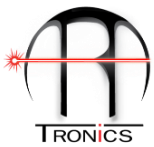


Applied-Science & Robotics Laboratory
for Applied-Mechatronics

German University in Cairo
Faculty of Engineering and Materials Science (EMS)
Mechatronics Engineering Department (MCTR)

Four Degrees of Freedom SCARA Robot Used for Laser Engraving

A thesis submitted for the degree of Bachelor of Science in Mechatronics Engineering

Marwan Mohammed Aly Sallam
46-3401

Supervised by

Assoc. Prof. Dr. Eng. Amir Roushdy

Prof. Dr. Eng. Elsaid Imam Morgan

Year
2022

Declaration

This is to declare that:

I. The thesis comprises only my original work towards the Bachelor degree.

II. Due acknowledgment has been made in text to all other material used.

_____

Marwan Mohammed Aly Sallam

12 June, 2022

Acknowledgments

First of all, I sincerely would like to thank my family for their continuous encouragement and support throughout my study years.

I also would like to express my sincere gratitude to Prof. Dr. Amir R. Ali for his continuous inspiration, motivation and support throughout my bachelor journey.

Furthermore, I would like to thank the lab engineer Eng. Malek for his technical support.

Finally, I would like to thank my lab mates in ARAtronics lab for their continuous guidance, support and encouragement throughout my Bachelor journey.

Abstract

In this bachelor thesis, a fully functional SCARA robot was designed and assembled. Using this SCARA, some work pieces have been laser engraved using the end-effector which is a laser engraver module. This SCARA can be used for other purposes by simply replacing the laser module with another end effector. The equation of motion of the robot was derived using Lagrangian method. To get forward kinematics D-H convention was used and geometrical method was used to get inverse kinematics. MATLAB/Simulink was used to test the forward kinematics and equation of motion of robot.

Table of Contents

List of figures

List of tables

Chapter 1 Introduction and Literature Review

1.1 Introduction to Robots

Because of the need for automation, robots are increasingly being used in sectors to reduce worker weariness and increase productivity. Material handling, operations, assembly, and inspection are the four main domains in which robots are used in industry. Aside from industry, robots are used in the household, health care, service sectors, agriculture and farming, as well as research and exploration. Robotic applications are only limited by the developers and end user's requirement and creativity. Robots have the ability to alter our economy, health, way of life, and the planet in which we live [1]. Before going any deeper let's take a step back and get a brief look on the history of robots. The word "robot" comes from the Slavic language. In the early twentieth century, Czechoslovakian playwright Karel Capek (1890-1938) adapted it to its current meaning. Capek invented mechanised equivalents for human employees in the play R.U.R. (Rossum's Universal Robots), who had a human worldview and were capable of "human" sensations [2]. In reality, the term "robot" was coined considerably later in history than the actual systems that bear that name. International standard ISO 8373 defines a "robot" as "An automatically controlled, reprogrammable, multipurpose, manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications." Although this definition can be used to define robots, there are many more available (ISO, 1994). Basic concept of the industrial robot can be seen in Fig.1.1.

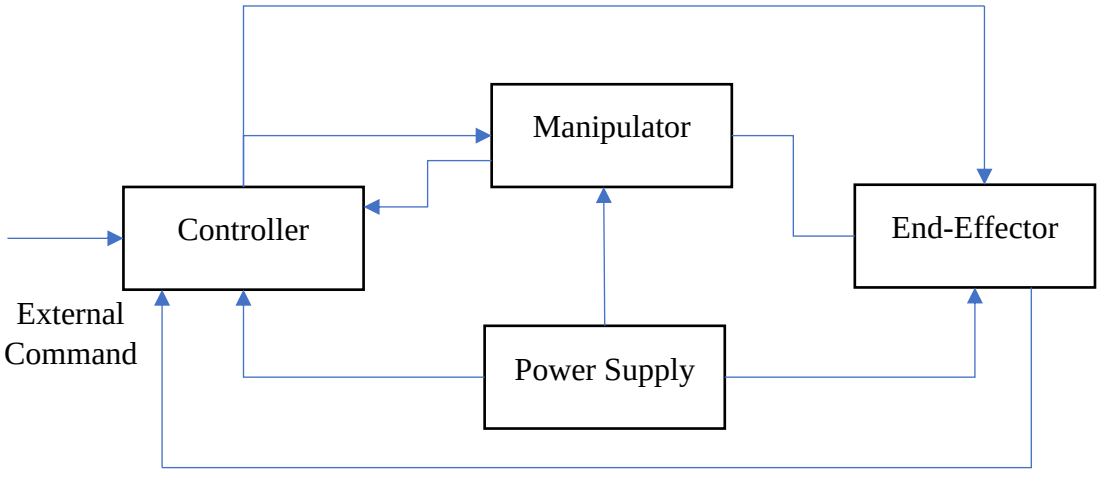


Fig.1.1 Basic concept of the industrial robot

The following conditions may be used as guidelines for using robots:

1. Dangerous (risky) or inconvenient working conditions: In situations when there is a risk of injury or health hazards (like heat, radiation, toxicity, etc.). Hot forging, die casting, spray painting, and even laser engraving may all be done using robots.
2. Difficult handling: If the work piece or tool in question is very pointed and small in shape or of a hefty kind, the robot may be able to perform the task more effortlessly.
3. Repetitive task: If the work cycle consists of a series of items that do not change from cycle to cycle, the robot may be able to perform the task. It alleviates workers' boredom with repetitive tasks.
4. Continuous operation: If a job is required to be operated for more than one hour, days, or months, then machine must be automated.

1.2 Robot Classification

Robots can be classified according to their type of control system, type of physical configuration or workspace geometry, their type of base, type of kinematic structure and type of drive technology.

1.2.1 Type of Base

I. Stationary Robots

Stationary robots are fixed in position and cannot move their base away from the work being done. and their classification is done according to the type of motion they offer. They are used in industries to aid or replace human interaction with the aim of increased productivity or reduce risk.

II. Mobile Robots

Mobile robots perform tasks with degree of autonomy which is desirable in fields such as spaceflight and their base is typically a platform with wheels or tracks attached. When producing a mobile robot extra requirement must be taken other than designing and manufacturing like stationary robots. These requirements include navigation systems, collide control systems, implemented AI or even machine learning ability.

1.2.2 Type of Physical Configuration/Workspace Geometry

I. Cartesian Robots

Cartesian robots, also known as linear robots or gantry robots, are industrial robots that operate on three linear axes using the Cartesian Coordinate System (X, Y, and Z), which means they move in straight lines on three axes (up and down, in and out, and side to side). Cartesian robots are popular because of their configuration flexibility, which allows users to alter the robot's speed, accuracy, stroke length, and size. Cartesian robots are one of the most prevalent sorts of industrial robots, and they're frequently employed in CNC machines and 3D printing. Positioning can be easily done by linear motion along the three principal axes. Its work area is a big box-shaped area.

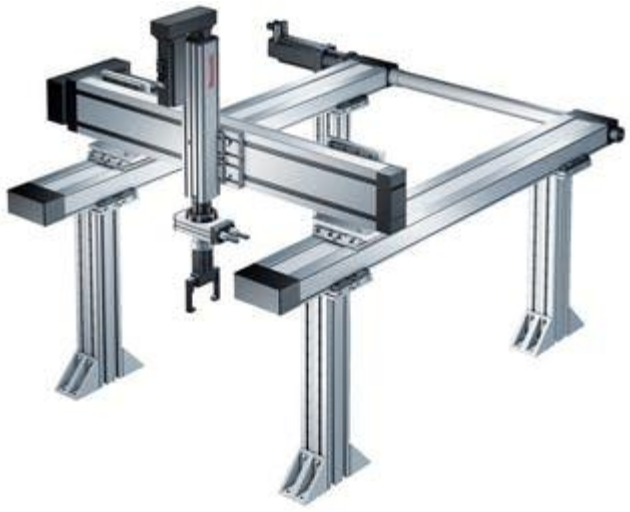


Fig.1.2 Cartesian Robot

II. Polar Robots

Polar robots, also known as spherical robots, feature an arm with two rotary joints and one linear joint that is coupled to a twisting joint at the base. Polar Robots are widely regarded as one of the original forms of industrial robots. Die casting, injection moulding, welding, and material handling are all popular uses for polar robots. It has two spherical axes and one transitional axis. The robot's axes combine to generate a polar coordinate, allowing the robot to have a spherical work envelope or more accurately its work area is the space between two concentric hemispheres.

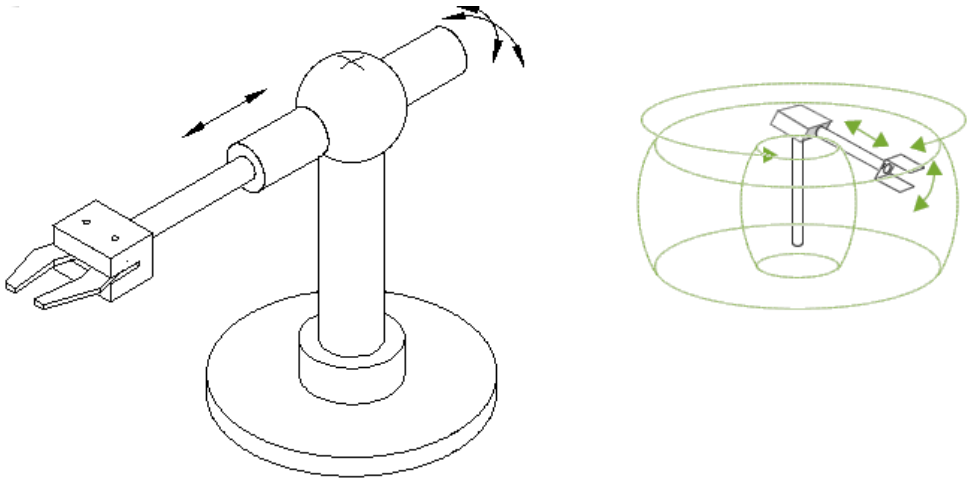


Fig.1.3 Polar Robot and its work envelope

III. Cylindrical Robots

Cylindrical Robots have a rotary joint at the base and a prismatic joint to connect the links. Because of their compact size, cylindrical robots are frequently utilised in small places for simple assembling, machine tending, or coating applications. The robots have a cylindrical work envelope or the space between two concentric cylinders of same height, which is created by a rotating shaft and an extensible arm that travels vertically and slides.

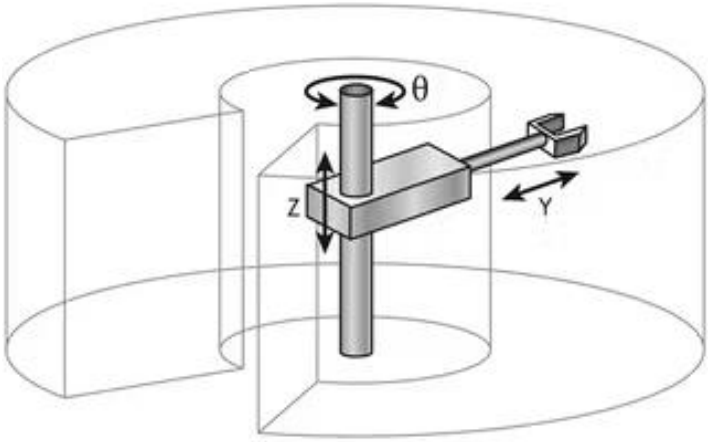


Fig.1.4 Cylindrical Robot and its work envelope

IV. Articulated Robots

Also known as jointed robots. The mechanical movement and arrangement of an articulated robot closely matches that of a human arm. A twisting joint connects the arm to the base. The arm can have anywhere from two to ten rotary joints acting as axes, with each extra joint or axis allowing for more motion. The majority of articulated robots have four or six axes. Assembly, arc welding, material handling, machine tending, and packing are all common uses for Articulated Robots. Its workspace differs depending on the number of links and size of robot. Fig.1.6 right picture shows an example of an articulated robot.

V. SCARA Robots

Selective Compliance Assembly Robot Arm or Selective Compliance Articulated Robot Arm is the abbreviation for SCARA. SCARA Robots feature three axes of motion (X, Y, and Z), as well as a rotational motion. SCARA Robots are better at lateral motions than Cartesian Robots and are usually faster and easier to integrate. SCARA robots are commonly used for assembly, palletizing, and biomedical applications. Fig.1.6 left picture shows an example of a SCARA robot.

VI. Delta Robots

Delta Robots, also known as parallel robots, have three arms that are all attached to a single base that is suspended above the workstation. Because each joint of the end effector is directly controlled by all three arms, Delta Robots can operate both softly and accurately at high speeds. Delta Robots are frequently utilised in the food, pharmaceutical, and electrical sectors for quick pick and place applications. Fig.1.7 shows an example of a delta robot.

VII. Collaborative Robots

Cobots, or collaborative robots, are robots that can engage directly and securely with people in a shared office. On the market, there are many different types and brands of collaborative robots. Pick and place, palletizing, quality checking, and machine tending are all common uses for cobots. It's one of the new and uprising types of robots.

Fig.1.5 ABB YUMI COBOT

1.2.3 Type of Control System

I. Axis Limit Controller

It is the simplest type of motion control where the maximum and limit of each axis is controlled by limit switches or adjustable stops. It does not offer any speed control but is of very low cost.

II. Point to Point (PTP) Controller

The PTP robots has the ability to move from one location to another but only specified point within the work envelope. The coordinates are saved to the control memory. The path taken by PTP robots to go from one spot to the next is not within their control. More flexible than axis limit but still offers no speed control. This is generally practical for component insertion, hole drilling and spot welding.

III. Continuous-Path (CP) Controller

Also known as contouring controllers. The CP robot is capable of moving along the predetermined course. The robot may halt at any point along the regulated path using CP from a single control. All of the points along the path must be saved in the robot's control memory directly. The simplest example of this sort of robot is one that moves in a straight path. Some continuous-path driven robots can also follow a smooth curve path that has been programmed by the programmer. In such circumstances, the programmer directs the robot arm along the intended path manually, and the controller unit memorises a huge number of specific point positions along the path (teach-in). It utilizes feedback loop and offers speed control. It is practical for spray painting, surface finishing and arc welding.

IV. Controlled-Path Controller

Also known as line tracking controller. It is the most complex and used in cases involving robots performing operations alongside a moving conveyor. It offers extra simultaneous tracking of external objects speed variations relative to time. In controlled-path robots, the control equipment can generate paths of different geometry such as straight lines, circles, and interpolated curves with a high degree of accuracy. Good accuracy can be obtained at any point along the specified path. Only the start and finish points and the path definition function must be stored in the robot’s control memory. It is important to mention that all controlled-path robots have a servo capability to correct their path.

1.2.4 Drive Technology

Despite the fact that there are three basic driving technologies: electric, hydraulic, and pneumatic, most manipulators employ electric servo motors or stepper motors since their benefits are more visible. Hydraulic and pneumatic drive, on the other hand, have some advantages, such as great load-carrying capacities.

1.2.5 Kinematic Structure

The kinematic structures of the robots are also categorized. A "serial robot" or "open loop manipulator" is a robot with an open loop chain kinematic framework. A "parallel manipulator" is one that has a closed loop-chain kinematic structure. Furthermore, a robot system that combines both structural types is known as a "hybrid manipulator." [3]

1.3 Introduction to Manipulators

A robot manipulator, according to B.Z. Sandler [4], is "a system, generally consisting of a number of segments, jointed or sliding relative to one another, for grabbing and manipulating objects in various degrees of freedom." It may be operated remotely by a computer or a person. Robot manipulators are divided into three categories: Serial, parallel, and hybrid manipulators.

1.3.1 Serial manipulators

The most popular industrial robots are serial manipulators, which are made up of a series of stiff linkages joined by joints. The majority of their joints are revolute. Anthropomorphic arm structures are common among serial manipulators. They have a shoulder like joint (first two joints), an elbow like joint (third joint), and a wrist like joint (last three joints). According to rigid body motion, a robot must have at least six degrees of freedom to place a controlled item in an arbitrary location and orientation in the robot's workspace. As a result, most serial robots have six joints. SCARA robots, on the other hand, have only four degrees of freedom, making them one of the most frequent serial robot applications. It is a special assembly robot and used in generally pick and place applications. Fig.1.6 shows examples of serial manipulators.

Advantages of serial manipulators:

1. Larger dexterous workspace.
2. Simplicity of the forward and inverse position and velocity kinematics.
3. Able to achieve high velocities and accelerations.
4. Have revolute joints which are cheaper rather than prismatic joints.

Disadvantages of serial manipulators:

1. They are very heavy because the links must be stiff.
2. Errors are accumulated and amplified from link to link.
3. Not energy efficient.

Fig.1.6 SCARA (left) and PUMA (right)

1.3.2 Parallel Manipulators

A closed chain mechanism is a parallel manipulator with a fixed (base) and moveable (end effector) platform. A number of "legs" link the base platform to the end effector platform. Spherical or universal joints link these legs to the platforms. An actuator is in charge of each leg. The number of actuated legs determines the degree of freedom. The parallel manipulators' main benefits are their precise positioning capabilities and light constructions, as the linkages only experience traction or compression, not bending. Furthermore, actuators can be put in the foundation platform, reducing the weight of moveable structure [5]. Fig.1.7 shows an example of parallel manipulators.

Advantages of parallel manipulators:

1. High structural stiffness and load capacity.
2. High bandwidth motion capability.

Disadvantages of parallel manipulators:

1. Limited workspace.
2. Loosing stiffness in singular position completely.

Fig.1.7 ABB parallel manipulator

1.3.3 Hybrid Manipulators

Parallel manipulators became the top choice for robotic applications that demand properties like as precision, fast speed, and rigidity. However, industrial robots often employ serial manipulators with an open kinematic chain because parallel manipulators have restricted work space, which is insufficient for many applications. Because parallel and serial manipulators have different benefits, a hybrid type manipulation system combines the two manipulators and includes characteristics from each. Furthermore, hybrid manipulators overcome the parallel manipulators' restricted workspace. One serial and one parallel manipulator, or two parallel manipulators, make up hybrid manipulators. Hybrid manipulators can also be utilised to boost the system's degree of freedom. A hybrid manipulator with 10 DOF is created by combining a serial manipulator with 7 DOF with a parallel manipulator with 3 DOF. Fig.1.8 shows an example of a hybrid manipulator.

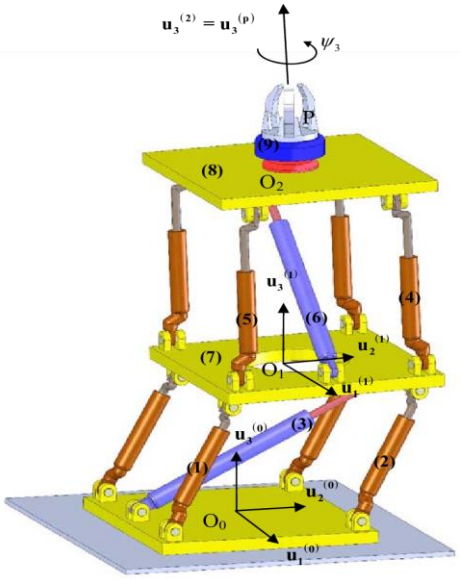


Fig.1.8 Hybrid Manipulator

1.4 Introduction to SCARA

SCARA (Selective Compliant Assembly Robot Arm or Selective Compliant Articulated Robot Arm) is an industrial robot created at Yamanashi University in Japan to bridge the gap between simple pneumatic pick-and-place robots and servo-controlled robots modelled like human arms like Unimation PUMA. Fig.1.9 shows the first protype of SCARA. It's made this way to provide horizontal compliance, which is crucial for vertical insertion processes as well. The inverse of stiffness is compliance. Compliance is the value displacement divided by force, whereas stiffness or spring constant is the value force divided by displacement. When compliance is high, the equipment moves a lot for a given force, and it's called soft or flexible. The compliance of the SCARA should differ with the direction and thus this characteristic is termed as "selective compliance". The SCARA uses a byobu-like structure to achieve this property, as seen in Fig.1.10. A byobu is a traditional Japanese room divider composed of wood and paper or linen that can be folded. It glides quickly horizontally but stiffly vertically. The SCARA's first two arms revolve along the vertical axis and function similarly to the folding portions of a byobu.[6]

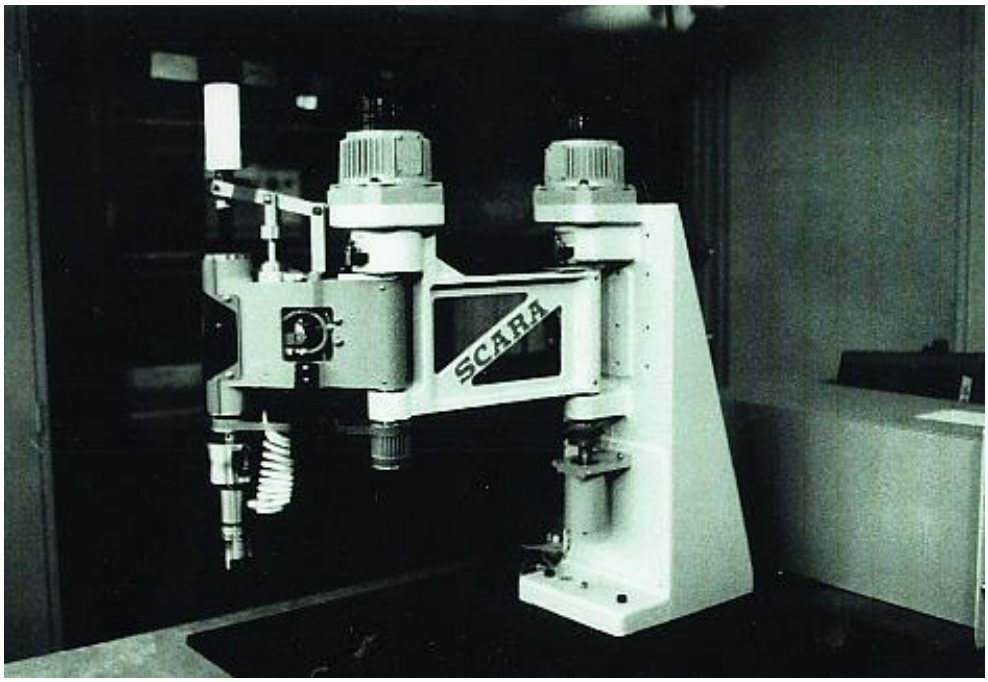


Fig.1.9 The first prototype of SCARA

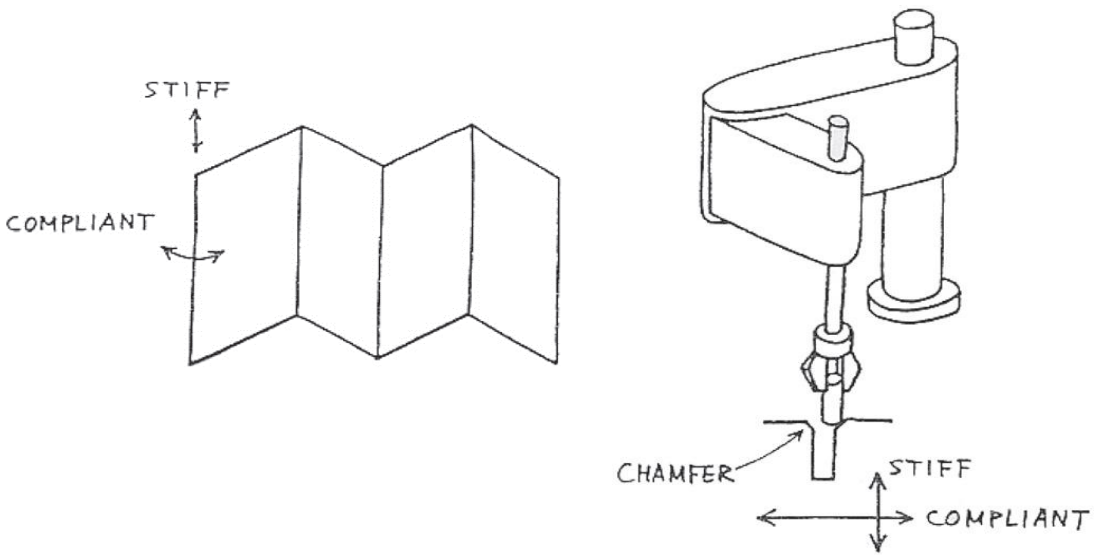


Fig.1.10 Selective Compliance

The SCARA robot's success can be directly related to its working approach. SCARA works on a level surface and accepts some imperfections in its results this is all due to its difference in compliance or selective compliance feature. These errors are readily corrected by the robot. If a non-SCARA robot attempted to put a peg into a hole and the hole was not in the expected location, the robot would be unable to appropriately account for the misplacement. As a result, it would be unable to accomplish the task. A SCARA robot could adapt to the new hole location and finish the insertion procedure if it tried to insert the peg into the same out-of-place hole. SCARA is a good match for constrained assembly operations like packaging or electrical applications that require work piece access from above. The robot arms' motion capabilities are restricted to revolute and translation motions in the horizontal plane. The Gripper Axis is a vertical axis. SCARA is basically an anthropomorphic or jointed-arm (RRP) structure, where R represents a revolute, or hinged, joint, and P is a prismatic, or sliding, joint [7]. Robot having 2 parallel rotary joints to provide compliance in X-Y direction; sufficiently rigid in Z-direction.

SCARA features four degrees of freedom (DOF), with three rotating axes operating in the X-Y plane and one vertical axis operating in the Z plane. Joint1 (main arm) provides the first rotating motion, Joint2 (forearm) provides the second rotational motion, and Joint3 provides the third rotational motion (gripper). The gripper is normally powered by a motor at the arm's fixed end.

This design is important because it keeps the gripper, and hence the work piece, at a consistent angle to the bench regardless of arm movement. SCARA's vertical axis is also crucial for placement.

SCARA robots are still quite valuable in the automation industry. When it comes to transporting objects from point A to point B for jobs like as dispensing, pick and place operations, product assembly, and pallet loading, SCARA is the best for such jobs. SCARA robot deployment has increased due to its ability to handle greater payloads, improved working relationships with other work stations on the line, and increased capability and diversity of end effectors. Because of their speed and cheaper installation costs, articulated robots have taken over some of the job traditionally performed by SCARA robots. No matter happens SCARA robots will always be in use as SCARA robots can perform tasks that no other robots can. While the capabilities of SCARA robots have improved, so has the simplicity of installation and usage of the equipment's such as installing laser modules to use for laser engraving.

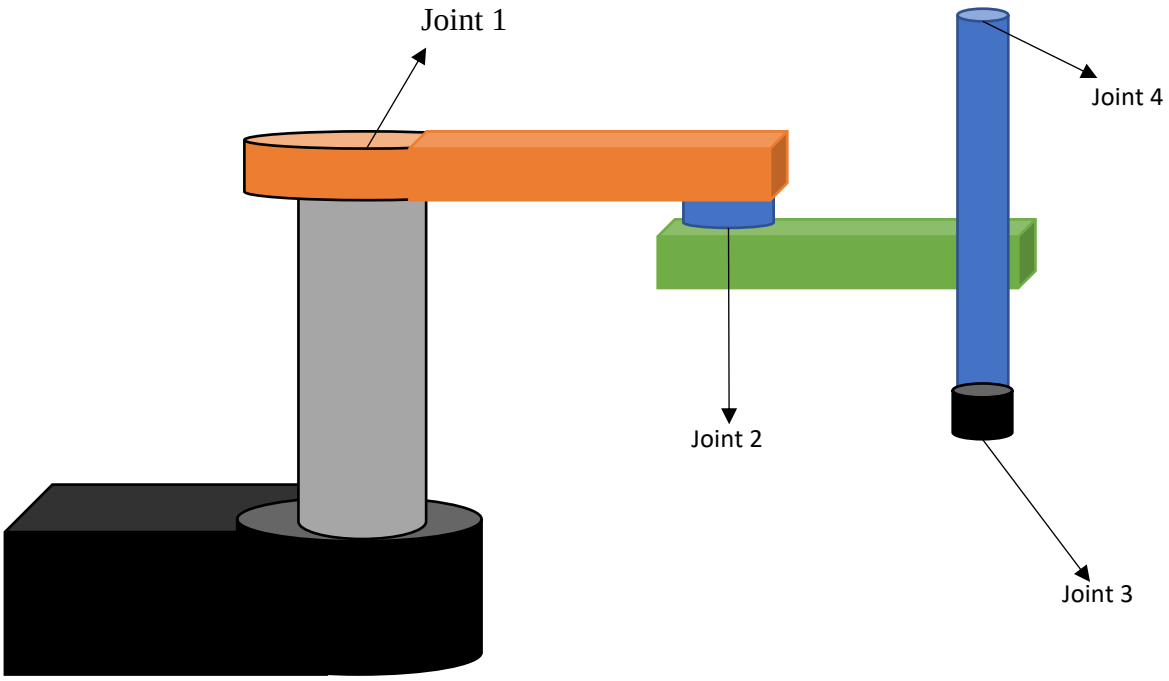


Fig.1.11 Simple Diagram of SCARA

1.5 Literature Review

There have been a lot of previous work on the SCARA with all being and have intriguing qualities. In literature [8] a unique attempt is provided to model PRRP (Prismatic-Revolute-Revolute-Prismatic), a configuration redundant SCARA (Selective Compliance Articulated Robot Arm) robot with a Multi spindle drilling tool (MSDT) using SolidWorks CAD software, as well as a dynamic analysis using MATLAB/SimMechanics. Another interesting find was literature [9] where they tried to achieve the balancing of the inverted pendulum using SCARA with 2 DOF. The first ever prototype of an automated supermarket checkout system has been constructed, utilizing the SCARA robot's capabilities and adding machine vision. The system can distinguish different things placed on a conveyor by the consumer, move them to a container, pack them neatly, and complete the cost [10]. A novel coordinated control strategy based on decoupling servo control in this study [11] for designing a 4-DOF direct-drive SCARA robot for wafer handling has been presented. The goal of this literature [12] is to design and create a mechanical structure for a SCARA robot that can execute activities such as pick and place for educational, research, and demonstration purposes. The stages involved in designing and developing a four-degree-of-freedom (DOF) SCARA robot are discussed in literature [12], which include specification definition, conceptual design, product development, and testing. Implementation of SCARA and artificial neural networks has been exploited in literature [13]. The results of using artificial neural networks to control the position of a genuine SCARA manipulator robot are described in this research [13]. A neural controller runs in tandem with a conventional controller based on the feedback error learning architecture in the general control strategy. The research [14] offers a synthesis of the Double SCARA Robot modelling, which leads to an ideal solution in terms of workspace, accuracy, and stability of the end-effector (laser or pick and place) in completing the intended trajectory. The study [15] details the creation and testing of a desktop SCARA with three degrees of freedom (DOF) that can pick and place items with great speed and precision. The basic modelling of the SCARA robot was done in Autodesk inventor. This paper [15] future scope includes using laser engraver with the SCARA to perform laser engraving task.

1.6 Summary

SCARA (Selective Compliant Assembly Robot Arm or Selective Compliant Articulated Robot Arm) has four degrees of freedom (DOF) with three rotating axes acting in the X-Y plane and one vertical axis functioning in the Z plane. SCARA works on a flat surface and accommodates minor flaws which makes it a major success. In the following chapters the following will be covered starting with the robot's mathematical model which will basically be derived by the DH convention. Then control algorithms will be discussed but in short, an open-source software Marlin with the help of Arduino control code will be used to handle this. After that, there will results and discussion where the work is evaluated and graphs are compared. Experimental work will be next and, in this chapter, assembling and any adjustments done on the robot will be discussed. Finally, conclusion and future work were everything is summed and any shortcomings is discussed for future work.

Chapter 2 Mathematical Model

2.1 Introduction to Kinematics

Kinematics is the study of object motion without taking into account the forces that create that motion. Because we aren't concerned with things like forces, inertia, or the effects of gravity, it is actually a study of geometry, or the geometry of motion. It's a lot simpler than something like dynamics from that standpoint (which is concerned with forces). The name "kinematics" is derived from the Greek word "kinesis," which means "motion," and is connected to words like "cinema" (movies) and "kinesiology" (the study of human motion). The technique of measuring the kinematic characteristics needed to characterize motion is known as kinematic analysis. Kinematics may be reduced to simple mathematical equations that can be used to compute velocity, acceleration, displacement, time, and trajectory. The pose is the collective name for the posture and orientation of a rigid body in space. Robot kinematics thus defines the pose, velocity, acceleration, and any higher-order derivatives of the pose of the bodies that make up a mechanism. [16].

Kinematics investigates the trajectories of points, lines, and other geometric objects, as well as their differential qualities, to characterize motion (such as velocity and acceleration). In astronomy, kinematics is used to explain the motion of celestial planets and systems; in mechanical engineering, robotics, and biomechanics, it is used to describe the motion of systems made up of connected elements (such as an engine, a robotic arm, or the skeleton of the human body).

When we talk about robot kinematics, we are talking about the robot's geometry and how it changes as the robot moves. If we are to comprehend how to move and operate the robot, we must first establish its kinematics.

Forward kinematics describes how we go 'forward' from the robot's base and establish the location of the end-effector by travelling from the base to each succeeding link which may be done using coordinate frames, relative motion of nearby links, and manipulator geometry. So, we begin at the bottom and work our way up to the end-effector. Forward kinematics is used when we need to find the position and orientation of the end effector from the given joint angles.

On the other hand, inverse kinematics is the opposite of forward kinematics and is used when we need to find the joint angles for a given position of the end effector. This method makes more sense in robotics as most of the time we want the robot to position its tool to a particular location or particular X, Y and Z coordinates.

We talked about how we need to put reference frames on each link and extremely common approach used is the Denavit-Hartenberg (DH) Convention. The DH method is used to attach reference frames to the links of a kinematic chain, similar to how a typical robot manipulator does. When it comes to giving reference frames to robot connections, this is a common convention. There's also a modified DH convention that's occasionally utilized. It was invented by Jacques Denavit and Richard Hartenberg in 1955, but it wasn't until almost 30 years later that its potential in robotics was discovered. The DH convention is just a process for defining the relative positions of robot links and joints and will be used in this paper to find the robots kinematics. This may be done by employing a number of reference frames. A coordinate frame is attached to each joint to determine DH parameters.

First, a technique for numbering the joints and linkages is needed. Link 0 is the robot's base, usually for stationary robots. The first link after the base – the first moving link – is link 1. And so on and so forth. The numbering for joints is slightly different. The number one is allocated to the first joint, two to the second, and so on. Now we need to specify the connections and joints, as well as how they are positioned and orientated in relation to one another. Four parameters in total will be needed which are called Denavit-Hartenberg parameters (known as ‘DH’ for short).

A link can be defined using two parameters. One of the characteristics is link length, which is the distance between the joints of the link. It's also necessary to comprehend the connection twist, which is a little more challenging. The angle measured from the link's end joint to the link's beginning joint is the twist. In other words, is the connection between two joints rotating in opposite directions "twisted"? The link's twist is zero if the two joints' axes are parallel.

The link offset and joint angle are the only two remaining parameters. The link offset describes how far apart the joints on either side of the link are from one another. If the connection is straight and connects both links, the offset will be zero. The joints will most likely be offset if the link has

a kink or bend in it. The joint angle is simply a measurement of how far down a connection the first joint is slanted in relation to the second joint.

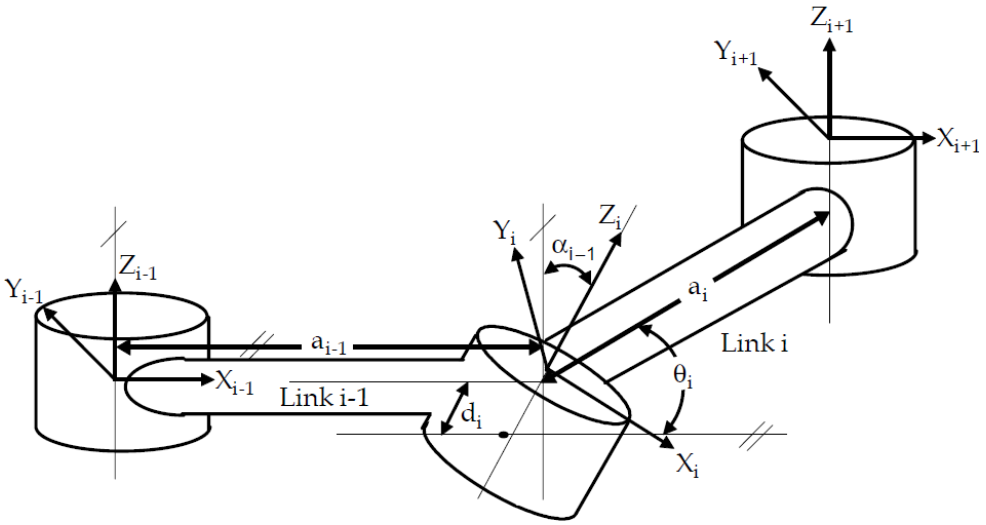


Fig.2.1 Coordinate frame assignment for a general manipulator.

The DH parameters are akin to the configuration of the robot. The kinematics equations for various manipulator architectures are not identical. Furthermore, manipulator kinematics equations based on the DH convention contain some singularity, making the equations difficult or impossible to solve in some instances. Furthermore, when the axes of two joints are parallel, the common normal is incorrectly determined by the DH standard. The DH technique has a singularity in this scenario [17], where a little change in the spatial coordinates of the parallel joint axes can result in a massive misconfiguration in the representation of the DH coordinates of their relative position and resolved by using another alternative method to DH known as Product of Exponentials (PH). This paper will focus on DH convention.

To sum up DH, as shown in Fig.2.1, Z_i axis of the coordinate frame is pointing along the rotary or sliding direction of the joints. The distance from Z_{i-1} to Z_i measured along X_{i-1} is assigned as a_{i-1} which is the link length, the angle between Z_{i-1} and Z_i measured along X_i is assigned as α_{i-1} which is the link twist, the distance from X_{i-1} to X_i measured along Z_i is assigned as d_i which is the link offset and the angle between X_{i-1} to X_i measured about Z_i is assigned as θ_i which is the link angle [18]. The general transformation matrix T_i^{i-1} for a single link can be obtained as follows [19].

$$T_i^{i-1} = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)Q_i(d_i)$$

$$T_i^{i-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where R_x and R_z present rotation, D_x and Q_i denote translation. The forward kinematics of the end-effector with respect to the base frame is determined by multiplying all of the T_i^{i-1}matrices. Where n represent number of links.

$$T_{end-effector}^{base} = T_1^0 T_2^1 \dots T_n^{n-1}$$

An alternative representation of $T_{end-effector}^{base}$ can be written as

$$T_{end-effector}^{base} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where r_{kj}'s represent the rotational elements of transformation matrix (k and j=1, 2 and 3). p_x, p_y and p_z denote the elements of the position vector.

2.2 Forward Kinematics

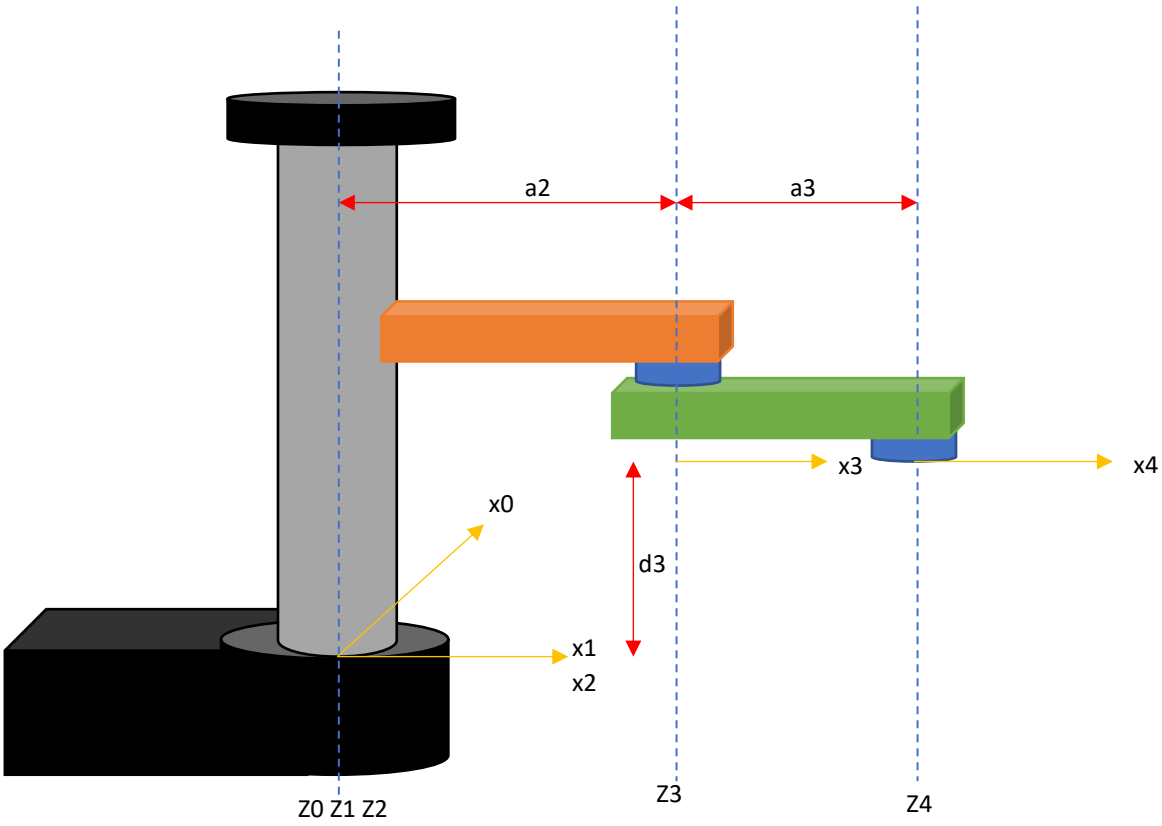


Fig.2.2 SCARA Robot with DH Parameters and Coordinate Frames

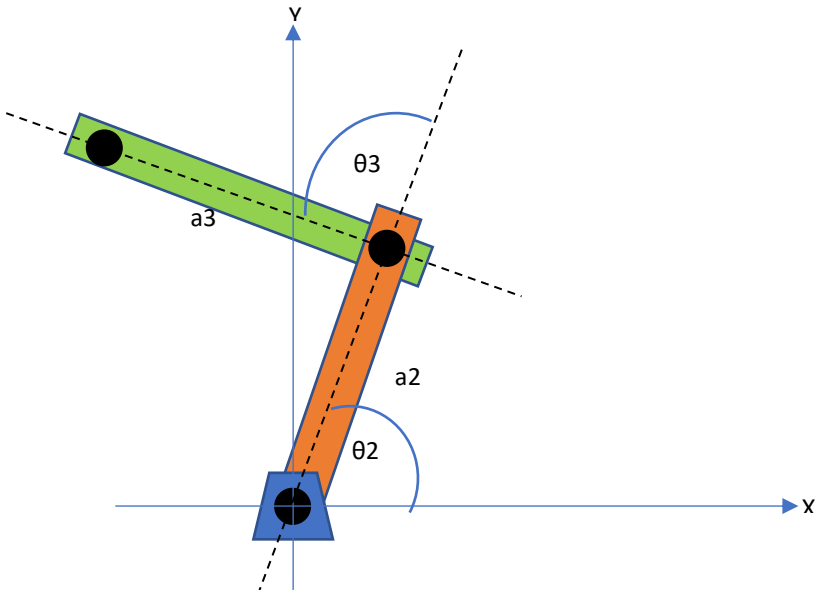


Fig.2.3 Simplified view X-Y Top view of SCARA Robot links

All of the calculations were done on MATLAB/Mupad and the main focus is the second and third as they make up the main movement of the robot which is the X and Y movement Transformation matrices are calculated in the upcoming in order to get the robots kinematics.

i	Link Number
α	Twist Angle
a	Link Length
d	Link Offset
θ	Link Angle
w	Link Width
m	Link Mass
x_2, x_3	x positions of the links
y_2, y_3	y positions of the links
z_2, z_3	z positions of the links

Table 2.1 DH Parameters Definitions Table

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	0
2	0	0	0	θ_2
3	0	a_2	d_3	θ_3
4	0	a_3	0	θ_4

Table 2.2 DH Parameters Table

$$T_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

$$T_3^2 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_2 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & a_3 \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} \cos(\theta_2 + \theta_3 + \theta_4) & -\sin(\theta_2 + \theta_3 + \theta_4) & 0 & a_2\cos(\theta_2) + a_3\cos(\theta_2 + \theta_3) \\ \sin(\theta_2 + \theta_3 + \theta_4) & \cos(\theta_2 + \theta_3 + \theta_4) & 0 & a_2\sin(\theta_2) + a_3\sin(\theta_2 + \theta_3) \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$T_1^0$is a matrix whose column vectors are the coordinates of the (unit vectors along the) axes of frame $O_1X_1Y_1Z_1$ expressed relative to frame $O_0X_0Y_0Z_0$. My interest was the last column of T_4^0. This is the X, Y, Z forward kinematic equations of the end effector. So, I got the equations (1) and (2) and (3).

$$x_3 = a_2\cos(\theta_2) + a_3\cos(\theta_2 + \theta_3) \quad (1)$$

$$y_3 = a_2\sin(\theta_2) + a_3\sin(\theta_2 + \theta_3) \quad (2)$$

$$z_3 = d_3 \quad (3)$$

From equation (1) and (2), I got the equations of displacement of link 2 and 3 by simply changing thetas with respect to time.

$x_2 = a_2 \cos(\theta_2(t))$ (4)

$y_2 = a_2 \sin(\theta_2(t))$ (5)

$x_3 = a_2 \cos(\theta_2(t)) + a_3 \cos(\theta_2(t) + \theta_3(t))$ (6)

$y_3 = a_2 \sin(\theta_2(t)) + a_3 \sin(\theta_2(t) + \theta_3(t))$ (7)

From equations (4) and (5) and (6) and (7), I got the equations of velocity of link 2 and 3 by simply differentiating with respect to time.

$v_{x2} = a_2 \theta_2'(t) \sin(\theta_2(t))$ (8)

$v_{y2} = a_2 \theta_2'(t) \cos(\theta_2(t))$ (9)

$v_{x3} = -a_2 \theta_2'(t) \sin(\theta_2(t)) - a_3 \sin(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t))$ (10)

$v_{y3} = a_2 \theta_2'(t) \cos(\theta_2(t)) + a_3 \cos(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t))$ (11)

From equations (8) and (9) and (10) and (11), I got the equations of acceleration of link 2 and 3 by simply differentiating with respect to time.

$a_{x2} = -a_2 \theta_2'(t)^2 \cos(\theta_2(t)) - a_2 \sin(\theta_2(t)) \theta_2''(t)$ (12)

$a_{y2} = -a_2 \theta_2'(t)^2 \sin(\theta_2(t)) + a_2 \cos(\theta_2(t)) \theta_2''(t)$ (13)

$a_{x3} = -a_3 \cos(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t))^2 - a_2 \sin(\theta_2(t)) \theta_2''(t) - a_2 \theta_2'(t)^2 \cos(\theta_2(t)) - a_3 \sin(\theta_2(t) + \theta_3(t))(\theta_2''(t) + \theta_3''(t))$ (14)

$a_{y3} = -a_3 \sin(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t))^2 + a_2 \cos(\theta_2(t)) \theta_2''(t) - a_2 \theta_2'(t)^2 \sin(\theta_2(t)) + a_3 \cos(\theta_2(t) + \theta_3(t))(\theta_2''(t) + \theta_3''(t))$ (15)

2.3 Inverse Kinematics

In order to derive the inverse kinematics solution analytically, two methodologies are used: geometric and algebraic. The geometry becomes significantly more laborious for manipulators with more connections and arms that extend into three dimensions. To find the inverse kinematics solution using algebra, it should be solved for q_i as function of the known elements of $T_{end-effector}^{base}$. qi is the joint variable (revolute joint or prismatic joint) for joint i. To find q_1 firstly as a function of the known elements, the link transformation inverses are pre-multiplied as follows [20].

$$[T_1^0]^{-1} T_{end-effector}^{base} = T_2^1 \dots T_n^{n-1}$$

To find the other joint variables, obtained in a similar manner.

$$[T_2^1]^{-1}[T_1^0]^{-1} T_{end-effector}^{base} = T_3^2 \dots T_n^{n-1}$$

For a 4 DOF robot there are 8 simultaneous set of nonlinear equations to be solved which is simply too much work specially when there is another faster and easier way. Hence, algebraic approach is chosen for the inverse kinematics solution as the geometry is simple and method is very straightforward. The geometric approach has proven to be much easier than algebraic approach.

Using Pythagoras theorem on the right triangle (law of cosines) on equations (1) and (2) to find the inverse kinematics.

$$x_3^2 + y_3^2 = (a_2 \cos(\theta_2) + a_3 \cos(\theta_2 + \theta_3))^2 + ((a_2 \sin(\theta_2) + a_3 \sin(\theta_2 + \theta_3))^2 \quad (16)$$

Simplifying equation (16), I got:

$$x_3^2 + y_3^2 = a_2^2 + 2a_2 a_3 \cos(\theta_3) + a_3^2 \quad (17)$$

Solving for θ_3 in equation (17) and assuming a_2 and a_3 are greater than zero:

$$\theta_3 = arcos(\frac{x_3^2 + y_3^2 - a_3^2 - a_2^2}{2a_2 a_3}) \quad (18)$$

Expanding (1) and (2) and rearranging:

$$x_3 = (a_2 + a_3\cos(\theta_3))\cos(\theta_2) + (-a_3\sin(\theta_3))\sin(\theta_2) \quad (19)$$

$$y_3 = (a_3\sin(\theta_3))\cos(\theta_2) + (a_2 + a_3\cos(\theta_3))\sin(\theta_2) \quad (20)$$

Putting (19) and (20) in matrix form AX = B:

$$A = \begin{bmatrix} a_2 + a_3\cos(\theta_3) & -a_3\sin(\theta_3) \\ a_3\sin(\theta_3) & a_2 + a_3\cos(\theta_3) \end{bmatrix}$$

$$X = \begin{bmatrix} \cos(\theta_2) \\ \sin(\theta_2) \end{bmatrix}$$

$$B = \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$$

Solving for X and dividing $\sin(\theta_2)$ by $\cos(\theta_2)$ and making θ_2 the subject, I got:

$$\theta_2 = \arctan\left(\frac{y_3a_2 + y_3a_3\cos(\theta_3) - x_3a_3\sin(\theta_3)}{x_3a_2 + x_3a_3\cos(\theta_3) + y_3a_3\sin(\theta_3)}\right) \quad (21)$$

The atan2 function calculates a point's polar angle, arctan(x/y), using its cartesian coordinates. It is extensively used to recover the phase of a signal in digital signal processing. This function returns an angle in the closed interval [-pi, pi] and is part of the standard mathematics library. Atan2(x, y) retains track of the relative signs of x and y, unlike a simple division and the arctan function which returns an angle in [-pi/2, pi/2]. It's used to find the phase of a complex number, such as x + iy [21]. Thus, instead of using arctan, atan2 function will be used.

$$\theta_2 = \text{atan2}\left(y_3a_2 + y_3a_3\cos(\theta_3) - x_3a_3\sin(\theta_3),\ x_3a_2 + x_3a_3\cos(\theta_3) + y_3a_3\sin(\theta_3)\right) \quad (22)$$

2.4 Dynamics

There are two commonly used methods to get the dynamics of a system. One of them is the Euler-Lagrange method and the other is the Newtonian approach. Newtonian mechanics use forces and constraints. The vectorial structure of the Newtonian force-momentum formulation includes cause and effect. The Newtonian approach's directional features help with understanding when putting up a problem as it's based on Based on Newton's laws of motion.

Lagrangian mechanics use energies and generalized coordinates. The Lagrangian method is expressed in terms of kinetic and potential energies, both of which are based on scalar functions, and the equations of motion are derived from a single scalar function, i.e., Lagrangian. The Lagrangian technique is theoretically easier when the mechanical system is more complicated. Lagrangian mechanics is also more extensible to other physical theories than Newtonian mechanics.

The two methods usually produce the same equations. However, in problems involving more than one variable, it usually turns out to be much easier to write down kinetic energies(T) and potential energies(V), as opposed to writing down all the forces. This is because T and V are nice and simple scalars. The forces, on the other hand, are vectors, and it is easy to get confused as they can point in various directions. The Lagrangian method has the advantage that once L = T – V is written down, it's just a matter of getting some derivatives [22]. The Lagrangian approach will be used to get the dynamics of this system.

To get dynamics, everything must be with respect to center of mas for easier calculation.

Center of mass displacements:

$$x_{cm2} = \frac{a_2}{2}\cos\left(\theta_2(t)\right) \tag{23}$$

$$y_{cm2} = \frac{a_2}{2}\sin\left(\theta_2(t)\right) \tag{24}$$

$$x_{cm3} = \frac{a_2}{2}\cos(\theta_2(t)) + \frac{a_3}{2}\cos\left(\theta_2(t) + \theta_3(t)\right) \tag{25}$$

$$y_{cm3} = \frac{a_2}{2}\sin(\theta_2(t)) + \frac{a_3}{2}\sin\left(\theta_2(t) + \theta_3(t)\right) \tag{26}$$

Differentiating (23) (24) (25) (26), I got center of mass velocities:

$v_{cmx2} = \frac{a_2}{2}\theta_2'(t)\sin(\theta_2(t))$ (27)

$v_{cmy2} = \frac{a_2}{2}\theta_2'(t)\cos(\theta_2(t))$ (28)

$v_{cmx3} = -a_2\theta_2'(t)\sin(\theta_2(t)) - \frac{a_3}{2}\sin(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t)$ (29)

$v_{cmy3} = a_2\theta_2'(t)\cos(\theta_2(t)) + \frac{a_3}{2}\cos(\theta_2(t) + \theta_3(t))(\theta_2'(t) + \theta_3'(t)$ (30)

Mass moments of inertia about center of mass:

$I_{cm2} = \frac{m_2(a_2^2 + w_2^2)}{12}$ (31)

$I_{cm3} = \frac{m_3(a_3^2 + w_3^2)}{12}$ (32)

Euler-Lagrange equation of motion:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} - \frac{\partial L}{\partial q_k} = \tau_k; k = 1 \dots n \quad (33)$$

where n is the DOF of the system, $\{q_1 \dots q_n\}$ is a set of generalized coordinates, $\{\tau_1 \dots \tau_n\}$ is the set of generalized forces associated with those coordinates, and the Lagrangian: L=K-P, is defined as the difference between the kinetic and potential energy of the n-DOF system.

$q = \{\theta_2, \theta_3\}$

$\tau = \{\tau_2, \tau_3\}$

Kinetic energy of system is the total of kinetic energy of second link (rotational and translational) and kinetic energy of third link (rotational and translational):

$$K = \frac{a_2^2 m_3 \theta_2'(t)^2}{2} + \frac{a_3^2 m_3 \theta_2'(t)^2}{6} + \frac{a_3^2 m_3 \theta_3'(t)^2}{6} + \frac{m_3 w_3^2 \theta_2'(t)^2}{24} + \frac{m_3 w_3^2 \theta_3'(t)^2}{24} + \frac{m_2 \theta_2'(t)^2 (4a_2^2 + w_2^2)}{24} + \frac{a_3^2 m_3 \theta_2'(t)\theta_3'(t)}{3} + \frac{m_3 w_3^2 \theta_2'(t)\theta_3'(t)}{12} + \frac{a_2 a_3 m_3 \theta_2'(t)^2 \cos(\theta_3(t))}{2} + \frac{a_2 a_3 m_3 \theta_2'(t)\theta_3'(t)\cos(\theta_3(t))}{2} \quad (34)$$

Potential energy of system will most likely be unchanging as the robot's main movement is in the X-Y direction.

$$P = gm_2z_{cm2} + gm_3z_{cm3} \quad (35)$$

The Lagrangian will be as follow:

$$L = \frac{a_2^2m_3\theta_2'(t)^2}{2} + \frac{a_3^2m_3\theta_2'(t)^2}{6} + \frac{a_3^2m_3\theta_3'(t)^2}{6} + \frac{m_3w_3^2\theta_2'(t)^2}{24} + \frac{m_3w_3^2\theta_3'(t)^2}{24} + \frac{m_2\theta_2'(t)^2(4a_2^2+w_2^2)}{24} + \frac{a_3^2m_3\theta_2'(t)\theta_3'(t)}{3} + \frac{m_3w_3^2\theta_2'(t)\theta_3'(t)}{12} + \frac{a_2a_3m_3\theta_2'(t)^2\cos(\theta_3(t))}{2} + \frac{a_2a_3m_3\theta_2'(t)\theta_3'(t)\cos(\theta_3(t))}{2} - gm_2z_{cm2} - gm_3z_{cm3} \quad (36)$$

Using the Lagrange equation, I got:

$$a_2^2m_3\theta_2''(t) + \frac{a_3^2m_3\theta_2''(t)}{3} + \frac{a_3^2m_3\theta_3''(t)}{3} + \frac{m_3w_3^2\theta_2''(t)}{12} + \frac{m_3w_3^2\theta_3''(t)}{12} + \frac{m_2\theta_2''(t)(4a_2^2+w_2^2)}{12} + a_2a_3m_3\cos(\theta_3(t))\,\theta_2''(t) + \frac{a_2a_3m_3\cos(\theta_3(t))\theta_3''(t)}{2} - \frac{a_2a_3m_3\theta_3'(t)^2\sin(\theta_3(t))}{2} - a_2a_3m_3\theta_2'(t)\theta_3'(t)\sin(\theta_3(t)) = \tau_2 \quad (37)$$

Rearranging and gathering like terms:

$$\left(a_2^2m_3 + \frac{a_3^2m_3}{3} + \frac{m_3w_3^2}{12} + \frac{m_2(4a_2^2+w_2^2)}{12} + a_2a_3m_3\cos(\theta_3(t))\right)\theta_2''(t) + \left(\frac{a_3^2m_3}{3} + \frac{m_3w_3^2}{12} + \frac{a_2a_3m_3\cos(\theta_3(t))}{2}\right)\theta_3''(t) + \left(-a_2a_3m_3\sin(\theta_3(t))\right)\theta_2'(t)\theta_3'(t) + \left(-\frac{a_2a_3m_3\sin(\theta_3(t))}{2}\right)\theta_3'(t)^2 = \tau_2 \quad (38)$$

Term associated with $\theta_2''(t)$ is the effective inertia of link 1.

Term associated with $\theta_3''(t)$ is the coupling inertia.

Term associated with $\theta_2'(t)\theta_3'(t)$ is the inertia associated with Coriolis acceleration.

Term associated with $\theta_3'(t)^2$ is the inertia associated with centripetal acceleration.

And the second equation is:

$$\frac{a_3^2 m_3 \theta_2''(t)}{3} + \frac{a_3^2 m_3 \theta_3''(t)}{3} + \frac{m_3 w_3^2 \theta_2''(t)}{12} + \frac{m_3 w_3^2 \theta_3''(t)}{12} + \frac{a_2 a_3 m_3 \cos(\theta_3(t))\theta_2''(t)}{2} - \frac{a_2 a_3 m_3 \theta_2'(t)^2 \sin(\theta_3(t))}{2} = \tau_3 \quad (39)$$

Rearranging and gathering like terms:

$$\left(\frac{a_3^2 m_3}{3} + \frac{m_3 w_3^2}{12}\right)\theta_3''(t) + \left(\frac{a_3^2 m_3}{3} + \frac{m_3 w_3^2}{12} + \frac{a_2 a_3 m_3 \cos(\theta_3(t))}{2}\right)\theta_2''(t) + \left(\frac{a_2 a_3 m_3 \sin(\theta_3(t))}{2}\right)\theta_2'(t)^2 = \tau_3 \quad (40)$$

Term associated with $\theta_3''(t)$ is the effective inertia of link 2.

Term associated with $\theta_2''(t)$ is the coupling inertia.

Term associated with $\theta_2'(t)^2$ is the inertia associated with centripetal acceleration.

To get equations there are two approaches: the first analytical and the other using matrices which is much more reliable.

I. Analytical Approach:

Factor $\theta_2''(t)$ out of equation (38):

$$\theta_2''(t) = \frac{T_2 - \frac{a_3^2 m_3 \theta_3''(t)}{3} - \frac{m_3 w_3^2 \theta_3''(t)}{12} - \frac{a_2 a_3 m_3 \cos(\theta_3(t))\,\theta_3''(t)}{2} + \frac{a_2 a_3 m_3 \theta_3'(t)^2 \sin(\theta_3(t))}{2} + a_2 a_3 m_3 \theta_2'(t)\,\theta_3'(t)\sin(\theta_3(t))}{\frac{m_2\left(4 a_2^2 + w_2^2\right)}{12} + a_2^2 m_3 + \frac{a_3^2 m_3}{3} + \frac{m_3 w_3^2}{12} + a_2 a_3 m_3 \cos(\theta_3(t))} \quad (41)$$

Factor $\theta_3''(t)$ out of equation (40):

$$\theta_3''(t) = -\frac{m_3\,\theta_2''(t)\,a_3^2\,4 + a_2 m_3 \cos(\theta_3(t))\,\theta_2''(t)\,a_3\,6 + m_3\,\theta_2''(t)\,w_3^2 + a_2 m_3 \sin(\theta_3(t))\,\theta_2'(t)^2\,a_3\,6 - 12\,T_3}{m_3\left(4\,a_3^2 + w_3^2\right)} \quad (42)$$

II. Matrix approach:

Putting equations (38) and (40) in matrix form AZ + B = C

$$A = \begin{pmatrix} \frac{m_2\left(4\,{a_2}^2+{w_2}^2\right)}{12}+{a_2}^2\,m_3+\frac{{a_3}^2\,m_3}{3}+\frac{m_3\,{w_3}^2}{12}+a_2\,a_3\,m_3\cos(\theta_3(t)) & \sigma_1 \\ \sigma_1 & \frac{m_3\left(4\,{a_3}^2+{w_3}^2\right)}{12} \end{pmatrix}$$

where

$$\sigma_1 = \frac{m_3\,{a_3}^2}{3}+\frac{a_2\,m_3\cos(\theta_3(t))\,a_3}{2}+\frac{m_3\,{w_3}^2}{12}$$

$$Z = \begin{pmatrix} \theta_2''(t) \\ \theta_3''(t) \end{pmatrix}$$

$$B = \begin{pmatrix} -\frac{a_2\,a_3\,m_3\,\theta_3'(t)\sin(\theta_3(t))\left(2\,\theta_2'(t)-\theta_3'(t)\right)}{2} \\ \frac{a_2\,a_3\,m_3\,\theta_2'(t)^2\sin(\theta_3(t))}{2} \end{pmatrix}$$

$$C = \begin{pmatrix} T_2 \\ T_3 \end{pmatrix}$$

Solving for Z I got:

$$\begin{pmatrix} \theta_2''(t) \\ \theta_3''(t) \end{pmatrix} = \begin{pmatrix} \frac{12\left(4\,a_3^{\,2}+w_3^{\,2}\right)\sigma_1}{\sigma_4} - \frac{12\,\sigma_2\,\sigma_3}{\sigma_4} \\ \frac{12\,\sigma_2\left(\sigma_5+12\,a_2^{\,2}\,m_3+4\,a_3^{\,2}\,m_3+m_3\,w_3^{\,2}+12\,a_2\,a_3\,m_3\cos\left(\theta_3(t)\right)\right)}{m_3\,\sigma_4} - \frac{12\,\sigma_1\,\sigma_3}{\sigma_4} \end{pmatrix}$$

where

$$\sigma_1 = T_2 + \frac{a_2\,a_3\,m_3\,\theta_3'(t)\sin\left(\theta_3(t)\right)\left(2\,\theta_2'(t)-\theta_3'(t)\right)}{2}$$

$$\sigma_2 = T_3 - \frac{a_2\,a_3\,m_3\,\theta_2'(t)^2\sin\left(\theta_3(t)\right)}{2}$$

$$\sigma_3 = 4\,a_3^{\,2} + 6\,a_2\cos(\theta_3(t))\,a_3 + w_3^{\,2}$$

$$\sigma_4 = 4\,a_3^{\,2}\,\sigma_5 + w_3^{\,2}\,\sigma_5 + 48\,a_2^{\,2}\,a_3^{\,2}\,m_3 + 12\,a_2^{\,2}\,m_3\,w_3^{\,2} - 36\,a_2^{\,2}\,a_3^{\,2}\,m_3\cos(\theta_3(t))^2$$

$$\sigma_5 = m_2\left(4\,a_2^{\,2} + w_2^{\,2}\right)$$

2.5 Linearization and state space representation

Looking at equations (38) and (40) it can be clearly seen that the equation of motion of SCARA robot can be written like the following:

$$\tau = M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right) + G(\theta) \quad (43)$$

where M is n*n moment of inertia matrix

C is n*1 the vector with centrifugal and Coriolis effects

G is a n*1 vector with gravitational effect

G (θ) is usually zero in SCARA robot as gravity does not affect the robot and as previously said potential energy of system will most likely be unchanging as the robot's main movement is in the X-Y direction. To simplify the construction of a non-linear resilient controller, start with a basic issue. Ignoring non-linear terms. Each arm functions as a load system that is linked to the motor. Then, for each arm, consider the following simple governing equation:

$$\tau = M_{eff}\ddot{\theta} + C_{eff}\dot{\theta} \quad (44)$$

Where $\ddot{\theta}$ is the acceleration

$\dot{\theta}$ is the velocity

τ is the required torque

M_{eff} is the effective moment of inertia

C_{eff} are the effective centrifugal and Coriolis effects

Also, it can be written approximately as [23]:

$$J_0(\theta)\ddot{\theta} + D_0(\theta)\dot{\theta}^2 + F_0\dot{\theta} = T \quad (45)$$

where $J_0(\theta) = \begin{bmatrix} J_{11} & J_{12}\cos(\theta_3 - \theta_2) \\ J_{21}\cos(\theta_3 - \theta_2) & J_{22} \end{bmatrix}$

$D_0(\theta) = \begin{bmatrix} 0 & J_{21}\cos(\theta_3 - \theta_2) \\ -J_{21}\sin(\theta_3 - \theta_2) & 0 \end{bmatrix}$

$F_0(\theta) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$; Dissipative energy due to friction for example is not considered

$J_{11} = a_2^2 m_3 + \frac{a_2^2 m_2}{3} + \frac{m_2 w_2^2}{12}$

$J_{12} = \frac{a_2 a_3 m_3 \cos(\theta_3(t))}{2}$

$J_{21} = J_{12}$

$J_{22} = m_3(\frac{4a_3^2 + w_3^2}{12})$

First, I had to perform all the kinematics and dynamics again but with respect to Fig.2.6 and then rearranging to get equation (45). Then linearizing around $\theta_2 = \theta_3$ I got $\sin(\theta_3 - \theta_2) = 0$ and $\cos(\theta_3 - \theta_2) = 1$ and I got the following state space representation:

$\dot{X} = AX + BU$

$Y = CX$

where $\dot{X} = \begin{bmatrix} \dot{\theta}_2 \\ \dot{\theta}_3 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix}$: (4 x 1)

$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -J_0(\theta)^{-1} \\ 0 & 0 & \end{bmatrix}$: (4 x 4)

$$X = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} : (4 \text{ x } 1)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ J_0(\theta)^{-1} \end{bmatrix} : (4 \text{ x } 2)$$

$$U = \begin{bmatrix} T2 \\ T3 \end{bmatrix} : (2 \text{ x } 1)$$

$$Y = \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix} : (2 \text{ x } 1)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} : (2 \text{ x } 4)$$

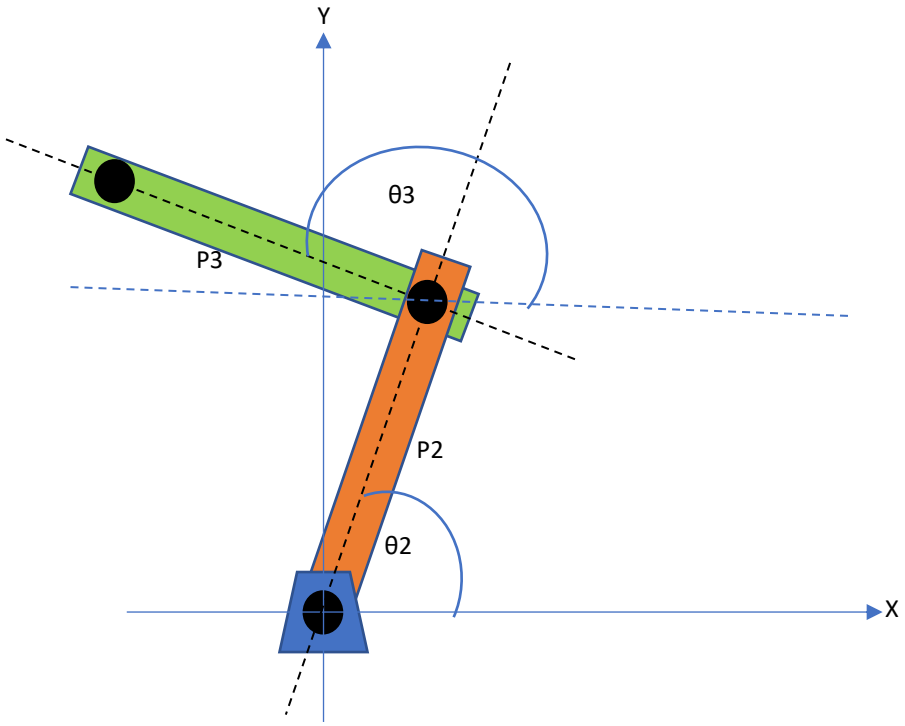


Fig.2.4 Simplified view X-Y Top view of SCARA Robot links with different coordinate setup

Chapter 3 Control Algorithms

3.1 Control Code

I will be using the Marlin firmware which is an open-source firmware for 3D printers. It is a simple, dependable, and customizable printer driver that simply works. Marlin also has the ability to control CNC machines and laser engravers. Instead of using Arduino IDE, I utilised PlatformIO IDE on VS code to install the Marlin firmware on the Arduino MEGA board. Because the Marlin firmware is large and sophisticated, the Arduino IDE may occasionally fail. In the following I will be discussing how I configured the firmware to work with the robot.

Once I downloaded the firmware, I went to the Configuration repository on Github.com from where I can download some pre-tested configuration files which are suitable for the robot. In the "Examples" folder I went to the "SCARA" folder and copied the configuration files of the "MP_SCARA" folder and paste them into our Marlin folder. Then I opened the marlin firmware using PlatformIO IDE, and started configuring and editing some parts to suit the robot.

First, I opened the Configuration.h file and edited some parameters according to the robot. The first thing I selected the type of SCARA robot. There are two which are the "MORGAN_SCARA" for parallel SCARA robot and the "MP_SCARA" for serial SCARA robot, just like the one I have. MP_SCARA is based on an open-source design by Tyler Williams which implements inverse kinematics on the G-codes generated for normal cartesian systems and which only depends on the lengths of the links similar to what I got in equation (18) and (22).

Then, I entered the length of the robot arms which are 228mm for the first arm and 136.5mm for the second arm. The next parameter is the zero position of the work area. This value is entered with the SCARA_OFFSET_X and Y values and it's the distance from the first joint relative to the to the work area zero position which will be in bottom left corner. In this case the SCARA_OFFSET_X is 240 mm and the SCARA_OFFSET_Y is -80 mm. The number signs are very crucial here.

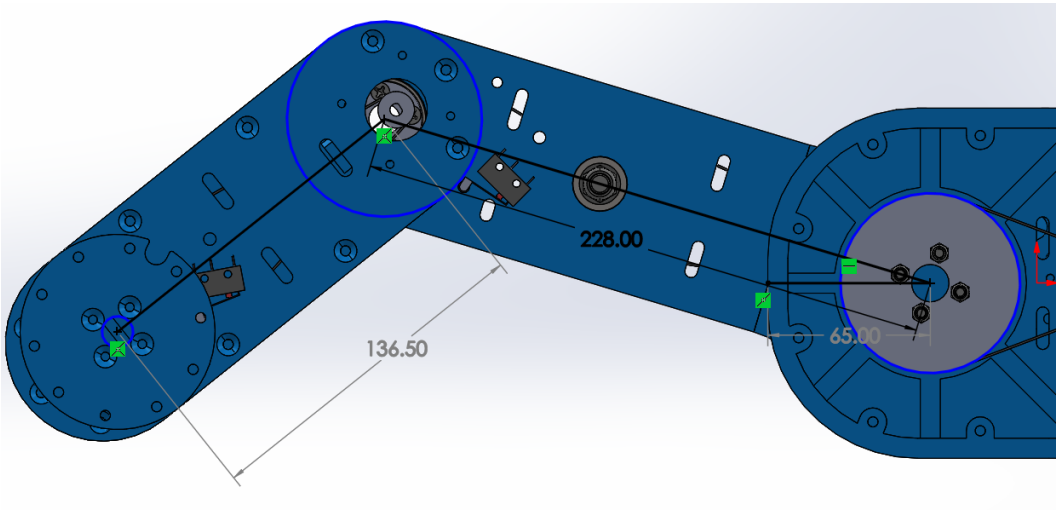


Fig.3.1 Robot model used to get precise lengths of robot arms

Then the MANUAL_X_HOME_POS for both X and Y were changed. These values are the X and Y distances from the robot arm end effector or the laser when it's in home position, relative to the work area zero position that was previously set. In order to visualise and get these values, I made a simple sketch in SOLIDWORKS with the two lines representing the two arms of the SCARA robot. In this case the MANUAL_X_HOME_POS was 240 mm and the MANUAL_Y_HOME_POS was -444.50 mm.

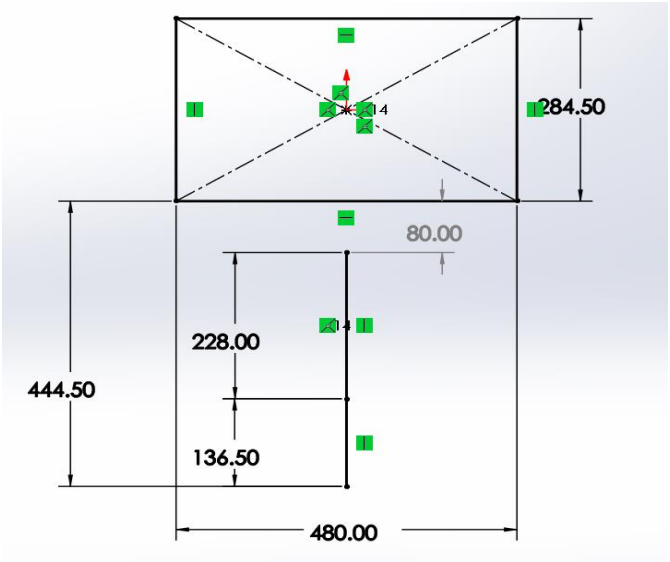


Fig.3.2 Sketch of work area and robot at home offset position

Next, I defined the end-stop connectors. In this case I have one end-stop for each axis, which is at minimum position. Next, I defined X_HOME_DIR to –1 which indicate the direction of the end-stops when homing. Also, I made sure all __MIN_ENDSTOP_INVERTING variables are set to “FALSE”.

Then I adjusted the DEFAULT_AXIS_STEPS_PER_UNIT values to match with the robot. These numbers indicate how many steps the software should transmit to the stepper drivers in order for the motor to travel one unit, which in this case represents a one-degree rotation of the joint.

The NEMA 17 stepper motors require 200 steps to complete a full rotation, but because I was using 16th step resolution, I will need to transmit 3200 steps to the driver for the motor to complete a full 360-degree revolution. In addition, the first joint has a 10.125:1 speed decrease, requiring 64000 steps to complete a full revolution. When I divided this amount by 360, I got 90, which is the number of steps that the software must transmit to the driver in order to spin the joint one unit or one degree. The reduction speed was calculated as follow number of teeth of driven pulley divided by number of teeth of driver pulley.

The second arm has a 3.286:1 speed decrease thus doing the same above-mentioned method I got the value 29.20634921. I was using an 8mm pitch lead screw for the Z axis, which implies that one full revolution of the motor will result in an 8mm linear monition. Because one unit equals one mm, I divided the 3200 steps by 8, yielding 400, which is the number of steps required to move the Z axis one unit or one mm. The fourth value is for the extruder stepper which can be used later to control the fourth motor and orientation of the end tool and it has a 1.323529412:1 speed decrease and doing the same math as before I got the fourth value as 11.76470588.

Furthermore, the direction of the stepper motor is determined by how the wires are connected. Direction can be controlled either by reversing the wires or adjusting it here in the firmware by changing the INVERT X DIR variables to TRUE if it is incorrect.

A few final changes I made as Marlin is primarily a firmware for classical cartesian style printers, the SCARA mode that it features, has some small bugs. On line 75 of scara.cpp, I changed MORGAN_SCARA to MP_SCARA. I had to comment lines 1716 and 1717 in motion.cpp. In

Configuration.h, commented VALIDATE_HOMING_ENDSTOPS, and In Configuration_adv.h, uncommented QUICK_HOME.

3.2 Interfacing with the robot

So, once I have loaded the Marlin firmware into the controller, I needed a computer software to operate the robot and transmit G-codes to it. There were many options but I went with Repetier-Host, a popular 3D printing program. This program is available for free download and usage. After I have installed it, I needed to make few changes in the software configuration to properly work the robot.

I selected the Baud Rate which I set in the Marlin firmware which was 115200. Then I adjusted the X and Y max values of printer Area which represent basically the work area of the robot.

Then I connected the robot and homed it. Before homing, I manually brought the joints closer to the limit switches. Then I started homing the robot and the robot started moving towards the limit switches. Actually, the Z axis started moving away from its limit switch then the second arm started moving towards its limit switch while the first arm moved also move towards its switch. Once the second arm or the Y axis hit its limit switch, the first arm or the X axis then started moving toward its limit switch as well. Once the X limit switch is reached, the robot went for the Z limit switch. In this case I noticed the robot was moving in the wrong direction, I simply reversed the wires of the stepper motors as mentioned before rather than changing the code. Sometimes the X axis or first arm does not home and in that case I home the X axis alone after the homing is done.

3.3 Generating G-Codes

Generating G-code to suit the Laser Engraving Scara I used Inkscape for that purpose which is an open-source vector graphics software.

I must first convert the picture to a vector format using the Trace Bitmap method. G-codes can only be produced from vector shapes; therefore, I utilised the Inkscape-Lasertools plugin for that.

This plugin may be downloaded from Github.com and installed by copying and pasting the files from the zip file into the Inkscape extensions directory.

With this plugin, I may configure and alter a variety of variables. I can activate the laser by sending the M106 instruction because the laser is attached to the D9 pin, which is used to drive a fan. If I wish to reduce the intensity of the laser in the infill region when selecting an infill, I may add a PWM value to the M106 command to set the laser intensity, for example, M106 S128 for 50% laser power. If I keep the laser intensity at 100% but increase the travel speed in the infill area, I can get a similar result.

I can load the G-code into the Repetier-Host application once I've produced it. I chose the check box "Show Travel Motions" to examine the G-code because this is not 3D printing, but rather travel moves.

Chapter 4 Results and Discussion

4.1 MATLAB Simulation for dynamics of system

The second order differential equations were solved using the MATLAB/Mupad numerical solution. First, I converted the two second order differential equations to a vector field of first order ode's suitable for numerical solution at a given initial condition using (numeric::ode2vectorfield()) function which changes a given system of second order differential equations and initial conditions to a vector field of first order odes. Then, solving the first order ode using the numerical solution tool (numeric::odesolve2()).

The system was first simulated with the following equations (41) and (42) and the initial condition and with variable values as:

$$\theta_2(0) = 0,\ \theta_2'(0) = 1,\ \theta_3(0) = 0, \theta_3'(0) = 1$$

$$m2 = 1, m3 = 1, a2 = 0.228, a3 = 0.1365, w2 = 0.078, w3 = 0.078, z_{cm_2} = 1, z_{cm_3} = 1, g = 9.8, T2 = 0, T3 = 0$$

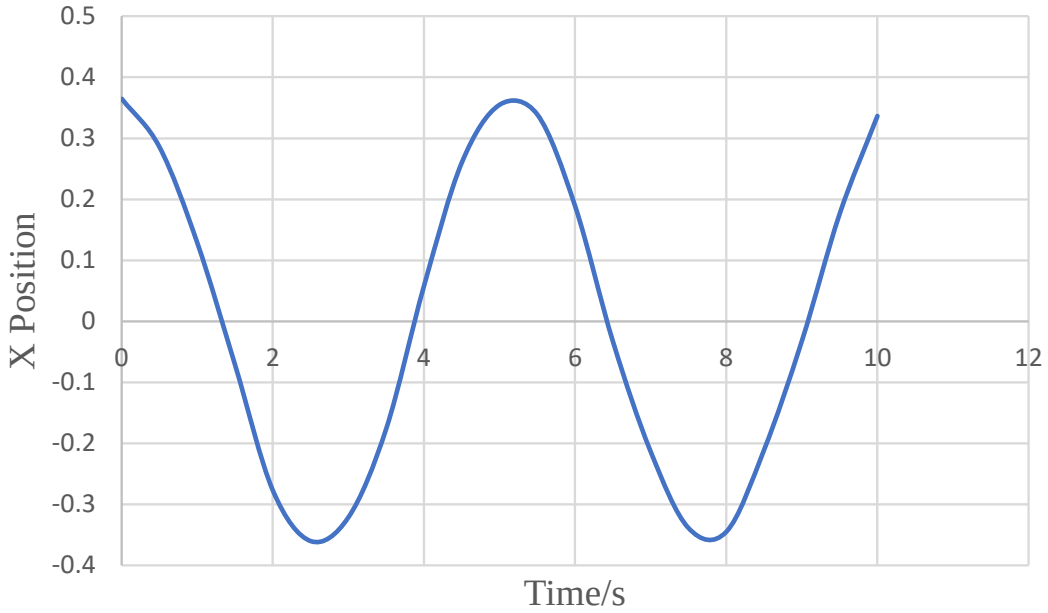


Fig.4.1 X position of end effector vs Time using equations (41) and (42)

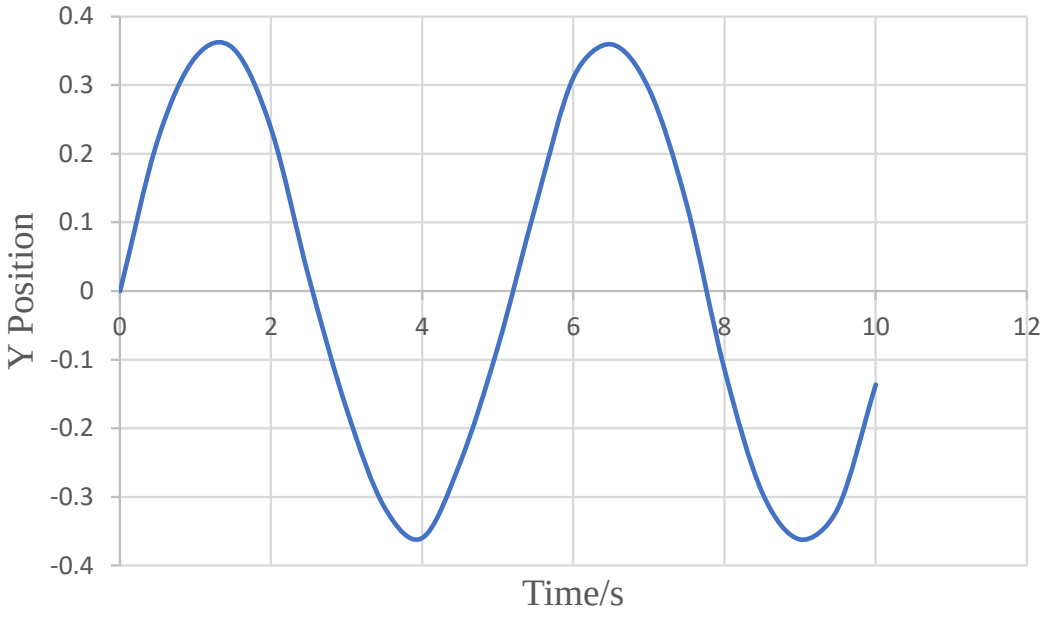


Fig.4.2 Y position of end effector vs Time using equations (41) and (42)

Then the system was simulated using the equations derived from the matrix Z and the same initial condition and with variable values as driven the previous system:

$\theta_2(0) = 0,\ \theta_2'(0) = 1,\ \theta_3(0) = 0, \theta_3'(0) = 1$

$m2 = 1, m3 = 1, a2 = 0.228, a3 = 0.1365, w2 = 0.078, w3 = 0.078, z_{cm_2} = 1, z_{cm_3} = 1, g = 9.8, T2 = 0, T3 = 0$

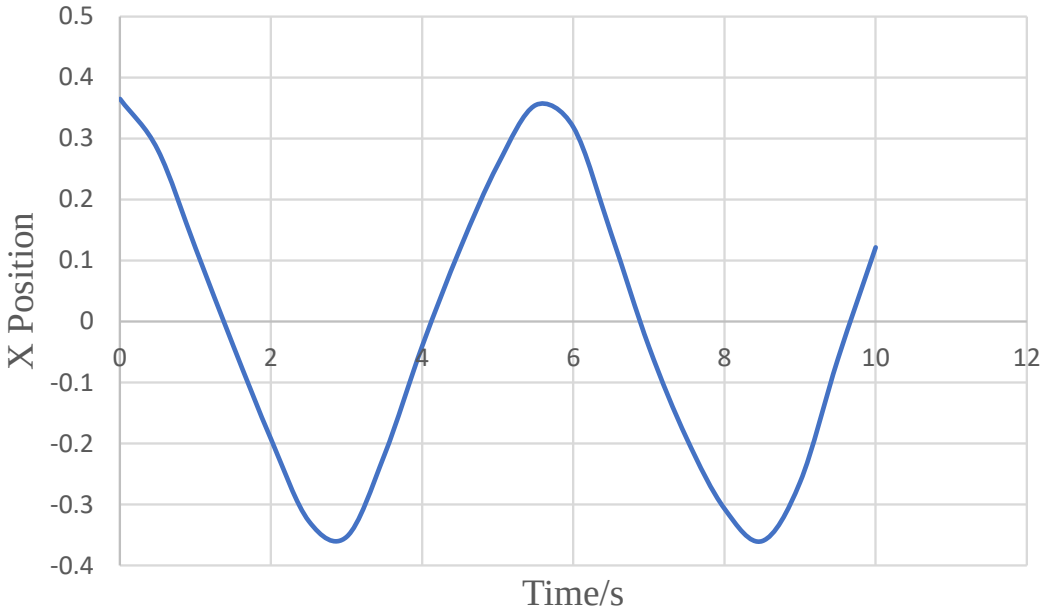


Fig.4.3 X position of end effector vs Time using matrix Z

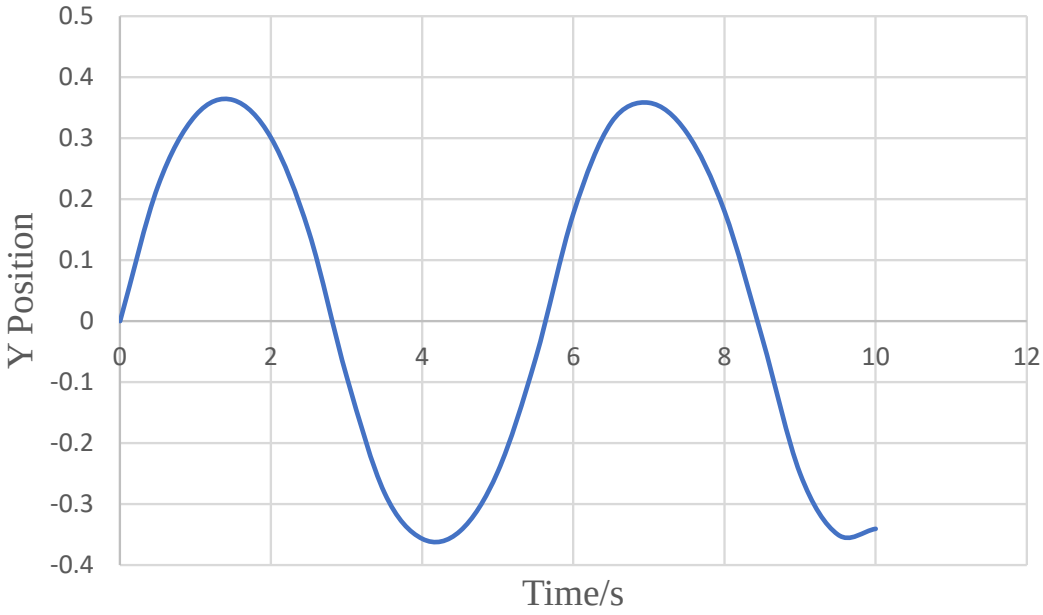


Fig.4.4 Y position of end effector vs Time using matrix Z

Both systems were then animated in the MATLAB/Mupad environment.

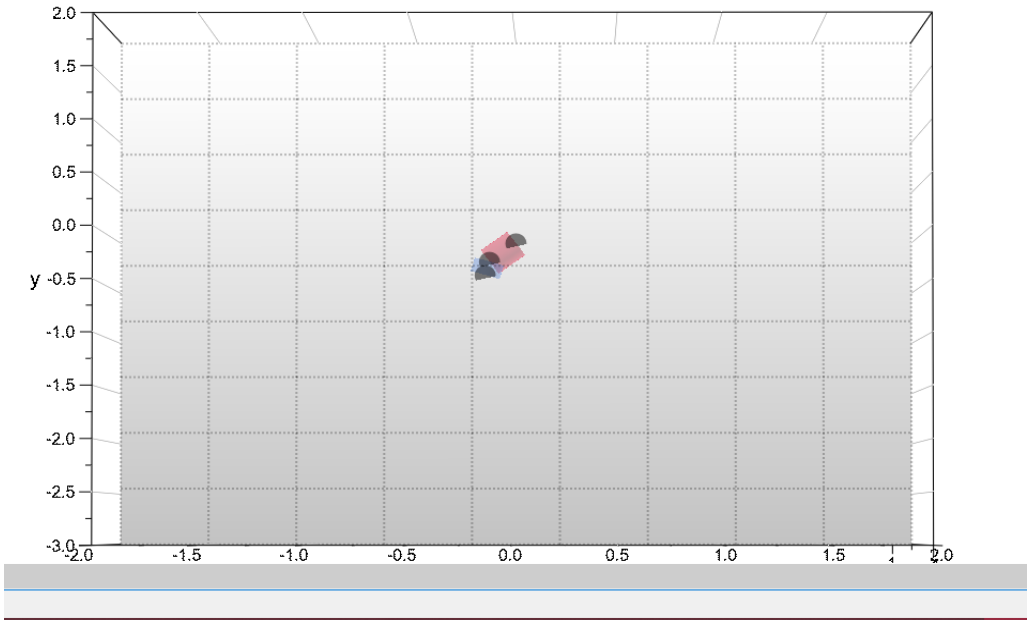


Fig.4.5 System animation in MATLAB/Mupad environment

Then the robot was setup in the Simulink Simscape Multibody environment and the arms were given the same initial conditions and same variable values as was done in MATLAB/Mupad environment.

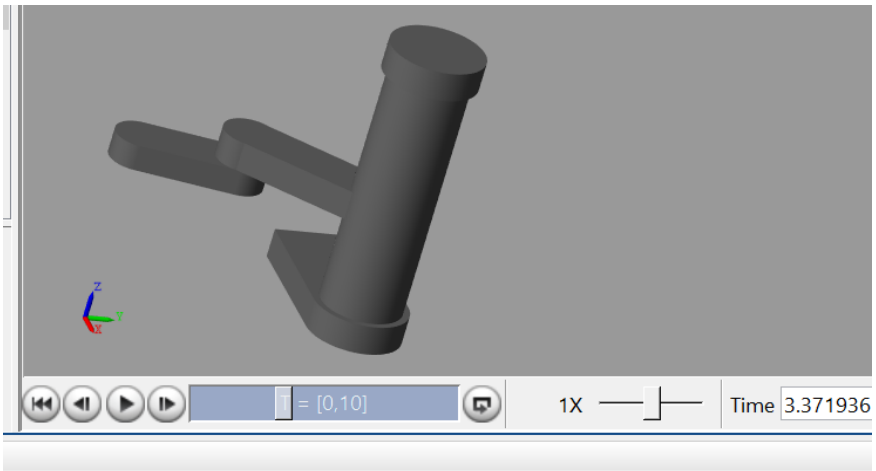


Fig.4.6 System animation in Simulink Simscape Multibody environment

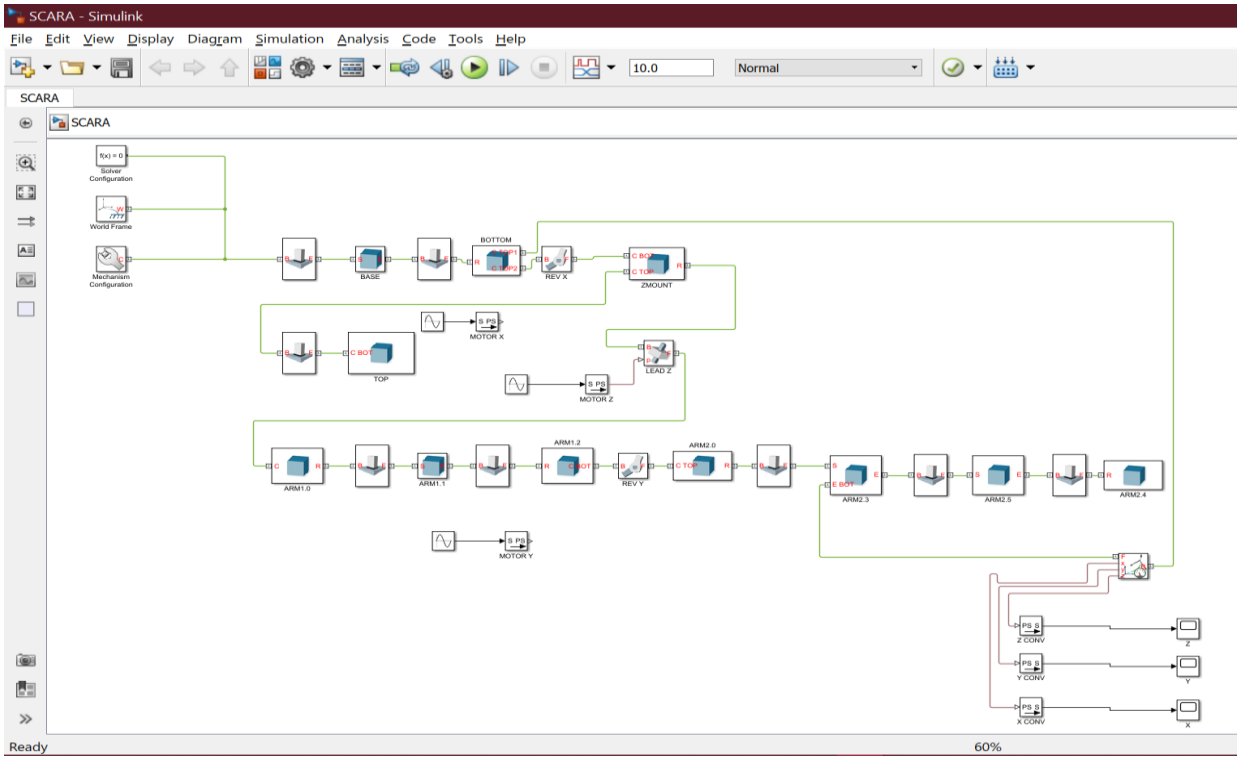


Fig.4.7 System setup in Simulink Simscape Multibody environment

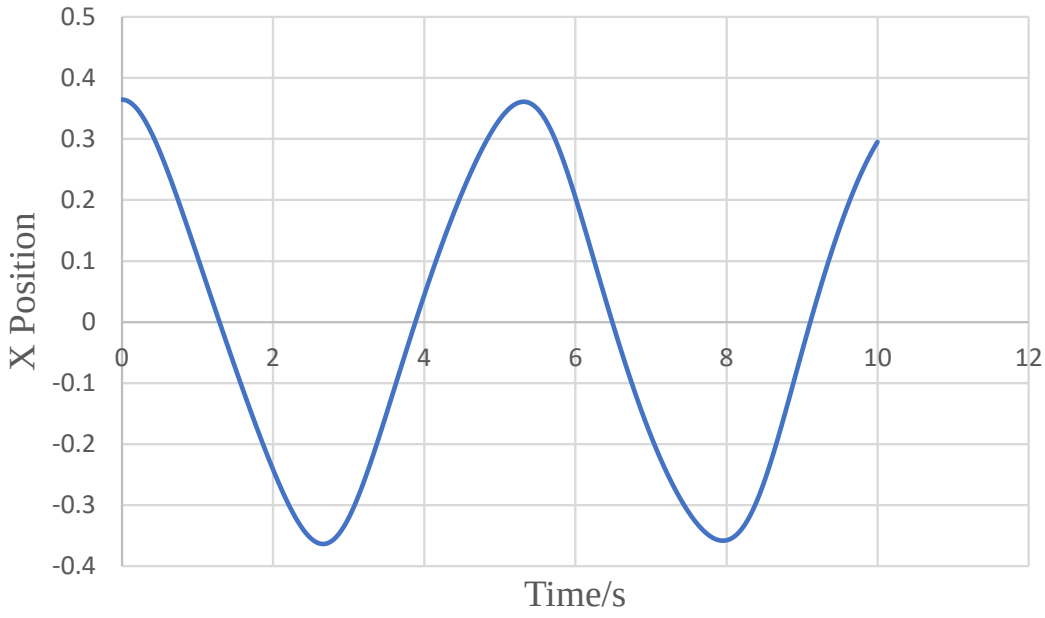


Fig.4.8 X position of end effector vs Time from Simulink Simscape Multibody environment

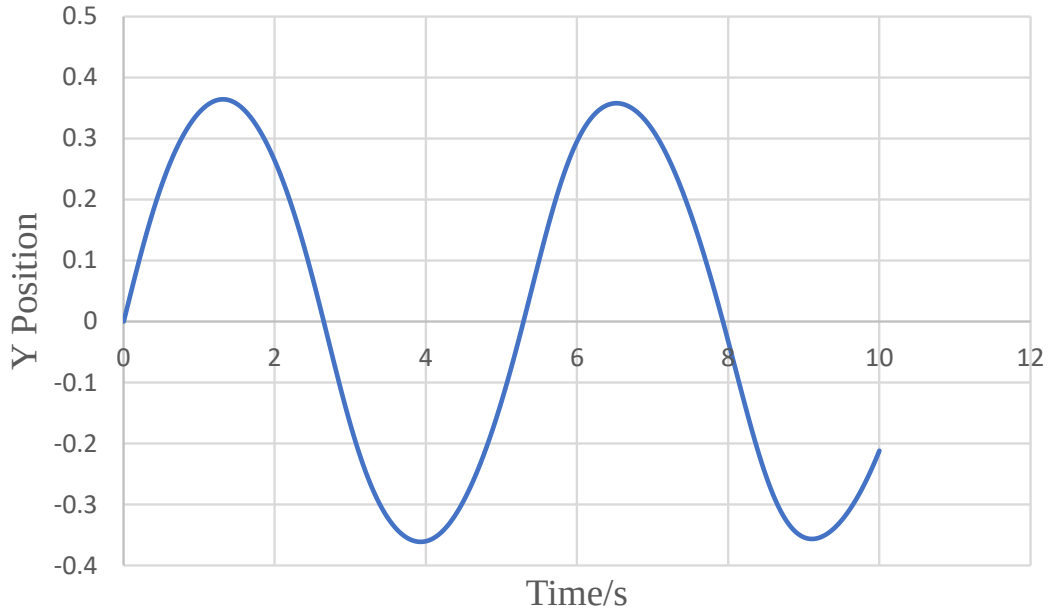


Fig.4.9 Y position of end effector vs Time from Simulink Simscape Multibody environment

It can be clearly seen that Fig.4.1 and Fig.4.2 and Fig.4.3 and Fig.4.4 are closely the same graph proving that both solutions are more or less the same and not only that but comparing those figures with Fig.4.8 and Fig.4.9, it is obvious to see that the simulation results closely match those of the numerical solution. Moreover, the animation of system on Fig.4.5 compared to those on Fig.4.6 were almost if not completely the same.

4.2 MATLAB Simulation for forward kinematics of system

The robot's links were forced to follow a certain motion in this case, link 2 angle followed a sinusoidal wave and also angle of link 3 followed a sinusoidal wave and x_3 and y_3 position were traced in MATLAB/Mupad environment using equation (1) and (2). In other words, $\theta_2 = \sin(t)$ and $\theta_3 = \sin(t)$ was used in equations (1) and (2) and solved during a certain interval and given certain link lengths which were $a2 = 0.228, a3 = 0.1365$.

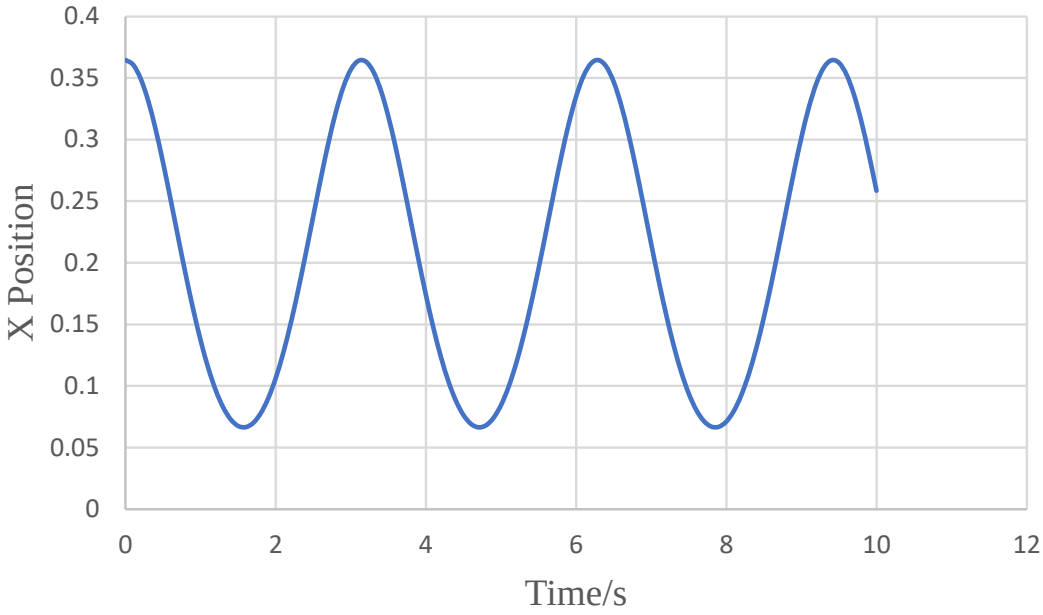


Fig.4.10 X position of end effector vs Time

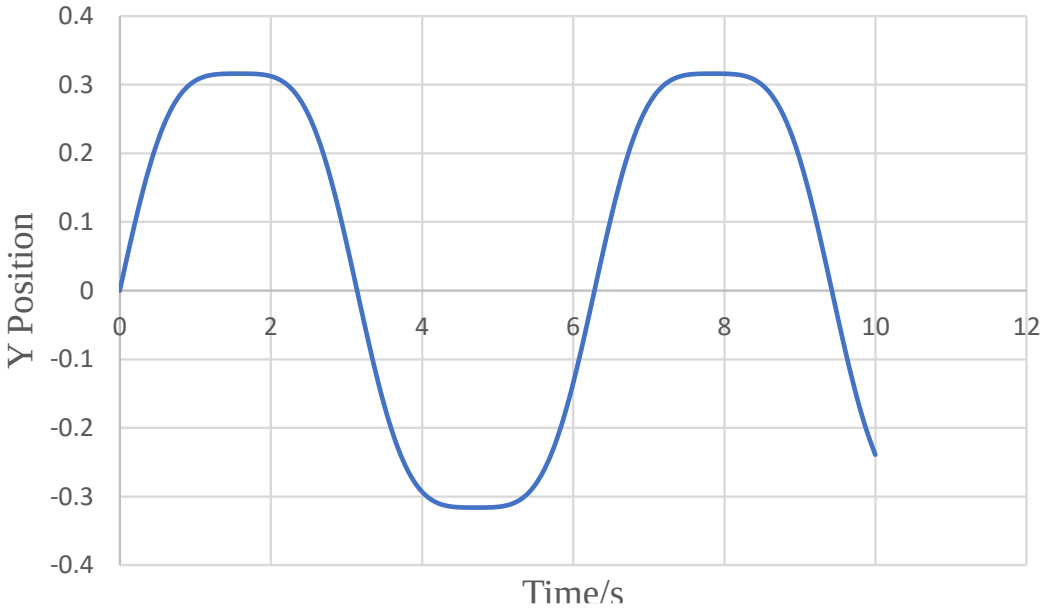


Fig.4.11 Y position of end effector vs Time

Then the robot was setup in the Simulink Simscape Multibody environment just like on Fig.4.7 with just a very small change that is the joint angles are given a certain motion to follow which is the same as was done in the MATLAB/Mupad environment. Sinusoidal motion was given to both joints and the end effector positions were traced.

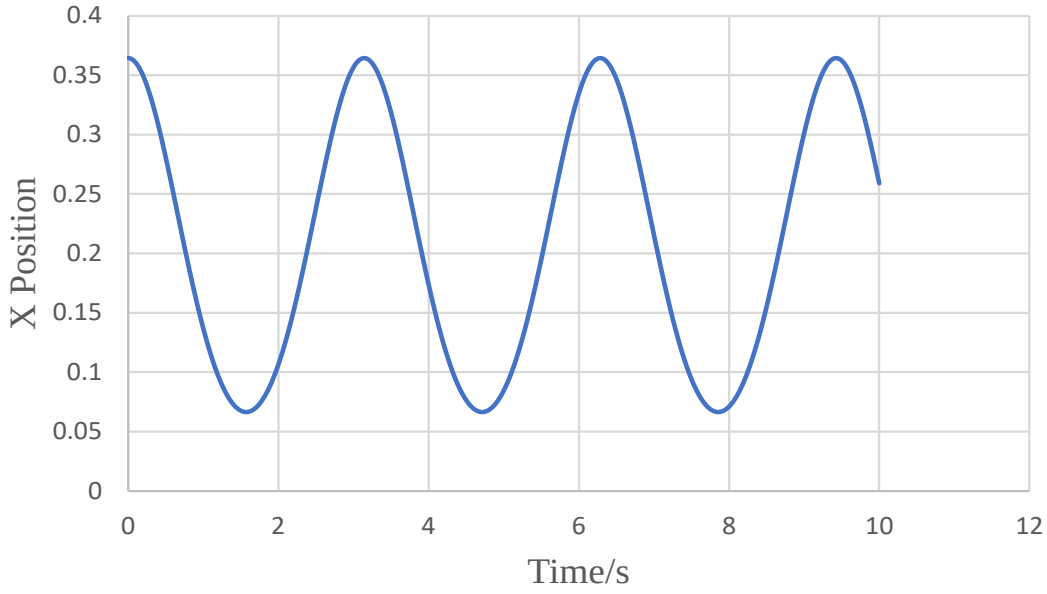


Fig.4.12 X position of end effector vs Time from Simulink Simscape Multibody environment

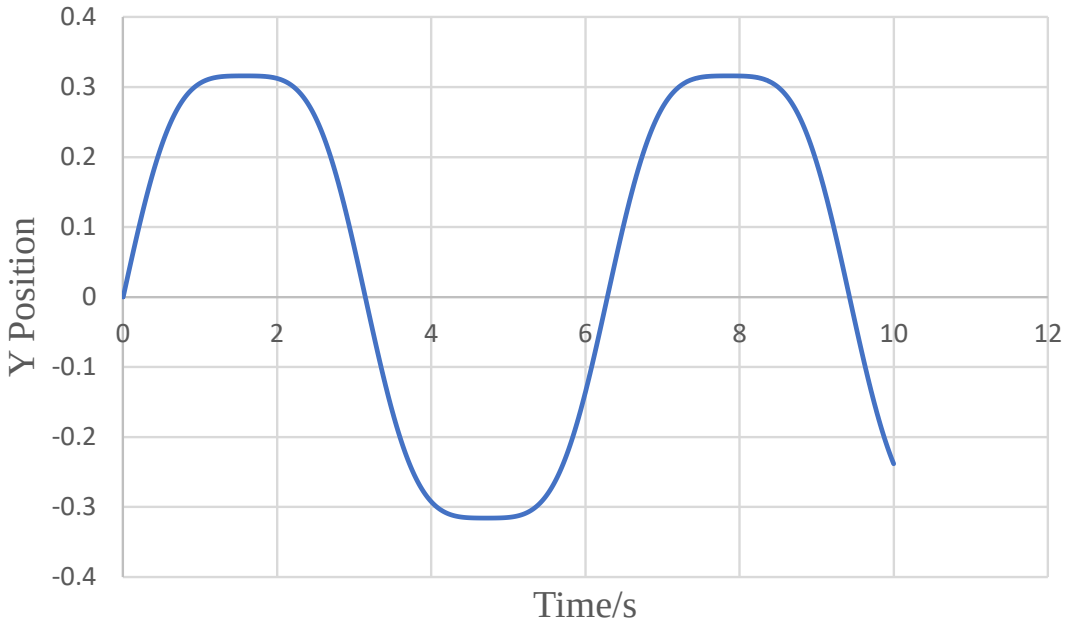


Fig.4.13 Y position of end effector vs Time from Simulink Simscape Multibody environment

It can be clearly seen that Fig.4.10 and Fig.4.11 are identical to those on Fig.4.12 and Fig.4.13 which proves the kinematic equations are correct. Animations similar to Fig.4.5 and Fig.4.6 were also done and in both animations the system behaved the same exact way.

4.3 MATLAB Simulation for inverse kinematics of system

The robot was setup in the Simulink Simscape Multibody environment and the robot was made to follow certain points inputted by the user and then these points were used to calculate the joint angles using the inverse kinematics equation derived and then inputting those joint angles to each respective joint. Then the end effector points were measured and compared it to the inputted points.

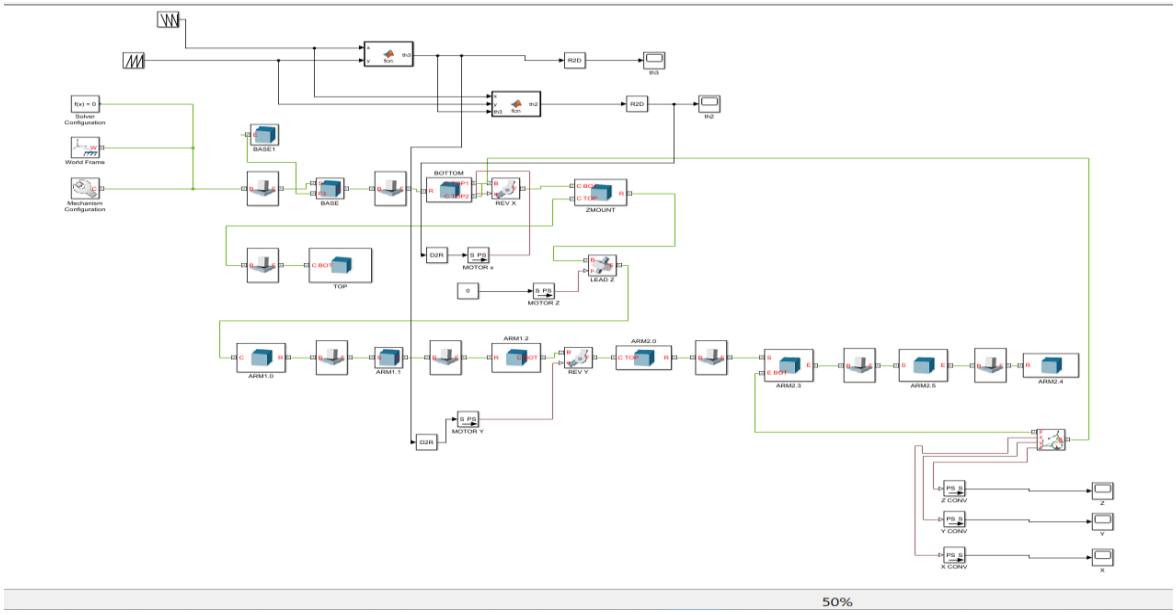


Fig.4.14 System setup in Simulink Simscape Multibody environment for inverse kinematics

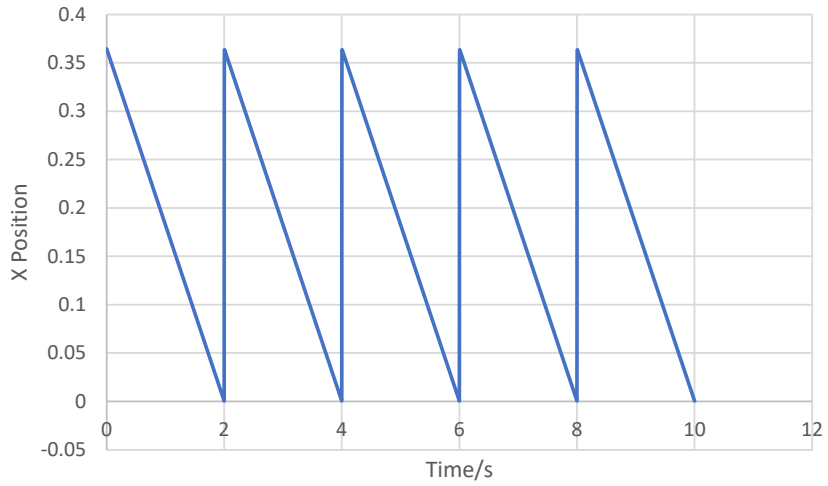


Fig.4.15 X position of end effector inputted by user

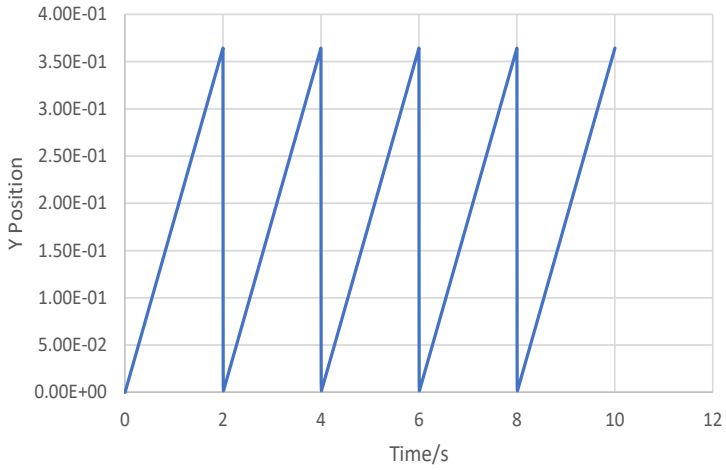


Fig.4.16 Y position of end effector inputted by user

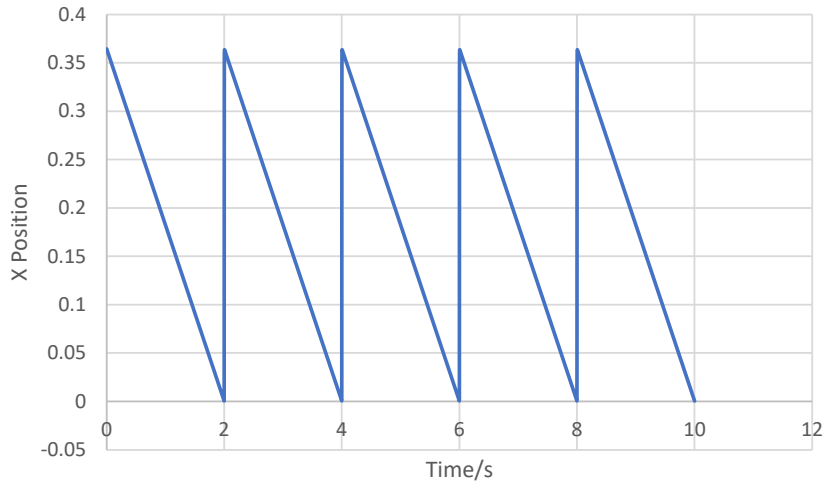


Fig.4.17 X position of end effector measured

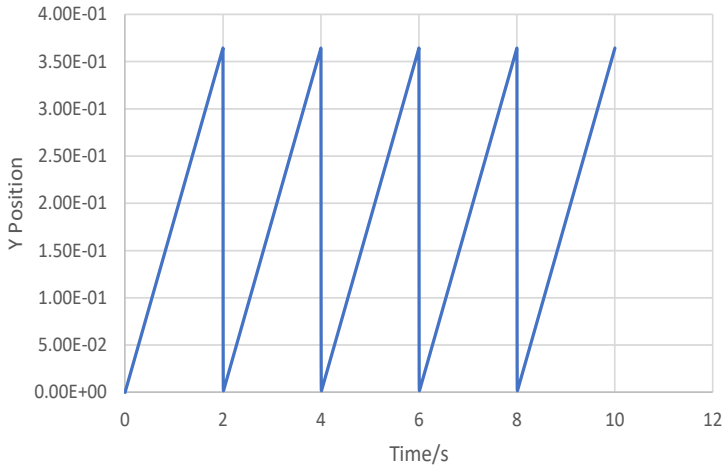


Fig.4.18 Y position of end effector measured

It can be clearly seen that the robot exactly tracks the point inputted by the user using the inverse kinematic equation.

4.4 Practical Results

4.4.1 Experiment 1 Results

In the first experiment where the robot was to move in a kind of oscillatory motion, the robot moved to a point and exactly returned to the previous point and this proved that there is no slippage in the belt pulley system and the robot was moving correctly.

4.4.2 Experiment 2 Results

In the second experiment the robot drew the square shape but there was an error while drawing. The square shaped was slightly off as the robot is not that rigid, when the arms are extended the weight of the arms themself make the robot lean a little forward. This causes the actual dimension or position of the arms to be a little bit off. Also, in this experiment a 50% laser power was used in combination with 400mm/min feed rate was used.

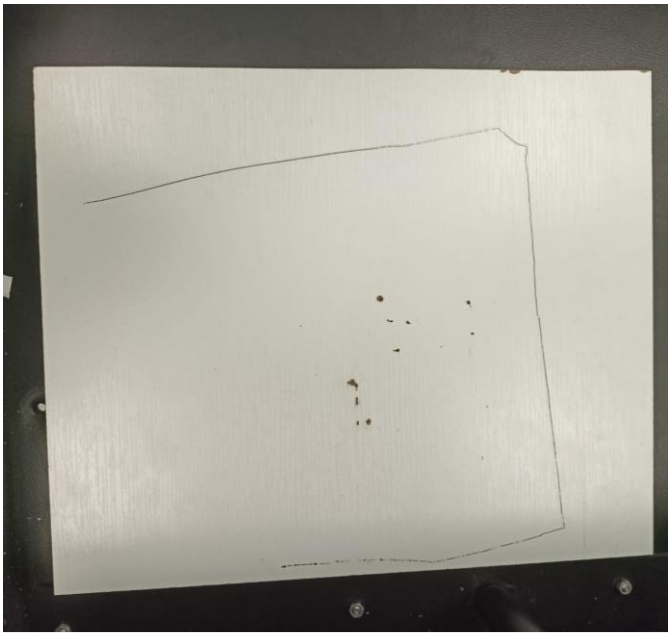


Fig.4.19 Square shape contour engraved

4.4.3 Experiment 3 Results

In the third experiment, after readjusting some parameters in the firmware I printed the word “GUC 20 YEARS” and the result was almost perfect nevertheless satisfactory and this showed

that the robot is very good at repeatability however the engraving was still a little tilted. In this experiment I used a 100% laser power in combination with 1000mm/min feed rate. Similar engraving results has been outcome using different combination of laser power and feed rate or movement speed of arms.

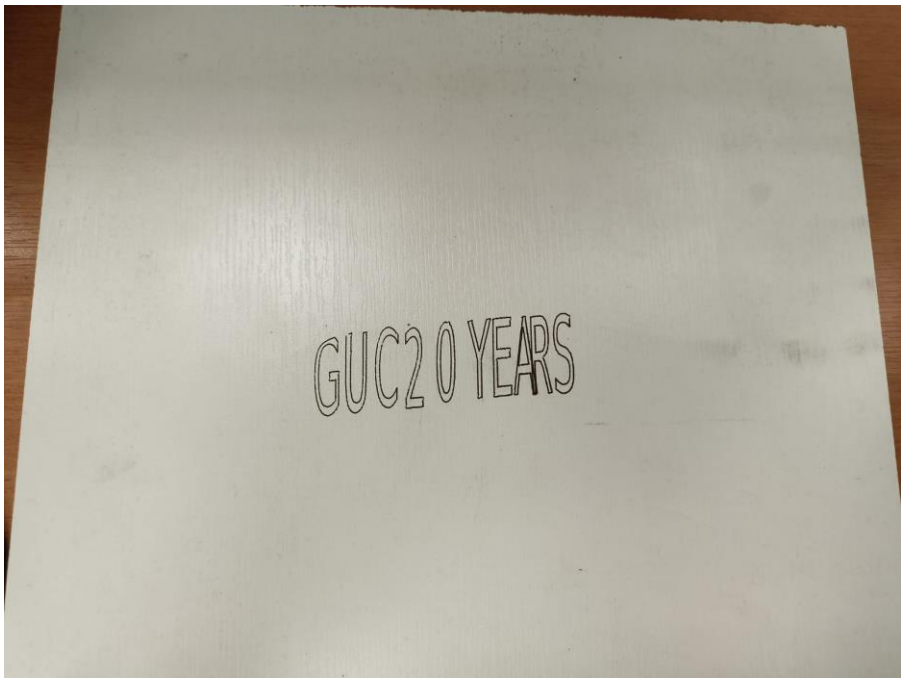


Fig.4.20 “GUC 20 YEARS” contour engraved

4.4.4 Experiment 4 Results

In the fourth experiment, I engraved the ARAtronics logo after readjusting again some parameter and the overall logo was correct yet the end result was unsatisfactory. In this experiment I used a 100% laser power in combination with 1000mm/min feed rate.

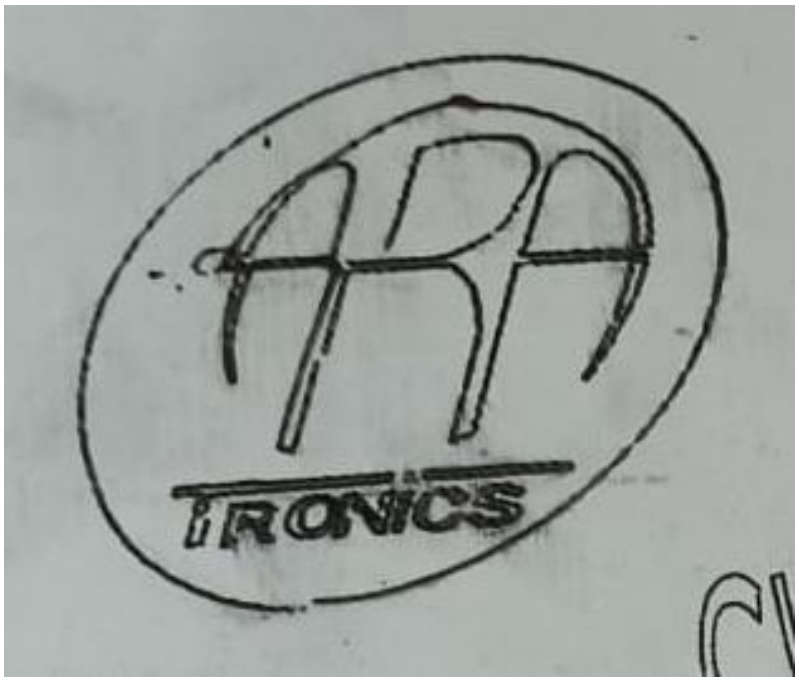


Fig.4.21 ARAtronics logo contour engraved

4.4.5 Experiment 5 Results

In the fifth experiment, I engraved a bat shape after readjusting again some parameters and the overall shape was correct and after various trials the bat shape turned out perfect and almost perfectly oriented. In this experiment I used a 50% laser power in combination with 200mm/min feed rate. This reduced rate of movement will prevent high jerk in the robot's movement and thus better end result.

Fig.4.22 Bat shape contour engraved

4.4.6 Experiment 6 Results

In the sixth experiment, I engraved a bat shape but this time with an infill and after readjusting again some parameters and the overall shape was correct and after the first try the bat shape turned out good but not aligned with the contour. After trying for a second time the shape came out perfect. In this experiment I used a 50% laser power in combination with 200mm/min feed rate.

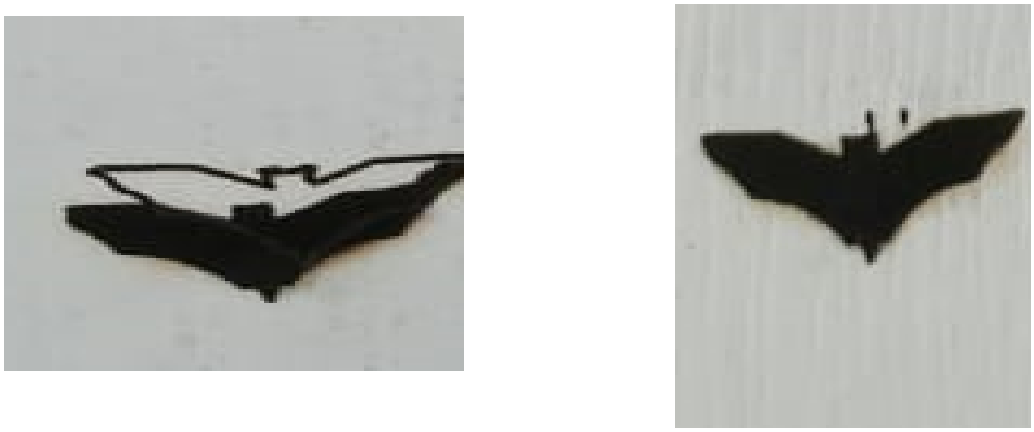


Fig.4.23 Bat shape infill engraved

4.4.7 Experiment 7 Results

In the seventh experiment, I engraved another word “ROBOTS” and compared it to the first word I engraved. This is a much obvious improvement over the first engraved word. In this experiment I used a 50% laser power in combination with 200mm/min feed rate.

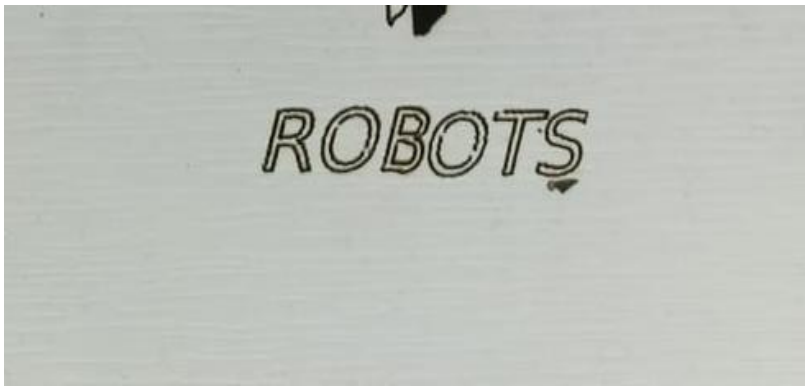


Fig.4.24 “ROBOTS” word engraved

Chapter 5 Experimental Work

5.1 Design of robot

The complete design for the Robot already existed in the ARAtronics Lab and I just adjusted some parts on the design to fit with the motors and actuators in the local market.

To start off I just engraved the name of the robot “SCARAtronics 2.0” on the side of the robot’s arm 1.

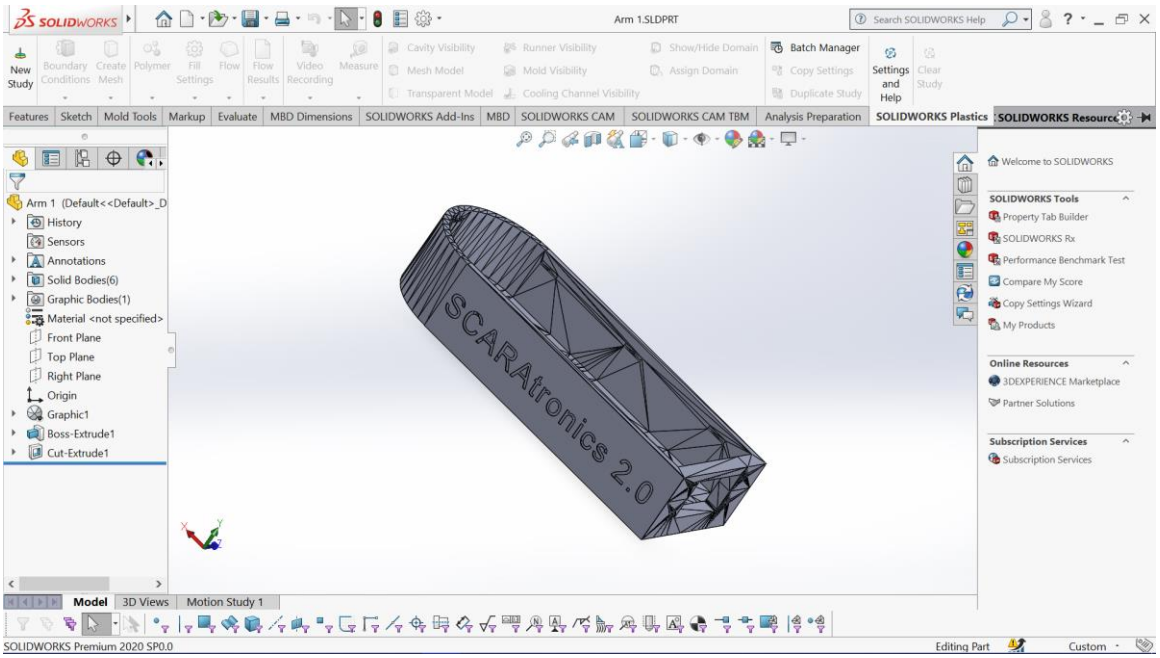


Fig.5.1 Arm 1 of robot

Next, I redesigned the pulleys to fit the belts available in the market. The following equation (46) was used to calculate the correct diameter of the pulleys:

$$L = \pi(r_1 + r_2) + 2x + \frac{(r_1 + r_1)^2}{x} \quad (46)$$

Where r_1 and r_2 are the radii of the larger and smaller pulleys respectively

x is the distance between the center of two pulleys

L is the total length of the belt

α is the angle of lap which is taken here as π

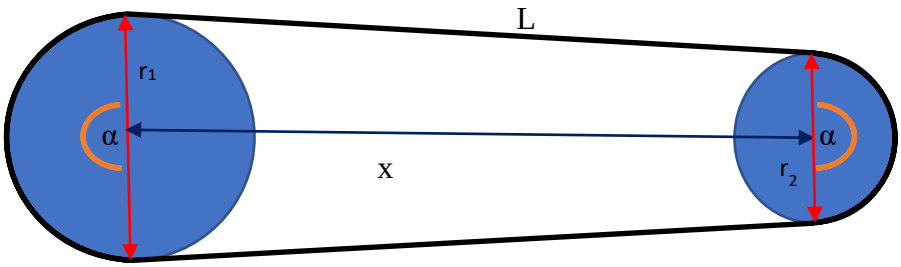


Fig.5.2 Illustration of simple pulley belt system

Moreover, when calculating the pulleys diameter there were some constraints being the length, width and depth of the part where the pulley is to be fitted in and this placed a restriction to how large the pulleys diameter can be. Nevertheless, in such cases I made the pulleys as large as possible and I designed idler pulleys to be placed between the pulleys to make sure the belt is tight.

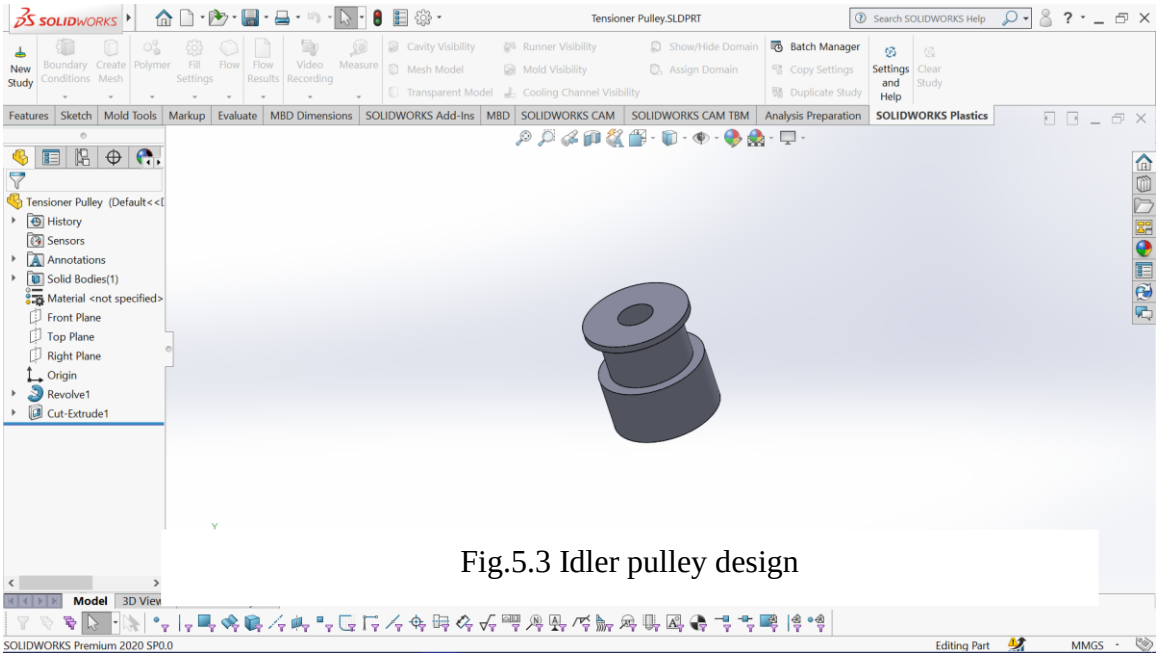


Fig.5.3 Idler pulley design

For the pulleys that will be attached to the motors I made sure the pulley hole has very low tolerance and a slot to ensure very tight fit with motor shaft. The number of teeth in a pulley was calculated using the following equation (47):

$$Z = \frac{Dp*\pi}{P} \quad (47)$$

Where Z is the number of teeth

Dp is the pitch diameter

P is the pitch circle diameter

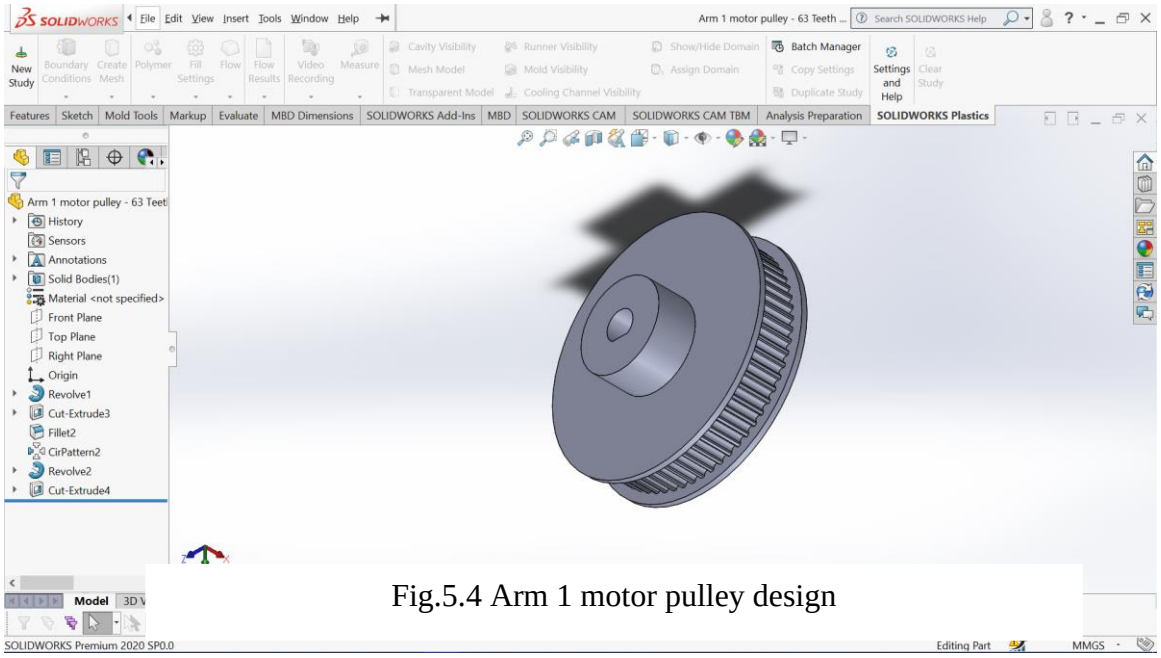


Fig.5.4 Arm 1 motor pulley design

The robot's workspace will be between a large circle and a key shaped outline and has a height of 40 cm. The robot's workspace can be seen in Fig.5.5.

Fig.5.5 Robots workspace

5.2 Assembly of robot

After all the redesigning was done I 3D printed the parts. I used the Fused Filament Fusion (FFF) 3D Printing technique on high precision European made 3D printers. The material I used was High quality black PLA filament made in the Netherlands. Resolution/Layer height of 100 - 150 microns and shell thickness of 0.7 - 1.0 mm and an infill of 20 to 35% which gave the parts strength and durability and also made sure the parts are as lightweight as possible for the robot to move smoothly. The printing process took around a week or 8 days to print as there were many parts to print and there were some large parts that required a quite large amount of time.

After the 3D printing was done and I collected the required components such as bearings, lead screws and rod shaft, I started assembling the robot. I started the with base and I inserted a radial ball bearing with 35mm inner and 47mm outer diameter which is used to ensure smooth rotation. Then thrust bearing which has 40mm inner and 60mm outer diameter was placed between the pulley and base. On the other side of the base, I used another thrust bearing of the same size together with joint coupler then I coupled the pulley and upper part using some bolts and nuts. Next, I used two 608 ball bearings, one on the top and the other at bottom side of the base and a M8 bolt and nut to install the middle pulley which connects motor to joint. This middle pulley is paired with the joint pulley with a 320mm GT2 belt. Then I checked how tight the belt was and accordingly I placed the idler pulley in the correct position to make sure it is very tight.

Fig.5.6 Base with joint pulley installed

Next, using some bolts I installed the stepper motor with base motor pulley pressure fitted with it. The stepper motor used was the 2-phase hybrid stepper motor with 42 by 42 cross section. It had the following specs: 17HS4401 series model with 1.8 degree step angle, 40 mm motor length, 1.7 A rated current, 1.5 ohm phase resistance, 2.8 mH phase inductance, 40 N.cm minimum holding torque, 2.2 N.cm maximum detent torque, 54 g.cm^2 rotor inertia, 4 lead wires or pins, 280 g motor weight. The stepper was paired with the middle pulley with a 200mm belt.

Then I installed the limit switch for the first joint. Before securing it in place, I already soldered the wires to the normally closed pin and common pin for easier installation. After soldering any wires, a heat shrink wrap was used to cover up the soldered part to give a cleaner look and to make sure the wires do not touch each other causing a malfunction. In the case of joint one, I ended up using only one bolt to secure the limit switch as the joint coupler was restricted by the second bolt.

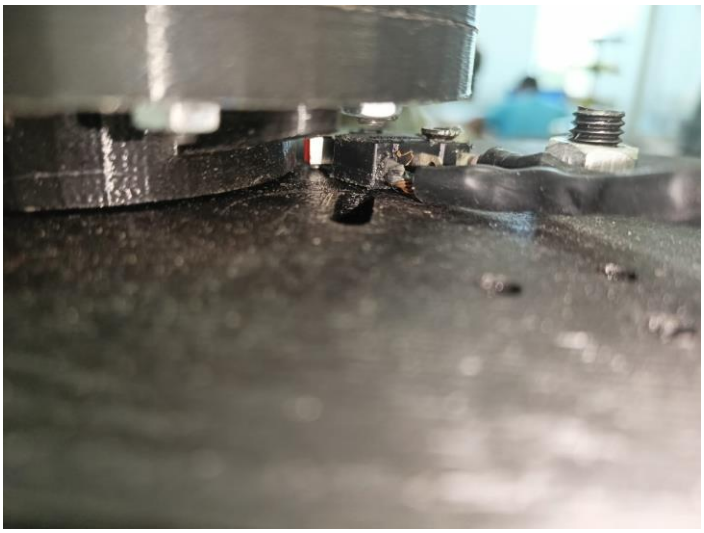


Fig.5.7 Limit switch of base

Then I started putting the Z-axis together. I bolted the four clamps for the smooth rods on top of the Z-axis bottom plate component, which was fixed on top of the joint coupler. The smooth rods were then placed into position. The rod clamps were then tightened using bolts to ensure that the rods were held tightly in position. The lead screw bearing was fitted, and then a basic cover was added to give the robot a cleaner appearance.

Fig.5.8 Base with Z platform installed

After that, I began putting together the robot's first arm. The initial arm was constructed from two bolted-together components. Installing the linear bearings that will move through the smooth rods is the first step. When inserting the linear bearings, however, the holes were slightly larger than the linear bearings, causing the linear bearings to fall out. To address this issue, the linear bearings were wrapped with layers of double face tape, then installed and carefully secured in place. Then I used bolts and nuts to join the two pieces of the arm together and secured the lead screw nut in place.

Fig.5.9 Arm one with joint pulley installed

I inserted a radial ball bearing with 30mm inner and 42mm outer diameter which is used to ensure smooth rotation. Then thrust bearing which has 35mm inner and 52mm outer diameter was placed between the pulley and arm one. On the other side of the arm, I used another thrust bearing of the same size together with joint coupler then I coupled the pulley and lower part using some

bolts and nuts. Before coupling the second joint, I placed bolts in the coupler joint which served for attaching the second arm to the joint. Next, I used two 608 ball bearings, one on the top and the other at bottom side of the base and a M8 bolt and nut to install the middle pulley which connects motor to joint. A belt with 320mm length was used to pair the middle pulley with joint and then I checked how tight the belt was and accordingly I placed the idler pulley in the correct position to make sure it is very tight. Next, using some bolts I installed a stepper motor, similar to that of arm one, with arm two motor pulley pressure fitted with it and was paired with middle pulley with a 460mm and this was also checked if it was tight or not and accordingly added idler puller. Then to finish arm one I secured the second limit switch in place.

Fig.5.10 Arm one limit switch with Arm two attached

After that, I started assembling the robot's second arm. Using the bolts that had previously been inserted in arm one, I attached the top half of the second arm to the first arm joint coupler. Then I moved on to the bottom half of arm number two. I started with the third joint, expecting to use a smaller stepper motor than the others to make the robot as light as possible, but it was not available in the local market. Then I proceeded to assemble the belts, third joint pulley, and limit switch, following the same process as the previous two joints and using a 460 mm belt. The limit switch, motor wires, and laser module cables were then fed through the second joint before joining the bottom half of the arm to the higher part. I also attached the laser module driver in the lower part of arm two using some zip ties which would allow easy access to connect the laser module to it. After that, I fitted some nuts into the higher arm's slots, which will be used to connect the lower section to it, and I fastened the bottom part to the top part. The cables were then routed down one side of the arm to prevent contact with the moving parts. Finally, I used a snap-fit joint cover to cover the first arm. The I simply insert this whole assembly to the Z-axis rods using the linear bearings in the first arm.

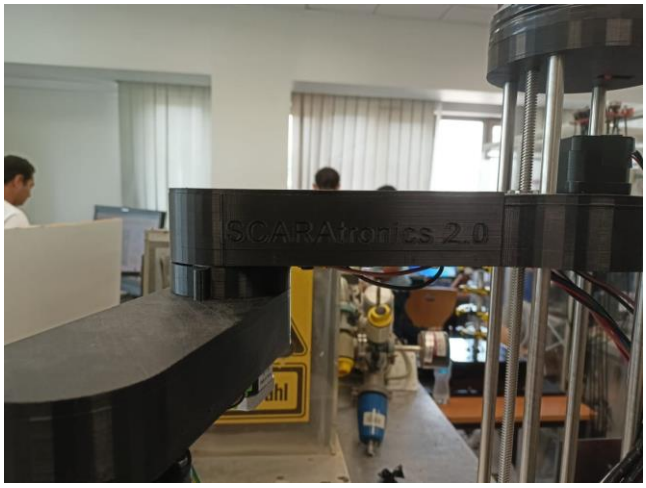


Fig.5.11 Both arms assembled in Z-axis rods

Then I made the top plate for the Z-axis, which would retain the rods' higher ends. I started by attaching the for clamps to the plate and installing the Z-axis limit switch. Before installing the top plate, I added a basic cover, similar to the one used on joint one, to conceal the clamps, bolts, and limit switch. The top plate was then installed and fastened to the rods using the clamps and bolts. After that, I put in an 8 mm lead screw. The fourth stepper motor was then secured in place using a 5mm to 8mm shaft coupler to link the motor to the lead screw. Finally, I ran the motor and limit switch wires through the cover and secured the covered with two bolts to the top plate.

Fig.5.12 Z-axis top plate with limit switch and clamps

Then I simply screwed the laser module into its holder, fed the laser wires through a hole in the holder, and linked the holder to the robot's third joint using bolts and nuts. The laser module was then linked to its driver.

Fig.5.13 Laser module and its holder

The SCARA robot was now finished, and all that was left was to attach the base to anything. I utilized a metal platform I discovered in the lab for this. There are 12 holes on the bottom side of the robot's base for anchoring it to the platform. As a result, I produced a design of the robot base with holes noted on it and used it to drill holes in the metal platform. I placed nuts into the base slots before fastening the robot's base, and then fastened the metal platform to the robot's base using some bolts.

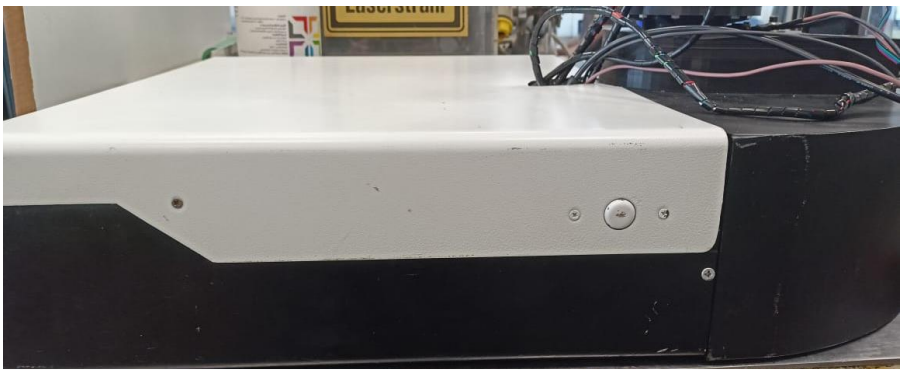


Fig.5.14 Metal platform attached to the base of robot

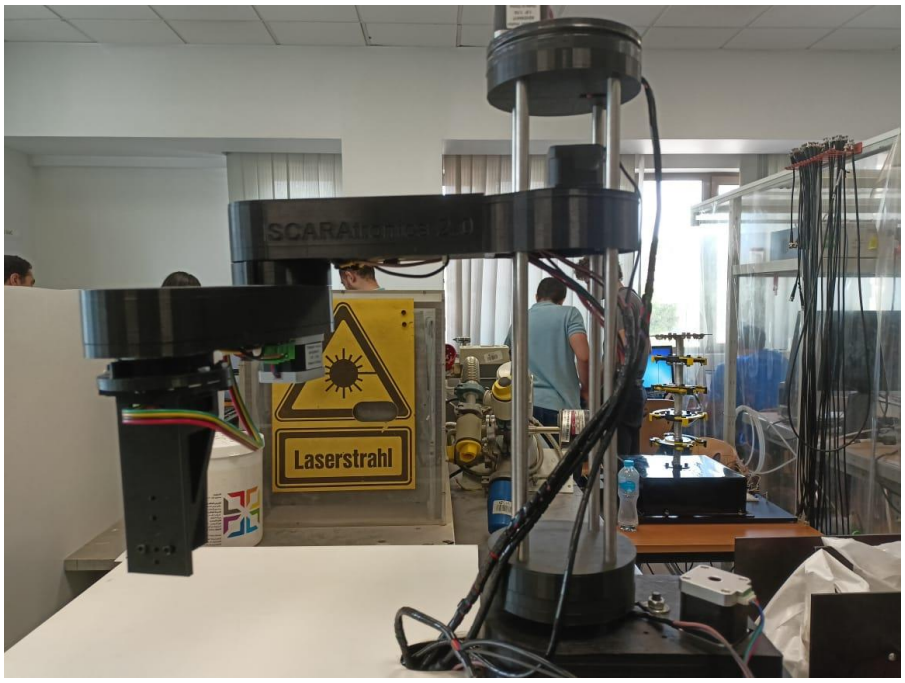


Fig.5.15 Finished robot

5.3 Circuitry of robot

I needed to connect the cables to the controller, which in this case was an Arduino mega with a ramps 1.6 attached to it, once I finished the robot. The connections should be clearly displayed in Fig.5.15. For smoother motion, I added three jumpers to each A4988 stepper driver holder in the ramps 1.6 to pick the maximum feasible resolution. The A4988 stepper driver can withstand up to 2A if given proper heat ventilation and in this case, I used the heat sink it came with the stepper driver. I utilized pin D9, which is normally used to control a fan in a 3D printer configuration, to control the laser module. Because this pin is PWM capable, I was able to change the laser's intensity with it, as I indicated before in the control section. I required a 12V power source with a minimum current rate of 8.6A to power everything, but I ended up utilizing a 12V power supply with a current rate of 12A. Once everything was connected, I placed the controller in the Arduino mega case and covered the electronics with the case cover to keep things tidy. To make things

more organized and simpler to analyze later, spiral cable wrap was employed to hold each stepper motor wire and its matching limit switch wires together.

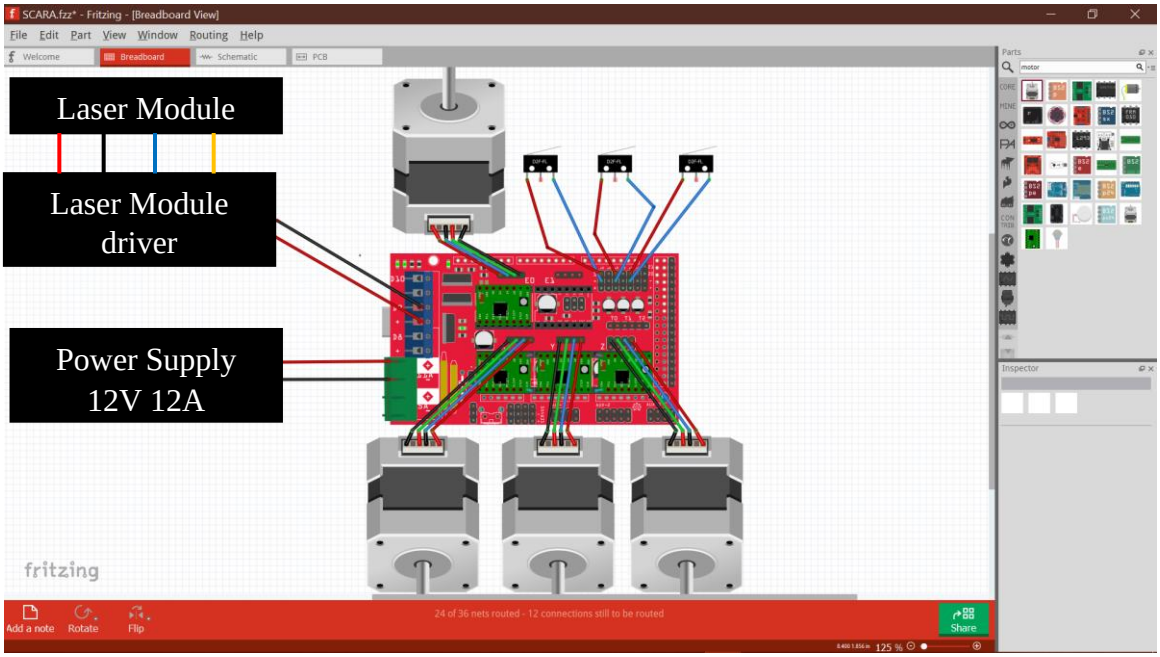


Fig.5.16 Circuitry sketch of robot

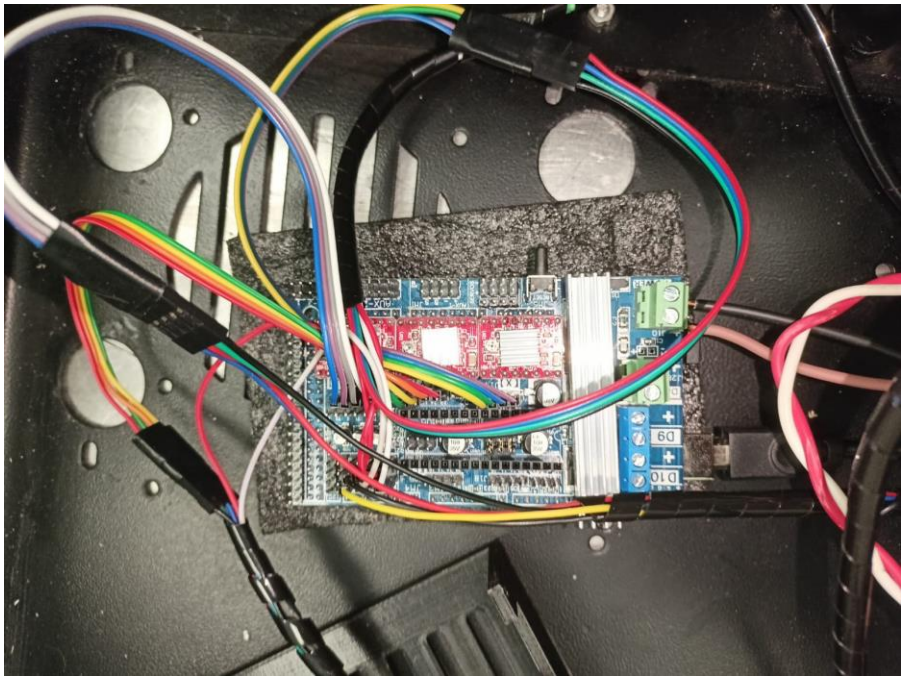


Fig.5.17 Circuitry of robot

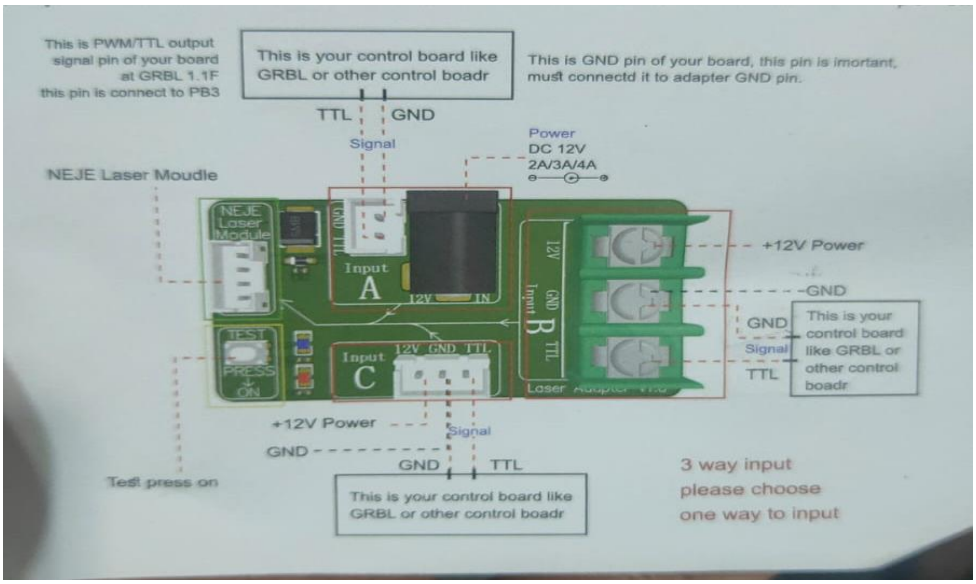


Fig.5.18 Laser module driver pinout

5.4 Motors and Limit Switch and Laser Module

Before assembling the robot, I examined each stepper motor to know which stepper wire belongs to which coil. For that purpose, I created a simple Arduino code and setup a stepper motor driver on a breadboard and connected the driver pins to Arduino. Start by connecting two random motor wires to any two coil pins. If the interconnected wires belong to one coil, they will "create" a magnetic field in the motor when trying to manually rotate the shaft. This will be felt by the increased resistance to turning the shaft, or a slight bouncing sensation. However, if the shaft turns slightly and a difference is not felt between when the wires are connected and when they are not, the pair of wires selected do not belong to the same coil. Then after checking for the pair of coils, I marked each wire with its respective coil pin. For a double checking I tested the motor to turn clockwise and anticlockwise after marking each wire.

After everything was connected, circuitry and hardware, I had to test the motors and adjusted each stepper driver current accordingly to its stepper motor and to test the limit switches. Adjusting the stepper drivers current was straightforward, I simply made a simple Arduino code where I setup all pins and then I uploaded the code to controller and then connected the power. To adjust the stepper driver current, I used the following equation (48)

$$V_{ref} = I_{mot} * 8 * R_{sen} \quad (48)$$

Where V_{ref} is the reference voltage.

$I_{mot}$is the maximum motor current.

R_{sen} is the current sensing resistance.

The current sensing resistance for all the drivers I used was 0.1 ohms and can be found on the stepper driver. Only the X-axis or the bas motor had a maximum motor current of 1.5A while the rest had 1.7A. So, for the base motor V_{ref} was 1.2V while the others were 1.36V. The V_{ref} is measured by putting the positive multimeter lead on the metal part of a small screwdriver on the stepper driver and the negative multimeter lead to any ground pin. Turning the screwdriver anticlockwise decreases the voltage and clockwise increases the voltage.

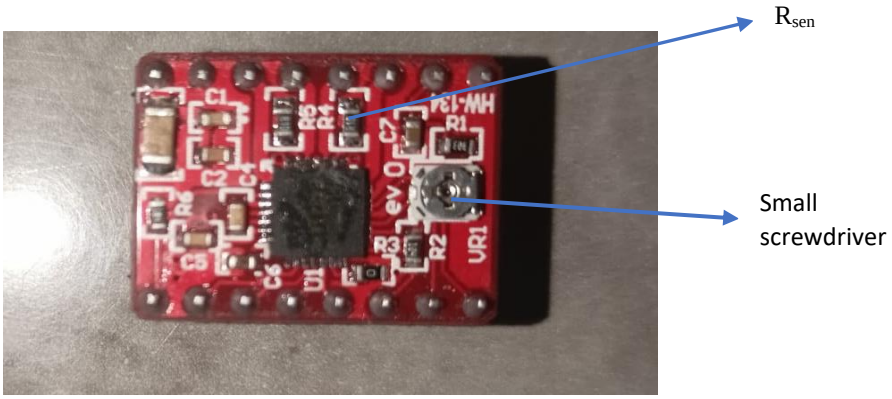


Fig.5.19 A4988 Stepper Driver

Then after adjusting the stepper driver current limit, I wanted to test the stepper motors to make sure the stepper motors have enough torque to move the joints. For that purpose, I made a simple code where each motor was made to step in the positive direction then after a while step in the negative direction. Using this I noticed some motors struggling to move or skipping steps and this was solved by changing the step delay between each step high and step low of the stepper motor. Increasing the delay gives the stepper motor high torque but low speed and the opposite is true. Brushless motors are used in stepper motors. When the rotor leads or lags the desired position by one complete step, the motor can deliver the maximum torque at any given speed. If the difference

between the commanded and rotor positions is more than two complete steps, the motor will stall. With this I made sure the stepper drivers and stepper motors were working perfectly.

Then I moved on testing the limit switch which was very easy. I simply made an Arduino code when switch is pressed gives signal to computer. I manually pressed each switch and checked the signal and turns out every switch was working properly.

Another very important step was adjusting the laser module to be focused and sharp. This was done by uploading the Marlin firmware and doing the adjustments as mentioned in the control section 3.1 and using the Repetier-Host program homed the robot. Then moved the robot in the center of work area and in the Z-axis direction closer to a work piece and leaving a distance of approximately 5cm. Then the laser was activated at a very low intensity such as 5% laser power by adjusting PWM value to S12.8 and manually the focus was adjusted by rotating the laser lens at the bottom of the laser and adjusting it until I got the smallest dot possible.

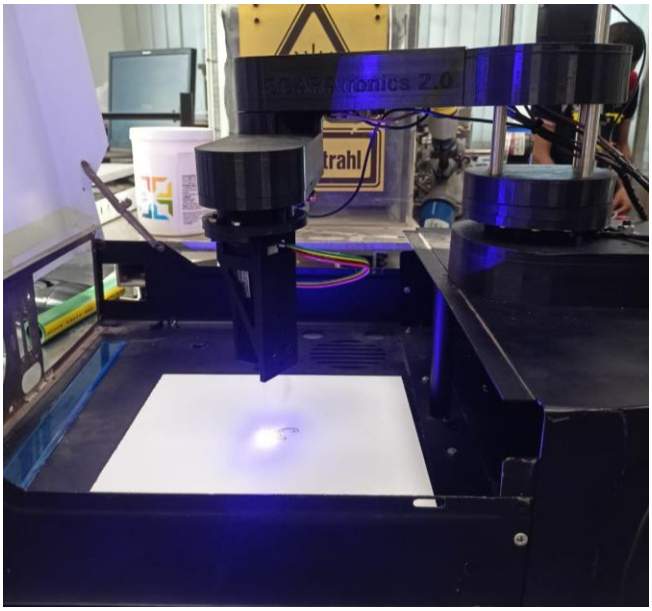


Fig.5.20 Adjusting the laser

5.5 Practical experiments

When homing I noticed that the Z-axis limit was not being pressed so I designed a part and placed on the Z-axis stepper motor so now when the Z-axis is homing the part will press the Z-axis limit switch. Also, when homing the robot arms needed to be collinear to each other for best results and for that I designed another part and placed it on the joint coupler of the second arm such that when the limit switch of joint 2 is pressed both arms are collinear.

5.5.1 Experiment 1

In the first experiment, I sent an exact number of steps for each stepper motor in the positive direction and in the negative direction. It was an oscillatory motion.

5.5.2 Experiment 2

In the second experiment, I wanted the robot to draw a contour of a square shape.

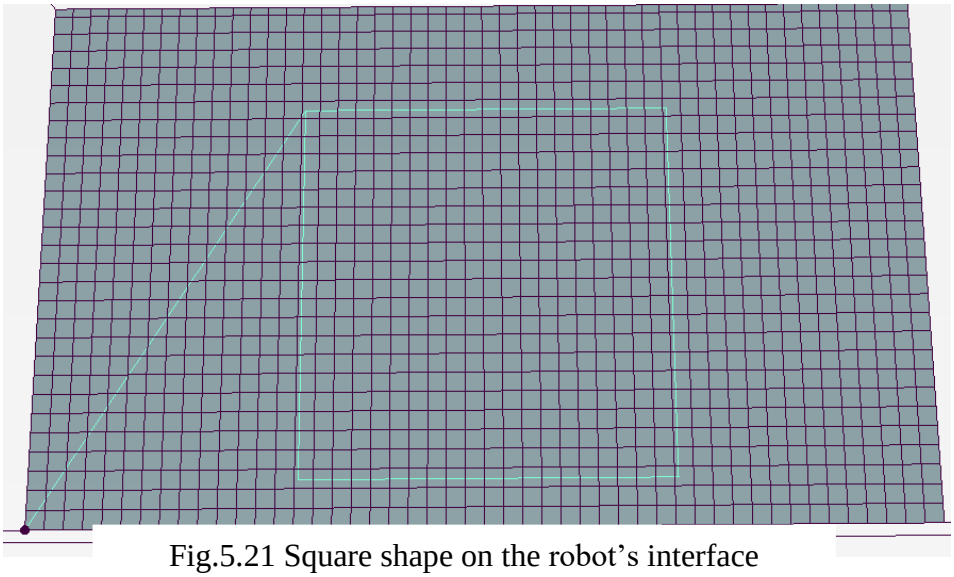


Fig.5.21 Square shape on the robot's interface

5.5.3 Experiment 3

In the third experiment, I wanted the robot to draw the word “GUC 20 YEARS” to see how accurate the robot was.

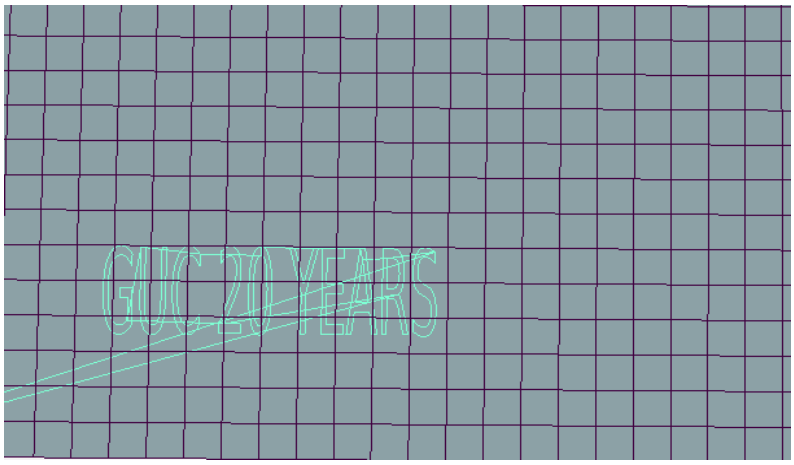


Fig.5.22 Contour of the word “GUC 20 YEARS” on the robot’s interface

5.5.4 Experiment 4

In the fourth experiment, I wanted the robot to draw a picture which was the ARAtronics logo to see how accurate the robot was at drawing pictures.

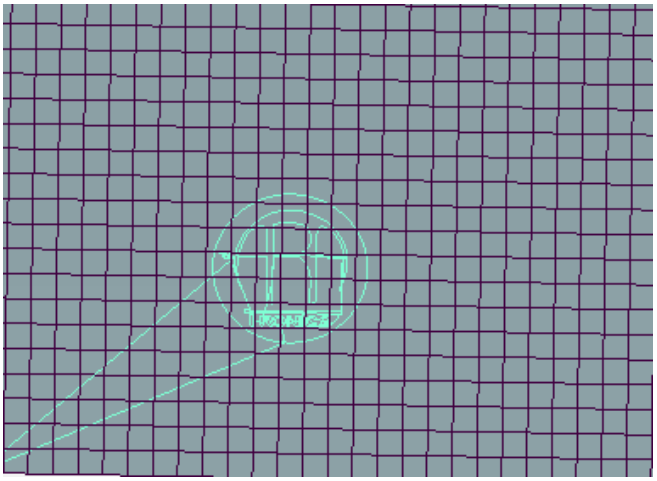


Fig.5.23 Contour of the ARAtronics logo on the robot’s interface

5.5.5 Experiment 5

In the fifth experiment, I wanted the robot to draw a small detailed shape which was the bat shape to see how accurate the robot was at drawing small shapes.

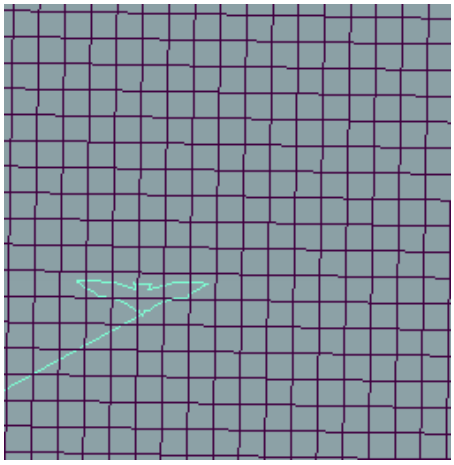


Fig.5.24 Contour of the Bat shape on the robot's interface

5.5.6 Experiment 6

In the sixth experiment, I wanted the robot to draw the same small detailed shape which was the bat shape but this time with an infill to see how accurate the robot was at drawing infill shapes.

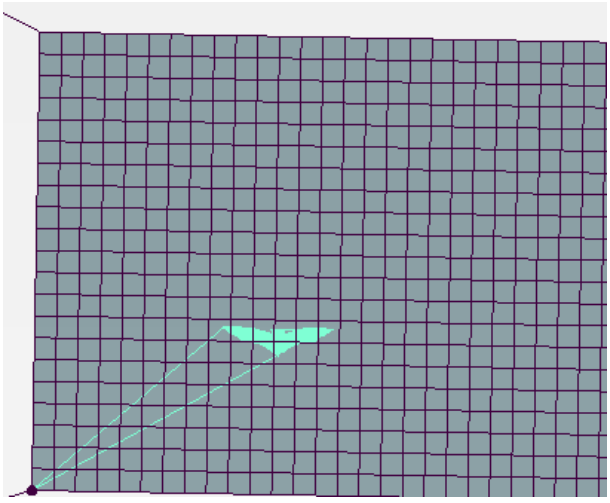


Fig.5.25 Infill of the Bat shape on the robot's interface

5.5.7 Experiment 7

In the seventh experiment, I wanted the robot to engrave another word which was “ROBOTS” to compare it to the first word I engraved.

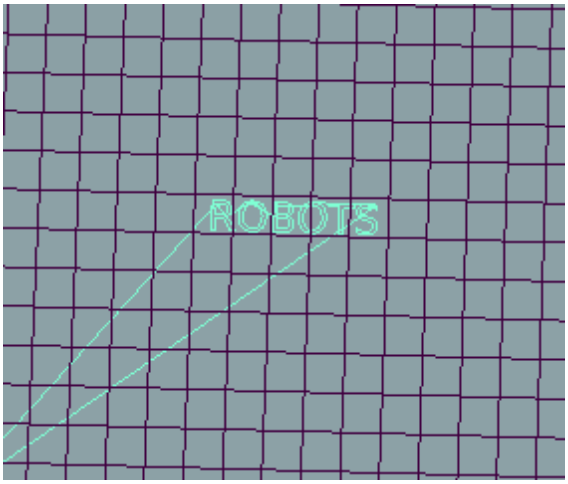


Fig.5.26 Contour of the word “ROBOTS” on the robot’s interface

Chapter 6 Conclusion and Recommendation

6.1 Conclusion

A complete four degrees of freedom SCARA was designed and built from scratch. This SCARA was used to laser engrave some shapes and writing on work pieces. Forward kinematics equations were derived by using Denavit-Hartenberg notation along with the inverse kinematics as well as equation of motion and dynamics of SCARA were derived. Simulation studies were performed using MATLAB and Simulink.

6.2 Recommendation for the future

This SCARA has the ability for its end effector to be changed and with this in mind I would recommend using soft grippers to handle delicate objects. A camera may also be utilized to capture the real-time surroundings, and the robotic arm can employ image processing to calculate the coordinates for picking and placing things instead of entering predetermined coordinates [24]. The robotic arm may also be controlled via other control modalities such as joysticks, mice, gestures, and so on. Another possibility for the laser engraving SCARA is that it could be utilized in the process of pcb production.

References

[1] R. K. Mittal, R. and I.J. Nagrath.; “Robotics & Control.” (22nd ed.); Tata McGraw Hill; (2013).

[2] K. Capek. ˇ R. U. R. Les Cahiers Dramatiques. Th´eˆatre et Comœdia Illustr´e, 1924.

[3] Lung-Wen Tsai; “ROBOT ANALYSIS: The Mechanics of Serial and Parallel Manipulators.”; John Wiley & Sons; (1999).

[4] Ben-Zion Sandler; “ROBOTICS: Designing the Mechanisms for Automated Machinery.” (2nd ed.); Academic Press; (1999).

[5] Herman Bruyninckx and Joris De Schutter; "Introduction to Intelligent Robotics."; Katholieke Universiteit Leuven; (2001).

[6] Hiroshi Makino; "Development of the SCARA."; Journal of Robotics and Mechatronics; 26(1); 5-8; (2014)

[7] Alba Perez and J. M. McCarthy; " GEOMETRIC DESIGN OF RRP, RPR AND PRR SERIAL CHAINS."; Mechanism and Machine Theory; 40(11); 1294-1311; (2005).

[8] Saravana Mohan Mariappan and Anbumalar Veerabathiran; "Modelling and simulation of multi spindle drilling redundant SCARA robot using SolidWorks and MATLAB/SimMechanics." Revista Facultad de Ingeniería. Universidad de Antioquia; 81; 63-72; (2016).

[9] Bernhard Sprenger, Ladislav Kucera, and Safer Mourad; "Balancing of an inverted pendulum with a SCARA robot."; IEEE/ASME transactions on mechatronics; 3(2); 91-97; (1998).

[10] Yesenia Aquilina, Michael A. Saliba; "An automated supermarket checkout system utilizing a SCARA robot: preliminary prototype development."; Procedia Manufacturing; 38; 1558-1565; (2019).

[11] Yunbo He, Xiquan Mai, Chengqiang Cui, Jian Gao, Zhijun Yang, Kai Zhang, Xun Chen, Yun Chen and Hui Tang; " Dynamic Modeling, Simulation, and Experimental Verification of a Wafer Handling SCARA Robot With Decoupling Servo Control."; IEEE Access; 7; 47143-47153; (2019).

[12] Khairul Salleh Mohamed Sahari, Khor Hong Weng, Yap Wee Han, Adzly Anuar, Mohd Zafri Baharuddin and Syed Sulaiman Kaja Mohideen; "DESIGN AND DEVELOPMENT OF A 4-DOF SCARA ROBOT FOR EDUCATIONAL PURPOSES."; Jurnal Teknologi; 54; 193-215; (2011).

[13] Fernando Passold and Marcelo Ricardo Stemmer; " Feedback Error Learning Neural Network Applied to a Scara Robot."; Fourth International Workshop on Robot Motion and Control (RoMoCo'04). IEEE; 197-202; (2004).

[14] Iosif Tempea, Adriana Livadaru and Alexandra Micu; "STATIC AND DYNAMIC ANALYSIS UNDER MECHANICAL AND THERMAL LOADS OF THE DOUBLE SCARA ROBOT."; Fiabilitate si Durabilitate - Fiability & Durability; 1; 10-16; (2016).

[15] Pranav Shevkar, Sayali Bankar, Akash Vanjare, Pranita Shinde, Vaibhav Redekar and Shubham Chidrewar; "A Desktop SCARA Robot using Stepper Motors."; International Research Journal of Engineering and Technology (IRJET); 6(2); (2019).

[16] Kenneth J. Waldron and James Schmiedeler; "Kinematics."; Springer Handbook of Robotics. Springer. Cham; 11-36; (2016).

[17] Amin A. Mohammed and M. Sunar; "Kinematics Modeling of a 4-DOF Robotic Arm."; 2015 International Conference on Control, Automation and Robotics. IEEE; 87-91; (2015).

[18] John J. Craig; "Introduction to robotics: mechanics and control" (3rd ed); Pearson Educacion; (2005).

[19] Serdar KüCük and Zafer Bingül; "Robot kinematics: Forward and inverse kinematics."; Industrial Robotics: Theory, Modelling and Control; 117-148; (2006).

[20] Serdar KüCük and Zafer Bingül; "The inverse kinematics solutions of industrial robot manipulators."; Proceedings of the IEEE International Conference on Mechatronics. ICM'04. IEEE; 274-279; (2004).

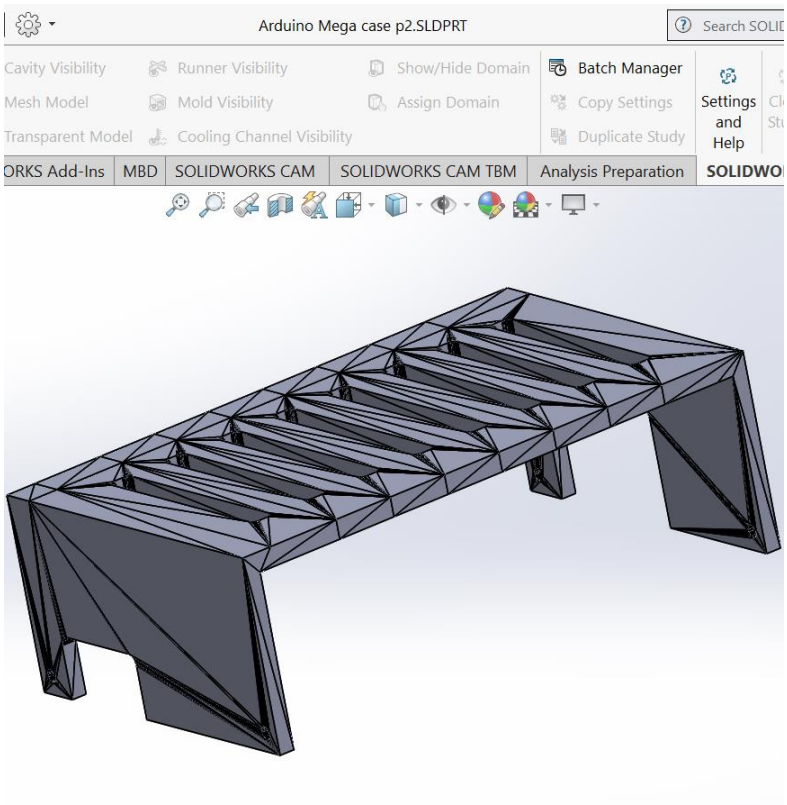
[21] Florent de Dinechin and Matei Istoan; “Hardware implementations of fixed-point Atan2.”; 22nd IEEE Symposium on Computer Arithmetic; (2015).

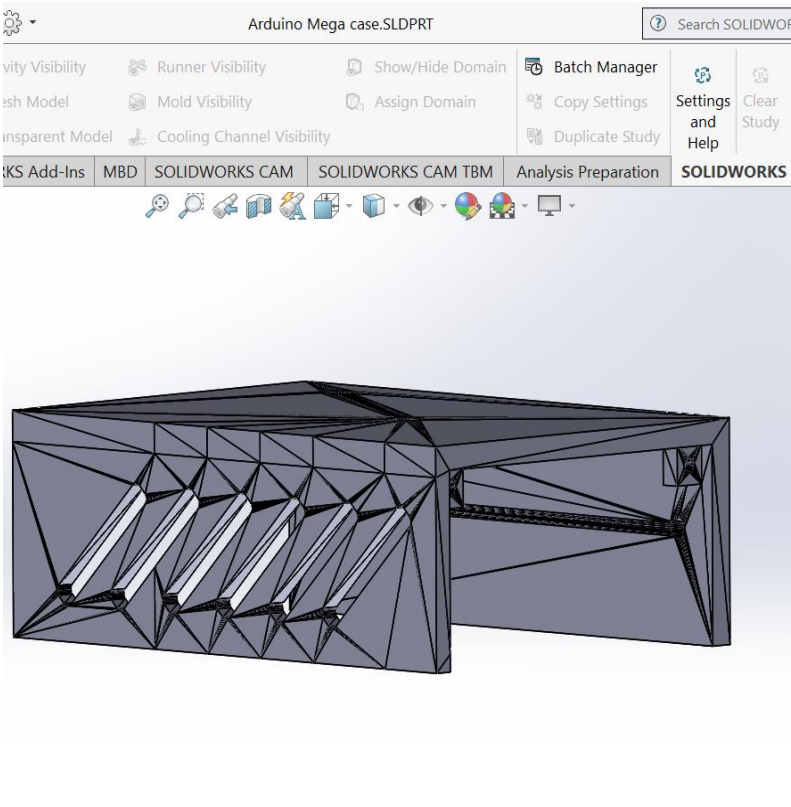
[22] David Morin; "The Lagrangian Method."; Introduction to Classical Mechanics, With Problems and Solutions; (2007).

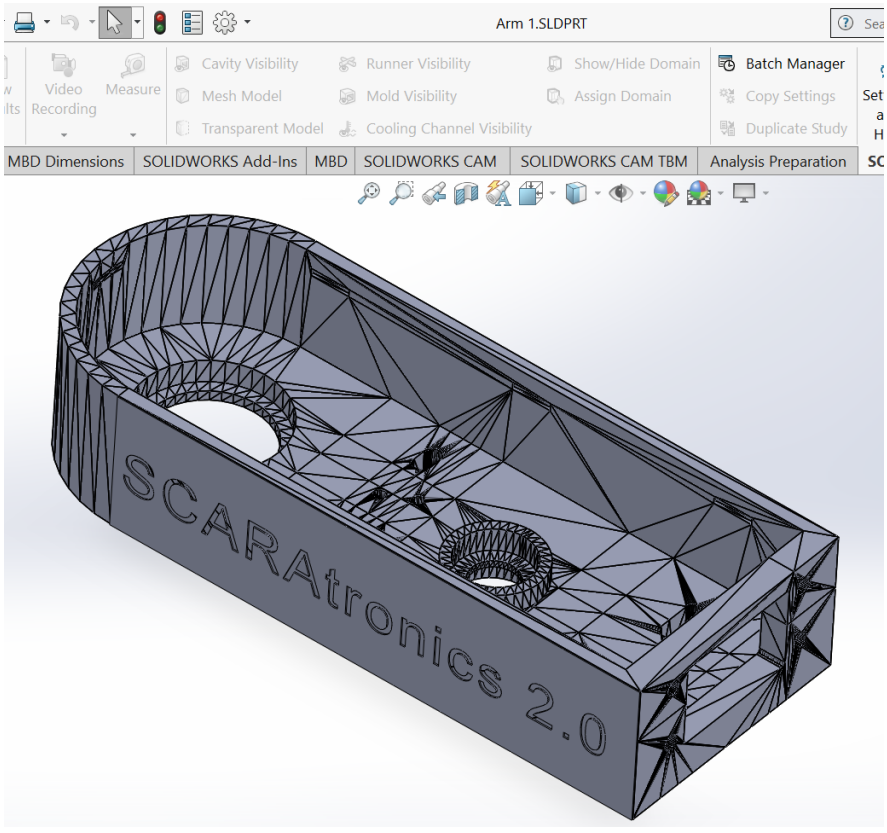
[23] Shinya Akamatsu, Masami Konishi, and Jun Imai; "Position Control of 2-Link SCARA Robot by using InternalModel Control."; Memoirs of the Faculty of Engineering, Okayama University; 43; 49-54; (2009).

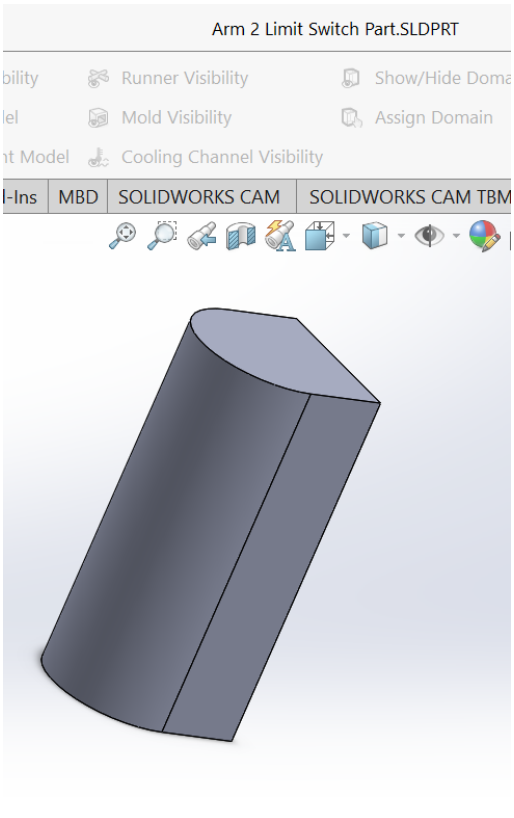
[24] Meng Joo Er, Moo Teng Lim, and Hui Song Lim; "Real-time hybrid adaptive fuzzy control of a SCARA robot."; Microprocessors and Microsystems; 25(8); 369-378; (2001).

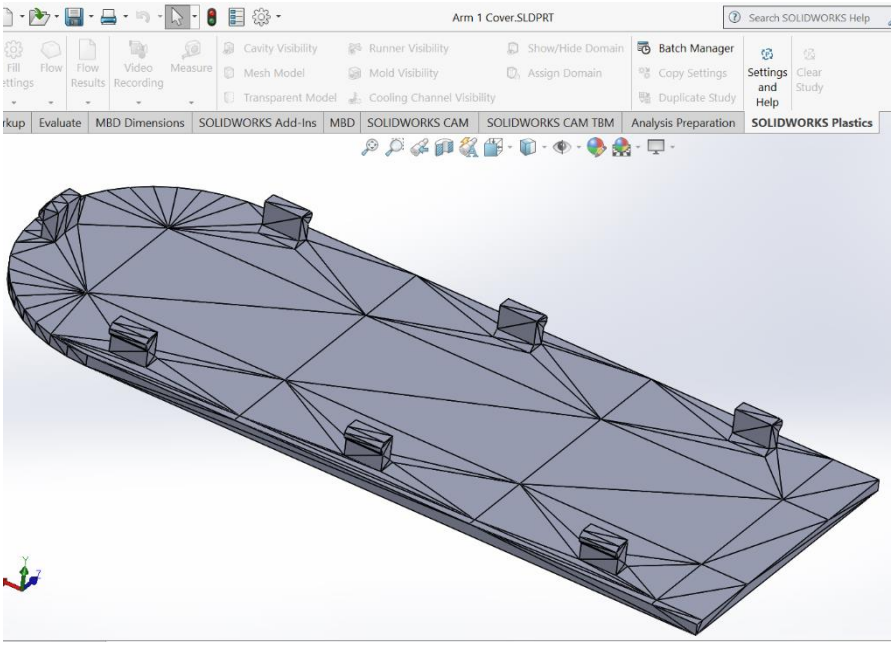
Appendix A: Solidworks

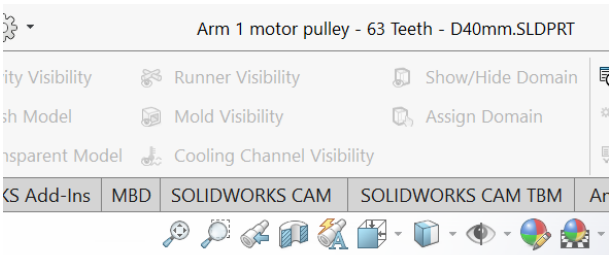


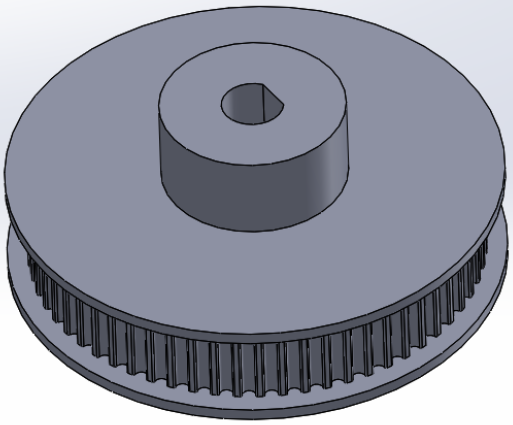


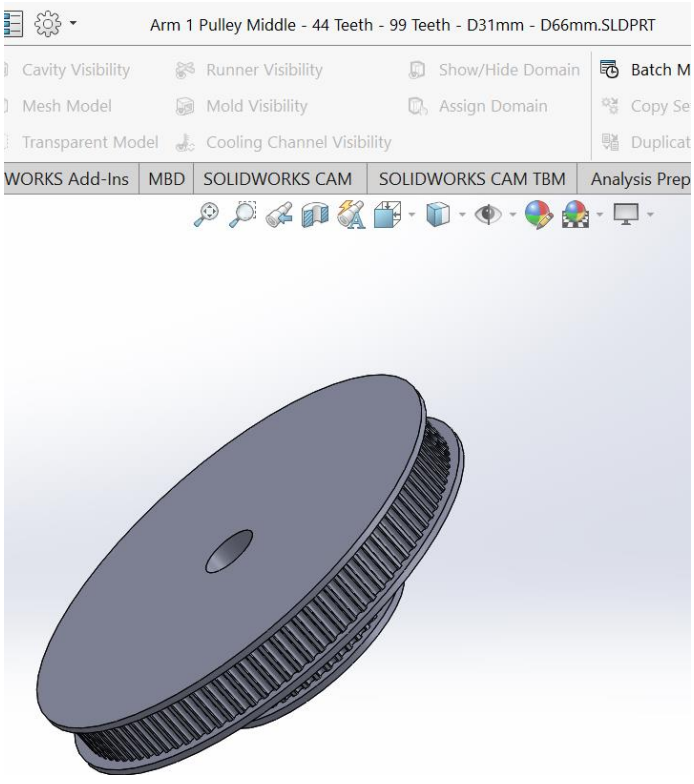

Arm 1.SLDPRT
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manager
Copy Settings
Duplicate Study
Video Recording
Measure
MBD Dimensions
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation
SCARAtronics 2.0

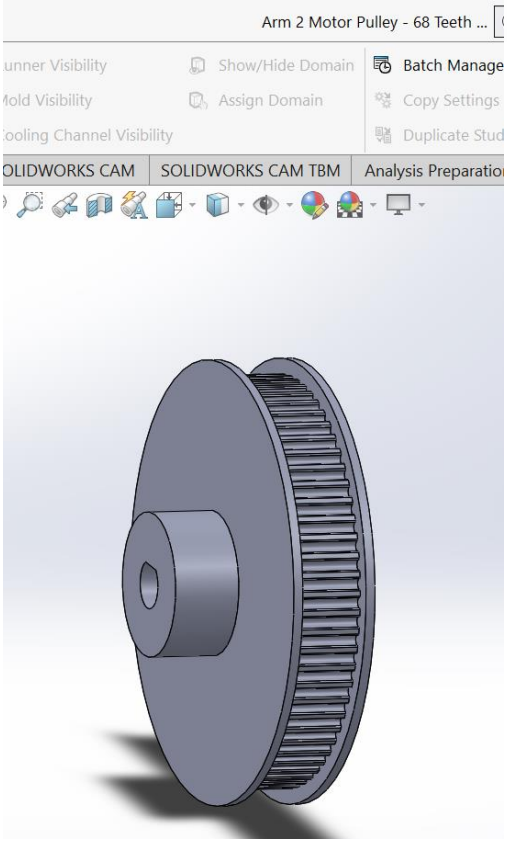

Arm 2 Limit Switch Part.SLDPRT
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Assign Domain
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

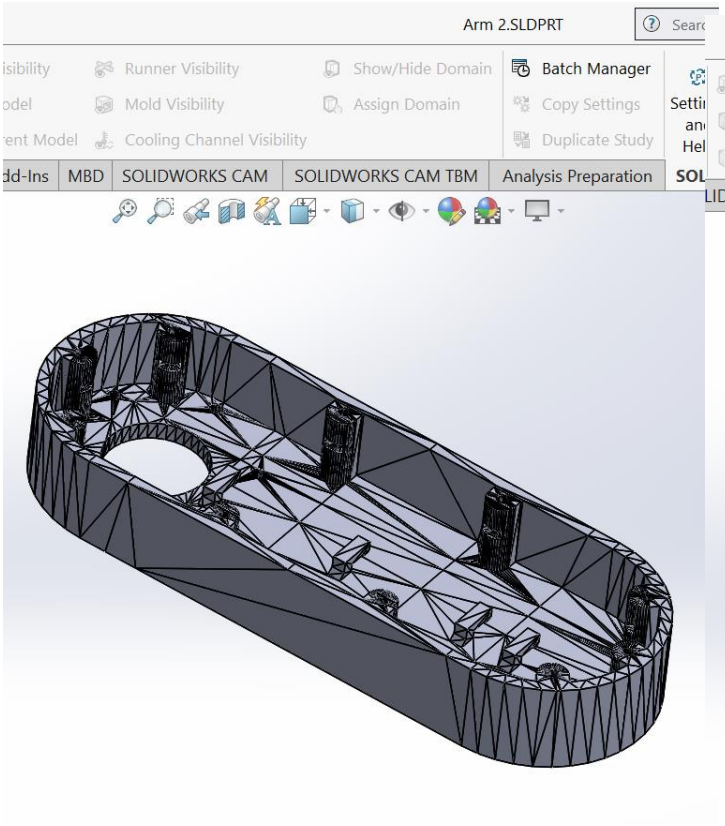

Arm 1 Cover.SLDPRT
Search SOLIDWORKS Help
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
Settings and Help
Clear Study
Evaluate
MBD Dimensions
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation
SOLIDWORKS Plastics

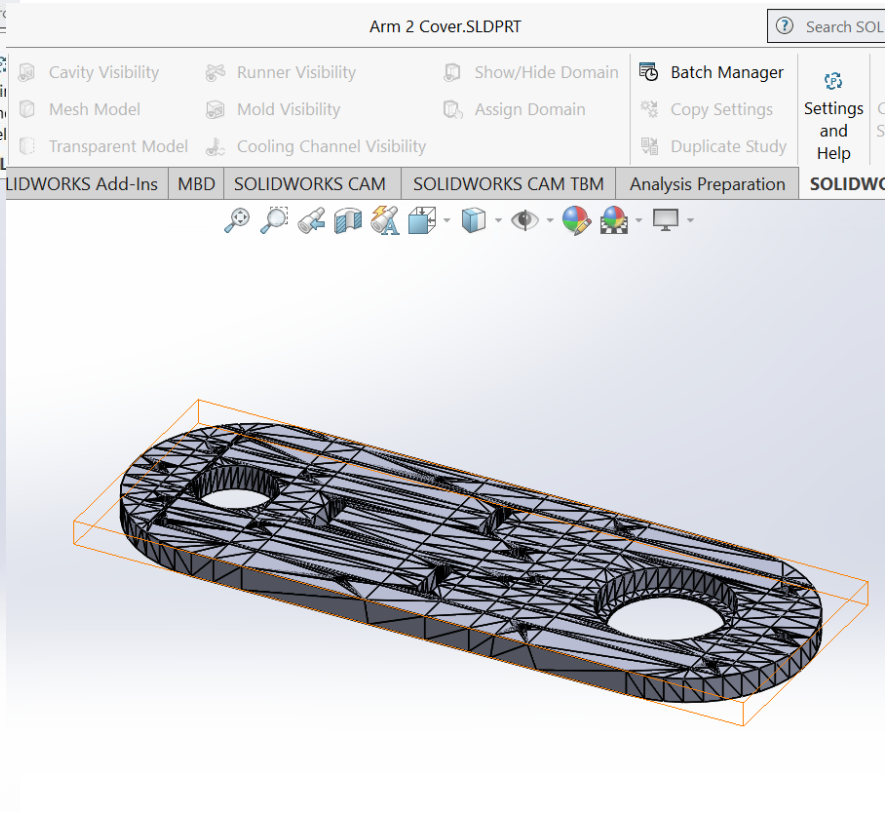

Arm 1 motor pulley - 63 Teeth - D40mm.SLDPRT
Runner Visibility
Show/Hide Domain
Mold Visibility
Assign Domain
Cooling Channel Visibility
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

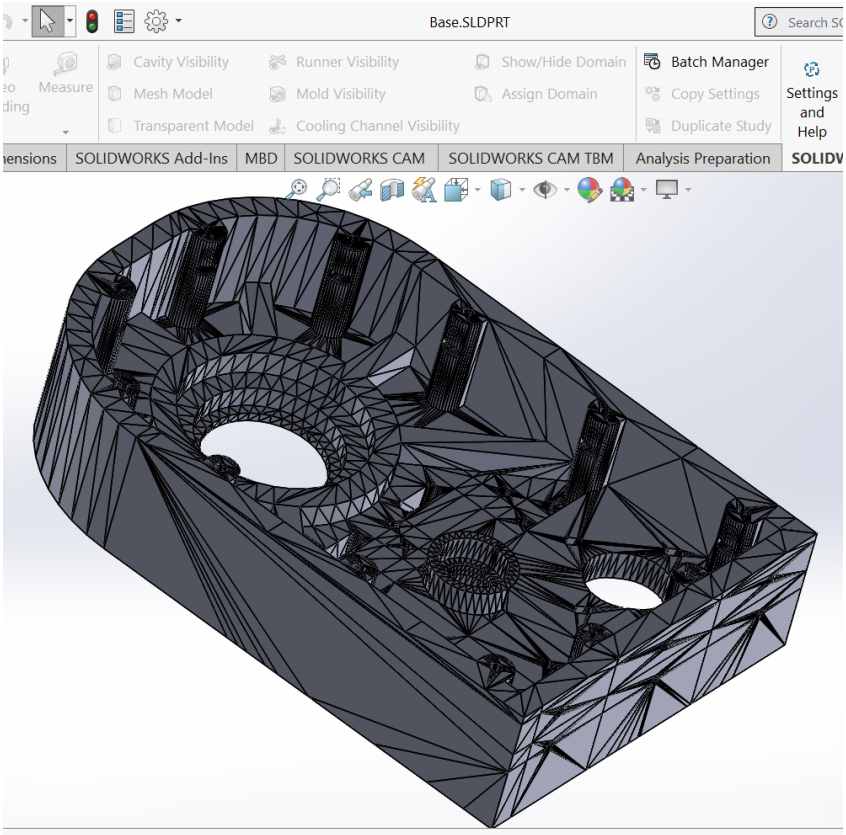


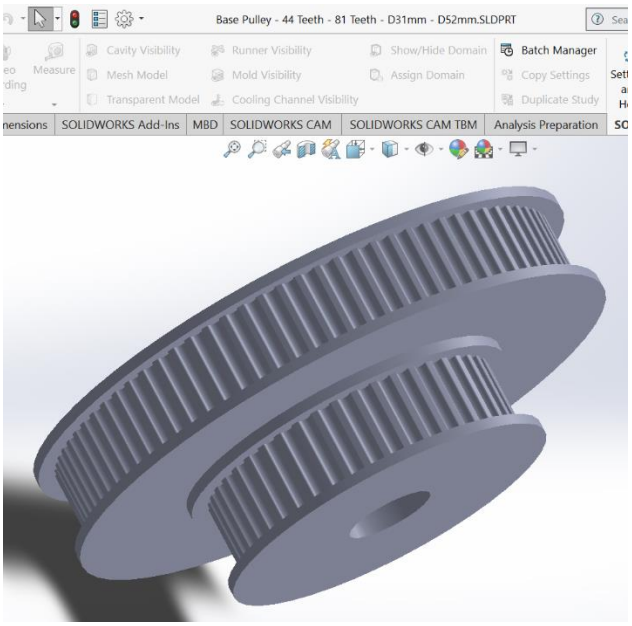

Arm 1 Pulley Middle - 44 Teeth - 99 Teeth - D31mm - D66mm.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Mesh Model
Mold Visibility
Assign Domain
Transparent Model
Cooling Channel Visibility
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

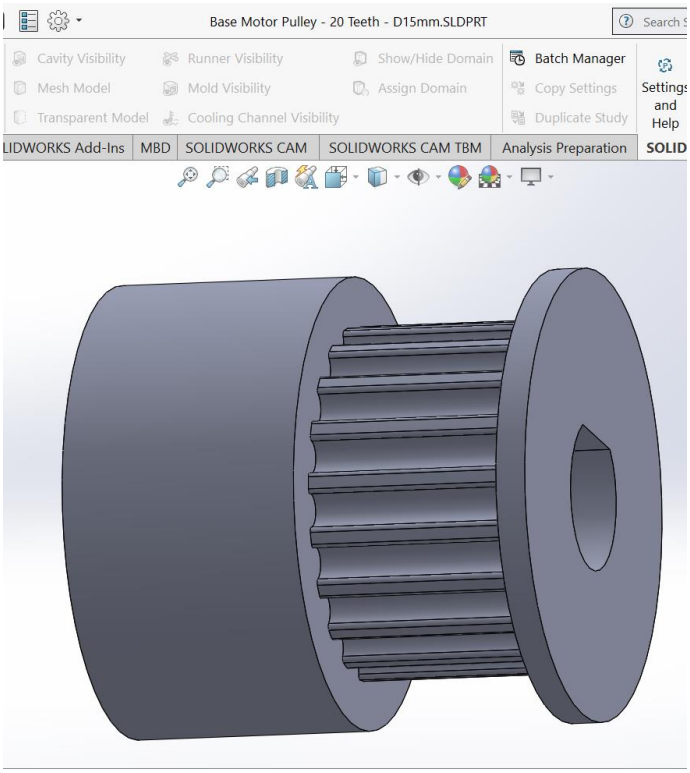

Arm 2 Motor Pulley - 68 Teeth ...
Show/Hide Domain
Batch Manage
Assign Domain
Copy Settings
Duplicate Stud
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

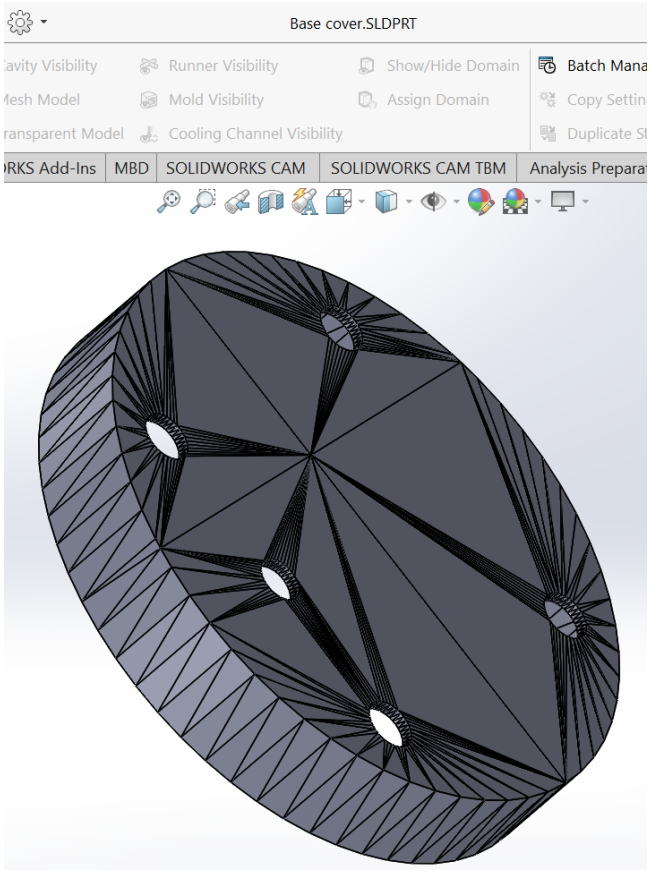

Arm 2.SLDPRT
Runner Visibility
Show/Hide Domain
Batch Manager
Mold Visibility
Assign Domain
Copy Settings
Cooling Channel Visibility
Duplicate Study
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

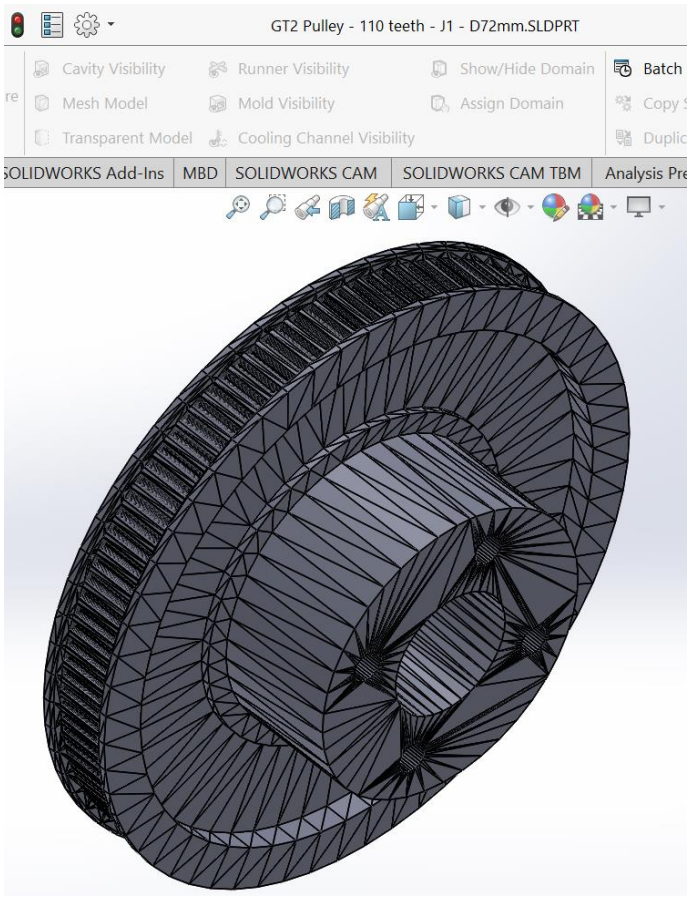

Arm 2 Cover.SLDPRT
Search SOL
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
Settings and Help
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

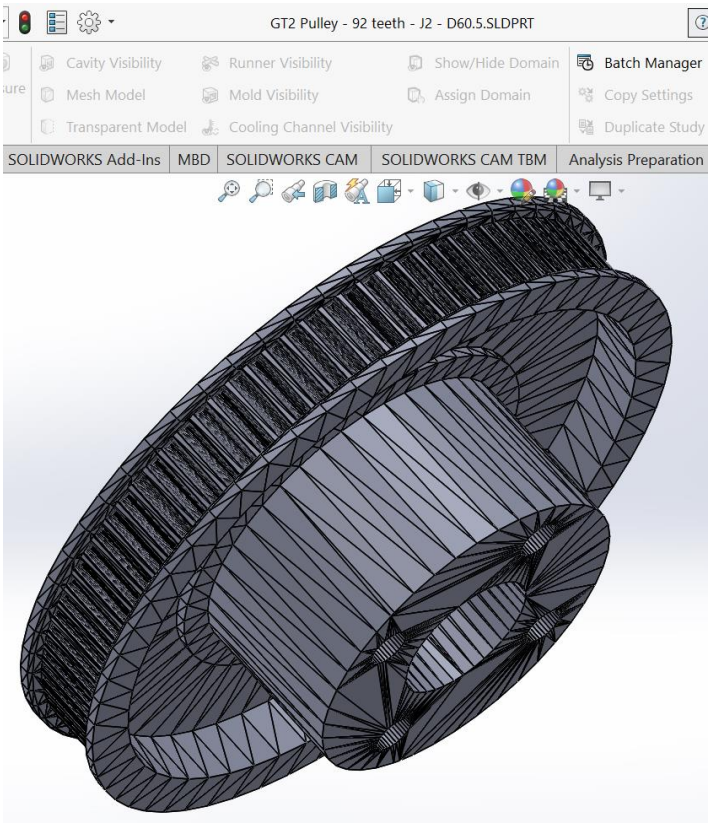

Base.SLDPRT
Search SO
Measure
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
Settings and Help
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

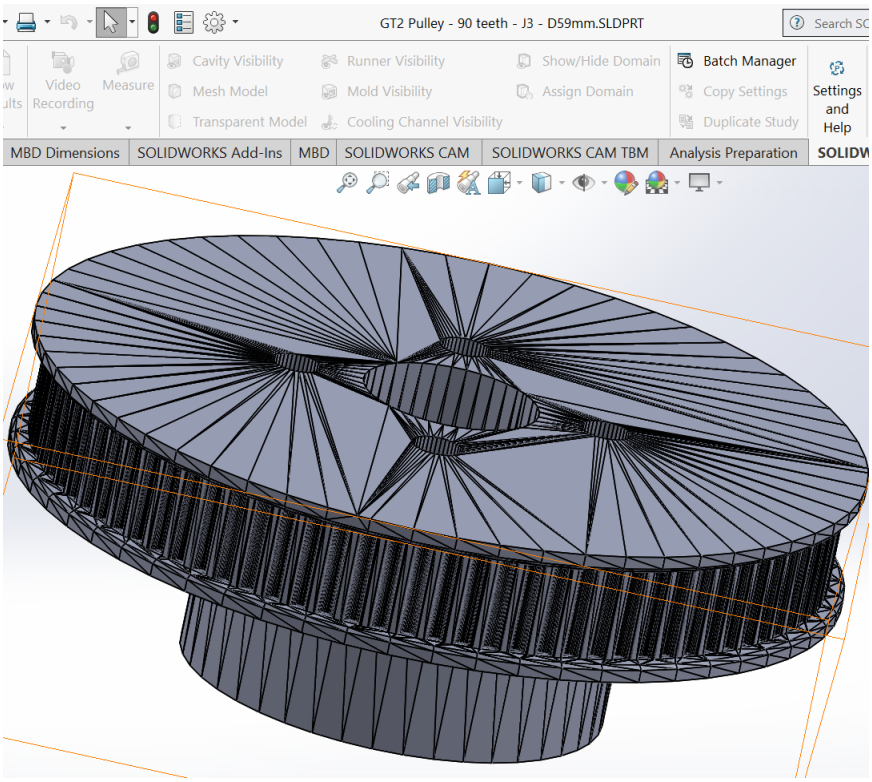

Base Pulley - 44 Teeth - 81 Teeth - D31mm - D52mm.SLDPRT
Measure
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

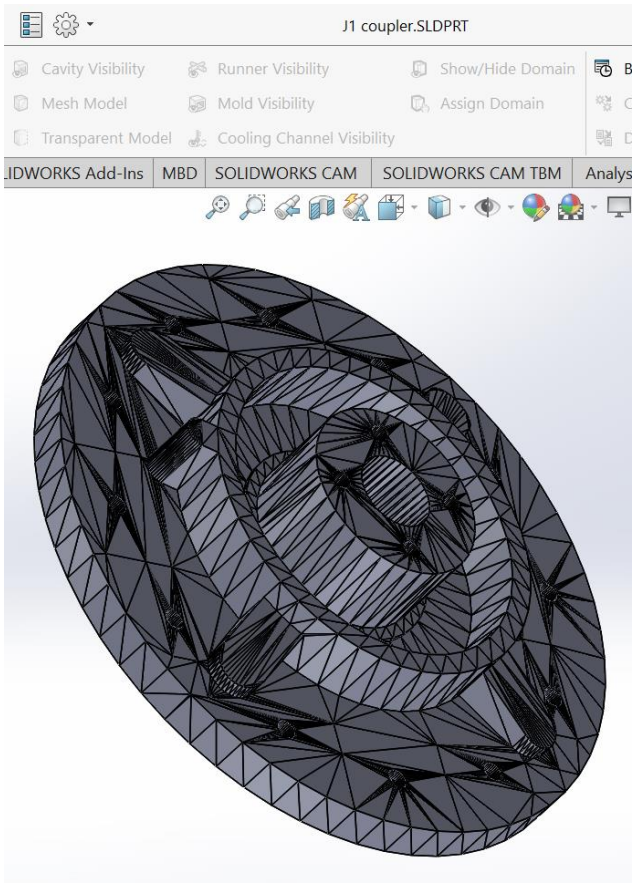

Base Motor Pulley - 20 Teeth - D15mm.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
Settings and Help
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

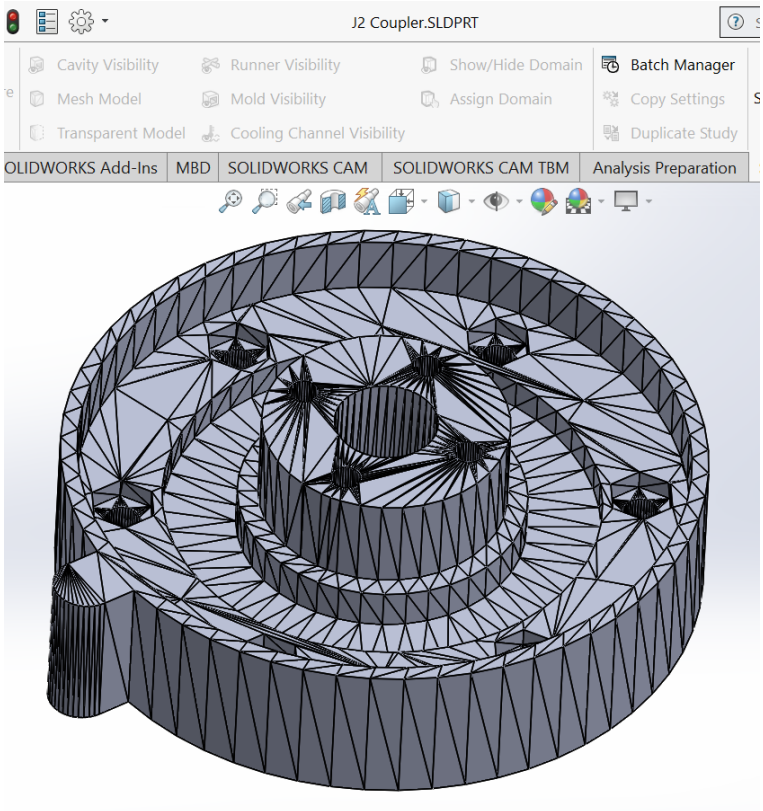

Base cover.SLDPRT
Runner Visibility
Show/Hide Domain
Mold Visibility
Assign Domain
Cooling Channel Visibility
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

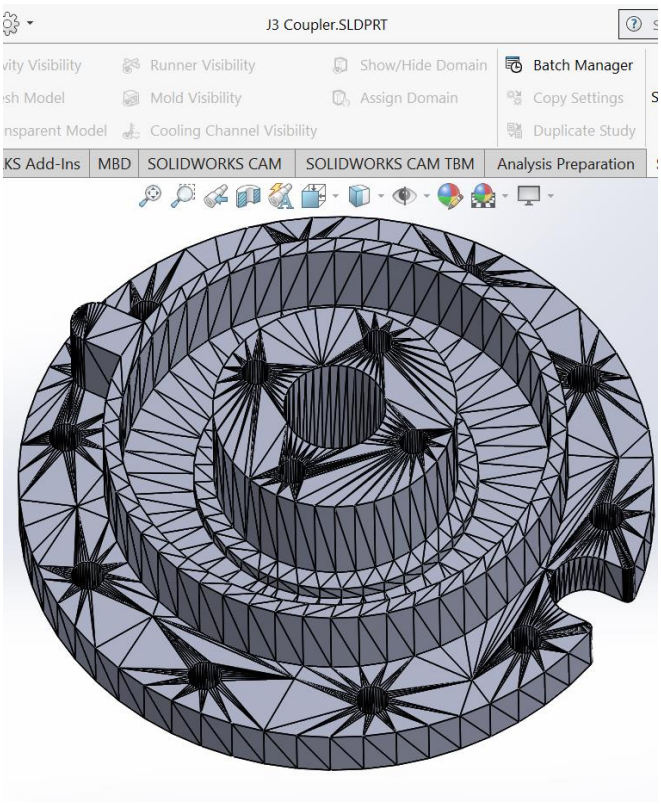

GT2 Pulley - 110 teeth - J1 - D72mm.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Mesh Model
Mold Visibility
Assign Domain
Transparent Model
Cooling Channel Visibility
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

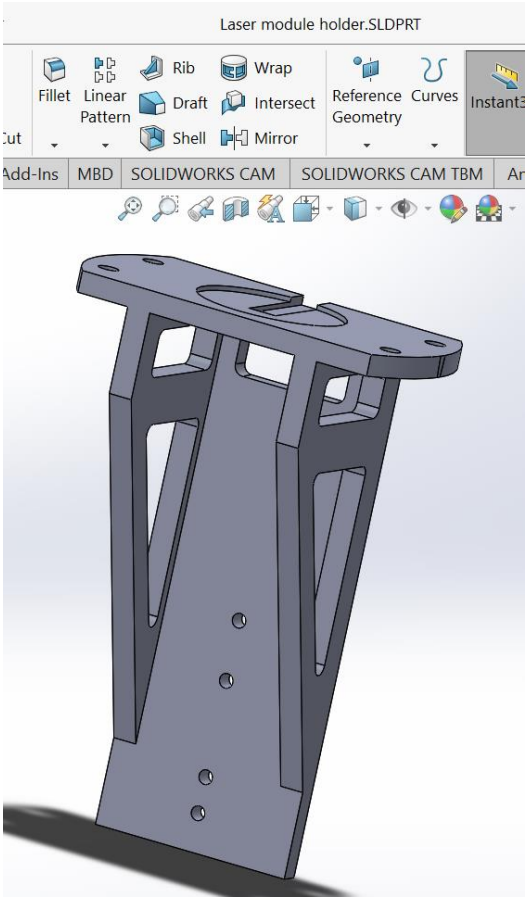

GT2 Pulley - 92 teeth - J2 - D60.5.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

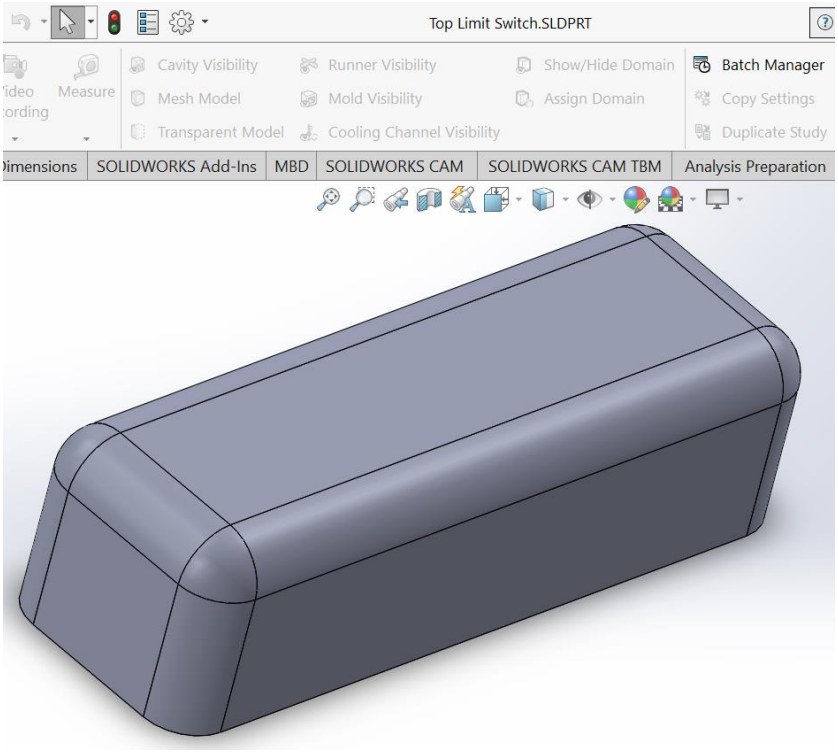

GT2 Pulley - 90 teeth - J3 - D59mm.SLDPRT
Search SC
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Video Recording
Measure
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Settings and Help
Transparent Model
Cooling Channel Visibility
Duplicate Study
MBD Dimensions
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

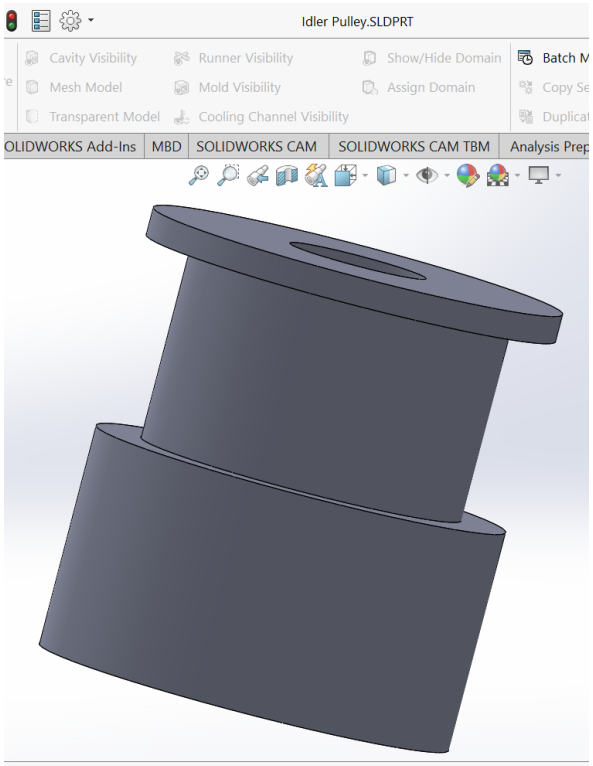

J1 coupler.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Mesh Model
Mold Visibility
Assign Domain
Transparent Model
Cooling Channel Visibility
IDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

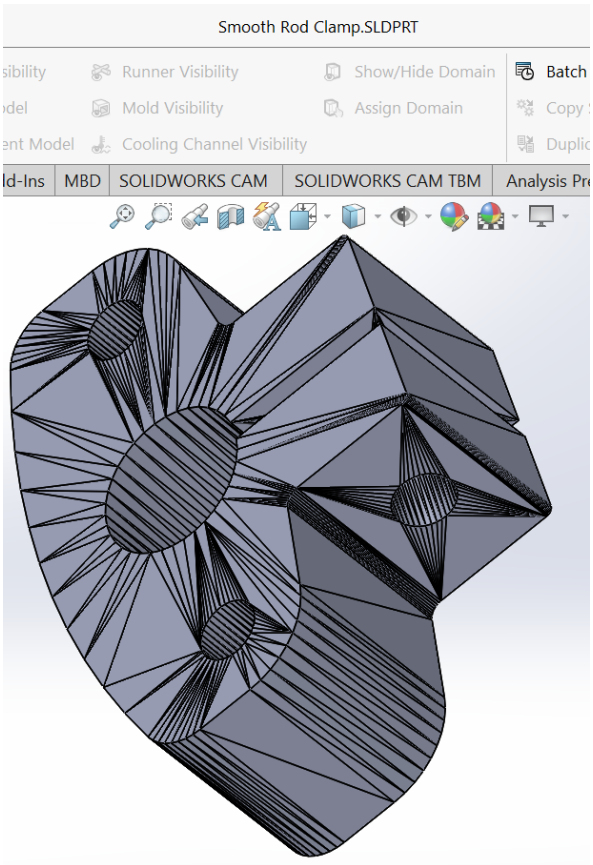

J2 Coupler.SLDPRT
Cavity Visibility
Runner Visibility
Show/Hide Domain
Batch Manager
Mesh Model
Mold Visibility
Assign Domain
Copy Settings
Transparent Model
Cooling Channel Visibility
Duplicate Study
OLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

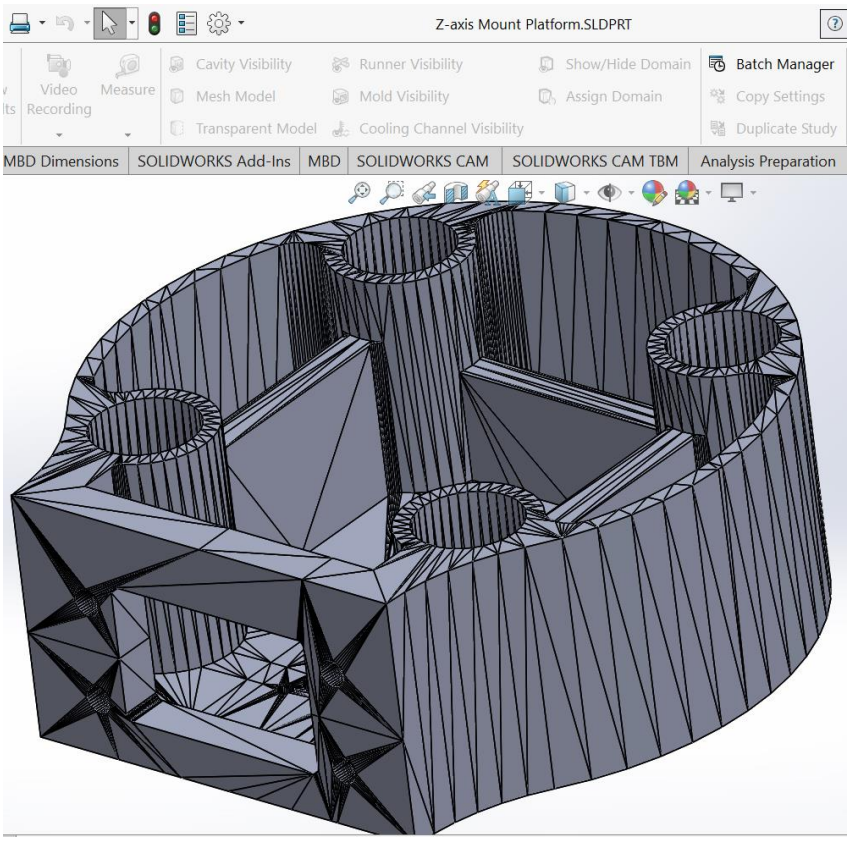

J3 Coupler.SLDPRT
Runner Visibility
Show/Hide Domain
Batch Manager
Mold Visibility
Assign Domain
Copy Settings
Cooling Channel Visibility
Duplicate Study
KS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

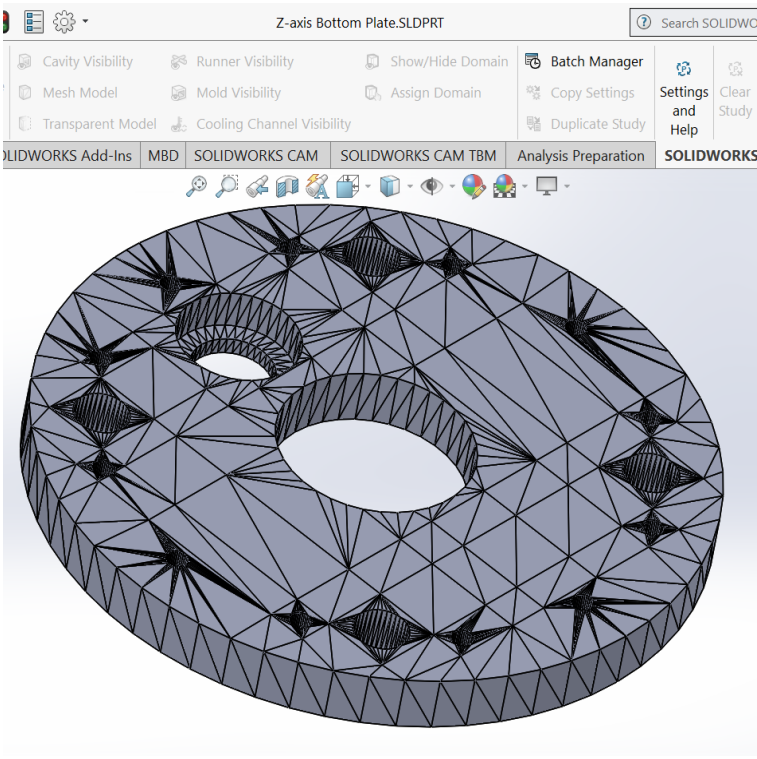

Laser module holder.SLDPRT
Fillet
Linear Pattern
Rib
Draft
Shell
Wrap
Intersect
Mirror
Reference Geometry
Curves
Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

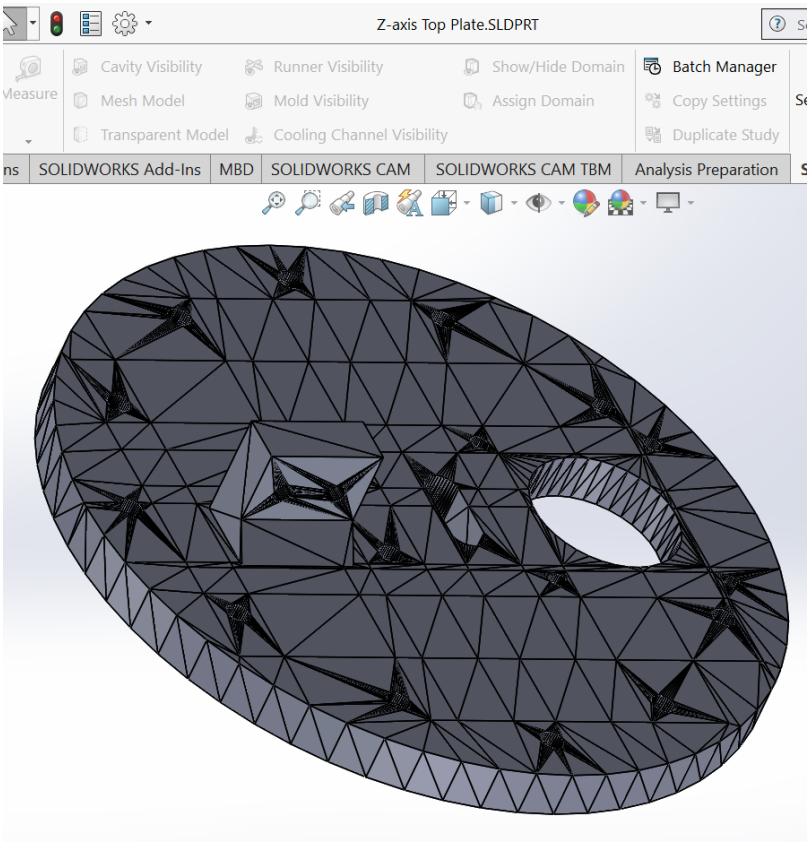

Top Limit Switch.SLDPRT
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manager
Copy Settings
Duplicate Study
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation

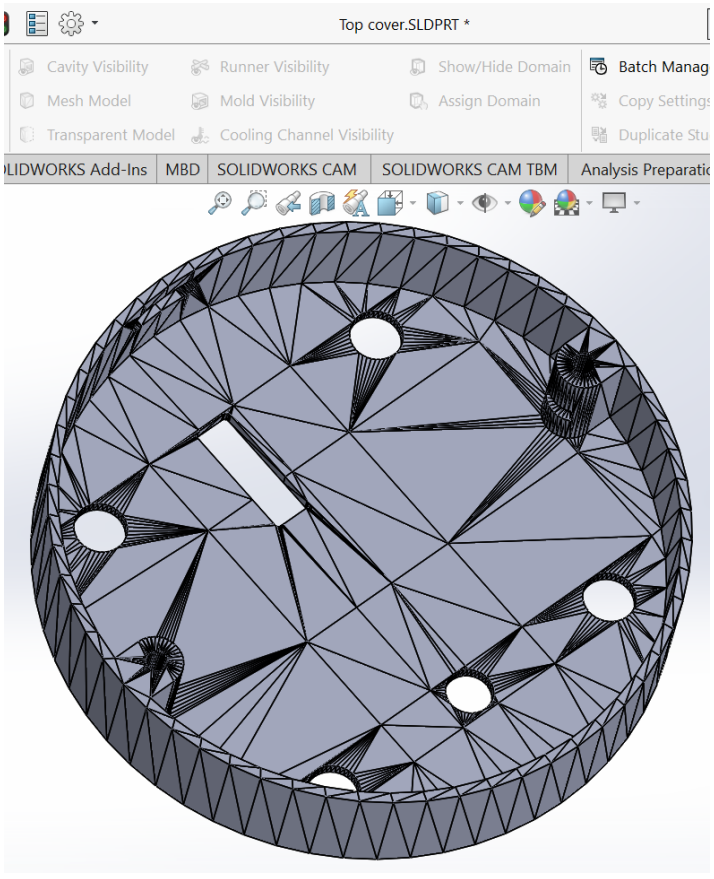

Idler Pulley.SLDPRT
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM

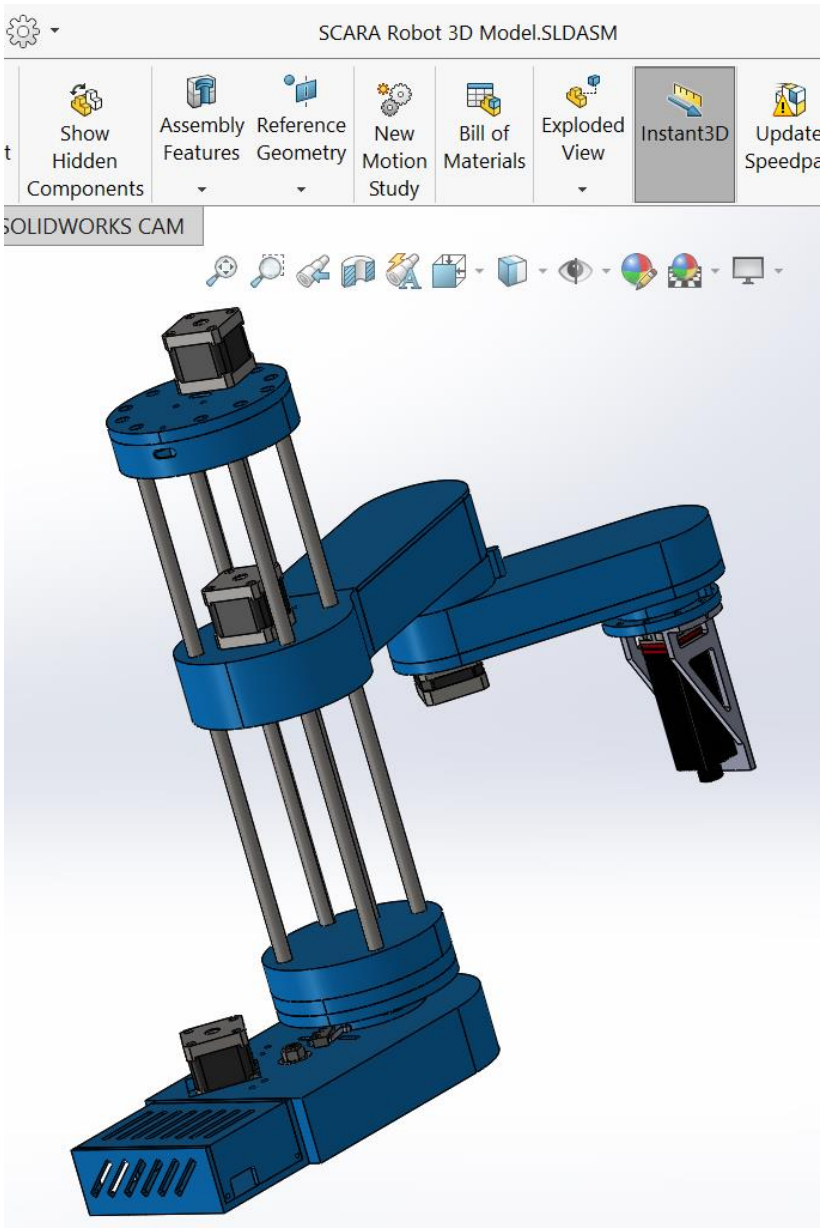

Smooth Rod Clamp.SLDPRT
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM


Z-axis Mount Platform.SLDPRT
Video Recording
Measure
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manager
Copy Settings
Duplicate Study
MBD Dimensions
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation


Z-axis Bottom Plate.SLDPRT
Search SOLIDWO
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manager
Copy Settings
Duplicate Study
Settings and Help
Clear Study
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation
SOLIDWORKS


Z-axis Top Plate.SLDPRT
Measure
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manager
Copy Settings
Duplicate Study
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation


Top cover.SLDPRT *
Cavity Visibility
Mesh Model
Transparent Model
Runner Visibility
Mold Visibility
Cooling Channel Visibility
Show/Hide Domain
Assign Domain
Batch Manag
Copy Settings
Duplicate Stu
SOLIDWORKS Add-Ins
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparatio


SCARA Robot 3D Model.SLDASM
Show Hidden Components
Assembly Features
Reference Geometry
New Motion Study
Bill of Materials
Exploded View
Instant3D
Update Speedpa
SOLIDWORKS CAM

Appendix B: Arduino motor and limit switch test codes

Motors code:

```
#define X_STEP_PIN         54
#define X_DIR_PIN          55
#define X_ENABLE_PIN       38
#define X_MIN_PIN           3
#define Y_STEP_PIN         60
#define Y_DIR_PIN          61
#define Y_ENABLE_PIN       56
#define Y_MIN_PIN          14
#define Z_STEP_PIN         46
#define Z_DIR_PIN          48
#define Z_ENABLE_PIN       62
#define Z_MIN_PIN          18
#define E_STEP_PIN         26
#define E_DIR_PIN          28
#define E_ENABLE_PIN       24
#define E_Min_Pin_Z_MAX_PIN          19
#define SDPOWER            -1
#define SDSS               53
#define LED_PIN            13
#define LASER_PIN           9
#define PS_ON_PIN          12
#define KILL_PIN           -1
void setup() {
  Serial.begin(9600);
  pinMode(X_STEP_PIN  , OUTPUT);
  pinMode(X_DIR_PIN    , OUTPUT);
  pinMode(X_ENABLE_PIN    , OUTPUT);
  pinMode(Y_STEP_PIN  , OUTPUT);
```

```
  pinMode(Y_DIR_PIN    , OUTPUT);
  pinMode(Y_ENABLE_PIN    , OUTPUT);
  pinMode(Z_STEP_PIN  , OUTPUT);
  pinMode(Z_DIR_PIN    , OUTPUT);
  pinMode(Z_ENABLE_PIN    , OUTPUT);
  pinMode(E_STEP_PIN  , OUTPUT);
  pinMode(E_DIR_PIN    , OUTPUT);
  pinMode(E_ENABLE_PIN    , OUTPUT);
  digitalWrite(X_ENABLE_PIN, LOW);
  digitalWrite(X_DIR_PIN,LOW);
  digitalWrite(X_STEP_PIN,LOW);
  digitalWrite(Y_ENABLE_PIN, LOW);
  digitalWrite(Y_DIR_PIN,LOW);
  digitalWrite(Y_STEP_PIN,LOW);
  digitalWrite(Z_ENABLE_PIN, LOW);
  digitalWrite(Z_DIR_PIN,LOW);
  digitalWrite(Z_STEP_PIN,LOW);
  digitalWrite(E_ENABLE_PIN, LOW);
  digitalWrite(E_DIR_PIN,LOW);
  digitalWrite(E_STEP_PIN,LOW);
}
void loop () {
  digitalWrite(X_DIR_PIN,HIGH);
  digitalWrite(X_STEP_PIN,LOW);
  delay(500);
  // 200 pulses makes one full cycle rotation
  for(int x = 0; x < 1000; x++) {
    digitalWrite(X_STEP_PIN,HIGH);
    delayMicroseconds(1000);
    digitalWrite(X_STEP_PIN,LOW);
    delayMicroseconds(1000);
```

```
}
digitalWrite(X_DIR_PIN,LOW);
digitalWrite(X_STEP_PIN,LOW);
delay(500);
for(int x = 0; x < 1000; x++) {
  digitalWrite(X_STEP_PIN,HIGH);
  delayMicroseconds(1000);
  digitalWrite(X_STEP_PIN,LOW);
  delayMicroseconds(1000);
}
digitalWrite(X_STEP_PIN,LOW);
digitalWrite(Y_DIR_PIN,HIGH);
digitalWrite(Y_STEP_PIN,LOW);
delay(500);
// 200 pulses makes one full cycle rotation
for(int x = 0; x < 1000; x++) {
  digitalWrite(Y_STEP_PIN,HIGH);
  delayMicroseconds(1000);
  digitalWrite(Y_STEP_PIN,LOW);
  delayMicroseconds(1000);
}
digitalWrite(Y_DIR_PIN,LOW);
digitalWrite(Y_STEP_PIN,LOW);
delay(500);
for(int x = 0; x < 1000; x++) {
  digitalWrite(Y_STEP_PIN,HIGH);
  delayMicroseconds(1000);
  digitalWrite(Y_STEP_PIN,LOW);
  delayMicroseconds(1000);
}
digitalWrite(Y_STEP_PIN,LOW);
```

```
digitalWrite(Z_DIR_PIN,HIGH);
digitalWrite(Z_STEP_PIN,LOW);
delay(500);
// 200 pulses makes one full cycle rotation
for(int x = 0; x < 2000; x++) {
  digitalWrite(Z_STEP_PIN,HIGH);
  delayMicroseconds(500);
  digitalWrite(Z_STEP_PIN,LOW);
  delayMicroseconds(500);
}
digitalWrite(Z_DIR_PIN,LOW);
digitalWrite(Z_STEP_PIN,LOW);
delay(500);
for(int x = 0; x < 2000; x++) {
  digitalWrite(Z_STEP_PIN,HIGH);
  delayMicroseconds(500);
  digitalWrite(Z_STEP_PIN,LOW);
  delayMicroseconds(500);
  digitalWrite(Z_STEP_PIN,LOW);
}
digitalWrite(E_DIR_PIN,HIGH);
digitalWrite(E_STEP_PIN,LOW);
delay(500);
// 200 pulses makes one full cycle rotation
for(int x = 0; x < 1000; x++) {
  digitalWrite(E_STEP_PIN,HIGH);
  delayMicroseconds(1000);
  digitalWrite(E_STEP_PIN,LOW);
  delayMicroseconds(1000);
}
digitalWrite(E_DIR_PIN,LOW);
```

```
    digitalWrite(E_STEP_PIN,LOW);
    delay(500);
    for(int x = 0; x < 1000; x++) {
      digitalWrite(E_STEP_PIN,HIGH);
      delayMicroseconds(1000);
      digitalWrite(E_STEP_PIN,LOW);
      delayMicroseconds(1000);
    }
    digitalWrite(E_STEP_PIN,LOW);
  }
```

Steppr driver code:

```
const int stepPin = 4;
const int dirPin = 3;
const int sleepPin = 5;
const int resPin = 6;
const int ePin = 7;
void setup() {
  // Sets the two pins as Outputs
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(sleepPin,OUTPUT);
  pinMode(resPin,OUTPUT);
  pinMode(ePin,OUTPUT);
  digitalWrite(sleepPin, HIGH);
  digitalWrite(resPin, HIGH);
  digitalWrite(ePin, LOW);
  digitalWrite(stepPin,LOW);
  digitalWrite(dirPin,LOW);
}
void loop() {
```

```
  digitalWrite(dirPin,HIGH); // Enables the motor to move in a particular direction
  digitalWrite(stepPin,LOW);
  delay(3000);
  // Makes 200 pulses for making one full cycle rotation
  for(int x = 0; x < 100; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(1000);
  }
  digitalWrite(dirPin,LOW); //Changes the rotations direction
  digitalWrite(stepPin,LOW);
  delay(3000);
  // Makes 400 pulses for making two full cycle rotation
  for(int x = 0; x < 100; x++) {
    digitalWrite(stepPin,HIGH);
    delayMicroseconds(1000);
    digitalWrite(stepPin,LOW);
    delayMicroseconds(1000);
  }
}
```

Limit Swtich code:

```
#define X_STEP_PIN         54
#define X_DIR_PIN          55
#define X_ENABLE_PIN       38
#define X_MIN_PIN           3
#define Y_STEP_PIN         60
#define Y_DIR_PIN          61
#define Y_ENABLE_PIN       56
#define Y_MIN_PIN          14
```

```
#define Z_STEP_PIN         46
#define Z_DIR_PIN          48
#define Z_ENABLE_PIN       62
#define Z_MIN_PIN          18
#define E_STEP_PIN         26
#define E_DIR_PIN          28
#define E_ENABLE_PIN       24
#define E_Min_Pin_Z_MAX_PIN          19
#define SDPOWER            -1
#define SDSS               53
#define LED_PIN            13
#define LASER_PIN           9
#define PS_ON_PIN          12
#define KILL_PIN           -1
void setup() {
  Serial.begin(9600);
  //COM pin connect to - port
  //NC pin connect to s port
  pinMode(Z_MIN_PIN  , INPUT_PULLUP);
}
void loop () {
 if(digitalRead(Z_MIN_PIN) == LOW){
  Serial.println("UNPRESSED");
}
 else if(digitalRead(Z_MIN_PIN) == HIGH){
  Serial.println("PRESSED");
 }
}
```