EXAMENSARBETE INOM MASKINTEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, SVERIGE 2021

Construction of a Selective Compliance Articulated Robot Arm

Konstruktion av en utvalt eftergivlig robotarm

And evaluation of its accuracy
Och utvärdering av dess precision

ANTON LABBÉ

BENJAMIN STRÖM

KTH
SKOLAN FÖR INDUSTRIELL TEKNIK OCH MANAGEMENT

Construction of a Selective Compliance Articulated Robot Arm

And evaluating its accuracy

ANTON LABBÉ & BENJAMIN STRÖM

Bachelor Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:41

Abstract

The concept of a robotic manipulator is widely used throughout many industries. In this project, a manipulator of the type SCARA, *selective compliance articulated robot arm*, is constructed. The aim was to examine how such a robot could be constructed using 3D-printing and how accurate it would be. Other than 3D-printing, parts in the form of guiding rods, lead screw, bearings, pulleys and timing belts were used. Together with a microcontroller, the robot operates using three stepper motors. In the end it resulted in a SCARA with reasonable accuracy considering the methods used, more specifically the largest average error was 3.6 cm in the X direction and 2.3 cm in the Y direction. The largest drawback of the final construction was the negative balance between tightening the belts and friction in the inner joint. Tightening the belts meant larger friction and thereby undesired movement properties. Doing the opposite meant that the belts could start slipping and enabled backlash.

Keywords: Mechatronics, 3D-printing, Robotics, Robot, Arduino, Inverse kinematics

Referat

Konstruktion av en Selective Compliance Articulated Robot Arm

Konceptet av en robotarm används brett inom många industrier. Detta projekt syftar till att konstruera en robot av typen SCARA, *selective compliance articulated robot arm*. Målet var att undersöka hur en sådan robot kan 3D-printas och dess precision. Förutom 3D-printade delar användes även guidestänger, kullager, kamremmar och remskivor. Robotens rörelser styrs tillsammans med en mikrokontroller och tre stegmotorer. Med tillvägagångssätten i åtanke resulterade projektet in en SCARA med rimlig precision. Mer specifikt var medelfelet 3.6 cm i X-led och 2.3 cm i Y-led. Den största nackdelen med den slutgiltiga konstruktionen var den negativa jämvikten mellan att spänna kamremmarna och friktionen i den inre armleden. Att spänna kamremmarna innebar en ökning i friktion och därmed oönskade rörelseegenskaper. Att göra tvärtom innebar att bältena löpte större risk att glida ur och möjliggjorde dödgång.

Nyckelord: Mekatronik, 3D-printing, Robotik, Robot, Arduino, Invers kinematik

Acknowledgements

We would like to thank Nihad Subasic for his lectures and help throughout the project. We would also like to thank Amir Avdic for his general guidance and Jan Stamer for helping us machine our parts. Lastly we would also like to thank our peers for giving us feedback on our work, especially Samuel Stenow for the helpful thoughts he gave us regarding software and Rayan Alnakar for helping out with 3D-printing.

Anton Labbé & Benjamin Ström
Stockholm, May 2021

Nomenclature

List of acronyms and abbreviations

- CAD - Computer Aided Design
- CNC - Computer Numerical Control
- DC - Direct Current
- DOF - Degrees Of Freedom
- GUI - Graphical User Interface
- LCD - Liquid Crystal Display
- MOSFET - Metal Oxide Semiconductor Field Effect Transistor
- PLA - Polylactic Acid
- PM - Permanent Magnet
- PTC - Postive Temperature Coefficient device
- SCARA - Selective Compliance Articulated Robot Arm
- SD - Secure Digital
- STL - STereoLithography

Contents

1 Introduction **1**
1.1 Background 1
1.2 Purpose 1
1.3 Method 1
1.4 Scope 2

2 Theoretical background **3**
2.1 Control theory 3
2.2 Components 4
2.2.1 Arduino Mega 2560 4
2.2.2 Arduino shield 4
2.2.3 Stepper motor 4
2.2.4 Stepper motor driver 6
2.2.5 Bearings 6
2.3 3D-printing 6
2.4 Robot kinematics 7

3 Execution **8**
3.1 Software 9
3.1.1 Computer aided design - CAD 9
3.1.2 Simulation 9
3.1.3 Code 10
3.2 Components 10
3.2.1 Base 10
3.2.2 Mid-plate 11
3.2.3 Arm 11
3.2.4 Arm mount 11
3.2.5 Top-plate 11
3.2.6 Electronics 11
3.2.7 Guiding rods 12
3.2.8 Bearings 12
3.3 Accuracy 12

4 Results **14**
4.1 Non-printed parts 14
4.2 3D-printing a robot 14
4.2.1 Base 14
4.2.2 Top-plate 15
4.2.3 Mid-plate 15
4.2.4 Arms 16
4.3 How accurate is the arm? 17
4.3.1 Important aspects 17

5 Discussion **19**
5.1 Hollow axis 19
5.2 Arm 19
5.3 Improving the accuracy 20

Bibliography **21**

Appendices **22**

A KH54KM2-801 Datasheet **23**

B Usongshine Nema 17 (17HS4401) Datasheet **26**

C Arduino Code **30**

D Acumen model of Z-axis movement **37**

E Acumen model of arms **39**

List of Figures

2.1 Open and closed loop system block diagrams [Created with Notability] . 3
2.2 Arduino mega 2560 [5] 4
2.3 A two-phase PM stepper motor circuit schematic diagram [8] 5
2.4 DRV8825 pinout[9] 6

3.1 Rendered image of the full SCARA robot [Created with KeyShot 8] . . 8
3.2 A model of the two link planar system [Created with Notability] 9
3.3 Code flowchart [made with draw.io] 10
3.4 Schematic diagram for electronics [Created with Fritzing] 12

4.1 The finalized baseplate [Created with KeyShot 8] 15
4.2 Configuration of the smaller stepper motors. [created in KeyShot 8] . . 16
4.3 Magnification of inner and outer joint configuration. [Created with KeyShot 8 & Adobe Photoshop] 16
4.4 Distribution of coordinates achieved in practice [Created with Desmos] . 17

List of Tables

4.1 The difference between theoretical and practical coordinate achieved . . 17

Chapter 1

Introduction

1.1 Background

The project consisted of the construction of a non-scale serial manipulator in the form of a selective articulated compliance robot arm (SCARA). A SCARA is a robot of cylindrical type, meaning it performs tasks in its circular vicinity and is generally applied for pick-and-place like machine operations. In combination with the construction, the aspiration was also to measure the accuracy of the arm. The results presented are applicable when considering any type of construction using the same manufacturing methods and mechanisms. [1]

1.2 Purpose

The project looked to create a robotic arm following the SCARA principle. The research questions that were answered are:

- How can a SCARA robot be constructed using 3D-printed parts?
- When building a SCARA, what are the most important aspects to take into consideration?
- How accurate is the arm when placed in a coordinate system?

1.3 Method

The project was divided into three parts. First of all the general design of the robot required decision, allowing for an overview of the size and construction method. This step would constitute the outlines of the project and involved a CAD model, the placing of critical components and an idea of how movements should be transferred. With that being done, the second step was programming the robot arm, i.e. modelling its movements. Some adjustments would have to be made along the way, for example changes in design or substitution of components. The last part was testing, in which the robot was placed in a cartesian coordinate system and the differences between the requested and the achieved coordinate value in the x and y directions were evaluated.

1.4 Scope

As time and resources were limited due to being part of a course (MF133X), the project mainly aimed to deliver a working prototype of a SCARA. The design, for example, was simplified and the robot arm did not have a commercial appearance. Since the arm was meant for showcasing one variant of a SCARA, and not for lifting especially heavy or extremely delicate things, little to no solid mechanical calculations were made.

Chapter 2

Theoretical background

2.1 Control theory

Control theory is about making systems behave the way you want them to. Systems can be controlled in many ways, for example by controlling angles, speed, torque etc [2]. These variables in turn can mainly be controlled in two ways, as either open loop, or closed loop systems. In an open loop system the control signal is not measuring the condition of the output, while in closed loop systems the control signals are dependent on feedback from the output, see figure 2.1.

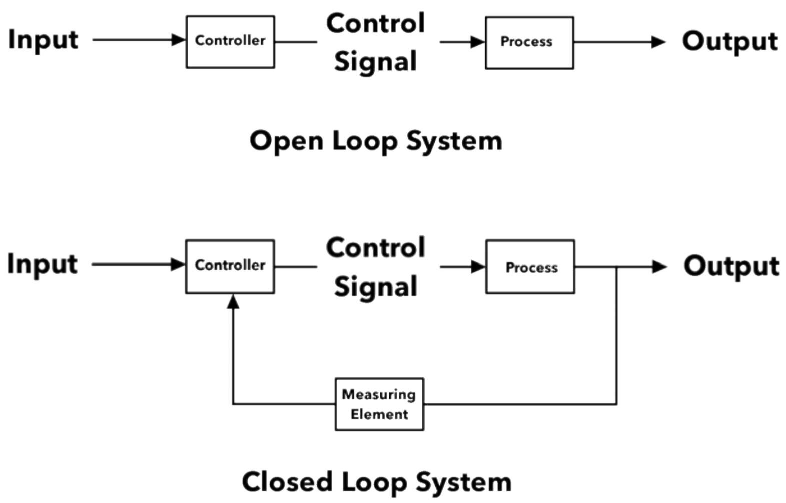


Figure 2.1: Open and closed loop system block diagrams [Created with Notability]

2.2 Components

The following sections describe different components that are relevant to the construction of the developed SCARA.

2.2.1 Arduino Mega 2560

Arduino is an open source microcontroller platform that was developed by experts within electrical engineering in order to make use of electronics and electrical components easier for the less experienced consumer. It is widely used in both simpler and more complex projects due to its versatility and affordable price. The modern Arduino boards are available in different sizes, with the largest one being the Arduino MEGA 2560 it consists of 54 digital input/output which is seen in figure 2.2. [5]

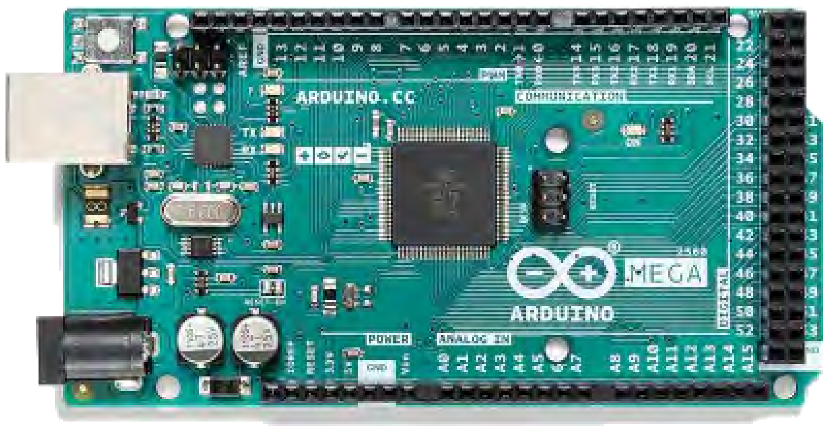


Figure 2.2: Arduino mega 2560 [5]

2.2.2 Arduino shield

Shields are modular circuit boards that can be mounted on top of the Arduino to extend its capabilities or to ease implementation of certain components with the Arduino. The shield used is called RAMPS 1.4 and is part of the ”RepRap” project which aims to simplify replication of 3D-printer technology at home. It has support for up to 5 stepper motors, MOSFET switches for controlling heating elements and other DC loads, power supply with PTC fuses and connections for limit switches, servos, SD card readers, LCD and more. [6]

2.2.3 Stepper motor

A stepper motor generally consists of a rotor that is a gear shaped permanent magnet which is surrounded by the windings of a stator. The windings are alternately powered to incrementally rotate the rotor on which the shaft is attached. This results in the ability to precisely control the angular position of the shaft[7], see figure 2.3.

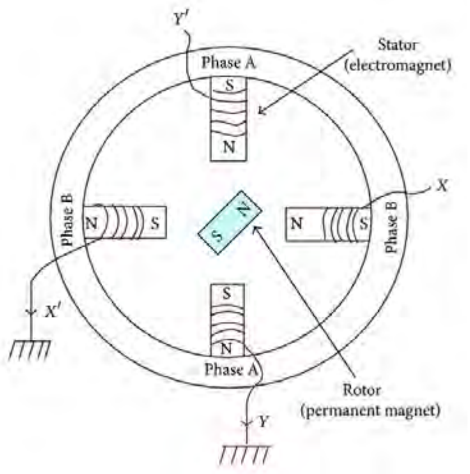


Figure 2.3: A two-phase PM stepper motor circuit schematic diagram [8]

2.2.4 Stepper motor driver

Due to the fact that the powered phase must be alternated to rotate the shaft (view figure 2.3), control electronics that allow for rapid changes in direction and amplitude of the current in the windings are necessary. Such electronics are known as drivers and are available in many different forms. In figure 2.4 the motor driver DRV8825 is showcased.

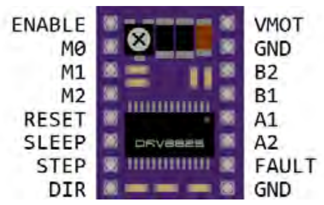


Figure 2.4: DRV8825 pinout[9]

2.2.5 Bearings

Bearings are machine elements used keep components such as a fixed axis in place while also relieving its load. An often desirable trait of a bearing is its low friction, allowing the connected component to move with ease. Bearings are classified by their allowed range of motion or in what direction the load is being applied. [11]

2.3 3D-printing

3D printing is a manufacturing process in which material is laid down layer by layer to form a three-dimensional object. It is useful for high quality control and precision manufacturing [3]. In this project 3D printing is used to create the chassis for the SCARA with the use of Polylactic Acid (PLA) as material. Although this is not the most durable material it is suitable for this type of project. Before printing, settings such as layer thickness, infill and support material needs to be specified. Infill determines how hollow the print will be, from a scale 0-100, where 100 is completely solid. Support material ensures the printability and can for example help prohibit part deformation. It is worth mentioning that the stepper motors run the risk of heating up, and as PLA has a glass transition temperature at around 60° Celsius, the temperature of the stepper motors has to be monitored.[4]

2.4 Robot kinematics

A SCARA normally has four degrees of freedom (DOF), in other words four independent parameters which describes its state [12]. More specifically a SCARA can be modelled as a two link planar system able to move vertically. To track its position the concept inverse kinematics will be used, which in turn utilizes coordinate transformation from polar to cartesian. Inverse kinematics is used when calculating what angles the joints need to assume for the end-effector to reach the desired position. [13] The modelling is further explained in section 3.1.

Chapter 3

Execution

This chapter will describe the working process, from start to finish. It includes both the practical and theoretical work needed for the completion of the project. An overview of the robot is seen in figure 3.1.

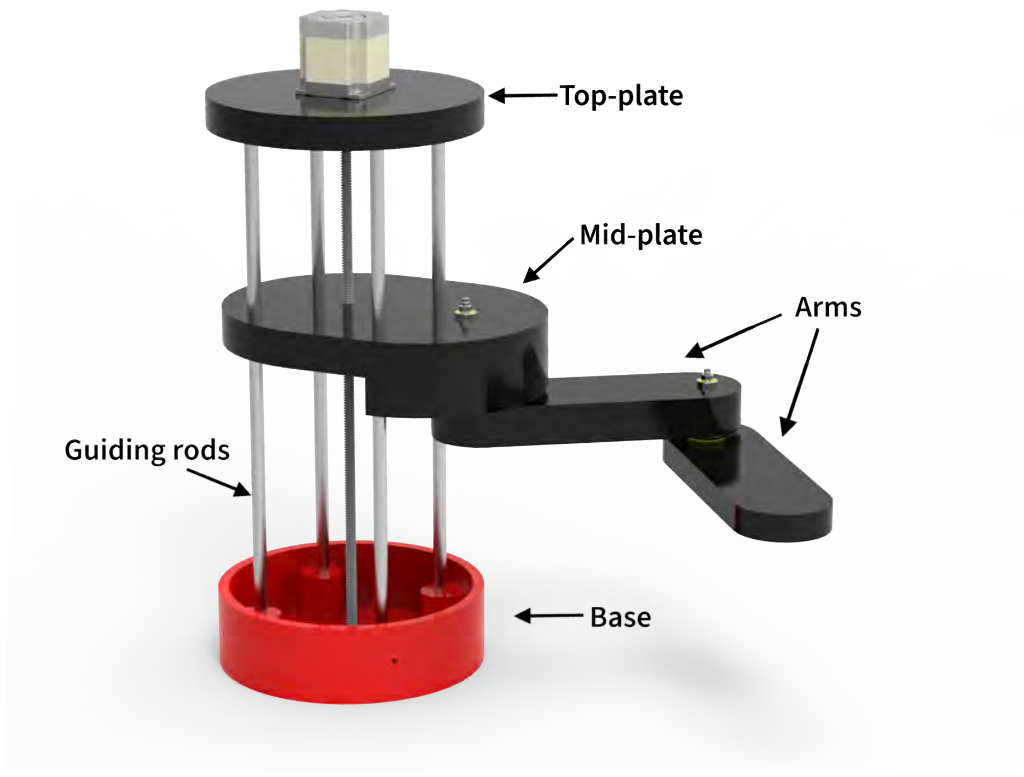


Figure 3.1: Rendered image of the full SCARA robot [Created with KeyShot 8]

3.1 Software

3.1.1 Computer aided design - CAD

The visual and partly functional concept was created in Siemens' SolidEdge software. This enabled a perspicuous view of the layout and size of the SCARA, for example the placement of the stepper motors and gearing of the arms. After a clear overview had been made, a more detailed version could be made, ready to be 3D-printed.

3.1.2 Simulation

Before any code was developed, the movement patterns of the arm were simulated with the simulating tool Acumen, see code in Appendix D and E. Mainly because there was no hardware available for software testing until the near end of the project. Figure 3.2 was used to derive the inverse kinematic expressions for the arms joints. In other words, what angle each joint would need to assume in order to reach a certain position with the end effector. This resulted in the following equations:

$$\Theta_2 = \arccos \frac{x^2 + y^2 - l_1^2 + l_2^2}{2l_1l_2} \tag{3.1}$$

$$\Theta_1 = \arctan \frac{y}{x} - \arctan \frac{l_2 \sin \Theta_2}{l_1 + l_2 \cos \Theta_2}, \tag{3.2}$$

where x and y are the coordinates of the end effector and l_1 and l_2 are the lengths of the inner and outer arm. As seen in figure 3.2, Θ_1 and Θ_2 are the angles between the x-axis and l_1 and between l_1 and l_2, respectively. Combined with the movement described in figure 3.2, the robot is able to move vertically along the z-axis, giving it three DOF.

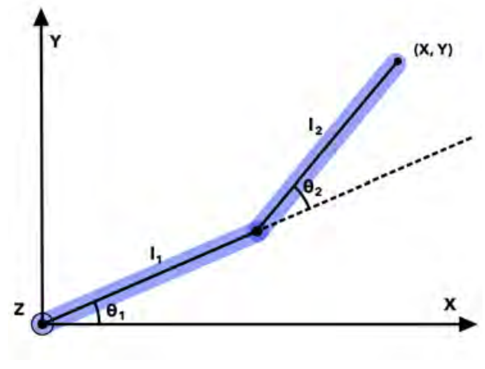


Figure 3.2: A model of the two link planar system [Created with Notability]

3.1.3 Code

The code is based upon the principle of inputting a coordinate (X,Y) and then performing the required actions to reach said coordinate. Because of the fact that a stepper motor can not simply start rotating at a requested speed, an object oriented library featuring retardation and acceleration was used, namely AccelStepper library[14]. It also featured simple ways of controlling several stepper motors at once. With the basics of how to run one or more stepper motors cleared up, the flowchart for the robots code is seen in figure 3.3. The full code is presented in Appendix C.

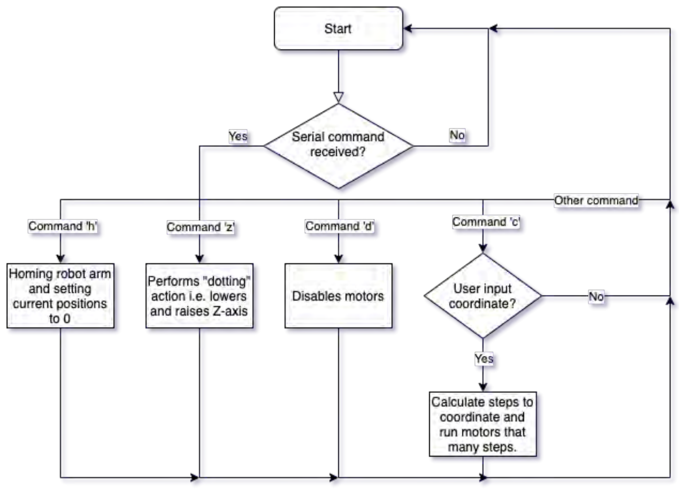

Start
Serial command received?
Yes
No
Other command
Command 'h'
Command 'z'
Command 'd'
Command 'c'
Homing robot arm and setting current positions to 0
Performs "dotting" action i.e. lowers and raises Z-axis
Disables motors
User input coordinate?
No
Yes
Calculate steps to coordinate and run motors that many steps.

Figure 3.3: Code flowchart [made with draw.io]

3.2 Components

The main components are described. All parts were 3D-printed except for the stepper motors, guiding rods, lead screw, pulleys and timing belts. The 3D-printers used were from the brand Ultimaker together with their software Ultimaker Cura which utilized SolidEdge files, in STL format.

3.2.1 Base

The base was initially the foundation of the robot, housing the big stepper motor. It was also the mounting place for the guiding rods along with a protrusion fitted with a bearing for the lead screw to rotate in. As the project progressed, the big stepper motor was mounted on the top plate instead.

3.2.2 Mid-plate

This was the only part that needed to be divided into two halves to be able to 3D-print. The mid-plate, together with the arm mounting plate, were the mounting place for three crucial parts; the lead screw and the two smaller stepper motors. This component also enabled the arms to move vertically as it was connected to the lead screw.

3.2.3 Arm

This was the most mechanically complex subsystem of the robot to design. From the beginning, one of the main ideas was to centralize the two small stepper motors close to the Z-axis and not have them mounted on the arm. The reason being to minimize the weight of the arm and thus to reduce their static, tilting torque. To solve this, the outgoing axis' driven by the small stepper motors had to be rotating coaxially, which was solved by developing a ”hollow axis”.

3.2.4 Arm mount

This part was connected directly to the mid-plate and together they served as a mounting point for the arm. They also created housing for the components operating the arm. In detail those components were both of the smaller stepper motors, with attached pulleys and corresponding timing belts, radial and axial bearings and the hollow axis.

3.2.5 Top-plate

The top-plate was initially the part with the least functionality, with the sole purpose of holding the guiding rods together, along with the lead screw. As iterations of the robot were made, it became apparent that mounting the big stepper motor on the top-plate was a better solution than having it mounted on the base.

3.2.6 Electronics

The robot housed two types of stepper motors, one bigger from Japan Servo Co ltd., named KH56KM2 with the purpose of controlling the vertical movement (datasheet in Appendix A) and two smaller ones from Usongshine, called 17HS4401 (Nema 17) which were used for rotating the arm (datasheet in Appendix B). All three stepper motors have 200 steps per rotation meaning they move in 1.8 degree increments but since the driver DRV8825 is used, microstepping up to 1/32 is possible, which would result in 6400 steps per revolution. The stepper motors were used in each joint of the robot as well as for the vertical movements of the arm. Since the motor moves in steps, with no feedback on what position it is currently at, a stepper motor by itself is an open loop system. Therefore microswitches were used to determine the home position of the arm. The operations were computed by the Arduino mega 2560 with a Ramps 1.4 shield mounted on top. The devices were connected according to figure 3.4.

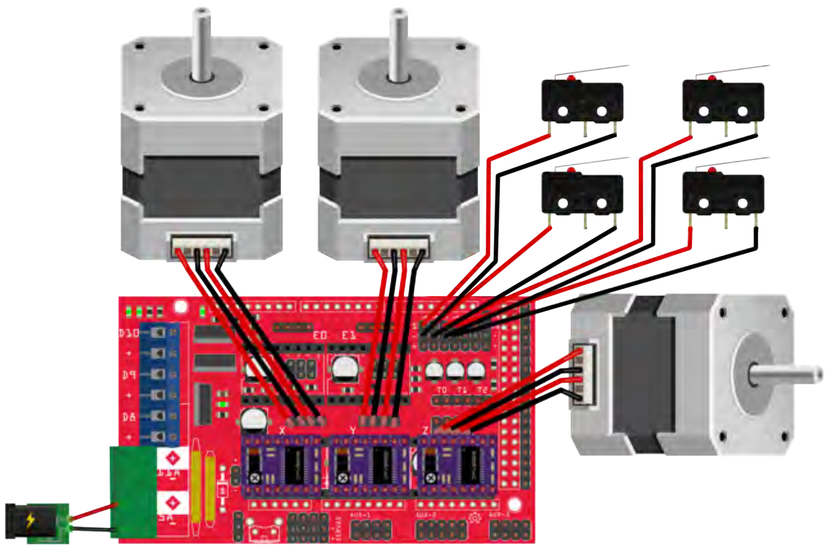


Figure 3.4: Schematic diagram for electronics [Created with Fritzing]

3.2.7 Guiding rods

The main function of the guiding rods was to distribute the load and serve as a rail for the mid-plate to move along. They were initially made from aluminium for cost and availability reasons. As development progressed, the choice of material changed to stainless steel due to aluminium being too soft for the linear bearings. Also the stainless steel's polished surface allowed for lower friction.

3.2.8 Bearings

Three types of bearings were used in the project. The guiding rods used four linear bearings, making sure the vertical movement was smooth. The arms used both radial ball bearings and axial ball bearings for two reasons. Firstly because it made the arm rotate as smoothly as possible and thus reducing the torque needed from the smaller stepper motors and the second reason being accuracy when rotating.

3.3 Accuracy

To evaluate the robots precision, an accuracy test was performed. The robot was placed in a large coordinate system, scaled the same length as the robots fully extended arm (35 cm in each direction). The inner joint was placed in origo and the arm was equipped with a pencil as end effector. The test was performed as a sequence of movements, where each movement was aimed at a theoretical coordinate. Each movement ended with the arm lowering itself vertically and a dot being made on the paper where the robot placed the theoretical coordinate. The sequence was repeated six times. The accuracy was evaluated as the difference in the x and y directions between the theoretical coordinate and the coordinates achieved in practice.

Chapter 4

Results

4.1 Non-printed parts

Unlike the printed parts, the guiding rods, lead screw, pulleys and timing belt needed to be of higher tolerance. When trying to print the pulley it resulted in the gear teeth not being deep enough which led to the timing belt not being able to grip it. In combination with tolerance, 3D-printing is also constrained by the strength and durability of the printed materials, in this case PLA. Due to this, parts that stabilized the robot, i.e. the guiding rods and the lead screw, demanded a more durable material like metal. Lastly, the timing belts could not be printed either as they needed to be flexible, an attribute PLA does not possess.

4.2 3D-printing a robot

As briefly mentioned in the execution the majority of the components were 3D-printed, mainly due to the availability of 3D-printers but also because of the customization 3D-printing entails. This allowed for new parts to easily be changed and re-produced if any error occurred or if a new sub-part needed to be incorporated. While customizability was high, 3D-printing also had some drawbacks or quirks. For example, a printed hole always had to be designed a few tenths of a millimeter larger to fit the real object supposed to pass through said hole.

4.2.1 Base

Initially the base housed the big stepper motor, but as testing begun, it became clear that the robots vertical movement was unstable with quite significant vibrations. The base contributed to this by having weak mounting holes for the guiding rods. Also, when trying to enlarge the holes manually using a drill press, the hole structure would melt due to the heat caused by friction. In attempt to eliminate the vibrations, the robot was flipped upside down. Having the big stepper motor push down on the construction while also allowing the Z-axis to align itself through gravity improved the performance and the motor was consequently moved to sit on the top plate instead. At the same time, the base was re-designed and given more robust mounting holes where the guiding rods could be screwed in place. Also, the lead screw for the Z-axis was given space in the bottom to be fitted with a bearing, enabling less friction and further simplifying alignment throughout the axis. See figure 4.1

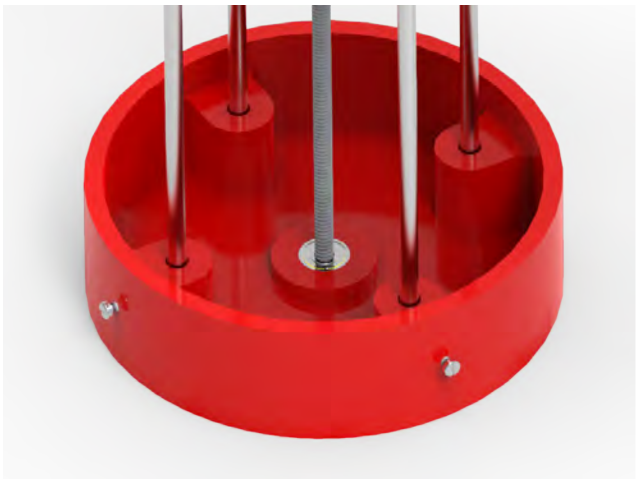


Figure 4.1: The finalized baseplate [Created with KeyShot 8]

4.2.2 Top-plate

As seen earlier, the developments to the top-plate were relatively sparse. The main issue was similar to the one of the base, where the mounting holes for the guiding rods lacked stability. Other than that, the top-plate was fitted with holes for screwing the big stepper motor in place, as its position changed from being housed in the base to being housed on top of the top-plate.

4.2.3 Mid-plate

As mentioned in 4.1.1, the robot suffered from vibrations in its first iteration and this was mainly due to the mid-plate. More specifically the vibrations originated from the guiding-rod holes being of very low tolerance. The low tolerance of the holes caused the mid-plate to wobble its way up the guiding rods once the Z-axis was powered. The inaccuracies were a property obtained by drilling the holes manually. Just like in the case with the base plate, the PLA started melting when trying to enlarge the holes. To prevent the vibrations by increasing the standard of the holes, linear bearings were installed together with higher tolerance guiding rods, stainless steel instead of aluminum. The linear bearings were also stabilized with a flange to reinforce their desired vertical orientation. Together with the arm mount it housed the two smaller stepper motors according to figure 4.2.

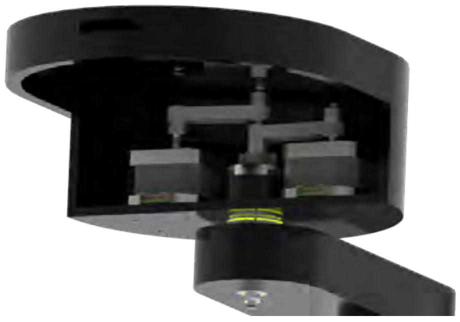


Figure 4.2: Configuration of the smaller stepper motors. [created in KeyShot 8]

4.2.4 Arms

This subsystem demanded many iterations before resulting in a working solution. In the final version, the two smaller stepper motors were fitted close to the rotating axis'. As mentioned in 3.2.3, a coaxial solution named the "Hollow Axis" solved the issue of having the stepper motors centralized. It served two purposes; to move the inner arm and to be a passage for the inner axis to move the outer arm, see figure 4.3. The hollow axis transfers its torque via a simple spline shaft, formed as a cross. The stepper motors and axis' were connected via GT2 pulleys and corresponding timing belts. Two SKF Explorer 625 radial ball bearings were placed on the top and bottom of both arms making it easier for the threaded axis' to rotate. Another 6002RS radial ball bearing were fitted inside the arm mount centering and fixating the hollow axis. Lastly, a pair of axial bearings were fitted to reduce the friction between the arm mount, inner and outer arm.

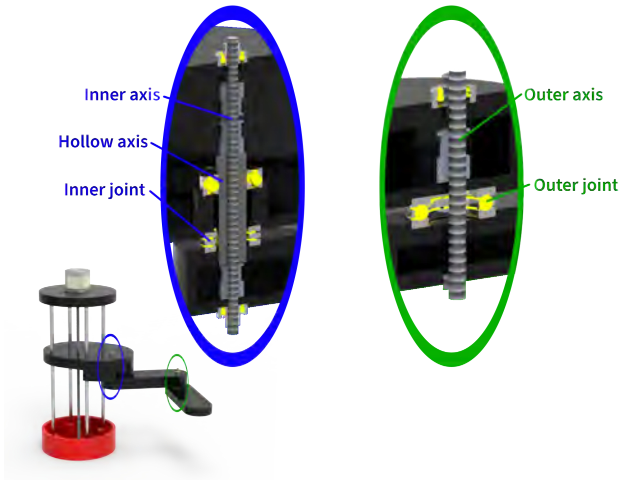


Figure 4.3: Magnification of inner and outer joint configuration. [Created with KeyShot 8 & Adobe Photoshop]

4.3 How accurate is the arm?

Table 4.1 presents the data obtained from the accuracy test. The coordinates in bold represents the theoretical coordinates. Other than the differences between the theoretical and practical value, average distribution for each coordinate and for each sequence was calculated, see Figure 4.4.

	Coordinates						
Difference [cm]	**(5,30)**	**(15,15)**	**(30,5)**	**(-5,30)**	**(-10,25)**	**(-35,0)**	**Average**
1. (Δx, Δy)	(1.3, -2.1)	(0.8, 0.7)	(-1.3, 2.2)	(5.0,-1.8)	(-6.0, 0.9)	(-0.5, -2.0)	(-0.1, 0.4)
2. (Δx, Δy)	(1.6, -2.4)	(-0.7, -1.3)	(-4.5, 0)	(2.3, -0.3)	(1.1, 0.9)	(-0.4, 3.3)	(-0.1, 0)
3. (Δx, Δy)	(1.3, -1.2)	(-0.6, -1.0)	(-4.2, 0.7)	(2.9, -0.2)	(1.7, 1.0)	(-0.3, 2.9)	(0.1, 0.4)
4. (Δx, Δy)	(2.0, -2.3)	(-0.3, -1.5)	(-3.9, -0.4)	(2.0, 0.1)	(0.9, 1.3)	(-0.4, 4.1)	(0.1, 0.2)
5. (Δx, Δy)	(1.0, -1.8)	(-0.6, -0.5)	(-3.7, 1)	(3.3, 0)	(2.2, 1.1)	(-0.3, 2.2)	(0.3, 0.3)
6. (Δx, Δy)	(1.2, -2.1)	(-0.6, -2.2)	(-4.1, 0.5)	(0.6, 0.8)	(2.0, 2.5)	(-0.5, 3.5)	(-0.2, 0.5)
Average	(1.4, -2.0)	(-0.4, -1)	(-3.6, 0.7)	(2.7, 0.2)	(0.3, 1.3)	(-0.4, 2.3)	-

Table 4.1: The difference between theoretical and practical coordinate achieved

Every theoretical coordinate and its corresponding practical coordinates are represented by the same colour. Filled circles represents the theoretical value while the unfilled circles represents the practical coordinates achieved.

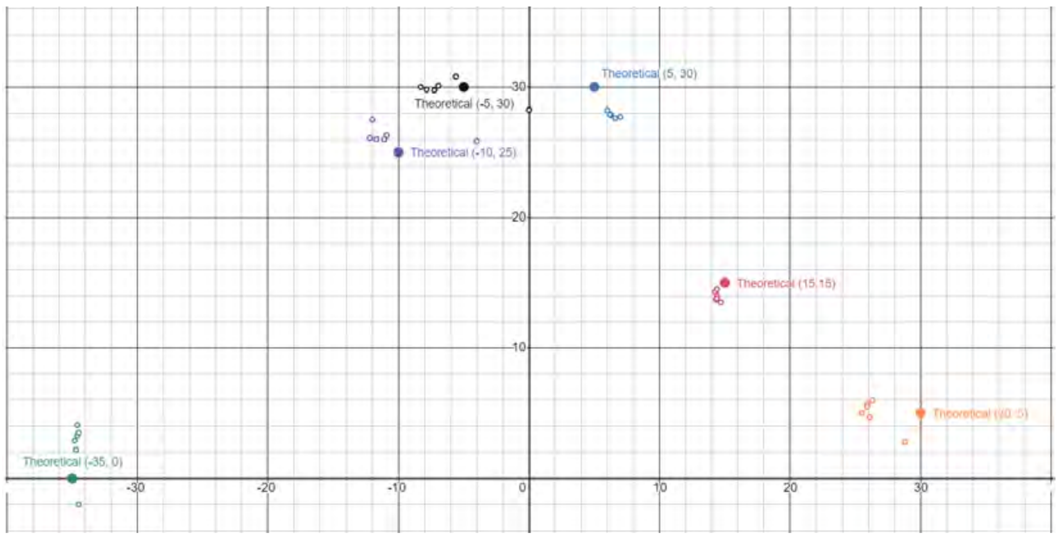


Figure 4.4: Distribution of coordinates achieved in practice [Created with Desmos]

4.3.1 Important aspects

Throughout the project, the different aspects of constructing a SCARA were considered. In retrospect the idea of centralizing the masses of the motors is something that should not have been prioritized. Although it works, a solution like this is not needed if the motor for the Z-axis is overqualified as in the case of this report. Aspects that came to light as being more important than originally thought were the placement of the microswitches, as they determined the home position of the whole robot and hence the starting position for all of its movements. In the case of this project, the microswitches were placed on the robot solely for the purpose of testing the accuracy and their placement was not considered a priority. In making a commercially viable SCARA however, the microswitches would have to be prioritized, as they can affect and limit the movements of the arm greatly.

Chapter 5

Discussion

In the end, the result was a functional SCARA, built completely from scratch and with reasonable accuracy. However there are still flaws and improvements to be discussed.

5.1 Hollow axis

One of the more central parts and a concept that more or less was kept from the beginning. When realizing the concept and throughout implementation, testing of the part revealed several flaws. To begin with the 3D-printers did not have the accuracy to print a GT2 pulley incorporated in the axis itself as the groove/teeth became too shallow. This was the same reason why the hollow axis transferred its torque via the cross-shaped spline, and not a full multi teeth spline. Furthermore, because of the fragile size of the component, processing it was deemed inconvenient. Another issue was found when putting the robot together, or more specifically when tightening the timing belts. If the belts were tightened too much the inner axis would rub against the hollow axis, creating enough friction to rotate the hollow axis, and thus the inner arm, unintentionally. To solve this a radial bearing could have been fitted inside the hollow axis and around the inner axis and by that drastically reduced the friction.

5.2 Arm

The first and perhaps easiest improvement would be changing all the radial bearings to axial ones, as the bearings predominantly are supporting axial load. The two reasons for not doing this from the beginning was supply and cost. The utilized bearings were from school stock and therefore free and directly accessible. If the robot was to be used for pick and place tasks, further changes or upgrades would be strengthening the joints of the arm. Under load, the components affected by the torque are mainly the joints, especially when fully extended. Another concept which would demand joint reinforcement would be finding an alternative to the hollow axis. The natural way of doing this would be connecting the second smaller stepper motor directly to the joint between the inner and outer arm. This would make assembling the robot easier and also increase transmission efficiency of the outer arm, at the cost of strengthening the joints. Mounting the smaller stepper motors this way would also make positioning and programming easier as the outer arm would follow the movements of the inner arm. Another easy fix would be moving the arm slightly outwards from the center due to the inner axis hitting the top-plate and base when the arm is at its lowest or highest point.

5.3 Improving the accuracy

Considering the methods used and the fact that the robot was not intended for industrial production, its accuracy is reasonable. The accuracy could be improved in many ways. Microstepping would for example lower the 1.8° per step by 1/32 of that and hence make the movements more delicate. In the current version, the inner arm has a tendency to move slightly due to the inner axis rubbing against the hollow axis. Less dependent movement could be achieved if the hollow axis was allowed to rotate with less friction, for example by fitting a radial bearing as described in 5.2. The accuracy could also be improved by tightening the timing belts and by that reducing slippage. However, as previously stated, in the final construction tightening the belts came with the undesired property of increasing the friction between the inner and hollow axis'. The current tension in the belt requires a few extra steps in the demanded direction before transmission of rotation. This is known as backlash. As seen in table 4.1 when moving from coordinate (15, 15) to coordinate (5,30), the error in the x direction grows. The outer arm has then had the belt tightened in one direction when moving to (15,15) and is then rotated outwards in trying to reach (5,30) which is further away from the origin than (15,15). As just stated, this means that the stepper motor will ”eat up” some of the steps required while the belt is tensioned in the opposite direction. This is further seen in figure 4.4 at coordinate (30,5), where the unfilled circles indicate that the outer arm did not move as far as it should have. Supposing the problem regarding friction between the inner axis and the hollow axis did not exist, the belts would have been tightened to a point where the arm could reverse its rotational direction without needing to perform any extra steps.

Bibliography

[1] R. Hagelberg, MG1002 Automatiseringsteknik, 16.1 ed. Stockholm: KTH Industriell Produktion, pp. 391–392.

[2] Torkel Glad & Lennart Ljung. *Reglerteknik - Grundläggande teori*. swe. Lund: Studentlitteratur AB, 2018. ISBN: 978-91-44-02275-8

[3] Joan Horvath & Rich Cameron. *Mastering 3D Printing*. eng. Springer International Publishing, 2020. ISBN: 978-1-4842-5842-2

[4] D. Garlotta, "A Literature Review of Poly(Lactic Acid)," Journal of Polymers and the Environment, vol. 9, no. 2, pp. 63–84, 2001, doi: 10.1023/a:1020200822435.

[5] Arduino 2018. *Arduino official website*. Accessed 12 February 2021. URL: `https://www.arduino.cc/en`

[6] Johnny Russel. *RAMPS 1.4*. Accessed 24 March 2021. URL: `https://reprap.org/wiki/RAMPS_1.4`

[7] Marc Bodson, John N. Chiasson, Robert T. Novotnak & Ronald B. Rekowski. *High-Performance Nonlinear Feedback Control of a Permanent Magnet Stepper Motor*. eng.

[8] Stefanos Theodoropoulos, Dionisis Kandris, Maria Samarakou and Grigorios Koulouras *Fuzzy regulator design for wind turbine yaw control*. Eng. Hindawi Publishing Corporation, The Scientific World Journal 2018. Volume 2014, Article ID 516394, 9 pages.

[9] Texas Instruments 2014. *DRV8825 Stepper Motor Controller IC*. Accessed 9 March 2021. URL:`https://www.ti.com/lit/ds/symlink/drv8825.pdf?ts=1617921558126&ref_url=https%253A%252F%252Fwww.google.com%252F`

[10] Sustek Michal et al. *DC motors and servo-motors controlled by Raspberry Pi 2B*. eng. In: MATEC Web of Conferences 125 (2017). Last access: 15/2-2021. URL: `https://doaj.org/article/6ed186431bbc43509e5b90491131fd01`

[11] Bernard J. Hamrock & William J. Anderson. 1983. Rolling-Element Bearings. *Nasa Reference Publication.* 1105. URL:`http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.693.8552&rep=rep1&type=pdf`

[12] John J Craig. *Introduction to Robotics: Mechanics Control.* eng. Reading, Mass.: Addison-Wesley, 1986. ISBN: 978-1-292-04004-2

[13] Taha, Walid M., Taha, Abd-Elhamid M. & Thunberg, Johan. *Cyber-Physical Systems: A Model-Based Approach.* eng. Springer International Publishing, 2021. ISBN: 978-3-030-36071-9

[14] M. McCauley, “AccelStepper: AccelStepper library for Arduino”. Accessed 15 April 2021. URL: `https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html`

Appendix A

KH54KM2-801 Datasheet

A Nidec Group Company SERVO

2-Phase Hybrid Stepping Motor 1.8°

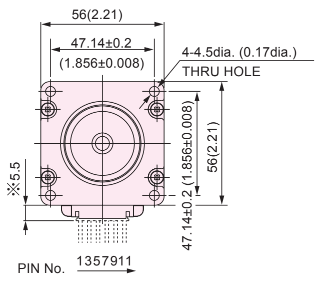
KH56 series 800 type

HIGH TORQUE, LOW VIBRATION AND LOW NOISE

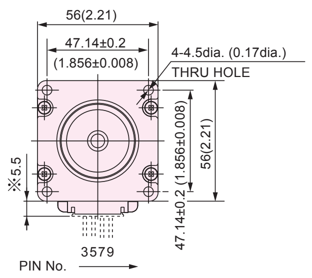
STANDARD SPECIFICATIONS

MODEL		KH56KM2			
		-801	-802	-803	-851
DRIVE METHOD	————	UNI-POLAR			BI-POLAR
NUMBER OF PHASES	————	2			2
STEP ANGLE	deg./step	1.8			1.8
VOLTAGE	V	2.4	3.7	6.8	4.35
CURRENT	A/PHASE	3.0	2.0	1.0	1.5
WINDING RESISTANCE	Ω/PHASE	0.8	1.85	6.8	2.9
INDUCTANCE	mH/PHASE	1.1	3.3	13.5	10.7
HOLDING TORQUE	mN • m	833	833	833	981
	oz • in	118	118	118	132
DETENT TORQUE	mN • m	37	37	37	37
	oz • in	5.6	5.6	5.6	5.6
ROTOR INERTIA	g • cm2	270	270	270	270
	oz • in2	1.48	1.48	1.48	1.48
WEIGHTS	g	650	650	650	650
	lb	1.4	1.4	1.4	1.4
INSULATION CLASS	————	JIS Class E (120°C 248°F) (UL VALUE : CLASS B 130°C 266°F)			
INSULATION RESISTANCE	————	500VDC 100MΩmin.			
DIELECTRIC STRENGTH	————	500VAC 50HZ 1min.			
OPERATING TEMP. RANGE	°C	0 to 50			
ALLOWABLE TEMP. RISE	deg.	70			

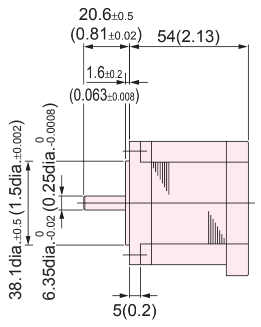
DIMENSIONS unit = mm (inch)



UNI-POLLAR



Bi-POLAR



20.6±0.5
(0.81±0.02)
54(2.13)
1.6±0.2
(0.063±0.008)
38.1dia.±0.5 (1.5dia.±0.002)
6.35dia.-0.02 0 (0.25dia.-0.0008 0)
5(0.2)

SINGLE SHAFT

color-technik
Antriebstechnik GmbH
Starkenburgstr. 6 * 64546 Mörfelden
Tel.:06105 24044 * Fax:06105 25593
info@color-technik.net
www.color-technik.net

Features
- Stronger torque generated in higher speed zone
- Lowered Vibration by increased stiffness of body construction
- Improved Efficiency

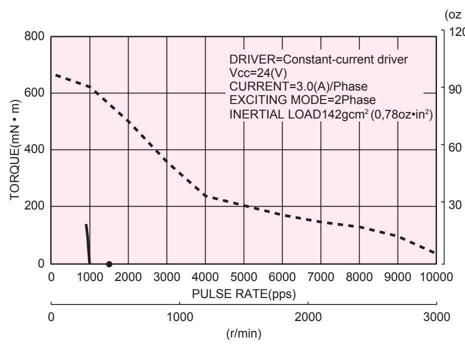
LOAD OF SHAFT:
30N THRUST LOAD
40N RADIAL LOAD 20mm FROM FRONTPLATE

—— PULL-OUT
—● PULL-IN

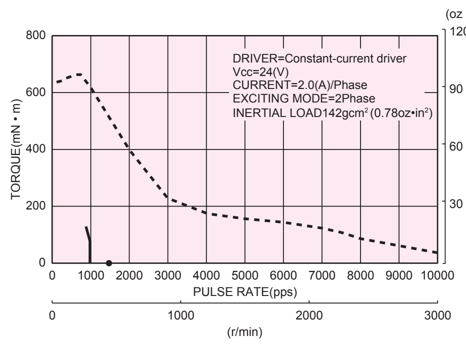
TORQUE CHARACTERISTICS vs. PULSE RATE

UNI-POLAR

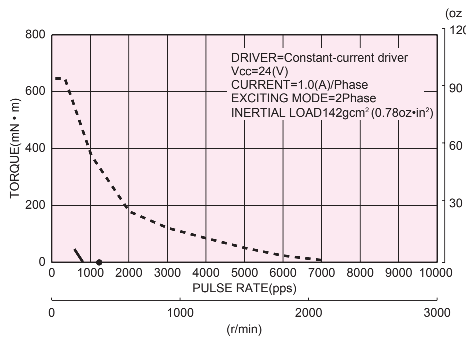
KH56KM2-801



KH56KM2-802,

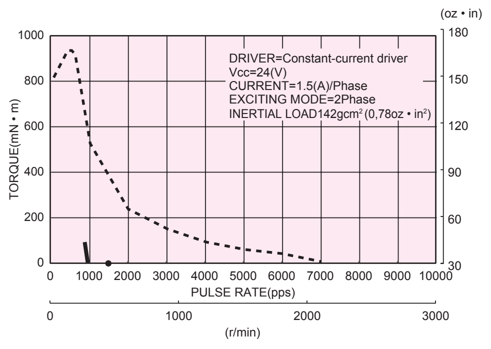


KH56KM2-803



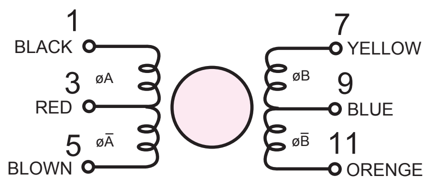
BI-POLAR

KH56KM2-851



CONNECTION DIAGRAMS

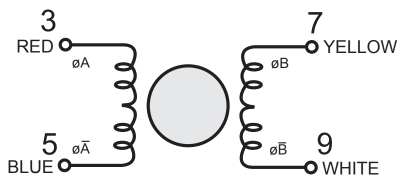
UNI-POLAR



EXCITATION SEQUENCE

STEP	1	2	3	4
BLACK	–			–
YELLOW	–	–		
BLOWN		–	–	
ORENGE			–	–
RED	+	+	+	+
BLUE	+	+	+	+

BI-POLAR

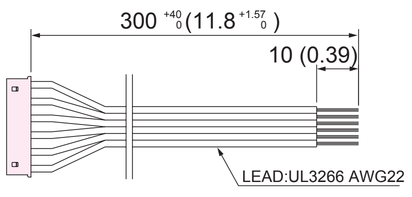


EXCITATION SEQUENCE

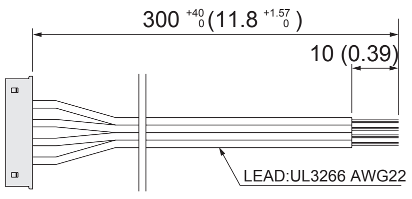
STEP	1	2	3	4
RED	+	+	–	–
YELLOW	–	+	+	–
BLUE	–	–	+	+
WHITE	+	–	–	+

CONNECTION CABLE TO MOTOR unit = mm (inch)

UNI-POLAR



BI-POLAR



JST XHP-11

Appendix B

Usongshine Nema 17 (17HS4401) Datasheet

Data Specs

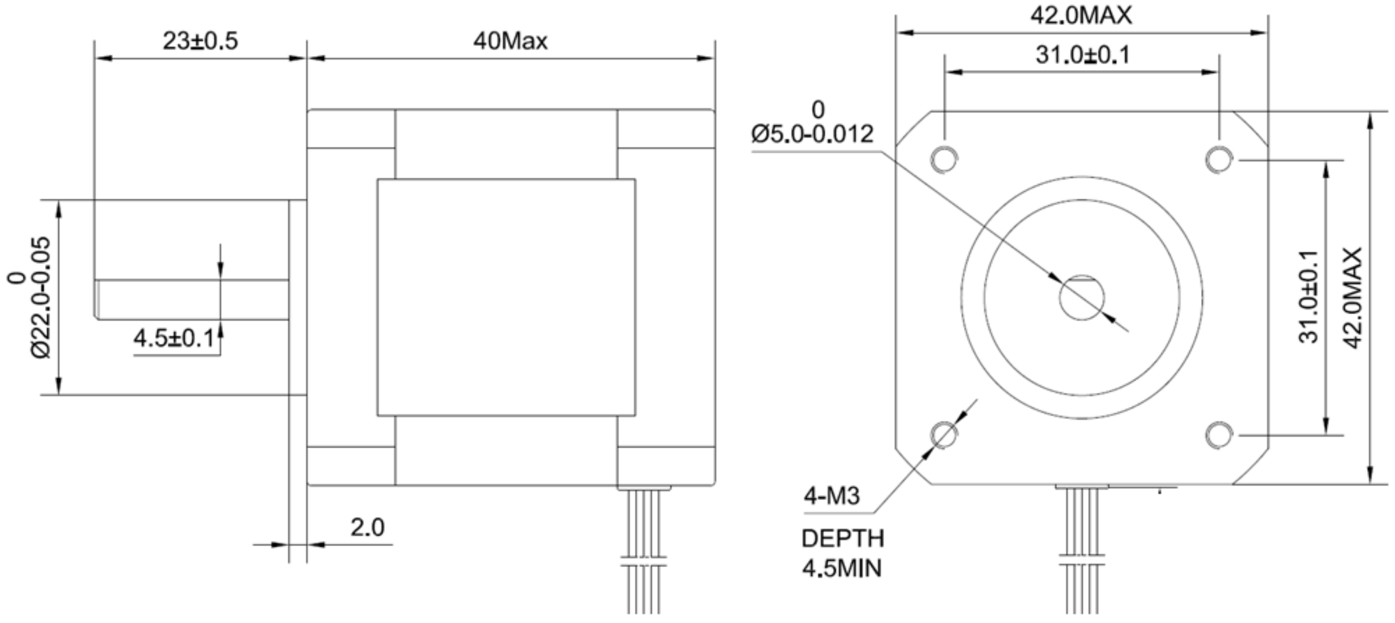
17HS4401S 1.7A Torque:43N.cm Stepper Motor

A stepper motor to satisfy all your 3D-Printer, robotics, Linear Motion projects needs! This 4-wire bipolar stepper has 1.8° per step for smooth motion and a nice holding torque. The motor was specified to have a max current of 1.7A/phase so that it could be driven easily with common motor shield for Arduino (or other motor driver) and a wall adapter or lead-acid battery. The motors are supplied with a 50cm long power cable with a 4-pin Harwin female connector already fitted - ready to plug and print!

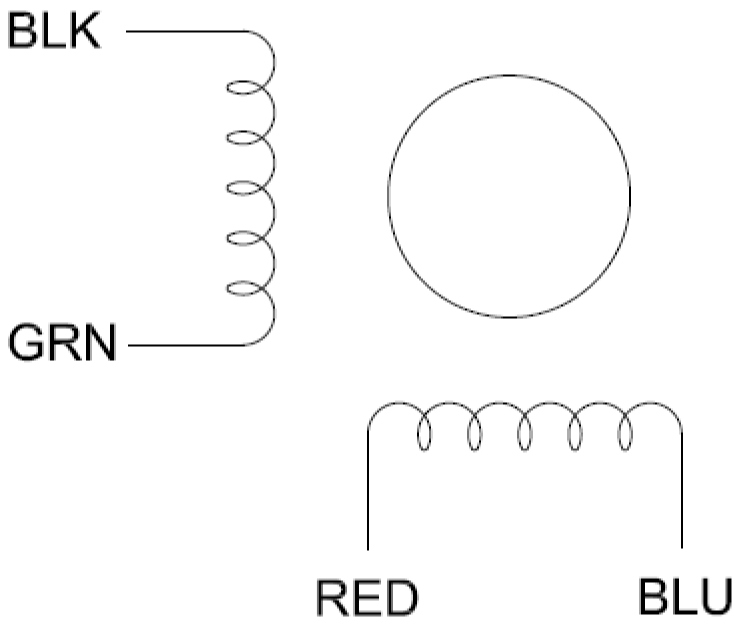
Brief Data:

- Nema17 Bipolar.
- Number of Phase: 2.
- Step Angle: 1.8°.
- Phase Voltage: 2.6Vdc.
- Phase Current: 1.7A.
- Resistance/Phase: 1.5Ω ±10%.
- Inductance: 2.8mH ±20% (1KHz).
- Number of Wire: 4 (100cm Length).
- Holding Torque: 43Ncm.
- Shaft Diameter: Ø5mm.
- Motor Length: 40mm.
- Rotor Inertia: 54gcm^2.
- Temperature rise: 80°C Max.
- Insulation Class: B.
- Dielectric Strength: 500VAC/1-minute.
- Mass: 280g.

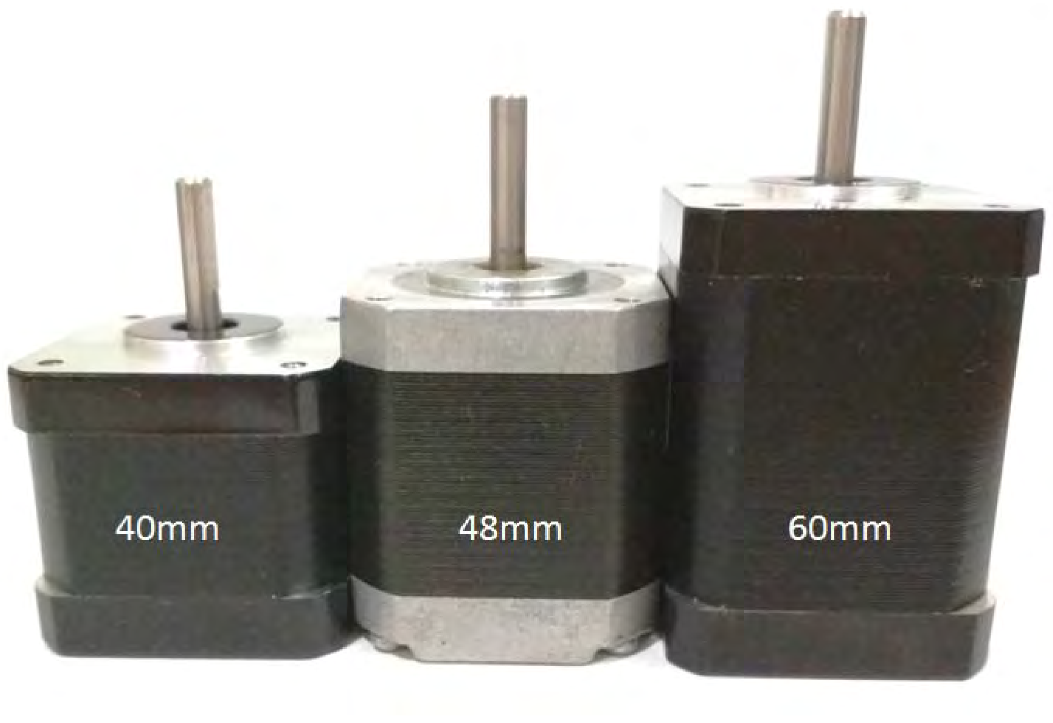
Mechanical Dimensions:



Connection:



Nema-17 Stepper Motor Body Length Comparison:



Appendix C

Arduino Code

```
/* ---SCARA robot---
 *
 *  Date: 09-05-2021
 *  Written by: Benjamin Strom och Anton Labbe
 *  Examiner: Nihad Subasic
 *   TRITAnr : 2021:41
 *  Coursecode: MF133X
 *
 *  Software developed for a selective compliance assembly robot arm
 *  (SCARA) to be able to reach arbitrary coordinates and at the users
 *  order perform a "dotting" action in which the robot arm lowers itself
 *  and makes a dot in a placed coordinate system.
 *
 */

// Include libraries
#include <AccelStepper.h>

//Define pins (The ones that are on the ramps)
#define X_step A0
#define X_dir A1
#define X_max 2
#define X_min 3
#define X_en 38
#define Y_step A6
#define Y_dir A7
#define Y_max 15
#define Y_min 19 //In reality 14 on the ramps
#define Y_en A2
#define Z_step 46
#define Z_dir 48
#define Z_max 14 //In reality 19 on the ramps
#define Z_min 18
#define Z_en A8
```

```
//define stepper motors
const int motorInterfaceType = 1; //using driver (DRV8825)
AccelStepper Xstep = AccelStepper(motorInterfaceType,X_step,X_dir);
    //Inner arm stepper (connected to X on ramps board)
AccelStepper Ystep = AccelStepper(motorInterfaceType,Y_step,Y_dir);
    //Outer arm stepper (connected to Y on ramps board)
AccelStepper Zstep = AccelStepper(motorInterfaceType,Z_step,Z_dir);
    //Z-axis stepper (lead screw, connected to Z on ramps board)

//define variables
const int Xmaxdist = 100; //maximum X movement
const int Ymaxdist = 100; //maximum Y movement
const int Zmaxdist = 7400; //maximum Z-axis movement (7300 steps MAX)
const int L1 = 18;
const int L2 = 17;

bool newData,runcont,runcoord, runZ = false; //flags for future loops
//bool newData, Xruncont, Yruncont, Zruncont = false;
char command; //command from serial monitor (character, h = HOME, c =
    coordinate, d = disable motors(while not running), z = perform dotting
    action with Z-axis)
int X, Y, theta1, theta2, stepstomoveinner, stepstomoveouter; //variabler
    fr koordinatinmatning, vinklar och steg att rra armarna

void setup() {
  Serial.begin(9600);

  //declare pinmodes of microswitch pins
  pinMode(X_min, INPUT_PULLUP); //set pinmode for microswitch to input (in
      this case with pullup to distinguish better between high/low)
  pinMode(Y_min, INPUT_PULLUP); //set pinmode for microswitch to input
  pinMode(Z_min, INPUT_PULLUP); //set pinmode for microswitch to input

  //attach microswitches as interrupt pins
  attachInterrupt(digitalPinToInterrupt(X_min),stopX,FALLING);
  attachInterrupt(digitalPinToInterrupt(Y_min),stopY,FALLING); // FALLING
      or LOW otherwise
  attachInterrupt(digitalPinToInterrupt(Z_min),stopZ,FALLING);

  Xstep.setMaxSpeed(200); //set speed
  Xstep.setAcceleration(50); // set acceleration
  Xstep.setEnablePin(X_en); // set enable pin for inner arm stepper(X)
  Xstep.setPinsInverted(false, false, true); //Inverting the enable
      pin(needed)
  Xstep.disableOutputs(); //disabling motor outputs until given other
      instruction

  Ystep.setMaxSpeed(200); // set speed
  Ystep.setAcceleration(50); // set acceleration
  Ystep.setEnablePin(Y_en); //set enable pin for outer arm stepper(Y)
  Ystep.setPinsInverted(false, false, true); //Inverting the enable
      pin(needed)
  Ystep.disableOutputs(); //disabling motor outputs until given other
      instruction

  Zstep.setMaxSpeed(2000); //set speed
  Zstep.setAcceleration(500); //set acceleration
  Zstep.setEnablePin(Z_en); //set enable pin for Z -axis stepper
  Zstep.setPinsInverted(false, false, true); //Inverting the enable pin
      (needed)
  Zstep.disableOutputs(); //disabling motor outputs until given other
      instruction
}

void loop(){
  checkSerial(); //Checks the serial monitor for inputs
  runContinuously(); //function used for homing action
  runToCoordinate(); //function used to run to a specific coordinate
}

void checkSerial(){ //checks Serial for user input and sets flags
    accordingly
  if(Serial.available() > 0){
    command = Serial.read(); //Read command
    newData = true; //Flag for new received data
  }
  if(newData==true){
    switch(command){
      case 'h': //homing
        runcont = true; //flag to step into runContinuously()
        //Xruncont = true;
        //Yruncont = true;
        //Zruncont = true;
        Serial.println("HOMING");
        Xstep.setMaxSpeed(20);
        Ystep.setMaxSpeed(20);
        Zstep.setMaxSpeed(200); //set homing speed
        Xstep.setAcceleration(5);
        Ystep.setAcceleration(5);
        Zstep.setAcceleration(50); //Set homing acceleration
        Xstep.move(400);
        Ystep.move(400);
        Zstep.move(10000); //set max distance to move (Should be larger
            than the maximum possible distance to run)
        break;

      case 'c': //runs to coordinate
```

```
        runcont = false;
        Serial.println("INPUT COORDINATE: X,Y");
        Serial.println("TESTSEQUENCE:
            (5,30);(15,15);(30,5);(-5,30);(-10,25);(-35,0)");

        X = Serial.parseInt(); //Parses int from Serial monitor, program is
            meant for any input separating two ints (for example c 15 15 or
            c 15,15)
        Y = Serial.parseInt(); //both do the same thing, run to the
            coordinate (15,15)

        coordToSteps(X,Y); //Call function to calculate amount of steps

        Serial.print("You input coordinate: (");
        Serial.print(X);
        Serial.print(",");
        Serial.print(Y);
        Serial.println(")");
        Xstep.setMaxSpeed(20);
        Ystep.setMaxSpeed(20);
        Xstep.setAcceleration(5);
        Ystep.setAcceleration(5);
        Xstep.moveTo(stepstomoveinner);
        Ystep.moveTo(stepstomoveouter); //step count to move to with inner
            (Should be larger than the maximum possible distance to run)
        runcoord = true; //flag to step into runToCoordinate()
        break;

      case 'z': //performs a lowering and raising of Z axis
        Serial.println("Performing dotting action...");
        Zstep.setMaxSpeed(1000); //set speed
        Zstep.setAcceleration(200); //Set acceleration
        Zstep.enableOutputs();
        Zstep.runToNewPosition(-6700);
        Zstep.runToNewPosition(0);
        Zstep.disableOutputs();
        break;

      case 'd': //disables motors
        runcont = false;
        //Xruncont = false;
        //Yruncont = false;
        //Zruncont = false;
        Xstep.disableOutputs();
        Ystep.disableOutputs();
        Zstep.disableOutputs();
        break;
    }
  }
  newData=false;
}

void runContinuously(){ //function that runs the motors continuously
    withing the maximum allowed amount of steps
  if(runcont == true){
    if(abs(Xstep.currentPosition()) < Xmaxdist ||
    abs(Ystep.currentPosition()) < Ymaxdist ||
    abs(Zstep.currentPosition()) < Zmaxdist){
      Xstep.enableOutputs();
      Ystep.enableOutputs();
      Zstep.enableOutputs();
      Xstep.run();
      Ystep.run();
      Zstep.run();
    }

    else{
      runcont = false;
      Xstep.disableOutputs();
      Ystep.disableOutputs();
      Zstep.disableOutputs();
      Xstep.setCurrentPosition(0);
      Ystep.setCurrentPosition(0);
      Zstep.setCurrentPosition(0);
    }
  }
  else{
    return;
  }
}

void runToCoordinate(){ //function that runs the stepper motors to input
    coordinate
  if(runcoord == true){
    if(abs(Xstep.distanceToGo()) > 0 && abs(Ystep.distanceToGo()) > 0){ //
        if the steppers haven't ran the full way
      Xstep.enableOutputs();
      Ystep.enableOutputs();
      Xstep.run();
      Ystep.run();
    }
    else if(abs(Xstep.distanceToGo()) > 0 && abs(Ystep.distanceToGo()) ==
        0){ //if X hasn't ran the full way
      Xstep.enableOutputs();
      Xstep.run();
      Ystep.stop();
    }
    else if(abs(Ystep.distanceToGo()) > 0 && abs(Xstep.distanceToGo()) ==
        0){ //if Y hasn't ran the full way
      Ystep.enableOutputs();
      Ystep.run();
      Xstep.stop();
    }
    else{
      runcoord = false; //when finished running, turn off the function
      Xstep.disableOutputs(); //and the motors
      Ystep.disableOutputs();
    }
  }
  else{
    return;
  }
}

void coordToSteps(float X, float Y){ // function to calculate amount of
    steps to move to a coordinate for each stepper motor
  float theta1, theta2, angletomoveinner, angletomoveouter;

  if(X < 0){
  theta2 = acos((pow(X,2)+pow(Y,2)-pow(L1,2)-pow(L2,2))/(2*L1*L2));
      //expression derived from model of arm with trigonometry
  theta1 = (atan(Y/abs(X))-atan(L2*sin(theta2)/(L1+L2*cos(theta2))));
      //arm angled "above" length to coordinate

  theta2 = theta2 * (180/PI); //todegrees
  theta1 = theta1 * (180/PI);

  angletomoveinner = -1*theta1; //translate the angles to our home position
  angletomoveouter = -1*(90+theta1+theta2);

  stepstomoveinner = round(angletomoveinner / 1.8); //one step is 1.8
      degrees
  stepstomoveouter = round(angletomoveouter / 1.8);
  }
  else{
  theta2 = -1*acos((pow(X,2)+pow(Y,2)-pow(L1,2)-pow(L2,2))/(2*L1*L2));
  theta1 = (atan(Y/X)-atan(L2*sin(theta2)/(L1+L2*cos(theta2)))); //arm
      angled "above" length to coordinate

  theta2 = theta2 * (180/PI);
  theta1 = theta1 * (180/PI);

  angletomoveinner = -1*(180-theta1);
  angletomoveouter = -1*(290-(theta1+theta2));

  stepstomoveinner = round(angletomoveinner / 1.8);
  stepstomoveouter = round(angletomoveouter / 1.8);
  }
}



void stopX(){ //function used in the X microswitch interrupt
    Xstep.setCurrentPosition(0);
    Xstep.stop();
    Xstep.disableOutputs();
}

void stopY(){ //function used in the Y microswitch interrupt
    Ystep.setCurrentPosition(0);
    Ystep.stop();
    Ystep.disableOutputs();
}

void stopZ(){ //function used in the Z microswitch interrupt
    Zstep.setCurrentPosition(0);
    Zstep.stop();
    Zstep.disableOutputs();
}
```

Appendix D

Acumen model of Z-axis movement

```
/* ACUMEN SIMULATION FOR SCARA ROBOT
COURSE MF133X BACHELORS DEGREE IN MECHATRONICS
MADE BY: Benjamin Strom och Anton Labbe */

model Main(simulator) =
initially //Creates models and variables
/*
Note: places to put the blocks
 base:
 initial pos = (0,0,0)
 size = (2,2,1)

 top:
 initial pos = (0,0,5)
 size = (2,2,1)

 mount:
 initial pos = (0,1/2,3)
 size = (1,1,1/2)

 Arm1:
 initial pos = (0,7/4,5/2)
 size = (1,5/2,1/2)

 Arm2:
 initial pos = (0,15/4,2)
 size = (1,5/2,1/2)
 */

//Creates models
base = create block((0,0,0),(2,2,1)),
axis = create axis((0,0,2.5)),
top = create block((0,0,5),(2,2,1)),
mon = create block((0,0,0),(1,1,1/2)),
armA = create block((0,0,0),(1,5/2,1/2)),
armB = create block((0,0,0),(1,5/2,1/2)),

//Create state-variables
z_mon = 0, z_mon' = 0,
z=0

always // What will happen with the state variables
z_mon' = 1,
z = 1.5*sin(z_mon), // Creates a sine function for up/down oscillating
    Z-axis movement
mon.pos = (0,1/2,z+2.7),
armA.pos = (0,7/4,z+2.2),
armB.pos = (0,15/4,z+1.7)

model block(pos,size) = //Model for blocks
initially
_3D = (), _3DView = ()

always
_3D = (Box //Base
center = pos
size = size //(x,y,z)
color = blue
transparency = 1)

model axis(pos) = //Model for z-axis
initially
_3D = (), _3DView = ()

always
_3D = (Cylinder //Z-axis is a cylinder
center = pos
size = (4,0.2) //size(height,radius)
color = black
rotation = (pi/2,0,0)
transparency = 1)
```

Appendix E

Acumen model of arms

```
/* ACUMEN SIMULATION FOR SCARA ROBOT
COURSE MF133X BACHELORS DEGREE IN MECHATRONICS
MADE BY: Benjamin Strom och Anton Labbe */

model Main(simulator) =
initially //Create state variables
theta1 = 0, theta1' = 0,
theta2 = 0,
l1 = 4, l2 = 4,
x1 = 0, y1 = 0,
x2 = 0, y2 = 0,
_3D = ()

always //Does a full rotation
if theta1 < 2*pi
then theta1' = 1
else theta1' = 0,
theta2 = 2*theta1,
x1 = (0.5*l1+0.5)*cos(theta1),
y1 = (0.5*l1+0.5)*sin(theta1),
x2 = (l1+0.1)*cos(theta1)+(0.5*l2-0.4)*cos(theta2),
y2 = (l1+0.1)*sin(theta1)+(0.5*l2-0.4)*sin(theta2),


_3D = (Box //Creates arm components
center = (x1,y1,0)
size = (l1,0.5,0.5)
color = (1,0,0)
rotation = (0,0,theta1)
Box
center = (x2,y2,-0.5)
size = (l2,0.5,0.5)
color = (1,0,0)
rotation = (0,0,theta2)
)
```

TRITA EX-ITM 2021:41