FACULTY OF ENGINEERING AND ARCHITECTURE
DEPARTMENT OF MECHATRONICS ENGINEERING

SIMULATION OF PARALLEL AND SERIAL MANIPULATORS IN ROS

ALİ AYDIN KÜÇÜKÇÖLLÜ - 160412046
BATUHAN YALÇINKAYA - 160412020
NURETTİN UĞUR ALAGAŞ - 160412010

SUPERVISOR: DR. FATİH CEMAL CAN

JUNE 2021

SIMULATION OF
PARALLEL AND SERIAL
MANIPULATORS IN ROS

A GRADUATION/DESIGN PROJECT

by

ALİ AYDIN KÜÇÜKÇÖLLÜ

BATUHAN YALÇINKAYA

NURETTİN UĞUR ALAGAŞ

submitted to the MECHATRONICS ENGINEERING of

İZMİR KÂTİP ÇELEBİ UNIVERSITY

Approved by:

Chair Name

(Signature)

Member Name	Member Name
(Signature)	(Signature)
Member Name	Member Name
(Signature)	(Signature)

JUNE 2021

ABSTRACT

In the current century, robots are rapidly replacing people. The importance of robots is increasing, especially in areas where human life is endangered, repetition is required or errors cannot be accepted. However, the production cost of robots is still quite high. Simulation environments are very important in the stage of testing robots before the production part. In this direction, the capabilities and limits of the robot are tested by adjusting the necessary environmental parameters thanks to the simulations. Thus, problems that may be encountered during the production phase are minimized. In this graduation project, the design, analysis and simulation of robot manipulators, which play a very important role in Industry 4.0, is the subject.

In this graduation project, it is aimed to design, analyze and simulate two manipulators which are one of them serial manipulator and the other one is parallel manipulator in ROS frame work. In this direction, the necessary literature survey about design, analysis and simulation of manipulators was done. The tasks, mobilities and joint types of both manipulators were determined at concept design part which is next step after literature survey. After that, position and velocity analysis of both manipulators were completed. In the detail design part, the mechanical designs of manipulators were designed with the help of SolidWorks based on concept design. The dynamic analysis of the serial manipulator was carried out. Then, the electronic circuit diagram of the serial manipulator was designed with the help of Proteus program. Finally, the task was performed in the simulation environment, which was determined by writing a script to perform the simulation and the parallel manipulator was simulated in SolidWorks.

Keywords: Serial and Parallel Manipulators, Simulation in ROS, Manipulator analysis in ROS, ROS

ÖZET

İçinde bulunduğumuz yüzyılda robotlar insanların yerini hızla doldurmaktadır. Özellikle insan hayatının tehlikeye atıldığı, tekrar gerektiren ya da hatanın kabul edilemeyeceği alanlarda robotların önemi artmaktadır. Buna rağmen, robotların üretim maliyeti halen oldukça yüksektir. Robotların üretimine geçmeden önce test edilme aşamasında, simülasyon ortamları oldukça önem taşımaktadır. Bu doğrultuda, simülasyonlar sayesinde gerekli çevresel parametreler ayarlanarak robotun kabiliyetleri ve limitleri test edilmiş olur. Böylece, üretim aşamasında karşılaşılabilecek sorunlar en aza indirgenir. Bu bitirme projesinde de Endüstri 4.0'da oldukça önemli rol üstlenen robot manipülatörlerinin tasarımı, analizi ve simülasyonu konu edinilmiştir.

Bu bitirme projesinde, biri seri diğeri paralel olmak üzere iki adet manipülatörün tasarlanıp, analiz edilerek ROS ortamında simülasyon edilmeleri amaçlanmıştır. Bu doğrultuda, manipülatör tasarımı, analizi ve simülasyonları hakkında gerekli literatür araştırmaları yapıldı. Bir sonraki adım olan konsept dizayn kısmında her iki manipülatörün görevleri, serbestlik dereceleri ve mafsal türleri belirlendi. Ardından, her iki manipülatörün pozisyon ve hız analizleri tamamlandı. Detaylı tasarımda, manipülatörlerin mekanik tasarımları konsept dizaynlar baz alınarak SolidWorks yardımıyla tasarlandı. Seri manipülatörün dinamik analizi yapılmıştır. Daha sonra Proteus programı yardımıyla seri manipülatörün elektronik devre şeması tasarlanmıştır. Son olarak simülasyonu gerçekleştirmek için bir komut dosyası yazılarak belirlenen simülasyon ortamında görev gerçekleştirilmiş ve paralel manipülatör SolidWorks'te simüle edilmiştir.

Anahtar kelimeler: Seri ve Paralel Manipülatörler, ROS'da Simülasyon, ROS'da manipülatör analizi, ROS

ACKNOWLEDGMENTS

Throughout the writing of this dissertation we have received a great deal of support and assistance.

We would like to thank our supervisor, Dr. Fatih Cemal Can, whose expertise was invaluable in finding solution that we faced-up problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

$\boldsymbol{i}$ Axis Number

$\boldsymbol{\theta}$ Joint Angle

$\boldsymbol{d}$ Link Offset

$\boldsymbol{a}$ Twist Angle

$\boldsymbol{a}$ Link Length

$\boldsymbol{Ø}$ Phi

$\boldsymbol{J}$ Jacobian Matrix

w Angular Velocity

v Linear Velocity

$\dot{w}$ Angular Acceleration

$\dot{v}$ Linear Acceleration

I Moment of Inertia

τ Torque

ACRONYMS

c(θ) cos(θ)

s(θ) sin(θ)

c(α) cos(α)

s(α) sin(α)

θ_{23} $\theta_2+\theta_3$

θ_{234} $\theta_2+\theta_3+\theta_4$

Δ Delta

SLAM Simultaneous Localization Mapping

ROS Robot Operating System

OpenRTM Open Robotics Technology Middleware

OROCOS Open Robot Control Software

OPRoS Open Platform for Robotic Services

3D 3-Dimentional

DARPA Defense Advanced Research Projects Agency

US United States

URDF Unified Robot Description Format

CAD Computer Aided Design

XML Extensible Markup Language

GUI General User Interface

DoF Degrees of Freedom

I. INTRODUCTION

The first term of *robot* – derived from "robota" which means subordinate labor in Slav languages – was first introduced by the Czech playwright Karel Čapek in his play "Rossum's Universal Robots (R.U.R)" in 1920.[18] In 1940, Russian science-fiction writer who is Isaac Asimov is mentioned about the *three fundamental laws of robots* in his novel- *I, Robot*:

1- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2- A robot must obey the orders given by human beings, except when such orders would conflict with the first law.
3- A robot must protect its own existence, as long as such protection does not conflict with the first or second law.

These three laws established rules of behavior to consider for the design of a robot. The meaning of *robot* and their designs have changed year by year. Today's meaning of robot is any automatically operated machine that replaces human effort, though it may not resemble human beings in appearance or perform functions in a humanlike manner. [19] Nowadays, the robots have lots of types and used in large of area such as robot manipulators in industry, mobile robots in military, etc.

1.1. Manipulators

The robot manipulators consist of two mechanical parts as *links* and *joints*. A manipulator is characterized by an *arm* that ensures mobility, a *wrist* that confers dexterity, and *end-effector* that performs the task required of the robot.[4] There are two types of manipulators in terms of kinematic chain. They are called as open kinematic chain, *serial manipulators*, and closed kinematic chain, *parallel manipulators*. There are lots of differences and similarities between serial and parallel manipulators. See Table 1.1.

1.1.1. Serial Manipulators

The serial manipulator, is also known as open kinematic chain, has limited accuracy due to the compounding of errors through each joint. Serial kinematic manipulator forms include Cartesian, cylindrical, spherical, SCARA as well as fully articulated configurations.[17] The usage area of serial manipulators is wide such as the Canadarm on the International Space Station, different types and marks of serial manipulators in the mass production, etc. See Figure 1.1.

Figure I-1 *KUKA Serial Manipulator* [22]

1.1.2. Parallel Manipulators

The parallel manipulator, is also known as closed kinematic chain. The development of *parallel manipulators* can be dated back to the early 1960s, when Gough and Whitehall first devised a six-linear jack system for use as a universal tire testing machine. Since 1980, there has been an increasing interest in the development of parallel manipulators. For a parallel kinematic mechanism, the kinematic equations will be considerably more complex due to the closed kinematic loops than for an open (serial) kinematic structure. [12] Parallel manipulators are used in medical applications, flight simulators, etc. See Figure 1.2.

Figure I-2 *Aircraft Simulator* [23]

Table I-1 *Comparison between parallel and serial manipulators* [12]

	Type Of Manipulator	
	Parallel Manipulator	**Serial Manipulator**
Type of manipulators	Closed loop	Open loop
End effectors	Platform	Gripper
Natural description	In Cartesian space	In joint space
Location of actuators	Near the immobile base	On the links
Inertia forces & stiffness	Less and high respectively	High and less respectively
Design considerations	Structure, workspace considerations, singularities, link interference	Strength and stiffness considerations, vibration characteristics
Preferred property	Stiffness	Dexterity
Use of direct kinematics	Difficult and complex	Straightforward and unique
Use of inverse kinematics	Straightforward and unique	Complicated
Singularity	Static	Kinematic
Direct force transformation	Well defined and unique	Not well defined; may be non-existent, unique or infinite
Preferred application	Precise positioning	Gross motion

1.2 Robot Software Platform

A platform is divided into software platform and hardware platform. A robot software platform includes tools that are used to develop robot application programs such as hardware abstraction, low-level device control, sensing, recognition, SLAM (Simultaneous Localization Mapping), navigation, manipulation and package management, libraries, debugging and development tools. [1]

Among these software platforms, major platforms are Robot Operating System (ROS), Japanese Open Robotics Technology Middleware (OpenRTM), European real-time control centered OROCOS, Korean OPRoS, etc. Although their names are different, the fundamental reason of advent of robot software platforms is because there are too many different kinds of robot software, and their complications are causing many problems. [1]

1.2.1 Robot Operating System (ROS)

ROS, is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.[20]

In this project the Robot Operating System(ROS) is used for this project. The ROS is chosen since it is most common robot software platform that used in many project and there are lots of advantages of ROS. ROS consist of a client library to support various programming languages. C++, Python, etc. can be used while developing projects. It is open-source that means lots of people can be develop their projects and they can be share as package. These are most popular two advantages of ROS.

Additionally, simulations are very useful tools to examine the behavior of robots in the real life. It is the process of designing a model of an actual or theoretical physical system, executing the model, and analyzing the execution output. A large amount of simulation software is available for robot systems, and it is already being used extensively. ROS contains some simulators such as RViz, and Gazebo.

Gazebo is a 3D simulator that provides robots, sensors, environment models for 3D simulation required for robot development, and offers realistic simulation with its physics engine. Gazebo is one of the most popular simulators in recent years and has been selected as the official simulator of the DARPA Robotics Challenge in the US. It is one of the most famous simulators in the field of robotics because of its high performance and various plugins. Moreover, Gazebo is developed and distributed by Open Robotics, which is in charge of ROS and its community, so it is compatible with ROS. [21] See Figure 1.3.

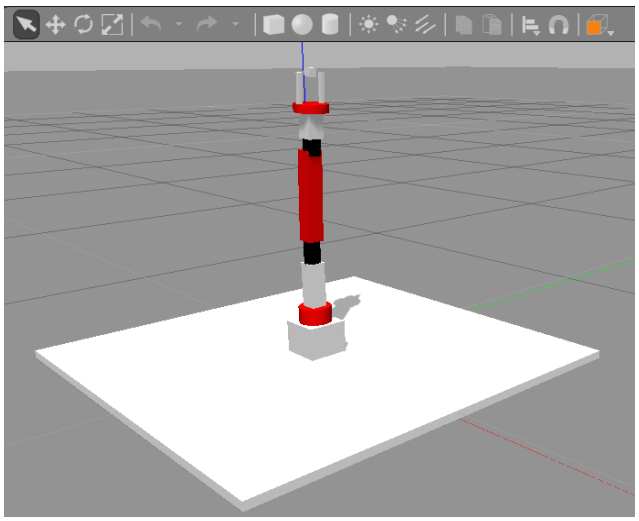

Figure I-3 *Simulation of manipulator in Gazebo* [21]

RViz is the 3D visualization tool of ROS. The main purpose is to show ROS messages in 3D, allowing us to visually verify data. It also supports various visualization using user specified polygons, and Interactive Markers allow users to perform interactive movements with commands and data received from the user node. In addition, ROS describes robots in Unified

Robot Description Format (URDF), which is expressed as a 3D model for which each model can be moved or operated according to their corresponding degree of freedom, so they can be used for simulation or control. [1] See in Figure 1.4.

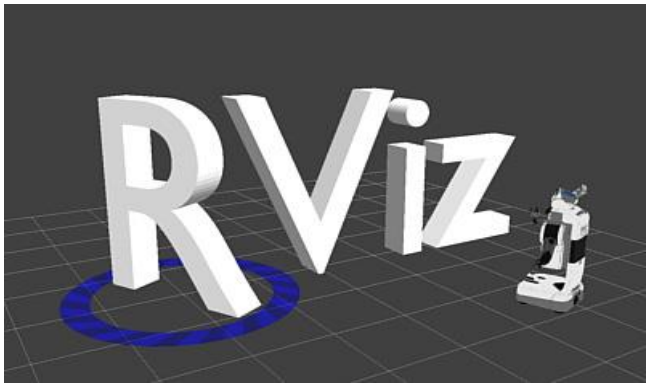

Figure I-4 *Loading screen of RViz* [1]

In addition, there is one more topic to use inverse kinematic in ROS. MoveIt is a set of packages and tools for doing mobile manipulation in ROS. It is a package that can be used with serial and parallel manipulators as efficiently and automatically. It is also programmable with C++ and Python programming languages. See Figure 1.5.

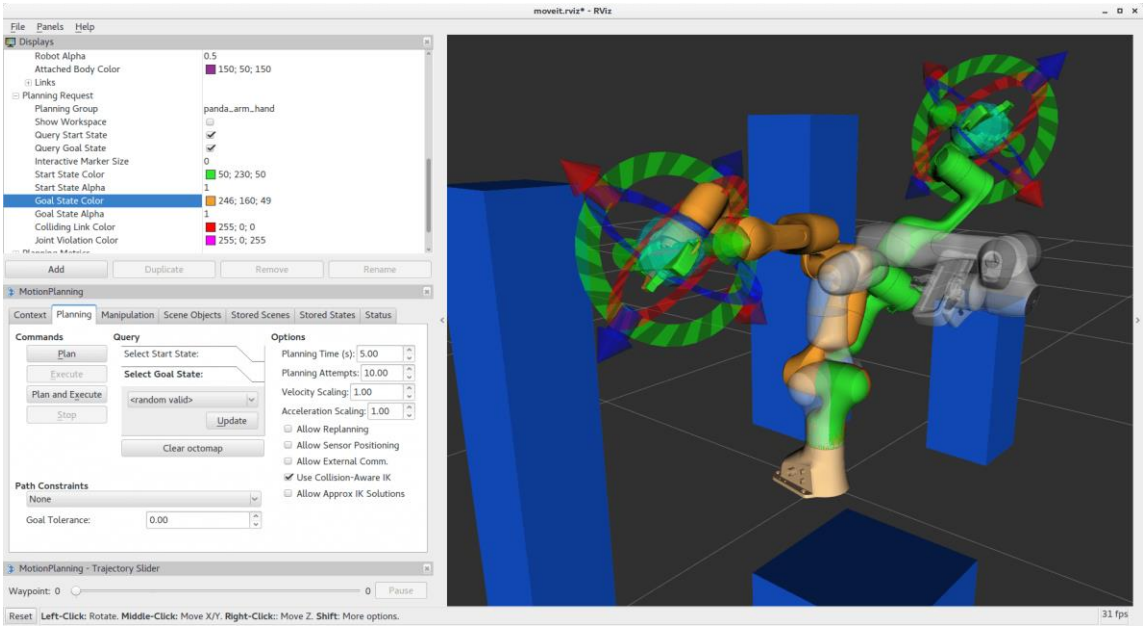

Figure I-5 *MoveIt in ROS* [24]

The main purpose of this project is that simulation of serial and parallel manipulators in ROS. In this direction, the serial and parallel manipulators are going to be designed, analyzed and simulated in ROS via using simulators; Gazebo and RViz. The MoveIt is going to be used to inverse kinematics.

II. CONCEPT DESIGN

In the concept design, Structural Synthesis, the main purpose is that design the mechanism according to task. Therefore, number of links and types of the joints revolute, prismatic, spherical, etc. should be chosen with respect to required mobility of task. The chosen joint types affect the mobility of mechanism. What determines the number of actual mobility of the system is the number of actuators which actuated system except redundant mechanisms.

2.1. Structural Synthesis of Serial Manipulator

A Serial Manipulator design be formed of open kinematic chain and what has changed is that joint types and place of the joints, parallel or perpendicular to the mechanism. Assigned task is pick and place for the serial manipulator. In this direction, four degrees of freedom, except end-effector, serial manipulator is designed via placing the revolute joints as parallel. Therefore, the serial manipulator has five link with ground and four revolute joint.

2.2. Structural Synthesis of Parallel Manipulator

There are lots of type Parallel Manipulators due to form of closed kinematic chain for instance, Hexapod, Delta Robots, etc. Linear Delta Robot is one of the type of Delta Robots and it is currently used in the industry such as 3D printing. Due to it is widely used in the industry, it was decided to design a linear delta robot. The Linear Delta Robots be formed of three prismatic joints and lots of spherical joints. The number of spherical joints can change

according to design of mechanism. Considering these, it is decided that the Linear Delta Robot will be formed of twelve spherical joint in addition of the three prismatic joint. The mobility of the mechanism is three via actuated the three prismatic joints with 3 step motors.

III. KINEMATIC ANALYSIS

In the kinematic analysis, the relation between joint states and end effector should be defined for both serial and parallel manipulators. This relation contains forward and inverse kinematics. In the serial and parallel manipulator analysis, the end effector position and orientation can be obtained by forward kinematics inversely, the joint states can be obtained by inverse kinematics. In addition of these, Jacobian Analysis is needed to reach linear and angular velocities of the end effectors.

3.1. Serial Manipulator Kinematics Analysis

The kinematic analysis of the serial manipulator changes depending on reference coordinates of the joints, placing the joints to the manipulator as parallel or perpendicular care of previous joint and offset(s) between the two joints. Considering the designed serial manipulator, there is an offset between first and second revolute joints because the joints are perpendicular each other and origin of reference coordinate is taken on the first joint.

3.1.1 Forward Kinematics

The Denavit-Hartenberg method is used to find end effector position and orientation. In this direction, homogenous transformation matrices of four revolute joints are found. A homogenous transformation matrix contains Joint Angle, Link Offset, Link Length and Twist Angle where Joint Angle is angle between axes x_{i-1} and x_i about z_{i-1} to be taken positive when rotation is made counter-clockwise, Link Offset is coordinate of $O_{i'}$, along z_{i-1}, Link Length is distance between O_i and $O_{i'}$, Twist Angle is angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise. The Denavit-Hartenberg

table of serial manipulator is shown as Table 3.1 and kinematic representation of the serial manipulator is given below as Figure 3.1.

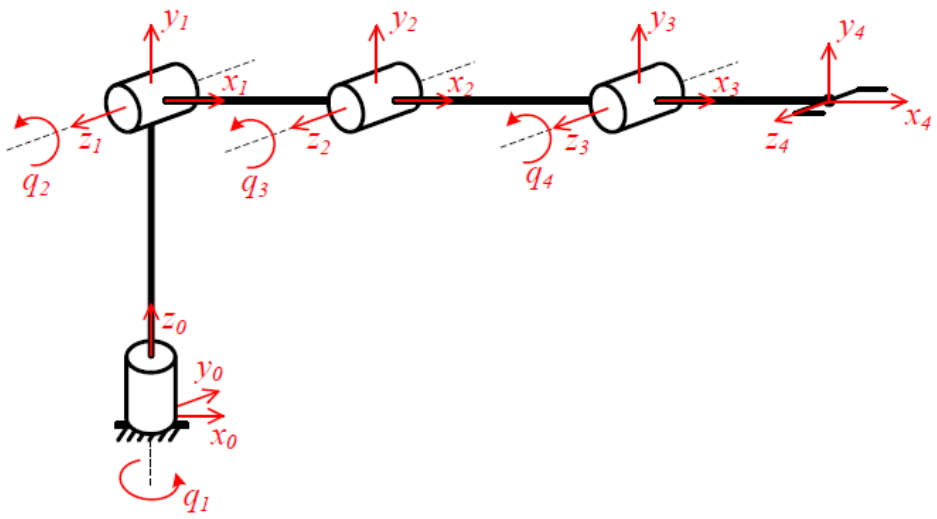

Figure III-1 *The kinematic representation of the serial manipulator*

Table III-1 *Denavit-Hartenberg Table of the serial manipulator*

Axis Number(i)	**Joint Angle(θ_i)**	**Link Offset(d_i)**	**Link Length(a_i)**	**Twist Angle(α_i)**
1	θ_1	d_1=L_1	0	$\pi/2$
2	θ_2	0	L_2	0
3	θ_3	0	L_3	0
4	θ_4	0	L_4	0

where, $\theta_1, \theta_2, \theta_3$ and θ_4 represent Joint Angles, d_1 represents Link Offset, it is also equals to L_1 and L_2, L_3 and L_4 represents link length.

A homogenous transformation matrix gives information about position and orientation of the related joint(s) and it is possible to obtain the final position and orientation of the end effector thanks to multiplying the homogeneous transformation matrices of the all joints with each other. The rotation matrix which is formed by three by three on the left top corner and position matrix which is formed by three by one on the right top corner and it gives the x, y and z coordinates.

$$H = \begin{bmatrix} R_{3x3} & P_{3x1} \\ 0\ 0\ 0 & 1 \end{bmatrix}$$

where, H is transformation matrix, R_{3x3} is rotation matrix and P_{3x1} is position matrix. The formula of the homogeneous transformation matrix occurs via multiplying of the four Denavit-Hartenberg parameter matrices for each joint is as following.

$$A_i = Rotation(z, \theta_i) * Translation(z, d_i) * Translation(x, a_i) * Rotation(x, \alpha_i)$$

$$A_i = \begin{bmatrix} \mathrm{c}(\theta_i) & -\mathrm{s}(\theta_i) & 0 & 0 \\ s(\theta_i) & \mathrm{c}(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{c}(\alpha_i) & -\mathrm{s}(\alpha_i) & 0 \\ 0 & \mathrm{s}(\alpha_i) & \mathrm{c}(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, $\mathrm{c}(\theta_i) = \cos(\theta_i)$, $s(\theta_i) = \sin(\theta_i)$ and $\mathrm{c}(\alpha_i) = \cos(\alpha_i)$, $\mathrm{s}(\alpha_i) = \sin(\alpha_i)$. The end effector transformation matrix is given below which consist of the homogeneous transformation matrices of the all joints.

$${}^{0}_{4}T = {}^{0}_{1}A * {}^{1}_{2}A * {}^{2}_{3}A * {}^{3}_{4}A$$

$${}^{0}_{4}T = \begin{bmatrix} \cos(\theta_1) * \cos(\theta_{234}) & -\cos(\theta_1) * \sin(\theta_{234}) & \sin(\theta_1) & P_x \\ \sin(\theta_1) * \cos(\theta_{234}) & -\sin(\theta_1) * \sin(\theta_{234}) & -\cos(\theta_1) & P_y \\ \sin(\theta_{234}) & \cos(\theta_{234}) & 0 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where,

$$P_x = L_2 * \cos(\theta_1) * \cos(\theta_2) + L_3 * \cos(\theta_1) * \cos(\theta_{23}) + L_4 * \cos(\theta_1) * \cos(\theta_{234})$$

$$P_y = L_2 * \sin(\theta_1) * \cos(\theta_2) + L_3 * \sin(\theta_1) * \cos(\theta_{23}) + L_4 * \sin(\theta_1) * \cos(\theta_{234})$$

$$P_z = L_1 + L_2 * \sin(\theta_2) + L_3 * \sin(\theta_{23}) + L_4 * \sin(\theta_{234}).$$

Therefore, the Forward Kinematic Analysis of the serial manipulator is completed.

3.1.2. Inverse Kinematics

In this step, the position and orientation of the end effector is already known from the Forward Kinematic Analysis of the serial manipulator. It is needed to calculate joint states with comparing the transformation matrices which are obtained from the Forward Kinematic Analysis and defined as parametric in the Inverse Kinematic Analysis and the parametric transformation matrix is given as following,

$$T_e = \begin{bmatrix} u_x & v_x & w_x & q_x \\ u_y & v_y & w_y & q_y \\ u_z & v_z & w_z & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, T_e is end effector transformation matrix.

If only rotation matrices compared with each other the matrices come like this,

$$\begin{bmatrix} \cos(\theta_1) * \cos(\theta_{234}) & -\cos(\theta_1) * \sin(\theta_{234}) & \sin(\theta_1) \\ \sin(\theta_1) * \cos(\theta_{234}) & -\sin(\theta_1) * \sin(\theta_{234}) & -\cos(\theta_1) \\ \sin(\theta_{234}) & \cos(\theta_{234}) & 0 \end{bmatrix} = \begin{bmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{bmatrix}$$

and the following equations can be taken from the matrices,

$$w_x = \sin(\theta_1), \qquad w_y = -\cos(\theta_1)$$

$$u_z = \sin(\theta_{234}), \qquad v_z = \cos(\theta_{234})$$

hence, θ_1 and θ_{234} can obtained as following.

$$\theta_1 = Atan2(-\cos(\theta_1), \sin(\theta_1))$$

and

$$\theta_{234} = Atan2(\cos(\theta_{234}), \sin(\theta_{234}))$$

It is also possible to reach the angle of third joint, θ_3 and the angle of second joint, θ_2, with some complex trigonometric equations as following,

$$\theta_3 = cos^{-1}\left[\frac{a^2 + b^2 + c^2 - {L_2}^2 - {L_3}^2}{2 * L_2 * L_3}\right]$$

where,

$$a = (q_x - L_4 * \cos(\theta_{234}) * \cos(\theta_1))$$
$$b = \left(q_y - L_4 * \cos(\theta_{234}) * \sin(\theta_1)\right)$$
$$c = (q_z - L_1 - L_4 * \sin(\theta_{234}))$$

$$\theta_2 = tan^{-1}[d, \pm f] - tan^{-1}(L_3 * \sin(\theta_3), L_2 + L_3 * \cos(\theta_3))$$

where,

$$d = q_z - L_1 * L_4 * \sin(\theta_{234})$$

$$f = \sqrt{((L_3 * \sin(\theta_3))^2 + (L_2 + L_3 * \cos(\theta_3))^2 - (q_z - L_1 - L_4 * \sin(\theta_{234}))^2)}.$$

The angle $\emptyset$ is known from the end effector angle and if it is consider that θ_2 and θ_3 are already known than the fourth joint angle can be found as,

$$\emptyset = \theta_2 + \theta_3 + \theta_4 \quad \text{then} \quad \theta_4 = \emptyset - \theta_2 - \theta_3.$$

3.1.3. Jacobian

The Jacobian matrix is used to find linear and angular velocities of the end effector depending on the joint velocities and the equation is given below. At the four by one matrix on the right hand side, $\dot{\theta}_\iota$ represents velocity of the joints and at the left hand side the six by one matrix represents the angular and linear velocities of the end effector.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = \begin{bmatrix} \quad J \quad \end{bmatrix} * \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \\ \dot{\theta}_4 \end{bmatrix}$$

The Jacobian matrix is formed as six by four and the reason is that six rows represent velocity parameters which are linear velocities along x, y, z and angular velocities around x, y, z axes and four columns represents number of joints. It is given as following,

$$J = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} \\ J_{21} & J_{22} & J_{23} & J_{24} \\ J_{31} & J_{32} & J_{33} & J_{34} \\ J_{41} & J_{42} & J_{43} & J_{44} \\ J_{51} & J_{52} & J_{53} & J_{54} \\ J_{61} & J_{62} & J_{63} & J_{64} \end{bmatrix}$$

where, first column,

$$J_{11} = -(L_2 * \sin(\theta_1) * \cos(\theta_2) + L_3 * \ \sin(\theta_1) * \cos(\theta_{23}) + L_4 * \sin(\theta_1) * \cos(\theta_{234}))$$

$$J_{21} = L_2 * \cos(\theta_1) * \cos(\theta_2) + L_3 * \ \cos(\theta_1) * \cos(\theta_{23}) + L_4 * \cos(\theta_1) * \cos(\theta_{234})$$

$$J_{31} = 0, J_{41} = 0, J_{51} = 0, J_{61} = 1$$

second column,

$$J_{12} = -\cos(\theta_1) * (L_2 * \sin(\theta_2) + L_3 * \sin(\theta_{23}) + L_4 * \sin(\theta_{234}))$$

$$J_{22} = -\sin(\theta_1) * (L_2 * \sin(\theta_2) + L_3 * \sin(\theta_{23}) + L_4 * \sin(\theta_{234}))$$

$$J_{32} = (\sin(\theta_1)) * (L_2 * \sin(\theta_1) * \cos(\theta_2) + L_3 * \sin(\theta_1) * \cos(\theta_{23}) + L_4 * \sin(\theta_1) * \cos(\theta_{234})) + (\cos(\theta_1)) * (L_2 * \cos(\theta_1) * \cos(\theta_2) + L_3 * \cos(\theta_1) * \cos(\theta_{23}) + L_4 * \cos(\theta_1) * \cos(\theta_{234}))$$

$$J_{42} = \sin(\theta_1), J_{52} = -\cos(\theta_1), J_{62} = 0$$

third column,

$$J_{13} = (-\cos(\theta_1)) * (L_4 * \sin(\theta_{234}) + L_3 * \sin(\theta_{23}))$$

$$J_{23} = (-\sin(\theta_1)) * (L_4 * \sin(\theta_{234}) + L_3 * \sin(\theta_{23}))$$

$$J_{33} = (\sin(\theta_1)) * (L_4 * \sin(\theta_1) * \cos(\theta_{234}) + L_3 * \sin(\theta_1) * \cos(\theta_{23})) + (\cos(\theta_1)) * (L_4 * \cos(\theta_1) * \cos(\theta_{234}) + L_3 * \cos(\theta_1) * \cos(\theta_{23}))$$

$$J_{43} = \sin(\theta_1), J_{53} = -\cos(\theta_1), J_{63} = 0$$

fourth column,

$$J_{14} = -\cos(\theta_1) * L_4 * \sin(\theta_{234})$$

$$J_{24} = -\sin(\theta_1) * L_4 * \sin(\theta_{234})$$

$$J_{34} = (\sin(\theta_1)) * (L_4 * \sin(\theta_1) * \cos(\theta_{234})) + (\cos(\theta_1)) * (L_4 * \cos(\theta_1) * \cos(\theta_{234}))$$

$$J_{44} = \sin(\theta_1), J_{54} = -\cos(\theta_1), J_{64} = 0$$

then, after the related calculations, the linear and angular velocities can be found as following,

$$\dot{x} = -(\dot{\theta_1}) * (L_2 * \sin(\theta_1) * \cos(\theta_2) + L_3 * \sin(\theta_1) * \cos(\theta_{23}) + L_4 * \sin(\theta_1) * \cos(\theta_{234})) - (\dot{\theta_3}) * \cos(\theta_1) * (L_4 * \sin(\theta_{234}) + L_3 * \sin(\theta_{23})) - (\dot{\theta_2}) * \cos(\theta_1) * (L_2 * \sin(\theta_2) + L_3 * \sin(\theta_{23}) + L_4 * \sin(\theta_{234})) - (\dot{\theta_4}) * (L_4 * \sin(\theta_{234}) * \cos(\theta_1))$$

$$\dot{y} = (\dot{\theta_1}) * (L_2 * \cos(\theta_1) * \cos(\theta_2) + L_3 * \cos(\theta_1) * \cos(\theta_{23}) + L_4 * \cos(\theta_1) * \cos(\theta_{234})) - (\dot{\theta_3}) * \sin(\theta_1) * (L_4 * \sin(\theta_{234}) + L_3 * \sin(\theta_{23})) - (\dot{\theta_2}) * \sin(\theta_1) * (L_2 * \sin(\theta_2) + L_3 * \sin(\theta_{23}) + L_4 * \sin(\theta_{234}) - (\dot{\theta_4}) * (L_4 * \sin(\theta_1) * \sin(\theta_{234})))$$

$$\dot{z} = (\dot{\theta_3}) * (\cos(\theta_1) * (L_3 * \cos(\theta_1) * \cos(\theta_{23}) + L_4 * \cos(\theta_1) * \cos(\theta_{234})) + \sin(\theta_1) * (L_3 * \sin(\theta_1) * \cos(\theta_{23}) + L_4 * \sin(\theta_1) * \cos(\theta_{234})) + (\dot{\theta_4}) * (L_4 * \cos(\theta_{234})) + (\dot{\theta_2}) * (\cos(\theta_1) * (L_3 * \cos(\theta_1) * \cos(\theta_{23}) + L_2 * \cos(\theta_1) * \cos(\theta_2) + L_4 * \cos(\theta_1) * \cos(\theta_{234})) + (\sin(\theta_1)) * (L_3 * \sin(\theta_1) * \cos(\theta_{23}) + L_2 * \sin(\theta_1) * \cos(\theta_2) + L_4 * \sin(\theta_1) * \cos(\theta_{234})))$$

$$\dot{w_x} = \sin(\theta_1) * ((\dot{\theta_2}) + (\dot{\theta_3}) + (\dot{\theta_4}))$$

$$\dot{w_y} = (-\cos(\theta_1)) * ((\dot{\theta_2}) + (\dot{\theta_3}) + (\dot{\theta_4}))$$

$$\dot{w_z} = (\dot{\theta_1}).$$

3.2. Parallel Manipulator Kinematic Analysis

The Kinematic Analysis of the Linear Delta Robot can be different according to motor position such as on the upper position of the base frame or on the lower position on the base frame and it also depends on the number of link(s) which is connected to the end effector.

3.2.1. Inverse Kinematics

In the Kinematic Analysis of the Linear Delta Robot, it is logical to start with inverse kinematics since it is easier than forward kinematics. The robot parameters are given below to realize the related calculations. See Figure 3.2.

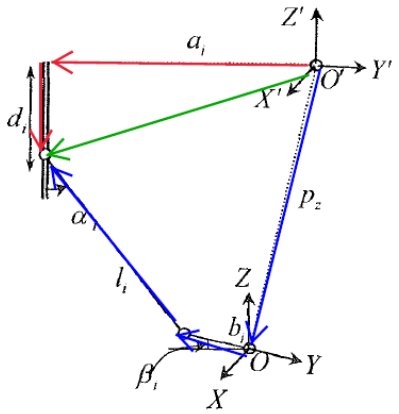

Figure III-2 *Loop closure vectors of the Linear Delta Robot*

d_i: *Linear displacement of the* i^{th} *linear motor in the vertical direction*

l_i: *The length of the* i^{th} *link.*

a_i: *Distance between base frame and location of the* i^{th} *linear motor.*

b_i: *Distance between center of moving platform and* i^{th} *mechanical link.*

p_z: *The movable vertical distance of moving platform.*

α_i: *Rotating angle of the* i^{th} *mechanical link.*

β_i: *Orientation angle of moving platform with* i^{th} *mechanical link*

The Loop Closure Equation can be obtained as following,

$$e_i + d_i - p = l_i$$

where, i =1, 2, 3 also, $e_i = (a_i - b_i)$. Then linear displacement can be found as,

$$d_i = P_z \pm \sqrt{-e_i^2 - P_x^2 - P_y^2 + 2e_{i_x}P_x + 2e_{i_y}P_y + l_i^2}.$$

3.2.2. Forward Kinematics

The purpose of the forward kinematic analysis of the Linear Delta Robot is that determining the end effector position and orientation of the robot. In addition of that, the angular displacement and angular velocities of the end effector would be zero since there is no capability of the robot to do orientation.

As it is mentioned before, the forward kinematics of the parallel manipulators are much more complex than serial manipulators. Therefore, after some complex and taking a long time of calculations the end effector positions P_x, P_y and P_z can be found as below.

$$\boldsymbol{P_x = \frac{\Delta e_{2y}(k_3 - P_z(\Delta e_{3z} + \Delta d_3)) - \Delta e_{3y}(k_2 - P_z(\Delta e_{2z} + \Delta d_2))}{\Delta e_{3y}\Delta e_{2x} - \Delta e_{3x}\Delta e_{2y}}}$$

$$\boldsymbol{P_y = \frac{\Delta e_{2x}(k_3 - P_z(\Delta e_{3z} + \Delta d_3)) - \Delta e_{3x}(k_2 - P_z(\Delta e_{2z} + \Delta d_2))}{\Delta e_{3y}\Delta e_{2x} - \Delta e_{3x}\Delta e_{2y}}}$$

$$\boldsymbol{P_z} = d_1 - \sqrt{-\boldsymbol{e_1}^2 - \boldsymbol{P_x}^2 - \boldsymbol{P_y}^2 + \boldsymbol{l_1}^2 + 2\,\boldsymbol{e_{1x}P_x} + 2\boldsymbol{e_{1y}P_y}}$$

where,

$$\Delta e_2 = (e_2 - e_1)$$
$$\Delta e_3 = (e_3 - e_1)$$

$$\Delta d_2 = (d_2 - d_1)$$
$$\Delta d_3 = (d_3 - d_1)$$

$$k_2 = P_x \Delta e_{2x} + P_y \Delta e_{2y} + P_z \Delta e_{2z} + P_z \Delta d_2$$
$$k_3 = P_x \Delta e_{3x} + P_y \Delta e_{3y} + P_z \Delta e_{3z} + P_z \Delta d_3.$$

3.2.3. Jacobian

In the velocity analysis of the Linear Delta Robot, linear velocities of the end effector are calculated and angular velocities of the end effector are zero since there is no motion around the axis of the end effector. Then, the matrix equation of the velocities is given with the Jacobian Matrix and it occurs three by three since there are only linear velocities and three prismatic joints and the equation can be shown as following,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \\ J \\ \\ \end{bmatrix}_{3x3} * \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \end{bmatrix}$$

and the Jacobian matrix can be found as following.

$$J = \begin{bmatrix} \frac{(e_{1x}-P_x)}{\sqrt{-e_1{}^2-P_x{}^2-P_y{}^2+2e_{1x}P_x+2e_{1y}P_y+l_1{}^2}} & \frac{(e_{1y}-P_y)}{\sqrt{-e_1{}^2-P_x{}^2-P_y{}^2+2e_{1x}P_x+2e_{1y}P_y+l_1{}^2}} & 1 \\ \frac{(e_{2x}-P_x)}{\sqrt{-e_2{}^2-P_x{}^2-P_y{}^2+2e_{2x}P_x+2e_{2y}P_y+l_2{}^2}} & \frac{(e_{2y}-P_y)}{\sqrt{-e_2{}^2-P_x{}^2-P_y{}^2+2e_{2x}P_x+2e_{2y}P_y+l_2{}^2}} & 1 \\ \frac{(e_{3x}-P_x)}{\sqrt{-e_3{}^2-P_x{}^2-P_y{}^2+2e_{3x}P_x+2e_{3y}P_y+l_3{}^2}} & \frac{(e_{3y}-P_y)}{\sqrt{-e_3{}^2-P_x{}^2-P_y{}^2+2e_{3x}P_x+2e_{3y}P_y+l_3{}^2}} & 1 \end{bmatrix}$$

Therefore, the velocity equation can rewrite as below.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} \frac{e_{1x}-P_x}{d_1-P_z} & \frac{e_{1y}-P_y}{d_1-P_z} & 1 \\ \frac{e_{2x}-P_x}{d_2-P_z} & \frac{e_{2y}-P_y}{d_2-P_z} & 1 \\ \frac{e_{3x}-P_x}{d_3-P_z} & \frac{e_{3y}-P_y}{d_3-P_z} & 1 \end{bmatrix} \begin{bmatrix} \dot{d}_1 \\ \dot{d}_2 \\ \dot{d}_3 \end{bmatrix}$$

At the end, the linear velocities of the end effector are found as following,

$$\dot{x} = \dot{d}_1 \left[\frac{e_{1x}-P_x}{d_1-P_z}\right] + \dot{d}_2 \left[\frac{e_{1y}-P_y}{d_1-P_z}\right] + \dot{d}_3$$

$$\dot{y} = \dot{d}_1 \left[\frac{e_{2x}-P_x}{d_2-P_z}\right] + \dot{d}_2 \left[\frac{e_{2y}-P_y}{d_2-P_z}\right] + \dot{d}_3$$

$$\dot{z} = \dot{d}_1 \left[\frac{e_{3x}-P_x}{d_3-P_z}\right] + \dot{d}_2 \left[\frac{e_{3y}-P_y}{d_3-P_z}\right] + \dot{d}_3.$$

IV. DETAIL MECHANICAL DESIGN

In the Detail Mechanical Design step, the serial and parallel manipulators were designed on the SolidWorks CAD program with consider Structural Synthesis of the manipulators. The detail contents of the designs as following.

4.1. Serial Manipulator

The serial manipulator is designed in detail considering concept design. In this direction, main final structure of the robot is shown in the Figure 4.1.

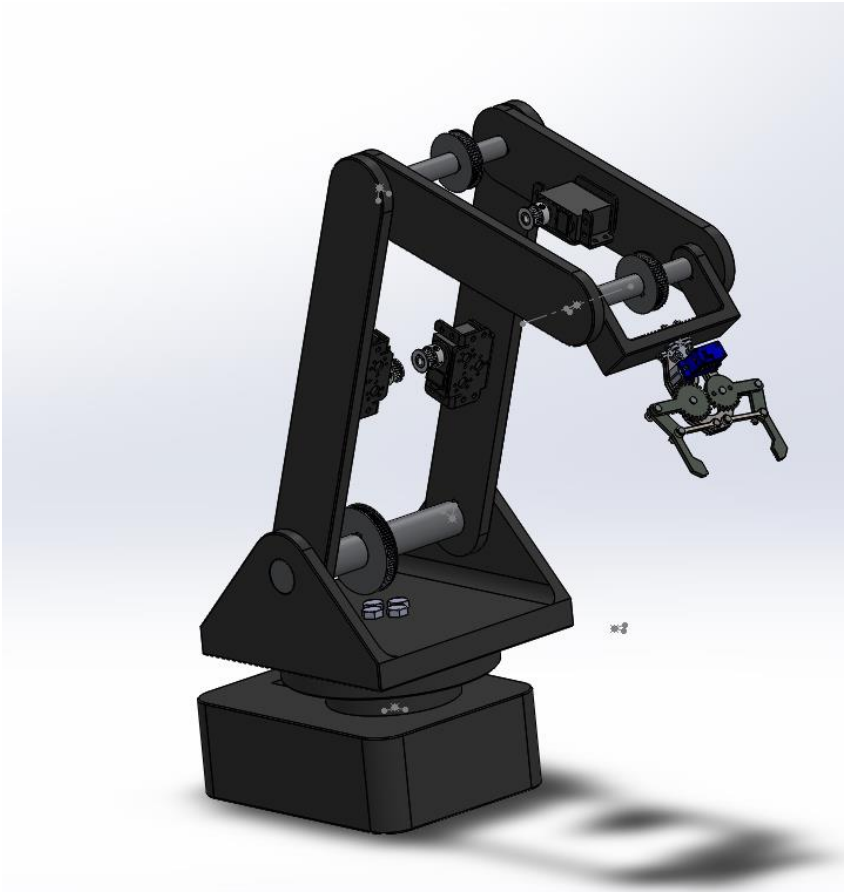

Figure IV-1 *Detail mechanical design of the serial manipulator*

There are four revolute joint and in addition of that there is one more mobility on the end effector, therefore the mobility is provided according to concept design. See Figure 4.2.

Figure IV-2 *Different perspective of the serial manipulator*

The lengths of the links were determined as 153 mm for first link that starts from bottom of base pulley to first joint, 300 mm for second link, 200 mm for third link and 50 mm for last link. At the base of the robot, a servo motor is placed as perpendicular to the base to rotate the pulley as first joint. The pulley is fixed on the base with M8x120 mm hex bolt and the motor is fixed by using M4x18 mm hex bolt. See Figure 4.3.

Figure IV-3 *Bottom view of the base*

Three bearing is used on the first revolute joint which rotates the robot around Z-axis. These bearings can be listed as two 51210 thrust bearing and one 61808 ball bearing. The thrust bearings are used for reducing the axial load and the ball bearing is used to reduce friction and able to motion as smooth as possible. See Figure 4.4. and Figure 4.5.

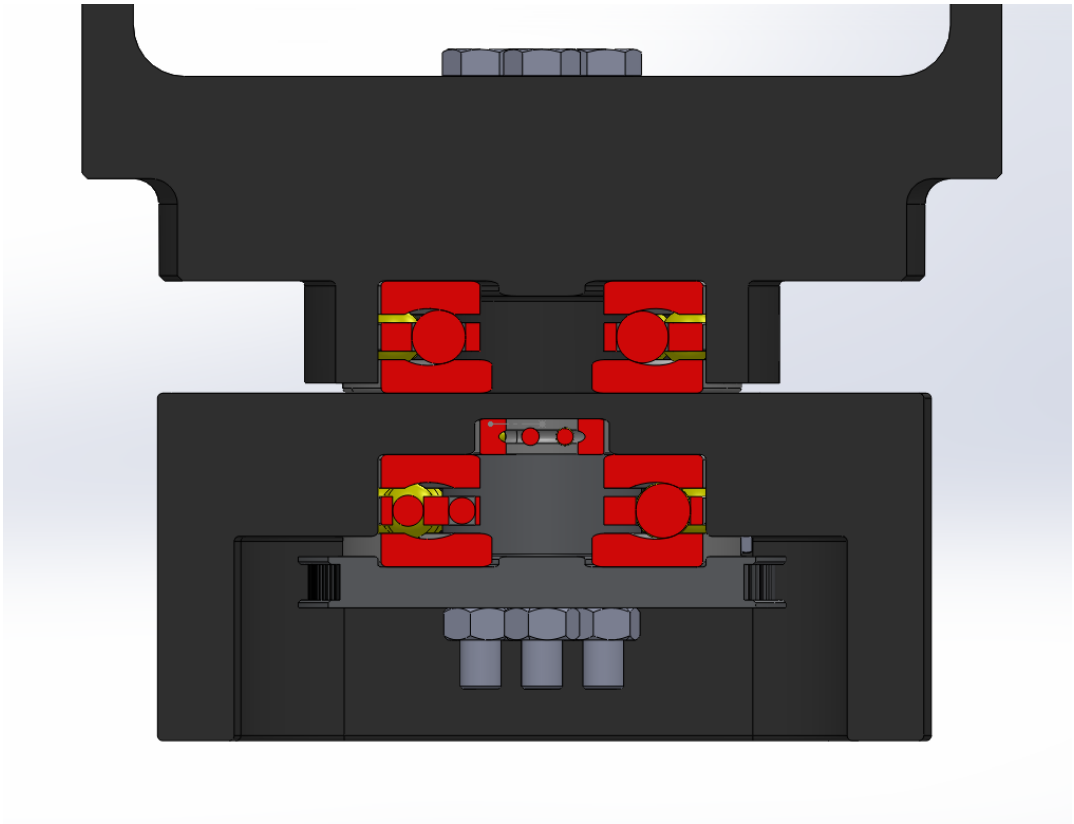

Figure IV-4 *Cross-sectional view of the first joint*

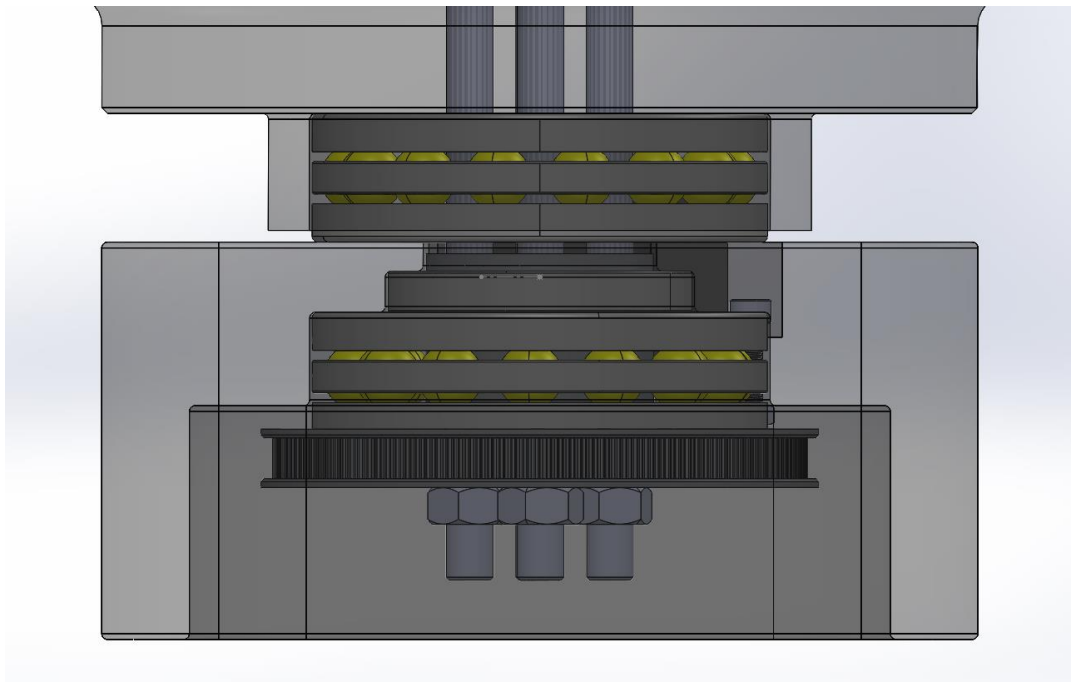

Figure IV-5 *Transparent view of the first joint*

In the first link a shaft is form a bearing with two 61905 ball bearings. A pulley and two parallel links are fixed on the shaft with bulges. See Figure 4.6 and Figure 4.7.

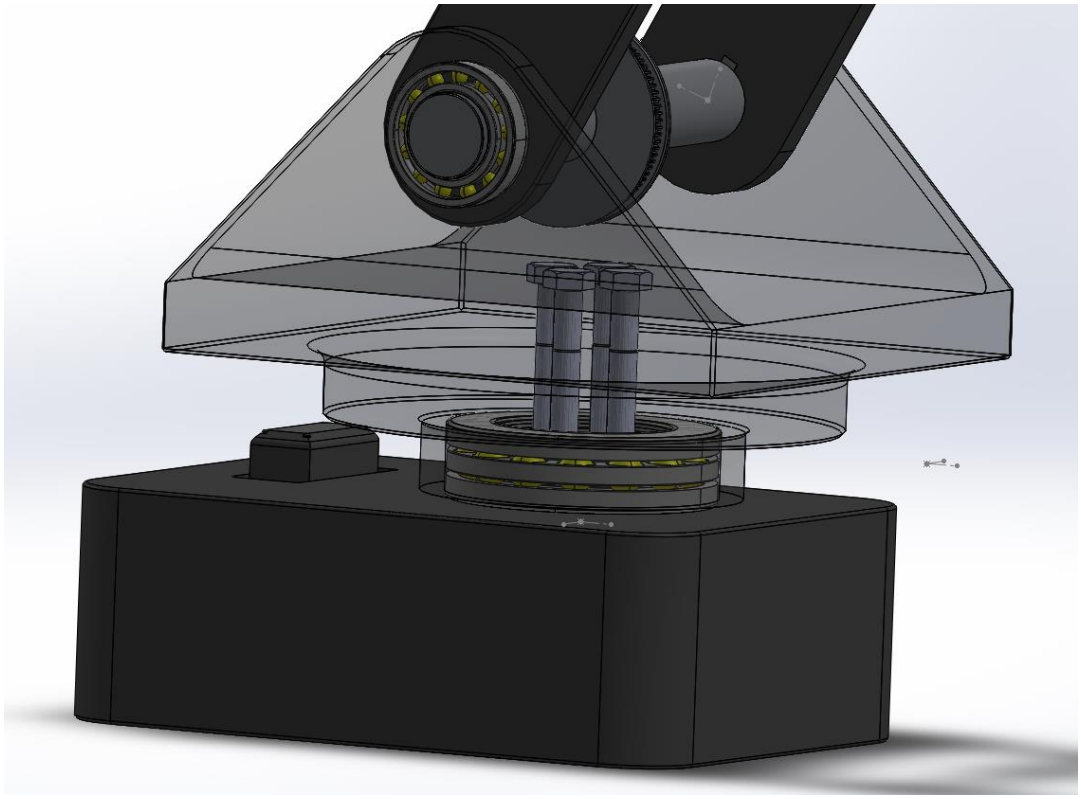

Figure IV-6 *Bearings on the shaft*

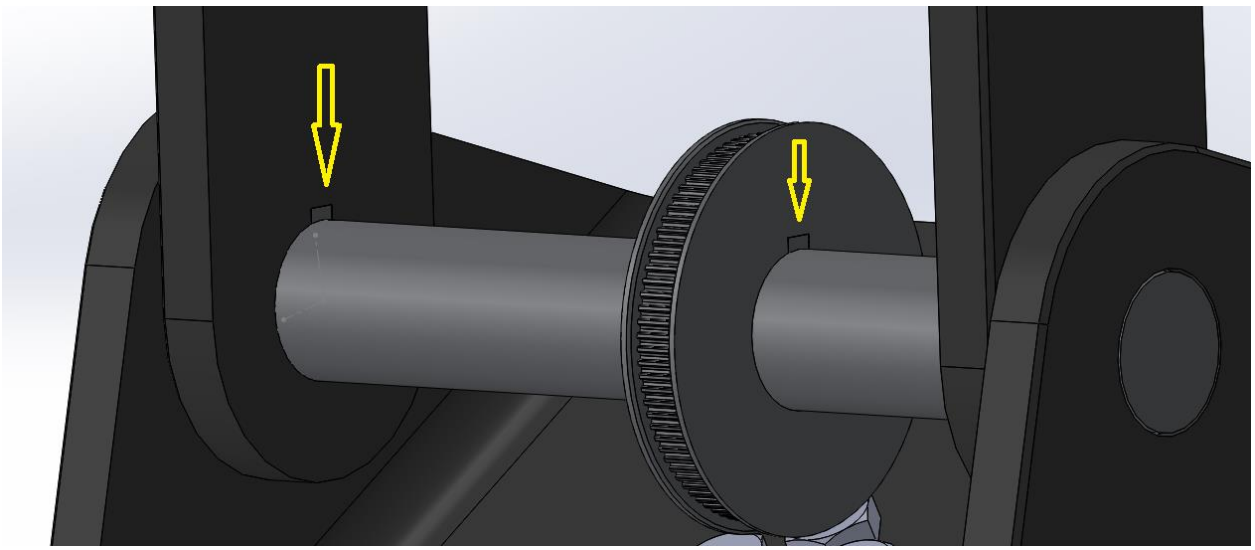

Figure IV-7 *Bulges on the shaft*

The rest of the actuators, servo motors are placed on the links. See Figure 4.8.

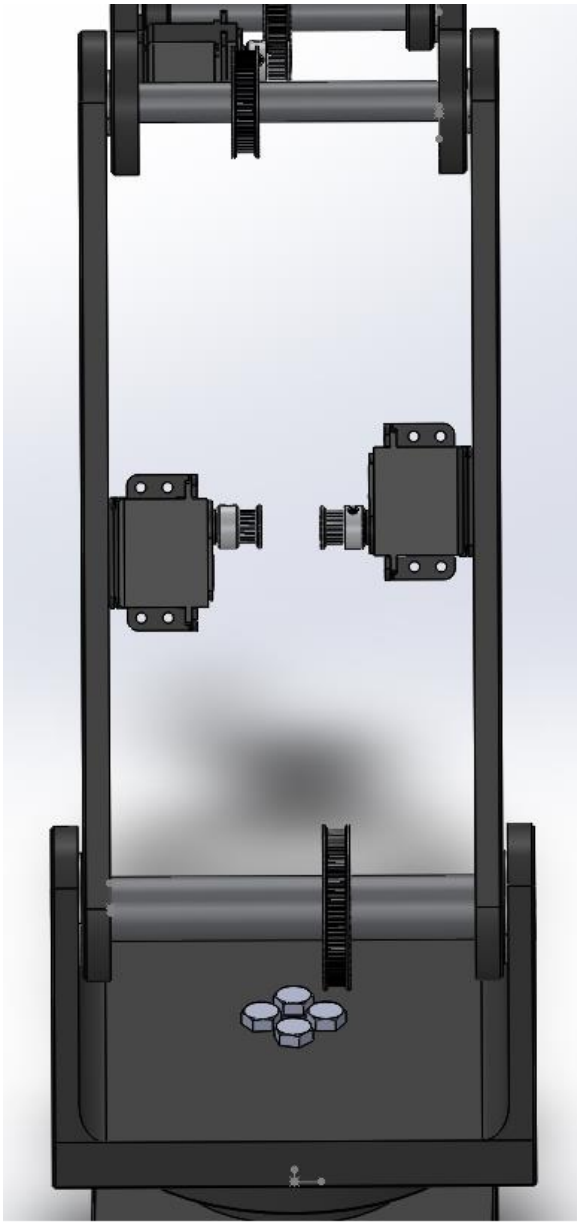

Figure IV-8 *Opposite view of the Serial Manipulator*

Finally, the end effector is placed on the last link and it has a gear mechanism which actuated via a servo motor as following. See Figure 4.9.

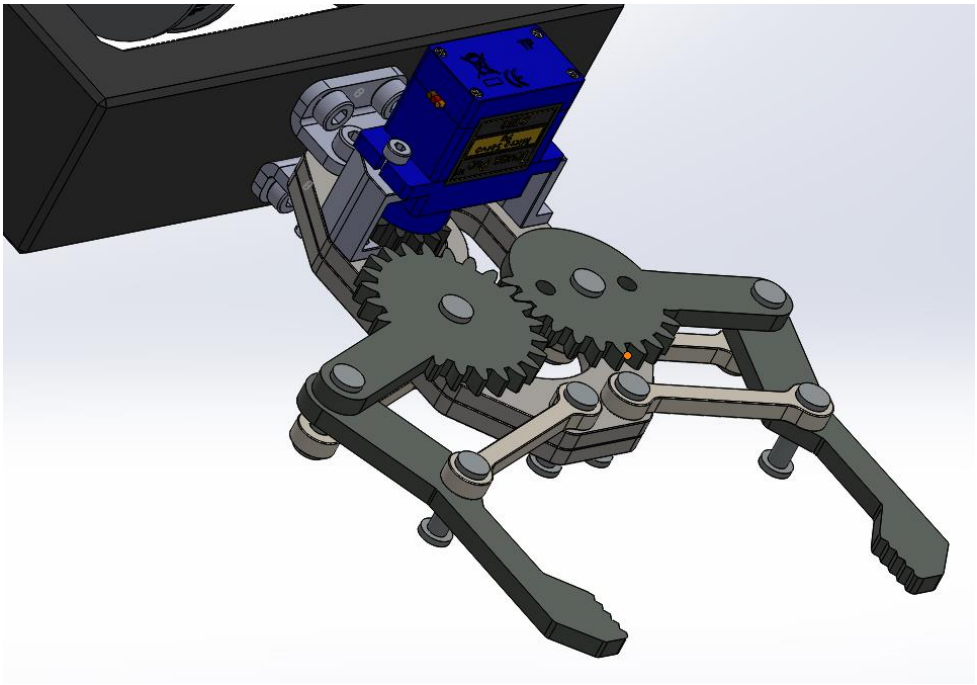

Figure IV-9 *End effector*

4.2. Parallel Manipulator

In the detail mechanical design of the parallel manipulator, the robot is designed in the same way with the serial manipulator in terms of concept design by SolidWorks. The parallel manipulator, Linear Delta Robot, is built on Aluminum sigma profile. In addition of that, the lengths of the links were determined as 130 mm and the strokes were determined as 80 mm. See Figure 4.10.

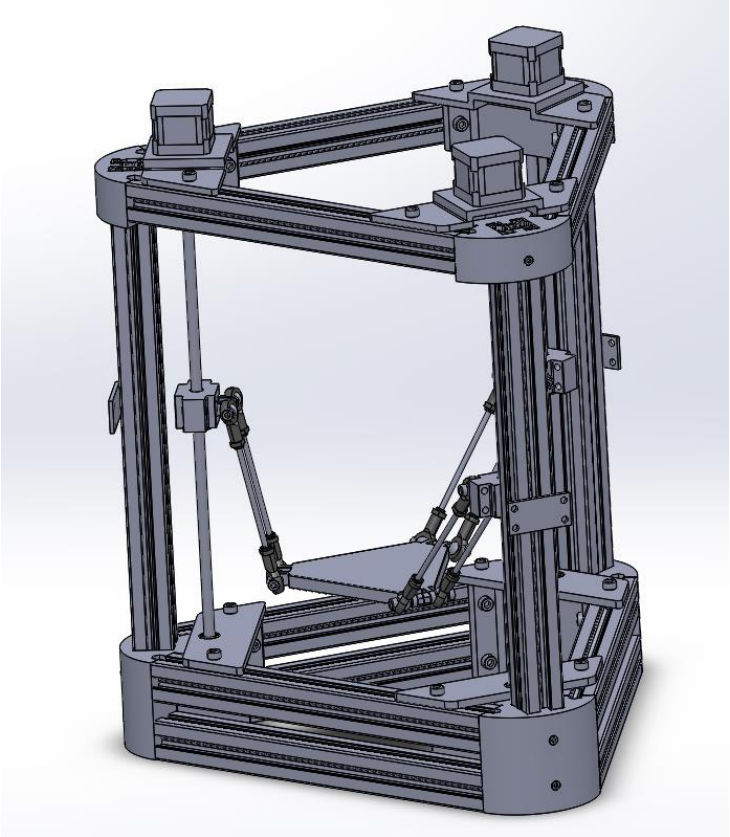

Figure IV-10 *Detail mechanical design of the Linear Delta Robot*

The kinematic analysis of the Linear Delta Robot change in terms of the actuator position such as upper or lower of the base frame. In this direction, at the kinematic analysis of the Linear Delta Robot is calculated according to the condition that the actuators are on the upper position of the base frame. Therefore, the detail mechanical design of the Linear Delta Robot is designed according to this condition. Three Nema-17 Step Motors were chosen as actuator. The step motors have lead screws as shaft and three prismatic joints move on these lead screws. See Figure 4.11 and Figure 4.12.

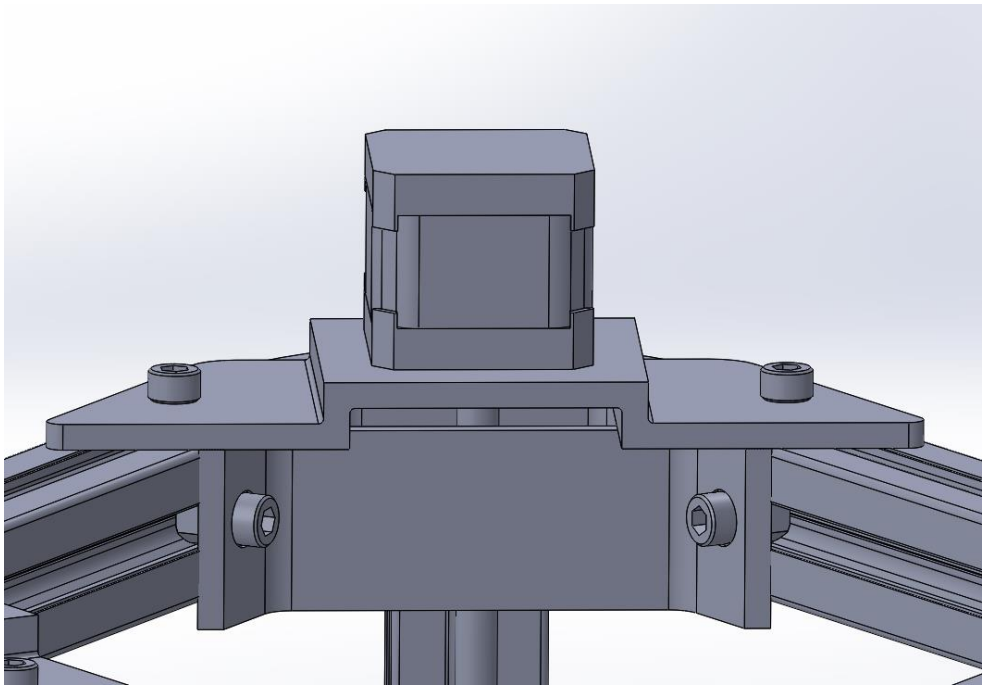

Figure IV-11 *Step motor on the top of base frame*

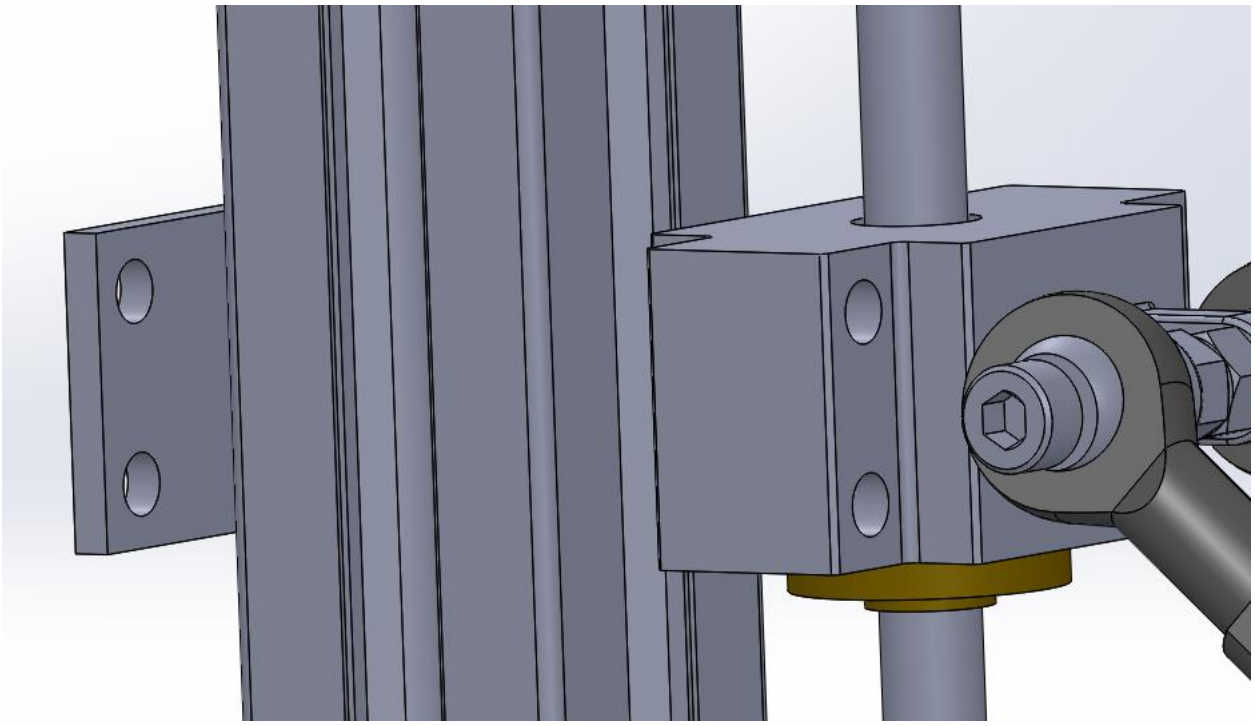

Figure IV-12 *Prismatic joint on the Linear Delta Robot*

The prismatic joints and end-effector are connected to each other with six link that have twelve spherical joints. The spherical joints are fixed to the prismatic joints with M5x20 mm Hex Bolts. See Figure 4.13.

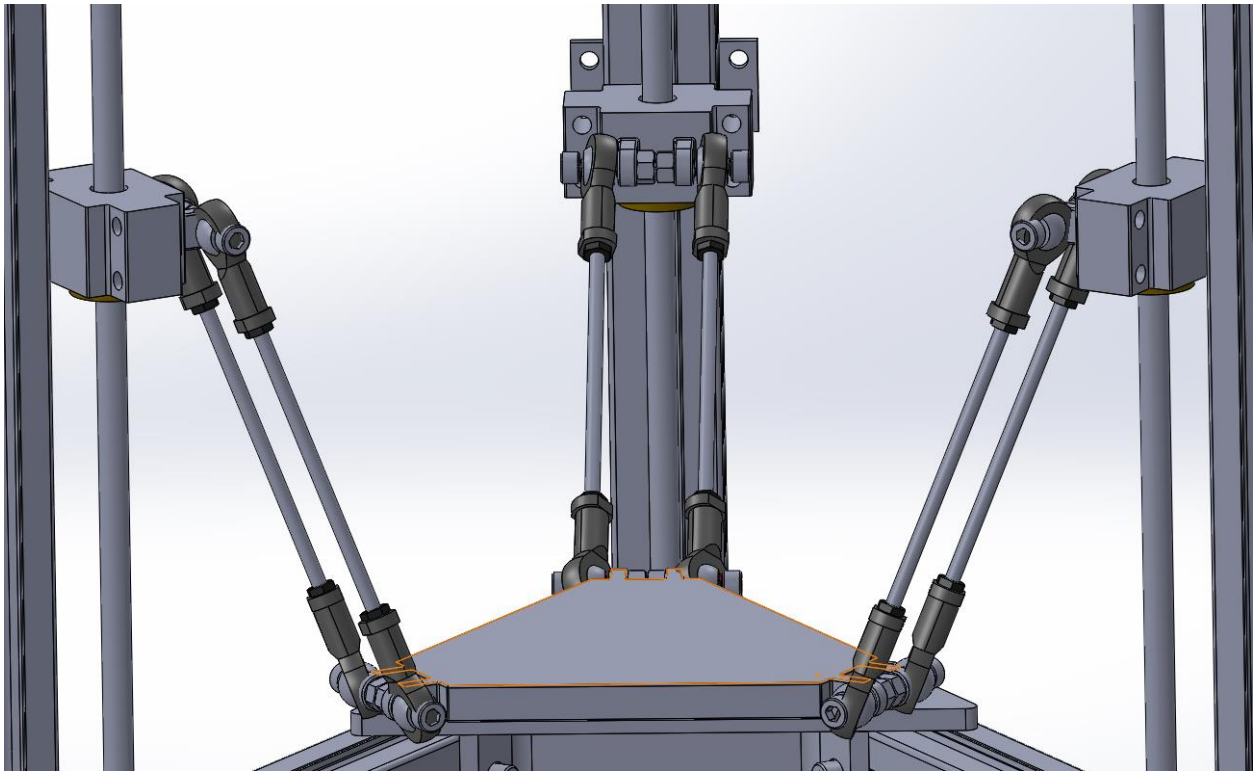

Figure IV-13 *Links and end effector connections with spherical joints*

At the lower part of the base frame, the lead screws are form of bearing with W618.8 ball bearings and Aluminum sigma profiles are assembled with each other with M6x10 mm hex bolt. See Figure 4.14.

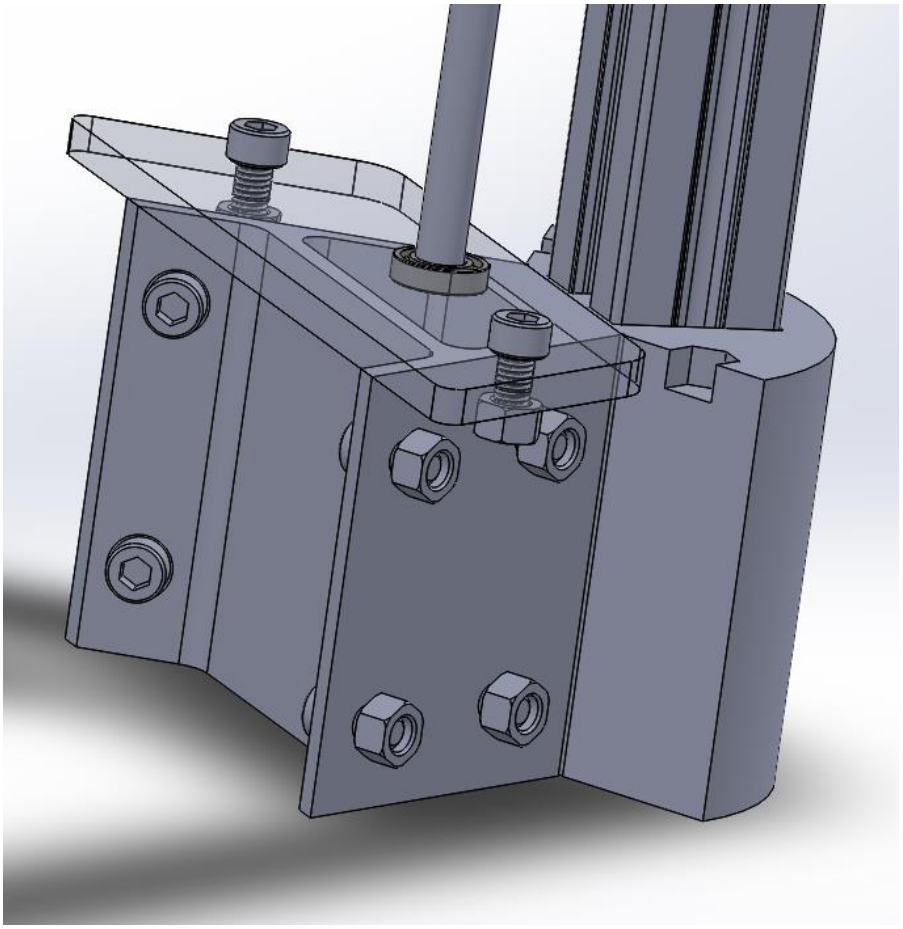

Figure IV-14 *Additional assembly parts*

V. SIMULATION

Simulation is a useful tool to see the physical world reactions of the systems in the virtual environment which is able to adjust the specific parameters. Robotics is one of the most popular research area in nowadays, in addition of that production cost of the robots is also high. Therefore, the simulation of the robots is a critical to see the results of the specific tasks before the production.

5.1. Serial Manipulator

As it is mentioned before, the serial and parallel manipulators were designed by SolidWorks and to transfer the designs to ROS which was planned to be used in the simulation, the *sw_urdf_exporter* package is used. The relation of the links and joints, joint

types, velocities of the joints, range of the joints as lower and upper, efforts were specified. The range of the joints were determined as -90 degree lower limit and +90 degree upper limit for first joint, -70 degree lower limit and +110 degree upper limit for second joint, -360 degree lower limit and -130 degree upper limit for third joint, -140 degree lower limit and +40 degree for upper limit for last joint. In addition of these, the inertias were calculated as automatically by SolidWorks. See Figure 5.1.

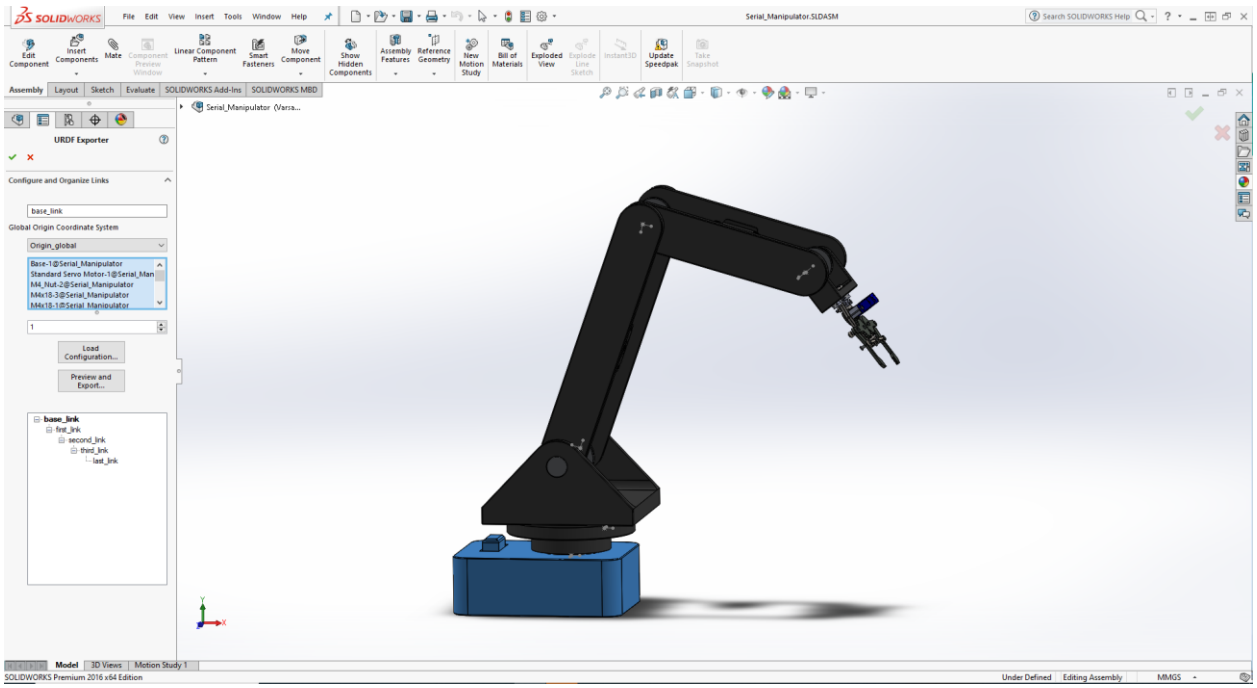

Figure V-1 *Export URDF step of the serial manipulator*

The package that created by *sw_urdf_exporter* was transferred to the ROS, and the final context of the package is as following. The *urdf* folder of the package includes the *four_dof_manipulator.urdf* file which contains the parameters of the serial manipulator or in the other words it is a XML format for represent the robot model. The *package.xml* file includes the package dependencies and package information and *CMakeLists.txt* file includes specific message, topic or service data, script paths, etc… See Figure 5.2.

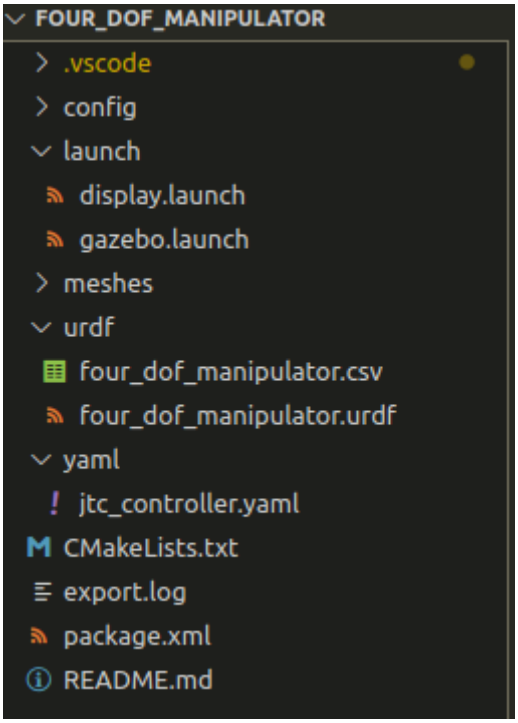

Figure V-2 *Content of the package*

5.1.1. RViz

In the *launch* folder, there are two launch files to run the robot in the Simulators. The *display.launch* file is used to simulate the robot in RViz and to execute the launch file following command should be run from terminal.

$roslaunch four_dof_manipulator display.launch

After the execute the mentioned command above, the robot should be added, fixed frame should be selected as *base_link* and in addition of these the color of the background can be adjusted to see the robot clearly. See Figure 5.3.

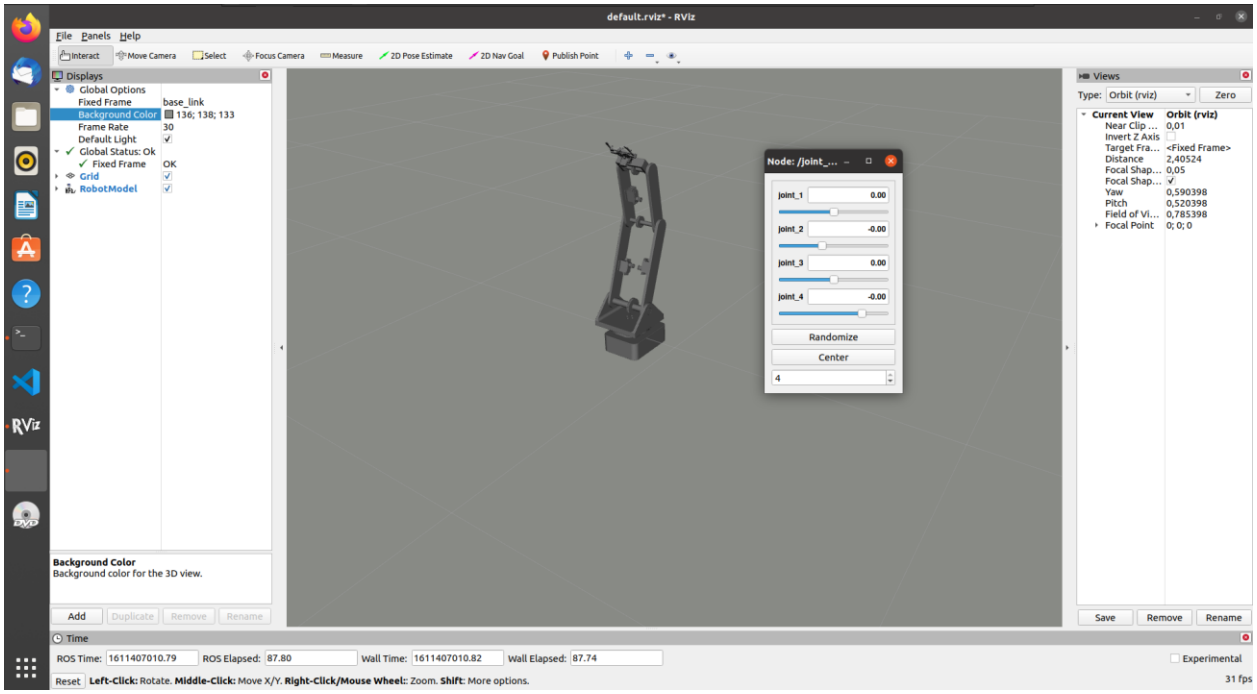

Figure V-3 *Serial manipulator in RViz*

Then, the serial manipulator can be manipulated via *joint_state_publisher* which is a GUI to control the joints of the serial manipulator. See Figure 5.4.

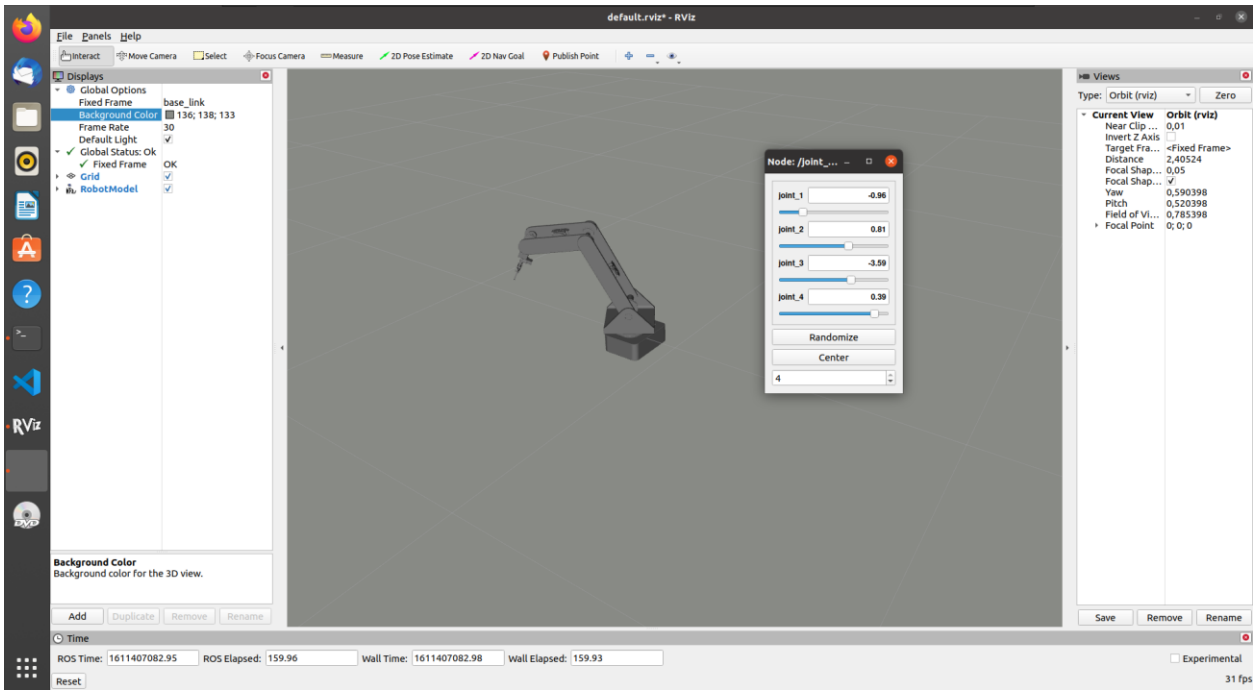

Figure V-4 *Serial manipulator simulation in RViz*

5.1.2. Gazebo

The other launch file in the launch folder is the *gazebo.launch* is used to simulate the serial manipulator in the Gazebo and to execute the serial manipulator in the Gazebo the following command should be executed from terminal.

$roslaunch four_dof_manipulator gazebo.launch

After the execution the command above, the Gazebo simulator works as following. See Figure 5.5.

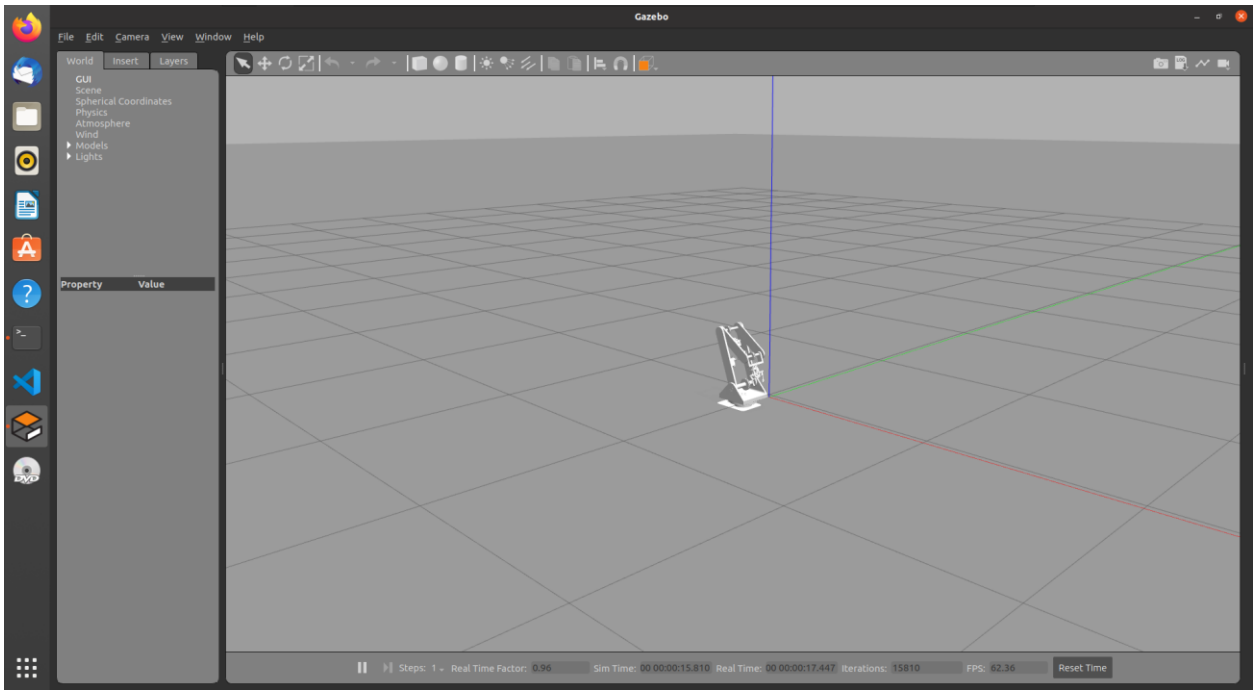

Figure V-5 *Serial manipulator in Gazebo*

In the *yaml* folder, the *jtc_controller.yaml* file contains the controller which is *position_controller/JointTrajectoryController* and thanks to this controller it is possible to control the joints with specific joint angles in terms of radians. The following command should be executed from terminal to operate the serial manipulator via mentioned controller.

$rostopic pub /arm_controller/command trajectory_msgs/JointTrajectory '{joint_names: ["joint_1","joint_2","joint_3","joint_4"],points:[{positions: [0.98,0.83,-3.62,0.19],time_from_start:[1,0]}]}' -1

The figures of the simulation are as following. See Figure 5.6 and Figure 5.7.

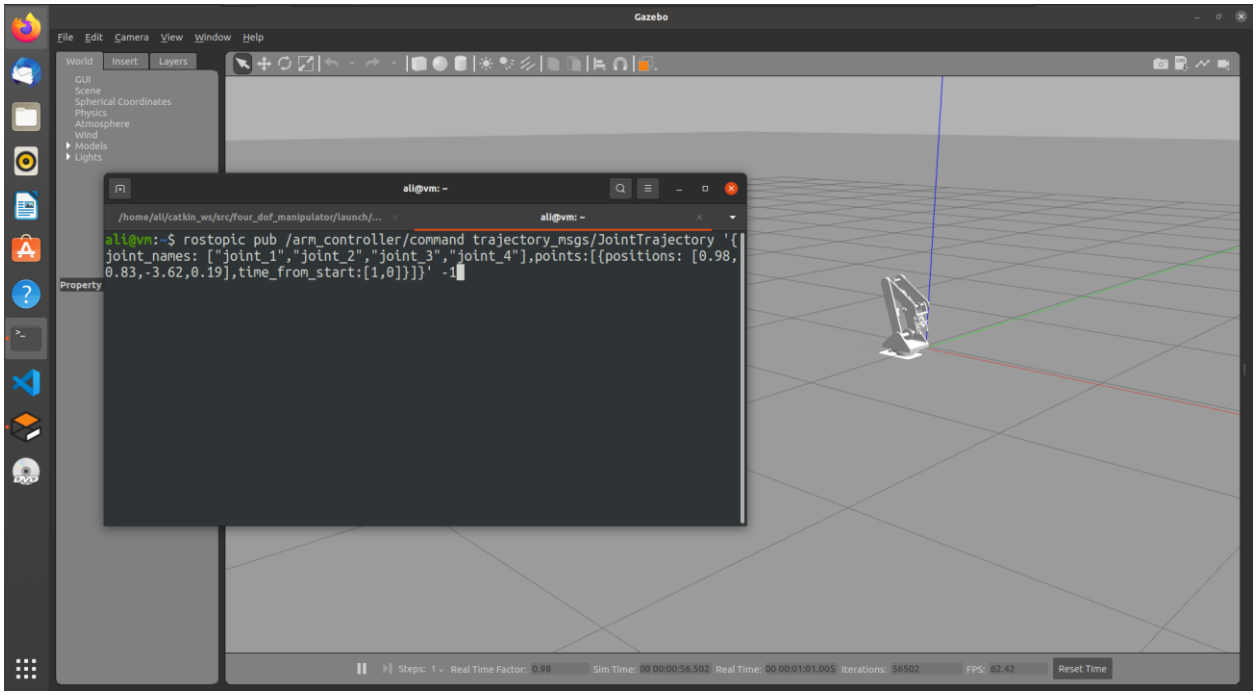

Figure V-6 *Sending command to serial manipulator in Gazebo*

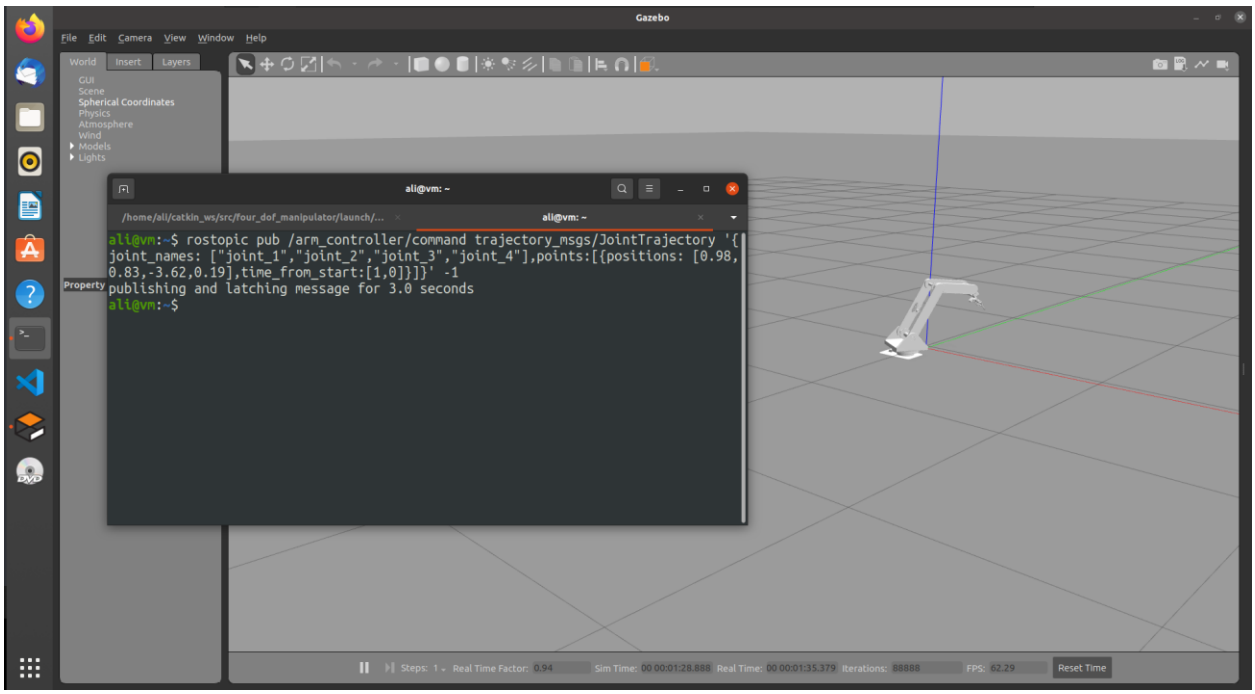

Figure V-7 *Serial manipulator simulation in Gazebo*

5.1.3. MoveIt

The RViz and Gazebo simulators executed to realize the forward kinematics up to this point. In this direction, the MoveIt simulator is used to realize the inverse kinematics and to execute the MoveIt firstly, the *setup_assistant.launch* file should be executed to create the MoveIt package for the serial manipulator. The final figure of the serial manipulator in the MoveIt setup assistant is as following. See Figure 5.8.

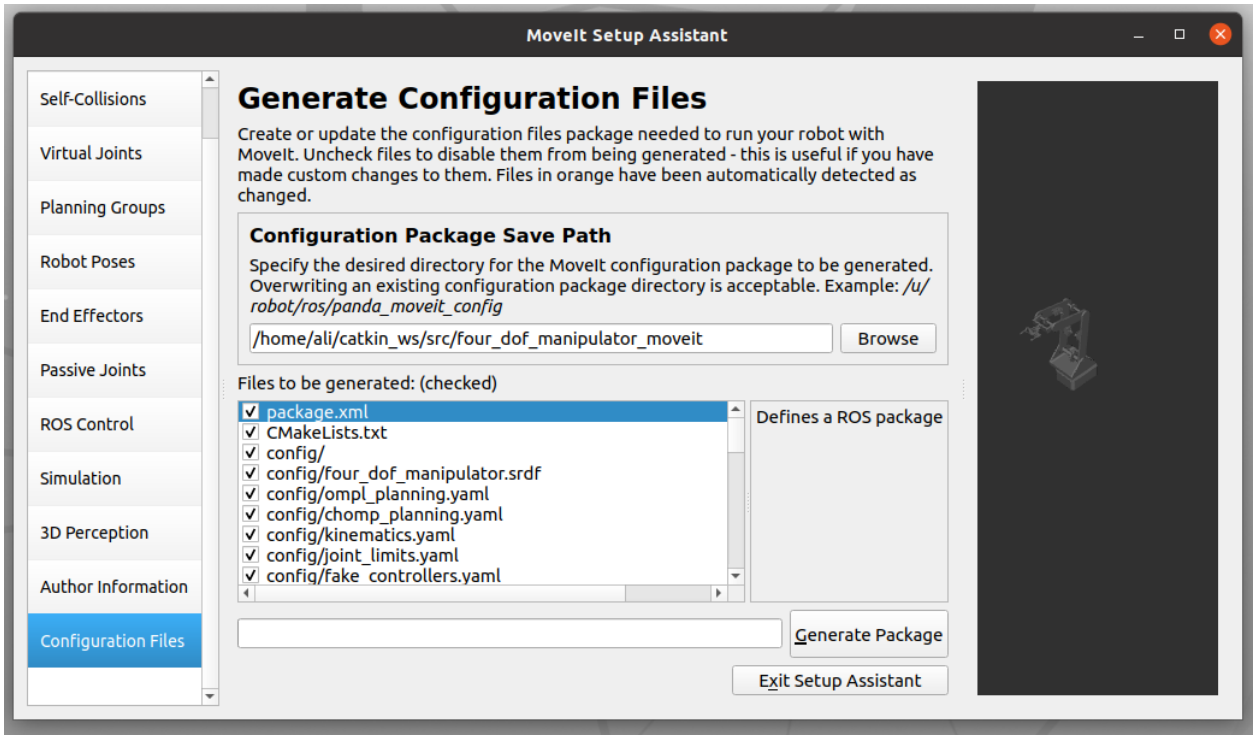

Figure V-8 *MoveIt setup assistant with serial manipulator*

After the setup assistant completed, the *four_dof_manipulator_moveit* package is ready to execute. The following command can be executed from terminal to simulate the serial manipulator in MoveIt.

$roslaunch four_dof_manipulator_moveit demo.launch

The simulation of the serial manipulator in MoveIt is as following. See Figure 5.9 and Figure 5.10.

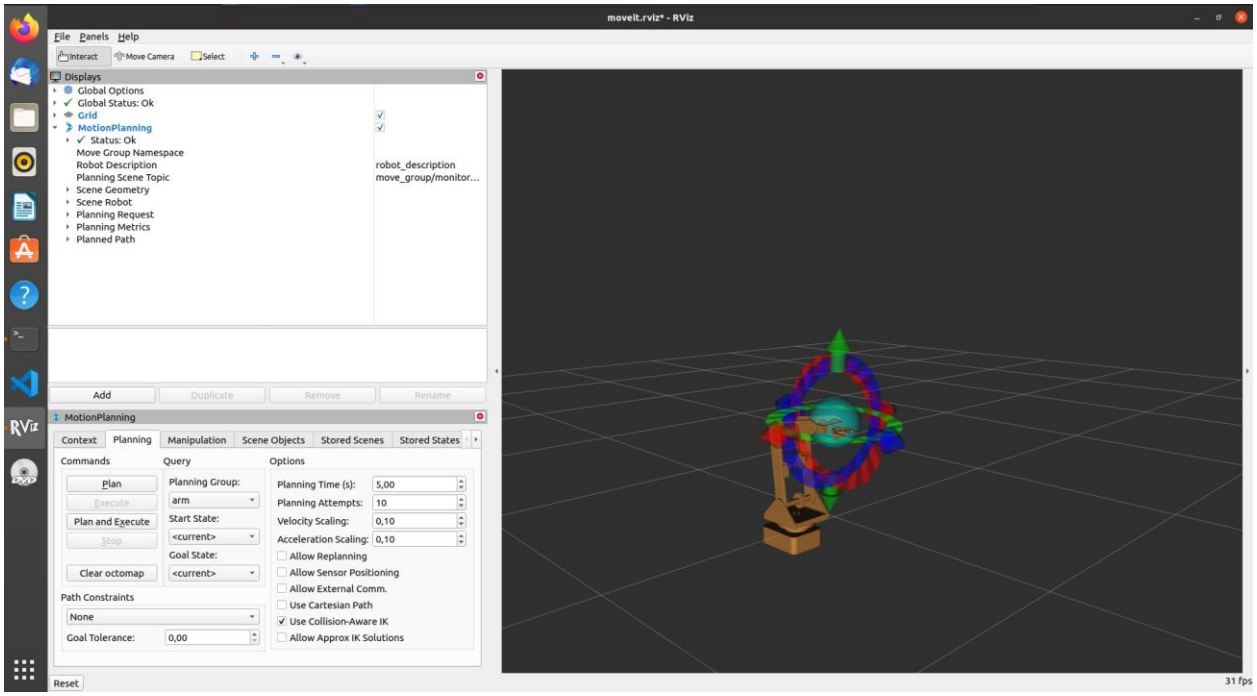

Figure V-9 *Serial manipulator in MoveIt*

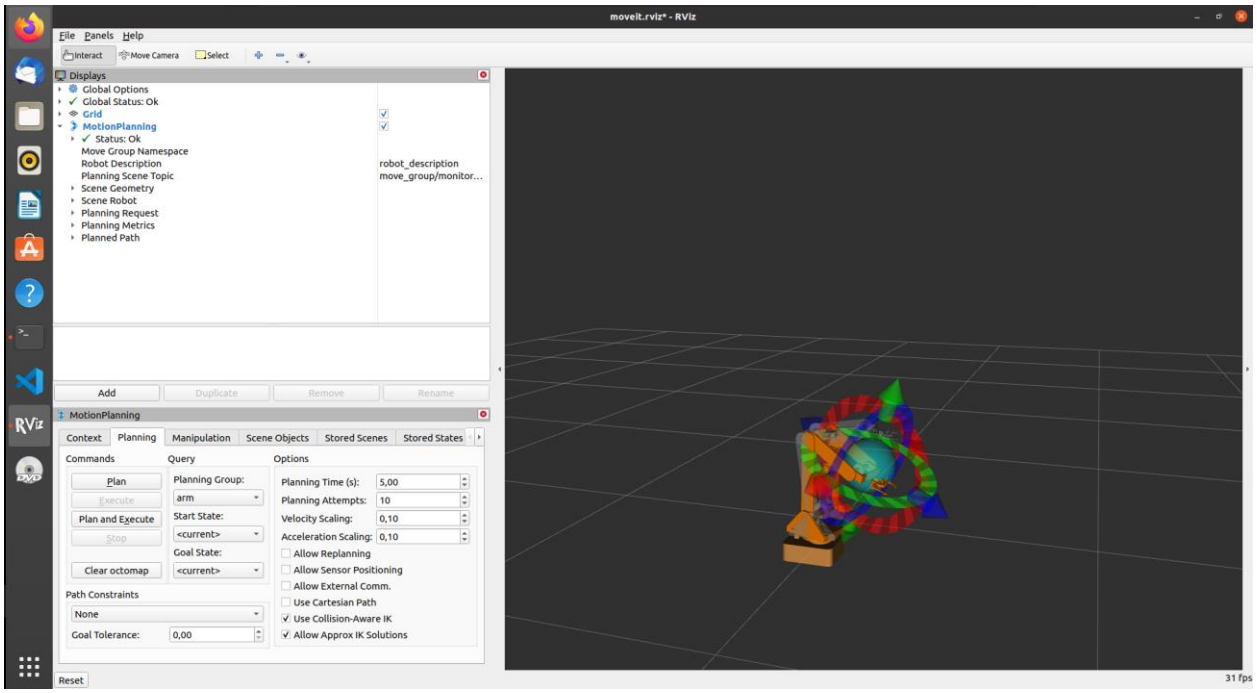

Figure V-10 *Serial manipulator simulation in MoveIt*

5.2. Parallel Manipulator

The simulation of the parallel manipulator was realized in SolidWorks due to limited time. The simulation in ROS will realize in the next semester.

5.2.1. SolidWorks

The simulation was realized via Motion Study of the SolidWorks. The actuators were added as linear motor and the data points in terms of distance and times were selected as desired. See Figure 5.11.

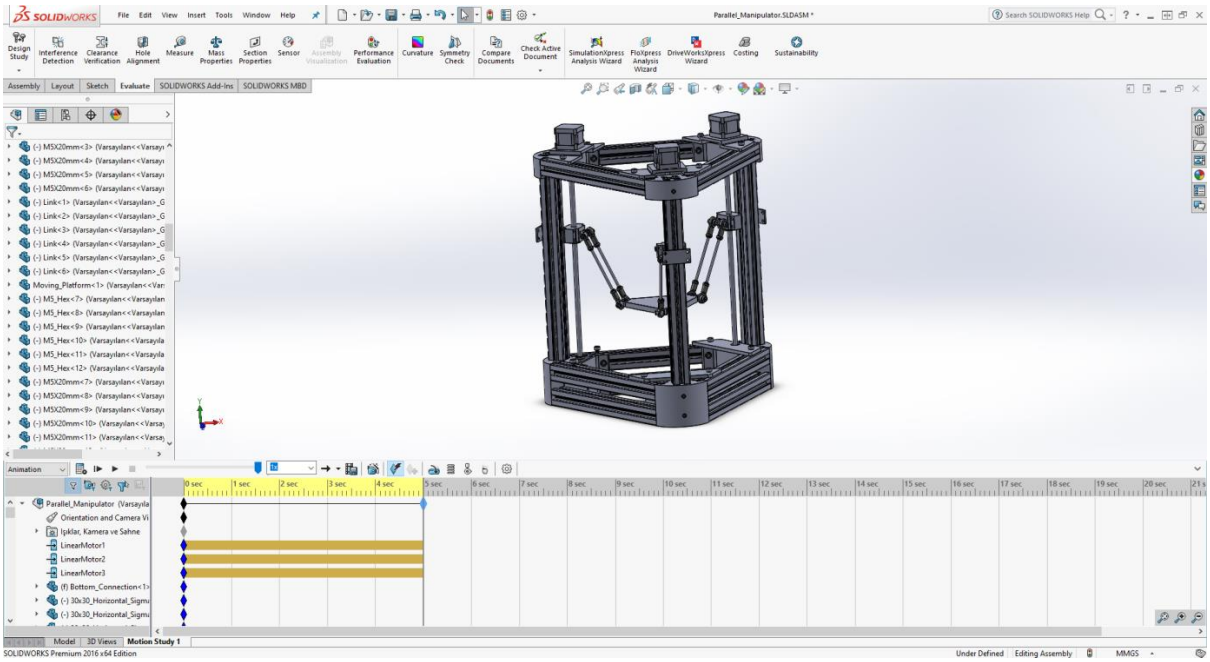

Figure V-11 *Simulation of the parallel manipulator*

The results of the simulation of the linear delta robot as following in terms of the position and velocity graphics. See Figure 5.12.

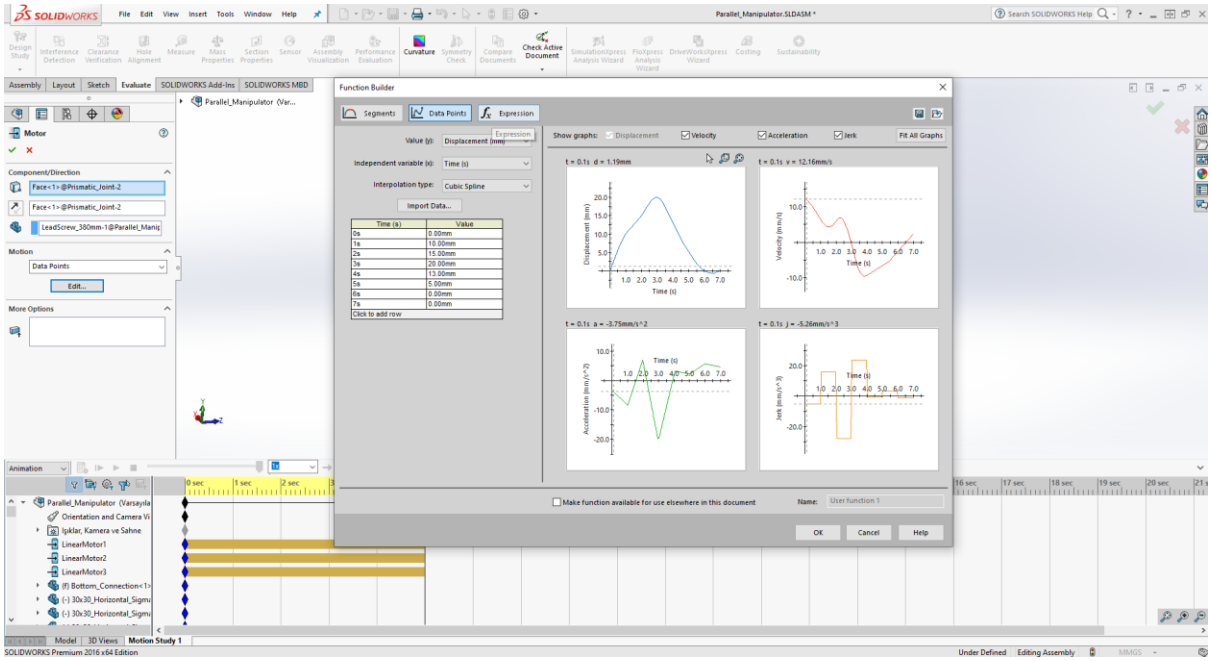

Figure V-12 *Simulation results of the parallel manipulator*

VI. DYNAMIC ANALYSIS

The Dynamic Analysis is necessity to determine forces, moments and torques acting on the links. At the end of the analysis, the founded torques can be used for selecting appropriate actuators. In the Dynamic Analysis, mainly, there are two methods as Lagrangian dynamics and Newton-Euler formulation. The Lagrangian dynamics is quite effective for robots with simple structures, e.g., with three or fewer degrees of freedom. On the other hand, the Newton-Euler formulation makes possible recursive algorithms for both inverse and forward dynamics that can also be integrated into closed-loop systems.

In this part of project, the Dynamic Analysis of serial manipulator has been completed by Newton-Euler formulation. It is divided into two parts as forward computation and backward computation. In the forward computation, angular velocity propagation, angular acceleration propagation, linear velocity propagation, linear acceleration propagation, linear acceleration of the center of mass and acceleration of gravity were determined. In the

backward computation, net forces, net moments and torques that exerted on the links were determined.

6.1. Recursive Newton-Euler Formulation

The constants which are masses and mass moment of inertias were obtained from URDF file and center of masses were obtained from SolidWorks. The rotation and position matrices were already obtained at the Kinematic Analysis.

6.1.1. Forward Computation

In the forward computation, the angular velocities, angular accelerations, linear velocities, linear accelerations, linear acceleration of center of mass are calculated with respect to $\theta, \dot{\theta}, \ddot{\theta}$.

Rotation matrices from homogeneous transformation matrices:

$$R_1 = \begin{bmatrix} cos(\theta_1) & 0 & sin(\theta_1) \\ sin(\theta_1) & 0 & -cos(\theta_1) \\ 0 & 1 & 0 \end{bmatrix}$$

$$R_2 = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 0 \\ sin(\theta_2) & cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_3 = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 \\ sin(\theta_3) & cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_4 = \begin{bmatrix} cos(\theta_4) & -sin(\theta_4) & 0 \\ sin(\theta_4) & cos(\theta_4) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Initial Conditions:

$$w_0 = \dot{w}_0 = v_0 = 0$$

$$z_0 = [0 \quad 0 \quad 1]^T = \begin{bmatrix}0\\0\\1\end{bmatrix}\ ,\ w_0 = \begin{bmatrix}0\\0\\0\end{bmatrix}\ ,\ \dot{w}_0 = \begin{bmatrix}0\\0\\0\end{bmatrix}\ ,\ v_0 = \begin{bmatrix}0\\0\\0\end{bmatrix}\ ,\ \dot{v}_0 = \begin{bmatrix}0\\0\\-9.81\end{bmatrix}$$

6.1.1.1. Angular Velocity Propagation

The general angular velocity propagation formula was represented as following. In this direction, the angular velocities were found for each joint as following.

$$w_{i+1} = R_{i+1}^T\left[w_i + z_0\ .\ \dot{\theta}_{i+1}\right], \text{ for revolute joint}$$

where i = 0,1,2,3

$$w_1 = R_1^T\left[w_0 + z_0\ .\ \dot{\theta}_1\right]$$

$$w_2 = R_2^T\left[w_1 + z_0\ .\ \dot{\theta}_2\right]$$

$$w_3 = R_3^T\left[w_2 + z_0\ .\ \dot{\theta}_3\right]$$

$$w_4 = R_4^T\left[w_3 + z_0\ .\ \dot{\theta}_4\right]$$

6.1.1.2. Angular Acceleration Propagation

The general angular acceleration propagation formula was represented as following. In this direction, the angular accelerations were found for each joint as following. The angular velocities w_1, w_2 and w_3 were already founded in the previous step.

$$\dot{w}_{i+1} = R_{i+1}^T\left[\dot{w}_i + z_0.\ddot{\theta}_{i+1} + w_i\ x\ (z_0.\dot{\theta}_{i+1})\right], \text{ for revolute joint}$$

where i = 0,1,2,3

$$\dot{w}_1 = R_1^T\left[\dot{w}_0 + z_0.\ddot{\theta}_1 + w_0\ x\ \left(z_0.\dot{\theta}_1\right)\right]$$

$$\dot{w}_2 = R_2^T\left[\dot{w}_1 + z_0.\ddot{\theta}_2 + w_1\ x\ \left(z_0.\dot{\theta}_2\right)\right]$$

$$\dot{w}_3 = R_3^T\left[\dot{w}_2 + z_0.\ddot{\theta}_3 + w_2\ x\ \left(z_0.\dot{\theta}_3\right)\right]$$

$$\dot{w}_4 = R_4^T\left[\dot{w}_3 + z_0.\ddot{\theta}_4 + w_3\ x\ \left(z_0.\dot{\theta}_4\right)\right]$$

6.1.1.3. Linear Velocity Propagation

The general linear velocity propagation formula was represented as following. In this direction, the linear velocities were found for each joint as following.

$$v_{i+1} = R_{i+1}^T.v_i + w_{i+1}\ x\ P_{i+1}, \text{ for revolute joint}$$

where *i* = 0,1,2,3

$$v_1 = R_1^T.v_0 + w_1\ x\ P_1$$

$$v_2 = R_2^T.v_1 + w_2\ x\ P_2$$

$$v_3 = R_3^T.v_2 + w_3\ x\ P_3$$

$$v_4 = R_4^T.v_3 + w_4\ x\ P_4$$

6.1.1.4. Linear Acceleration Propagation

The general linear acceleration propagation formula was represented as following. In this direction, the linear accelerations were found for each joint as following. The angular velocities v_1, v_2 and v_3 were already founded in the previous step.

$$\dot{v}_{i+1} = R_{i+1}^T.\dot{v}_i + \dot{w}_{i+1}\ x\ P_{i+1} + w_{i+1}\ x\ (w_{i+1}\ x\ P_{i+1}), \text{ for revolute joint}$$

where *i* = 0,1,2,3

$$\dot{v}_1 = R_1^T.\dot{v}_0 + \dot{w}_1\ x\ P_1 + w_1\ x\ (w_1\ x\ P_1)$$

$$\dot{v}_2 = R_2^T.\dot{v}_1 + \dot{w}_2\ x\ P_2 + w_2\ x\ (w_2\ x\ P_2)$$

$$\dot{v}_3 = R_3^T.\dot{v}_2 + \dot{w}_3\ x\ P_3 + w_3\ x\ (w_3\ x\ P_3)$$

$$\dot{v}_4 = R_4^T . \dot{v}_3 + \dot{w}_4 \; x \; P_4 + w_4 \; x \; (w_4 \; x \; P_4)$$

6.1.1.5. Linear Acceleration of the center of mass

$$a_i = \dot{w}_i \; x \; s_i + w_i \; x \; (w_i \; x \; s_i) + \dot{v}_i$$

$s_i =$ Position of center-of-mass

where *i* = 1,2,3,4

$$a_1 = \dot{w}_1 \; x \; s_1 + w_1 \; x \; (w_1 \; x \; s_1) + \dot{v}_1$$

$$a_2 = \dot{w}_2 \; x \; s_2 + w_2 \; x \; (w_2 \; x \; s_2) + \dot{v}_2$$

$$a_3 = \dot{w}_3 \; x \; s_3 + w_3 \; x \; (w_3 \; x \; s_3) + \dot{v}_3$$

$$a_4 = \dot{w}_4 \; x \; s_4 + w_4 \; x \; (w_4 \; x \; s_4) + \dot{v}_4$$

6.1.1.6. Acceleration of Gravity

$$g = [0 \; 0 \; -9.81 \;] \frac{m}{s^2}$$

6.1.2. Backward Computation

The moment of inertias must be obtained to achieve the force and moment equations. In this direction, moment of inertias was obtained from URDF.

Mass Moment of Inertias:

$$I_i = \begin{bmatrix} (I_i)_{xx} & (I_i)_{xy} & (I_i)_{xz} \\ (I_i)_{yx} & (I_i)_{yy} & (I_i)_{yz} \\ (I_i)_{zx} & (I_i)_{zy} & (I_i)_{zz} \end{bmatrix}$$

where *i* = 1,2,3,4

$$I_1 = \begin{bmatrix} (I_1)_{xx} & (I_1)_{xy} & (I_1)_{xz} \\ (I_1)_{yx} & (I_1)_{yy} & (I_1)_{yz} \\ (I_1)_{zx} & (I_1)_{zy} & (I_1)_{zz} \end{bmatrix}$$

$$I_2 = \begin{bmatrix} (I_2)_{xx} & (I_2)_{xy} & (I_2)_{xz} \\ (I_2)_{yx} & (I_2)_{yy} & (I_2)_{yz} \\ (I_2)_{zx} & (I_2)_{zy} & (I_2)_{zz} \end{bmatrix}$$

$$I_3 = \begin{bmatrix} (I_3)_{xx} & (I_3)_{xy} & (I_3)_{xz} \\ (I_3)_{yx} & (I_3)_{yy} & (I_3)_{yz} \\ (I_3)_{zx} & (I_3)_{zy} & (I_3)_{zz} \end{bmatrix}$$

$$I_4 = \begin{bmatrix} (I_4)_{xx} & (I_4)_{xy} & (I_4)_{xz} \\ (I_4)_{yx} & (I_4)_{yy} & (I_4)_{yz} \\ (I_4)_{zx} & (I_4)_{zy} & (I_4)_{zz} \end{bmatrix}$$

6.1.2.1. Net Force

The Newton second law must be applied to achieve the net force that acting on the links. Related computations were completed as following.

$$F_i = m_i \,.\, a_i$$

where *i* = 1,2,3,4

$$F_1 = m_1 \,.\, a_1$$

$$F_2 = m_2 \,.\, a_2$$

$$F_3 = m_3 \,.\, a_3$$

$$F_4 = m_4 \,.\, a_4$$

6.1.2.2. Net Moment

The net moments acting on the links as following,

$$N_i = I_i . \dot{w}_i + w_i \; x \; (I_i . w_i)$$

where *i* = 1,2,3,4

$$N_1 = I_1 . \dot{w}_1 + w_1 \; x \; (I_1 . w_1)$$

$$N_2 = I_2 . \dot{w}_2 + w_2 \; x \; (I_2 . w_2)$$

$$N_3 = I_3 . \dot{w}_3 + w_3 \; x \; (I_3 . w_3)$$

$$N_4 = I_4.\dot{w}_4 + w_4\ x\ (I_4.w_4)$$

6.1.2.3. Joint Torques

The backward computation must be completed to reach the torque values. The computation was completed with respect to no-load.

$$f_5 = \begin{bmatrix}0\\0\\0\end{bmatrix}\ ,\ \ n_5 = \begin{bmatrix}0\\0\\0\end{bmatrix}\ ,\ \ R_5 = \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$$

Force:

$$f_i = R_{i+1}^T.f_{i+1} + F_i$$

Moment:

$$n_i = R_{i+1}^T.n_{i+1} + P_i\ x\ f_i + N_i + S_i\ x\ F_i$$

Joint Torque:

$$\tau_i = n_i^T(R_{i+1}^T.z_0)$$

for i = 4,3,2,1

For Link # 4:

$$f_4 = R_5^T.f_5 + F_4$$

$$n_4 = R_5^T.n_5 + P_4\ x\ f_4 + N_4 + S_4\ x\ F_4$$

$$\tau_4 = n_4^T(R_5^T.z_0)$$

For Link # 3:

$$f_3 = R_4^T.f_4 + F_3$$

$$n_3 = R_4^T.n_4 + P_3\ x\ f_3 + N_3 + S_3\ x\ F_3$$

$$\tau_3 = n_3^T(R_4^T.z_0)$$

For Link # 2:

$$f_2 = R_3^T.f_3 + F_2$$

$$n_2 = R_3^T.n_3 + P_2\ x\ f_2 + N_2 + S_2\ x\ F_2$$

$$\tau_2 = n_2^T(R_3^T.z_0)$$

For Link # 1:

$$f_1 = R_2^T.f_2 + F_1$$

$$n_1 = R_2^T.n_2 + P_1\ x\ f_1 + N_1 + S_1\ x\ F_1$$

$$\tau_1 = n_1^T(R_2^T.z_0)$$

Torque Matrix:

$$\tau = \begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\end{bmatrix} = \begin{bmatrix}n_1^T(R_2^T.z_0)\\ n_2^T(R_3^T.z_0)\\ n_3^T(R_4^T.z_0)\\ n_4^T(R_5^T.z_0)\end{bmatrix}$$

Finally, the above formulas were processed into a script and torque graphs were drawn as given below. See Figure VI-1.

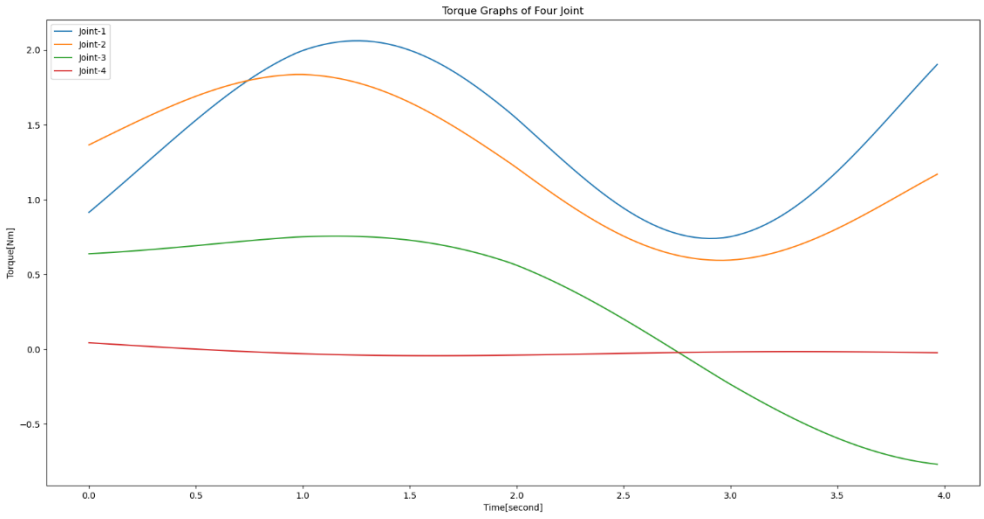

Figure VI-1 *Torque Graphs*

VII. ELECTRONIC CONTROLLER DESIGN

In the electronic controller design, all the electronical components used in the project are combined using some programs in this process. These electronical components can be motors, controller, buttons etc.

7.1. Proteus Design of Serial Manipulator

Proteus design is a software used for electronic design. The software is often used for technical drawing and electronic printing for printed circuit board manufacturing. In addition, in this program, various circuits related to the projects are designed and simulated before production.

In the Proteus design of serial manipulator, firstly, the electrical components used in the serial manipulator were determined and the electronic controller were designed by using these components. These components are servo motors, button, force sensor, Arduino and power supply. In this project, there are five servo motors. Each servo motor is used to rotate each joint and end effector. In addition, a button is used to stop the motors in case of an emergency. There is a force sensor at the end effector to check whether there is an object or not. Also, Arduino controller is used to control all motors and other electronic parts. There is also a power supply to power the Arduino controller.

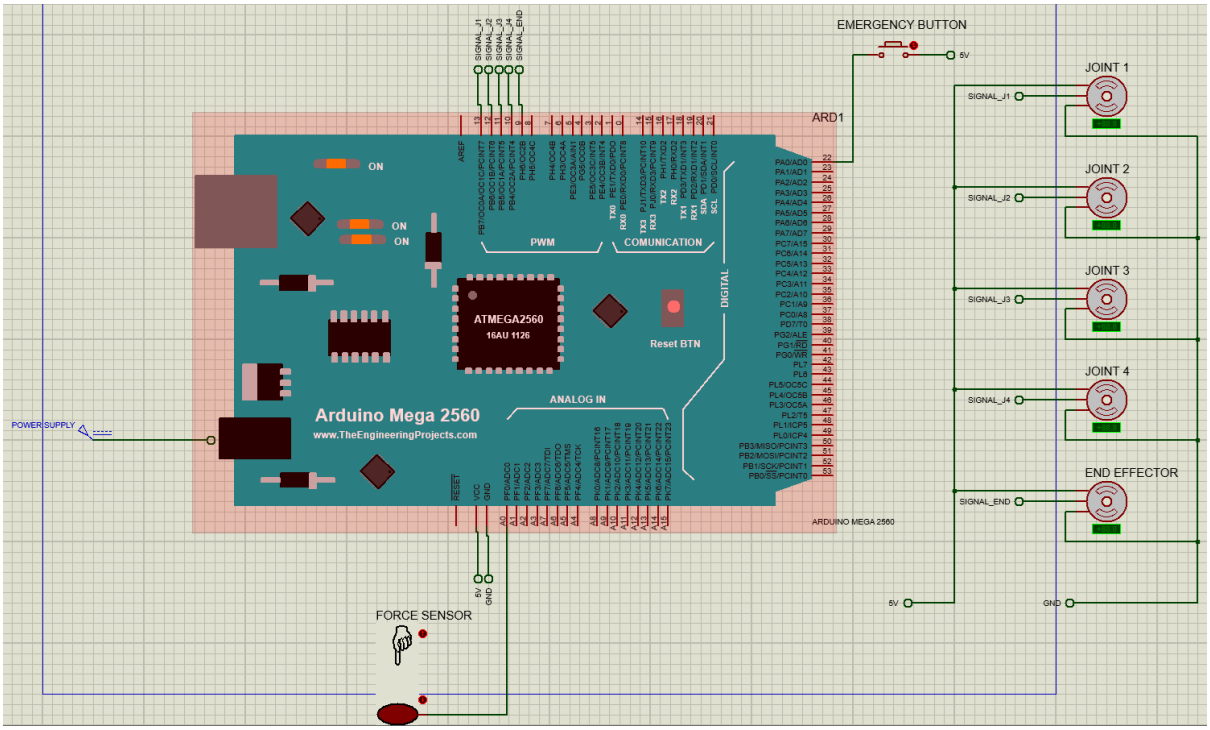

Figure VII-1 *Electronic Circuit Schematic in Proteus*

VIII. SIMULATION ENVIRONMENT

It is necessary to have an environment to realize the task. The task is going to be realized in the Gazebo with RViz and MoveIt. In this direction, the environment should be created in the Gazebo and saved with ". world" extension. After that the environment launch file should be included into the simulation launch file. Therefore, the simulation environments that has been created in the ROS as following.

8.1. Simulation Environment I

The first simulation environment was designed for perform pick and place application with serial manipulator. In this direction, a link with fixed joint has been added between ground and base link of the serial manipulator to raise the robot. In addition of this, two table were created. There are four object on the one table to pick and there is a place on the second table to place the objects. See the following figure.

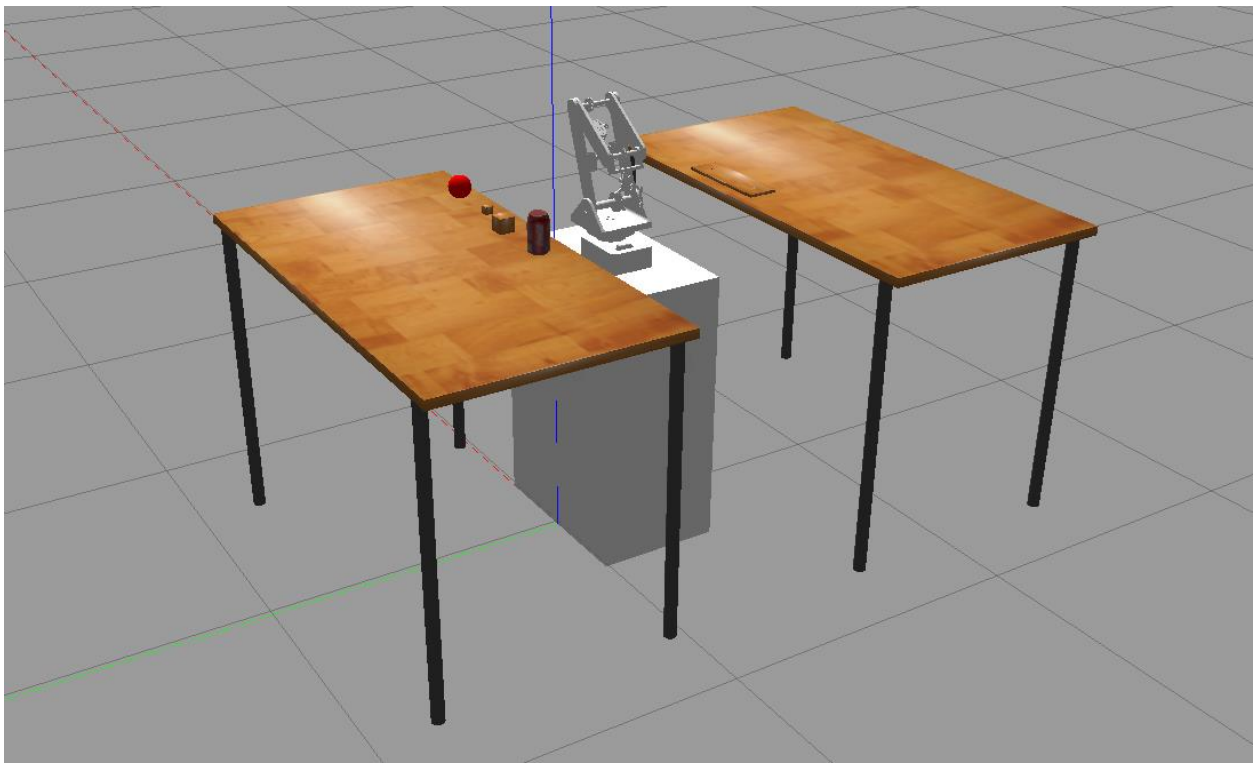

Figure VIII-1 *Simulation Environment I*

8.2. Simulation Environment II

Simulation environment II is designed with a factory environment in mind. The purpose in this environment is to transfer incoming product on the conveyor system to another system. For this reason, two four DoF manipulators were used. These robots manipulators are positioned at right and left on the conveyor but they are in same line. The reason why robot manipulators are on 2 different sides is, for example, when products in two different colors which are red and blue come, the robot manipulator where is at right side of conveyor will transfer the red products to a different system, while the manipulator where is at left side of conveyor will transmit the blue products to a different system.

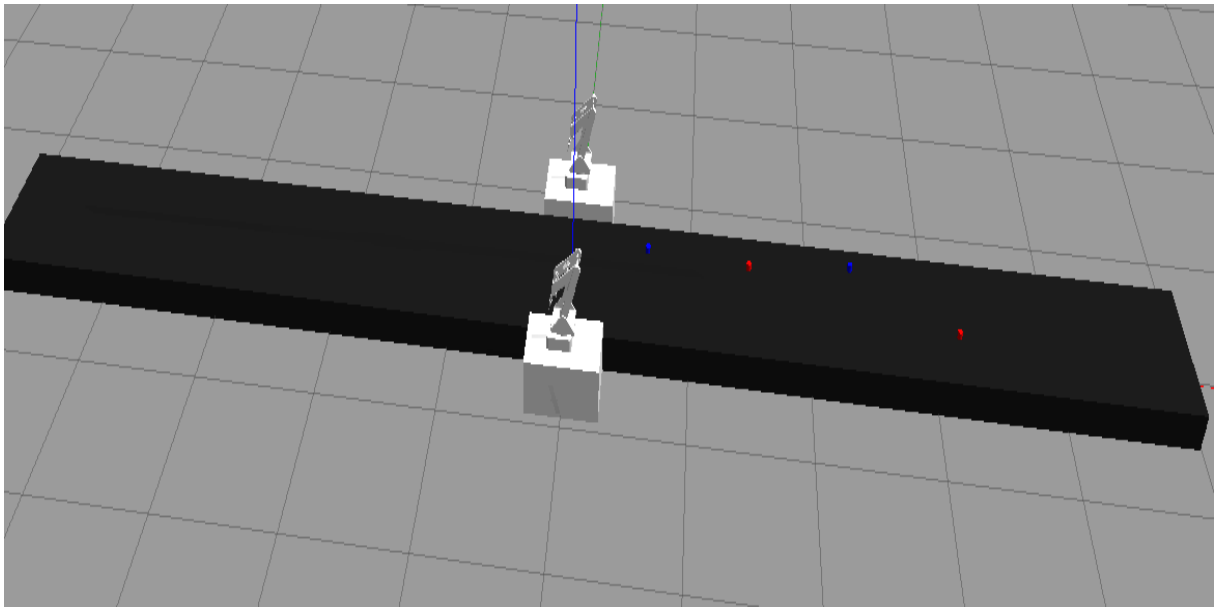

Figure VIII-2 *Simulation Environment II*

IX. CONTROLLER

As mentioned earlier, the “position_controllers/JointTrajectoryController” controller is used to control the manipulator. The necessary parameters for this controller are given in the image below. As a command can be sent to the controller via the terminal, it can also be sent by writing a script over a software language such as Python or C++.

```
bot@vmware:~$ rosmsg show trajectory_msgs/JointTrajectory
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
string[] joint_names
trajectory_msgs/JointTrajectoryPoint[] points
  float64[] positions
  float64[] velocities
  float64[] accelerations
  float64[] effort
  duration time_from_start
```

Figure IX-1 *trajectory_msgs/JointTrajectory message information*

Generally, there are two parts to control the robot. These are called “arm” and “hand”. The

"arm" part consists of “joint_1”, “joint_2”, “joint_3” and “joint_4”. To control this part, “arm_controller” is defined as mentioned before.

To perform the pick and place task, the end effector must be active. For this reason, two revolute joints were added to the gripper set. These joints are "finger1" and "finger2". In this direction, “hand_controller” is defined in the same *yaml* file and with the same controller to control the end effector. "hand_controller" controls "finger1" and "finger2". The final form of the controller is as given below.

```
arm_controller:
  type: position_controllers/JointTrajectoryController
  joints:
    - joint_1
    - joint_2
    - joint_3
    - joint_4

hand_controller:
  type: position_controllers/JointTrajectoryController
  joints:
      - finger1
      - finger2
```

Figure IX-2 *arm_controller and hand_controller*

X. PROGRAMMING AND SIMULATION

The main goal of the project is to realize the pick and place task. In this direction, different simulation environments were created. There are multiple methods to perform the task in these simulation environments. The manipulator can be controlled by sending the angle value directly from the terminal, with inverse kinematic via MoveIt, or by developing code on languages such as Python, C++. In addition, a plugin named “gazebo_grasp_fix” was used to grasp the object in the simulation environment.

As mentioned before, the manipulator controller consists of two parts, *arm_controller* and *hand_controller*. The manipulator can be moved by sending commands to these

controllers via the terminal. See Figure X-1 and Figure X-2.

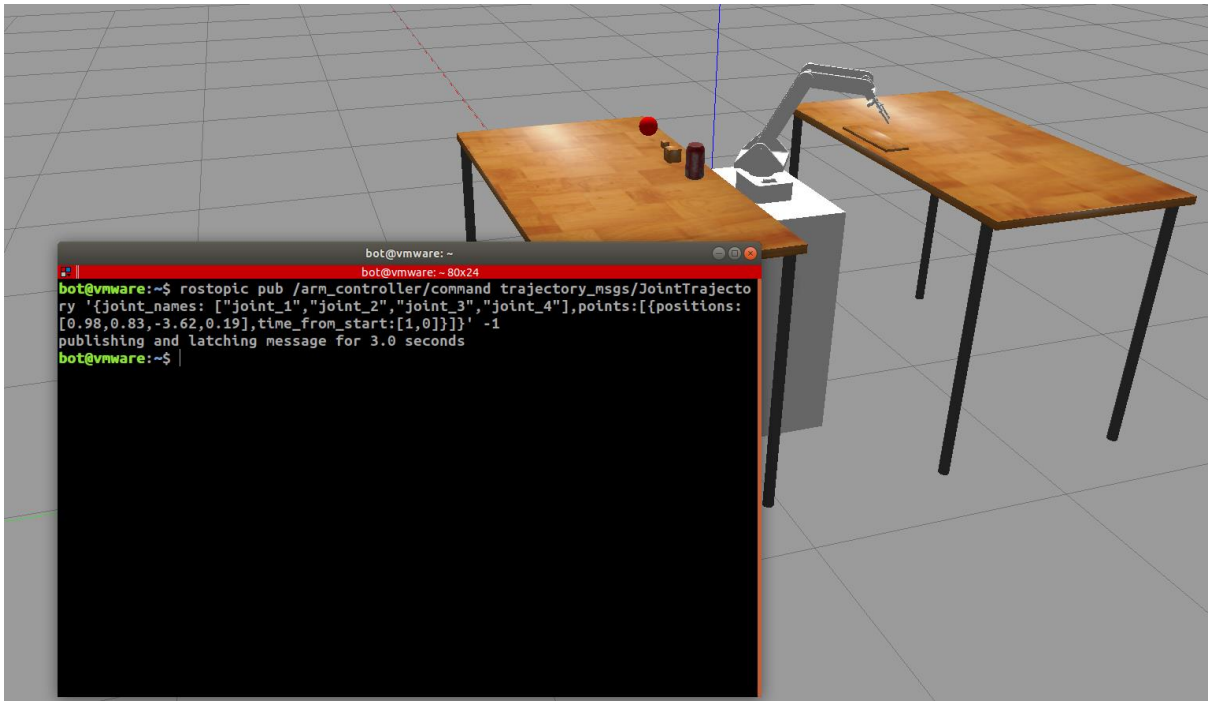

Figure X-1 *arm_controller publish from terminal*

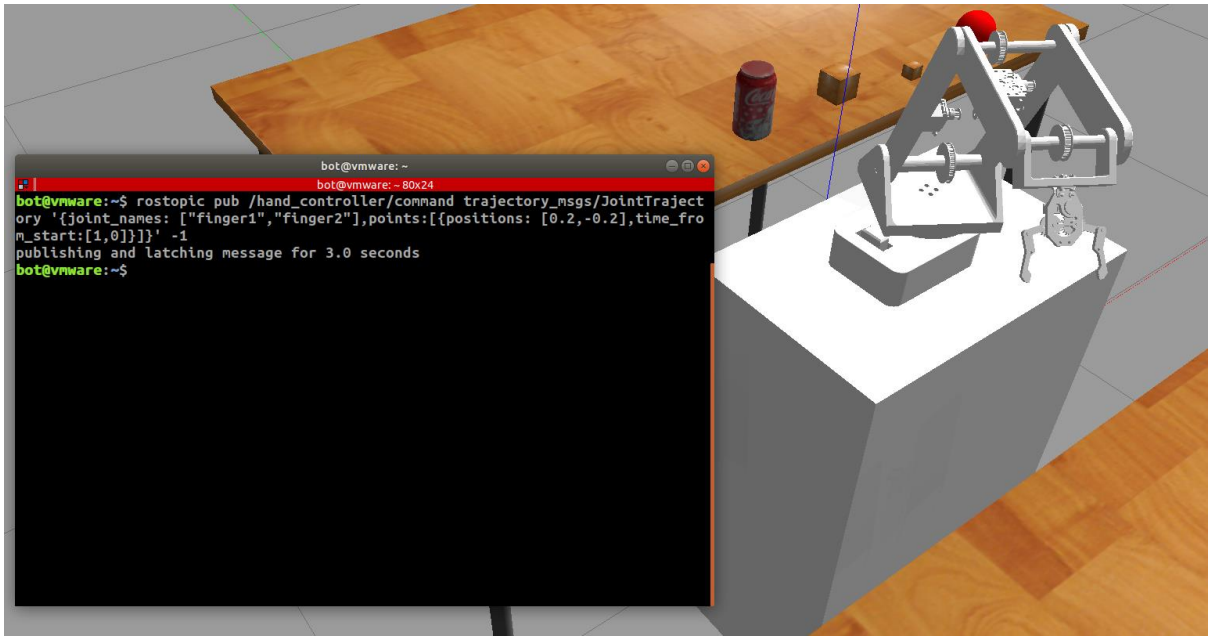

Figure X-2 *hand_controller publish from terminal*

In addition, the task can be performed by running different simulation programs MoveIt and Gazebo together. For this, necessary changes were made in the MoveIt package,

and a single launch file was created to run both simulation programs. Thus, the task can be performed in the Gazebo environment by directing the end effector over MoveIt. See Figure X-3.

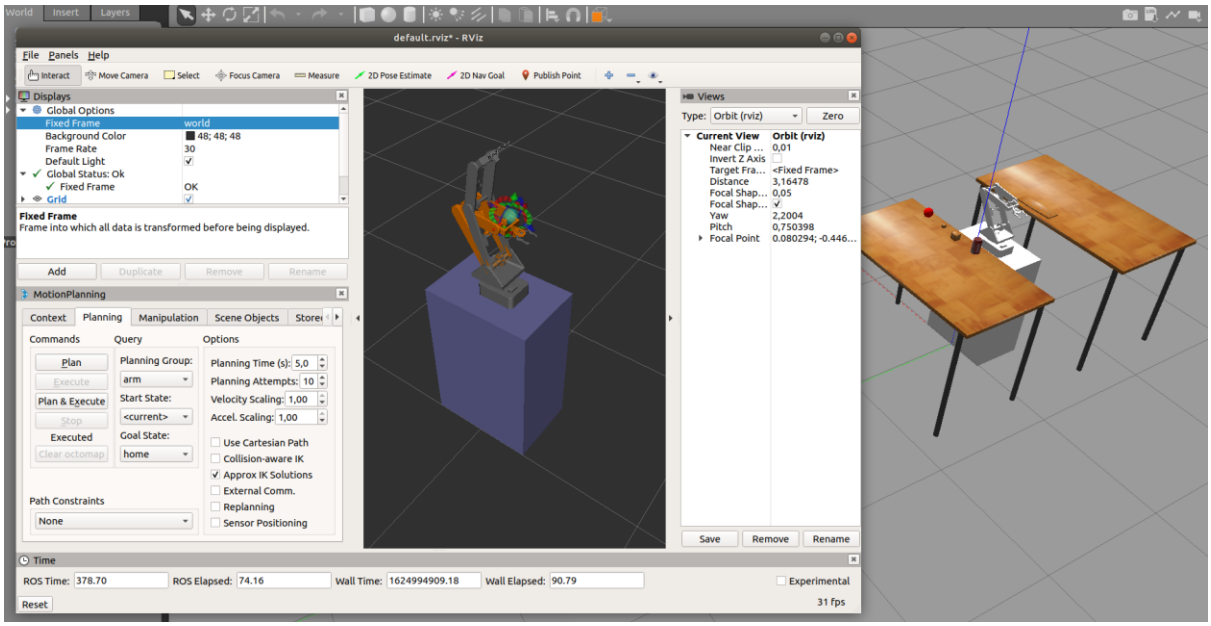

Figure X-3 *MoveIt and Gazebo Communication*

Another method is to write a script to perform the task in the program. With the help of this script, joint angles or goal pose values of the end effector can be sent to the manipulator controller. This method was used to perform the simulation.

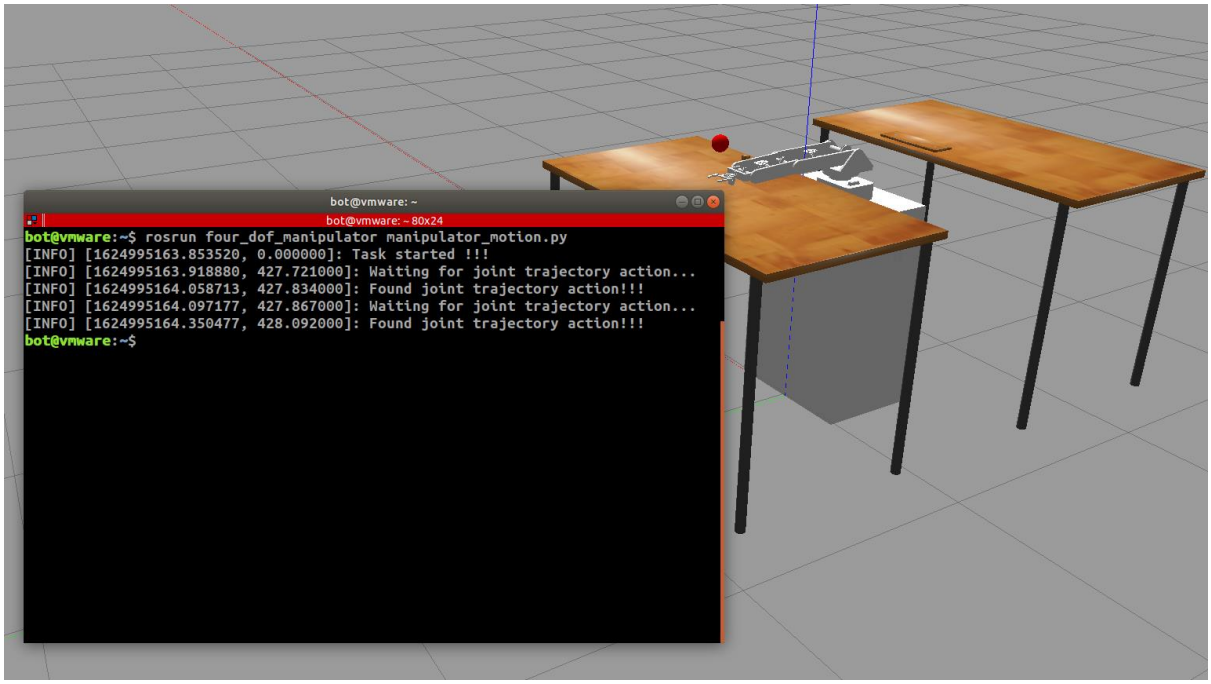

Figure X-4 *Simulation via Python code*

While the end effector is trying to hold the objects in the simulation environment, the objects are thrown out. The above mentioned plugin was used to solve this problem. In this plugin, there are parameters such as the name of the robot manipulator, gripper links.

```
<plugin name="gazebo_grasp_fix" filename="libgazebo_grasp_fix.so">
  <arm>
    <arm_name>four_dof_manipulator</arm_name>
    <palm_link>fourth_link</palm_link>
    <gripper_link>finger_1</gripper_link>
    <gripper_link>finger_2</gripper_link>
  </arm>
  <forces_angle_tolerance>100</forces_angle_tolerance>
  <update_rate>4</update_rate>
  <grip_count_threshold>4</grip_count_threshold>
  <max_grip_count>8</max_grip_count>
  <release_tolerance>0.008</release_tolerance>
  <disable_collisions_on_attach>false</disable_collisions_on_attach>
  <contact_topic>__default_topic__</contact_topic>
</plugin>
```

Figure X-5 *gazebo_grasp_fix plugin*

Finally, the sample simulation was carried out in the gazebo environment using Python script. As seen in the picture below, the object was held during the task and moved to the goal point.

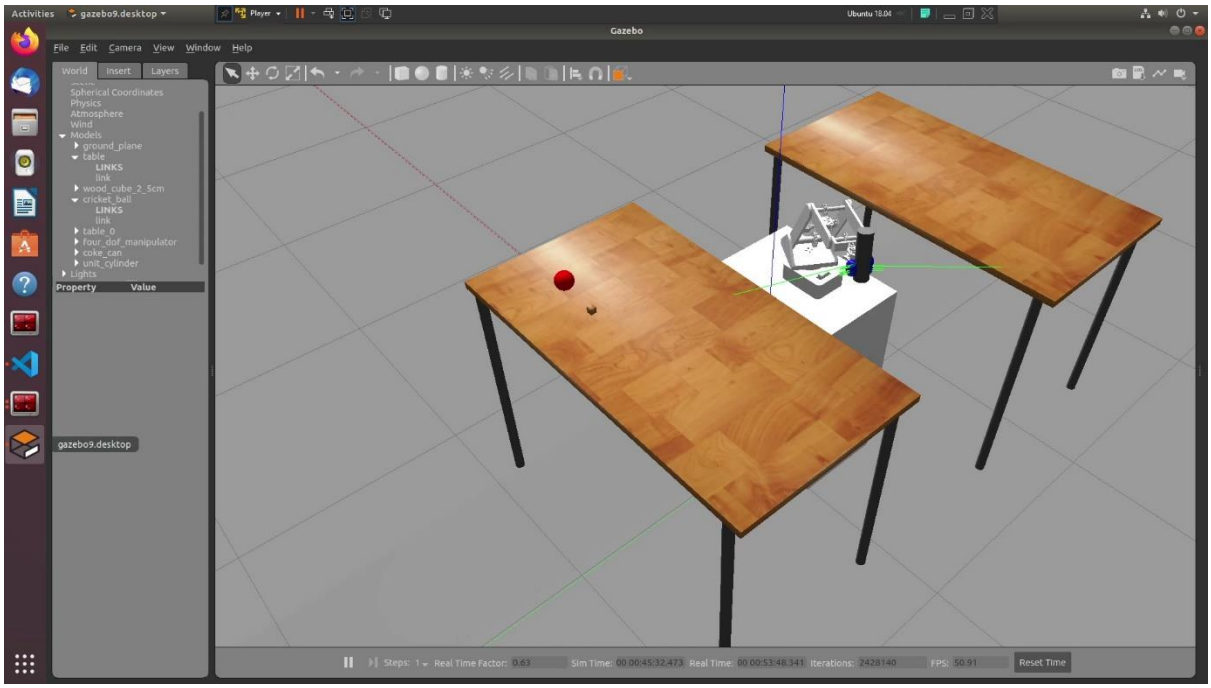

Figure X-6 *Final Simulation*

VI. CONCLUSION AND DISCUSSION

The project is to design two manipulators, one parallel and one serial, to perform kinematic and dynamic analysis and simulate them on the ROS ecosystem. In this direction, the literature survey was made by using necessary keywords and the degrees of freedoms and tasks of the both manipulators were determined in the conceptual design process. The position and velocity analysis was completed for both manipulators in the next step which is kinematic analysis. Then, the detailed mechanical designs of serial and parallel manipulators were designed in the SolidWorks by considering conceptual designs of manipulators. At the end of the first semester, the simulation of the serial manipulator was simulated in the ROS

and the simulation of the parallel manipulator was simulated in the SolidWorks.

In the literature survey process, the related sources about serial manipulator were found easily whereas the needed sources about parallel manipulator especially in simulation were hard to reach. The serial manipulator was designed to have four degrees of freedom, the types of joints that will be used was determined to have revolute joint and the task was determined as pick and place. Also for parallel manipulator, the linear delta robot was designed to have three degrees of freedom with prismatic and spherical joints. In the next step, kinematic analysis, while the related analysis was doing, there were problems to reach the necessary parameters about inverse kinematics of serial manipulator and forward kinematics of the parallel manipulator. The related materials for both manipulators were chosen from materials which are commonly used in the industry and the detailed mechanical designs were designed by considering this condition in the detailed mechanical design. The reason of this situation is that there is no any problem to provide materials that will be used in the production phase. The dynamic analysis phase of the project was not completed and delayed to next semester due to limited time. In the virtual machines, the needed simulation workspace was prepared to simulate designed manipulators. However, in this process there were some encountered problems due to virtual machines. The main reasons of this situation are that the virtual machines do not work as stable and performance. The export URDF, which is generally used to transfer the CAD models to ROS framework, was utilized. The first step of the simulation was realized with the package that is automatically generated with export URDF in the RViz. After that, in the Gazebo simulation process there were some problems about missing parameters as effort and collision that inside of the URDF file. These problems were corrected and the simulation in the Gazebo was completed with the added controller. Finally, the inverse kinematic simulation of the serial manipulator was realized in MoveIt thus the simulation of the serial manipulator was completed. As it is mentioned before, due to the limited time, the simulation of the parallel manipulator was realized in

SolidWorks and the simulation of the parallel manipulator in the ROS was planned to realize next semester.

In the second phase of the project, dynamic analysis and simulation of the parallel manipulator could not be performed due to time constraints. As mentioned before, the dynamic analysis of the serial manipulator, which could not be done, was performed. Then, the electronic circuit diagram of the serial manipulator was designed with the help of Proteus program. Two different simulation environments were designed to perform the simulation. In order to perform the pick and place task in the simulation, the end effector has been activated and the manipulator controller has been updated. Finally, the task was performed in the simulation environment, which was determined by writing a script to perform the simulation. Thus, the project was completed.

As a result of the processes carried out up to this stage, knowledge and experience have been gained about the analytics, designs, production processes, materials used in the construction stages, ROS ecosystem and simulations in this ecosystem of serial and parallel manipulators widely used in the industry.

REFERENCES

[1] Y. Pyo, H. Cho, R. Jung, T. Lim, *ROS Robot Programming*. Seoul: ROBOTIS Co., Ltd., 2017.

[2] M. Quigley, B.Gerkey, W. D. Smart, *Programming Robots with ROS*. CA: O'Reilly Media, Inc., 2015.

[3] L. Joseph, J. Cacace, *Mastering ROS for Robotics Programming*. Birmingham: Packt Publishing Ltd., 2018.

[4] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, *Robotics Modelling, Planning and Control*. London: Springer, 2009.

[5] Edited by S. Küçük, *Serial and Parallel Robot Manipulators-Kinematics, Dynamics, Control and Optimization*. Rijeka: InTech, 2012.

[6] M. W. Spong, S. Hutchinson, M. Vidyasagar, *Robot Dynamics and Control*. New Jersey: John Wiley & Sons, Inc., 2020.

[7] C. D. Crane, J. Duffy, *Kinematic Analysis of Robot Manipulators*. New York: Cambridge University Press, 1998.

[8] L. Tsai, *Robot Analysis-The Mechanics of Serial and Parallel Manipulators*. New Jersey: John Wiley & Sons, Inc., 1999.

[9] J. Kerr, K. Nickels, *Robot Operating Systems: Bridging the Gap between Human and Robot*. IEEE Southeastern Symposium on System Theory, 2012.

[10] D. Oetomo, D. Daney, J. Merlet, *Design Strategy of Serial Manipulators with Certified Constraint Satisfaction.* IEEE Transactions on Robotics, 2009.

[11] S. Patel, T. Sobh, *Goal Directed Design of Serial Robotic Manipulators*. American Society for Engineering Education, 2014.

[12] Y. Patel, P.M.George, *Parallel Manipulators Applications-A Survey*. Modern Mechanical Engineering, 2012.

[13] C. S. G. Lee, M. Ziegler, *A Geometric Approach in Solving the Inverse Kinematics of Puma Robots*. IEEE Transactions on Aerospace and Electronic Systems, 1984.

[14] T. Lens, J. Kunz, O. V. Stryk, *Dynamic Modeling of the 4 DoF BioRob Series Elastic Robot Arm for Simulation and Control*. Simulation, Modeling, and Programming for Autonomous Robots - Second International Conference, 2010.

[15] A. A. Mohammed, M. Sunar, *Kinematics Modeling of a 4-DOF Robotic Arm*. 2015 International Conference on Control, Automation and Robotics, 2015.

[16] K. A. Khan, *ROS-based Control of a Manipulator Arm for Balancing a Ball on a Plate.* American Society for Engineering Education, 2017.

[17] B. Kelly, J. Padayachee, G. Bright, *Quasi-serial Manipulator for Advanced Manufacturing Systems*. 16th International Conference on Informatics in Control, Automation and Robotics, 2019.

[18] B. Siciliano, O.Khatib, *Springer Handbook of Robotics.* Berlin: Springer-Verlag, 2008.

[19] "https://www.britannica.com/ 24.10.2020"

[20] "https://www.ros.org/ 24.10.2020"

[21] Z. Yılmaz, L. Bayındır, *Simulation of Lidar-Based Robot Detection Task using ROS and Gazebo.* European Journal of Science and Technology, 2019.

[22] "https://www.kuka.com/tr-tr# 26.10.2020"

[23] A.Khalid, S.Mekid, *Characteristic Analysis of Parallel Platform Simulators with Different Hardware Configurations.* Pakistan, 2010

[24] "https://moveit.ros.org/ 26.10.2020"

[25] M. J. Uddin, S. Refaat, S. Nahavandi, H. Trinh, *Kinematic and Dynamic Modelling of a Robotic Head with Linear Motors.* School of Engineering and Technology, Deakin University. Australia, 2011.

[26] W. S. Newman, *A Systematic Approach to Learning Robot Programming with ROS.* New York: Taylor & Francis Group, 2018.

[27] A. A. Mohammed, M. Sunar, *Kinematics Modeling of a 4-DOF Robotic Arm*. 2015 International Conference on Control, Automation and Robotics.

B. APPENDIX

URDF CODE OF THE SERIAL MANIPULATOR

```
<?xml version="1.0" encoding="utf-8"?>


<robot name="four_dof_manipulator">


 <!--Fixed the table to the ground(world)-->

 <link name="world" />


 <!-- Table -->

 <link name="table">

  <visual>

   <origin rpy="0 0 0" xyz="0 -0.2 0"/>

   <geometry>

    <box size="0.6 0.4 0.9"/>

   </geometry>

   <material name="grey">

    <color rgba="0.5 0.5 0.75 1"/>
```

```
    </material>
  </visual>
</link>


<!-- World-Table joint-->
<joint name="fixed" type="fixed">
  <parent link="world"/>
  <child link="table"/>
  <origin xyz="0.0 0.0 0.450" rpy="0.0 0.0 0.0" />
</joint>


<link name="base_link">
  <inertial>
    <origin
      xyz="-0.0330941580739677 -0.199915714499144 -0.0238474252824564"
      rpy="0 0 0" />
    <mass
      value="1.21592569506171" />
```

```
  <inertia
    ixx="0.00319394143397767"
    ixy="6.50916537053405E-06"
    ixz="5.00849983752122E-05"
    iyy="0.00517355396442031"
    iyz="-2.59154797593509E-06"
    izz="0.00752990865416603" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://four_dof_manipulator/meshes/base_link.STL" />
  </geometry>
  <material
    name="">
```

```
      <color
        rgba="0.5 0.5 0.5 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/base_link.STL" />
    </geometry>
  </collision>
</link>

<!--Table-Base Joint-->
<joint name="table_joint" type="fixed">
  <parent link="table"/>
```

```
  <child link="base_link"/>
  <origin xyz="0.0 0.0 0.518" rpy="0.0 0.0 0.0" />
</joint>

<!--joint name="fixed_joint" type="fixed">
  <parent link="world"/>
  <child link="base_link"/>
  <origin xyz="0.0 0.0 0.518" rpy="0.0 0.0 0.0" />
</joint-->

<link
  name="first_link">
  <inertial>
    <origin
      xyz="-7.33788030338189E-16 5.55111512312578E-17 -0.0544923827293373"
      rpy="0 0 0" />
    <mass
      value="1.20713988278658" />
```

```
  <inertia
    ixx="0.00387509235916713"
    ixy="1.0875136639203E-10"
    ixz="2.90490271371626E-17"
    iyy="0.00304586202042799"
    iyz="2.15933858679175E-18"
    izz="0.00589439778201493" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://four_dof_manipulator/meshes/first_link.STL" />
  </geometry>
  <material
    name="">
```

```
      <color
        rgba="0.5 0.5 0.5 1" />
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/first_link.STL" />
    </geometry>
  </collision>
</link>


<joint
  name="joint_1"
  type="revolute">
```

```
  <origin
    xyz="-0.00208795693759905 -0.199999999999975 -0.00419080295344085"
    rpy="-3.14159265358979 0 0.0110713534620342" />
  <parent
    link="base_link" />
  <child
    link="first_link" />
  <axis
    xyz="0 0 -1" />
  <limit
    lower="-1.5707963268"
    upper="1.5707963268"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran1">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_1">
```

```
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor1">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<link
  name="second_link">
  <inertial>
    <origin
      xyz="0.122656869389677 -0.000401462272982094 0.0686321634800344"
      rpy="0 0 0" />
    <mass
      value="0.564197355185099" />
    <inertia
      ixx="0.00247734722871887"
```

```
    ixy="4.32451514645994E-06"
    ixz="1.43966218295339E-06"
    iyy="0.00767638543625201"
    iyz="-1.39190385070749E-07"
    izz="0.0053992835651082" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/second_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.5 0.5 0.5 1" />
```

```
    </material>
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/second_link.STL" />
    </geometry>
  </collision>
</link>


<joint
  name="joint_2"
  type="revolute">
  <origin
    xyz="0 -0.0680000000000001 -0.117000000000003"
```

```
    rpy="-1.5707963267949 1.23097997933017 0" />
  <parent
    link="first_link" />
  <child
    link="second_link" />
  <axis
    xyz="0 0 1" />
  <limit
    lower="-1.2217304764"
    upper="1.9198621772"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran2">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_2">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
```

```
  <actuator name="motor2">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>



<link
  name="third_link">
  <inertial>
    <origin
      xyz="0.089483369514526 0.000333185437360051 0.0616705943939753"
      rpy="0 0 0" />
    <mass
      value="0.341491453338833" />
    <inertia
      ixx="0.0011223536458998"
      ixy="-8.94838519701279E-07"
```

```
    ixz="4.50772424468417E-07"
    iyy="0.00260413334203537"
    iyz="-3.46258625510181E-06"
    izz="0.00161078736660313" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/third_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.5 0.5 0.5 1" />
    </material>
```

```
  </visual>
  <collision>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/third_link.STL" />
    </geometry>
  </collision>
</link>


<joint
  name="joint_3"
  type="revolute">
  <origin
    xyz="0.3 0 0.124"
    rpy="-3.1416 0 -1.5068" />
```

```
  <parent
    link="second_link" />
  <child
    link="third_link" />
  <axis
    xyz="0 0 -1" />
  <limit
    lower="-6.2831853072"
    upper="-2.2689280276"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran3">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_3">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor3">
```

```
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<link
  name="fourth_link">
  <inertial>
    <origin
      xyz="-0.0560452574719653 -0.00176057766334337 0.032176511601107"
      rpy="0 0 0" />
    <mass
      value="0.114529356723246" />
    <inertia
      ixx="0.000111734436978805"
      ixy="4.99245419880645E-07"
      ixz="-2.76231736304427E-06"
      iyy="0.00022500515976646"
```

```
    iyz="8.28933809672081E-06"
    izz="0.000134457409408795" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/fourth_link.STL" />
    </geometry>
    <material
      name="">
      <color
        rgba="0.5 0.5 0.5 1" />
    </material>
  </visual>
  <collision>
```

```
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/fourth_link.STL" />
    </geometry>
  </collision>
</link>

<joint
  name="joint_4"
  type="revolute">
  <origin
    xyz="0.2 0 0"
    rpy="0.645832947767902 1.5707963267949 0" />
  <parent
    link="third_link" />
```

```
  <child
    link="fourth_link" />
  <axis
    xyz="1 0 0" />
  <limit
    lower="-2.4434609528"
    upper="0.6981317008"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran4">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="joint_4">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor4">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
```

```
  </actuator>
</transmission>

<link
  name="finger_1">
  <inertial>
    <origin
      xyz="0.00414389676302235 -0.023459494198062 -0.0133757992997916"
      rpy="0 0 0" />
    <mass
      value="0.00517538664215202" />
    <inertia
      ixx="3.22454593367971E-06"
      ixy="1.9898922789917E-07"
      ixz="8.48766484652095E-08"
      iyy="6.36992094608751E-07"
      iyz="-5.48312994811873E-07"
      izz="2.66441097287338E-06" />
```

```
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://four_dof_manipulator/meshes/finger_1.STL" />
  </geometry>
  <material
    name="">
    <color
      rgba="0.5 0.5 0.5 1" />
  </material>
</visual>
<collision>
  <origin
    xyz="0 0 0"
```

```
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://four_dof_manipulator/meshes/finger_1.STL" />
    </geometry>
  </collision>
</link>


<joint
  name="finger1"
  type="revolute">
  <origin
    xyz="-0.0425 -0.006 0.092879"
    rpy="-1.4841 0 1.5708" />
  <parent
    link="fourth_link" />
  <child
    link="finger_1" />
```

```
  <axis
    xyz="1 0 0" />
  <limit
    lower="-0.3"
    upper="0.3"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran5">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="finger1">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor5">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

```
<link
  name="finger_2">
  <inertial>
    <origin
      xyz="0.0042346 0.023831 -0.015369"
      rpy="0 0 0" />
    <mass
      value="0.0050037" />
    <inertia
      ixx="3.0765E-06"
      ixy="-1.817E-07"
      ixz="9.3166E-08"
      iyy="6.8078E-07"
      iyz="6.4464E-07"
      izz="2.4699E-06" />
  </inertial>
  <visual>
```

```
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://four_dof_manipulator/meshes/finger_2.STL" />
  </geometry>
  <material
    name="">
    <color
      rgba="0.5 0.5 0.5 1" />
  </material>
</visual>
<collision>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
```

```
      <mesh
        filename="package://four_dof_manipulator/meshes/finger_2.STL" />
    </geometry>
  </collision>
</link>


<joint
  name="finger2"
  type="revolute">
  <origin
    xyz="-0.0695 -0.006 0.092879"
    rpy="1.5656 0 1.5708" />
  <parent
    link="fourth_link" />
  <child
    link="finger_2" />
  <axis
    xyz="1 0 0" />
```

```
  <limit
    lower="-0.3"
    upper="0.3"
    effort="500"
    velocity="2" />
</joint>
<transmission name="tran6">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="finger2">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
  </joint>
  <actuator name="motor6">
    <hardwareInterface>PositionJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>


<!--Gazebo simulator part-->
```

```
<gazebo>
  <plugin name="control" filename="libgazebo_ros_control.so"/>
  <plugin name="joint_state_publisher"
filename="libgazebo_ros_joint_state_publisher.so">
    <jointname>joint_1,joint_2,joint_3,joint_4,finger1,finger2</jointname>
  </plugin>
  <plugin name="gazebo_grasp_fix" filename="libgazebo_grasp_fix.so">
    <arm>
      <arm_name>four_dof_manipulator</arm_name>
      <palm_link>fourth_link</palm_link>
      <gripper_link>finger_1</gripper_link>
      <gripper_link>finger_2</gripper_link>
    </arm>
    <forces_angle_tolerance>100</forces_angle_tolerance>
    <update_rate>4</update_rate>
    <grip_count_threshold>4</grip_count_threshold>
    <max_grip_count>8</max_grip_count>
    <release_tolerance>0.008</release_tolerance>
    <disable_collisions_on_attach>false</disable_collisions_on_attach>
```

```
      <contact_topic>__default_topic__</contact_topic>
    </plugin>
  </gazebo>

</robot>
```

RVIZ LAUNCH CODE

```
    name="model" />
  <param
    name="robot_description"
    textfile="$(find four_dof_manipulator)/urdf/four_dof_manipulator.urdf" />
  <node
    name="joint_state_publisher_gui"
    pkg="joint_state_publisher_gui"
    type="joint_state_publisher_gui" />
  <node
    name="robot_state_publisher"
    pkg="robot_state_publisher"
    type="robot_state_publisher" />
  <node
    name="rviz"
    pkg="rviz"
    type="rviz"
    args="-d $(find four_dof_manipulator)/urdf.rviz" />
</launch>
```

GAZEBO LAUNCH CODE

```
<launch>


  <param name="robot_description" textfile="$(find
four_dof_manipulator)/urdf/four_dof_manipulator.urdf"/>


  <include file="$(find gazebo_ros)/launch/empty_world.launch">

    <arg name="world_name" value="$(find
four_dof_manipulator)/worlds/simulation_environment.world"/>

    <!--arg name="world_name" value="$(find
four_dof_manipulator)/worlds/pick_and_place.world"/-->

  </include>

  <node

    name="tf_footprint_base"

    pkg="tf"

    type="static_transform_publisher"

    args="0 0 0 0 0 0 base_link base_footprint 40" />

  <node

    name="spawn_model"

    pkg="gazebo_ros"

    type="spawn_model"

    args="-file $(find four_dof_manipulator)/urdf/four_dof_manipulator.urdf -urdf -model
four_dof_manipulator"

    output="screen" />


  <node

    name="fake_joint_calibration"

    pkg="rostopic"
```

```
    type="rostopic"

    args="pub /calibrated std_msgs/Bool true" />


  <node name="controller_spawner" pkg="controller_manager" type="spawner"
 args="arm_controller hand_controller"/>

  <rosparam file="$(find four_dof_manipulator)/yaml/jtc_controller.yaml"
 command="load"/>

  <node name="robot_state_publisher" pkg="robot_state_publisher"
 type="robot_state_publisher" output="screen"/>


  <!--rosparam file="$(find four_dof_manipulator)/yaml/gains.yaml" command="load"/-->




 </launch>
```

YAML CODE FOR CONTROLLER

```
arm_controller:
  type: position_controllers/JointTrajectoryController
  joints:
    - joint_1
    - joint_2
    - joint_3
    - joint_4

hand_controller:
  type: position_controllers/JointTrajectoryController
  joints:
     - finger1
        - finger2
```

PYTHON CODE FOR SIMULATION

```
#!/usr/bin/env python

from control_msgs.msg import FollowJointTrajectoryAction, FollowJointTrajectoryGoal
import roslib;roslib.load_manifest('four_dof_manipulator')
from trajectory_msgs.msg import JointTrajectoryPoint
from std_msgs.msg import Float64
import actionlib
import rospy
import time

class Manipulator:
    def __init__(self, robot_group):
        #arm_name should be b_arm or f_arm
        self.robot_group = robot_group
        self.jta                                                                    =
    actionlib.SimpleActionClient('/'+self.robot_group+'_controller/follow_joint_trajectory',
    FollowJointTrajectoryAction)
        rospy.loginfo('Waiting for joint trajectory action...')
        self.jta.wait_for_server()
        rospy.loginfo('Found joint trajectory action!!!')


    def move_joint(self, angles):
        goal = FollowJointTrajectoryGoal()
        # joint names
        goal.trajectory.joint_names = ['joint_1', 'joint_2','joint_3','joint_4']
        point = JointTrajectoryPoint()
        point.positions = angles
        point.time_from_start = rospy.Duration(3)
        goal.trajectory.points.append(point)
        self.jta.send_goal_and_wait(goal)

class Gripper(Manipulator):

    def move_joint(self, angles):
        goal = FollowJointTrajectoryGoal()
        # joint names
        goal.trajectory.joint_names = ['finger1', 'finger2']
        point = JointTrajectoryPoint()
        point.positions = angles
        point.time_from_start = rospy.Duration(3)
        goal.trajectory.points.append(point)
        self.jta.send_goal_and_wait(goal)
```

```
def main():
    rospy.loginfo('Task started !!!')
    arm = Manipulator('arm')
    hand = Gripper('hand')

    time.sleep(2)
    hand.move_joint([-0.05,0.08]) # keep
    time.sleep(2)
    arm.move_joint([0,0.39,-3.40,-0.69])
    time.sleep(2)
    arm.move_joint([1.57,1.92,-4.74,-0.69])
    time.sleep(2)
    hand.move_joint([0.25,-0.25])
    time.sleep(2)
    arm.move_joint([0,0,0,0])

if __name__ == '__main__':
      rospy.init_node('four_dof_manipulator_move')
          main()
```

VITA (ÖZGEÇMİŞ)

Ali Aydın Küçükçöllü was born in 1998 in Aydın, Turkey. He is studying in İzmir Katip Çelebi University and will receive the bachelor of science in mechatronics engineering in 2021. He worked as an intern in Jantsa Wheel Industry in the summer of 2019. Küçükçöllü has been a student member of Association Mechatronic Engineers since 2017.

Batuhan Yalçınkaya was born in 1998 in İzmir, Turkey. He is studying in İzmir Katip Çelebi University and will receive the bachelor of science in mechatronics engineering in 2021. Yalçınkaya has been a student member of Association Mechatronic Engineers since 2020.

Nurettin Uğur Alagaş was born in 1998 in İzmir, Turkey. He is studying in İzmir Katip Çelebi University and will receive the bachelor of science in mechatronics engineering in 2021. He worked as an intern in MDK in the summer of 2019. Alagaş has been a student member of Association Mechatronic Engineers since 2020.