10.1016/j.clinph.2012.01.010.
- [31] David J. Gladstone, Cynthia J. Danells, and Sandra E. Black. "The Fugl-Meyer Assessment of Motor Recovery after Stroke: A Critical Review of Its Measurement Properties". In: *Neurorehabilitation & Neural Repair* (Sept. 2002). doi: 10.1177/154596802401105171.
- [32] Lisa Zeltzer. *FUGL-MEYER ASSESSMENT OF SENSORIMOTOR RECOVERY AFTER STROKE (FMA)*. 2010. URL: http://www.strokengine.ca/indepth/fma_indepth/.
- [33] Patricia Staubli et al. "Effects of intensive arm training with the rehabilitation robot ARMin II in chronic stroke patients: four single-cases". In: *Journal of NeuroEngineering and Rehabilitation* (Dec. 2009). doi: 10.1186/1743-0003-6-46.

- [34] Oscar Sandoval-Gonzalez1 et al. "Design and Development of a Hand Exoskeleton Robot for Active and Passive Rehabilitation". In: *International Journal of Advanced Robotic Systems* (Feb. 2016). doi: 10.5772/62404.
- [35] Balasubramanian S, Klein J, and Burdet E. "Robot-assisted rehabilitation of hand function." In: (Dec. 2010). doi: 10.1097/WCO.0b013e32833e99a4.
- [36] Ann M. Simon, Brian M. Kelly, and Daniel P. Ferris. "Preliminary trial of symmetry-based resistance in individuals with post-stroke hemiparesis". In: (Sept. 2009).
- [37] John V Basmajian and Carlo J De Luca. *Muscles Alive: Their functions revealed by electromyography*. Fifth Edition. Williams & Wilkens, Baltimore, June 1985. ISBN: 978-0683004144.
- [38] Peter Konrad. *The ABC of EMG*. Noraxon U.S.A Inc., Mar. 2006. ISBN: 0-9771622-1-4.
- [39] *Kinetis K22F MCU Family Block Diagram*. URL: http://www.nxp.com/webapp/sps/components/hot.jsp?image=http://cache.nxp.com/files/graphic/block_diagram/products/microcontrollers/K22F-BD.jpg&code=K22_120&imgType=ProductBlock&fasp=undefined&fpfsp=true.
- [40] MX-28T / MX-28R. URL: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-28.htm.
- [41] MX-64T / MX-64R. URL: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-64.htm.
- [42] MX-106T / MX-106R. URL: http://support.robotis.com/en/product/actuator/dynamixel/mx_series/mx-106.htm.
- [43] *Trossen Robotics Dynamixel Guide*. URL: <http://learn.trossenrobotics.com/projects/159-trossen-robotics-dynamixel-guide.html>.
- [44] *6 Port RX/EX Power Hub*. URL: <http://www.trossenrobotics.com/6-port-rx-power-hub>.
- [45] Yutaka Yamamoto. "CONTROL SYSTEMS ARE UBIQUITOUS". In: (2013). URL: <http://www.ieeecss.org/control-systems-are-ubiquitous-2016>.
- [46] Mituhiko Araki. "PID Control". In: *CONTROL SYSTEMS, ROBOTICS, AND AUTOMATION*. Ed. by Heinz Unbehauen. Eolss, 2009, pp. 58–79.
- [47] Kiam Heong Ang, G. Chong, and Yun Li. "PID control system analysis, design, and technology". In: *IEEE Transactions on Control Systems Technology* (Volume: 13, Issue: 4, July 2005). IEEE, June 2005, pp. 559–576. doi: 10.1109/TCST.2005.847331. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1453566&tag=1>.
- [48] Aidan O'Dwyer. *Handbook of Pi and Pid Controller Tuning Rules*. 2nd ed. World Scientific Publishing Company, Feb. 2006. ISBN: 9781860949104.

- [49] Tim Wescott. *PID without a PhD*. EE Times-India. Oct. 2000. URL: <http://manuals.chudov.com/Servo-Tuning/PID-without-a-PhD.pdf>.
- [50] *ArbotiX-M Hardware Overview*. URL: <http://learn.trossenrobotics.com/arbotix/arbotix-getting-started/38-arbotix-m-hardware-overview.html#panel11-1>.
- [51] *+3.3V Low Power Half-Duplex RS-485 Transceivers with 10Mbps Data Rate*. URL: <https://www.sparkfun.com/datasheets/Components/General/sp3485CN-LTR.pdf>.
- [52] *UartSBee V4 (USB-Xbee-TTL Interface)*. URL: <http://www.trossenrobotics.com/uartsbee>.