DEGREE PROJECT IN MECHANICAL ENGINEERING,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Accuracy and Repeatability of a Robotic Arm

CARL-VICTOR LIDHOLM

VICTOR RUNNQUIST

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Accuracy and Repeatability of a Robotic Arm

VICTOR RUNNQUIST
CARL-VICTOR LIDHOLM

Bachelor's Thesis in Mechatronics at KTH
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:40

Abstract

The purpose of this project was to create a model of a robotic arm with 4-DOF equipped with a gripper that is able to move an object to a predetermined area. This thesis investigated the robotic arm's accuracy by repeating the same predetermined movement and measure the error. The error was measured based on the objects displacement from the target area after a series of repeated movements by the arm.
After constructing the robotic arm, an experiment was set up to measure the accuracy and repeatability of the arm. The robot achieved 2.506 mm to 0.922 mm for accuracy and 5.995 mm to 4.059 mm for repeatability depending on speed.

Keywords: Mechatronics, Robotic arm, Accuracy and Repeatability.

Referat

Noggrannhet och repeterbarhet för en Robotarm

Syftet med detta projekt var att skapa en modell av en robotarm med 4 frihetsgrader och en klo för att kunna flytta ett objekt till ett förbestämt område. Denna avhandling har undersökt robotens noggrannhet genom att upprepa en förbestämd rörelse och mäta felet. Felet bestämdes genom att flytta ett objekt till en förbestämd plats och mäta objektets position i förhållande till målet.
Den färdigkonstruerade roboten har en norggrannhet på 2.506 mm till 0.922 mm och en repeterbarhet på 5.995 mm till 4.059 mm beroende på hur fort roboten rör sig.

Nyckelord: Mekatronik, Robotarm, Noggrannhet, Repeterbarhet.

Acknowledgements

We would like to thank the course examiner Nihad Subasic for his lectures and feedback during this project. We would also like to thank the lab assistant Amir Avdic for all the help and useful information provided. We also want to thank Staffan Qvarnström for all the help and tips provided during this project, and also for providing the components needed. Lastly we would like to thank Lars Hässler and KTH Prototype Center for helping us with the laser cutting.

Carl-Victor Lidholm and Victor Runnquist
Stockholm, May 2021

Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two dimensional. 15
3D	Three dimensional. 14
CPS	Cyber Physical System. 7, 26
CPU	Central Processing Unit. 5
DOF	Degrees of freedom. 1, 6
KTH	Royal Institute of Technology. 2, title-1
PWM	Pulse width modulation. 5, 8
SEK	Swedish Crown (Currency). 2

Chapter 1

Introduction

1.1 Background

In today's industrial world there are many important but mechanically simple tasks that are both repetitive and precision sensitive. Such tasks include but are not limited to picking up and placing, screw fastening and welding [4].
A robotic arm may be perfect for such tasks since it can be programmed to execute the very same motion again and again without tiring. It is therefore not surprising that robotic arms can be found to be a part of many modern industrial, medical and military applications [1], for example the automotive industry [3].

1.2 Purpose

The purpose of this project is to build a small version of a robotic arm with four DOF that can be used for production or in a warehouse. The arm will be outfitted with a gripper attachment that can be removed if a task calls for another tool. This report will research:

- Can the gripper be made to lift and move an object?
- What is the maximum weight the robot arm can lift?
- How well can the arm execute a pre-planned movement, what is the accuracy and repeatability of that movement?

1.3 Scope

This project has a monetary budget of 1000 SEK outside of what the KTH course MF133X provides such as an Arduino Uno R3, cables and most necessary tools. The project is to be completed by May of 2021. Due to these limitations in time and budget there will only be one attachment to the robot arm. Therefore the robot arm will not be able to support different types of attachments that require additional wiring. Furthermore, since it was decided that the circuit board, breadboard and power supply would be attached to the arm's base rather than a more complicated integrated design, the user will have to mind the cables when operating the robot. Thus the arm's rotation will be limited and it will need to be returned to a neutral position even if the servo-motor could spin indefinitely.

1.4 Method

The robot is mainly constructed using plywood with metal rods as support. The plywood parts are cut from boards using a laser cutter [22] at the KTH Prototype Center [18]. The files for the laser cutter were made in AutoCAD [12]. The electrical parts consist of an Arduino circuit board wired to electrical servo-motors via a breadboard and a power supply. The servo-motors are controlled by a USB connection to a PC.
In order to test the arms ability to execute a pre-planned movement and determine it's accuracy a scripted movement sequence were made. The sequence made the robot pick up an object, straighten out, rotate and place the object in another location. This movement was repeated multiple times in order to measure the difference in placement between repetitions.

Chapter 2

Theory

This chapter will cover the necessary theory to build and test the robotic arm.

2.1 Past research

Previous work such as "Accuracy Measurements of Miniature Robot Using Optical CMM" [6] and the bachelor's theses of other students such as "Robotic Arm controlled by Arm Movements" [8] and "Sudoku robot" [7] inspired us to make our robot and to test it's accuracy and repeatability.

2.1.1 Accuracy and Repeatability

Two important parameters in robotics are accuracy and repeatability. The difference between the desired position and the obtained position is called the accuracy, i.e, the error, as shown in 2.1. Repeatability on the other hand, is the robots ability to repeat the same task, i.e., move back to a desired position repeatedly [2].

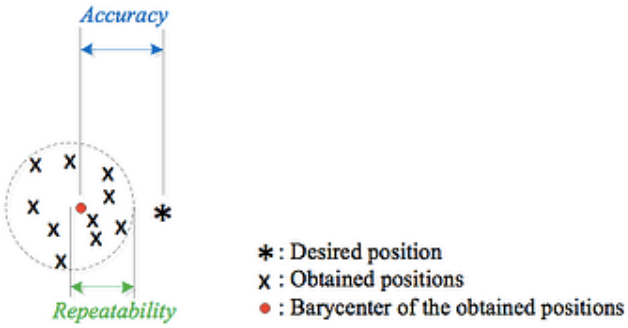

Figure 2.1: Accuracy & repeatability [15]

Both accuracy and repeatablity depends on many factors, e.g., friction, loading, motors, construction procedure, etc [5][2].

2.2 Servo-motors

A servo-motor generally has three connections: power, ground and control. Some servos have a fourth connection that gives information to the controller about the motors position. Inside the servo there is a small DC-motor connected to a potentiometer and a control circuit, as shown in figure 2.2. When the motor rotates, the resistance of the potentiometer changes and allows the control circuit to determine the amount of movement there is [20].

When the motor have reached the desired position, the motor stops and holds that position. If external forces try to push the servo back, the servo will resist and try to hold the desired position.

Figure 2.2: Insides of a servo-motor [20]

2.2.1 Pulse-width modulation

Pulse-width modulation (PWM) is a way of controlling analog devices with digital outputs. Servo-motors are controlled by these PWM-signals through a control wire. Based on the duration of the pulse sent through the control wire, the servo-motor either turn clockwise or counter-clockwise. These pulses have a minimum and maximum duration, usually between 1 ms and 2 ms, as shown in figure 2.3. These pulses repeat every 20 ms to update the servo of a potentially new desired position [9].

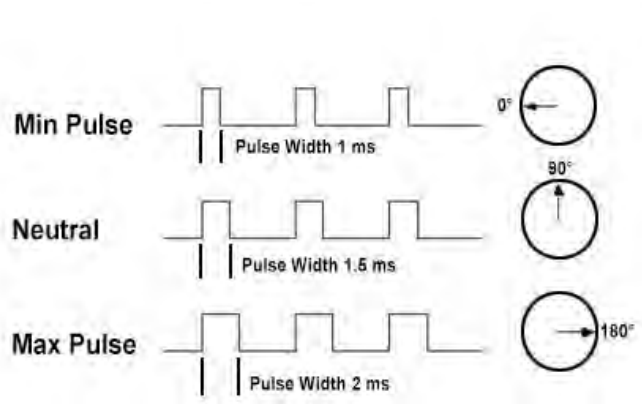

Figure 2.3: PWM duration and rotation [19]

2.3 Microcontroller

A microcontroller is a small computer with a CPU, memory and ports for inputs and outputs. In this project an Arduino Uno R3, shown in figure 2.4, will be used which is an open-source hardware. This controller uses the programming language C++. The controller has 14 pins, six of those use PWM. Since the Arduino can only supply either max voltage or zero voltage, these six PWM pins can be used to simulate voltages between maximum and minimum voltage. The maximum voltage that the Arduino can handle is five volts.

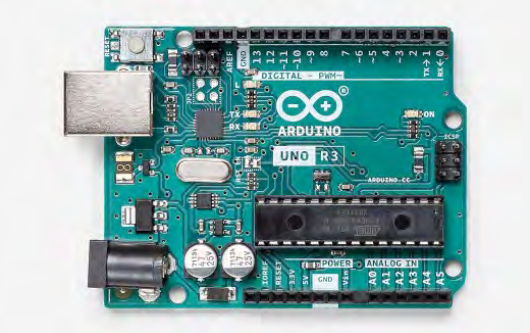

Figure 2.4: Arduino Uno R3 [11].

Chapter 3

Demonstrator

This chapter will describe the construction process of the robotic arm.

3.1 Construction

The arm has a plywood frame cut from plywood boards with a laser cutter [22]. The arm itself consists of three segments, each segment is made of two spaced apart plywood frame pieces stabilized in the middle of the segment by metal rods. The connections between the segments make up the joints of the arm. The first segment is attached to the base board and the last segment is made to accept the gripper attachment. The arm has a total of four DOF, allowing it to rotate at the base, swivel at the shoulder joint, swivel at the elbow joint, rotate at the wrist as well as open and close the gripper.

3.1.1 Components

The frame of the arm is laser cut from boards of plywood.
The circuit board is an Arduino Uno R3.
The servo-motors used in this build are:

- 3x Servo MS-6-40,
- 2x HXT 5010 Twin bearing Digital Servo,
- Gripper for a standard servo from Electrokit. See figure 3.1.

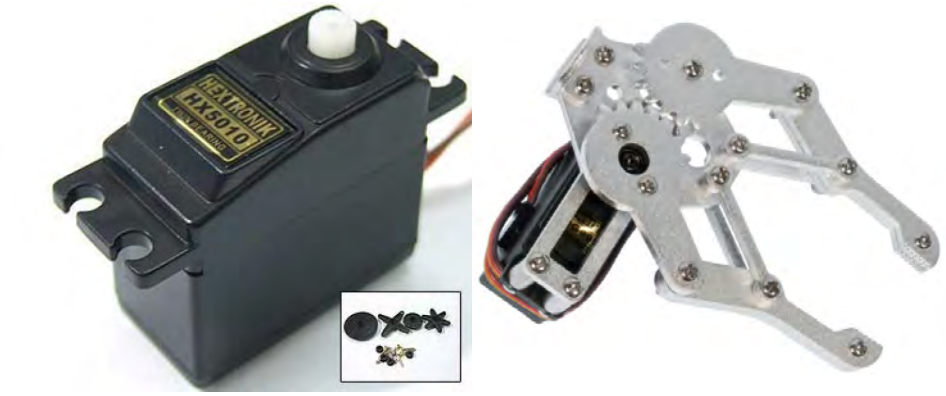

Figure 3.1: A HXT 5010 servo and a gripper from Elekctrokit [16].

3.1.2 Design

The basic design of the robot arm was made in Solid Edge [14] and Acumen CPS [10], as can be seen in figure 3.2.

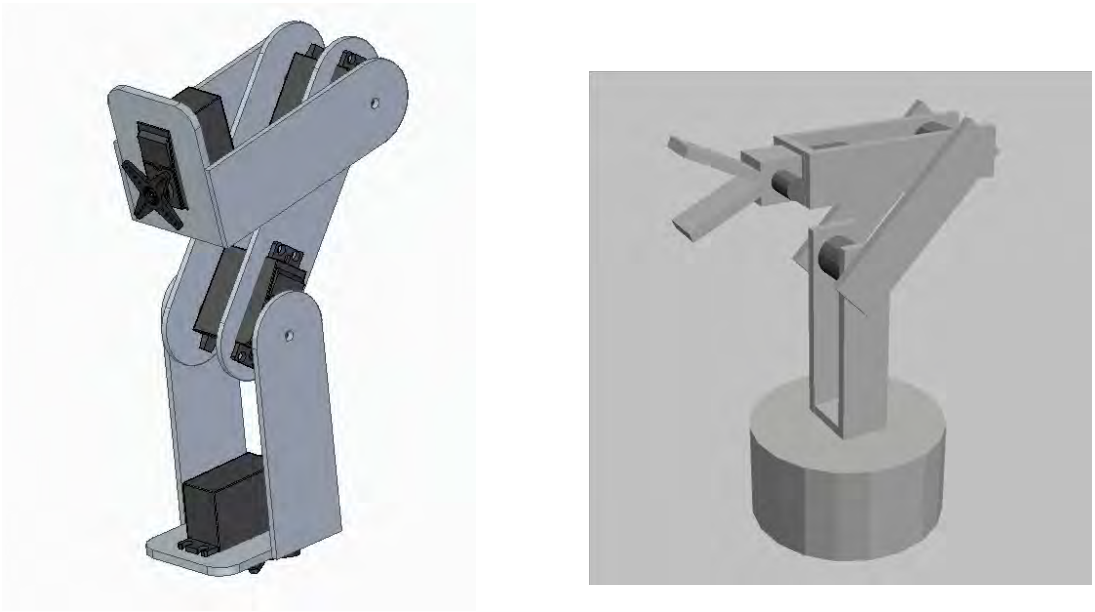

Figure 3.2: Robotic arm designed in Solid Edge [14] to the left and in Acumen [10] to the right.

3.1.3 Electronics

Each servo-motor is connected with three wires. These are the power, ground and control cables. All components, e.i., the Arduino and the five servos, share the same ground-connection from the powersupply. The Arduino is powered through the 5V USB-port while the five servos are connected to the 5V powersupply, as can be seen in figure 3.3. Every control cable on the servos go to an individual PWM enabled digital output on the Arduino.

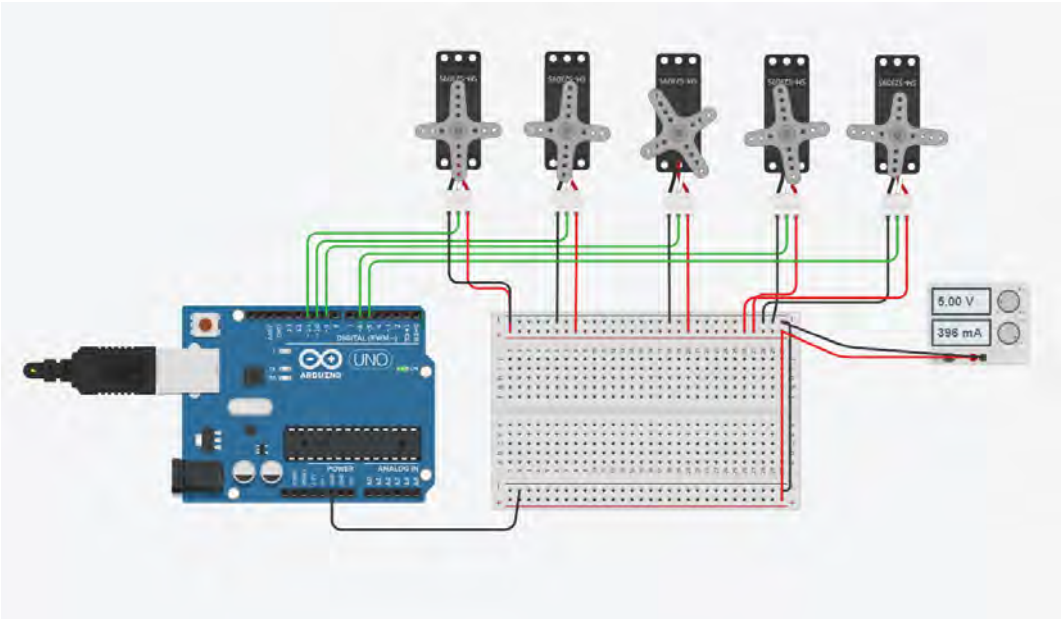

Figure 3.3: Servo-circuit made with Tinkercad [21]

3.1.4 Software

The controlling software was written in C++ using the Arduino library and was made entirely by authors. Calculations for the accuracy and repeatability was calculated using Matlab.

Chapter 4

Results

Below is a picture of the finished robotic arm. It is capable of lifting a weight up to 130 grams and can successfully grip and move an object.

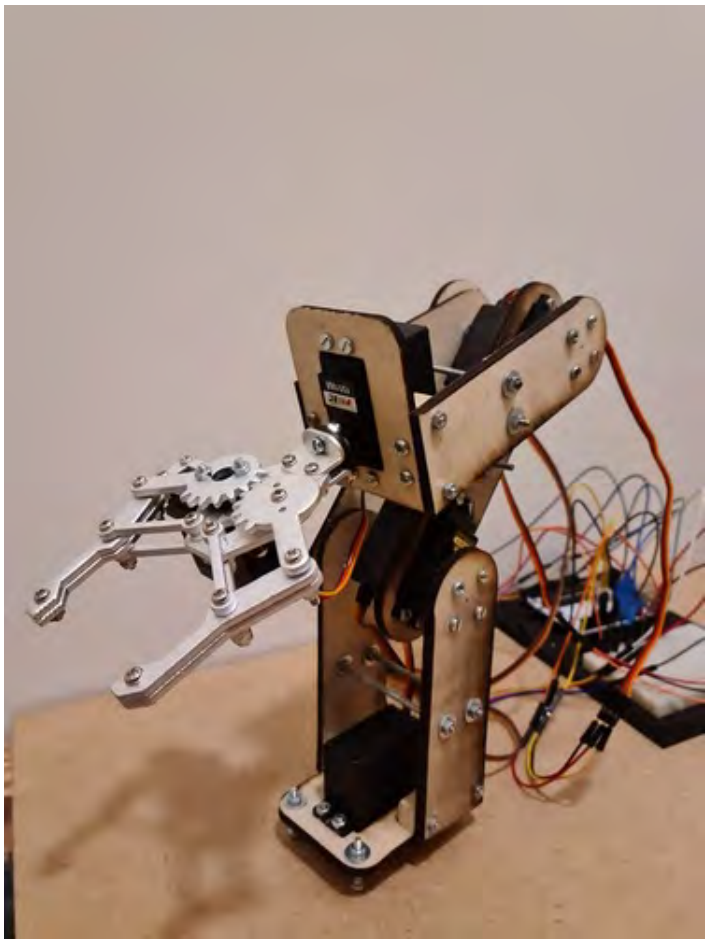

Figure 4.1: The completed robotic arm. Picture taken by authors.

The experiments were set up by placing a drawn coordinate system on top of a small box, see figure 4.2. This coordinate system was used to measure the error by comparing the objects displacement from the target position (origo).

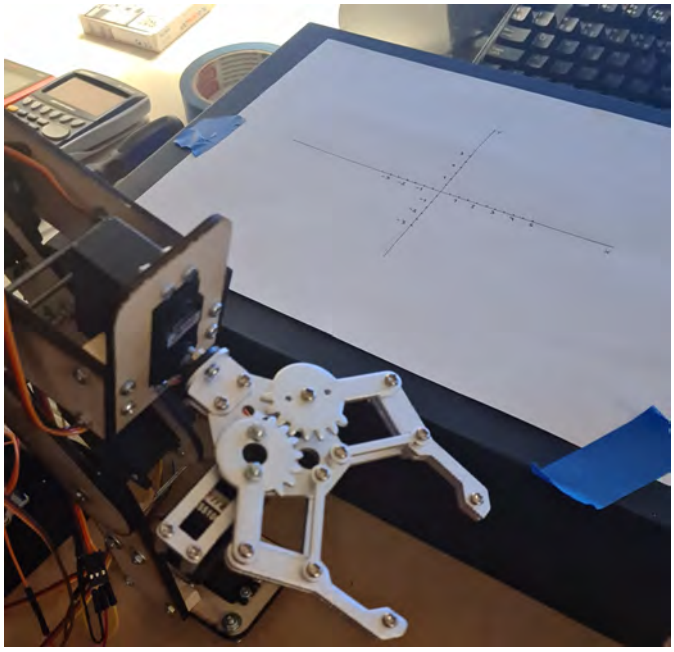

Figure 4.2: Coordinate system to measure the error. Picture taken by authors.

A series of test was conducted using this coordinatesystem to measure the error of the robotic arm. The results of these tests can be seen in the table 4.1 and 4.2. For every repetition, the displacement in both x & y directions were recorded.

4.1 Tables

The displacement of the object moved by the robotic arm are listed in these tables. Table 4.1 contains displacement in x-direction, measured in milimeters. Table 4.2 contains displacement in y-direction, also measured in milimeters.

Table 4.1: Error in accuracy along the X-axis

Repetition, X-direction	1	2	3	4	5	6	7	8	9	10
Error [mm], delay: 20ms	-2	-2	-1	2	1	-3	2	-3	-4	-2
Error [mm], delay: 30ms	-3	0	-2	-3	2	1	2	3	-1	-1
Error [mm], delay: 40ms	-1	-1	2	0	-2	-2	1	2	1	2

Table 4.2: Error in accuracy along the Y-axis

Repetition, Y-direction	1	2	3	4	5	6	7	8	9	10
Error [mm], delay: 20ms	2	4	3	1	2	2	3	-2	2	5
Error [mm], delay: 30ms	-1	-3	-2	-2	1	-3	-2	-4	-2	-3
Error [mm], delay: 40ms	2	-2	-1	-1	2	3	0	2	3	1

4.2 Calculations

Calculations for accuracy and repeatability are shown in this section. The calculations are based on ISO 9283 International Standard [17].

Accuracy

The accuracy is calculated by

$$AP_p = \sqrt{((\bar{x} - x_c)^2 + (\bar{y} - y_c)^2)} \tag{4.1}$$

with

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} x_j \quad , \quad \bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j \tag{4.2}$$

where $\bar{x} - x_c$ and $\bar{y} - y_c$ is the error along the x-axis and y-axis respectively. x_c and y_c is the commanded position, which in this case is origo.
Expressions for $\bar{x}$ and $\bar{y}$ given by equation 4.2 are the coordinates of the barycentre of the cluster of points obtained, see figure 4.3. x_j and y_j is the measured position and n is number of measurements.

Three sets of calculations were made for the accuracy, where the delay of the robotic arms motors varied.

Repeatability

The repeatability is calculated by

$$RP_l = \bar{l} + 3S_l \tag{4.3}$$

The distance from each point to the barycentre of the set is given by

$$l_j = \sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2} \tag{4.4}$$

and the mean of these distances are given by

$$\bar{l} = \frac{1}{n}\sum_{j=1}^{n} l_j \tag{4.5}$$

and the standard deviation S_l

$$S_l = \sqrt{\frac{\sum_{j=1}^{n}(l_j - \bar{l})^2}{n-1}} \tag{4.6}$$

The results from these calculations can be found below in the table 4.3. The calculations can also be found in section A.2 Matlab Code.

	$\mathrm{AP}_p[mm]$	$\mathrm{RP}_l[mm]$
20 [ms]	2.506	5.995
30 [ms]	2.109	5.509
40 [ms]	0.922	4.059

Table 4.3: Table with results of accuracy and repeatability.

In figure 4.3 below, the accuracy and repeatability of the tests can be seen. The red dots are where the object was placed by the robotic arm. The black circle indicates the repeatability, and the radius this circle is the value of repeatability RP_l shown in table 4.3.

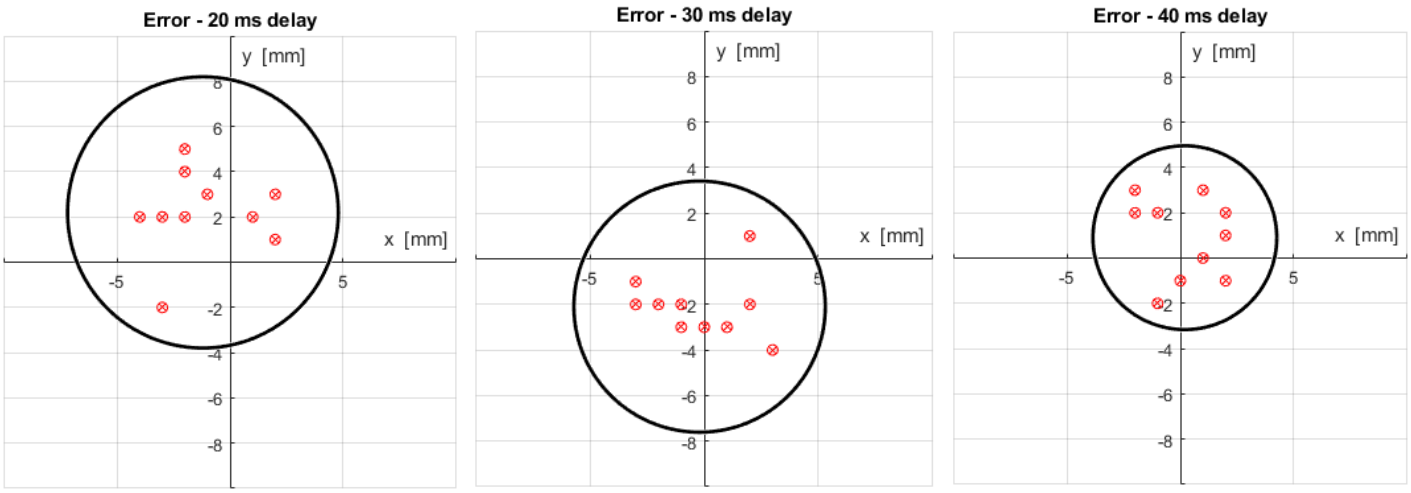

Figure 4.3: These figures show the object placement relative the target position. Plots made with Matlab [13].

Chapter 5

Discussion and conclusion

5.1 Discussion

While the robotic arm was capable of successfully performing the task we initially intended it to do as well as lifting up to 130 grams, there are certain issues with the construction that could see improvement.

One such issue is the stability of the system, such as when coming to a stop from moderate to high speed the arm has a tendency to wobble. The segments that make up the arm are very rigid thanks to the metal support rods. This makes us think the problem lies with the joints and therefore the servos since the rigidity of the joint is reliant upon the stiffness of the servo-motor that is part of that joint. It would be possible to reduce the magnitude of the arm's excess movement with stronger and stiffer servo-motors.

Another issue is the need to reset to a neutral position between movements as to not pull on the external cables. This limits the freedom of the arm since even if the servos allow for a full range of motion, the cables do not. This could either be improved upon by having very lengthy cables or redesigning the arm to move the electronics along with the arm. However having excessively lengthy cables is also not desirable.

During the design portion of this project we initially intended to 3D-print the arm rather than the frame construction we finally made. The main benefit of the 3D printed design is that electrical parts such as the circuit board and cables could be mounted inside the construction which is a better solution since the cables can otherwise obstruct the arm during operation. That said, we eventually realized that modeling several segments of a hollow 3D structure that needed to be precise enough to fit into one piece was to difficult for us. The robot was therefore redesigned into a frame construction.

Since the frame is drawn in 2D which could be sent to either a laser or a water cutter we could choose between using 2 mm thick aluminium or 4 mm thick plywood. We went with plywood since it is far easier to work with and would only weigh about 2.2% more than a frame made of aluminum.

The results show that when repeating the same action, the arm is able to place the object within 2.506 mm to 0.922 mm of the target position with a repeatability of 5.995 mm to 4.059 mm. What we believe to be the greatest cause of inaccuracy is the arm's tendency to wobble when a servo-motor stops moving. The fact that operating the robot at greater speeds leads to greater inaccuracies seems to support this since what causes the wobble is servo-motor's inability to immediately stop the momentum of the arm which increases with the speed of rotation.

5.2 Conclusion

The robotic arm was successfully made and fully capable of performing as per the requirements, even exceeding the strength requirement by 30%. While the arm functions well, the usability would be improved by a design that integrates the electronics onto the moving robot body since that would make cables less obstructing. The precision of the robot could be improved by using stronger and stiffer servo-motors since excess movement causes inaccuracy.

By making the robot move slower the accuracy and repeatability increased. These values range from 2.506 mm to 0.922 mm for accuracy and 5.995 mm to 4.059 mm for repeatability depending on speed.

Bibliography

[1] Haider AF Almurib, Haidar Fadhil Al-Qrimli, and Nandha Kumar. "A review of application industrial robotic design". In: *2011 Ninth International Conference on ICT and Knowledge Engineering*. IEEE. 2012. Date accessed: 2021-05-08 [Online], pp. 105–112. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6152387.

[2] Kevin L Conrad, Panayiotis S Shiakolas, and TC Yih. "Robotic calibration issues: Accuracy, repeatability and calibration". In: *Proceedings of the 8th Mediterranean Conference on Control and Automation (MED2000), Rio, Patras, Greece*. Vol. 1719. 2000. Date accessed: 2021-05-08 [Online]. URL: https://tinyurl.com/csj788dr.

[3] Rahul Gautam et al. "Review on development of industrial robotic arm". In: *International Research Journal of Engineering and Technology (IRJET)* 4.03 (2017. Date accessed: 2021-05-08 [Online]). URL: https://tinyurl.com/4v7thf8a.

[4] Kaustubh Ghadge et al. "Robotic arm for pick and place application". In: *International Journal of Mechanical Engineering and Technology* 9.1 (2018. Date accessed: 2021-05-08 [Online]), pp. 125–133. URL: https://tinyurl.com/rwbc98yw.

[5] HR Ismail et al. "The repeatability analysis of industrial robot under loaded conditions and various distances". In: *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*. 8. World Scientific, Engineering Academy, and Society. 2008. Date accessed: 2021-05-08 [Online]. URL: http://www.wseas.us/e-library/conferences/2008/hangzhou/rocom/11-586-215.pdf.

[6] Asser Vuola and Reijo Tuokko. "Accuracy measurements of miniature robot using optical CMM". In: *International Precision Assembly Seminar*. Springer. 2012. Date accessed: 2021-05-08 [Online], pp. 126–133. URL: https://tinyurl.com/28dny53x.

[7] EMELIE LAURENT and MAGNUS RAMSKÖLD. *Sodoku Robot*. 2020. Date accessed: 2021-05-08 [Online]. URL: https://www.diva-portal.org/smash/get/diva2:1462114/FULLTEXT01.pdf.

[8] Miko Nore and Caspar Westerberg. *Robotic arm controlled by arm movements*. 2019. Date accessed: 2021-05-08 [Online]. URL: https://www.diva-portal.org/smash/get/diva2:1373883/FULLTEXT01.pdf.

[9] Timothy Hirzel. *PWM*. 2018. Date accessed: 2021-05-08 [Online]. URL: https://www.arduino.cc/en/Tutorial/Foundations/PWM.

[10] Software. *Acumen, 2016 Version*. Date accessed: 2021-03-03 [Online]. URL: http://www.acumen-language.org/.

[11] Software. *Arduino, 2021 Version*. Date accessed: 2021-05-07 [Online]. URL: https://store.arduino.cc/arduino-uno-rev3.

[12] Software. *AutoCAD, 2021 Version*. Date accessed: 2021-05-01 [Online]. URL: https://www.autodesk.com/products/autocad/overview?term=1-YEAR.

[13] Software. *Matlab, 2018 Version*. Date accessed: 2021-05-07 [Online]. URL: https://www.mathworks.com/products/matlab.html.

[14] Software. *Solid Edge, 2019 Version*. Date accessed: 2021-05-06 [Online]. URL: https://solidedge.siemens.com/en/.

[15] Website. *Accuracy & Repeatability*. Date accessed: 2021-05-03 [Online]. URL: https://blog.robotiq.com/bid/72766/What-are-Accuracy-and-Repeatability-in-Industrial-Robots.

[16] Website. *Electrokit*. Date accessed: 2021-04-03 [Online]. URL: https://www.electrokit.com/produkt/gripklo-for-standardservon/.

[17] Website. *ISO 9283 International Standard*. Date accessed: 2021-05-07 [Online]. URL: https://tinyurl.com/7mafuw9w.

[18] Website. *KTH Prototype Center*. Date accessed: 2021-05-07 [Online]. URL: https://www.kthprototypecenter.com/.

[19] Website. *PWM*. Date accessed: 2021-04-01 [Online]. URL: https://docs.onion.io/omega2-maker-kit/maker-kit-servo-controlling-servo.html.

[20] Website. *Servo*. Date accessed: 2021-04-03 [Online]. URL: https://thestempedia.com/tutorials/what-is-a-servo-motor/.

[21] Website. *Tinkercad*. Date accessed: 2021-05-06 [Online]. URL: https://www.tinkercad.com/.

[22] Laser Cutter. *Epilog Fusion M2 32/40*. Date accessed: 2021-05-08 [Online]. URL: https://www.epiloglaser.se/lasermaskiner/fusionm2-techspecs.htm.

Appendix A

Code

A.1 Arduino Code

```
// ========================================
// Main program for control of robotic arm
// Bachelor's Thesis in Mechatronics, KTH
// Victor Runnquist & Carl-Victor Lidholm
// 2021-05-07
// ========================================

#include <Servo.h>
// Define servos
Servo sElbow;
Servo sShoulder;
Servo sBase;
Servo sWrist;
Servo sGrip;

// Define initial angle for the servos
int thetaElbow    = 170;
int thetaShoulder = 140;
int thetaBase     = 90;
int thetaWrist    = 90;
int thetaGrip     = 90;

void setup() {
Serial.begin(9600);   // Initialize serial connection to
↪  9600 bps

// Attach each servo to a Digital pin (PWM) on the arduino
sElbow.attach(3);
```

```
sShoulder.attach(5);
sBase.attach(9);
sWrist.attach(6);
sGrip.attach(10);

// Assign initial angle for each servo
sElbow.write(thetaElbow);
sShoulder.write(thetaShoulder);
sBase.write(thetaBase);
sWrist.write(thetaWrist);
sGrip.write(thetaGrip);
}

// ===========   VOID LOOP   =============
// In this void loop we will call on the
// desired servofunction and input the
// desired angle. The loop will go through
// our list of servofunctions and execute
// them in order, making our robotic arm
// move in a predictable and controlled way.
// ========================================
void loop() {

  // INITIAL ANGLES USED:
  // thetaElbow    = 170;
  // thetaShoulder = 140;
  // thetaBase     = 90;
  // thetaWrist    = 90;
  // thetaGrip     = 90;

  delay(10000);        // delay 10 s
  shoulderFun(80);     // Move shoulder DOWN 60 degrees
  delay(500);
  gripFun(115);        // Close gripper
  delay(500);
  shoulderFun(140);    // Move shoulder UP 60 degrees
   ↪  (startposition)
  elbowFun(140);       // Move elbow UP 30 degrees
  delay(500);
  wristFun(270);       // Turn wrist 180 degrees
  delay(500);
  baseFun(180);        // Turn base 90 degrees
  delay(500);
  elbowFun(125);       // Move elbow UP 15 degrees
```

```
  shoulderFun(105);    // Move shoulder DOWN 35 degrees
  delay(500);
  gripFun(90);         // Open gripper (startposition)
  delay(500);
  shoulderFun(140);    // Move shoulder UP 35 degrees
  ↪  (startposition)
  baseFun(90);         // Turn base 90 degrees
  ↪  (startposition)
  delay(500);
  wristFun(90);        // Turn wrist 180 degrees
  ↪  (startposition)
  elbowFun(170);       // Move elbow DOWN 45 degrees
  ↪  (startposition)

  // The sequence will then start again after the 10 s
  ↪  delay.
}

// ============  FUNCTIONS  ===============
// Functions to control each servo.
// INPUT: Desired angle.

// ========== HOW THESE FUNCTIONS WORK ========== //
// While angledifference does not equal to 0
// Check if current angle is less than desired angle
// If so, add +1 to current angle
//
// Else if current angle is greater than desired angle
// If so, subtract -1 to current angle
//
// Send new angle to servo
// Delay 20 milliseconds between every iteration to get
// a slower and more controlled servorotation.
// ============================================== //

// Elbow-servo
int elbowFun(int thetaElbow_desired) {
  while (thetaElbow-thetaElbow_desired != 0) {
    if (thetaElbow < thetaElbow_desired) {
      thetaElbow = thetaElbow + 1;
    }
    else if (thetaElbow > thetaElbow_desired) {
      thetaElbow = thetaElbow - 1;
    }
```

```
    sElbow.write(thetaElbow);
    delay(20);
  }
}

// Shoulder-servo
int shoulderFun(int thetaShoulder_desired) {
  while (thetaShoulder-thetaShoulder_desired != 0) {
    if (thetaShoulder < thetaShoulder_desired) {
      thetaShoulder = thetaShoulder + 1;
    }
    else if (thetaShoulder > thetaShoulder_desired) {
      thetaShoulder = thetaShoulder - 1;
    }
    sShoulder.write(thetaShoulder);
    delay(20);
  }
}

// Base-servo
int baseFun(int theta_desired) {
  while (thetaBase-theta_desired != 0) {
    if (thetaBase < theta_desired) {
      thetaBase = thetaBase + 1;
    }
    else if (thetaBase > theta_desired) {
      thetaBase = thetaBase - 1;
    }
    sBase.write(thetaBase);
    delay(20);
  }
}

// Wrist-servo
int wristFun(int theta_desired) {
  while (thetaWrist-theta_desired != 0) {
    if (thetaWrist < theta_desired){
      thetaWrist = thetaWrist + 1;
    }
    else if (thetaWrist > theta_desired) {
      thetaWrist = thetaWrist - 1;
    }
    sWrist.write(thetaWrist);
    delay(20);
```

```
  }
}

//Gripper-servo
int gripFun(int theta_desired) {
  while (thetaGrip-theta_desired != 0) {
    if (thetaGrip < theta_desired){
      thetaGrip = thetaGrip + 1;
    }
    else if (thetaGrip > theta_desired) {
      thetaGrip = thetaGrip - 1;
    }
    sGrip.write(thetaGrip);
    delay(20);
  }
}
```

A.2 Matlab Code

```
%% ======================================
%  Accuracy & Repeatability of robotic arm
%  using ISO 9283 International Standard.
%  Bachelor's Thesis in Mechatronics, KTH
%  Victor Runnquist & Carl-Victor Lidholm
%  2021-05-07
% ======================================

% ======== ERROR-VECTORS [mm] ============
% 20/30/40 stands for the delay in milliseconds
% between each angle rotation. This delay makes
% the servorotation slower.

% Error in x-direction:
X_20 = [-2 -2 -1 2 1 -3 2 -3 -4 -2];
% Error in y-direction:
Y_20 = [2 4 3 1 2 2 3 -2 2 5];

X_30 = [-3 0 -2 -3 2 1 2 3 -1 -1];
Y_30 = [-1 -3 -2 -2 1 -3 -2 -4 -2 -3];

X_40 = [-1 -1 2 0 -2 -2 1 2 1 2];
Y_40 = [2 -2 -1 -1 2 3 0 2 3 1];
```

```
n = 10;    % Number of repeated movements
xc = 0;    % x-coordinate of commanded position
yc = 0;    % y-coordinate of commanded position

% ============ ACCURACY  =============

% ============ DELAY(20) =============
% Mean error
xm20 = (1/n)*sum(X_20);
ym20 = (1/n)*sum(Y_20);
% Difference between mean & commanded position
APx20 = xm20 - xc;
APy20 = ym20 - yc;
% Positioning accuracy:
APp20 = sqrt((APx20).^2 + (APy20).^2)

% ============ DELAY(30) =============
% Mean error
xm30 = (1/n)*sum(X_30);
ym30 = (1/n)*sum(Y_30);
% Difference between mean & commanded position
APx30 = xm30 - xc;
APy30 = ym30 - yc;
% Positioning accuracy
APp30 = sqrt((APx30).^2 + (APy30).^2)

% ============ DELAY(40) =============
% Mean error
xm40 = (1/n)*sum(X_40);
ym40 = (1/n)*sum(Y_40);
% Difference between mean & commanded position
APx40 = xm40 - xc;
APy40 = ym40 - yc;
% Positioning accuracy
APp40 = sqrt((APx40).^2 + (APy40).^2)

% ====================================

% ========== REPEATABILITY ===========
% Repeatability for a given position is
% given by the value of RPl (radius of
% circle).
% ========== DELAY(20) ===============
lj20 = sqrt((X_20-xm20).^2 + (Y_20-ym20).^2);
```

```
lm20 = (1/n)*sum(lj20);
Sl20 = sqrt((1/(n-1))*sum((lj20-lm20).^2));

RPl_20 = lm20 + 3*Sl20
% ========== DELAY(30) ===============
lj30 = sqrt((X_30-xm30).^2 + (Y_30-ym30).^2);
lm30 = (1/n)*sum(lj30);
Sl30 = sqrt((1/(n-1))*sum((lj30-lm30).^2));

RPl_30 = lm30 + 3*Sl30

% ========== DELAY(40) ===============
lj40 = sqrt((X_40-xm40).^2 + (Y_40-ym40).^2);
lm40 = (1/n)*sum(lj40);
Sl40 = sqrt((1/(n-1))*sum((lj40-lm40).^2));

RPl_40 = lm40 + 3*Sl40


% ===================================
% ======= PLOTS =====================
figure(1)

viscircles([xm20,ym20], RPl_20,'Color','k');
axis equal
hold on

scatter(X_20,Y_20,'o','r')
xlim([-10 10])
ylim([-10 10])

ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

grid on
title('Error - 20 ms delay')
xlabel('x  [mm]');
ylabel('y  [mm]');

figure(2)

viscircles([xm30,ym30], RPl_30,'Color','k');
axis equal
```

```
hold on

scatter(X_30,Y_30,'o','r')
xlim([-10 10])
ylim([-10 10])

ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

grid on
title('Error - 30 ms delay')
xlabel('x  [mm]');
ylabel('y  [mm]');

figure(3)

viscircles([xm40,ym40], RPl_40,'Color','k');
axis equal
hold on

scatter(X_40,Y_40,'o','r')
xlim([-10 10])
ylim([-10 10])

ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

grid on
title('Error - 40 ms delay')
xlabel('x  [mm]');
ylabel('y  [mm]');
```

A.3 Acumen CPS Code

```
// =======================================
//  Accuracy & Repeatability of robotic arm
//  3D-model of robotic arm made with acumen CPS
//  Bachelor's Thesis in Mechatronics, KTH
//  Victor Runnquist & Carl-Victor Lidholm
//  28/03/2021
// =======================================

model Main(simulator) =
initially
 _3D = (Cylinder center = (0,0,-0.6) //Base, contains
  ↪  SwivelMotor
            length = 1
            radius = 1
            rotation=(3.1416/2,0,0)
            color = 0.7*white
        Box center = (0,0,-0.05) //part1
            length = 0.5
            width = 0.5
            height = 0.1
            color = 0.7*white,
        Box center = (0.3,0,0.9) //part1
            length = 0.1
            width = 0.5
            height = 2
            color = 0.7*white,
        Box center = (-0.3,0,0.9) //part1
            length = 0.1
            width = 0.5
            height = 2
            color = 0.7*white,
        Box center = (-0.4,0.5,2.1) //part2
            length = 0.1
            width = 0.5
            height = 2
            rotation=(-3.1416/4,0,0)
            color = 0.7*white,
        Box center = (0.4,0.5,2.1) //part2
            length = 0.1
            width = 0.5
            height = 2
            rotation=(-3.1416/4,0,0)
```

```
        color = 0.7*white,
    Box center = (-0.3,0.4,2.6) //part3
        length = 0.1
        width = 0.5
        height = 2
        rotation=(-3.1416/2,0,0)
        color = 0.7*white,
    Box center = (0.3,0.4,2.6) //part3
        length = 0.1
        width = 0.5
        height = 2
        rotation=(-3.1416/2,0,0)
        color = 0.7*white,
    Box center = (0,-0.55,2.6) //part3
        length = 0.5
        width = 0.5
        height = 0.1
        rotation=(-3.1416/2,0,0)
        color = 0.7*white
    Cylinder center = (0,0,1.6) //ShoulderMotor
        length = 0.5
        radius = 0.2
        rotation=(0,0,3.1416/2)
        color = 0.5*white,
    Cylinder center = (0,1.05,2.6) //ElbowMotor
        length = 0.5
        radius = 0.2
        rotation=(0,0,3.1416/2)
        color = 0.5*white,
    Cylinder center = (0,-0.3,2.6) //WristMotor
        length = 0.5
        radius = 0.2
        rotation=(-3.1416/1,0,0)
        color = 0.5*white,
    Box center = (0,-0.8,2.6) //gripper
        length = 0.4
        width = 0.4
        height = 0.4
        rotation=(-3.1416/2,0,0)
        color = 0.7*white
    Box center = (0,-1.3,2.4) //gripper
        length = 0.4
        width = 0.8
        height = 0.1
```

```
        rotation=(3.1416/7,0,0)
        color = 0.7*white
    Box center = (0,-1.3,2.8) //gripper
        length = 0.4
        width = 0.8
        height = 0.1
        rotation=(-3.1416/7,0,0)
        color = 0.7*white
    Cylinder center = (0.35,-0.8,2.6) //GripperMotor
        length = 0.3
        radius = 0.1
        rotation=(0,0,3.1416/2)
        color = 0.5*white
    )
```

Appendix B

Datasheets

B.1 Servo datasheet

Hextronik HX5010 - Twin Bearing Servo

Specifications

Modulation:	Analog
Torque:	**4.8V:** 96.00 oz-in (6.91 kg-cm)
Speed:	**4.8V:** 0.16 sec/60°
Weight:	1.38 oz (39.1 g)
Dimensions:	**Length:** 1.56 in (39.6 mm) **Width:** 0.79 in (20.1 mm) **Height:** 1.50 in (38.1 mm)
Motor Type:	(add)
Gear Type:	Plastic
Rotation/Support:	Dual Bearings
Rotational Range:	(add)
Pulse Cycle:	(add)
Pulse Width:	(add)
Connector Type:	(add)

Brand:	hexTronik
Product Number:	(add)
Typical Price:	5.95 USD
Compare:	add+

MS-6-40

Servo Motor MS-6-40

- Dimensions: 40.8 x 20.1 x 38 mm
- Operating Speed: 0.18sec/60degree (4.8V), 0.16sec/60degree (7V)
- Stall Torque: 5kg.cm/69.56oz.in(4.8V), 6kg.cm/83.47oz.in(6V)
- Operating Voltage: 4.8V~6V
- Control System: Analog
- Direction: CCW
- Operating Angle: 120degree
- Required Pulse: 900us-2100us
- Bearing Type: 2BB
- Gear Type: Plastic
- Motor Type: Metal
- Connector Wire Length: 30 cm

TRITA ITM-EX 2021:40