Human controlled robotic arm

Improving usability with haptic feedback

SIMON AHLIN HÖGFELDT

DANIEL SÖDERMAN
Human controlled robotic arm

SIMON AHLIN HÖGFELDT
DANIEL SÖDERMAN

Bachelor’s Thesis in Mechatronics
Supervisor: Fariba Rahimi
Examiner: Martin Edin Grimheden
Approved: 2016-06-07

TRITA MMKB 2016:25 MDAB086
Abstract

In our modern society, the usage of robotic arms are increasing. Much of the work in the industry is now done by robots. Even though they are able to do very precise work, difficulties appear when trying to do some of the tasks that humans do. This can be changed by making it easy for a human to control the robotic arm and to "teach it" how it's done.

The purpose of this project is to develop a robotic arm that is easily controlled by the user. This is done by using the user's own arm movement to control the robot. To make the usage more intuitive, a simple haptic feedback system will be implemented. This creates a greater experience where the user is able to "feel what the robot feels". To be able to create such a system, development of an easy control unit, robotic arm and feedback system has to be made. The steering of the robotic arm is created from reading the user’s arm movement with potentiometers, and mapping these values to servos on the robotic arm. Sensors on the robotic claw and on the user’s hand makes it possible for a DC motor to create a counter torque that gives a feel for the size of the object being lifted.

The feedback system seemed to improve the overall experience of using a robotic arm and with further work, today’s robotic industry could be improved.
Sammanfattning

Robotarm styrd av människans arm


Feedbacksystemet verkade förbättra användandet av robotarmen. Med framtida utveckling av detta projekt skulle framtidens robotindustri kunna förbättras avsevärt.
Preface

We would like to thank our class mates for all the helpful and valuable discussions. Staffan Qvarnström provided us with the electronic parts needed. Tomas Östberg for providing us with the tools needed for the construction. Fariba Rahimi for help with the project and PID control.

Daniel Söderman, Simon Ahlin Högfeldt,
Stockholm, May, 2016
Contents

Abstract iii

Sammanfattning v

Preface vii

Contents ix

Nomenclature xi

1 Introduction 1
  1.1 Background ................................................. 1
  1.2 Purpose .................................................. 2
  1.3 Scope .................................................. 2
  1.4 Method ................................................ 3
    1.4.1 Haptic feedback ....................................... 3
    1.4.2 Robotic arm .......................................... 3
    1.4.3 Arm unit ........................................... 4
    1.4.4 Measuring the real step response ................. 4
    1.4.5 Application testing .................................. 4

2 Theory 5
  2.1 Past Research ........................................... 5
  2.2 PID-control .............................................. 5

3 Demonstrator 7
  3.1 Problem formulation ....................................... 7
  3.2 Hardware ................................................ 8
    3.2.1 Chosen parts ......................................... 8
    3.2.2 Robotic arm .......................................... 8
    3.2.3 Arm unit ........................................... 11
  3.3 Electronics .............................................. 14
  3.4 Software ................................................ 16
    3.4.1 PID ................................................ 16
3.5 Measuring step response ........................................ 17
3.6 Application testing .................................................. 19
3.7 Results .................................................................. 19
   3.7.1 PID ................................................................. 20
   3.7.2 Application testing ............................................. 22

4 Discussion and conclusions ........................................... 23
   4.1 Discussion ............................................................ 23
   4.2 Conclusions .......................................................... 25

5 Recommendations and future work ................................. 27
   5.1 Recommendations .................................................. 27
   5.2 Future work ........................................................... 27

Bibliography ................................................................. 29

Appendices

A Datasheets ................................................................. 31
B Schematics ................................................................. 37
C Arduino Code ............................................................. 39
D Matlab Code .............................................................. 49
E Plots .................................................................. 57
Nomenclature

Symbols

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>Proportional part parameter</td>
</tr>
<tr>
<td>$I$</td>
<td>Integral part parameter</td>
</tr>
<tr>
<td>$D$</td>
<td>Derivative part parameter</td>
</tr>
<tr>
<td>$J$</td>
<td>Rotor inertia [$kg \cdot m^2$]</td>
</tr>
<tr>
<td>$b$</td>
<td>Motor viscous friction constant [$N \cdot m \cdot s$]</td>
</tr>
<tr>
<td>$K$</td>
<td>Torque constant [$Nm/A$]</td>
</tr>
<tr>
<td>$R$</td>
<td>Electric resistance [$\Omega$]</td>
</tr>
<tr>
<td>$L$</td>
<td>Electric inductance [$H$]</td>
</tr>
<tr>
<td>$G_{motor}$</td>
<td>Transfer function for DC motor</td>
</tr>
<tr>
<td>$F$</td>
<td>Transfer function for controller</td>
</tr>
</tbody>
</table>

Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSR</td>
<td>Force Sensitive Resistor</td>
</tr>
<tr>
<td>PID controller</td>
<td>Proportional–Integral–Derivative controller</td>
</tr>
<tr>
<td>DC motor</td>
<td>Direct Current motor</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

In our modern society, the development of technological solutions has been in great focus. Robotic arms has revolutionized the industry by making it more cost efficient, improving build quality, decreasing production time. [Keystone Electronics Corp, 2013] In the future, robotic arms will be even more important in manufacturing. [HAGERTY, 2015] At BMW there are robots doing the big jobs of fastening the body panels and doing the welding. However what remains is to use those robots on the assembly line where workers assemble the more complex parts. [IndustryWeek, 2012] Robotic arms are improving and the precision required for assembly may be achieved. But to control the movement of the robotic arm, an engineer is often required to program its software. A way to make robotic arms easier to control is to allow the robotic arm to be controlled by the user’s own arm.

You could almost say that we have mastered the technique of controlling our own arm movement. Because a robotic arm’s construction is similar to the human arm, it makes sense for the user to control it with her own arm. This would make it easy to understand its movement limitations.

When you move your own arm you get feedback as to what the arm is experiencing. Without some type of feedback the feeling of controlling the robotic arm is diminished. For full control of the robotic arm some type of force feedback would be helpful.

Research into robotic arms with feedback can often be seen in the surgical industry. However these are often built with a narrow purpose in mind and are also very expensive since surgical precision is needed. [Okamura, 2009] This project explores cheaper and simpler options that don’t have to follow the same standard as the medical industry.
CHAPTER 1. INTRODUCTION

1.2 Purpose

The purpose of this project is to improve the application of a standard robotic arm by implementing a control of the robotic arm with an arm unit that follows your arm’s movement. Simple haptic feedback will be implemented that transmits the force that the robotic arm is feeling back to your own hand that is connected to the arm unit. It should be possible for a person without any knowledge about robotic control to be able to control the robotic arm’s movement. This means that the controller should be made in an intuitive way i.e controlling it with your own arm. The haptic feedback will be constructed using a force sensor on the robotic arm’s claw and one on the arm unit. By implementing a motor on the arm unit, the force that the user feels in their hand can be altered to be equal to the force that is measured at the gripping claw. This requires an optimized PID controller. The system leads to the interesting question on how the real system differs from the theoretical calculations. Therefore, comparison of the step response for the real system and the one in theory will be done.

The research questions of this study are:

- "Which PID control parameters fits better to the designed system with feedback?"
- "How does the feedback system differ to the simulated model?"
- "How will the implemented haptic feedback affect the application of the robotic arm?"

1.3 Scope

This report covers a semester work for a Bachelors thesis at KTH with limited time and resources. Due to this, there must be some limitations to the construction. Focus will be on the haptic feedback and its functionality. The construction of the robotic arm and arm unit will therefore be as simplified as possible. The main goal for the robotic arm is to be able to grip an object and move it. The weight of the object and the distance is not important as this does not affect the main purpose of this project. Limitations of the robotic arm and arm unit is set to three degrees of freedom as shown in Figure 1.1. The feedback system will affect the whole hand and will not be different for the separate fingers. This means that the robotic claw will be a single flat surface, corresponding to a human hand with all their fingers closed together. The robotic arm will only be able to grab objects with simple rectangular shapes. When calculating the transfer function for the PID controller, some frictions will be neglected. Due to the fact that the purpose of the units are for scientific testing, the appearance and practical use is not in focus.
1.4 Method

1.4.1 Haptic feedback

The main goal of the design is that the user should be able to feel when the robotic arm grabs an object. A general feel of the object size should also be possible. For this, the robotic arm should be able to register the normal force from the object being grabbed. The same force should occur on the user’s hand. The arm unit therefore needs to register the force between the user’s hand and the unit. This is done by using a FSR on the robotic arm’s claw and one FSR on the arm unit. For the user to be able to feel the same force as the robotic arm, the signal from each FSR should be the same.

1.4.2 Robotic arm

The robotic arm is constructed with some simplifications. Most of its parts are 3D printed using a Ultimaker. The less complex parts are created in wood using a laser printer Epson. For reducing the errors during the construction, most parts has been modeled using Solidedge. To make the controlling of the arm as intuitive as possible, the robotic arm should have the same limitations as a human arm, i.e the same degrees of freedom. The movement of the robotic arm is created using three servos, one for each degree of freedom, see Figure 1.1. The claw will be built as plates, therefore it will be hard to grab circular objects.
1.4.3 Arm unit

Because of its simple design, the arm unit is mostly created in wood using a laser printer Epson [Epilog Laser, 2016]. The movement is registered using three potentiometers, one for each degree of freedom. These should correspond to the same degrees as on the robotic arm, see Figure 1.1. Using the signal from the potentiometers, the angles can be calculated. Knowing this angle, the same angle can be sent to the servo on the robotic arm. A DC motor is used to create the counter torque needed on the user’s hand. This torque makes the user feel a force, thus the value of the FSR will increase. This enables us to create a simple feedback system, where the signal to the motor is controlled to give different torques depending on how hard the robotic arm is grabbing an object.

1.4.4 Measuring the real step response

To get a measurement of the real step response, a real test was made. The test was conducted by having the user’s hand connected to the arm unit. The robotic arm was then given a predefined force. The feedback system will then try to get the value of the FSR on the arm unit to be the same as on the robotic arm. This is done via the DC motor mounted on the arm unit. By saving the values of the arm unit’s FSR during this procedure, the step response of the real system could be plotted.

1.4.5 Application testing

An interesting aspect of the haptic feedback system is how it affects the application for the user. To answer this question, a small amount of test subjects did multiple predefined tasks and then answered some questions about the application with the feedback system.
Chapter 2

Theory

2.1 Past Research

There have been projects where the robotic arm is controlled by the human arm. One uses accelerometers to register the movement of the human arm. [PATTNAIK, 2013]. Another solution is to use gyroscopes and bendable potentiometers [bluebean, 2012]. These methods are unnecessarily complex where some sort of filter is needed. The signal from a potentiometer is stable and will provide for easier control with less fluctuations. Therefore simple potentiometers will be used for registering the movement of the user.

2.2 PID-control

For a system to be able to read a reference value and then do something to make the system get to that value, the system needs to continuously calculate an error value. This is the difference between the desired value and the measured value. The system will then try to minimize this error over time by adjusting the system. The PID controller is made up of three parts. The proportional (P) part corresponds to the gain of the controller. If this part is increased the system gets a faster response at the cost of stability. Using only a P-controller has some drawbacks. The error will always be greater than zero. For a fast system (where the proportional constant value is high) the system will get large overshoots. Overshoot is when the system exceeds its final steady-state value. To overcome this problem, an integral part (I) needs to be implemented. Increasing the I part yields a smaller steady state error but once again at the cost of reduced stability. Adding a derivative part (D) part is used for better stability. An increased D gives greater stability. [Glad and Ljung, 2006] To apply this controller to a DC motor, the transfer function from voltage to speed in the laplace domain must be known. This is received from [University of Michigan, 2016]
\( G_{motor} = \frac{K}{((J\cdot s + b)\cdot(L\cdot s + R) + K^2)}. \) \hspace{1cm} (2.1)

Equation [2.1] does not consider friction forces. It only consider the DC motor. The system is now mathematically expressed and can be simulated using MATLAB Simulink, [MathWorks Nordic, 2015b]. This makes the closed loop system

\[
\frac{F \cdot G_{motor}}{1 + F \cdot G_{motor}}. \tag{2.2}
\]

Where \( F \) is the PI or PID controller in laplace domain. The PI controller is written in laplace domain as

\[
F = \frac{Ps + I}{s}. \tag{2.3}
\]

The PID controller can be written as

\[
F = \frac{Ps +Ds^2 + I}{s}. \tag{2.4}
\]

For tuning the controller, pole placement is used. This is done by placing the poles of the denominator of (2.2) and solving for the PI, and D part. The closed loop system will be of second order and have two poles. Which gives the following equation for the P,I,and D parameters

\[
(s + pole)^2 = 1 + F \cdot G_{motor}. \tag{2.5}
\]

Note that one or more of the parameters P, I and D can be set to zero. This gives different controllers, such as P, PI, PD and PID. Using the step response, the parameters of the controller can be tuned to give a desirable behavior.
Chapter 3

Demonstrator

3.1 Problem formulation

The robotic system can be divided into two major parts, the arm unit and the robotic arm. The arm unit will read the human arm movement. The robotic arm is the robot itself, which will do the movement given by the user. Each system requires a number of technical solutions. The requirements for the arm unit are:

- Easy to fasten user’s arm.
- Intuitive to use.
- Read the movement with potentiometers.
- Read the normal force created from a motor.
- Transfer torque from DC motor to user’s hand.

The requirements for the robotic arm are:

- Movement created by servos.
- Create same movement as read by the arm unit.
- Able to lift and move objects.
- Read the normal force of grabbed object.

To be able to create the correct counter torque from the motor on the user’s hand, a PID controller needs to be implemented. The problem to be solved is to optimize the values of the controller’s constants, but also to see what kind of controller is best for our purpose. To be able to optimize the PID, a set of requirements needs to be decided for the system. The requirements for the PID controller are:

- Stable system.
- Small to no overshoot.
- Settling time less than 0.5 seconds.
3.2 Hardware

3.2.1 Chosen parts

These are the chosen parts for our prototype.

- 3 Servos (Parallax Standard Servo 900-00005)
- 1 Motor (Maxon motor 110147 with gears 232766)
- H-bridge (L298)
- 2 FSR (Interlink 402)
- 3 Potentiometers 10KΩ (PIHER PC16SH10IP06-103A2020–TA)

When deciding which servos to buy, small servos was preferred, while being strong enough for lifting objects. The robot should be able to lift 0.5 kilograms. The servo’s strength is given in torque. To choose an appropriate servo, a simple calculation for the required torque was made. The largest strain is for the lifting movement. The distance from the servos rotational center point to the end of the gripping claw is approximately 0.35m. The weight of the arm and claw itself is approximately 0.2kg. Lifting a weight of 0.5kg would require a torque of approximately

\[(0.5 + 0.2) \cdot 0.35 = 0.245 \text{Nm}.\]  

As mentioned above, the chosen servo should be able to create a torque of approximately 0.26 Nm.

For the haptic feedback a DC motor was chosen because it is easy to control speed and torque compared to servos or stepper motors. When choosing the motor, the strength and required speed was unknown. The feedback system was built and the DC motor was chosen with trial and error. One important aspect was that the motor should not be too heavy. This would make the whole arm unit heavy and hard to use for the user. The motor mentioned above was both strong and fast enough to fulfill the requirements for the feedback system and was of a reasonable size.

The H-bridge was chosen with simplicity in mind. The component had to be easy to use, cheap and available. The purpose is to be able to control a DC motor.

The FSR needed to be cheap but reliable with good accuracy. The chosen FSR fulfilled these requirements. They were also easy to fastened.

Potentiometers were chosen to track the arm movement because they provide a logical connection with the angles of the servos. The resistance of 10KΩ is chosen to provide a more stable signal.

3.2.2 Robotic arm

The construction of the robotic arm uses a combination of 3D printing and laser cutting. The base is 3D printed to fit one servo. Its main purpose is to rotate the
3.2. HARDWARE

whole robotic arm. Wooden feet were laser cut and mounted to the base, see Figure 3.1.

Figure 3.1. The base of the robotic arm. Servo is mounted and fastened with four bolts.

The servo mounted in the base is connected to another housing. The torque is transferred via a connection to the servo arm. This is the part being horizontally rotated, see Figure 3.2.

Figure 3.2. The rotational part of the robotic arm mounted on the base. Torque is transferred via a tight fitting shape around the servo arm.

The second servo takes care of the second degree of freedom, which is lifting the arm vertically. This servo is mounted in the mentioned housing, see Figure 3.3. On this part, two wooden arms are connected via an attachment part. This part is a 3D printed U-shaped part that connects the two wooden arms to the servo. A similar connection to the servo arm is implemented for transferring its torque.
CHAPTER 3. DEMONSTRATOR

Figure 3.3. The arm part is mounted via a 3D printed U-shape connector. This connects to the servo and transfers the torque. The servo is fastened with four bolts.

The wooden arm parts are attached to a 3D printed claw connector. The third servo is fastened underneath this connector, see Figure 3.4. The claws are 3D printed with the gears designed to create movement in both claw parts with only one servo. For a claw with a bigger surface area, sheets of wood were laser cut as the final piece of the claw for better grip. A FSR is mounted on one of the wooden pieces. This will sense the force on the robotic arm.

Figure 3.4. The claw connected to the servo. It is fastened to the servo with a screw. The rotational movement transfers to the other claw via gears. The claws are 3D printed. The gripping part is made out of wooden pieces. A FSR is mounted on one of the plates.

The complete robotic arm, see Figure 3.5 is a result of solutions for transferring the torques from the three servos. It uses a total of three servos, one for each degree of freedom, see Figure 3.6.
3.2. HARDWARE

Figure 3.5. The final version of the robotic arm

Figure 3.6. The movements of the robotic arm with three degrees of freedom

3.2.3 Arm unit

The arm unit is constructed by sheet wood which has been laser cut using CAD designs. There are three major parts, as seen in Figure 3.7. One part for each degree of freedom with potentiometers paired with each servo.
CHAPTER 3. DEMONSTRATOR

The function of the base is to follow the user’s horizontal rotation of the arm. This is registered by a potentiometer. Its shaft is mounted to the rotational plate. The potentiometer itself is fastened to another plate underneath the rotational plate, see Figure 3.8. For stability this part has circular stands attached.

The user’s arm is fastened to a long wooden piece that connects the rotational base to the hand part. This piece is connected to a potentiometer that reads the rotation from lifting the hand, see Figure 3.9.
3.2. HARDWARE

Figure 3.9. The potentiometer is fastened via a bend metal piece. It registers the rotation of the vertical movement of the arm.

The hand part is a complex system that was built as the project progressed. The hand piece has two major functions. It measures how the user opens and closes her hand and transfers the torque produced by a mounted DC motor. The measuring of the user’s hand movement is done with a potentiometer, see Figure 3.10.

Figure 3.10. The potentiometer is fastened to the wooden piece at the bottom. This piece is fastened to a metal plate which is fastened to the rigid part of the hand unit. A similar part is connected to the potentiometer shaft, but connected to the outer part of the hand unit.

The haptic feedback is constructed by fastening the DC motor to the rigid part of the hand. The motor shaft is coupled with the closing part of the hand. This is done by laser cutting a part that fits around the shaft to transfer torque. Here acrylic glass, as shown in Figure 3.11, is used since sheet wood is too soft and will
not maintain the shape around the shaft when torque is applied. When the DC motor applies a torque, the hand unit will open, thus giving a counter force to the user that is trying to close her hand.

![Figure 3.11. The feedback construction with acrylic glass.](image1)

A FSR is mounted to the hand unit, see Figure 3.12. This reads the force on the user’s hand produced by a torque from the DC motor.

![Figure 3.12. A FSR is fastened to the outer part of the hand unit. A finger of the user is placed on the sensor. This registered the force on the hand produced from the torque in the DC motor.](image2)

### 3.3 Electronics

The entire system is run by an Arduino Uno. This is a micro controller that is controlled by a given code. The system uses three smaller PCBs, see
3.3. ELECTRONICS

Figure 3.13. The purpose for this was to try and make the connections intuitive, where one PCB connects all the parts for the robotic arm and another for the arm unit. GND and 5V from each PCB are then connected to a third PCB that contains a button and a LED. This card has two functions. It connects all the 5V and GND, which then goes to the Arduino Uno. The button and LED will give the user the option to toggle the feedback system ON and OFF. When the LED is turned ON, the feedback system is ON. The button requires a pull-down resistor, in this case, 10KΩ is used. For the LED a 330Ω resistor is used.

Analog potentiometers are used for reading the position of the users arm. Through the software, the angle is mapped and fed to the servo motors via the Arduino Uno. The servos are powered by an external source to provide enough power. According to the datasheet for the servos, see Appendix A, maximum voltage was used, around 9V.

To implement the PID control, two force sensitive sensors are used. To work, they need a pull-down resistor. A 10KΩ resistor was used for each sensor. The reference value is received from the sensor on the robotic arm. The output value is received from the sensor on the hand unit. The PID controller regulates the PWM signal to the DC motor which corresponds to a force at the output sensor. The DC motor is connected via a H-bridge to the Arduino Uno. The DC motor needs a separate voltage source. According to the datasheet for the DC motor, see Appendix A, the DC motor needs to be connected to 15V. For schematics of the system, see appendix A.

Figure 3.13. Connection diagram for the system
3.4 Software

The Arduino Uno reads the values from the potentiometers. These values are mapped so the angle of the servo corresponds to the same angle as the potentiometer. This means that the servos will follow the same angle movement as the input from the potentiometers, see Figure 3.14. The values from the two FSRs are measured by the Arduino Uno. The value from the robotic arm is seen as the setpoint signal while the value from the arm unit is the output. Using the PID library [Beauregard, 2016] it calculates the required signal to the DC motor via the H-bridge. The state of the button controls whether or not the PID State is ON which turns the haptic feedback on and off. The LED is also turned on or off depending on the PID state.

![Program flowchart](image)

**Figure 3.14.** Program flowchart

### 3.4.1 PID

When calculating the step response for the feedback system, MATLAB Simulink is used. When inserting values for the DC motor given by the datasheet, see Appendix A, into Equation 2.1, we get the closed loop system as shown in Figure 3.15. By inserting different values for the parameters P, I and D, the different step response
3.5. MEASURING STEP RESPONSE

for each calculated system can be plotted. By inspecting these plots, it’s possible to see if the calculated values fulfills the requirements given in chapter 3.1 Problem formulation.

Figure 3.15. Closed loop for the feedback system.

3.5 Measuring step response

To evaluate the response of the feedback system, a simple measuring method was implemented. The main idea was to apply a force to the robotic claw and measure how the feedback system responds to the given force. A hand was placed in the arm unit and force was applied to the robotic claw. The feedback system would then make the FSR on the arm unit go from zero to the setpoint given by the force applied to the robotic arm. A problem with this method was that the arm unit would start twitching. This was because of the user could not keep the hand steady. A solution to this problem was to start the measuring from a small force instead of starting from no force. By doing this, the twitches were reduced a lot. The changed test method was therefore:

- Apply small force to the robotic claw.
- Let this value be the start point. For the step function, set this value as the zero point.
- Rapidly increase the force and try to keep the force as constant as possible.
- Measure the values of the FSR on the arm unit.

The code for the Arduino Uno was modified to a code for the measuring, see Appendix C for code and Figure 3.16 for flowchart diagram. This code began the measuring when the FSR on the robotic claw was given a force signal larger than 350. When measuring, values from the FSR of the arm unit were saved and then printed when the measuring was finished. The test were done ten times.
The printed values were then imported into a MATLAB script for further calculations, see Appendix D. This script scaled the values so the start values were set to zero and the reference were set to one. Plotting of the step response for each test was made, but also a plot with the averaged step response.

Figure 3.16. Flowchart for measuring the step response.
3.6 Application testing

A small amount of randomly picked people tried the robotic arm. They had some time to feel different sized objects before the real testing began. The testing was done in two steps. The first was to feel two different objects while having their eyes shut. The test subject would then try to guess what object was the largest of the two. Different objects were used with the main focus to have two objects with large difference in size and two objects with a small difference in size. One of the tests was with the same object twice to see if they noticed it. This test was made multiple times, where two of the times, the difference in size was big, two times small and one time without difference.

Next test was to move an object once with the feedback turned ON and once with the feedback turned OFF. This was done with their eyes open. After the test the people had to answer the following questions:

1. Did the feedback system make the task easier?
2. Did the feedback system improve the experience?

The meaning of the second question is to see if the user feels like there is a better experience when being able to somewhat feel what the robotic arm is feeling.

Some weeks after the application testing was done, then idea of being able to feel objects with different softness was brought up. This test was done by grabbing a hard object and then a soft foam ball to see if a difference in feel could be noted.

3.7 Results

The robotic arm and the arm unit were placed on a wooden piece. This made the whole test station more compact and easy to move, see Figure 3.17.
3.7.1 PID

The theoretically calculated step response was made in MATLAB Simulink. Using pole placement calculations, different values for the parameters P, I and D were received. The overall best step response was found for a pole at \( s = -90 \), see Figure 3.18. This system does not use the D part, but fulfills the given requirements, see chapter 3.1 Problem formulation. The system has a very small overshoot with short settling time and almost no oscillations. Comparing this step response to the ones calculated with the pole placed at other positions did not give as good results, see Appendix E. The controllers using a D part did not improve the step response. As mentioned, this did not give as good result as for the PI controller with the pole at \( s = -90 \). The best result for a PID was for the one given in Figure 3.19.

\[ P = 0.0536, \ I = 2.4215 \]

**Figure 3.18.** Step response function with Simulink for the pole \( s = -90 \). The values are: \( P = 0.0536, I = 2.4215 \)
3.7. RESULTS

Figure 3.19. Step response function with Simulink for the pole \( s = -0.001 \). The values are: \( P = 1.4387, I = 0.0072, D = 71.9424 \). This system is very fast and has no overshoot. The problem with this system is that it has a constant error.

When measuring the step response for the real system, we got the results seen in Figure 3.20. The system is much slower than the theoretically calculated system.

Figure 3.20. Step response from measuring the real system. The measuring was done ten times. Each blue plot corresponds to one measurement. The red plot corresponds to a step function.

These measurements was then averaged to provide a more reliable result as seen
in Figure 3.21. This step response reduces the errors from the measuring itself.

![Step response function for real system (mean value)](image)

Figure 3.21. Step response function from measuring the feedback system. This is the mean value of all measured values. The red plot corresponds to a step function.

### 3.7.2 Application testing

The results from having a small amount of people do some application testing is given in Table 3.1. Every test subject could feel the difference of object size when the two objects differs a lot in size.

Big and small refers to the difference between the size of the grabbed objects. Big means that the two objects being grabbed has a big difference in size. Small means the opposite. No refers to having the subject grabbing the same object twice. The number '1' means that the answer is yes and '0' means no. For columns 'Big','Small' and 'No', '1' means that the test subject felt the difference and correctly guessed what object was the largest.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Big</th>
<th>Small</th>
<th>Small</th>
<th>No</th>
<th>Big</th>
<th>Does this make the task easier?</th>
<th>Does this improve the experience?</th>
<th>Questions</th>
<th>Result</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Big</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>Person 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Small</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>Person 3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>No</td>
<td>3</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>Person 4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Task-easier?</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Person 5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Task-experience?</td>
<td>5</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1. Test results from having test subjects using the robotic arm.
Chapter 4

Discussion and conclusions

4.1 Discussion

The servos did not seem to be able to create the torque given by the manufacturer. While it should be able to lift a weight of 0.5kg without any problem, it could hardly lift itself without added weight. This made us shorten the arm as much as possible (approximately 0.2m). This would require a torque around 0.14Nm. The robotic arm is still very weak and can hardly lift 0.1kg. Because of the design of the robotic arm mostly being created in CAD and 3D printed, buying a larger servo would require us to re-design most of the CAD models and 3D print them all again. This would be very time consuming. Because of the research question not being so dependent of the lifting part, we decided to leave the model as it was and focus on the feedback system itself.

The best step response was for a PI controller with poles at $s = -90$. With a very fast settling time, almost no overshoot, and very small oscillations, this system fulfilled our requirements from chapter 3.1. However with PID controller we get an error that is too large and also oscillations. This is probably due to the I part being too small. But with pole placement the parameters of the PID controller can’t be individually tuned which means that the inclusion of a D part causes the I part to shrink.

The response function of the real system differed a lot from the theoretical calculation that was made. As can be seen in Figure 3.21, the plot corresponds to a step response, but is much slower than the theoretical as can be seen in Figure 3.18. There are many reasons to why this is the case. When calculating the transfer function, friction and inertia of the construction were neglected. The transfer function only takes the DC motor into account. The test method for calculating the real system’s step response is also a great factor of why the result differs so much from the theoretical calculations. The measuring was done mostly by 'feel' and each test could differ a lot. The start and end value of the FSR of the robotic hand differed for each test. This would affect the amount of the error, thus the time would differ. When testing, ones hand was placed in the arm unit. Because of the human factor,
it was very difficult to be able to have a constant reaction to the torque from the motor. Also, to get the reference value, the FSR value on the robotic arm was printed. This value was equal to the value of the FSR at the end of the test. This means that the value could have differed a lot during the test. By doing several tests and calculating the mean value, we were able to reduce the error of the measuring technique. For more accurate results, a test rig should have been built, to minimize the factors which are affecting the result. In hindsight, it would have been much better if we had inserted a value to the FSR on the robotic arm, instead of applying the force ourselves. This would have removed the variance of the input FSR signal.

We had a very small amount of people to test the feedback system and answer some questions about its functionality. The reason of not having more people to test the unit was because of time limitations, but also that other areas felt more relevant. Even though the test data may not be sufficient, some interesting results may be considered. Every test subject were able to feel size difference in objects differing a lot in size, but also when the difference were not so large. However, when having the subjects grab the same objects twice, 60% thought that there is a difference in size. One reason for this may be that when testing, we might not have been clear enough that we would test the same object twice. Another aspect of this result is the construction itself. The connection between the DC motor and FSR with PID may lead to that the user’s hand closes a bit different every time grabbing an object. The system does only react to the reaction force from the grabbed object and not its actual size. If the user keep closing her hand when the counter torque is produced, the torque will increase but the hand will still be able to close. To be able to have a correct feeling of the object’s size, the user has to close the hand slowly and may have to decrease the force. 20% percent of the test subjects experienced that the moving task were easier with the feedback control. Perhaps this is because of the small distance between the user and the robotic arm. If the robotic arm were controlled from a greater distance via a screen, the result might have been different. Although the subjects did not feel that the design is making the task easier, 100% believed that it improved the experience i.e, the connection between the robot and the user, since they got a better feeling for what the robotic arm were grabbing.

When testing to grab objects with different softness, the difference was minimal. This is probably due to how the FSR work. It only registers the applied force. When grabbing the soft foam ball, the user were able to close her hand a bit more because of the compression of the ball. This made the user to believe that the object being grabbed is smaller.

The softness test was very simply executed. If more sensors were implemented on the users hand and the robotic arm, the chance of feeling different softness would have increased.
4.2 Conclusions

A PI controller with the values $P = 0.0536$ and $I = 2.4215$ seems to be best for our feedback system.

The PID system differed quite a lot from the simulated model. The reasons are many. We neglected some friction forces, but also the measuring of the feedback system had some drawbacks.

For improving the control of robotic arms, using a control method based on your own arm movement is a concept that seems to be beneficial. Controlling a robotic claw with haptic feedback where the user can get some kind of feeling for the size of the object being lifted, improves the application of controlling a robotic arm. A simple feedback system for robotic arms could be very beneficial for the future of robotics in the industry. When the user and robot are in different places this could improve the application a lot. Using potentiometer for registering the user’s arm movement and mapping this to servos seem to be simple but yet a good way of controlling a robotic arm. If the robotic arm would be of greater size, servos might not be the best solution.
Chapter 5

Recommendations and future work

5.1 Recommendations

We would recommend getting bigger and preferably higher quality servos for the robotic arm.

The measuring for the feedback system in reality should be done better. The input value for the FSR on the robotic arm should be inserted as an input value in the code rather than by pressing the FSR.

5.2 Future work

Since this unit was made for testing the feedback system, it fulfilled its requirements. For future testings, the robotic arm should be designed to fit stronger servos. This enables testing of moving larger objects.

The gripping claw could be improved by making it more similar to a hand with five fingers. This means having one FSR for each finger on the robotic arm and the arm unit. This would enable the testing of objects with different shapes.

One thing that we wanted to do was to be able to record the movement of the robotic arm. This would enable the user to do a task once, and the robotic arm would then be able to do the same task on its own. This leads to the fact that a user with no robotic experience could easily set up the robotic arm’s movement. This would be very beneficial to the industry, where you would not need an engineer to make a change in the settings for the robotic arm to do the new task.

The feedback system could be improved so that the user could be able to feel the softness of the object being grabbed. This may be done by building a gripping claw that has five fingers and five sensors. The arm unit would have one sensor and one dc motor for each finger. This might improve the aspect of being able to feel different softness.

In the future, a larger system could be built to be applied in industry.
Bibliography


Appendix A

Datasheets
### Datasheet for gear and DC Motor

**Planetary Gearhead GP 22 A Ø22 mm, 0.5 - 1.0 Nm, Metal Version**

Part number 134163

---

#### General Information

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gearhead type</td>
<td>DP</td>
</tr>
<tr>
<td>Outer diameter</td>
<td>Ø22 mm</td>
</tr>
<tr>
<td>Version</td>
<td>Standard version</td>
</tr>
</tbody>
</table>

#### Gearhead Data

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduction</td>
<td>53 : 1</td>
</tr>
<tr>
<td>Absolute reduction</td>
<td>3275/94</td>
</tr>
<tr>
<td>Max. motor shaft diameter</td>
<td>4 mm</td>
</tr>
<tr>
<td>Number of stages</td>
<td>3</td>
</tr>
<tr>
<td>Max. continuous torque</td>
<td>0.8 Nm</td>
</tr>
<tr>
<td>Max. intermittent torque</td>
<td>1.2 Nm</td>
</tr>
<tr>
<td>Direction of rotation, drive to output</td>
<td></td>
</tr>
<tr>
<td>Max. efficiency</td>
<td>99 %</td>
</tr>
<tr>
<td>Average backlash no load</td>
<td>1.6 g</td>
</tr>
<tr>
<td>Mass inertia</td>
<td>0.9 g/cm²</td>
</tr>
<tr>
<td>Gearhead length (L)</td>
<td>30.2 mm</td>
</tr>
<tr>
<td>Max. transmissible power (continuous)</td>
<td>5.0 W</td>
</tr>
<tr>
<td>Max. transmissible power (intermittent)</td>
<td>8.5 W</td>
</tr>
</tbody>
</table>

#### Technical Data

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radial play</td>
<td>max. 0.2 mm, 10 mm from flange</td>
</tr>
<tr>
<td>Radial play</td>
<td>0 - 0.2 mm</td>
</tr>
<tr>
<td>Max. radial load</td>
<td>55 N, 10 mm from flange</td>
</tr>
<tr>
<td>Max. axial load (dynamic)</td>
<td>100 N</td>
</tr>
<tr>
<td>Max. force for press fit</td>
<td>100 N</td>
</tr>
<tr>
<td>Max. continuous input speed</td>
<td>8000 rpm</td>
</tr>
<tr>
<td>Max. intermittent input speed</td>
<td>5000 rpm</td>
</tr>
<tr>
<td>Recommended temperature range</td>
<td>-20°C - 150°C</td>
</tr>
<tr>
<td>Extended temperature range</td>
<td>-40°C - 150°C</td>
</tr>
<tr>
<td>Number of backlash cycles</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Product

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>58 g</td>
</tr>
</tbody>
</table>
A-max 22 Ø22 mm, Precious Metal Brushes CLL, 5 Watt, with terminals
Part number 110123

Values at nominal voltage

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal voltage</td>
<td>15 V</td>
</tr>
<tr>
<td>No-load speed</td>
<td>12000 rpm</td>
</tr>
<tr>
<td>No-load current</td>
<td>12.1 mA</td>
</tr>
<tr>
<td>Nominal speed</td>
<td>7200 rpm</td>
</tr>
<tr>
<td>Nominal torque (max. continuous)</td>
<td>0.13 Nm</td>
</tr>
<tr>
<td>Nominal current (max. continuous)</td>
<td>0.456 A</td>
</tr>
<tr>
<td>Stall torque</td>
<td>23.3 Nm/km</td>
</tr>
<tr>
<td>Stall current</td>
<td>1.68 A</td>
</tr>
<tr>
<td>Max. efficiency</td>
<td>84 %</td>
</tr>
</tbody>
</table>

Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminal resistance</td>
<td>0.9 Ω</td>
</tr>
<tr>
<td>Terminal inductance</td>
<td>0.594 mH</td>
</tr>
<tr>
<td>Torque constant</td>
<td>13.8 mN/m A</td>
</tr>
<tr>
<td>Speed constant</td>
<td>686 rpm/V</td>
</tr>
<tr>
<td>Speed/torque gradient</td>
<td>445 rpm/kV/km</td>
</tr>
<tr>
<td>Mechanical time constant</td>
<td>150.3 ms</td>
</tr>
<tr>
<td>Rotor inertia</td>
<td>4.57 g/cm²</td>
</tr>
</tbody>
</table>

Thermal data

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thermal resistance, housing</td>
<td>20 K/W</td>
</tr>
<tr>
<td>Thermal resistance, winding</td>
<td>6 K/W</td>
</tr>
<tr>
<td>Thermal time constant, housing</td>
<td>9.24 s</td>
</tr>
<tr>
<td>Thermal time constant, winding</td>
<td>33.2 s</td>
</tr>
<tr>
<td>Ambient temperature</td>
<td>-20...+65 °C</td>
</tr>
<tr>
<td>Max. winding temperature</td>
<td>+65 °C</td>
</tr>
</tbody>
</table>

Mechanical data

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bearing type</td>
<td>sleeve bearing</td>
</tr>
<tr>
<td>Max. speed</td>
<td>18000 rpm</td>
</tr>
<tr>
<td>Axial play</td>
<td>0.05 - 0.15 mm</td>
</tr>
<tr>
<td>Radial play</td>
<td>0.013 mm</td>
</tr>
<tr>
<td>Max. axial load (dynamic)</td>
<td>1 N</td>
</tr>
<tr>
<td>Max. axial load (static)</td>
<td>80 N</td>
</tr>
<tr>
<td>[yield, shaft support]</td>
<td>0 N</td>
</tr>
<tr>
<td>Max. radial load</td>
<td>2.8 N, 5 mm from flange</td>
</tr>
</tbody>
</table>

Other specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pole pairs</td>
<td>1</td>
</tr>
<tr>
<td>Number of commutator segments</td>
<td>9</td>
</tr>
<tr>
<td>Number of commutation cycles</td>
<td>0</td>
</tr>
</tbody>
</table>

Product

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>84 g</td>
</tr>
</tbody>
</table>
Datasheet for servos

Parallax Standard Servo (#900-00005)

The Parallax Standard Servo provides 180° range of motion and position control to your project. Great for animatronics and robotics applications.

Features

- Holds any position between 0 and 180 degrees
- 38 oz-in torque at 6 VDC
- Accepts four mounting screws
- Easy to interface with any Parallax microcontroller or PWM-capable device
- Simple to control with the PULSOUT command in PBASIC
- High-precision gear made of POM (polyoxymethylene) makes for smooth operation with no backlash
- Weighs only 1.55 oz (44 g)

Key Specifications

- Power requirements: 4 to 6 VDC*; Maximum current draw is 140 +/- 50 mA at 6 VDC when operating in no load conditions, 15 mA when in static state
- Communication: Pulse-width modulation, 0.5%–2.5% ms high pulse, 20 ms intervals
- Dimensions approx. 2.2 x 0.8 x 1.6 in (5.58 x 1.9 x 4.06 cm) excluding servo horn
- Operating temperature range: 14 to 122 °F (-10 to +50 °C)

*Power Requirement Notes

The manufacturer specifies 4-6 VDC for this servo. However, we find that this servo is tolerant of a 9 V battery for very brief periods of time when there is no load, as used in some activities in the Stamps in Class series of tutorials. (Slight jittering may be observed when batteries are fresh, this does not cause damage). Do not use this servo with an unregulated wall-mount supply, or a regulated wall mount supply exceeding 6 VDC.

Servo current draw can spike while under load. Be sure that your application’s power supply and voltage regulator is prepared to supply adequate current for all servos used. Do not try to power this servo directly from a BASIC Stamp module’s or any microcontroller’s Vss or Vss pins; do not connect the servo’s Vss line directly to the BASIC Stamp module’s or any microcontroller’s Vss pin.
Quick-Start Circuit

Specifications

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
<th>Minimum</th>
<th>Typical</th>
<th>Maximum</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Signal</td>
<td>Input TTL or CMOS</td>
<td>3.3</td>
<td>5.0</td>
<td>Vservo + 0.2</td>
<td>V</td>
</tr>
<tr>
<td>2</td>
<td>Vservo</td>
<td>Power Supply</td>
<td>4.0</td>
<td>5.0</td>
<td>6.0</td>
<td>V</td>
</tr>
<tr>
<td>3</td>
<td>VSS</td>
<td>Ground</td>
<td>0</td>
<td></td>
<td></td>
<td>V</td>
</tr>
</tbody>
</table>

Power Precautions

- Do not use this servo with an unregulated wall-mount supply. Such power supplies may deliver variable voltage far above the stated voltage.
- Do not power this servo through the BASIC Stamp Module's Vdd pin.
- Servo current draw can spike while under peak load; be sure your application's regulator is prepared to supply adequate current for all servos used in combination.
- Some Stamps in Class tutorials, such as What's a Microcontroller? instruct the user to briefly power these servos with a 9 V battery when using a HomeWork Board and no load; this does not cause damage.

Board of Education Jumper Connection

When connecting the servo to the Board of Education Rev C's servo header, be sure the jumper is set to Vdd (regulated 5 VDC for this board) as shown in the figure below. Failure to place the jumper at this setting can cause damage your servo.

Using a Separate Power Supply on a HomeWork Board

The BASIC Stamp HomeWork Board uses a 9 V battery for a power supply. A servo can drain a fresh 9 V battery in under 20 minutes! Follow these directions to build two servo ports on the breadboard, and power them with a separate battery pack.

Hardware Required

1. BASIC Stamp HomeWork Board (serial #555-26158 or USB #555-26188)
2. Battery pack with tinned leads (Parallax #1753-00001)
3. Parallax standard servos (#900-00005)
4. 5-pin male-male headers (Parallax #441-00003)
5. Jumper wires (10-pack: Parallax #550-00016)
6. 1.5 V AA batteries

Copyright © Parallax Inc. Parallax Standard Servo (#900-00005) v2.2 10/24/2011 Page 2 of 5
Appendix B

Schematics
Schematic of the whole system
Appendix C

Arduino Code
APPENDIX C. ARDUINO CODE

Arduino code with servo control and haptic feedback

/* Description
This is the main code for the whole system.
It converts the signal from potentiometers to servos.
A PID control is implemented for controlling
the FSR signals. This uses a PID library.*/

#include <PID_v1.h> //Import PID library
#include <Servo.h> //import servo package
Servo myservo0; //Servo for robotic arm (base)
Servo myservo1; //Servo for robotic arm (arm)
Servo myservo2; //Servo for robotic arm (claw)

int potPin0 = 0; //Potentiometer 0, arm unit (base)
int potPin1 = 1; //Potentiometer 1, arm unit (arm)
int potPin2 = 2; //Potentiometer 2, arm unit (claw)

//Connections to H−bridge
int IN1 = 1; //Motor IN1
int IN2 = 2; //Motor IN2
int ENA = 3; //Motor ENA
double PWM;

int val0; //Initiate variable
int val1; //Initiate variable
int val2; //Initiate variable

//Variables for PID
double Actual, Error, Integral, P, I, D, ScaleFactor, Drive,
     Direction, Last;

int fsrPinHand = 3; //FSR and 10K pulldown connected to a3
int fsrPinRobot = 4; //FSR and 10K pulldown connected to a4
//The analog reading from the FSR resistor divider
double fsrReadingRobot, fsrReadingHand, fsrReadingRobotMap,
     fsrReadingHandMap;

double Setpoint, Input, Output;

//This part is for switching the PID on and off
int inPin = 5; //Input pin used for a on/off switch
int outPin = 4; //Output pin used for LED
int PIDstate = HIGH; //Current state. Turning feedback on/off
int Buttonreading; //The current reading from input pin
int previous = LOW; //The previous reading from input pin
long time = 0; //The last time the output pin was toggled
long debounce = 200; //Debounce time, increase if output flickers

/* Setting up PID using the library
* Input: The variable to be controlled (double)
* Output: The variable to adjust by the pid (double)
* Setpoint: The value we want to Input to maintain (double)
* Kp, Ki, Kd: Tuning Parameters (double>=0)
* Direction: DIRECT or REVERSE, determines which direction
* the output will move when faced with a given error.
*/

//The calculated constants for PID
double Kp = 0.0536, Ki = 2.4215, Kd = 0;
PID myPID(&fsrReadingHand, &Drive, &fsrReadingRobot, Kp, Ki, Kd, DIRECT);

void setup() {
    myPID.SetMode(AUTOMATIC); //Setting up the PID library

    pinMode(inPin, INPUT); //The switch is an input signal
    pinMode(outPin, OUTPUT); //The LED is an output signal

    Serial.begin(9600); //Serial between Arduino and computer

    myservo0.attach(9); //Digital pin 9 for servo 0
    myservo1.attach(10); //Digital pin 10 for servo 1
    myservo2.attach(11); //Digital pin 11 for servo 2

    pinMode(IN1, OUTPUT); //Digital pin = output
    pinMode(IN2, OUTPUT); //Digital pin = output
    digitalWrite(IN1, LOW); //Rotate forward
    digitalWrite(IN2, HIGH);
void loop()
{
  //On/Off part
  Buttonreading = digitalRead(inPin);
  /*if the input just went from LOW and HIGH and we’ve waited
   long enough
   to ignore any noise on the circuit, toggle the output pin
   and remember
   the time*/
  if (Buttonreading == HIGH && previous == LOW && millis() - time > debounce) {
    if (PIDstate == HIGH)
      PIDstate = LOW;
    else
      PIDstate = HIGH;
    time = millis();
  }

digitalWrite(outPin, PIDstate);

  previous = Buttonreading;

  //SERVO PART
  val0 = analogRead(potPin0); //Read analog 0
  val1 = analogRead(potPin1); //Read analog 1
  val2 = analogRead(potPin2); //Read analog 2

  //Change range of potentiometers
  val0 = map(val0, 0, 614, 0, 179);
  val1 = map(val1, 0, 614, 0, 179);
  val2 = map(val2, 791, 1023, 50, 140);

  //This part creates some limitations for the movement.
  //Servo 0 base
  if (val0 < 179)
    {
    myservo0.write(val0); //Potentiometer to servo
    }

  //servo 1 arm
  if (val1 < 179 && val1 > 80)
{  
  myservo1.write(val1);  //Potentiometer to servo
}

//servo 2 claw  
if (val2 < 179)
{
  myservo2.write(val2);  //Potentiometer to servo
}

//PID PART
fsrReadingRobot = analogRead(fsrPinRobot);  //Analog signal from robot FSR
fsrReadingRobotMap = map(fsrReadingRobot, 0, 995, 0, 255);  //Change range

fsrReadingHand = analogRead(fsrPinHand);  //Analog signal from hand FSR
fsrReadingHandMap = map(fsrReadingHand, 0, 995, 0, 255);  //Change range

myPID.Compute();  //Calculate PID

if (fsrReadingRobot == 0) {
  Drive = 0;
}

///// PID On or OFF////////
if (PIDstate == HIGH) {
  analogWrite(ENA, Drive);  //Send PWM command to motor board
  Serial.print("PID IS ON!!!");
}
else {
  Drive = 0;
  analogWrite(ENA, Drive);  //Send PWM command to motor board
  Serial.print("PID IS OFF!!!");
}

//This part prints important values to the serial monitor
Serial.print("Analog reading Robot = ");
Serial.print(fsrReadingRobot);
APPENDIX C. ARDUINO CODE

Serial.print("Analog reading Hand = ");
Serial.print(fsrReadingHand);

Serial.print("PWM = ");
Serial.print(Drive);

Serial.print("Pot0 = ");
Serial.print(val0);

Serial.print("Pot1 = ");
Serial.print(val1);

Serial.print("Pot2 = ");
Serial.println(val2);
Arduino code for step response measurement

/* Description
 * This code calculates the step response
 * for the system. When the FSR on the robotic
 * hand is pressed so that the value of the FSR
 * gets over 350, the measuring starts. Afterwards
 * the program prints the FSR values from the
 * arm unit. This data can be copied and pasted
 * into MATLAB for further calculations
 */

#include <PID_v1.h> //Import PID library

//Timer for the measuring
int time0 = millis();
int time1 = 0;
int interval2 = 50;
int interval = 50;
int array[40]; //Empty list for measured FSR

int led = 13; //LED pin

//Connections to H-bridge
int IN1=1; //Motor IN1
int IN2=2; //Motor IN2
int ENA=3; //Motor ENA
double Drive;
double PWM;

int fsrPinRobot = 4; //FSR and 10K pulldown connected to a3
int fsrPinHand = 3; //FSR and 10K pulldown connected to a4

double fsrReadingRobot , fsrReadingHand , fsrReadingRobotMap ,
fsrReadingHandMap; //The analog reading from the FSR resistor divider

double Setpoint , Input , Output;

//Specify the links and initial tuning parameters
double Kp = 0.0536 , Ki = 2.4215 , Kd = 0;

int inPin = 5; //The number of the input pin
int outPin = 4; //The number of the output pin
int PIDstate = HIGH;  //Current state of the output pin
int Buttonreading;    //Current reading from the input pin
int previous = LOW;   //Previous reading from the input pin

// the follow variables are long's because the time,
// measured in miliseconds,
// will quickly become a bigger number than can be stored in
// an int.
long time = 0;        //The last time the output pin was toggled
long debounce = 200;  //The debounce time, increase if the
                      // output flickers
int place = 0;
int count = 0;
char start = '0';
int STARTTID;

// Setting up PID using the library
*Input: The variable to be controlled (double)
*Output: The variable to adjust by the pid (double)
*Setpoint: The value we want to Input to maintain (double)
*Kp, Ki, Kd: Tuning Parameters (double>=0)
*Direction: DIRECT or REVERSE. determines which direction
the output will move when faced with a given error.
*/
PID myPID(&fsrReadingHand, &Drive, &fsrReadingRobot, Kp, Ki, Kd, DIRECT);
void setup(){
  myPID.SetMode(AUTOMATIC);  //Setting up the PID library
  pinMode(inPin, INPUT);
  pinMode(outPin, OUTPUT);
  Serial.begin(9600);    //Set up the serial
  pinMode(IN1,OUTPUT);  //Digital pin = output
  pinMode(IN2,OUTPUT);  //Digital pin = output
digitalWrite(IN1,LOW);    //Rotate forward
digitalWrite(IN2,HIGH);
  Serial.println("Begin testing");
}
void loop()
```cpp
{  
digitalWrite(outPin, PIDstate);
  
  //PID PART
  fsrReadingRobot = analogRead(fsrPinRobot);
  fsrReadingRobotMap = map(fsrReadingRobot, 0, 995, 0, 255);
  //Change range

  fsrReadingHand = analogRead(fsrPinHand);
  fsrReadingHandMap = map(fsrReadingHand, 0, 995, 0, 255); //Change range

  myPID.Compute(); //Calculate PID

  ///// PID On or OFF////////
  if (fsrReadingRobot == 0){
    Drive = 0;
  }

  analogWrite(ENA, Drive); //send PWM command to motor board
  Serial.println(Drive);
  Serial.print(fsrReadingRobot);

  //Start measuring if the FSR reading is greater than 350
  if (fsrReadingRobot >=350 &place == 0){
    STARTTID = millis();
    array[place] = fsrReadingHand;
    place++;
  }

  //Start measuring if the FSR reading is greater than 350
  if (fsrReadingRobot >=350){
    if (millis()−STARTTID >= interval2){
      interval2 = interval2 + interval;
      array[place] = fsrReadingHand;
      place++;
    }
  }

  //Print the values after measuring is done, 2 seconds
  if (millis()−STARTTID >= 2000){
    for (int i=0;i<=40;i++){
      Serial.print(array[i]);
    }
  }
```

47
APPENDIX C. ARDUINO CODE

```cpp
123     Serial.print(",*");
124   }
125   Serial.println("*");
126   Serial.println("EFTER ARRAY");
127   Serial.println(fsrReadingRobot);
128   delay(100000);
129   }
130 }
131 
count++;
133 }
```
Appendix D

Matlab Code
Matlab program for real step response

%This code calculates the step response for the real system.
%
The system is measured 10 times. The mean value is then calculated. The time of
%measurement is 2 seconds. The variable target is the systems target value.
%The measurement values are taken from a test using an Arduino UNO.

clear all, clc, close all

numberOfMeasurements = 10;
%
target = target value for the system.
%measure = FSR analog read value.
target{1} = 574; %targetx = target value for the system.
measure{1} = [349,357,344,358,368,376,401,436,492,500,...
517,533,542,556,557,564,567,568,569,572,...
569,568,564,569,571,571,572,572,568,567,...
570,575,576,575,575,577,574,576,575];

target{2} = 553;
measure{2} = [298,325,332,341,362,430,444,460,481,507,...
521,528,542,543,547,546,553,553,555,558,...
563,565,568,574,579,580,574,574,573,574,...
573,576,577,577,577,576,576,573,574,566];

target{3} = 659;
measure{3} = [350,359,368,402,417,444,472,499,531,547,...
556,556,575,586,594,600,616,624,623,624,...
627,629,632,632,637,644,639,641,644,647,...
655,660,665,666,670,671,667,662,662];

target{4} = 593;
measure{4} = [281,280,271,294,345,427,467,488,508,519,...
534,542,550,557,564,567,573,585,595,596,...
600,601,603,599,609,611,611,609,613,612,...
615,614,609,616,622,623,620,623,630,634];

%target = target value for the system.
%measure = FSR analog read value.
target{5} = 563;
measure{5} = [291,293,300,306,329,346,375,440,455,485,...
515,527,546,544,553,556,560,569,564,567,...
564,570,562,568,566,567,566,565,558,547,...
536,527,523,526,524,528,529,537,547,553];
```matlab
% Target and measurement values for different time intervals

% Target values for time interval 6
target{6} = 589;
measure{6} = [254, 265, 288, 364, 435, 462, 483, 494, 523, 529, ...
            533, 540, 545, 555, 563, 568, 572, 578, 577, ...
            576, 580, 586, 596, 599, 604, 598, 596, 595, ...
            603, 591, 582, 577, 575, 578, 579, 589, 590, 584];

% Target values for time interval 7
target{7} = 573;
measure{7} = [248, 255, 329, 355, 398, 432, 495, 521, 561, 573, ...
            566, 560, 558, 557, 550, 547, 546, 554, 558, 560, ...
            567, 566, 566, 562, 559, 561, 558, 559, 551, 549, ...
            550, 552, 552, 553, 553, 559, 561, 562, 563, 568];

% Target values for time interval 8
target{8} = 513;
measure{8} = [238, 260, 326, 351, 383, 431, 457, 503, 512, 503, ...
            510, 514, 509, 513, 513, 508, 512, 514, 512, 511, ...
            506, 504, 500, 502, 504, 496, 500, 502, 502, 500, ...
            510, 514, 514, 523, 524, 532, 538, 538, 538, 532];

% Target values for time interval 9
target{9} = 579;
measure{9} = [281, 305, 373, 392, 460, 480, 512, 529, 545, 567, ...
            578, 591, 598, 600, 599, 605, 606, 610, 618, 619, ...
            616, 609, 612, 615, 623, 626, 628, 626, 627, 626, ...
            626, 621, 617, 613, 613, 607, 607, 599, 593, 574];

% Target values for time interval 10
target{10} = 558;
measure{10} = [242, 255, 287, 311, 374, 400, 421, 442, 464, 489, ...
              493, 496, 499, 502, 503, 503, 505, 506, 510, 511, ...
              513, 517, 521, 526, 525, 522, 525, 525, 529, 526, ...
              523, 529, 532, 529, 527, 535, 537, 530, 534, 537];

% Time goes from 1 to 3 seconds
interval = 0.05;
measureTime = 1:0.05:3 - interval;
totalTime = 0:interval:3 - 0.05

% Creates a step response function for reference signal
refSignal = [0 0 1 1 1]
x=[0 1 1 2 3]
plot(x,refSignal,'r')
hold on

% Creates empty lists.
startVal = [];```
targetVal = []; %Start value for total value of all measurements.
totalVal = 0
for i = 1:numberOfMeasurements %Create lists with start values and target values.
    startVal(i) = measure{i}(1);
    targetVal(i) = target{i};
%To make the measurements start from zero for our %transfer function, subtract the starting value.
    measurementFromZero{i} = measure{i}−startVal(i);
%To make the measurement go to 1 (step function) we scale the %measurement.
    scaledMeasurement{i} = measurementFromZero{i}./(targetVal(i)−startVal(i));
%Creates a plot with all the measurements
figure (1)
plot (measureTime, scaledMeasurement{i}, 'b')
title ('Step response function of real system')
xlabel ('Time [s]')
ylabel ('Step response')
axis ([0 3−interval 0 1.5])
hold on
grid on
%Calculate the total value of all the scaled measurements. This will be %used for calculation of the mean value for the measured system.
totalVal = totalVal+scaledMeasurement{i};
end
%Calculate mean value.
meanVal = totalVal/numberOfMeasurements;
%Creates a plot with the mean value.
figure (2)
plot (measureTime, meanVal)
title ('Step response function for real system (mean value)')
xlabel ('Time [s]')
ylabel('Step response')
axis([0 3 interval 0 1.5])
grid on
hold on
%add the reference signal in the plot
plot(x,refSignal,'r')
Matlab program for pole placement

%This program calculates the appropriate constants for different controllers

clear all, clc, close all

% Values for the motor from datasheet (maxon motor 110123)
J = 4.27 + 0.4; %gcm^2 (Rotor inertia)
J = J * 1e^-3 * 1e^-4; %kgm^2
b = 0; %N*m*s (motor viscous friction constant)
K = 13.9; %mNm/A (Torque constant)
K = K * 1e^-3;
R = 8.9;%Ohm (electric resistance)
L = 0.585 * 1e^-3; %electric inductance
L = 0;

% Transfer function for the motor
s = tf('s');
G_motor = K/((J*s+b)*((L*s+R)+K^2))

SystemPoles = pole(G_motor); % The poles of the motor transfer function

% Using pole placement to calculate P, I, and D parameters by placing poles for the denominator of the closed loop system
% Using only PI controller

P47 = 94*(4.156e-6/0.0139) - 0.0001932
I47 = (2209-1)*(4.156e-6/0.0139)

P70 = 140*(4.156e-6/0.0139) - 0.0001932
I70 = (4900-1)*(4.156e-6/0.0139);

P100 = 200*(4.156e-6/0.0139) - 0.0001932;
I100 = (10000-1)*(4.156e-6/0.0139);

P150 = 250*(4.156e-6/0.0139) - 0.0001932;
P150 = 300*(4.156e−6/0.0139) − 0.0001932;
I150 = (22500−1)*(4.156e−6/0.0139);

% Pole at s = −90
PoleEquation90 = (s+90)^2;
P90 = 180*(4.156e−6/0.0139) − 0.0001932;
I90 = (8100−1)*(4.156e−6/0.0139);

% Using the whole PID
% Pole at s = −47
P47_2 = 94/0.0139 − 0.0001932;
I47_2 = 2209/0.0139;
D47_2 = 1/0.0139 − 4.156e−6;

% Pole at s = −0.1
PoleEquation01 = (s+0.1)^2;
P01_2 = 0.2/0.0139 − 0.0001932;
I01_2 = 0.01/0.0139;
D01_2 = 1/0.0139 − 4.156e−6;

% Pole at s = −0.01
PoleEquation001 = (s+0.01)^2;
P001_2 = 0.02/0.0139 − 0.0001932;
I001_2 = 0.0001/0.0139;
D001_2 = 1/0.0139 − 4.156e−6;
Appendix E

Plots
Plots of step response for different poles with PI controller

Figure E.1. Step response function with Simulink for the pole $s = -47$. The values are: $P = 0.0279, I = 0.6602$ The system is quite fast and with no overshoot.
Figure E.2. Step response function with Simulink for the pole $s = -70$). The values are: $P = 0.0417$, $I = 1.4648$. This system has a shorter settling time than the system with the pole $s = -47$. 
Figure E.3. Step response function with Simulink for the $s = -100$). The values are: $P = 0.0596, I = 2.9896$. This system has a shorter settling time than the the systems for pole $s = -47$ and $s = -70$. The problem with this system is that it produces a small overshoot.
Figure E.4. Step response function with Simulink for the pole $s = -150$). The values are: $P = 0.0895, I = 6.7270$. This is a very fast system, but has quite a large overshoot.