A. KAHVECİ

IZMIR KATIP CELEBI UNIVERSITY

IZMIR KATIP CELEBI UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND ROBOT OPERATING SYSTEM BASED CONTROL OF A MODULAR ROBOT MANIPULATOR

M.Sc. THESIS

Aytaç KAHVECİ

Department of Mechanical Engineering

JULY 2019

2019

IZMIR KATIP CELEBI UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DESIGN AND ROBOT OPERATING SYSTEM BASED CONTROL OF A MODULAR ROBOT MANIPULATOR

M.Sc. THESIS

Aytaç KAHVECİ
(Y160217009)

Department of Mechanical Engineering

Thesis Advisors: Asst. Prof. Dr. Özgün BAŞER
Asst. Prof. Dr. Erkin GEZGİN

JULY 2019

İZMİR KATİP ÇELEBİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MODÜLER ROBOT MANİPÜLATÖR TASARIMI VE ROBOT İŞLETİM SİSTEMİ TABANLI KONTROLÜ

YÜKSEK LİSANS TEZİ

Aytaç KAHVECİ
(Y160217009)

Makine Mühendisliği Ana Bilim Dalı

Tez Danışmanları: Dr. Öğr. Üyesi Özgün BAŞER
Dr. Öğr. Üyesi Erkin GEZGİN

TEMMUZ 2019

Aytaç KAHVECİ, a **M.Sc.** student of **IKCU Graduate School Of Natural And Applied Sciences**, successfully defended the thesis entitled "**DESIGN AND ROBOT OPERATING SYSTEM BASED CONTROL OF A MODULAR ROBOT MANIPULATOR"**, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :

Asst. Prof. Dr. Özgün BAŞER .....
İzmir Katip Çelebi University

Thesis Co-Advisor :

Asst. Prof. Dr. Erkin GEZGİN .....
İzmir Katip Çelebi University

Jury Members :

Assoc. Prof. Dr. Levent MALGACA .....
Dokuz Eylül University

Assoc. Prof. Dr. Ahmet ÖZKURT .....
Dokuz Eylül University

Asst. Prof. Dr. Fatih Cemal CAN .....
İzmir Katip Çelebi University

Date of Submission : 02.07.2019
Date of Defense : 22.07.2019

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS

ABS : Acrylonitrile Butadiene Styrene
DH : Denavit-Hartenberg
DOF : Degrees of Freedom
GUI : Graphical User Interface
KDL : The Kinematics and Dynamics Library
MDF : Medium Density Fiberboard
MIMO : Multiple Input Multiple Output
OROCOS : Open Robot Control Software
PC : Personal Computer
ROS : Robot Operating System
RRT* : Rapidly-Exploring Random Tree Star
Rviz : ROS Visualizer
SDK : Software Development Kit
SISO : Single Iput Single Output
STL : Stereolithography
URDF : Unified Robot Description Format
XML : Extensible Markup Language

LIST OF TABLES

Page

LIST OF FIGURES

Page

DESIGN AND ROBOT OPERATING SYSTEM BASED CONTROL OF A MODULAR ROBOT MANIPULATOR

ABSTRACT

In this thesis, a new design of the modular robot manipulator and its control with Robot Operating System is presented. The modules are designed to have an adjustable twist angle, which enables to create different robot manipulator configurations. Kinematic synthesis on the gripper module is conducted and structural parameters of the gripper is determined. Axis sets on the modules are determined by Denavit-Hartenberg method. Inverse and forward kinematic analysis and singularity analysis are performed. Numerical inverse kinematic solver is proposed for modular robot manipulators with more than 3 degrees of freedom and the effectiveness of solver is evaluated.

The robot dynamic model, which is also important in the control of robot manipulators, is created for modular robot manipulator. Modular robot manipulator model is created in ROS environment in accordance with determined kinematic structure and dynamic model and the resulted model is verified by comparing it with MATLAB simulation results. Computed torque controller is developed in ROS and its performance is tested on simulation model as well as single joint controllers in ROS.

In order to test the performance of modular robot manipulator an experimental rig is created. Joint trajectory controller is implemented on 2 DOF and 3 DOF modular robot manipulators and trajectory tracking performance of the controller is given.

Motion planning studies are performed both on simulation model and experimental setup. By means of developed graphical user interface, it is allowed to control the robot with jog mode, teach points to the robot and plan motions with predefined command sets.

MODÜLER ROBOT MANİPÜLATÖR TASARIMI VE ROBOT İŞLETİM SİSTEMİ TABANLI KONTROLÜ

ÖZET

Bu tezde yeni modüler robot manipulator tasarımı ve Robot İşletim Sistemi ile kontrolü sunulmaktadır. Modüller, farklı robot manipulator konfigürasyonları oluşturmaya izin veren ayarlanabilir büküm açılarına sahip olacak şekilde tasarlanmıştır. Tutucu modülünün kinematik sentezi gerçekleştirilmiş ve tutucunun yapısal parametreleri belirlenmiştir. Modüllerdeki eksen takımları Denavit-Hartenberg yöntemi kullanılarak belirlenmiştir. İleri ve geri yön kinematic analizleri ve tekillik analizleri gerçekleştirilmiştir. Üç serbestlik derecesinden fazla modüler robot manipülatörler için nümerik ters kinematik çözücüsü önerilmiş ve etkinliği değerlendirilmiştir.

Robot manipülatörlerin kontrolü için de önemli olan robot dinamik modeli modüler robot manipülatör için oluşturmuştur. Belirlenen kinematik yapıya uygun olarak ROS ortamında modüler robot manipülatör modeli oluşturulmuş ve model, MATLAB simülasyon sonuçları ile kıyaslanarak doğrulanmıştır. Hesaplanan tork kontrolü ROS ta geliştirilmiş ve performansı tek eklem kontrolcüsünde olduğu gibi simülasyon ortamında test edilmiştir.

Modüler robot manipülatörün performansını test etmek için bir deney düzeneği oluşturulmuştur. Eklem yörünge kontrolcüsü 2 serbestlik dereceli ve 3 serbestlik dereceli modular robot manipülatörler için uygulanmış ve yörünge takip performansları verilmiştir.

Simülasyon modeli üzerinde ve deney düzeneği üzerinde hareket planlaması alışmaları gerçekleştirilmiştir. Geliştirilen kullanıcı arayüzü sayesinde robotu manuel kontrol etme, robota nokta öğretme ve ön tanımlı komut setleri ile hareket planlaması yapma imkanı tanınmıştır.

1. INTRODUCTION

Nowadays, most industrial robots can be considered as universal solutions to certain types of tasks. These industrial robots usually have minimal mechanical flexibility in terms of reconfigurability for achieving different robot configurations. That fixed structure of the robot does not allow it to adapt itself optimally to different tasks. Therefore, usage of these fixed-structure industrial robots in higher versatility in tasks is not always possible and it is vital selecting robot manipulator configuration which is capable of fulfilling aimed tasks.

On contrary to fixed-structure industrial robots, modular robots can be adapted to wide variety of tasks due to their construction which include sets of reconfigurable modules. Interchangeable modular structure of the robots increases their availability rates in the production line. These capabilities of the modular robots create a high potential for use in industrial areas.

This study presents a new modular robot manipulator design and its control with up-to-date robot control framework ROS. This chapter provides an overview of the thesis.

The thesis is organized as follows. Section 1.3 describes the motivation for the thesis. Section 1.4 provides overview of studies that are made in the modular robotic area. Section 2 explains conceptual design of the modules. Section 3 includes detailed kinematic and dynamic analysis of the modules. Section 4 introduces ROS and components which are used in the ROS. Section 5 presents controller design for modular robot manipulator. Section 6 presents used experimental setup. Section 7 gives the results of the analysis and experiments.

1.1 Definition of the Problem

In today's industry, the use of flexible automation lines is needed in production. Especially with the developments in Industry 4.0, the importance of flexible automation lines has increased. In Industry 4.0, smart components in the production

lines communicates with each other and can make decisions themselves without human interactions. By accessing and analyzing more data on the process, these components optimize themselves to enhance production quality and efficiency.

In order to achieve this change, industrial robots, one of the most widely used equipment in automation, also need to gain structural flexibility. Nowadays, studies on modular robot manipulators are being carried out in order to address this issue. Reconfigurable structures of the modular robot manipulators ensure the flexibility in production lines. By creating smart modules, traceability of the production can be enhanced and by gathering more data optimization of the process can be achieved.

1.2 Motivation

Motivation to this thesis is considered in four topics.

Lack of modular robots for industrial areas

As it will be stated in the Literature Review section 1.3, modular robots have not yet fully been adapted to the industrial areas. By studying in this research topic, it is aimed to contribute the application areas of the modular robots. When literature is reviewed it is observed that modular robot manipulators are mostly used for research purposes. However, in this study, it is intended to use of modular robot manipulator in assembly line in Industry 4.0.

Higher potential of modular robots

In the last decade, modular robots are started to be used in mechatronics education, since modularity gives advantages to both students and educators in the comprehension of the basic concepts related with the field of robotics and mechatronics. For this reason, the secondary aim of this thesis will be to design a user-friendly modular robot which can be used in mechatronics education.

Because of their higher potentials as described in the Literature Review, this research topic was considered in order to follow increasing trend.

Lack of modular robot applications in ROS

Although ROS is utilized in many robotic applications, there are still very few studies on integration of ROS with the modular systems. For this reason, the third aim of this

thesis is to prove the effectiveness of application of popular robotic software framework ROS on modular systems and contribute to the ROS community.

Low cost robot manipulator solution

The fourth aim of this study is to offer low cost robot manipulator solution to the industrial applications. Open source software structure is important for this purpose. It reduces the costs and this is the advantage of the ROS integrated modular robots when compared with expensive industrial robots. It is also possible to offer low cost robotic integration with modular robot manipulators. Instead of buying two or three industrial robots, more tasks can be carried on with having modular robot manipulator and reconfiguring it with different robot structure.

1.3 Literature Review

Reconfigurable and modular robot manipulators are mechatronic systems composed of interchangeable modules. The modular nature of these systems allows them to be adapted for different applications, which is a clear advantage with respect to fixed-structure robots.

Modular and reconfigurable robot systems have been popular research topics for more than 20 years [1]. Modular robots are generally categorized according to their configuration properties as manually configurable and self-reconfigurable systems. [2] Self-reconfigurable modular robots are those that are able to change their configuration on their own, while manually configurable modular robots are modular robots that have to be assembled by operator.

Earlier researches are focused on self-reconfigurable modular robots because of their higher potentials. CEBOT [3] , PolyPod [4] and Tetrobot [5], KAIRO3 [6] and Modular Amphibious Snake-like Robot [7] and SambotII [8] are some of the self-reconfigurable modular robots manufactured. On the other hand, Fable II [9] and Alligator [10] are the some of the manually reconfigurable modular robots manufactured.

The standard parts of the modular robots can be produced by using 3D printer technology. It is showed that modules can be joined (attached) together via magnets as well as mechanical components. Joined through magnets are chosen for robots

which do not carry loads. It is not the good coupled technique for industrial robots. M-blocks [11] is one of the studies which use magnetic coupling techniques.

Thanks to their reconfigurable structures modular robots can adapt themselves to the great versatile applications. In literature, it is seen that modular robots are used especially for search and rescue operations [12], for space explorations [13] and as service robots [14]. One of the challenges of modular robot manipulators is to overcome the negative impact of gravity. Since the modules are added end-to-end, the center of gravity moves away from the base module under the effects of each actuator weights. However, for the robotic applications in space explorations, this disadvantage of the modular robot manipulator can be neglected due to the lack of gravity.

Modular robot manipulators have the following advantages over the fixed-structure robot manipulators [15]:

Versatility: reconfigurable robotic systems are more adaptive than conventional systems. The ability to reconfigure allows a robot to disassemble and reassemble to form new morphologies that are better suited for new tasks.

Robustness: Interruptions in production and assembly lines are crucial for the production efficiency. Since the robot parts are interchangeable in the modular structure, the defective parts can be very quickly replaced automatically (or manually) replaced. Hence, this property has a positive effect on system robustness.

Low cost: reconfigurable robotic systems can potentially lower overall robot cost by making many copies of one type of modules so economies of scale and mass production come into play. Also, a range of complex solutions can be made from one set of modules, saving costs through reuse.

However, these advantages have not yet been fully implemented in real life, and this has led to the lack of modular design of industrial robots used in production. In addition, the change in the degree of freedom of the modular robots increases the robot's mechanical capability and control complexity as well as increasing the diversity of the functionality of the robots.

As it can be understood from the literature, modular robots have been examined mostly for the research purpose until today. However, there are also some studies that have been made in last years showing the application of modular robots for industrial purposes. G. J. Hamlin and A. C. Sanderson designed a modular robotic system for

industrial applications. They made two types of modules; joint modules and they simulated a SCARA robot by using these modules [16]. Xinan Pan, Hongguang Wang and Yong Jiang developed a calibration method for modular robot manipulators in order to overcome machining errors of modules and assembly errors between modules [17]. Andrea Giusti and Matthias addressed the problem of controlling reconfigurable robot manipulators which are made of heterogeneous modules. They developed an automatic centralized controller to synthesize model-based control laws. [18] A. Valente proposed a configuration algorithm for reconfigurable robots which gives optimal configurations for the given task in terms of statics, kinematics and dynamics. [19]

All of these studies show that a well-designed modular robot system can be used to provide production diversity and efficiency in flexible automation assembly lines.

In addition, open source software is needed to take advantage of these modular robot manipulators and in particular to provide cost-effective solutions. To this end, ROS provides a suitable open source software framework. Its open and modular structure best fits to the modular robot manipulator concept.

ROS developed for robot applications is shown up as a new approach for robot programming. Especially modular structure of the ROS is an important reason for selecting it to be used in modular robot applications [20]. In one of the studies that has been made on this area, designing and programming of a robotic surgical device was realized by using ROS [21]. In another study, programming of a hybrid “pick and place” robot was realized by using ROS libraries [22]. In another study, the problem of the trajectory following of an unmanned air vehicle was solved by using ROS [23]. In addition to these, there are a lot of ROS applications in the literature [24] [25].

Modularity in the software architecture is also a research topic in this field. The unified software frameworks for modular robot manipulators developed in [26] and [27] but these frameworks have not yet been accepted as a standard. In this thesis ROS was used to ensure modularity in software architecture.

Contribution of the research to literature is the new design of the modular robot manipulator and the open source software-based control for modular robot manipulator.

2. CONCEPTUAL DESIGN OF THE MODULES

In this thesis, the modularity of the links comes with their structures. This means that in order to design R – R type modules, we need the minimum specifications related with their structures. Those specifications come from the design constraints. Although this seems to be a disadvantage of the modularity, with the assembly of more than two modules different sort of configurations can be obtained; hence, the capacity of the modular robot manipulator increases.

Throughout this thesis, maximum reachability constraint is determined as a constraint and modular structure is determined to be accomplished with adjustable twist angle. According to this constraint, conceptual modular robot manipulator is created.

In order to show the procedural approach, a case study is utilized in section 2.3.

2.1 Module Types and Structures

Modular robot manipulator is made of three types of modules.

- Active module
- Base module
- Gripper module

Active module consists of housing part for the actuator and adjustable coupling part for module to module assembly. By means of its reconfigurable coupling part, it allows to create different robot configurations.

Base module consists of stationary mounting points for the active modules. It allows to assembly active modules in different configurations. It is responsible for creating base frame between world frame and robot frame in the robot manipulator.

Gripper module which is appended to the active modules, is an end effector tool for modular robot manipulator. It is responsible for performing pick and place actions in the robot manipulator.

In the following chapters, each module types and its kinematic properties are discussed.

2.1.1 Active module

Active module is made of two parts; housing part and coupling part. In Figure 2.1, active module and its components are shown.

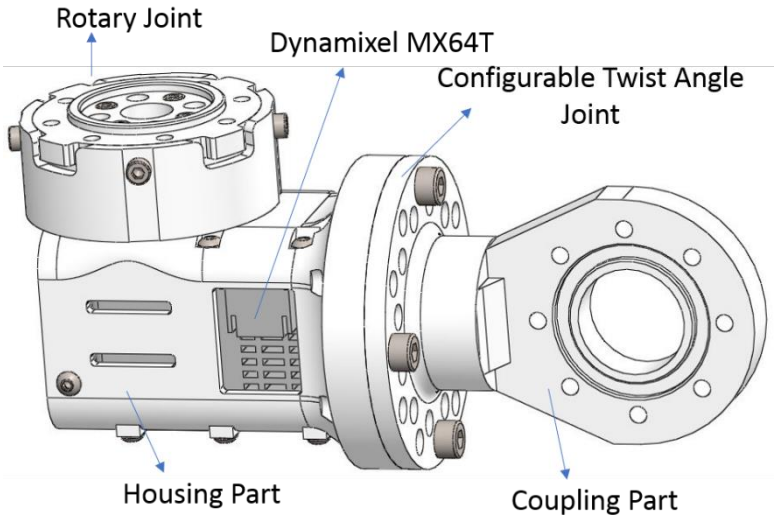


Figure 2.1 : Active module.

Housing part includes a DYNAMIXEL MX64 actuator which is creating moving joints structure of the modular robot manipulator. By means of its mounting holes, actuator is fixed to the module with bolts. The shaft of the actuator is bedding to the module by means of radial and needle roller bearings. Rotary joint which is mounted on the shaft has mounting holes for module to module assembly. Housing part of the active module is given in Figure 2.2.

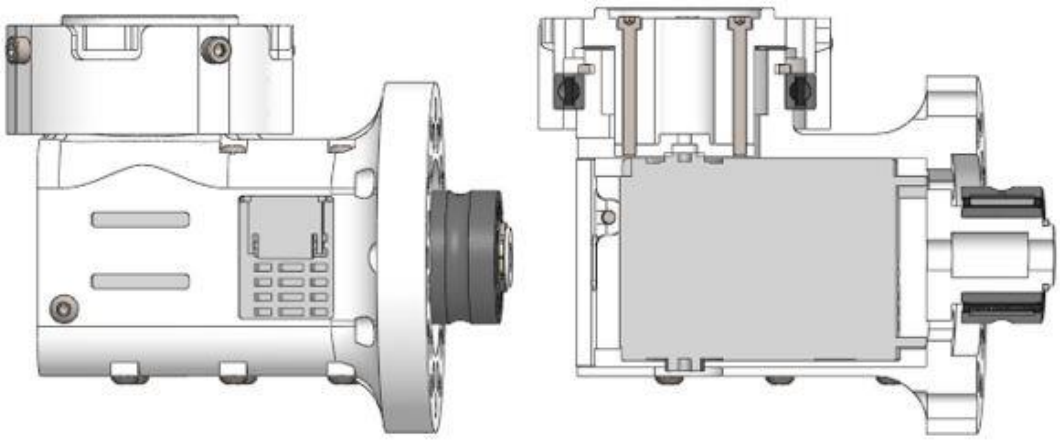


Figure 2.2 : Active Module Housing Part.

Coupling part creates configurable structure of the modular robot manipulator. By rotating in its longitude axis, twist angle of the consecutive module is configured. It has 12 mounting holes and alignment holes to assemble coupling part to the housing part of the active module. Therefore, the twist angle of the consecutive module can be configured with 30 degrees resolution. Once the twist angle is determined, coupling part is mounted to the housing part with bolts. It has needle bearing for bedding which is utilized in the configuration process. Coupling part of the active module is given in Figure 2.3.

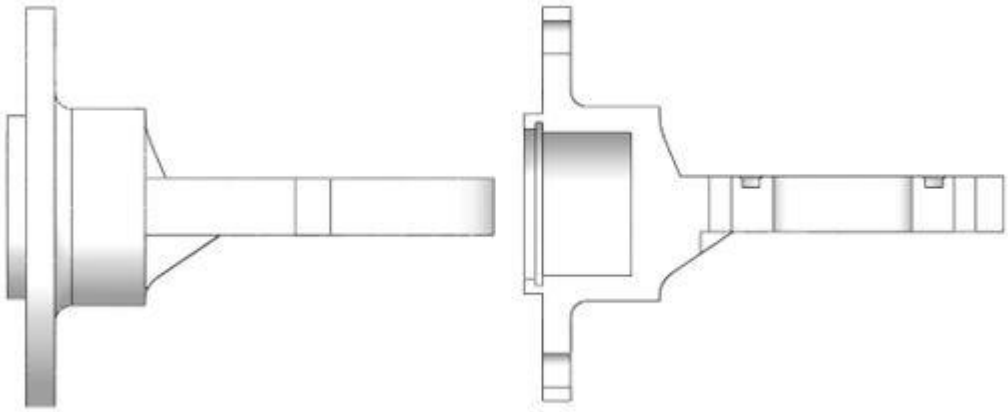


Figure 2.3 : Active Module Coupling Part.

In the both side of the active module, there are holes that enable to connect power and communication cables to the actuator.

2.1.2 Base module

Base module has two separate mounting points to connect active modules. This connection is made by using rectangular coupler part. Active module is mounted to the rectangular coupler part first and then it is mounted to the base module.

In Figure 2.4, base module and coupler part are shown with two possible mounting situations for coupler part.

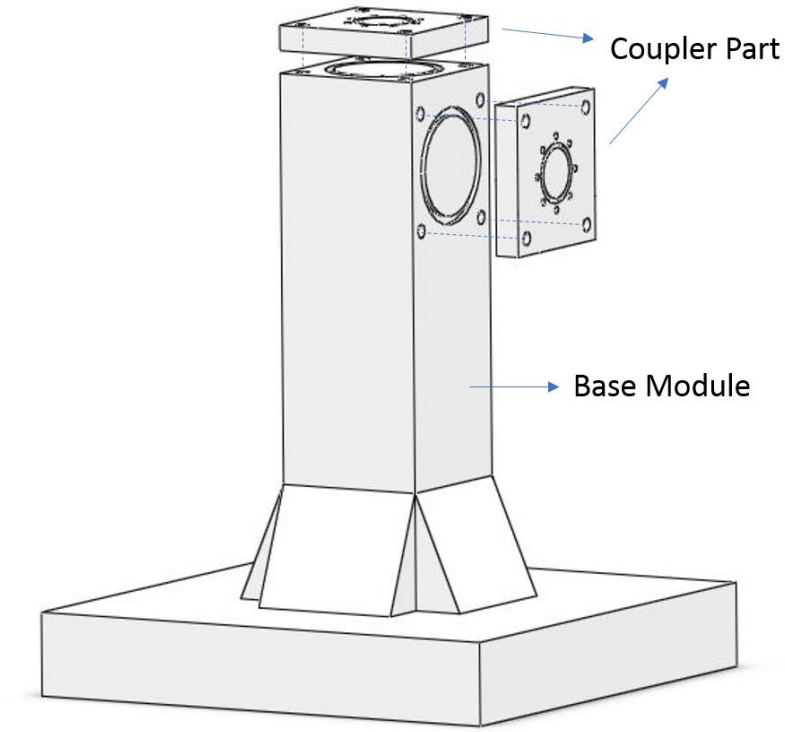


Figure 2.4 : Base module and coupler part.

2.1.3 Gripper module

Gripper module is an end-effector tool for the modular robot manipulator structure. The fingers in the gripper module consist of two symmetrical four-bar mechanisms. These fingers are actuated with a DC servo motor. There are 10 mounting points where the gripper module is connected to the housing part of the active module. In Figure 2.5 gripper module is shown.

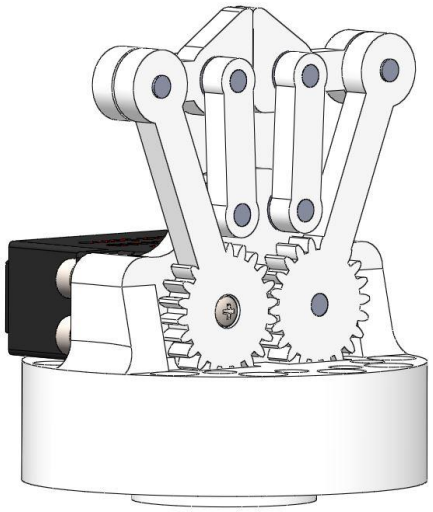


Figure 2.5 : Gripper Module.

2.2 Determination of the DH Parameters

The kinematic model of the robot represents the motion of the robot mechanism regardless of the force and torque that actuates the motion. Kinematic solution allows to determine set of joint variables for the given end-effector pose of the robot and vise-versa.

Denavit-Hartenberg (DH) notation is widely used to describe the kinematic model of a robot. In DH convention, each homogeneous transformation ${}^{i-1}_{i}T$ is represented as a product of four basic transformations:

$$
{}^{\mathbf{i-1}}_{\mathbf{i}}\mathrm{T} = \mathrm{Rot_x}(\alpha_{i-1})\mathrm{D_x}(a_{i-1})R_z(\theta_i)D_z(d_i) =
$$

$$
\begin{bmatrix}
c\theta_i & -s\theta_i & 0 & a_{i-1} \\
s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\
s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\
0 & 0 & 0 & 1
\end{bmatrix} \qquad (2.1)
$$

where $\sin(\theta)$ is abbreviated as $s\theta$ and $\cos(\theta)$ as $c\theta$, Rot_i denotes rotation about the axis i, and D_i denotes translation along axis i.

In the equation 2.1, four DH parameters are described with the following:

- a_{i-1} denotes the link length measured between the Z_{i-1} and Z_i along the X_{i-1} axis.
- α_{i-1} denotes the link twist angle measured between Z_{i-1} and Z_i along the X_{i-1} axis.

- d_i denotes the link offset measured between X_{i-1} and X_i along the Z_i axis.
- θ_i denotes the joint angle measured between X_{i-1} and X_i along the Z_i axis.

As shown in Figure 2.6, where visual representation of DH parameters and coordinate frames allocations are given.

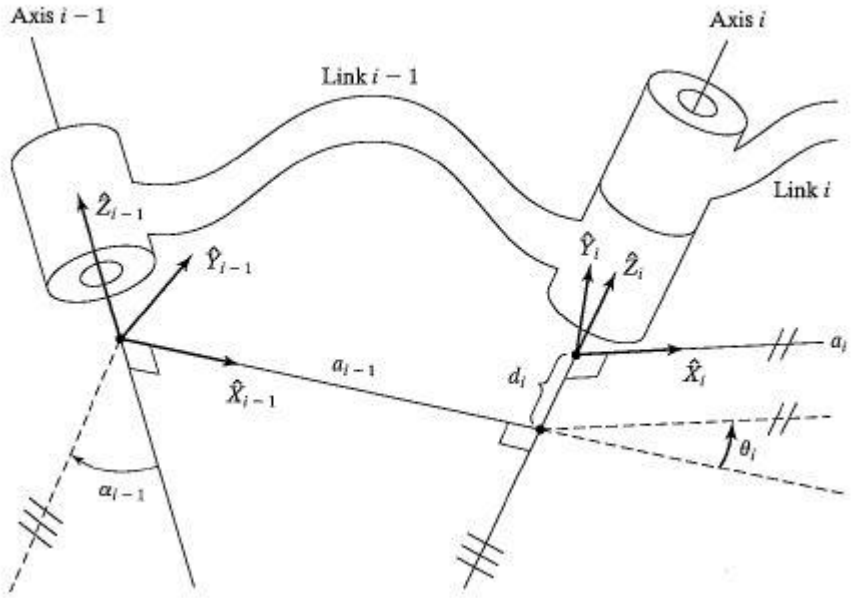


Figure 2.6 : DH convention [28].

Once the DH parameters determined, transformation matrix of the end-effector relative to the base of robot manipulator can be calculated with equation 2.2.

$$\mathbf{{}^0_nT} = \mathbf{{}^0_1T\,{}^1_2T} \ldots \mathbf{{}^{n-1}_nT} \qquad (2.2)$$

Coordinate frames allocations for modular robot manipulator was made as it is shown in Figure 2.7.

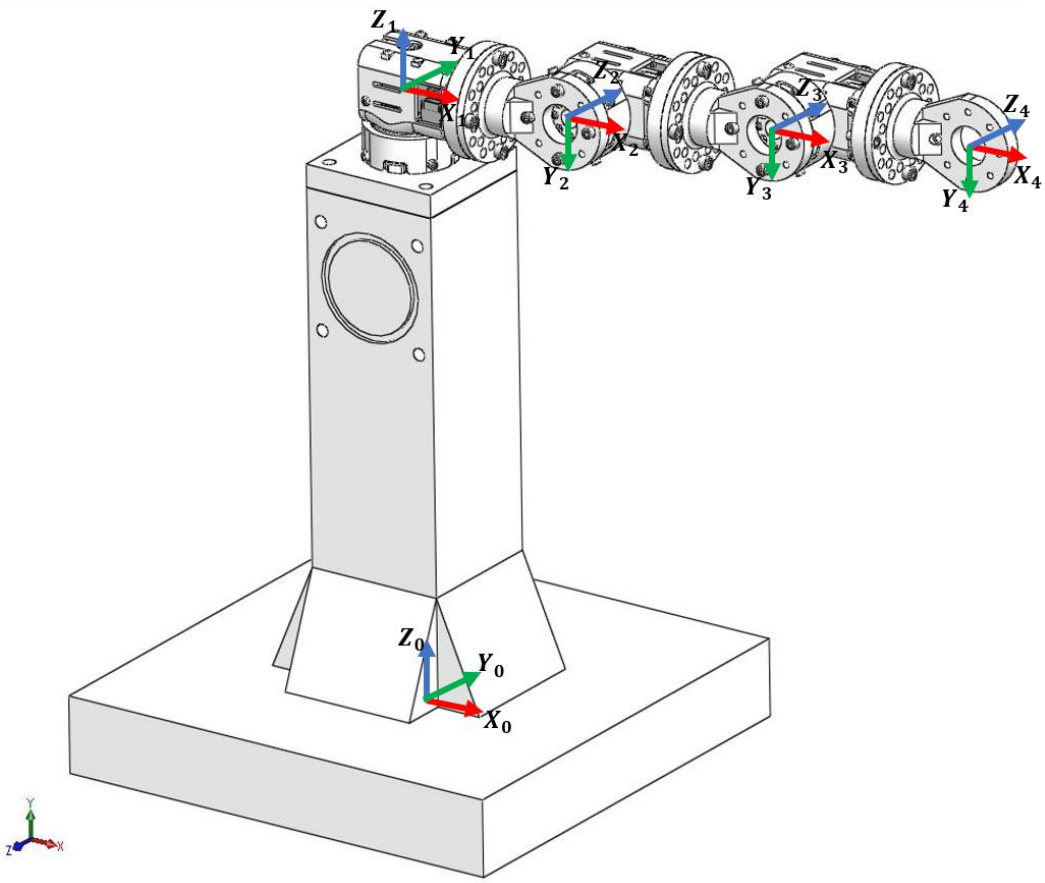


Figure 2.7 : DH coordinate frames.

The DH parameters corresponding to the selected coordinate frames are listed in the Table 2.1.

Table 2.1 : DH Parameters.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	α_0	0	d_1	θ_1
2	α_1	a_1	d_2	θ_2
3	α_2	a_2	d_3	θ_3
4	α_3	a_3	d_4	0°

In the Table 2.1, α variables are configuration parameters, d and a variables are structural parameters and θ variables are joint control parameters for modular robot manipulator. Because all joints are revolute joints, the unit of the θ variables are in radians.

Homogenous transformation matrices of the given 3 DOF modular robot manipulator configuration are obtained as following:

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 c\alpha_0 & c\theta_1 c\alpha_0 & -s\alpha_0 & -s\alpha_0 d_1 \\ s\theta_1 s\alpha_0 & c\theta_1 s\alpha_0 & c\alpha_0 & c\alpha_0 d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.3) $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_1 \\ s\theta_2 c\alpha_1 & c\theta_2 c\alpha_1 & -s\alpha_1 & -s\alpha_1 d_2 \\ s\theta_2 s\alpha_1 & c\theta_2 s\alpha_1 & c\alpha_1 & c\alpha_1 d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.4) $$

$$ {}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_2 \\ s\theta_3 c\alpha_2 & c\theta_3 c\alpha_2 & -s\alpha_2 & -s\alpha_2 d_3 \\ s\theta_3 s\alpha_2 & c\theta_3 s\alpha_2 & c\alpha_2 & c\alpha_2 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.5) $$

$$ {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & c\alpha_3 & -s\alpha_3 & -s\alpha_3 d_4 \\ 0 & s\alpha_3 & c\alpha_3 & c\alpha_3 d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.6) $$

2.3 Structural Design of Modular Robot Manipulator

2.3.1 Structural design of gripper

In the gripper design, usage of the four-bar mechanism was decided and kinematic synthesis of the gripper was proceeded upon 4 bar mechanism.

Synthesis of the four-bar mechanism can be conducted with path generation, body guidance or function generation methods. Due to importance on the orientation of the gripper initial and final positions, the synthesis was conducted with body guidance method. In body guidance method, entire body is guided with desired body poses.

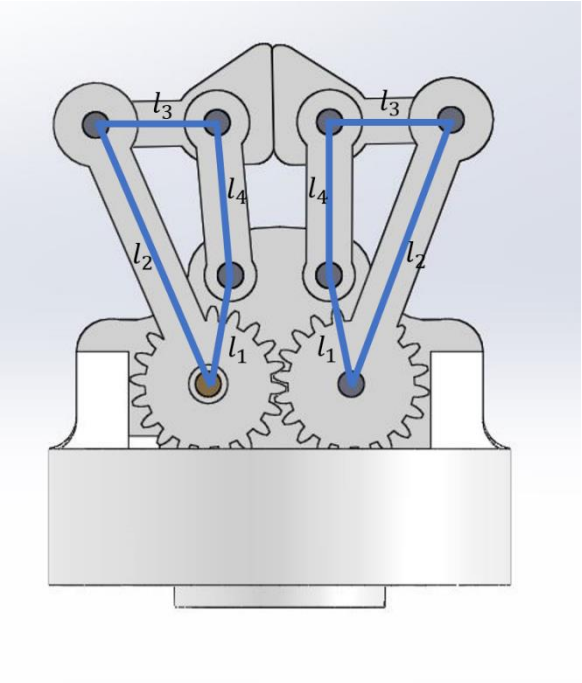


Figure 2.8 : Gripper module design.

In Figure 2.8 gripper design is given. The designed gripper consists of two pieces of four-bar mechanism. Both four-bar fingers are intended to be symmetrical according to origin axis of the gripper in the design phase. Therefore, in order to synthesize the mechanism only one of the fingers is considered.

By splitting the four-bar mechanism as it is given in Figure 2.9 into two 2 DOF mechanisms, link lengths can be obtained.

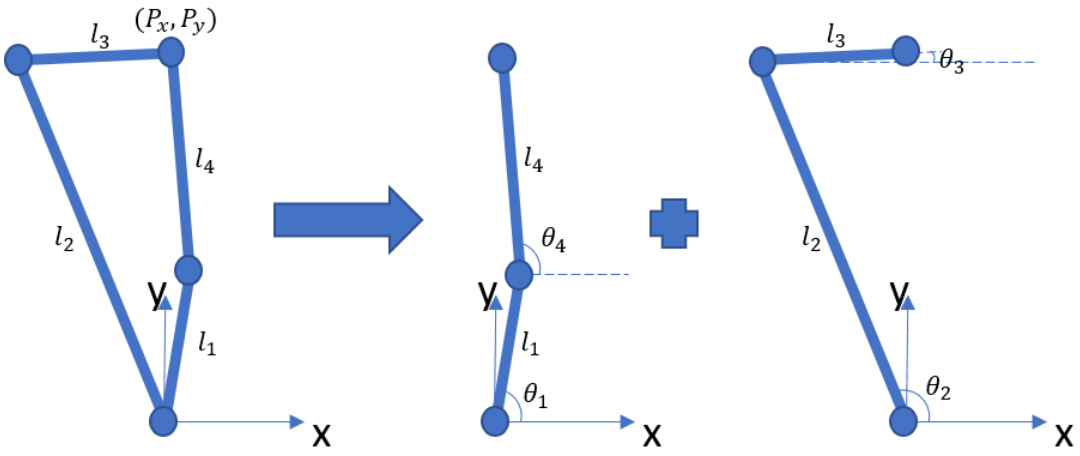


Figure 2.9 : Splitting of the four-bar mechanism.

By using vector-loop equation for the first 2 DOF mechanism, following equations are obtained:

$$l_1 e^{i\theta_1} + l_4 e^{i\theta_4} = P_x + iP_y \quad (2.7)$$

$$l_1 \cos\theta_1 + l_4 \cos\theta_4 = P_x \quad (2.8)$$

$$l_1 \sin\theta_1 + l_4 \sin\theta_4 = P_y \quad (2.9)$$

By rearranging equations 2.8 and 2.9 as leaving θ_1 parameters alone on the left side:

$$l_1 \cos\theta_1 = P_x - l_4 \cos\theta_4 \quad (2.10)$$

$$l_1 \sin\theta_1 = P_y - l_4 \sin\theta_4 \quad (2.11)$$

and by squaring both equations and adding them equation 2.12 is obtained:

$${l_1}^2 = {P_x}^2 + {P_y}^2 - 2P_x l_4 \cos\theta_4 - 2P_y l_4 \sin\theta_4 + {l_4}^2 \quad (2.12)$$

If the obtained equation is divided to the $2l_4$:

$$\frac{({l_4}^2 - {l_1}^2)}{2l_4} + ({P_x}^2 + {P_y}^2)\frac{1}{2l_4} - P_x \cos\theta_4 - P_y \sin\theta_4 = 0 \quad (2.13)$$

is obtained. The equation 2.13 is the objective function of the kinematic synthesis and can be introduced as following:

$$f_{1i}p_1 + f_{2i}p_2 + F_i = 0 \ \ (i = 1, 2) \quad (2.14)$$

where p_k are constant coefficients, i is number of poses, f_{ki} are linearly independent continuous function of the motion variables:

$$F_i = -P_x \cos\theta_4 - P_y \sin\theta_4 \quad (2.15)$$

$$p_1 = \frac{({l_4}^2 - {l_1}^2)}{2l_4} \quad (2.16)$$

$$f_{1i} = 1 \quad (2.17)$$

$$p_2 = \frac{1}{2l_4} \quad (2.18)$$

$$f_{2i} = ({P_x}^2 + {P_y}^2) \quad (2.19)$$

When the 2.14 is represented in matrix form, we obtain:

$$\begin{bmatrix} f_{11} & f_{21} \\ f_{12} & f_{22} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} -F_1 \\ -F_2 \end{bmatrix} \quad (2.20)$$

In the equation 2.20, p_1 and p_2 coefficients are calculated with the following equation:

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} f_{11} & f_{21} \\ f_{12} & f_{22} \end{bmatrix}^{-1} \begin{bmatrix} -F_1 \\ -F_2 \end{bmatrix} \quad (2.21)$$

After the p_1 and p_2 coefficients are calculated, l_1 and l_4 link lengths are calculated as:

$$l_4 = \frac{1}{2p_2} \quad (2.22)$$

$$l_1 = \sqrt{({l_4}^2 - 2p_1 l_4)} \quad (2.23)$$

By applying the same steps for the second 2DOF structure, l_2 and l_3 link lengths are calculated. By using vector-loop equation following equations are obtained:

$$\overrightarrow{l_2} + \overrightarrow{l_3} = \vec{P} \quad (2.24)$$

$$l_2 \cos\theta_2 + l_3 \cos\theta_3 = P_x \quad (2.25)$$

$$l_2 \sin\theta_2 + l_3 \sin\theta_3 = P_y \quad (2.26)$$

By rearranging equations 2.25 and 2.26 as leaving θ_2 parameters alone on the left-hand side we obtain:

$$l_2 \cos\theta_2 = P_x - l_3 \cos\theta_3 \quad (2.27)$$

$$l_2 \sin\theta_2 = P_y - l_3 \sin\theta_3 \quad (2.28)$$

and by squaring both equations and adding them we get:

$${P_x}^2 + {P_y}^2 - 2P_x l_3 \cos\theta_3 - 2P_y l_3 \sin\theta_3 + {l_3}^2 - {l_2}^2 = 0 \quad (2.29)$$

The equation 2.29 is expressed as a polynomial function:

$$F_i + f_{1i}P_1 + f_{2i}P_2 + f_{3i}P_3 = 0 \ \ (i = 1, 2) \quad (2.30)$$

where,

$$F_i = -P_x \cos\theta_3 - P_y \sin\theta_3 \quad (2.31)$$

$$p_1 = \frac{({l_3}^2 - {l_2}^2)}{2l_3} \quad (2.32)$$

$$f_{1i} = 1 \quad (2.33)$$

$$p_2 = \frac{1}{2l_3} \quad (2.34)$$

$$f_{2i} = ({P_x}^2 + {P_y}^2) \quad (2.35)$$

When the equation 2.30 is represented in matrix form, we have:

$$\begin{bmatrix} f_{11} & f_{21} \\ f_{12} & f_{22} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} -F_1 \\ -F_2 \end{bmatrix} \quad (2.36)$$

In the equation 2.36, p_1 and p_2 coefficients are calculated with the following equation:

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} f_{11} & f_{21} \\ f_{12} & f_{22} \end{bmatrix}^{-1} \begin{bmatrix} -F_1 \\ -F_2 \end{bmatrix} \quad (2.37)$$

After the p_1 and p_2 coefficients are calculated, l_2 and l_3 link lengths are calculated as:

$$l_3 = \frac{1}{2p_2} \quad (2.38)$$

$$l_2 = \sqrt{({l_3}^2 - 2p_1 l_3)} \quad (2.39)$$

In the gripper design two precision points were chosen as following:

P1(x = -0.019.98m, y = 0.020.05m, θ_3 = 0 rad, θ_4 = 3.0018 rad)

P2(x = 0.002.36m, y = 0.040.36m, θ_3 = 0 rad, θ_4 = 1.6158 rad)

The link lengths according to the P1 and P2 precision points were computed by applying the discussed procedure above:

l_1 = 0.0171024 m, l_2 = 0.0435236 m

l_3 = 0.0186503 m, l_4 = 0.0236277 m

2.3.2 Case study for R-R type modular robot manipulator

In this section, it is aimed to determine structural parameters α_1, d_1 and d_2 of 2 DOF modular robot manipulator configuration which is given in Figure 2.10 for a given task space path.

DH parameters of the robot is given in Table 2.2 as parametric symbols.

Table 2.2 : DH parameters of the robot which is given in Figure 2.10.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	α_0	a_0	d_1	θ_1
2	α_1	a_1	d_2	θ_2
3	α_2	a_2	d_3	0

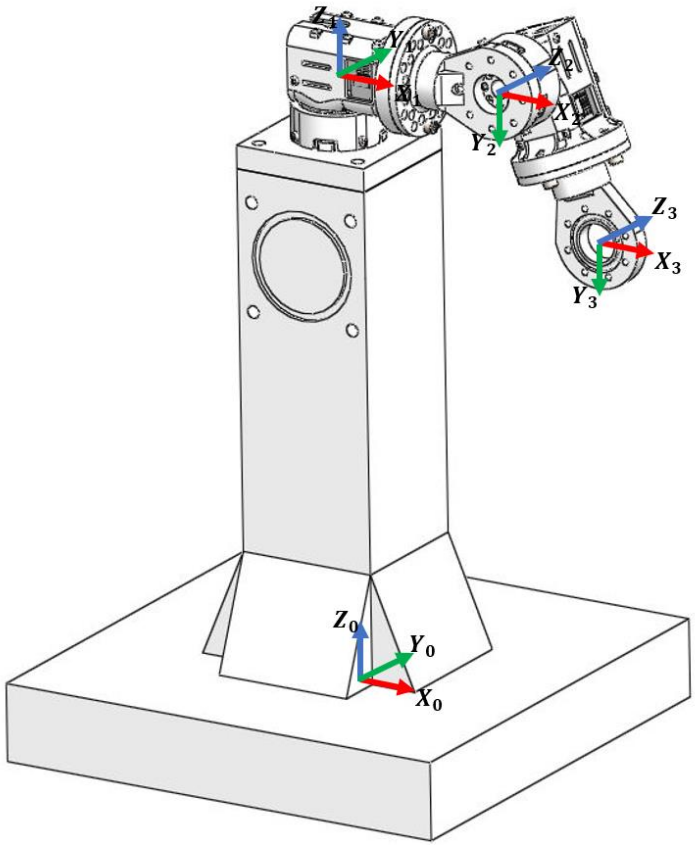


Figure 2.10 : 2 DOF modular robot manipulator.

As it will be discussed detailed in section 3.1, forward kinematics solutions for the 2 DOF modular robot manipulator configuration is obtained as following:

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & a_0 \\ s\theta_1 c\alpha_0 & c\theta_1 c\alpha_0 & -s\alpha_0 & -s\alpha_0 d_1 \\ s\theta_1 s\alpha_0 & c\theta_1 s\alpha_0 & c\alpha_0 & c\alpha_0 d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.40) $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_1 \\ s\theta_2 c\alpha_1 & c\theta_2 c\alpha_1 & -s\alpha_1 & -s\alpha_1 d_2 \\ s\theta_2 s\alpha_1 & c\theta_2 s\alpha_1 & c\alpha_1 & c\alpha_1 d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.41) $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & c\alpha_2 & -s\alpha_2 & -s\alpha_2 d_3 \\ 0 & s\alpha_2 & c\alpha_2 & c\alpha_2 d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.42) $$

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \quad (2.43) $$

Position part of the homogenous transformation matrix ${}^{0}_{3}T$ is obtained with the following equations:

$$ x = c_1\beta - s_1\gamma + a_0 \quad (2.44) $$

$$ y = s_1 c_{\alpha_0}\beta + c_1 c_{\alpha_0}\gamma - s_{\alpha_0}\tau - s_{\alpha_0} d_1 \quad (2.45) $$

$$ z = s_1 s_{\alpha_0}\beta + c_1 s_{\alpha_0}\gamma + c_{\alpha_0}\tau + c_{\alpha_0} d_1 \quad (2.46) $$

where,

$$ \beta = c_2 a_2 + s_2 s_{\alpha_2} d_3 + a_1 \quad (2.47) $$

$$ \gamma = s_2 c_{\alpha_1} a_2 - c_2 c_{\alpha_1} s_{\alpha_2} d_3 - s_{\alpha_1} c_{\alpha_2} d_3 - s_{\alpha_1} d_2 \quad (2.48) $$

$$ \tau = s_2 s_{\alpha_1} a_2 - c_2 s_{\alpha_1} s_{\alpha_2} d_3 + c_{\alpha_1} c_{\alpha_2} d_3 + c_{\alpha_1} d_2 \quad (2.49) $$

and in the equations, s_i represents $sine(i)$, c_i represents $cosine(i)$, s_{α_i} represents $sine(\alpha_i)$, c_{α_i} represents $cosine(\alpha_i)$.

Equation 2.44 is rearranged with the following transformation:

$$ x - a_0 = c_1\beta - s_1\gamma \quad (2.50) $$

By squaring and adding equations 2.50, 2.42 and 2.43 following equations are obtained:

$$(x - a_0)^2 + y^2 + z^2 = \beta^2 + \gamma^2 + \tau^2 + {d_1}^2 + 2\tau d_1 \quad (2.51)$$

$$\beta^2 + \gamma^2 + \tau^2 = {a_2}^2 + {a_1}^2 + {d_3}^2 + {d_2}^2 + 2c_{\alpha_2} d_3 d_2 + 2c_2 a_2 a_1 + 2s_2 s_{\alpha_2} d_3 a_1 \quad (2.52)$$

$$(x - a_0)^2 + y^2 + z^2 = {a_2}^2 + {a_1}^2 + {d_3}^2 + {d_2}^2 + 2c_{\alpha_2} d_3 d_2 + 2c_2 a_2 a_1 + 2s_2 s_{\alpha_2} d_3 a_1 + 2s_2 s_{\alpha_1} a_2 d_1 - 2c_2 s_{\alpha_1} s_{\alpha_2} d_3 d_1 + 2c_{\alpha_1} c_{\alpha_2} d_3 d_1 + 2c_{\alpha_1} d_2 + {d_1}^2 \quad (2.53)$$

When the equations are reordered as:

$$As_2 + Bc_2 = C \quad (2.54)$$

where,

$$A = 2s_{\alpha_1} a_2 d_1 + 2s_{\alpha_2} d_3 a_1 \quad (2.55)$$

$$B = 2a_2 a_1 - 2s_{\alpha_1} s_{\alpha_2} d_3 d_1 \quad (2.56)$$

$$C = (x - a_0)^2 + y^2 + z^2 - {a_2}^2 - {a_1}^2 - {d_3}^2 - {d_2}^2 - 2c_{\alpha_2} d_3 d_2 - 2c_{\alpha_1} c_{\alpha_2} d_3 d_1 - 2c_{\alpha_1} d_2 - {d_1}^2 \quad (2.57)$$

By using trigonometric equation in the equation 2.58, θ_2 is obtained as following:

$$\theta_2 = Atan2(A, B) \pm Atan2(\sqrt{A^2 + B^2 - C^2}, C) \quad (2.58)$$

By substituting θ_2 into equation 2.53, objective function which depends on structural parameters of the modular robot manipulator is obtained as follows:

$$obj(x, y, z, a_0, a_1, a_2, \alpha_0, \alpha_1, \alpha_2, d_1, d_2, d_3) = (x - a_0)^2 + y^2 + z^2 - \Big[{a_2}^2 + {a_1}^2 + {d_3}^2 + {d_2}^2 + 2c_{\alpha_2} d_3 d_2 + 2\cos\Big(Atan2(-B, A) \pm Atan2\big(\sqrt{A^2 + B^2 - C^2}, C\big)\Big) a_2 a_1 + 2\sin\Big(Atan2(-B, A) \pm Atan2\big(\sqrt{A^2 + B^2 - C^2}, C\big)\Big) s_{\alpha_2} d_3 a_1 + 2\sin\Big(Atan2(-B, A) \pm Atan2\big(\sqrt{A^2 + B^2 - C^2}, C\big)\Big) s_{\alpha_1} a_2 d_1 - \quad (2.59)$$

$$2\cos\left(Atan2(-B,A) \pm Atan2\left(\sqrt{A^2+B^2-C^2},C\right)\right) s_{\alpha_1} s_{\alpha_2} d_3 d_1 + 2c_{\alpha_1} c_{\alpha_2} d_3 d_1 + 2c_{\alpha_1} d_2 + {d_1}^2 \Big] = 0$$

In this case study, a_0, a_1, a_2, d_3, α_0 and α_2 are taken as constant parameters in this procedure.

$$\begin{aligned} a_0 &= 0 \\ a_1 &= a_2 = 0.112 \\ d_3 &= 0.04131m \\ a_0 &= 0\, rad \\ a_2 &= -\pi \end{aligned} \qquad (2.60)$$

The optimized structural parameters α_1, d_1 and d_2 are bounded as follows:

$$\begin{aligned} 0 < \alpha_1 &< 2\pi \\ 0.427m < d_1 &< 0.527m \\ 0.04131m < d_2 &< 0.1m \end{aligned} \qquad (2.61)$$

α_1, d_1 and d_2 parameters are sampled within their determined limit values and by sweeping joint angles θ_1 and θ_2 between 0 and 2π for each sample, modular robot manipulator's possible workspace is created as in Figure 2.11.

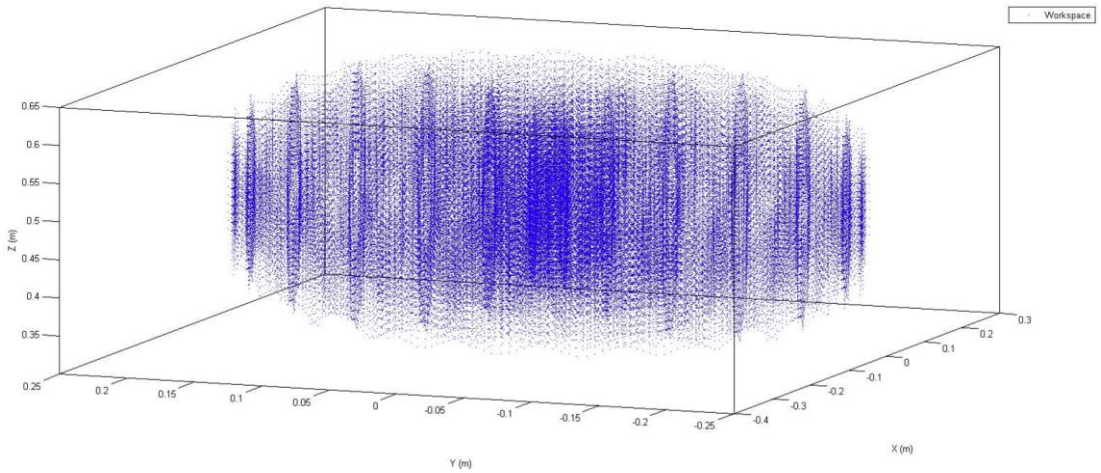


Figure 2.11 : Possible workspace of the modular robot manipulator.

Inside the workspace, 13 random points selected as a task to be accomplished as it is given in Figure 2.12.

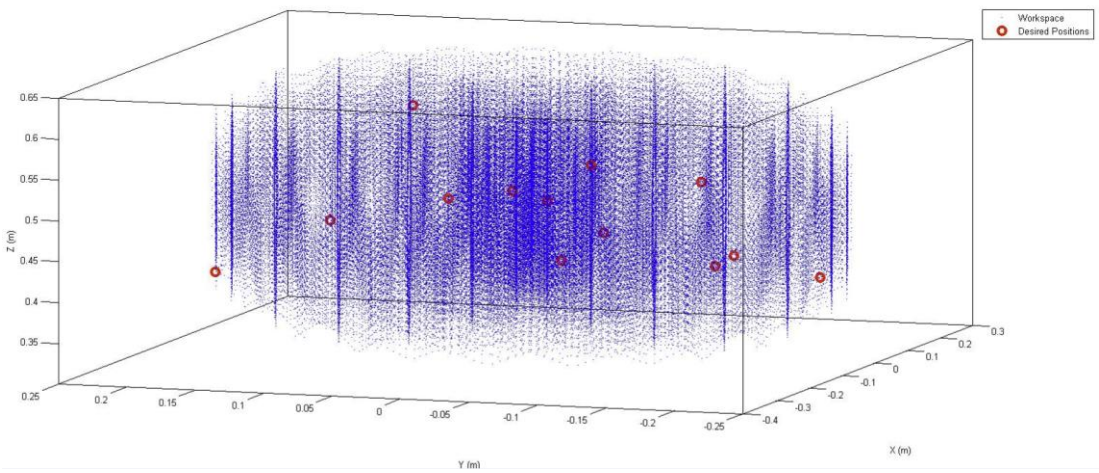


Figure 2.12 : Selected task points inside the possible workspace of the modular robot manipulator.

By using x, y and z coordinates of the selected path, α_1, d_1 and d_2 structural parameters which makes the root mean square error of the objective function minimum was searched via brute force method.

As a result of the brute force search, minimum root mean square error is obtained as 1.2×10^{-2} m with the following structural parameters:

$$\alpha_1 = 1.79\, rad$$

$$d_1 = 44.54 \times 10^{-2}\, m$$

$$d_2 = 52.1 \times 10^{-3}\, m$$

Workspace of the modular robot manipulator with the found structural parameters is shown in Figure 2.13.

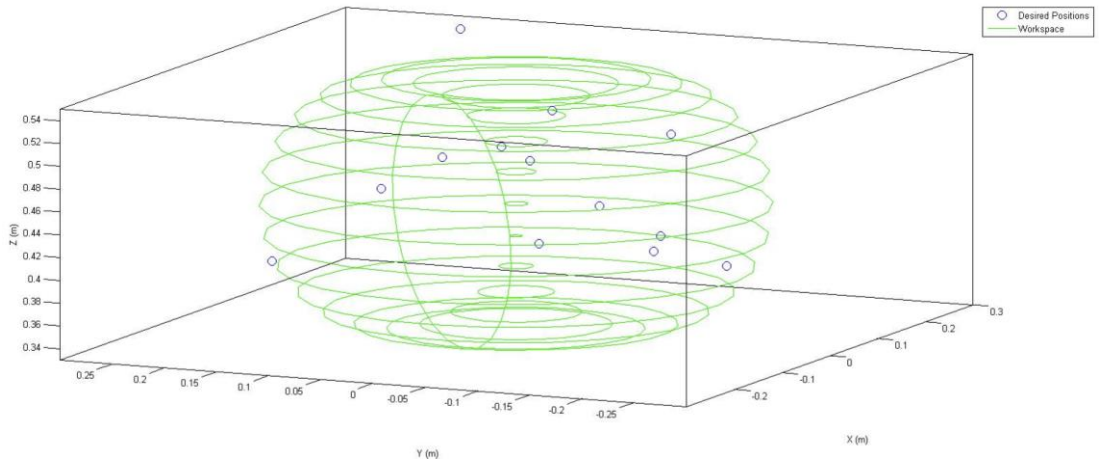


Figure 2.13 : Workspace with calculated α_1, d_1 and d_2 parameters.

3. KINEMATICS AND DYNAMICS OF MODULAR ROBOT MANIPULATOR

3.1 Forward Kinematics

The main aim of the forward kinematics is to determine the pose of the end-effector from given set of joint variables. In order to examine forward kinematics of the modular robot manipulator, 3 DOF configuration given in the Figure 3.1 was considered.

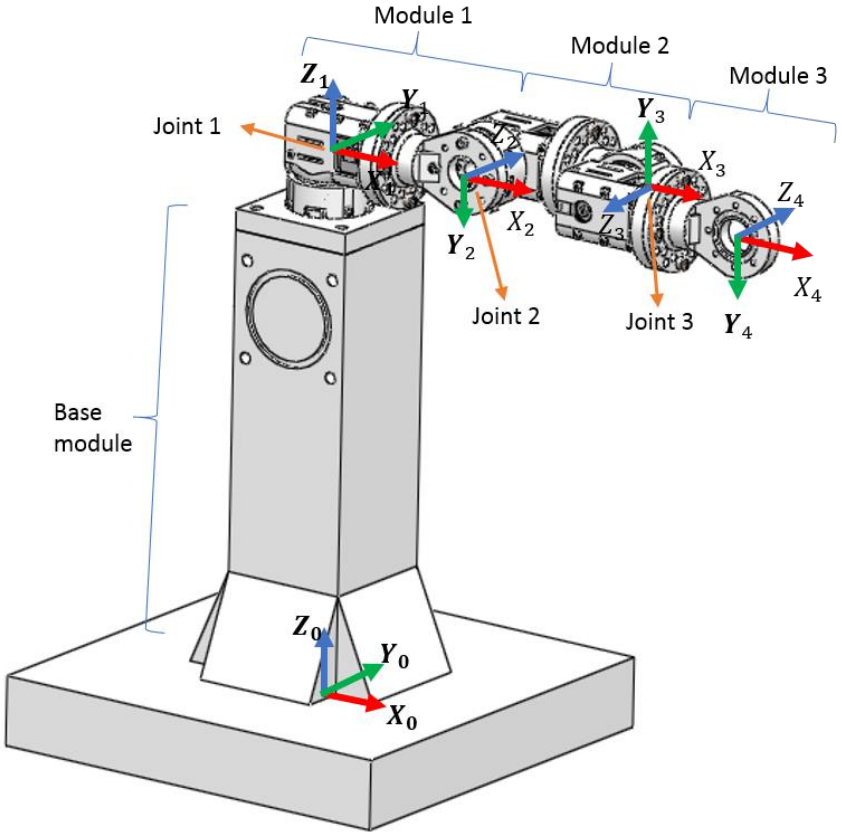


Figure 3.1 : 3 DOF modular robot configuration.

The DH parameters of the given configuration are listed in the Table 3.1.

Table 3.1 : DH parameters of the modular robot in Figure 3.1.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0°	0	0.427m	θ_1
2	-90°	0.112m	0	θ_2
3	180°	0.112m	0.0434m	θ_3
4	180°	0.112m	0.0434m	0°

Following transformation matrixes were found by utilizing DH convention.

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0.427 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.1) $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0.112 \\ 0 & 0 & 1 & 0 \\ -s\theta_2 & -c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2) $$

$$ {}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0.112 \\ -s\theta_3 & -c\theta_3 & 0 & 0 \\ 0 & 0 & -1 & -0.0434 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3) $$

$$ {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & 0.112 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -0.0434 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4) $$

The resultant transformation matrix which denotes the end-effector pose as measured from base frame then is obtained as following:

$$ {}^{0}_{4}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T = \begin{bmatrix} c\theta_1 c(\theta_2 - \theta_3) & -c\theta_1 s(\theta_2 + \theta_3) & -s\theta_1 & p_x \\ s\theta_1 c(\theta_2 - \theta_3) & -s\theta_1 s(\theta_2 - \theta_3) & c\theta_1 & p_y \\ -s(\theta_2 - \theta_3) & -c(\theta_2 - \theta_3) & 0 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.5) $$

where,

$$ p_x = 0.112(c\theta_1 c(\theta_2 - \theta_3)) + 0.112(c\theta_1 c\theta_2) + 0.0434 s\theta_1 + 0.112 c\theta_1 - 0.0434 s\theta_1 \quad (3.6) $$

$$p_y = 0.112(s\theta_1 c(\theta_2 - \theta_3)) + 0.112(s\theta_1 c\theta_2) - 0.0434c\theta_1 + 0.112s\theta_1 + 0.0434c\theta_1 \quad (3.7)$$

$$p_z = -0.112(s(\theta_2 - \theta_3)) - 0.112(s\theta_2) + 0.427 \quad (3.8)$$

In order to verify the solution, initial position of the robot manipulator can be checked from the solution.

When $\theta_1, \theta_2, \theta_3 = 0$ is substituted into the equation 3.6, 3.7 and 3.8 following end-effector position is obtained.

$p_x = 0.336$ m

$p_y = 0$ m

$p_z = 0.427$ m

According to the obtained result, correct location of the end-effector is observed.

3.2 Jacobian Analysis

Jacobian matrices map joint velocities to the cartesian velocities of the end effector. Jacobian matrices were used for determining singular positions of the robot manipulator in the 3.3 section for inverse kinematics procedure and for deriving equation of motion of the robot manipulator in 3.4 section.

By using geometric method, Jacobian matrices for the modular robot manipulator configuration given in Figure 3.2 was determined as follows:

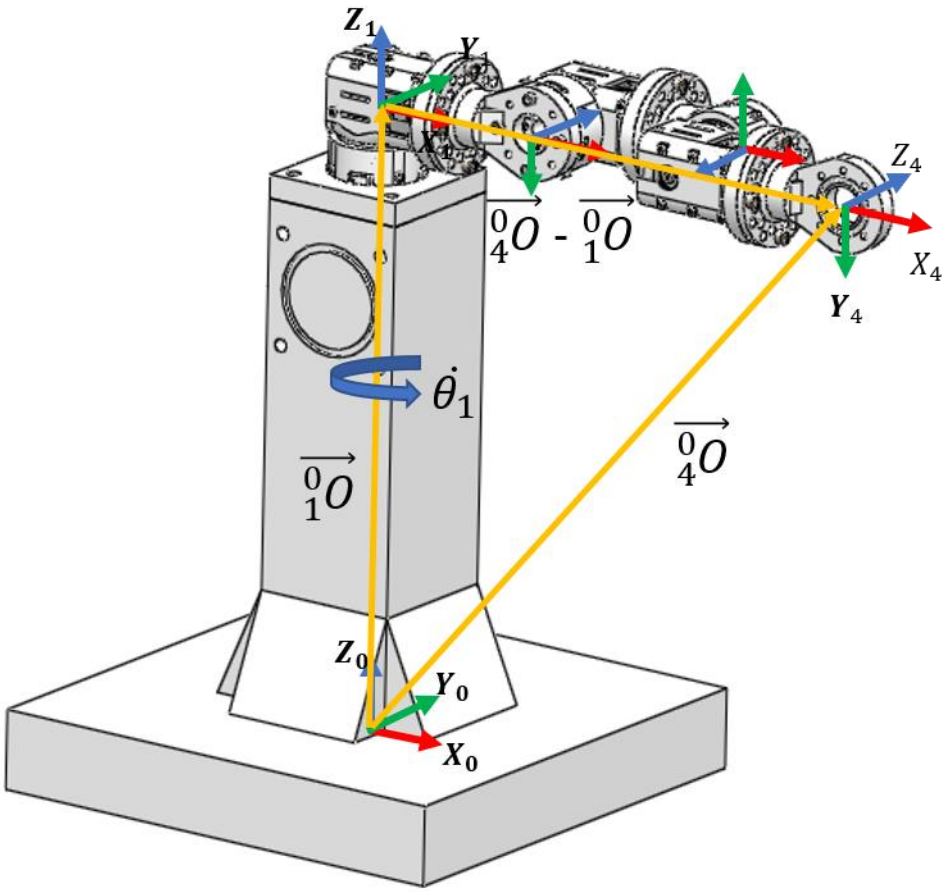


Figure 3.2 : Geometric Jacobian.

$$ {}^0_4J = \begin{bmatrix} ({}^0_1Z \; x \; ({}^0_4O - {}^0_1O))_{3x1} & ({}^0_2Z \; x \; ({}^0_4O - {}^0_2O))_{3x1} & ({}^0_3Z \; x \; ({}^0_4O - {}^0_3O))_{3x1} \\ ({}^0_1Z)_{3x1} & ({}^0_2Z)_{3x1} & ({}^0_3Z)_{3x1} \end{bmatrix} \quad (3.9) $$

Where ${}^0_4O$ represents to end effector position relative to the base frame, ${}^0_xO$ represents the position of the reference frame x relative to the base frame and ${}^0_xZ$ represents the third column of the ${}^0_xT$ transformation matrix.

Singular positions of the robot manipulator are determined by the following condition:

$$ det(J) = 0 \quad (3.10) $$

Because of the Jacobian matrix for the 3 DOF robot manipulator is not in square matrix form, in order to calculate determinant of the Jacobian matrix Singular Value Decomposition method was used.

$$ J = USV^T \quad (3.11) $$

In the equation 3.11, U represents the left singular matrix, S represents singular matrix and V represents right singular matrix. When the Jacobian matrix is decomposed in the U, S and V matrices, determinant of the Jacobian is obtained as follows:

$$det(J) = det(U)\, det(S)\, det(V^T) \qquad (3.12)$$

In the section 7.1.3 validation of the singularity analysis is given.

3.3 Inverse Kinematics

Inverse kinematics is a problem of calculating joints angles from the given end effector cartesian poses.

There is not a unique solution for inverse kinematic problems. In literature there are some methods used for solving inverse kinematic problems [29] and these are mainly categorized in two groups; analytical and numerical solutions. Analytical solutions offer fast and closed form solutions but the solution is not always found for complicated robot kinematic configurations. Contrary to analytical solutions, numerical solutions offer approximate solutions with sufficient accuracy.

Due to its reconfigurable structure of the modular robot manipulator, it is not always possible to find an analytical solution. Therefore, for higher than 3 DOF modular robot manipulator configurations, a numerical method was considered to the inverse kinematic problem.

3.3.1 Inverse kinematic solution with analytic method

In this section, inverse kinematic solution for 3DOF modular robot manipulator is explained. For that purpose, 3 DOF robot manipulator configuration which is given in Figure 3.1 was used.

Forward kinematics equation for the robot manipulator is as the equation 3.5**.** When the equation 3.5 is multiplied by ${}^{0}_{1}T^{-1}$:

$$({}^{0}_{1}T)^{-1}\,{}^{0}_{4}T = ({}^{0}_{1}T)^{-1}\,{}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \qquad (3.13)$$

Because of $({}^{0}_{1}T)^{-1}\,{}^{0}_{1}T$ multiplication is equal to the identity matrix following equation is obtained:

$$({}^{0}_{1}T)^{-1}\,{}^{0}_{4}T = {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \qquad (3.14)$$

By using the equation 3.14 inverse kinematics solution is found. ${}^{0}_{1}T$, ${}^{1}_{2}T$, ${}^{2}_{3}T$, ${}^{3}_{4}T$ and ${}^{0}_{4}T$ transformation matrices were found as equation 3.1, 3.2, 3.3, 3.4 and 3.5 respectively and inverse of the ${}^{0}_{1}T$ matrix is obtained as following

$${}^{0}_{1}T^{-1} = \begin{bmatrix} c\theta_1 & s\theta_1 & 0 & 0 \\ -s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.15)$$

When the multiplication of the equation 3.14 is made for both sides and position parts of the matrices are extracted as,

$$\begin{bmatrix} \dots & \dots & \dots & p_x c\theta_1 + p_y s\theta_1 \\ \dots & \dots & \dots & -p_x s\theta_1 + p_y c\theta_1 \\ \dots & \dots & \dots & p_z - d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \dots & \dots & \dots & a_4 c\theta_{23} + a_3 c\theta_2 + a_2 \\ \dots & \dots & \dots & d_4 - d_3 \\ \dots & \dots & \dots & -a_4 s\theta_{23} - a_3 s\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (3.16)$$

the following equations are obtained:

$$p_x c\theta_1 + p_y s\theta_1 = a_4 c\theta_{23} + a_3 c\theta_2 + a_2 \qquad (3.17)$$

$$-p_x s\theta_1 + p_y c\theta_1 = d_4 - d_3 \qquad (3.18)$$

$$p_z - d_1 = -a_4 s\theta_{23} - a_3 s\theta_2 \qquad (3.19)$$

By using equation 3.18, θ_1 is obtained as follows:

$$\theta_1 = \mathrm{Atan2}(-\mathrm{px}, \mathrm{py}) \pm Atan2\left(\sqrt{(-p_x)^2 + {p_y}^2 - (d_4 - d_3)^2}\,,\ (d_4 - d_3)\right) \qquad (3.20)$$

In order to obtain θ_3, equation 3.17, 3.18 and 3.19 are used. Equation 3.17 is reordered by taking a_2 to the left side of the equation as:

$$p_x c\theta_1 + p_y s\theta_1 - a_2 = a_4 c\theta_{23} + a_3 c\theta_2 \qquad (3.21)$$

By squaring and adding both side of the equations 3.21, 3.18 and 3.19:

$$p_x{}^2 + p_y{}^2 + a_2{}^2 - 2p_x a_2 c\theta_1 - 2p_y a_2 s\theta_1 + (p_z - d_1)^2 = a_4{}^2 + a_3{}^2 + 2a_4 a_3 (c\theta_{23} c\theta_2 + s\theta_{23} s\theta_2) + (d_4 - d_3)^2 \quad (3.22)$$

When the trigonometric equation 3.23 is substituted into the 3.22:

$$(c\theta_{23} c\theta_2 + s\theta_{23} s\theta_2) = c\theta_3 \quad (3.23)$$

$$p_x{}^2 + p_y{}^2 + a_2{}^2 - 2p_x a_2 c\theta_1 - 2p_y a_2 s\theta_1 + (p_z - d_1)^2 = a_4{}^2 + a_3{}^2 + 2a_4 a_3 c\theta_3 + (d_4 - d_3)^2 \quad (3.24)$$

From the above equation 3.24, $c\theta_3$ is obtained is as following:

$$c\theta_3 = \frac{p_x{}^2 + p_y{}^2 + a_2{}^2 - 2p_x a_2 c\theta_1 - 2p_y a_2 s\theta_1 + (p_z - d_1)^2 - a_4{}^2 - a_3{}^2 - (d_4 - d_3)^2}{2a_4 a_3} \quad (3.25)$$

Finally, θ_3 is found with the equation 3.27:

$$k = \frac{p_x{}^2 + p_y{}^2 + a_2{}^2 - 2p_x a_2 c\theta_1 - 2p_y a_2 s\theta_1 + (p_z - d_1)^2 - a_4{}^2 - a_3{}^2 - (d_4 - d_3)^2}{2a_4 a_3} \quad (3.26)$$

$$\theta_3 = Atan2(\pm\sqrt{1 - (k)^2}, k) \quad (3.27)$$

In order to obtain θ_2, equation 3.19 is used. By substituting trigonometric function 3.28 into 3.19, the following equations are obtained**:**

$$s\theta_{23} = (s\theta_2 c\theta_3 - c\theta_2 s\theta_3) \quad (3.28)$$

$$p_z - d_1 = -a_4 s\theta_2 c\theta_3 + a_4 c\theta_2 s\theta_3 - a_3 s\theta_2 \quad (3.29)$$

By reordering the equation 3.29 we have:

$$p_z - d_1 = (-a_4 c\theta_3 - a_3) s\theta_2 + (a_4 s\theta_3) c\theta_2 \quad (3.30)$$

θ_2 then is obtained as follows:

$$\theta_2 = Atan2(-a_4 c\theta_3 - a_3,\ a_4 s\theta_3) \pm Atan2\left(\sqrt{(-a_4 c\theta_3 - a_3)^2 + (a_4 s\theta_3)^2 - (p_z - d_1)^2},\ (p_z - d_1)\right) \quad (3.31)$$

3.3.2 Inverse kinematic solution with numerical method

For higher than 3 DOF modular robot manipulator configurations, a numerical method is considered to the inverse kinematic problem. Among the numerical methods, solving the linear least square problem with Newton-Raphson method was chosen and implemented with KDL library.

Cartesian pose of the end-effector is expressed as a function of set of joint variables:

$$Y_{6x1} = f(q_1, q_2, \dots, q_n) \quad (3.32)$$

where q_i denotes the joint angle and Y denotes the $6x1$ cartesian pose matrix of the end effector.

When it is taken the derivatives of the cartesian pose matrix with respect to joint angles the are taken, the following equations are obtained.

$$\delta y_1 = \frac{\delta f_1}{\delta q_1}\delta q_1 + \frac{\delta f_1}{\delta q_2}\delta q_2 + \cdots + \frac{\delta f_1}{\delta q_n}\delta q_n \quad (3.33)$$

$$\delta y_2 = \frac{\delta f_2}{\delta q_1}\delta q_1 + \frac{\delta f_2}{\delta q_2}\delta q_2 + \cdots + \frac{\delta f_2}{\delta q_n}\delta q_n \quad (3.34)$$

$$\delta y_6 = \frac{\delta f_n}{\delta q_1}\delta q_1 + \frac{\delta f_n}{\delta q_2}\delta q_2 + \cdots + \frac{\delta f_n}{\delta q_n}\delta q_n \quad (3.35)$$

In matrix form Jacobian matrix is obtained with the following:

$$J(q) = \frac{\delta y_i}{\delta q_1} \quad (3.36)$$

Iterative methods use the Jacobian matrix which is a linear approximation of a differentiable function near a given point. Relation between change of the end effector pose and the change of the joint angles are obtained with this linear approximation.

$$\Delta Y = J(q)\Delta q \quad (3.37)$$

From this equation change of the joint angles is expressed as follows:

$$\Delta q = J(q)^{-1}\Delta Y \quad (3.38)$$

Linear function of the forward kinematics equation can be obtained with Taylor expansion:

$$f(q) \approx f_{linear}(q) = f(q) + J(q)\Delta q \quad (3.39)$$

By using this linear approximation to the forward kinematics, the aim of the inverse kinematic is to find change of the joint angles with respect to the change of the end-effector pose.

Error function which is tried to be minimized iteratively, is expressed with the following equation:

$$E = |D - f_{linear}(q)| \quad (3.40)$$

In 3.40, E denotes the error matrix, D denotes desired pose matrix of the end effector and $f_{linear}(q)$ represents forward kinematic solution matrix obtained with linear approximation.

KDL library includes inverse kinematic solvers which implements numerical inverse kinematic solution with Newton-Raphson method. KDL library is chosen for kinematic solver because it is allowed to directly being used with ROS environment. In the numerical solver, KDL calculates the inverse of the Jacobian matrix with the singular value decomposition method in order to handle with non-square Jacobian matrices which it depends to the degrees of freedom of the robot.

In the Figure 3.3 flow chart of the KDL numerical solver is given.

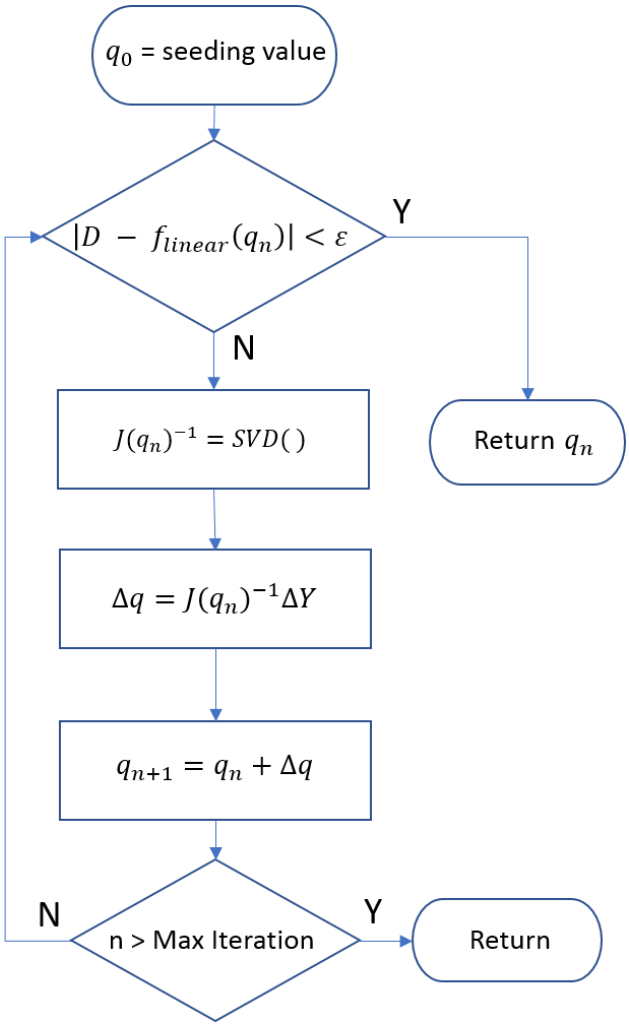


Figure 3.3 : KDL numerical inverse kinematic flow chart.

Performance of the numerical inverse kinematic solver was evaluated with different DOF's and robot configurations and results were given in the section 7.1.1.

3.4 Dynamics Modeling of the Modular Robot Manipulator

Dynamic model of the robot represents the relation between the joint actuator torques and the resulting motion. An accurate dynamics model of the robot manipulator is vital for the design of motion control systems, the analysis of mechanical design and simulation of manipulator motion. Generally, the dynamic performance of the robot depends on implementing an effective control algorithm and obtaining an appropriate dynamic model of the robot.

Robot manipulator dynamics model is commonly used in the form:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau \qquad (3.41)$$

where q denotes the joint angles vector; M(q) is the symmetric, bounded, positive definite inertia matrix with size of nxn and n is the degree of freedom of the robot arm; $C(q,\dot{q})$ denotes the Coriolis and Centrifugal force; G(q) is the gravitational force, and τ is the actuator torques vector. This equation then can be used to calculate either forward dynamics, where the manipulator motion is calculated based on a vector of applied torques, or the inverse dynamics where the torques for a given set of joint parameters can be calculated.

There are two commonly used methods for obtaining dynamics model of the robot; Lagrange-Euler method and Recursive Newton-Euler method.

The Lagrange-Euler method depends on calculating the kinetic and potential energies of a rigid body system. This method provides the closed form of the robot dynamics, and it can be applicable to the analytical computation of robot dynamics and it can be used to design joint space control strategies. The Lagrange-Euler method can also be used for forward and inverse dynamic calculation, but it requires high computational load because of the large number of coefficients in the inertia matrix and $C(q,\dot{q})$ matrix.

The Newton-Euler method depends on a balance of all the forces acting on the link of the manipulator. This method constitutes a set of equations with a recursive solution. A forward recursion of the process includes obtaining link velocities and accelerations, and backward recursion includes obtaining the forces and torques acting on each part of the robot manipulator. This recursion structure of the method reduces the computational load of the forward and dynamics calculations; therefore, it allows implementation of real time control methods of robot manipulators.

3.4.1 Dynamics modeling procedure

In order to obtain dynamics model of the modular robot manipulator, Lagrange-Euler method was implemented. In this section, implementation of the Lagrange-Euler method is presented for analyzing the dynamics parameters in controller design.

Nomenclatures that are used in this section are given in the Table 3.2.

Table 3.2 : Nomenclatures.

n	Degrees of freedom of the manipulator
q, $\dot{q}, \ddot{q}$	Vector of position (rad), angular velocity(rad/s) and acceleration(rad/s^2), respectively
a, d, α, θ	Variables denoting the Denavit-Hartenberg parameters
Ii	Inertia tensor of link i (kg/m^2)
m	Mass of link (kg)
$\bar{r}_i$	Center of mass link I (m)
${}^{i}_{j}T$	Homogenous transform matrix from link i to j

3.4.1.1 Active module mass properties

In active module design, ABS material was chosen at the design phase. In Figure 3.4, coordinate frame which is used to measure mass properties is shown.

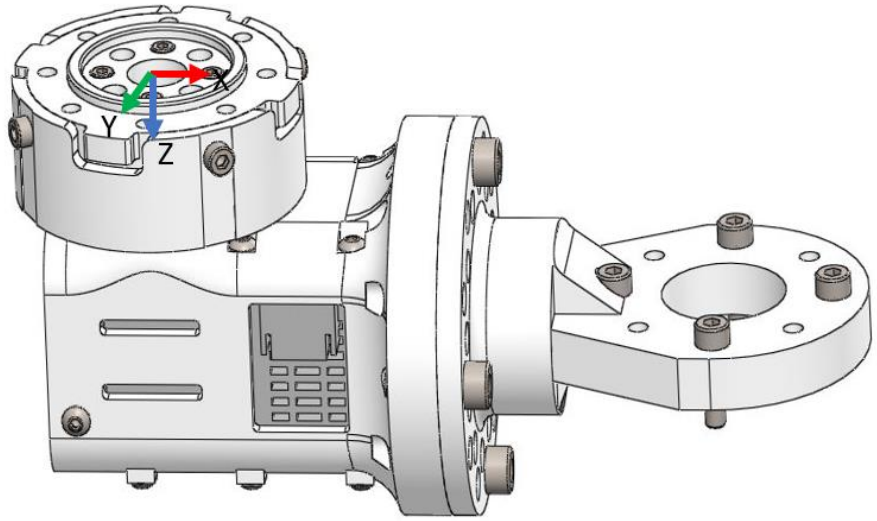


Figure 3.4 : Active module output coordinate frame.

Center of mass

Active module mass was obtained as 0.35 kg from SolidWorks and the center of mass of the module measured from the output coordinate frame is listed in the Table 3.3**.** The units are meters.

Table 3.3 : Center of mass of the active module.

$\bar{x}$	$\bar{y}$	$\bar{z}$
28.11×10^{-3}	8×10^{-5}	39.59×10^{-3}

Link Inertia Tensors

Moments of inertia taken at the center of mass and aligned with the output coordinate system is listed in the Table 3.4. The units used in the Table 3.4 are kg/m^2.

Table 3.4 : Inertia matrix of the active module which is taken at the center of mass and alligned with the ouput coordinate system.

Ixx	**Iyy**	**Izz**
19.25×10^{-5}	55.36×10^{-5}	48.91×10^{-5}
Ixy	**Iyz**	**Ixz**
-39×10^{-8}	27×10^{-8}	78.11×10^{-6}

3.4.1.2 Gripper module mass properties

In gripper module, ABS material was used and coordinate frame which is used to measure mass properties is given in Figure 3.5.

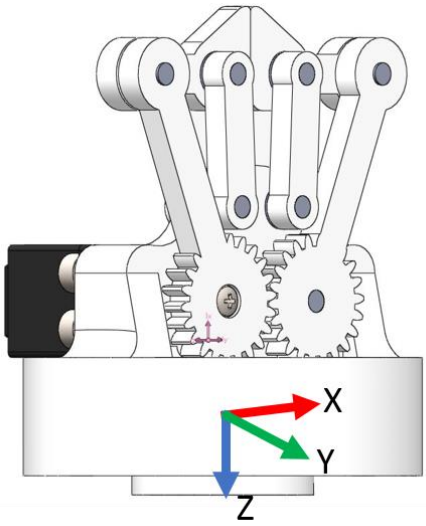


Figure 3.5 : Gripper module output coordinate frame.

Center of mass

Gripper module mass was obtained as 0.15 kg from SolidWorks and the center of mass of the gripper module measured from the output coordinate frame is listed in the Table 3.5**.** The units are meters.

Table 3.5 : Center of mass of the gripper module.

$\bar{x}$	$\bar{y}$	$\bar{z}$
-40.6×10^{-7}	-35.84×10^{-4}	-80.82×10^{-4}

Link Inertia Tensors

Moments of inertia taken at the center of mass and aligned with the output coordinate system is listed in the Table 3.6. The units used in the Table 3.6 are kg/m^2.

Table 3.6 : Inertia matrix of the gripper module which is taken at the center of mass and alligned with the ouput coordinate system.

Ixx	**Iyy**	**Izz**
68.9×10^{-6}	71.1×10^{-6}	66.8×10^{-6}
Ixy	**Iyz**	**Ixz**
0	0	0

3.4.1.3 Base module mass properties

In base module MDF material was used and in the Figure 3.6, coordinate frame which is used to measure mass properties is given.

Center of mass

Base module mass was obtained as 5.68 kg from SolidWorks and the center of mass of the base module measured from the output coordinate frame is listed in the Table 3.7**.** The units are meters.

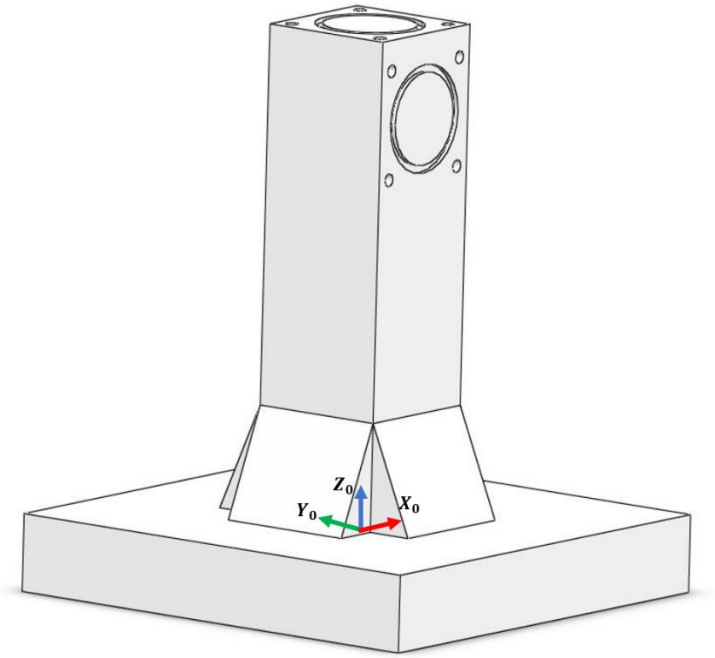


Figure 3.6 : Base module output coordinate frame

Table 3.7 : Center of mass of the base module.

$\bar{x}$	$\bar{y}$	$\bar{z}$
0	28×10^{-6}	95.73×10^{-3}

Link Inertia Tensors

Moments of inertia taken at the center of mass and aligned with the output coordinate system is listed in the Table 3.8. The units used in the Table 3.8 are kg/m^2.

Table 3.8 : Inertia matrix of the gripper module which is taken at the center of mass and alligned with the ouput coordinate system.

Ixx	**Iyy**	**Izz**
93.43×10^{-3}	93.43×10^{-3}	54.41×10^{-3}
Ixy	**Iyz**	**Ixz**
0	34.34×10^{-6}	0

Lagrange-Euler Method

The Lagrange-Euler equations of motion for a conservative system are given by:

$$L(q,\dot{q}) = K(q,\dot{q}) - P(q), \quad \tau = \frac{d}{dt}\frac{\partial L(q,\dot{q})}{\partial \dot{q}} - \frac{\partial L(q,\dot{q})}{\partial q} \qquad (3.42)$$

where K denotes kinetic energy and P denotes the potential of the rigid body.

The kinetic energy of the i'th link is given by:

$$K(q,\dot{q}) = \frac{1}{2}\sum_{i=1}^{n}[(v_i{}^T)m_i v_i + (w_i{}^T)I_i w_i] \quad (3.43)$$

If inertia tensor of the link is located at the center of mass of the link, according to the principles axis theorem inertia tensor is shown as:

$$I_m = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \quad (3.44)$$

Then inertia tensor of the link according to the base frame can be found as:

$$I_i = {}^0_i R I_m ({}^0_i R)^T \quad (3.45)$$

The linear and angular velocities of the link is found by using Jacobian matrix:

$$v_i = J_{v_i}\dot{q} \quad (3.46)$$

$$w_i = J_{w_i}\dot{q} \quad (3.47)$$

Total kinetic energy is obtained is as following:

$$K(q,\dot{q}) = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n}[(J_{v_i}{}^T)m_i J_{v_i} + (J_{w_i}{}^T)I_i J_{w_i}]\dot{q} \quad (3.48)$$

This equation can be rewritten as including manipulator inertia matrix $D(q)$:

$$K(q,\dot{q}) = \frac{1}{2}\dot{q}^T D(q)\dot{q} \quad (3.49)$$

where,

$$D(q) = \sum_{i=1}^{n}[(J_{v_i}{}^T)m_i J_{v_i} + (J_{w_i}{}^T)I_i J_{w_i}] \quad (3.50)$$

The potential energy of the i'th link is given by:

$$P(q) = \sum_{i=1}^{n} m_i g^T h_i \quad (3.51)$$

When kinetic and potential energy equations substituted into equation 3.52**,** dynamic model of the robot obtained as:

$$L(q,\dot{q}) = \frac{1}{2}\dot{q}^T D(q)\dot{q} + mg^T h \quad (3.52)$$

$$\tau_i = \sum_{j=1}^{n} D_{ij}(q)\ddot{q}_j + \sum_{k=1}^{n}\sum_{j=1}^{n} c_{kj}^{i}(q)\dot{q}_k\dot{q}_j + y_i(q) \quad (3.53)$$

$$\tau = D(q)\ddot{q} + C(q,\dot{q}) + G(q) \quad (3.54)$$

3.4.2 Validating dynamic model of the modular robot manipulator

In order to verify the obtained dynamic model, an example joint space trajectory was selected and by performing selected trajectory, results were measured from SolidWorks software and results compared with the calculated torque values by implementing equation 3.54 in the MATLAB. Following joint positions were selected as inputs to the trajectory generation.

$$q_1(t_0) = 0,\ q_1(t_f) = 360^{\circ},\ \dot{q}_1(t_0) = \dot{q}_1(t_f) = 0,\quad \ddot{q}_1(t_0) = \ddot{q}_1(t_f) = 0 \quad (3.55)$$

$$q_2(t_0) = 0,\ q_2(t_f) = -90^{\circ},\ \dot{q}_1(t_0) = \dot{q}_2(t_f) = 0,\quad \ddot{q}_2(t_0) = \ddot{q}_2(t_f) = 0 \quad (3.56)$$

$$q_3(t_0) = 0,\ q_3(t_f) = -90^{\circ},\ \dot{q}_3(t_0) = \dot{q}_3(t_f) = 0,\quad \ddot{q}_3(t_0) = \ddot{q}_3(t_f) = 0 \quad (3.57)$$

In the equations, q_n, $\dot{q}_n$ and $\ddot{q}_n$ denotes to the position, velocity and acceleration of the n'th module at time t, t_0 represents starting time, t_f represents end time which in this case 10 seconds.

In order to create joint space trajectories in SolidWorks, cubic spline interpolation method was selected for given points. Besides, in order to create the same trajectory, a Simulink block was created in MATLAB. In this block, following cubic polynomial equations used for interpolating between points.

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \quad (3.58)$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \quad (3.59)$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t \qquad (3.60)$$

In the equations above, a_0, a_1, a_2 and a_3 coefficients were calculated with the following equations.

$$a_0 = q_n(t_0) \qquad (3.61)$$

$$a_1 = \dot{q_n}(t_0) \qquad (3.62)$$

$$a_2 = \frac{3\left(q_n(t_f) - q_n(t_0)\right)}{{t_f}^2} - \frac{2\dot{q_n}(t_0) + \dot{q_n}(t_f)}{t_f} \qquad (3.63)$$

$$a_3 = \frac{-2\left(q_n(t_f) - q_n(t_0)\right)}{{t_f}^3} + \frac{\dot{q_n}(t_f) + \dot{q_n}(t_0)}{{t_f}^2} \qquad (3.64)$$

After polynomial coefficients were determined, joint position, velocity and acceleration for modules at time sample t were fed to the dynamic model. This block diagram is given in the Figure 3.17.

Trajectory for joint 1 is given in the Figure 3.7, Figure 3.8 and Figure 3.9.

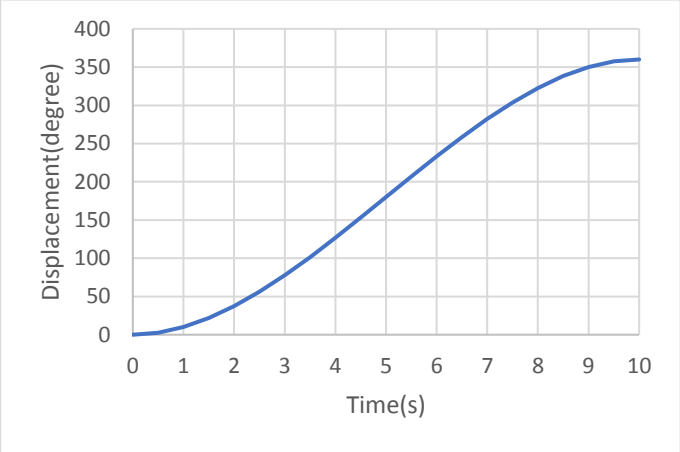


Figure 3.7 : Displacement in the trajectory for Joint 1

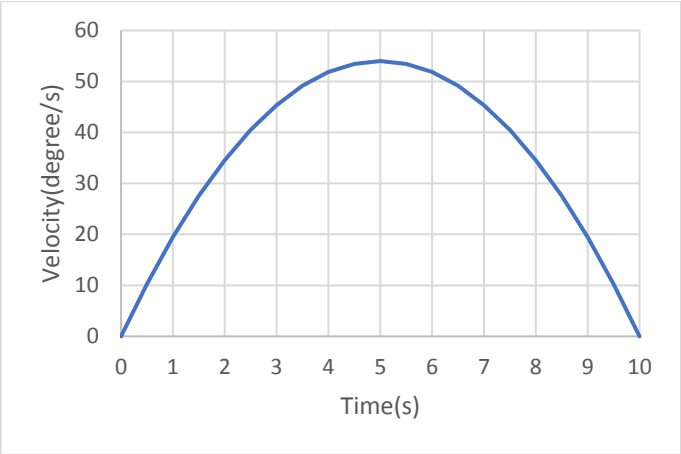


Figure 3.8 : Velocity in the trajectory for Joint1.

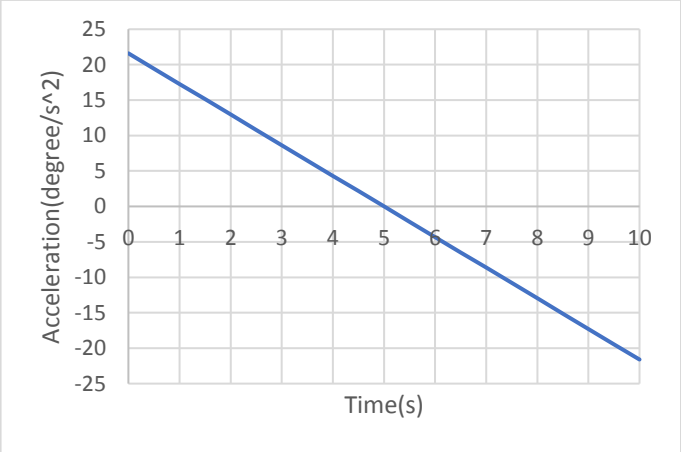


Figure 3.9 : Acceleration in the trajectory for Joint1.

Trajectory for joint 2 is given in the Figure 3.10, Figure 3.11 and Figure 3.12.

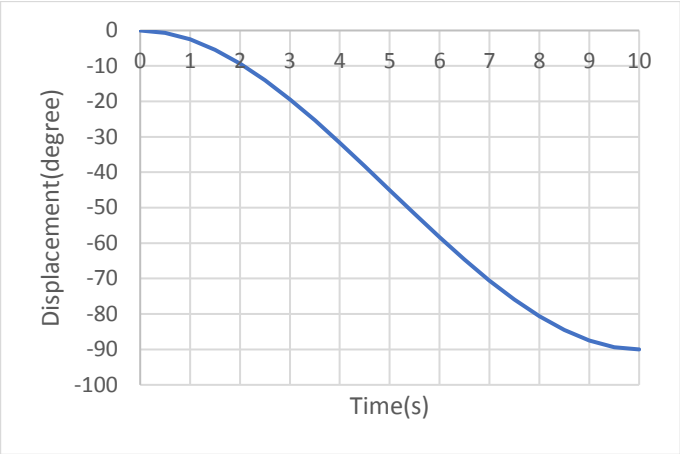


Figure 3.10 : Displacement in the trajectory for Joint2.

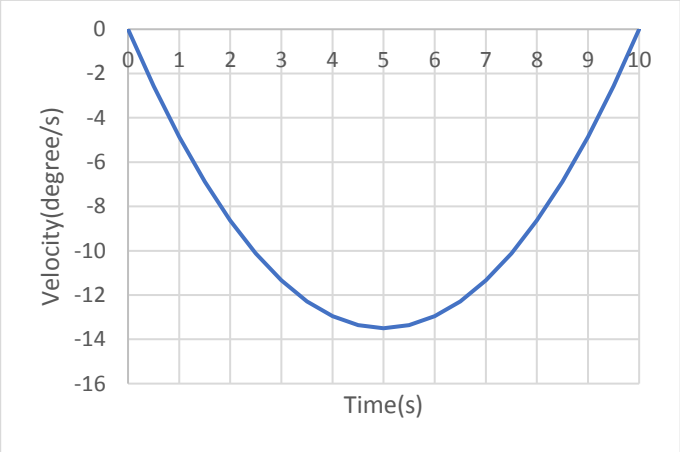


Figure 3.11 : Velocity in the trajectory for Joint2.

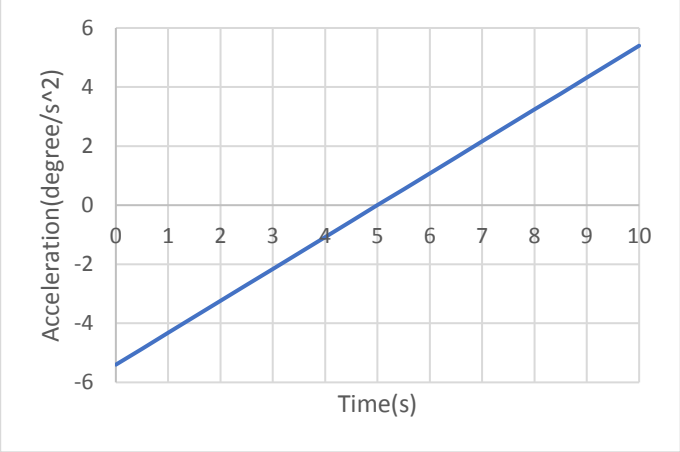


Figure 3.12 : Acceleration in the trajectory for Joint3.

Trajectory for joint 3 is given in Figure 3.13, Figure 3.14 and Figure 3.15.

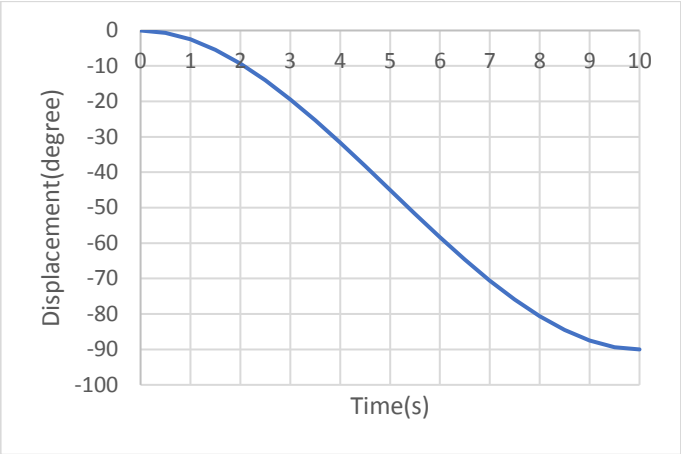


Figure 3.13 : Displacement in the trajectory for Joint3.

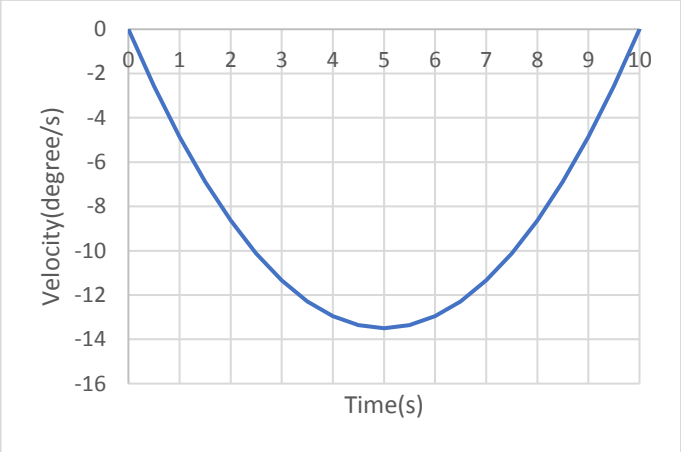


Figure 3.14 : Velocity in the trajectory for Joint3.

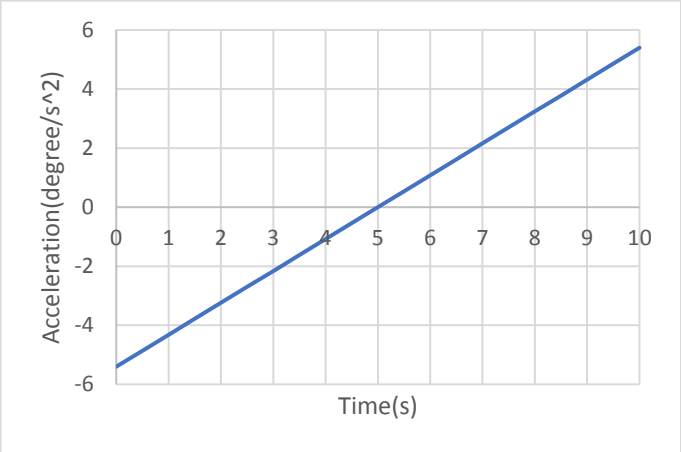


Figure 3.15 : Acceleration in the trajectory for Joint3.

The tip point of the modular robot manipulator follows the path displayed in the Figure 3.16 as a result of selected joint space trajectory.

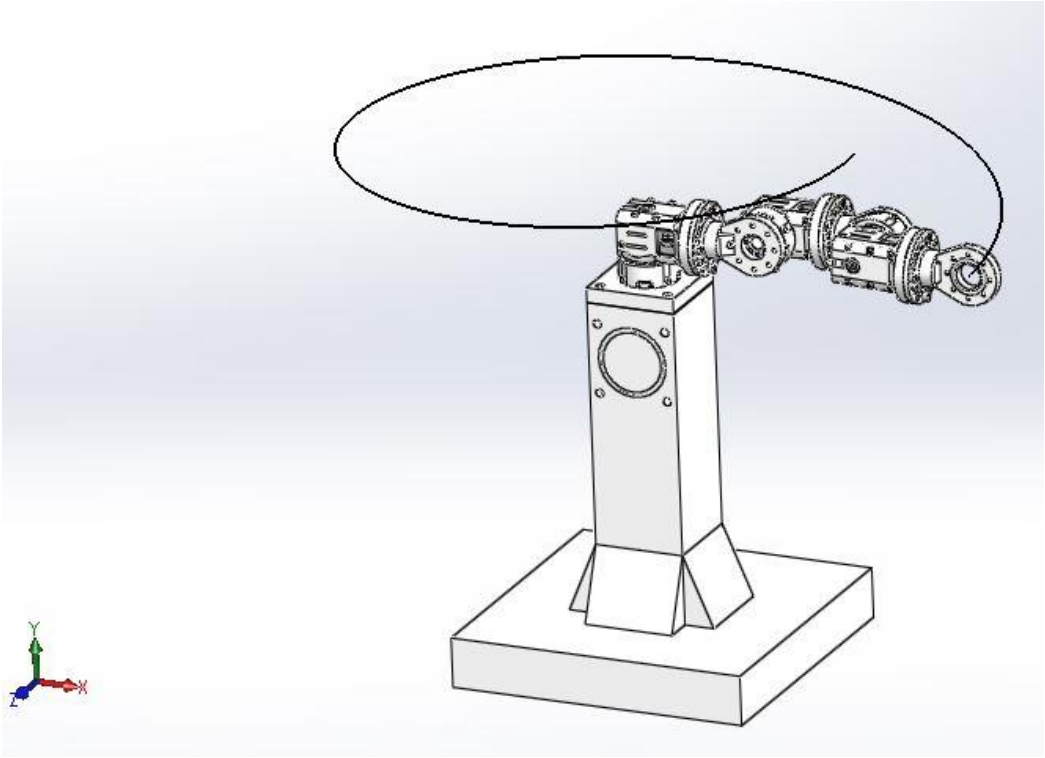


Figure 3.16 : Input path for dynamic analysis.

The Figure 3.17 shows the MATLAB Simulink diagram of the process of calculating torque values.

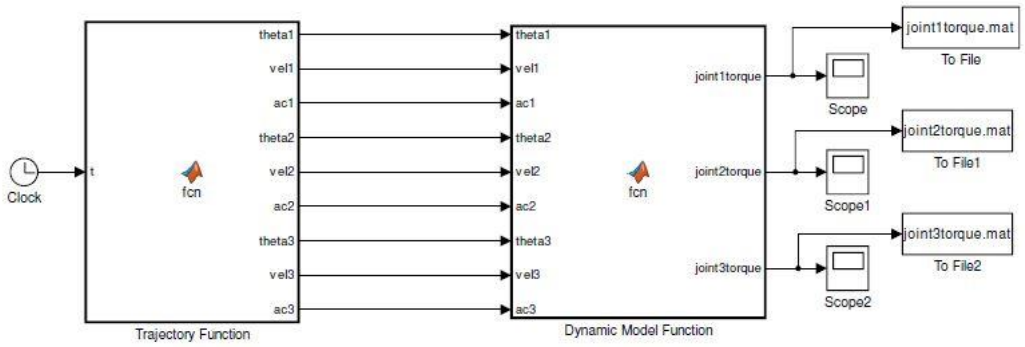


Figure 3.17 : Matlab simulink diagram for dynamic analysis.

The comparisons of the measured torque values which are obtained during the trajectory execution in the SolidWorks motion analysis and calculated from MATLAB are shown in the Figure 3.18, Figure 3.19 and Figure 3.20.

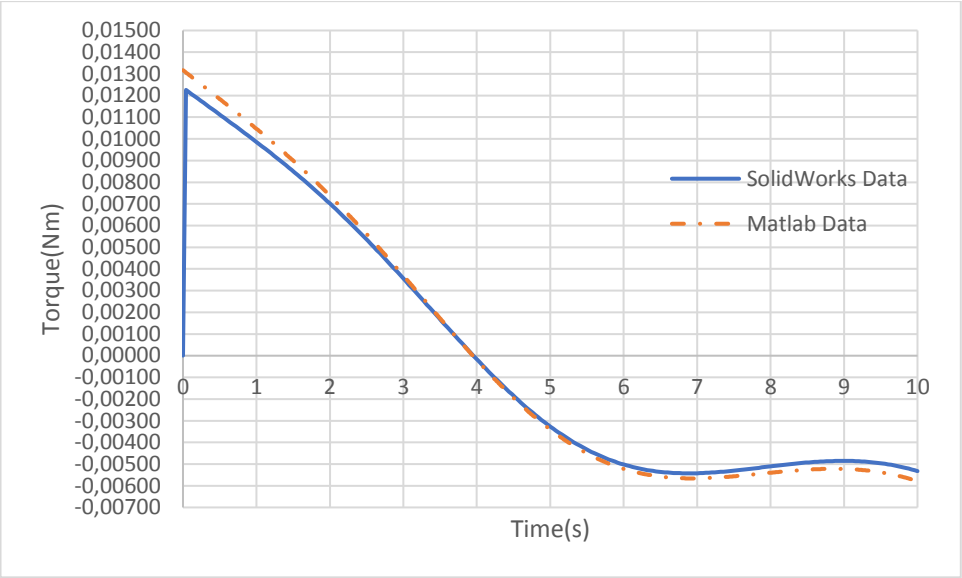


Figure 3.18 : Torque Results for Joint 1.

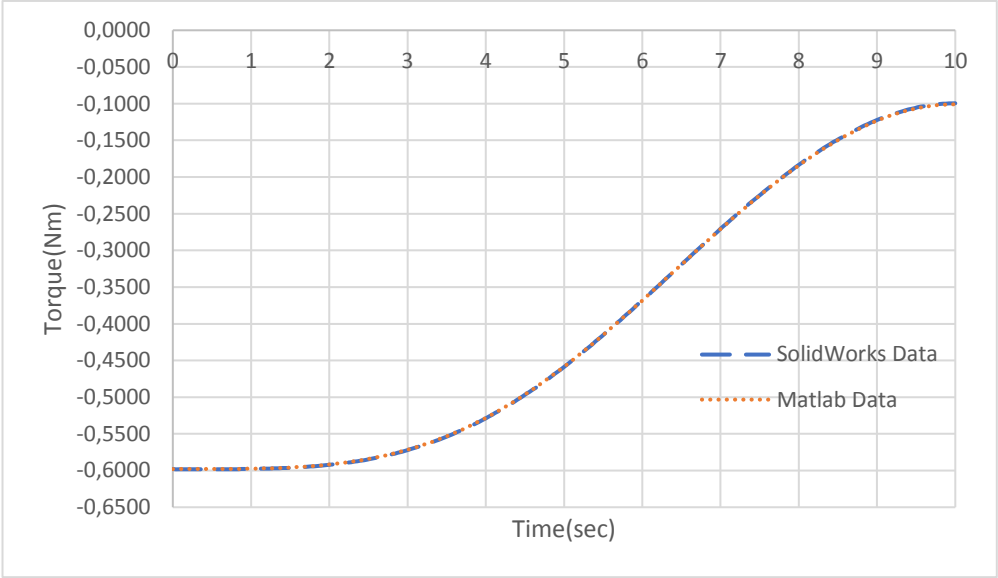


Figure 3.19 : Torque Results for Joint 2.

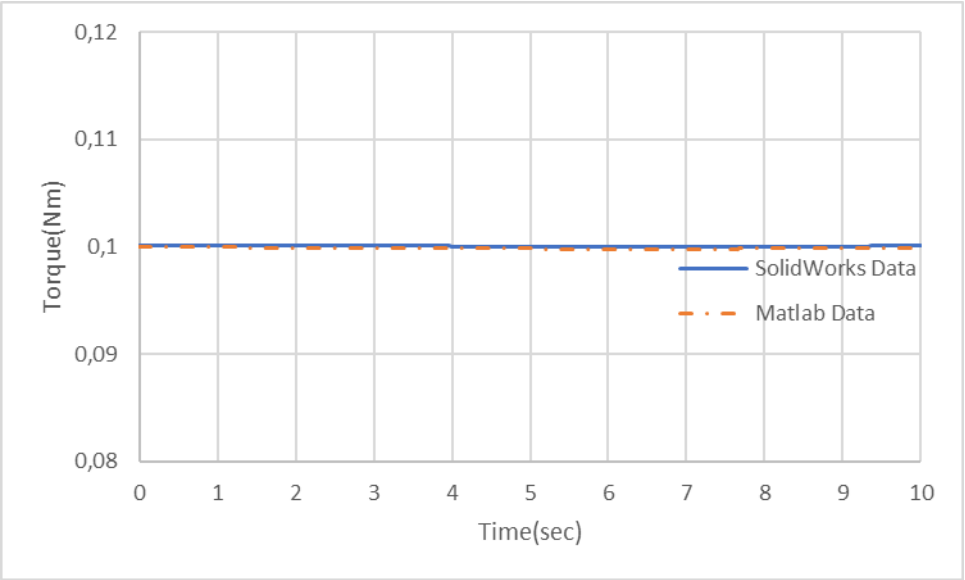


Figure 3.20 : Torque Results for Joint 3.

3.5 Payload Analysis of the Modular Robot Manipulator

In order to determine maximum payload of the modular robot manipulator, static force diagram and bending moment diagram were analyzed by applying payload to the gripper. In the payload analysis of the modular robot manipulator 3 DOF, 4 DOF and 5 DOF modular robot manipulator configurations were evaluated with the extended arm positions.

Because maximum torque of the DYNAMIXEL MX64 smart servo motor is 6 Nm, payload which exerts joint torques higher than 6 Nm was considered as maximum payload of the modular robot manipulator. It is needed the remark that this maximum payload may be differ for dynamic analyses.

3.5.1 Payload analysis of 3 DOF modular robot manipulator

In order to evaluate maximum torque values, robot configuration which is given in Figure 3.21 was considered.

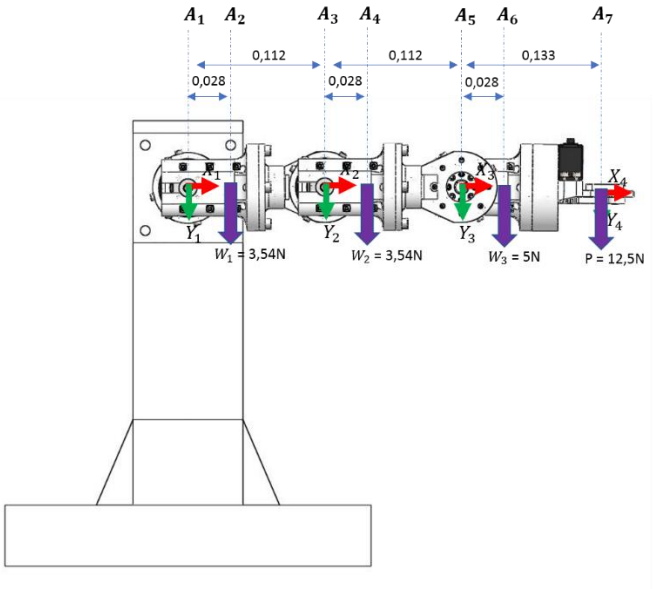


Figure 3.21 : 3 DOF modular robot manipulator payload analysis.

3.5.1.1 Shear force and bending moment diagram

Shear force and bending moment diagram of the modular robot manipulator were obtained as Figure 3.22 by applying 1.25 kg payload to the gripper. According to the conducted analysis, maximum payload was determined as 1.25 kg for 3 DOF modular robot manipulator.

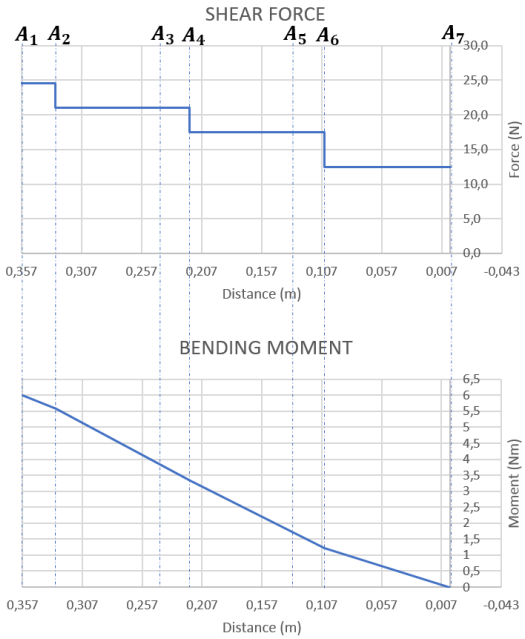


Figure 3.22 : Shear force and bending moment diagram for 3 DOF modular robot manipulator.

3.5.2 Payload analysis of 4 DOF modular robot manipulator

In order to evaluate maximum torque values, robot configuration which is given in Figure 3.23 was considered.

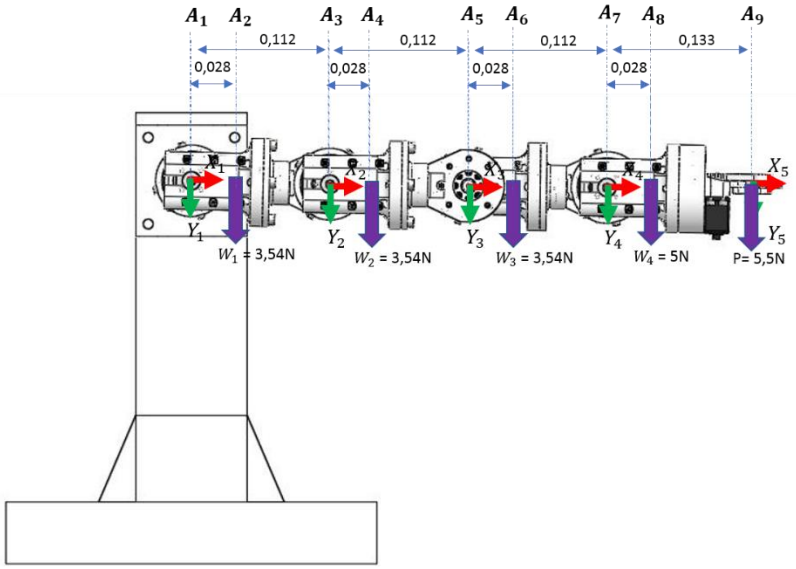


Figure 3.23 : 4 DOF modular robot manipulator payload analysis.

3.5.2.1 Shear force and bending moment diagram

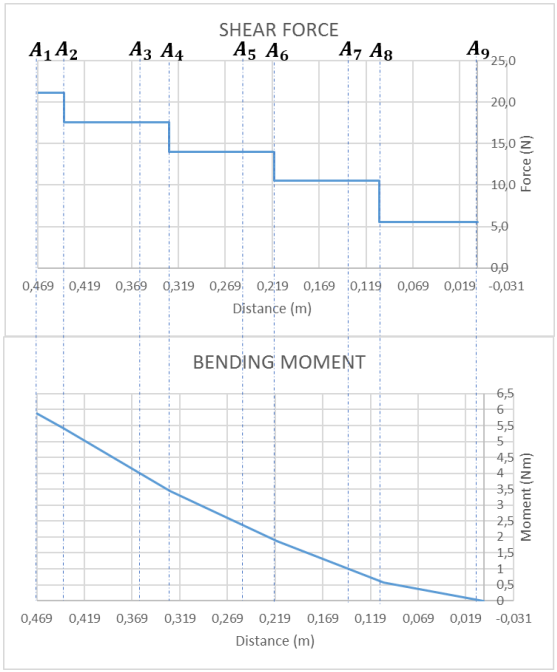


Figure 3.24 : Shear force and bending moment diagram for 4 DOF modular robot manipulator.

Shear force and bending moment diagram of the modular robot manipulator were obtained as Figure 3.24 by applying 0.55 kg payload to the gripper. According to the

conducted analysis, maximum payload was determined as 0.55 kg for 4 DOF modular robot manipulator.

3.5.3 Payload analysis of 5 DOF modular robot manipulator

In order to evaluate maximum torque values, robot configuration which is given in Figure 3.25 was considered.

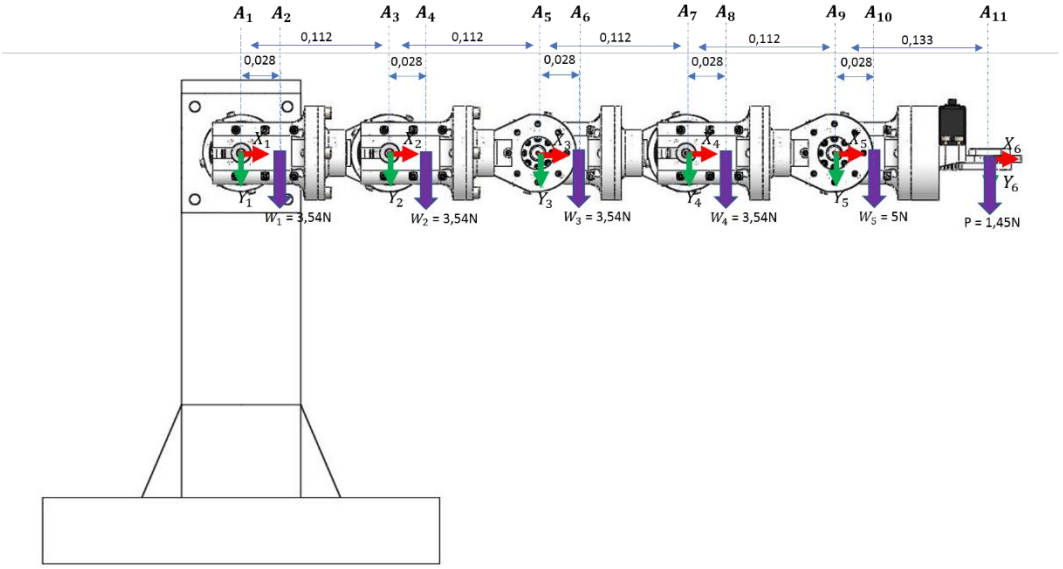


Figure 3.25 : 5 DOF modular robot manipulator payload analysis.

3.5.3.1 Shear force and bending moment diagram

Shear force and bending moment diagram of the modular robot manipulator were obtained as Figure 3.26 by applying 0.145 kg payload to the gripper. According to the conducted analysis, maximum payload was determined as 0.145 kg for 5 DOF modular robot manipulator.

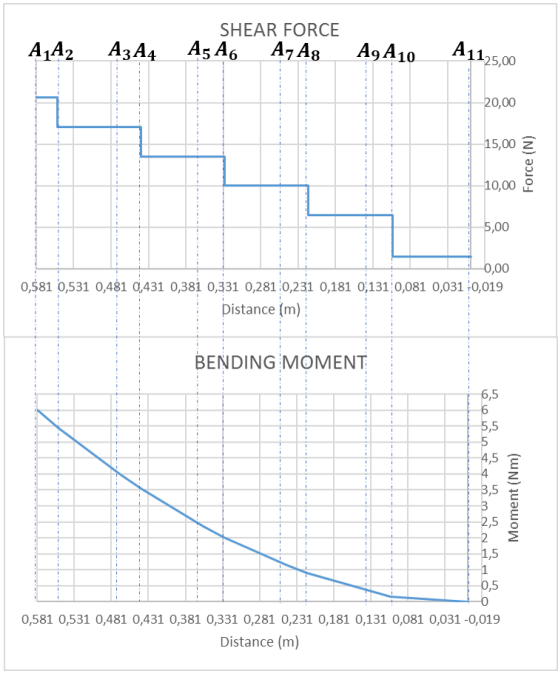


Figure 3.26 : Shear force and bending moment diagram for 5 DOF modular robot manipulator.

4. ROS BASED ANALYSIS OF MODULAR ROBOT MANIPULATOR

4.1 Introduction to ROS

The Robot Operating System is an open source framework for creating robot applications. On contrary to its name, ROS is not an operating system itself. It consists of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It provides the services including hardware abstraction, low-level device control, implementation of commonly-used functionality, message passing between processes, and package management.

ROS has distributed and modular structure. The modularity of ROS allows users to pick and use ready-made packages which are available in the ROS ecosystem in their particular projects. This is particularly useful during system development since several research groups can easily connect their respective computers to a working system. Also distributed nature of ROS fosters a large community of user-contributed packages that add a lot of value on top of the core ROS system.

In this project ROS is dedicated as software framework because of following reasons:

- Its natural modular structure best fits to the modular robot manipulator concept. Besides mechanical modularity of the robot, software modularity of the robot is accomplished by using ROS. Each module of the robot manipulator has the same controller structure but the different controller parameters. Therefore, in order to add a new module to the robot, only parameters changing is enough for controlling it and this builds software modularity to the robot.
- ROS tools enable to simulate the designed system in the development process before it is manufactured. The designed system can be controlled in the physic engine which is compatible with ROS. It permits developers to create their controllers and to test them on the simulated model. In this project, besides SolidWorks simulation tools, Gazebo physic engine is used to evaluate the dynamic performance of the robot. The designed controller structures are first

tested in the Gazebo and according to the results obtained they can be improved before the modules are manufactured.

- There are versatile ROS packages available and they can be easily used for custom robots. In this project, when evaluating kinematics and dynamics analyses of the modular robot manipulator, MoveIt! package is used. MoveIt! package can solve the inverse kinematics of the custom robot and generate collision free trajectories. Both in the simulation analyses and control of the robot manipulator, trajectories generated from MoveIt! package is used.
- ROS visualization tools enable to visualize the execution process of the robot. By using these tools, information like transformation relationships between robot links, generated trajectories in the motion planning can be monitored from the computer in the real time.
- In real-time applications it is crucial to log the output data in order to validate the performance of the system. The ***rosbag*** tool makes it easier to record those data from the system runtime. By using the ***rosbag*** tool, the system log files (states of the joints, generated trajectories, controller specific parameters etc.) are obtained and used for the performance analysis.

4.2 ROS Computation Graph

Computation Graph is the peer-to-peer network of ROS processes that are processing data together. The basic Computation Graph concepts of ROS are nodes, Master, Parameter Server, messages, services, topics, and bags, all of which provide data to the graph in different ways. [30]

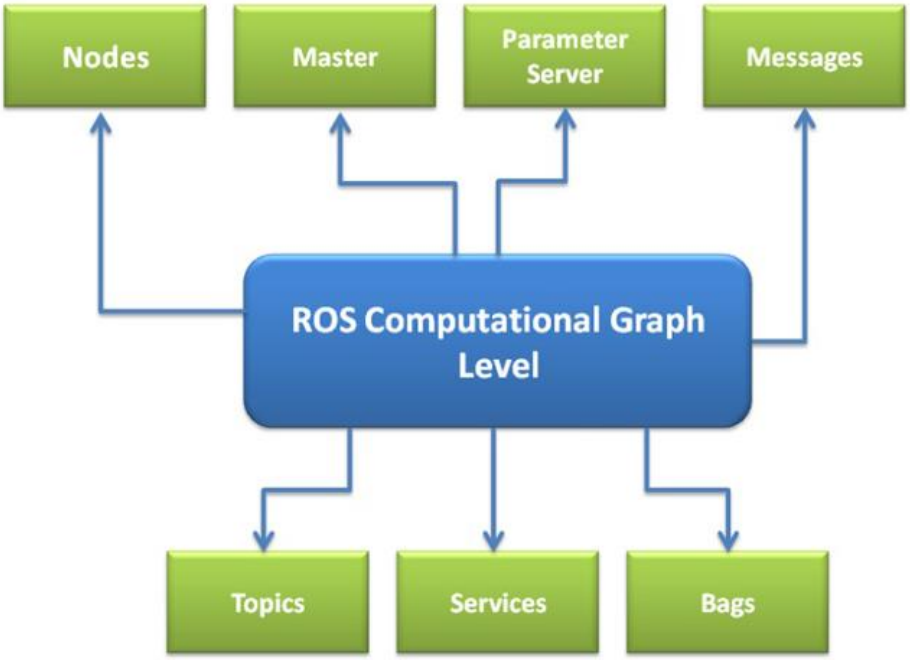


Figure 4.1 : ROS Computation Graph.

- Nodes: Nodes are the executable files that fulfills desired tasks. Each node has a unique id in the ROS network and they are communicated each other by using topics and services. In this project hardware interface is a node and it subscribes a reference commands and directed them to the servo motors.
- Master: The master provides name registration for nodes so that they can find each other in the ROS network. It permits message transferring between nodes and invoking services.
- Parameter Server: The Parameter Server is a shared place for nodes to store data. In the Parameter Server, data are stored by key variables. In this project, motor specific parameters and gain parameters for the controllers are stored in the Parameter Server.
- Messages: Data packets sent on the network are defined in ROS messages. A message is a data structure containing variable types. Standard primitive types (integer, floating point, boolean, etc.) are supported by ROS and by using these standard types different data structures can be obtained.
- Topics: In the ROS network messages are routed via transport system with publish/subscribe semantics. The topic is a name that is used to identify the

content of the message. A node can connect to a topic by its name either as a publisher in order to send data or as a subscriber in order to receive these data.

- Services: The Services work with request/reply interaction. It constitutes one-way transport in the ROS network. A client uses the services by sending the request message and awaiting the reply.
- Bags: Bags are special file formats for storing ROS messages. It is useful for collecting large data sets from nodes. It also allows to replay the collected data sets by republishing them with timestamps in the ROS network.

4.3 Used ROS Packages

4.3.1 Unified robot description format (URDF):

The Unified Robot Description Format (URDF) is an XML specification to describe a robot. [31] In order to describe a robot in ROS, a way is to use the package named URDF. This package represents the physical geometry, the kinematic and dynamic properties, the collision model, material, color and texture of the robot. The important limitation of the URDF format is that only tree structures can be represented therefore parallel robots cannot be introduced in URDF file.

The representation of a robot in the URDF includes a set of link elements, and a set of joint elements which connects the links together. Some XML specifications of the URDF are as follows:

- **link element:** The link element represents a rigid body with an inertia, visual and collision features.
- **joint element:** The joint element represents the kinematics and dynamics properties of the joint and also specifies the joint enabled limits (maximum torques, velocities etc.).
- **transmission element:** The transmission element is used to specify relationship between actuator and joint. This allows to model gear ratios which are especially useful in order to use URDF model of the robot in the simulation environment.

- **gazebo element:** The gazebo element is an extension to the URDF robot description format and it is used for simulation purpose in the Gazebo simulator.

4.3.2 MoveIt!

MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation [32]. It provides an easy-to-use platform for developing advanced robotics applications, evaluating new robot designs and building integrated robotics products for industrial, commercial, research and developments and other domains. [32] In this project, MoveIt! package is used in order to evaluate and verify the kinematic solver used in modular robot manipulator and to control the robot by taking the generated trajectories as inputs.

MoveIt! uses a plugin-based architecture for solving inverse kinematics and a native implementation for solving forward kinematics. While the default kinematics plugin currently used by MoveIt! is KDL kinematics plugins, users can add their custom solvers. The KDL kinematics plugins includes the numerical inverse kinematics solver provided by the OROCOS KDL package.

MoveIt! comes with a plugin for the ROS Visualizer (Rviz). The plugin allows to generate plans and visualize the output and interact directly with a visualized robot. It allows evaluating kinematics solver on the simulated robot manipulator. By using interactive marker on the Rviz, it can be tested if inverse kinematics solver finds correct solution or not.

4.3.3 OROCOS kinematics and dynamics library (KDL)

The computations required for the kinematics and dynamic models of a robot can be realized by using OROCOS Kinematics and Dynamics Library (KDL).

KDL is an application independent framework for modelling and computation of kinematics chains, such as robots, biomechanical human models, computer-animated figures, machine tools, etc. [33] It provides class libraries for geometrical objects, kinematic chains of various families, and their motion specification and interpolation.

KDL constructs kinematic chains and it enables to reach their kinematic and dynamic properties by its included functions, such as inverse and forward kinematics and dynamics of the chains.

In this project, KDL is used to compute the dynamic model of the robot in the computed torque controller. After constructing the robot model with URDF format, this model is transformed to the KDL chain. Then the inverse dynamics of the chain is obtained and used in the computed torque controller

4.3.4 Gazebo

Gazebo is an open source physic engine for simulating the designed robots in the realistic world conditions. It includes ROS packages for testing controllers created in the ROS.

In this thesis, Gazebo is used to evaluate controllers for modular robot manipulator in the simulation environment. In order to simulate the controllers in ROS, elements for the ***gazebo_ros_control*** plugin were added to the URDF model. Created URDF model in Gazebo is shown in Figure 4.2.

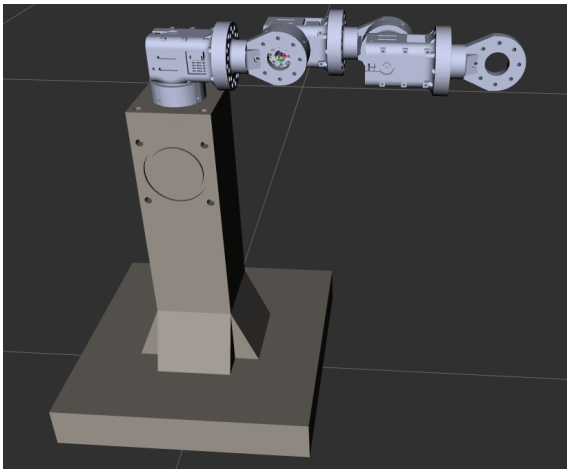


Figure 4.2 : Modular robot manipulator in Gazebo.

4.3.5 Rviz

Rviz is a 3D visualization tool for ROS [34]. It enables to display the obtained sensor data and state information on the simulated model of the robot.

4.4 Kinematic Analysis Using ROS

4.4.1 URDF model of the modular robot manipulator

In the URDF model of the modular robot manipulator, following parts were treated as a single link:

- Base Module
- Active module housing part
- Active module coupling part
- Gripper module

Then joints between these links were created according to the desired robot configurations. Link frames for base module and gripper module were selected as they are shown in Figure 3.5 and Figure 3.6 respectively. Link frame for active module was selected by splitting active module into two separate links. In order to ensure twist angle configuration, a fixed joint was located between them as it is seen in Figure 4.3. This fixed joint enables to configure the URDF model with a single parameter in the assembly process.

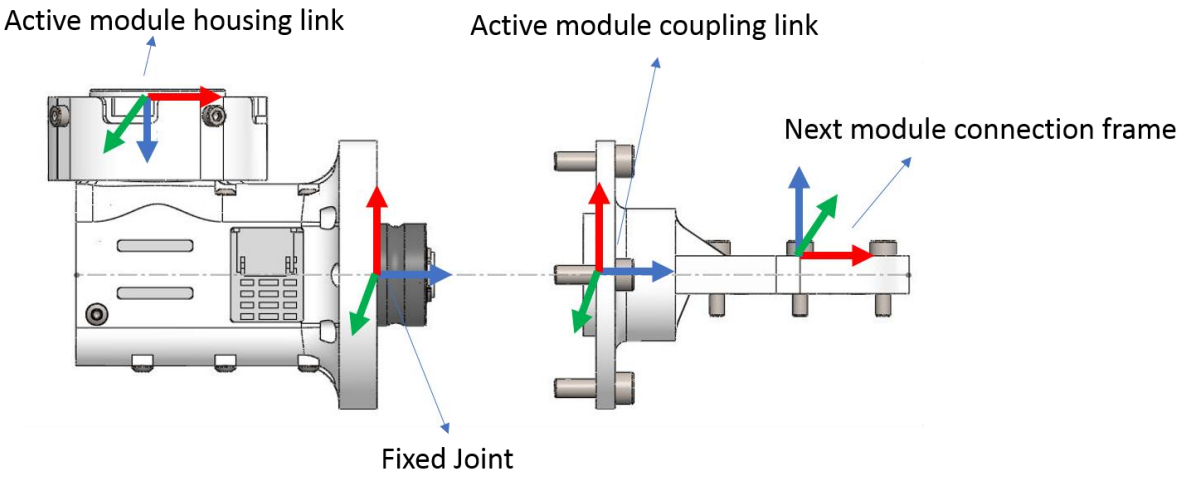


Figure 4.3 : Active module frames in URDF model.

Inertia matrices for base module and gripper module were taken from SolidWorks as it is described in section 3.4.1. Inertia matrices for active module housing part and active module coupling part were taken according to the assigned coordinate frames in Figure 4.3. Correct inertia matrices are vital for simulation in the Gazebo physics

engine. Selected inertia matrices and mass properties of the links then were validated in section 7.1.4.

Collision properties of the URDF model is utilized in the collision detection by MoveIt! package. Therefore, detailed collision model has important role in motion planning. Visual and collision properties of the links were obtained from SolidWorks as a STL file format. For base module and gripper module, STL files were obtained according to the output coordinate frames that are used in the section 3.4.1**.** Besides, for active module two separate STL files were exported as it is shown in Figure 4.3.

Resulted URDF model of the robot is given in the Figure 4.4 and graph visualization of the URDF model is given in the appendix A.

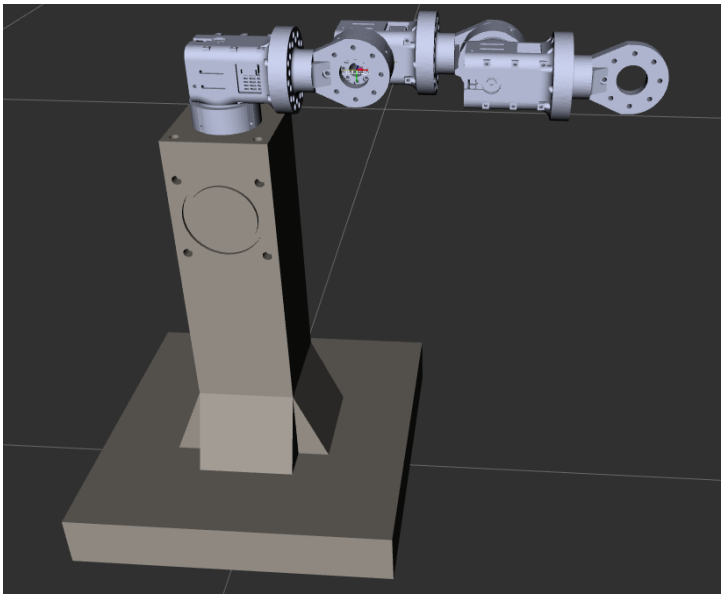


Figure 4.4 : URDF model of the modular robot manipulator.

4.4.2 MoveIt! setup of the modular robot manipulator

MoveIt! configuration for the modular robot manipulator was created with MoveIt! Setup Assistant. URDF model was imported to the setup assistant. Self-collision matrix was generated in order to determine pairs of links on the robot that are disabled in the collision checking. MoveIt! Package determine these pairs of links when they are always in collision, never in collision, in collision in the robot's default position or when the links are adjacent to each other on the kinematic chain. Self-collision matrix decreases the motion planning processing time.

A planning group was created and kinematic chain from the base link to the end effector was added to the group. KDL kinematics solver was chosen as a kinematic

solver as it is discussed in section 3.3.2 and RRT* path planner was selected as it is discussed in section 5.4.1.

In order to control the robot directly from MoveIt! with the planned motion, action controller name was inserted to the configuration file.

Modular robot in MoveIt! Package is shown in Figure 4.5 and generated configuration file is given in appendix B.

4.4.3 Obtaining simulation results on ROS

Inverse Kinematic solution tests were conducted with MoveIt! package. By transporting interactive marker which is located on the end effector frame to the example poses in the workspace of the robot, availability of the inverse kinematic solution was verified in Rviz. This process is showed in Figure 4.5.

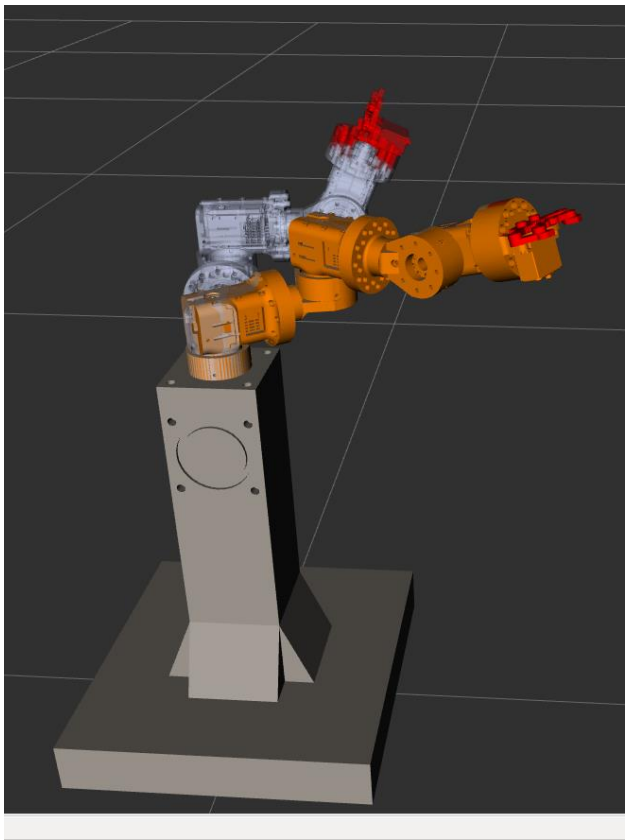


Figure 4.5 : Motion planning with MoveIt!

4.5 Graphical User Interface Plugin for Rviz

Graphical user interface was designed in order to plan motions for modular robot manipulator. Qt4 library was used for GUI development. GUI is given in Figure 4.6.

Joint space jog control of the modular robot manipulator is provided from GUI with the push buttons. During the jog control, self-collision detection is checked by using URDF model of the robot and invalid commands are avoided.

GUI allows user to teach points directly from MoveIt! with interactive markers in Rviz or from jogging robot to the desired pose. For the taught points a database created and it is allowed to access these points both in programming with GUI and programming with custom codes.

In order to allow user to plan point to point and cartesian motions, a program text box is located in the GUI plugin. A command structure created for input to the program text box. Command structure includes "Motion Point1 Point2 VEL VALUE;" sequence where "Motion" command can be "PTP" for point to point motion and "LIN" for cartesian motion, "Point1" and "Point2" can be any point stored in the database and VEL command represents velocity percentage that is used to scale output trajectory.

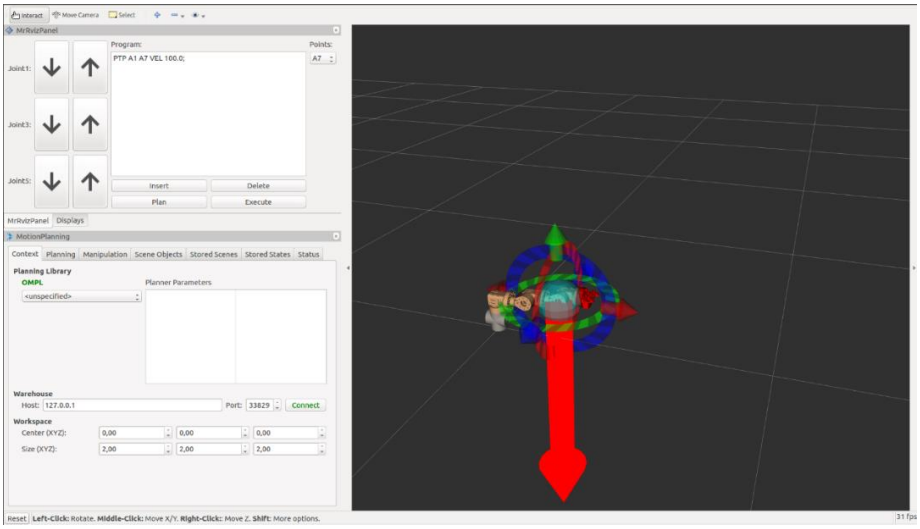


Figure 4.6 : Modular robot GUI plugin in Rviz

5. CONTROL OF MODULAR ROBOT MANIPULATOR

The control problem for robot manipulators is determining joint inputs required for manipulator to follow desired trajectory signal. The joint inputs may be joint forces, torques or voltage inputs to the actuators.

In order to realize a better control structure, it is required to obtain an accurate dynamic model of the robot manipulator. There are many control techniques and methodologies for designing control structure of the manipulator; independent joint control, computed torque control, adaptive control etc.

5.1 Basic Control Strategies

5.1.1 Independent joint control:

In this type of control, each module of the robot manipulator is controlled as a single-input/single-output (SISO) system. The basic structure of this control system is shown in the Figure 5.1.

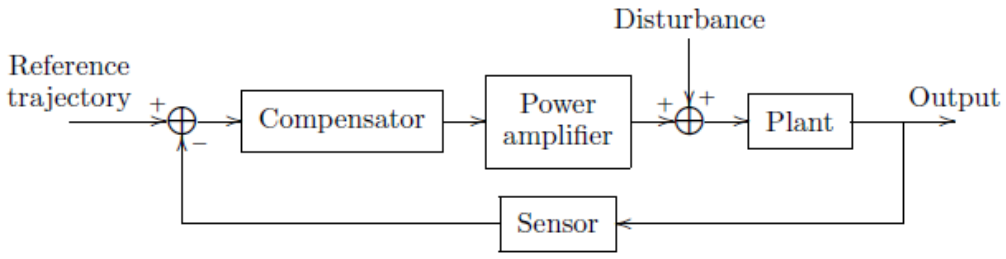


Figure 5.1 : Basic structure of a feedback control system.

The purpose of control design is to make the robot respond in a predictable and desirable fashion to a set of reference input signals. Therefore, the design objective is to choose the compensator which makes the plant output follow the given reference signal. There are also disturbances acting on the system which influence the behavior of the output. The controller must be designed in a such way that reducing the effects of the disturbances.

5.2 Controller Design for Modular Robot Manipulator

5.2.1 Actuator dynamics:

Inside each active module of the modular robot manipulator, a DYNAMIXEL MX64 DC smart servo motor was used as an actuator. Therefore, dynamic model of an armature-controlled DC motor is introduced in this section. In Figure 5.2, an example DC motor model is given.

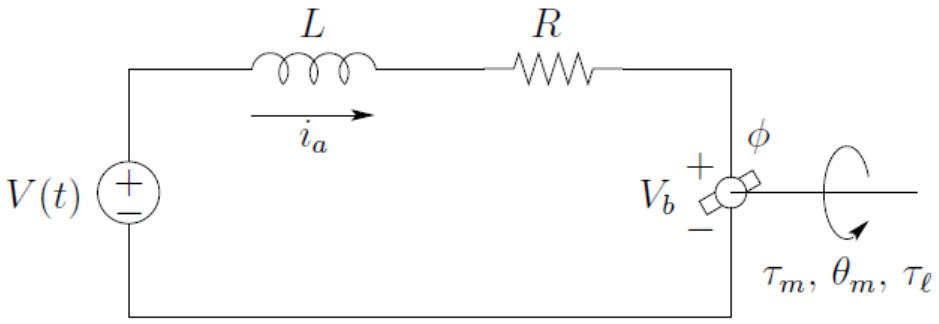


Figure 5.2 : DC motor model.

The magnitude of the motor torque is proportional to the armature current:

$$\tau_m = K_m i_a \qquad (5.1)$$

The voltage generated across the terminal of the motor as it is moving:

$$V_b = K_b \dot{\theta}_m \qquad (5.2)$$

In the Laplace domain differential equations related by electrical and mechanical parts of the motor then can be written as:

$$(Ls + R)I_a(s) = V(s) - K_b s\theta_m(s) \qquad (5.3)$$

$$(J_m s^2 + B_m s)\theta_m(s) = K_m I_a(s) - \tau_l(s)/r \qquad (5.4)$$

The block diagram of this system is shown in the Figure 5.3:

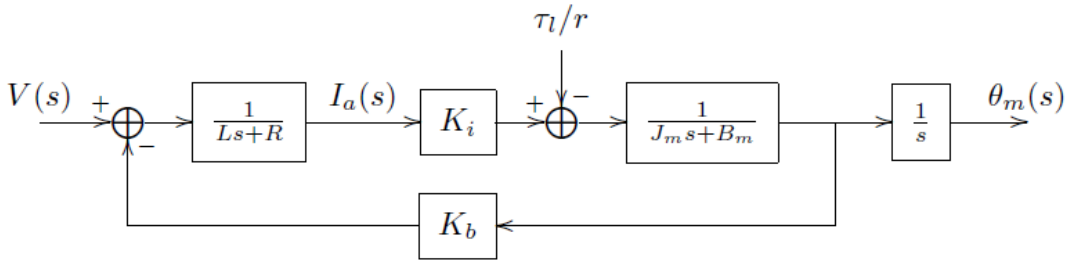


Figure 5.3 : DC motor block diagram.

When it is assumed that the electrical time constant $\frac{L}{R}$ is much smaller than the mechanical time constant $\frac{J_m}{B_m}$, electrical time constant can be neglected and the final dynamic model of the DC motor is obtained as following:

$$J_m\ddot{\theta}_m(t) + \left(B_m + \frac{K_b K_m}{R}\right)\dot{\theta}_m(t) = \left(\frac{K_m}{R}\right)V(t) - \frac{\tau_l(t)}{r} \qquad (5.5)$$

The block diagram of the reduced order system is shown in Figure 5.4.

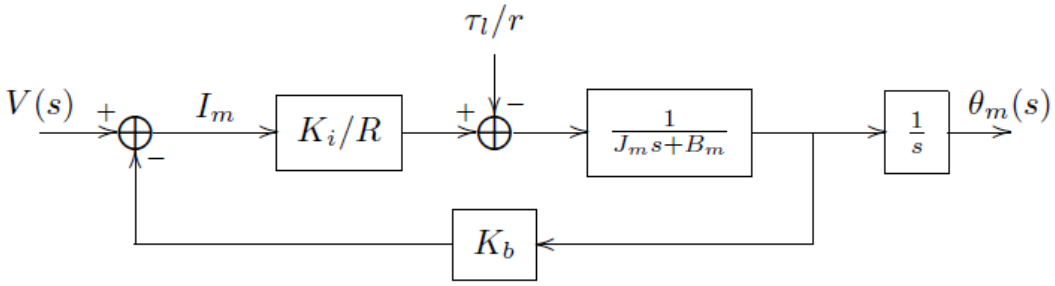


Figure 5.4 : Reduced block diagram of a DC motor.

In the thesis [35], DYNAMIXEL MX64 smart servo motor plant parameters were identified and these plant parameters were utilized in this thesis.

5.2.2 Independent joint dynamics:

In case of gear ratios of the actuators r_i are very large, it has important effect to simplify the design of robot manipulator controllers. Large values of r_i reduces the effect of the nonlinear function in the robot manipulator dynamics. Therefore, the robot manipulator dynamics can be obtained by n decoupled second-order differential equations with constant coefficients.

$$\left(J_{m_i} + \frac{d_{ii}(q)}{r_i^2}\right)\ddot{q}_{m_i} + (B_{mi} + \frac{K_{b_i}K_{m_i}}{R_i})\dot{q}_{m_i} = \frac{K_{m_i}}{R_i}V_i - d_i \qquad (5.6)$$

where d_i is a disturbance given by:

$$d_i = \frac{1}{r_i}\sum_{j\neq i} d_{ij}\ddot{q}_j + \sum_{j,k} c_{jki}\dot{q}_j\dot{q}_k + g_i \qquad (5.7)$$

And effective inertia and damping of the system is denoted by equations 5.8 and 5.9 respectively.

$$J_{eff} = J_{m_i} + \frac{d_{ii}(q)}{r_i^2} \qquad (5.8)$$

$$B_{eff} = B_{mi} + \frac{K_{b_i}K_{m_i}}{R_i} \qquad (5.9)$$

And u denotes the input to the system:

$$u_i = \frac{K_{m_i}}{R_i}V_i \qquad (5.10)$$

The dynamic equation which nonlinear effect of the nonlinear coupling terms is treated as disturbance d_i, then can be obtained as following:

$$J_{eff}\ddot{q}_{m_i} + B_{eff}\dot{q}_{m_i} = u_i - d_i \qquad (5.11)$$

5.2.3 PID compensator:

In order to perform independent joint control a PID compensator can be created as it is shown in Figure 5.5 and the system transfer function is obtained as equation 5.12.

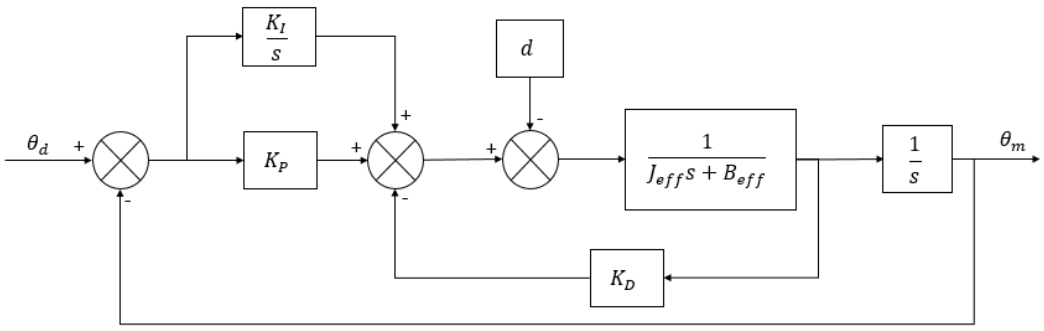


Figure 5.5 : Independent joint control block diagram.

$$\theta_m(s) = \frac{K_D s^2 + K_P s + K_I}{J_{eff}s^3 + (B_{eff} + KK_D)s^2 + KK_P s + KK_I}\theta_d(s) - \frac{rs}{J_{eff}s^3 + (B_{eff} + KK_D)s^2 + KK_P s + KK_I}D(s) \quad (5.12)$$

5.2.4 PID based joint trajectory controller:

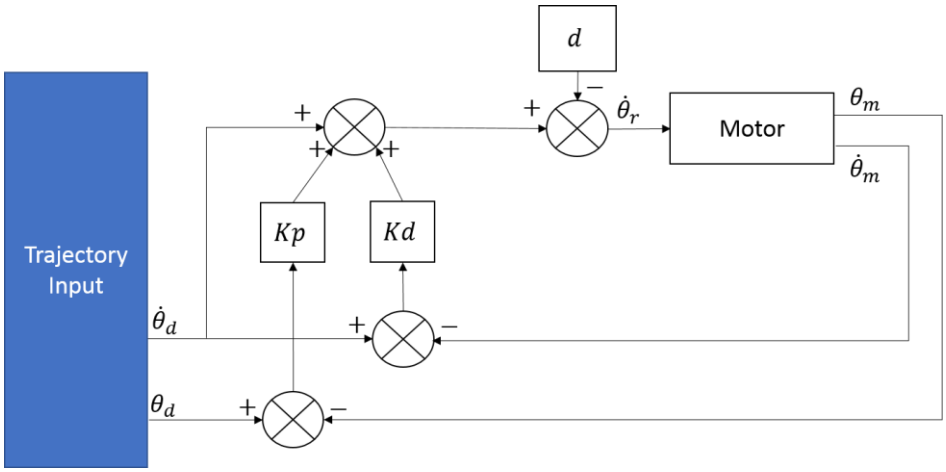


Figure 5.6 : Independent joint trajectory controller block diagram.

Single input single output based independent joint trajectory controller which is given in Figure 5.6 was used for controlling modular robot manipulator on simulation. On the simulation model, joints which actuate with velocity reference was added to the model. For tracking given trajectory input, PD control law is implemented to compute commanded velocities. The trajectory tracking result for the controller is given in the section 7.1.5.1.

Because of DYNAMIXEL motors have internal PID controllers which take position, velocity and torque reference inputs, joint trajectory controller will have two control loops as it is shown in Figure 5.7. In the inner control loop block, DYNAMIXEL

velocity controller [36] was given. On the experimental setup this block diagram was used, and tuning process of the outer and inner loops and trajectory tracking performance of the controller are given in section 7.1.6.

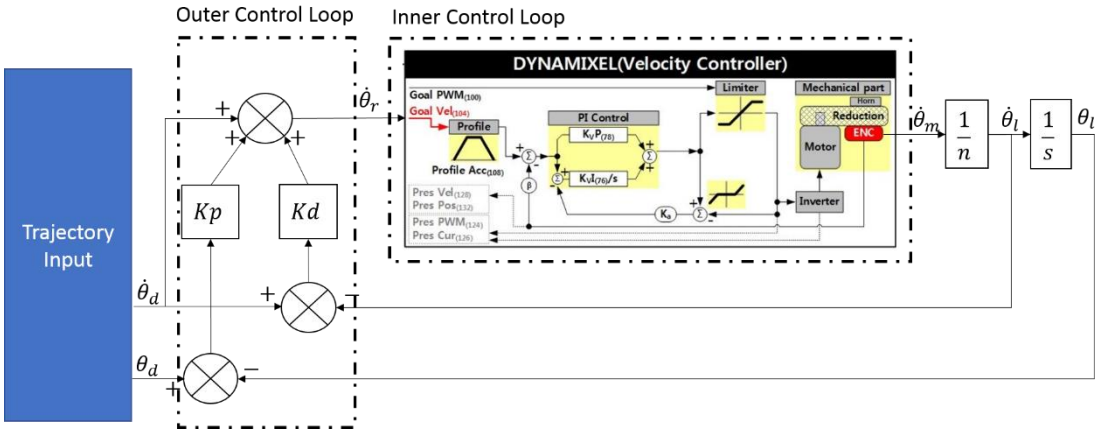


Figure 5.7 : Independent joint trajectory controller with inner PID control loop.

Implementation of the controller in ROS and its test results are discussed in the following sections.

5.2.5 Computed torque controller:

Computed torque control method can be applied effectively when the loads that may come from system dynamics can be measured. Because dynamic analysis of the modular robot manipulator was obtained and validated, computed torque controller is created in ROS by using dynamic model of the modular robot manipulator and test results are validated.

The dynamics model of the robot manipulator obtained in the section 3.4.1 is as follows:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + \tau_d = \tau \qquad (5.13)$$

where q denotes the joint variable, τ denotes the input voltage and τ_d denotes a disturbance.

When a desired trajectory $q_d(t)$ is given to the manipulator to ensure trajectory tracking by the joint variable, the tracking error can be defined as:

$$e(t) = q_d(t) - q(t) \qquad (5.14)$$

$$\dot{e} = \dot{q}_d - \dot{q} \quad (5.15)$$

$$\ddot{e} = \ddot{q}_d - \ddot{q} \quad (5.16)$$

If $\ddot{q}$ is solved from equation 5.13 when τ_d is ignored and substituting into the 5.16 yields:

$$\ddot{e} = \ddot{q}_d + M^{-1}(C(q,\dot{q}) + G(q) - \tau) \quad (5.17)$$

Input function can be defined as linearized equation:

$$u = \ddot{q}_d + M^{-1}(C(q,\dot{q}) + G(q) - \tau) \quad (5.18)$$

We may define a state x(t) as:

$$x = \begin{pmatrix} e \\ \dot{e} \end{pmatrix} \quad (5.19)$$

Then tracking error dynamics will be a linear error system as 5.20:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{I} \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{I} \end{bmatrix} u \quad (5.20)$$

Equation 5.18 can be expressed as following equation:

$$\tau = M(\ddot{q}_d - u) + C(q,\dot{q}) + G(q) \quad (5.21)$$

If u(t) is selected so that e(t) goes to zero from equation 5.20, then nonlinear control input given by $\tau(t)$ in equation 5.21 will cause trajectory following in the manipulator. The control input u(t) can be selected as the PD feedback:

$$u = -K_p e - K_d \dot{e} \quad (5.22)$$

Then the overall dynamic equation becomes:

$$\tau = M\left(\ddot{q}_d + K_p e + K_d \dot{e}\right) + C(q,\dot{q}) + G(q) \quad (5.23)$$

The block diagram of the computed torque control is shown in Figure 5.8.

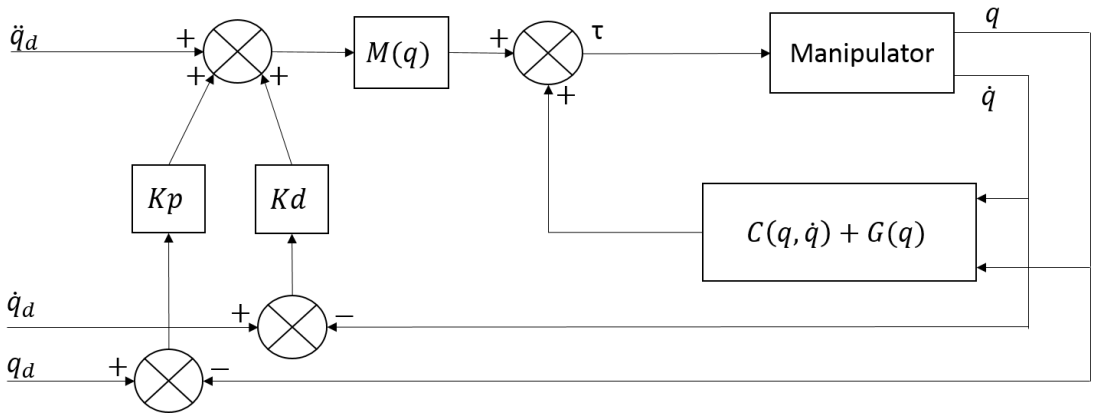


Figure 5.8 : Computed torque control block diagram.

The closed loop error dynamics are:

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \qquad (5.24)$$

Equation 5.24 shows that by choosing the matrices K_p and K_d in a diagonal form, a decouple closed loop system can be obtained where the behavior of each joint error is given by a second order differential equation.

The closed loop characteristic polynomial is:

$$s^2 I + K_d s + K_p = 0 \qquad (5.25)$$

The desired damping ratio ζ and natural frequency w_n for joint error is calculated with these equations:

$$K_d = diag(k_{d_i}) \qquad (5.26)$$

$$K_p = diag(k_{p_i}) \qquad (5.27)$$

$$k_{p_i} = w_n^2 \qquad (5.28)$$

$$k_{d_i} = 2\zeta w_n \qquad (5.29)$$

Which ***diag()*** function represents diagonal matrix and for the appropriate k_{p_i} and k_{d_i} which are selected as positive numbers system will be asymptotically stable. In order to avoid for overshoots in the robot motion, the PD gain is selected for critical damping ζ=1.

Computed torque controller was implemented in ROS and trajectory tracking results was tested on Gazebo. In the following sections implementation and test results are given.

5.3 Implementation of ROS to Modular Robot Control

In order to create a consistent interface for modular robot in ROS environment, ***ros_control*** [35] framework was used when creating controllers. The ***ros_control*** framework provides the capabilities to implement and manage robot controllers with a real-time performance and enables sharing of controllers in a robot-agnostic way [35]. Thanks to its modular architecture, different controllers can be switched in real-time. Also, thanks to its plugins, designed controllers can be simulated on Gazebo before implementing them on the real robot.

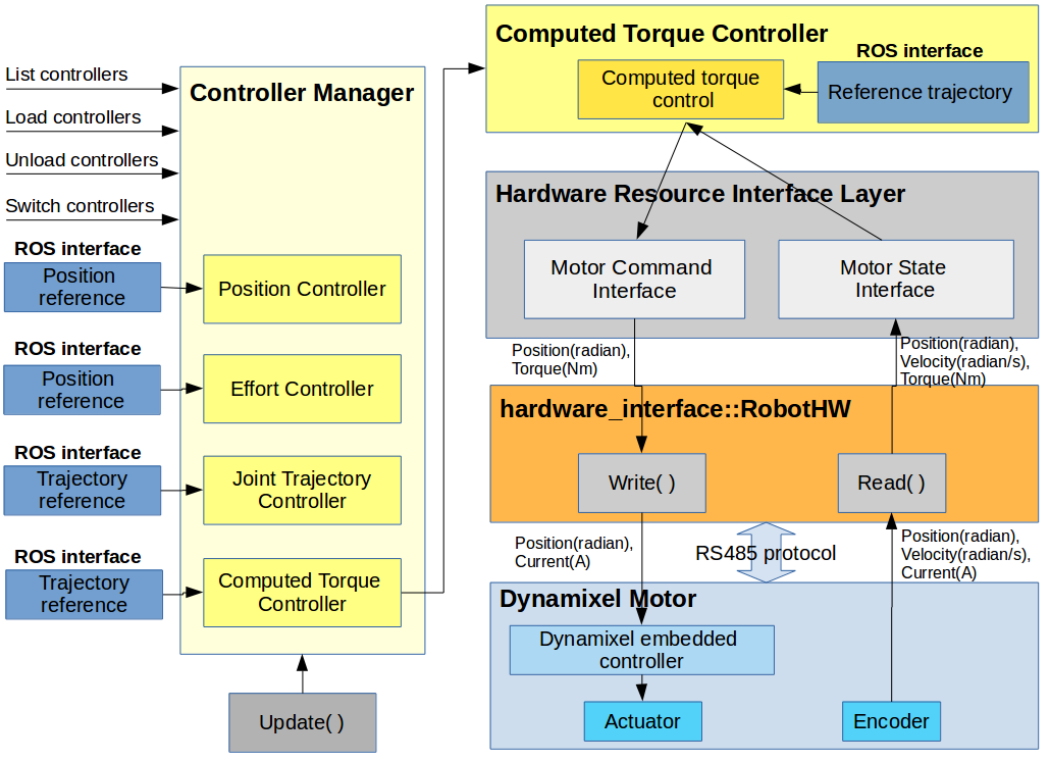


Figure 5.9 : Active module ROS controller diagram.

In the Figure 5.9, ROS controller diagram which is used to control each active modules of the modular robot manipulator is presented.

There are two important functions in ***Hardware Resource Interface Layer***; "write" and "read" which are responsible for interacting with hardware. While creating these functions DynamixelSDK [36] library was used owing to it includes communication

functions with appropriate communication protocols from ROBOTIS company. By using ***Read ( )*** function, PC that runs ROS, acquires joint states which are positions, velocities and torques of the motors and by using ***Write ( )*** function it sends an appropriate control variable which can be positions or torques to the motors.

Dynamixel Motor block represents DYNAMIXEL motor's inner structure. It includes an embedded controller inside which implements PID control to drive the motor and sensors which can measure the position, velocity and current of the motor. It also, allows to change its PID coefficients for each motor control mode with serial communication.

Controller Manager is in charge of managing controller plug-ins at runtime. It deploys ROS services that can list, load, unload and switch controllers. By means of its well-designed architecture, at the moment while switching controllers it prevents joints not to remain uncontrolled to prevent falling the links. It detects resource conflicts between controllers and handles those situations. Real-time loop inside the controller manager follows the read, update and write sequences. In read process, controller manager reads joint states from ***Hardware Resource Interface Layer***. In update process of the ***Controller Manager***, control method is implemented from currently loaded controller for given joint states. In write process, calculated joint command in update process is commanded to the hardware.

5.3.1 Position controller

Position Controller does not include a control law. It takes position reference command in joint space and directs them to the actuators. Control law is expected to be implemented in actuators themselves.

5.3.2 Effort controllers

Effort controllers are single input single output controllers. In effort controllers control variable is torque and they expect effort controllable joint which is ***EffortJointInterface*** type of hardware interface which can be driven by torque input. It includes effort-based position, velocity and effort controllers. In position controller, PID control is applied and calculated torque command is directed to the actuator, according to the position reference command. In velocity controller, according to the velocity reference command, PID control is applied by calculating velocity error and

torque command is directed to the actuator. On the other hand, in effort controller, according to the torque reference command, PID control is applied by calculating torque error and torque command is directed to the actuator.

5.3.3 Joint trajectory controller

Joint trajectory controller takes trajectory input in the joint space as a set of waypoints and makes interpolation between the trajectory waypoints. In interpolation process it utilizes from the waypoint's timestamps and according to the waypoints specifications it uses linear, cubic or quantic interpolation methods.

According to the used hardware interface type, it calculates the position and velocity trajectory following error and apply PID control.

Joint trajectory controller currently includes single input single output-based position controllers, velocity controllers and effort controllers. The designed computed torque controller is implemented for joint trajectory controller.

5.3.4 Computed torque controller

Computed torque control method which is presented in section 5.2.5 is implemented as a ROS controller plugin. KDL library was used for computing inverse dynamics of the modular robot manipulator. KDL library offers recursive Newton-Euler solver for the inverse dynamics problem.

Computed torque controller consists of several methods inherited from ***RobotHW*** class. In the ***init ( )*** method, KDL [33] library dependent KDL chain object is created from the URDF model of the robot manipulator. KDL library uses this chain, in order to determine kinematic model of the robot. Also, PID controller object is initialized in here for outer loop control in computed torque control method.

In the ***start ( )*** method, control variables are reset. Because of the fact that it will be a plugin that will be loaded by controller manager, in ***start ( )*** method all control variables are assigned to their initial values, to prevent unpredictable situations when controller switching is demanded by controller manager.

Main algorithm of the computed torque controller is conducted in ***update ( )*** method. This method is called by ***Controller Manager*** in real time control loop. According to the given positions and velocities of the joints, the trajectory following error is

computed. By applying PID control in the outer loop of the controller, acceleration input is computed for the trajectory following error. Determined acceleration is inputted to the recursive Newton-Euler solver and joint torques are computed. Then calculated torque commands are assigned to the joint handles variables and they are directed to the actuators by controller manager.

5.3.5 Joint state controller

In contrast to its name, it is not a controller but is a controller plugin in ***ros_control***. It takes joint states from joint handle resources in the update process of the controller manager and publishes them in ROS environment. The other ROS components, i.e. MoveIt [32] for motion planning, are aware of the joint states by means of joint state controller.

5.3.6 Gripper controller

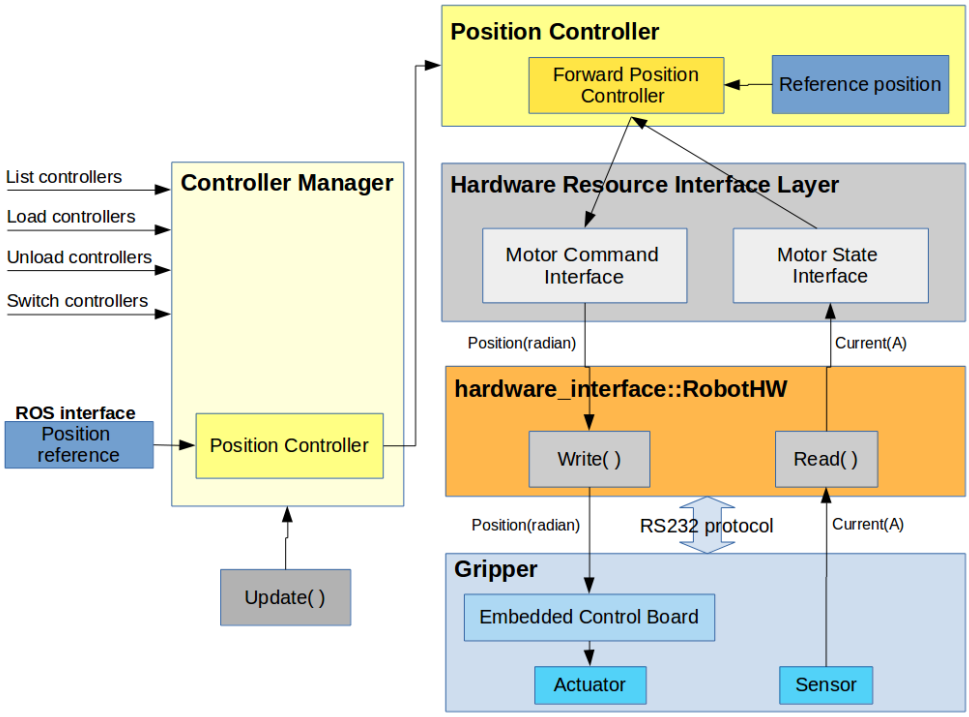


Figure 5.10 : Gripper module ROS controller diagram.

Gripper has an analog servo motor which includes position controller inside it. According to the angle reference command, servo motor applies its inner control law to reach the reference angle. As position feedback of the servo motor is not available, it is just recorded the last reference angle sent to the motor and published in ROS

environment. Gripper motion is controlled according to the current feedback from the motor drawn. Gripper controller is given in Figure 5.10.

In the low-level controller, Arduino UNO microcontroller is used. Communication between ROS and Arduino DUE is established with RS232 protocol. Two types of packets are created for data exchanging between ROS controller and Arduino controller. In the write packet PC sends desired position command to the microcontroller and in the read packet PC receives last commanded position of the gripper and current of the motor.

Microcontroller receives position reference command from ROS controller and sends its inner control variable which holds last commanded reference angle to the servo motor, to the ROS controller. Because of at startup, the last commanded reference angle is empty, embedded control has an initialized procedure which drives servo to the predefined angle just to ensure that servo is in state of last commanded angle. It also includes home procedure which drives the servo motor to a predefined position.

In the ROS environment, gripper is controlled with position controller and the given reference commands are directed to the motors without using any control law. In case that current reaches the defined threshold, gripper movement is stopped.

An action server is created for gripper controller to handle opening and closing actions of the gripper. Action server takes a goal which can be open and close gripper and sends appropriate joint angle references to the position controller. If the last commanded joint angle and load value of the controller are not obtained from microcontroller within timeout tolerances, action is canceled and reported to the action client.

In closing action, gripper moves to the desired angle and if the returned current feedback is higher than the threshold, success is reported to the client otherwise failed is reported to the client and action is canceled.

In opening action, gripper moves to the set angle for opening situation.

5.4 Motion Planning

MoveIt! package is used for motion planning framework. MoveIt! is an open source software used for robot motion planning. It uses Open Motion Planning Library

(OMPL), Stochastic Trajectory Optimization for Motion Planning (STOMP), Search-Based Planning Library (SBPL) and Covariant Hamiltonian Optimization for Motion Planning (CHOMP) libraries for motion planning implementations. [37]

MoveIt! has a plugin-based architecture for solving inverse kinematics and a native implementation for solving forward kinematics. While the default kinematics plugin currently used by MoveIt! is KDL kinematics plugins, users can add their custom solvers. In this thesis, Orocos KDL kinematics plugin described in inverse kinematics chapter is used in MoveIt!

For the used path planners in MoveIt! package, there is a benchmarking [38]. According to the benchmarking results and experimental tests which are conducted in this thesis, RRT* solver is determined in order to create collision free paths.

5.4.1 RRT* algorithm

RRT* algorithm operates in the configuration space which has free cells that robot can reach and occupied cells by obstacles that cannot be reached by robot. [39]

RRT* algorithm begins when selecting initial and goal state to find collision free path. Processed path is expanded in each iteration towards to the goal. In each iteration, RRT* randomly generates points in the configuration space and evaluates them. If generated point lies outside of an obstacle and point is reached from nearest node, RRT* chains the point to the tree.

In this chaining process, algorithm searches nearest nodes of the random generated point in the tree and by applying cost to them selects best parent to the random point.

Algorithm iteratively searches for path until the generated node is within the goal region or the termination limit is reached. In Figure 5.11, expanding process of the RRT* algorithm is shown.

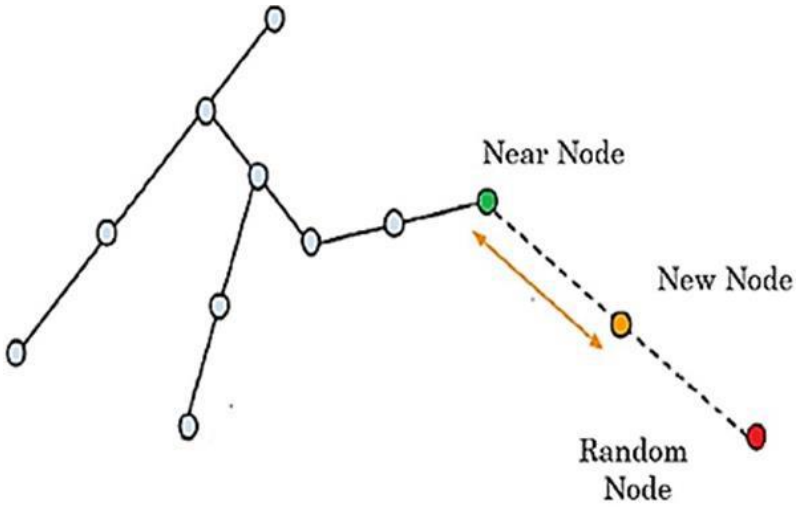


Figure 5.11 : RRT* algorithm visualization. [39]

5.4.2 Motion planning flow chart

Motion planning flow chart that is implemented in this thesis is given in Figure 5.12. In the Task Plan block, set of desired end effector poses are determined for robot manipulator. These end effector poses should be inside or outside of the reachable workspace of the robot. This information will send back to the user on the next sections.

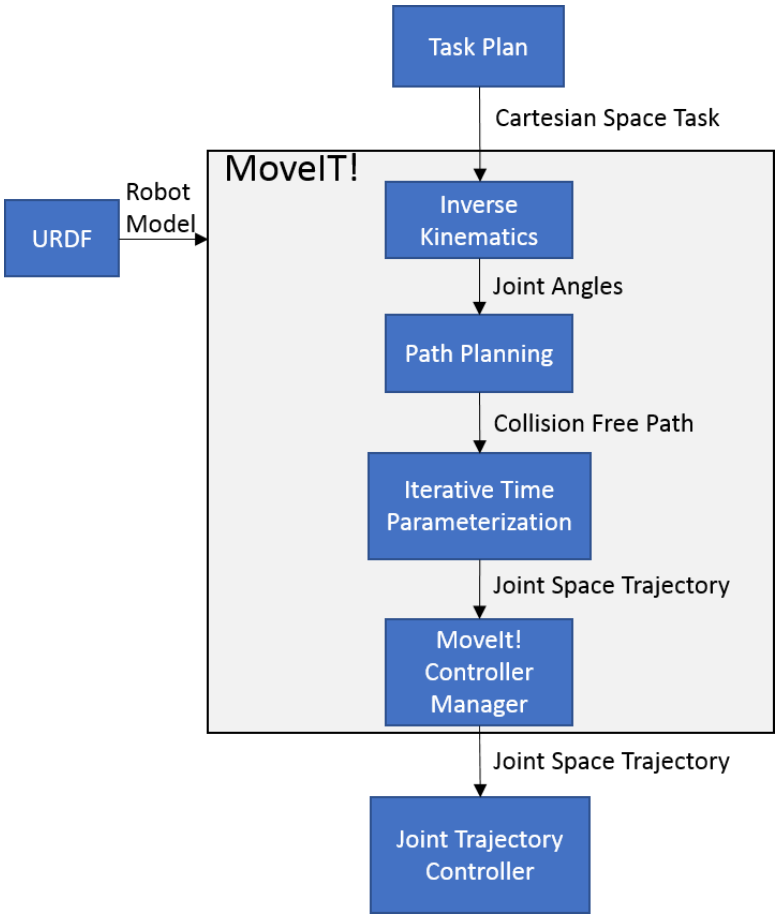


Figure 5.12 : Motion planning flow chart

In the inverse kinematics block, for the given cartesian space end effector poses, joint angles are calculated by KDL numerical solver discussed before. If the inverse kinematic solver could not find the solution, it is seed random state of the manipulator to the numerical solver and inverse kinematic solver is executed again. This process is repeated until the number of tries is reached or inverse kinematic solution is found.

In the path planning block, found joint angles from the inverse kinematic solver is taken as joint space goal and path planning with RRT* algorithm is conducted from current state of the robot to the goal state. During path planning, inner collision detection is implemented by using collision description in the URDF model of the robot. If the planning scene of the environment is given, RRT* algorithm makes collision checking with the objects located in the environment. In this process kinematic constraints can also be specified to the RRT* algorithm.

In the Iterative Time Parameterization block, joint space trajectory is generated from found path. According to the maximum joint velocities and accelerations, spline interpolation is made among the points along the path. Maximum velocities and

accelerations information of the robot manipulator are taken from the URDF model. The calculated joint trajectory is then can be post processed for ensuring the desired joint velocities. By using joint space jump detection, it tries to avoid robot singularities.

MoveIt! has controller manager inside and it expects action client server for controlling joints. Created joint trajectory controller is introduced to the MoveIt! and then MoveIt! is connected to the action server as a client. MoveIt! takes joint states from the ROS topic and it updates inner kinematic model of the robot according to the current joint states. After it succeeded to plan a trajectory, it executes joint space trajectory by sending it to the joint trajectory controller action server. Trajectory following information are obtained as a feedback from action server and according to the goal tolerances and if tolerances are exceeded, the goal is aborted and information is sent to the user.

6. EXPERIMENTAL RIG DESIGN

In this section, experimental rig design of the modular robot manipulator is discussed.

6.1 Manufacturing of Modules

In the manufacturing of active module and gripper module uPrint 3D printer was used with ABS filament. Besides, MDF material was used to manufacture base module.

In order to create a 3 DOF modular robot manipulator three active modules, one gripper module and base module were manufactured. Manufactured modules are given in Figure 6.1, Figure 6.2 and Figure 6.3.

Figure 6.1 : Manufactured active module

Figure 6.2 : Manufactured gripper module which is mounted to the housing part of active module

Figure 6.3 : Manufactured base module

6.2 Mechanic Accessories

In order to actuate the modular robot manipulator following components are used:

- DYNAMIXEL MX-64T Servo Motor

In the active modules DYNAMIXEL MX-64T servo motors used as actuator. In Figure 6.4 DYNAMIXEL MX-64T servo motor is shown and its hardware specifications are given in the Table 6.1.

Figure 6.4 : Dynamixel MX-64T servo motor.

Table 6.1 : Dynamixel MX-64T Hardware Specifications.

Resolution	0.088°
Stall Torque	6 Nm
Voltage	12 V
Communication Protocol	TTL
Weight	135 g
Running Degree	Endless Turn

- PowerHD 1201 MG Servo Motor

In the gripper module, PowerHD 1201 MG servo motor which is shown in Figure 6.5 was used as actuator.

Figure 6.5 : PowerHD 1201 MG servo motor

Its hardware specifications are listed in Table 6.2.

Table 6.2 : PowerHD 1201 MG Hardware Specifications.

Stall Torque	1,294 Nm
Voltage	6 V
Weight	60 g
Running Degree	0° ~ 180°

6.3 Electronic Accessories

In the modular robot manipulator, the following electronic components are used:

- Arduino UNO

Figure 6.6 : Arduino UNO microcontroller

Arduino UNO microcontroller is used for controlling gripper. The microcontroller is shown in Figure 6.6.

- DYNAMIXEL Expansion Board

Figure 6.7 : OpenCM 485 Expansion Board

In order to power DYNAMIXEL MX64 smart servo motors, OpenCM 485 DYNAMIXEL Expansion Board was used. In the board, there are connectors for TTL and RS485 types of DYNAMIXEL servo motors. It allows to connect 12 Volt SMPS unit to the board with the power jack. In Figure 6.7, OpenCM 485 expansion board is shown.

- TTL to USB converter

Figure 6.8 : USB2Dynamixel

DYNAMIXEL MX64 servo motors which used in this project have TTL communication protocol. TTL to USB converter from ROBOTIS is used for communication between DYNAMIXEL servo motors and PC. USB2Dynamixel converter is shown in Figure 6.8.

- Power Supply

12V 5A power supply is used for supplying power to the active modules and 6V 3.5A power supply was used for supplying power to the gripper module.

6.4 Experimental Setup

3 DOF and 2 DOF modular robot manipulator configurations were created with the manufactured modules as experimental setup. Experimental setups are shown in Figure 6.9 and Figure 6.10.

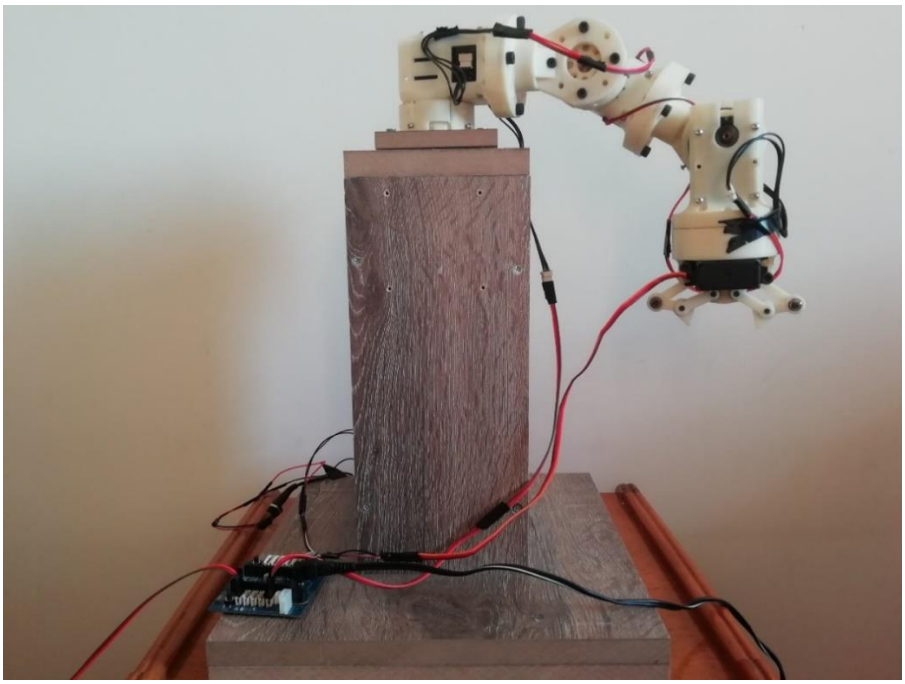


Figure 6.9 : 3 DOF Experimental setup.

Figure 6.10 : 2 DOF Experimental setup.

7. RESULTS AND DISCUSSIONS

7.1 Validation and Verification

7.1.1 Numerical inverse kinematic solver validation

In order to validate the kinematic solver performance which is discussed in section 3.3.2, 4 to 6 DOF robot configurations were evaluated. 1000 random sample points that are reachable for robot configurations were selected by calculating forward kinematics. Joint angles used in the forward kinematics and sample points were logged. Inverse kinematic solver was tested by giving sample points as inputs to the solver and calculated joint angles were logged if the inverse kinematic solution found.

Kinematic solver results were then evaluated with inverse kinematic solution success rate, average time passed during calculating solution, average end-effector position error, average end-effector roll, pitch and yaw angle errors.

Created test program is added to the appendix C and results are listed in the Table 7.1.

Table 7.1 : KDL Inverse kinematic solver statistics.

DOF	4	5	6
Success Rate (%)	0.99	1	0,88
Average Time (s)	0.03	81.21×10^{-3}	4.20
AEE	2.37×10^{-7}	1.23×10^{-6}	2.07×10^{-6}
ARE	8.69×10^{-7}	1.97×10^{-6}	5.54×10^{-7}
APE	8.69×10^{-7}	3.96×10^{-7}	5.59×10^{-8}
AYE	9.20×10^{-7}	2.06×10^{-6}	5.67×10^{-7}

In the Table 7.1, AEE denotes average end-effector position error and unit is meters, ARE, APE and AYE denotes end-effector roll, pitch and yaw angle errors respectively and their units are in radians.

Success Rate metric is calculated as following:

$$Success\ Rate = \frac{Number\ of\ Sucessive\ Results}{1000} \quad (7.1)$$

Average time is calculated with the following equation:

$$Avarage\ Time = \frac{\sum_{i=1}^{1000} t_i}{1000} \quad (7.2)$$

In which t_i denotes time passed to inverse kinematic solving process of the pose i.

Average end-effector position error (AEE) is calculated with the following equation:

$$AEE = \frac{\sum_{i=1}^{1000}(d_i - p_i)^2}{1000} \quad (7.3)$$

In equation, d_i and p_i represents desired position of the end-effector and calculated position from inverse kinematic solver respectively.

ARE, APE and AYE errors are calculated with equation X, Y, Z respectively as:

$$\mathrm{ARE} = \frac{\sum_{i=1}^{1000}(\mathrm{dr_i} - \mathrm{pr_i})^2}{1000} \quad (7.4)$$

$$APE = \frac{\sum_{i=1}^{1000}(dp_i - pp_i)^2}{1000} \quad (7.5)$$

$$AYE = \frac{\sum_{i=1}^{1000}(dy_i - py_i)^2}{1000} \quad (7.6)$$

In the equations above, dr_i, $dp_i$and dy_i represents desired roll, pitch and yaw angles of the end-effector respectively, and pr_i, pp_i and py_i denotes calculated roll, pitch and yaw angles respectively.

According to the obtained results, it is observed that success rate is sufficient but decreases when the DOF of the manipulator increases and average time required to solve inverse kinematics increases with DOF of the robot. It is also observed that results of the inverse kinematic solution highly depend to the solver seeding state.

7.1.2 Analytic inverse kinematic solution validation

Inverse kinematic solution for the 3 DOF modular robot manipulator in the Figure 3.1 was found in the section 3.1. In this section, analytic inverse kinematic solution is tested with a known end effector position.

For the given joint angles, the end effector position of the robot manipulator is determined from forward kinematics solution:

$$\theta_1 = 1.57, \theta_2 = 0.7, \theta_3 = 0.45 => X = 2 \times 10^{-4}, Y = 30.62 \times 10^{-3},$$

$$Z = 32.71 \times 10^{-1}$$

By taking the output end-effector position of the forward kinematics solution, inverse kinematic procedure was conducted and the following eight solution sets were obtained:

$$\text{IK 1: } \{\theta_1 = 1.57, \theta_2 = 0.6994, \theta_3 = 0.4486\}$$

$$\text{IK 2: } \{\theta_1 = 1.57, \theta_2 = -3.3924, \theta_3 = 0.4486\}$$

$$\text{IK 3: } \{\theta_1 = 1.57, \theta_2 = 0.2508, \theta_3 = -0.4486\}$$

$$\text{IK 4: } \{\theta_1 = 1.57, \theta_2 = -3.8410, \theta_3 = -0.4486\}$$

$$\text{IK 5: } \{\theta_1 = -1.57, \theta_2 = 0.4623, \theta_3 = 0\}$$

$$\text{IK 6: } \{\theta_1 = -1.57, \theta_2 = -3.6039, \theta_3 = 0\}$$

$$\text{IK 7: } \{\theta_1 = -1.57, \theta_2 = 0.4623, \theta_3 = 0\}$$

$$\text{IK 8: } \{\theta_1 = -1.57, \theta_2 = -3.6039, \theta_3 = 0\}$$

According to obtained solution sets, IK1 solution set ensured the given joint angles input.

7.1.3 Singularity analysis validation

In order to conduct singularity analysis for 3 DOF modular robot manipulator shown in the Figure 3.1, known singular pose was considered. Jacobian matrix of the 3 DOF modular robot manipulator in this singular pose was obtained as in equation 7.7:

$$J = \begin{bmatrix} -0.1121 & -0.001 & 0.001 \\ 0.0435 & -0.11 & 0.1120 \\ 0 & 0.0001 & 0.0001 \\ 0 & -1 & 1 \\ 0 & 0.0008 & 0.0008 \\ 1 & 0 & 0 \end{bmatrix} \quad (7.7)$$

In order to calculate determinant of the Jacobian matrix, SVD of Jacobian was obtained with the MATLAB function that is given in appendix D.

$$\mathrm{U} = \begin{bmatrix} -0.0004 & 0.1113 & -0.1089 & 0.0278 & 0.0739 & 0.9846 \\ 0.1115 & -0.0421 & -0.6585 & -0.587 & 0.4474 & -0.085 \\ 0.001 & 0 & -0.0843 & 0.6670 & 0.7355 & -0.083 \\ 0.9938 & 0.0096 & 0.0744 & 0.0653 & -0.049 & 0.0095 \\ -0.0008 & 0 & 0.7360 & -0.452 & 0.5008 & 0.0566 \\ 0.0048 & 0.9929 & 0.0164 & 0.0287 & -0.011 & 0.1140 \end{bmatrix} \quad (7.8)$$

$$S = \begin{bmatrix} 1.4231 & 0 & 0 \\ 0 & 1.0072 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (7.9)$$

$$V = \begin{bmatrix} 0.068 & -1 & 0 \\ -0.7071 & -0.0048 & 0.7071 \\ 0.7071 & 0.0048 & 0.7071 \end{bmatrix} \quad (7.10)$$

In the equations above, U represents left singular vector, V represents right singular vector, S represents singular values of the Jacobian matrix.

Determinant of the Jacobian matrix then was obtained by using the equation 3.12:

$$\det(\mathrm{J}) = -1.6867\mathrm{x}10^{-16} \quad (7.11)$$

Because determinant of the Jacobian is too close to the zero, this end-effector position is verified as singular point for the robot manipulator.

7.1.4 Gazebo simulation model validation

As it is stated, modular robot manipulator controllers first tested in the Gazebo physics engine. For a realistic simulation it is necessary creating a simulation model accurately. In order to validate simulation model of the robot, torque results of the joints when they are controlled with a PD controller are evaluated in this section.

In MATLAB, output dynamic model of the section 3.4 was used with PD controller. Proportional and derivative gains parameters of the controllers were taken the same for both MATLAB and Gazebo simulations and motor dynamics properties were ignored. An example joint space trajectory was decided as input and the torque results were acquired during the trajectory tracking of the joints.

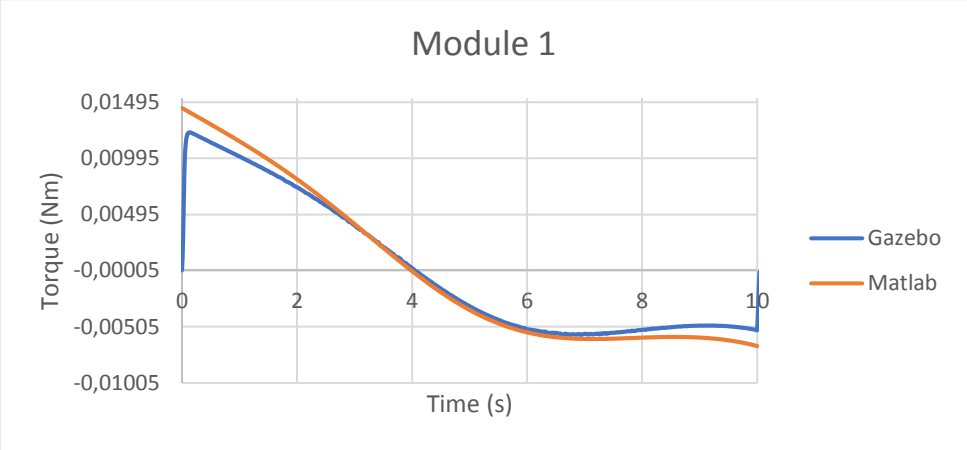


Figure 7.1 : Torque values for module 1 taken from both MATLAB and Gazebo.

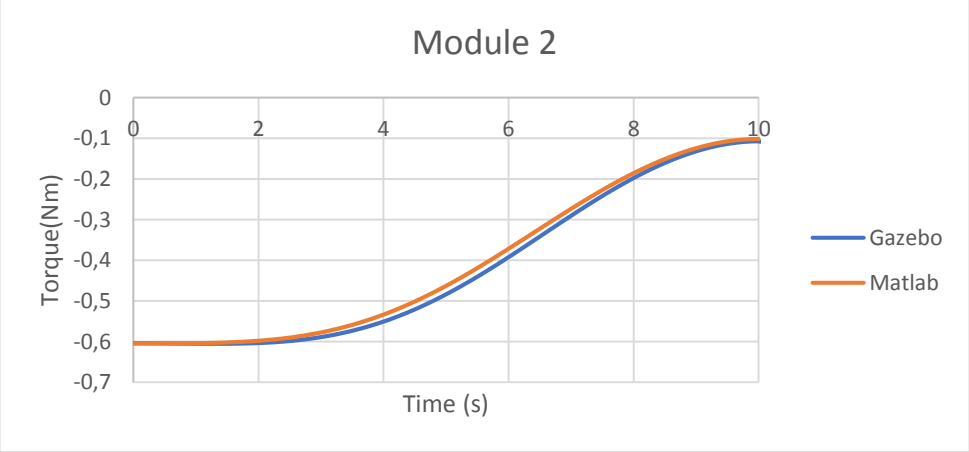


Figure 7.2 : Torque values for module 2 taken from both MATLAB and Gazebo.

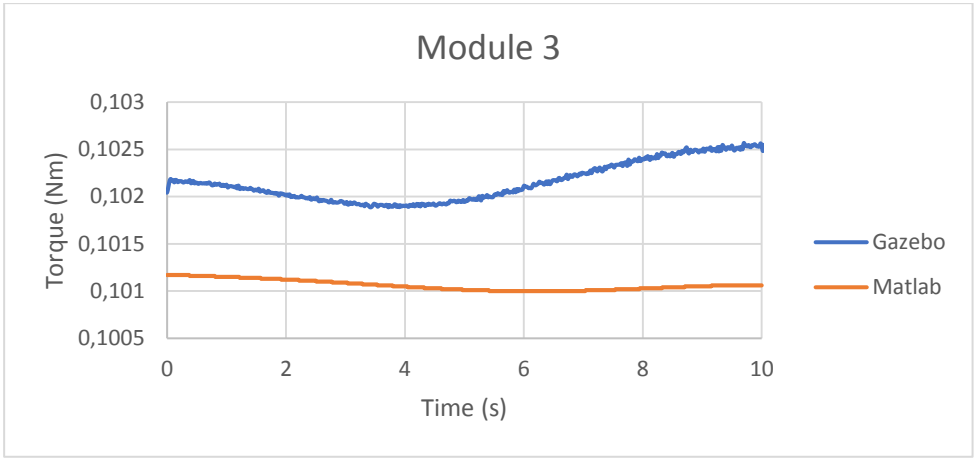


Figure 7.3 : Torque values for module 3 taken from both MATLAB and Gazebo.

According to the results obtained that are seen in Figure 7.1, Figure 7.2 and Figure 7.3, close torque values were observed both in MATLAB and Gazebo simulation. These results validate the Gazebo simulation model of the modular robot manipulator.

7.1.5 Controller results in simulation

In this section, control results of the modular robot manipulator in the Gazebo are evaluated. In order to evaluate the trajectory tracking performances of the modules, modular robot manipulator configuration which is given in Figure 3.1 was considered.

7.1.5.1 Velocity based joint trajectory controller

As a result of the implemented velocity based joint trajectory controller, tracking results were observed as they are given in Figure 7.4, Figure 7.5 and Figure 7.6.

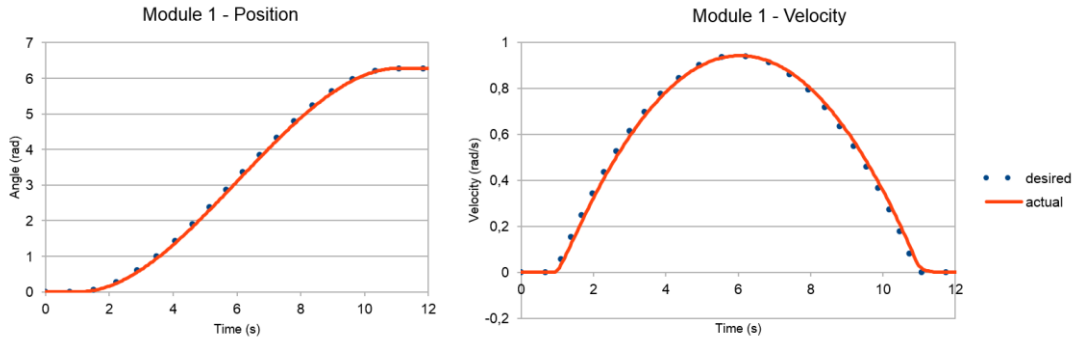


Figure 7.4 : Module 1 trajectory tracking results.

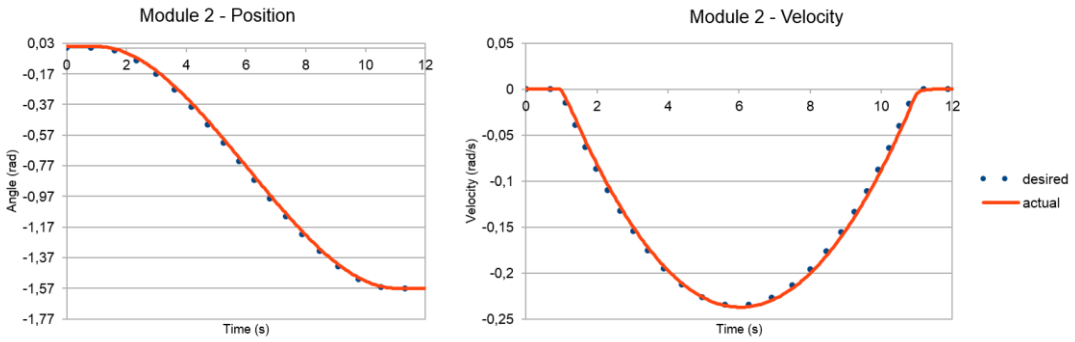


Figure 7.5 : Module 2 trajectory tracking results.

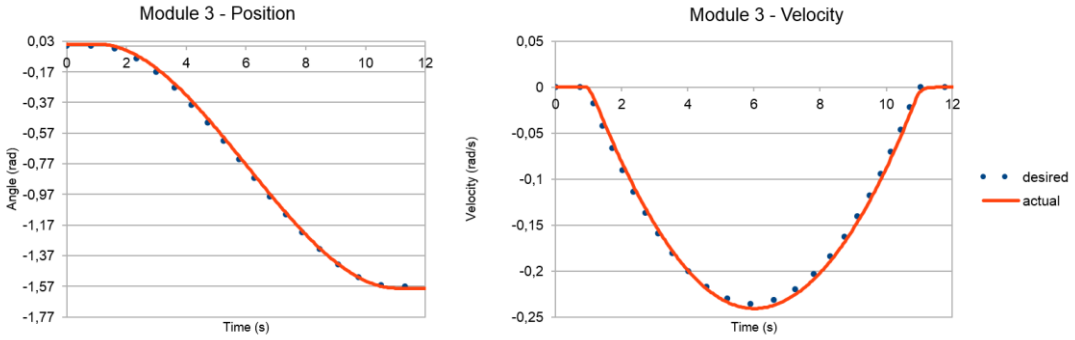


Figure 7.6 : Module 3 trajectory tracking results

In the figures above it can be seen that minimum position and velocity tracking errors were obtained with the controller.

7.1.5.2 Computed torque controller

Computed torque controller was created as a joint trajectory controller as it is discussed in section 5.3.4. For the given joint space trajectory, tracking results were obtained as they are seen in Figure 7.7, Figure 7.8 and Figure 7.9.

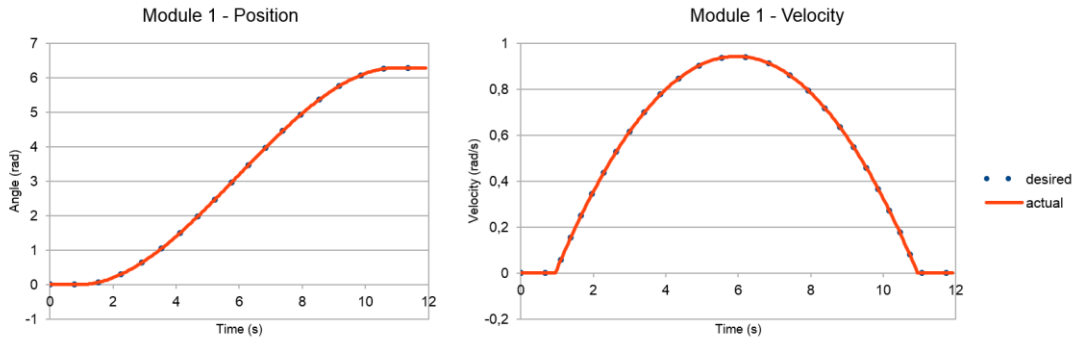


Figure 7.7 : Module 1 trajectory tracking results.

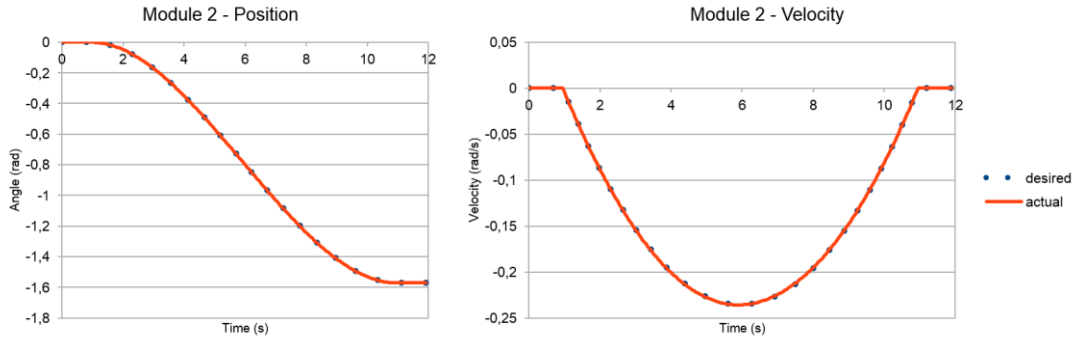


Figure 7.8 : Module 2 trajectory tracking results.

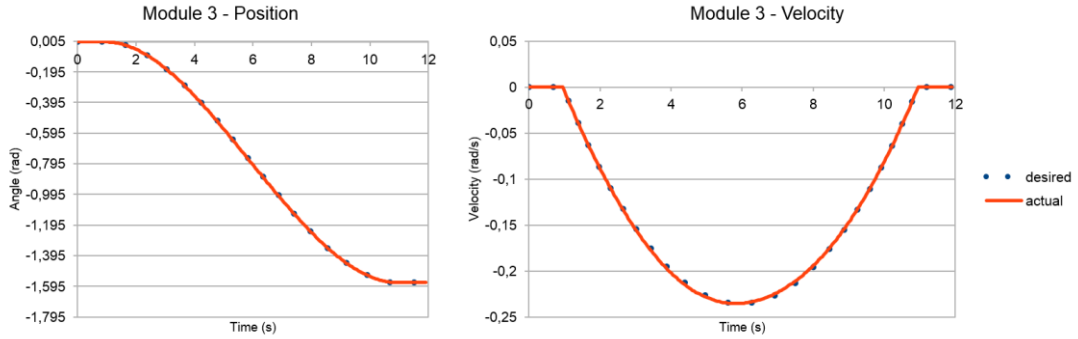


Figure 7.9 : Module 3 trajectory tracking results.

In the figures above it can be seen that position and velocity tracking performance were observed to be better than velocity based joint trajectory controller.

7.1.6 Velocity based joint trajectory controller tracking results on experimental setup

In this section, velocity based joint trajectory controller is evaluated on the experimental setup. 3 DOF and 2 DOF modular robot manipulator configurations were assembled as they are shown in Figure 6.9 and Figure 6.10.

In the Table 7.2 effects of PID coefficients on the closed loop system are given. By taking into account to these effects of PID coefficients, the joint trajectory controller was tuned. PID tuning process was conducted from tip module to base module.

Table 7.2 : Effects of PID coefficients

Response	**OVERSHOOT**	**STEADY-STATE ERROR**	**RISE TIME**	**SETTLING TIME**
K_p	Increase	Decrease	Decrease	Small Change
K_i	Increase	Eliminate	Decrease	Increase
K_d	Decrease	No Change	Small Change	Decrease

In order to tune inner loop controller of the tip module in the 3 DOF modular robot manipulator which is given in Figure 6.9, the following procedure was implemented. Tuning process was started from inner PI loop which is executed in DYNAMIXEL motor's built-in controller. First, K_p coefficient is determined such that overshoot was observed in step response. Step response of the controller with the different K_p coefficients was observed as in Figure 7.10.

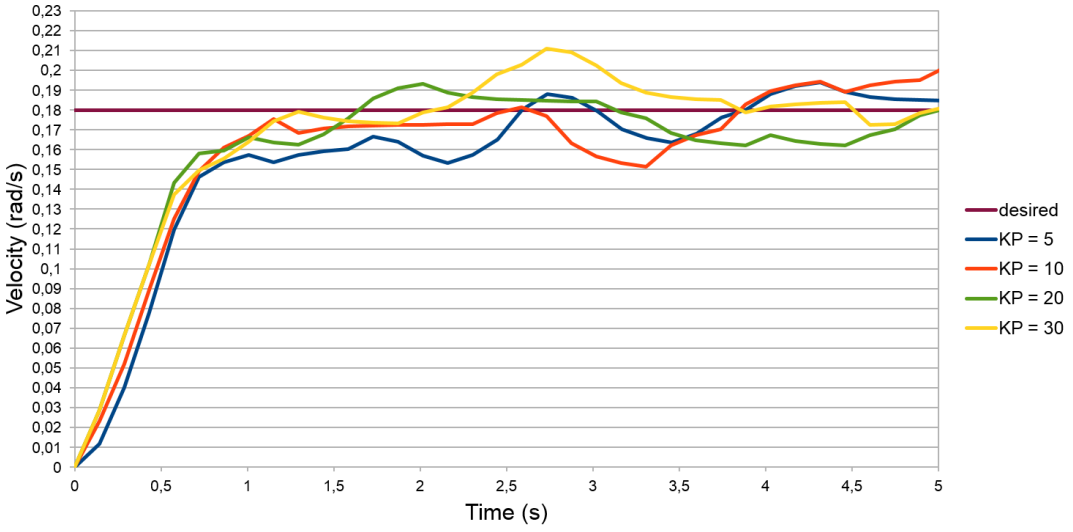


Figure 7.10 : K_p coefficient determination of inner loop controller.

According to the step response of the controller in Figure 7.10, K_p coefficient was determined as 20. Then in order to eliminate steady-state error, K_i coefficient was

determined when K_p was 20. Step response of the controller for this process is given in the Figure 7.11.

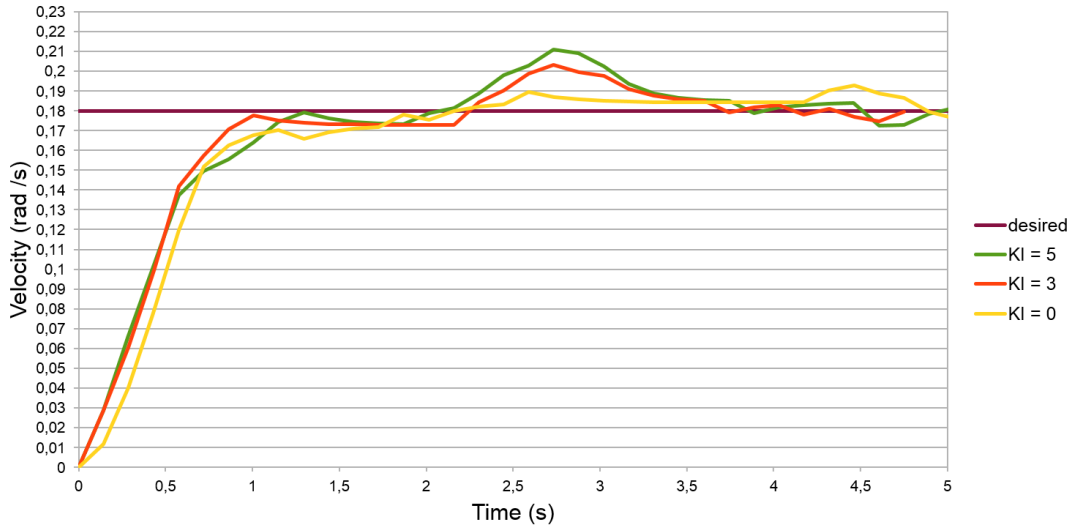


Figure 7.11 : K_i coefficient determination of inner loop controller.

According to the experiment results in Figure 7.11, K_i coefficient chosen as 3.

After PID parameters of the inner loop controller was determined, outer loop controller was tuned by starting with K_p coefficient selection. K_p coefficient was determined such that position tracking error minimized. Position and velocity tracking results for different K_p coefficients are given in Figure 7.12 and Figure 7.13.

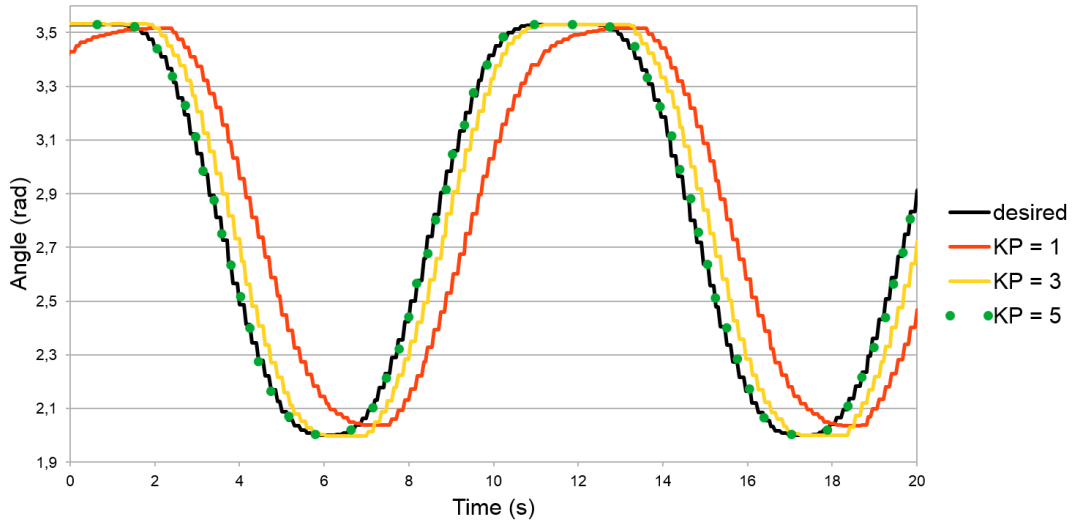


Figure 7.12 : Position tracking results of K_p coefficient determination of outer loop controller.

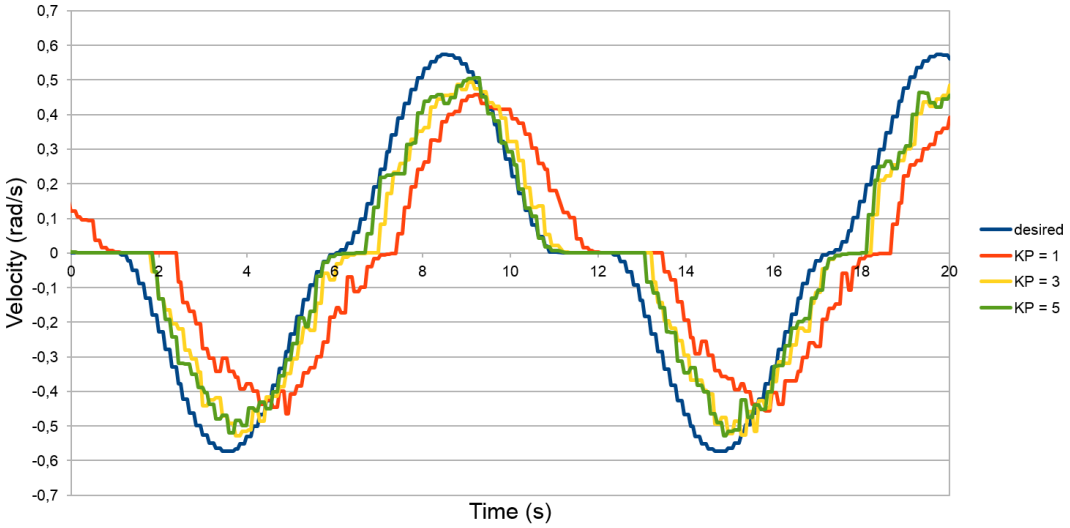


Figure 7.13 : Velocity tracking results of K_p coefficient determination of outer loop controller.

In Figure 7.12, minimum position tracking error was observed when K_p was 5 and this coefficient was chosen for outer loop controller.

Then in order to reduce velocity tracking error, K_d coefficient was determined when K_p was 5. In Figure 7.14 and Figure 7.15, effects of the K_d coefficient on position and velocity tracking are given.

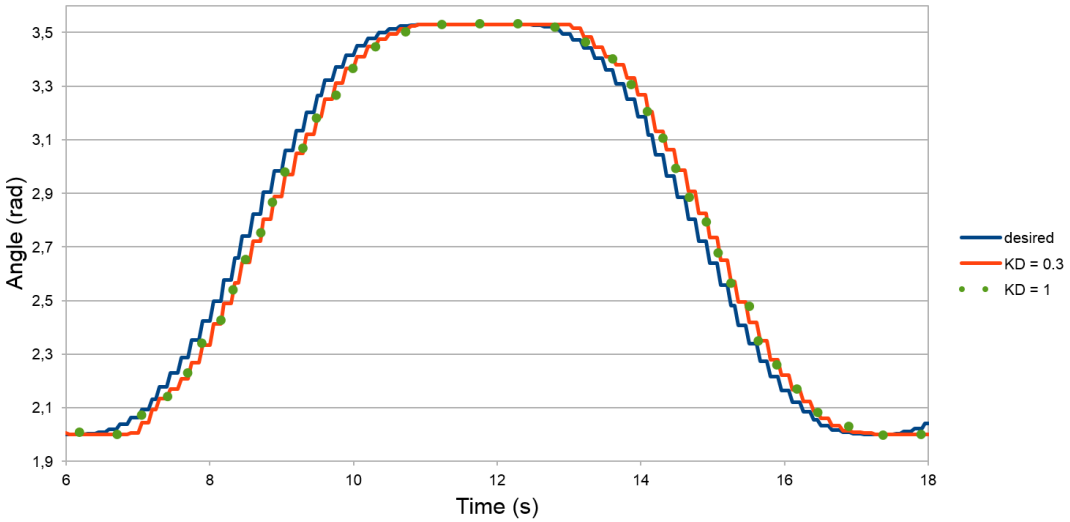


Figure 7.14 : Position tracking results of K_d coefficient determination of outer loop controller.

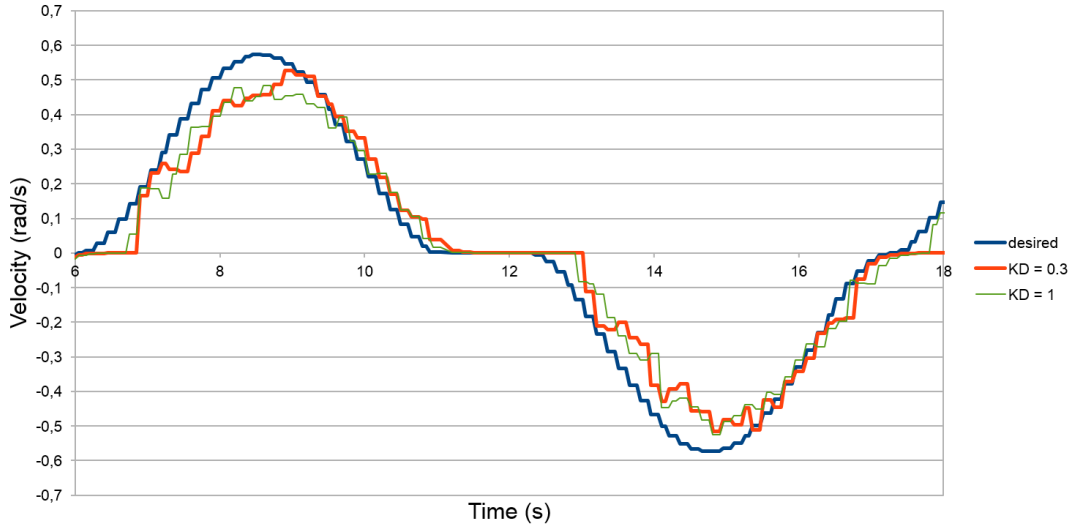


Figure 7.15 : Velocity tracking results of K_d coefficient determination of outer loop controller.

According to the experiment results, K_d coefficient was chosen as 1.

This tuning procedure was conducted for each module from farthest module to base module and trajectory tracking results are given in sections 7.1.6.1 and 7.1.6.2.

7.1.6.1 2 DOF modular robot

For 2 DOF robot configuration, velocity-based joint trajectory controller was executed with the pre-calculated joint space trajectories and the results were collected during the time trajectories executed. Trajectory tracking performance of the controller is shown in Figure 7.16 and Figure 7.17.

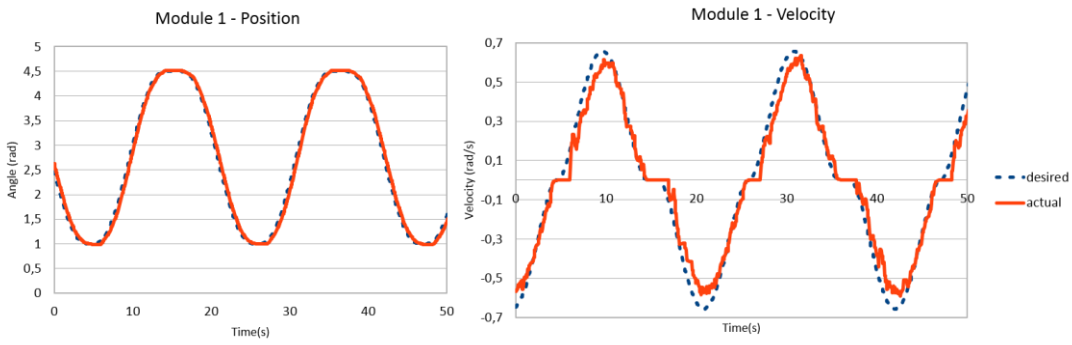


Figure 7.16 : Trajectory tracking results of the module 1.

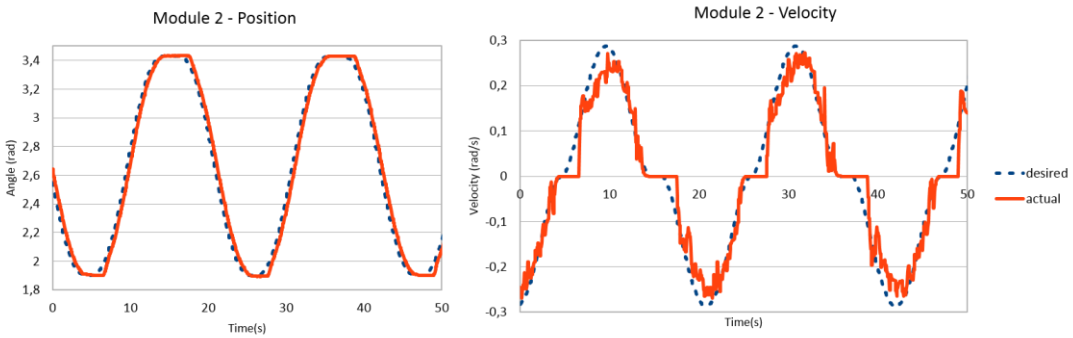


Figure 7.17 : Trajectory tracking results of the module 2.

7.1.6.2 3 DOF modular robot

For 3 DOF robot configuration, velocity-based joint trajectory controller was executed with the pre-calculated joint space trajectories and the trajectory tracking results are observed as in Figure 7.18, Figure 7.19 and Figure 7.20.

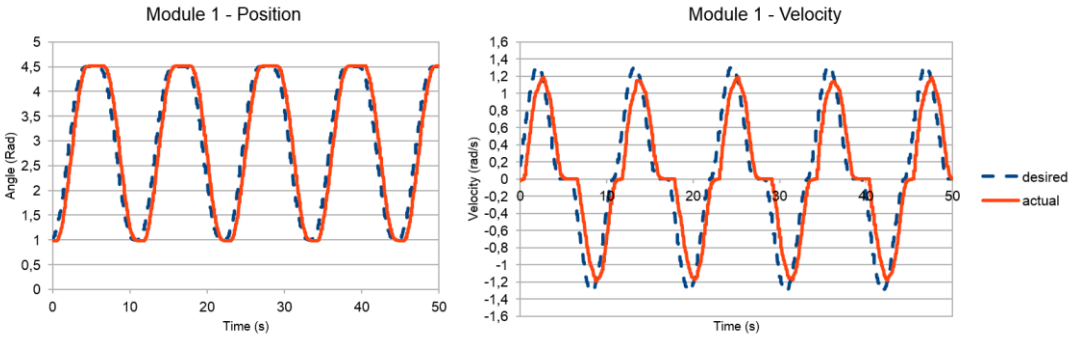


Figure 7.18 : Trajectory tracking results of Module 1.

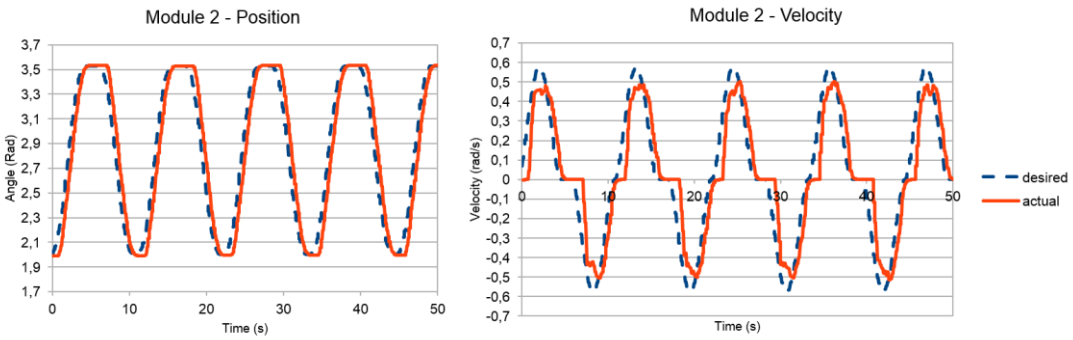


Figure 7.19 : Trajectory tracking results of Module 2.

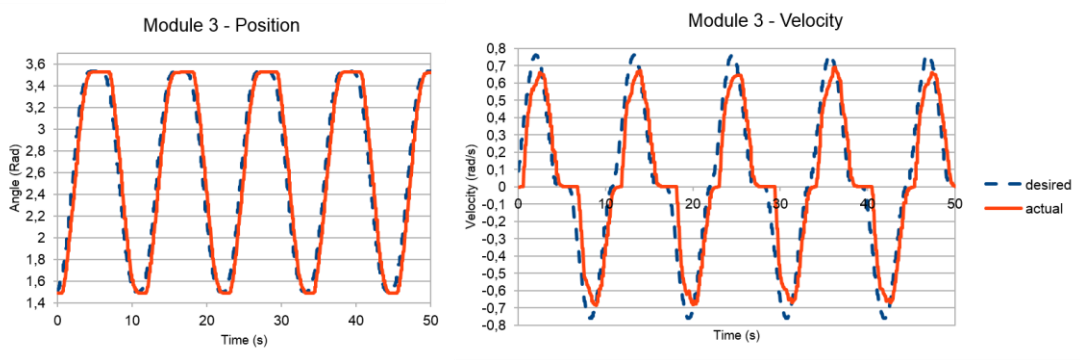


Figure 7.20 : Trajectory tracking results of Module 3.

7.1.7 Pick and place task

In this section, a pick and place task which is performed with experimental setup is discussed. Two separate points have been decided for pick and place application, one for picking and one for placing. A rubber was chosen as picked object and the object is fed from the fixed position at point A at each repetition. The position deviation where the end-effector placed the object at point B was observed by performing a repeated movement. This process is shown in the Figure 7.21.

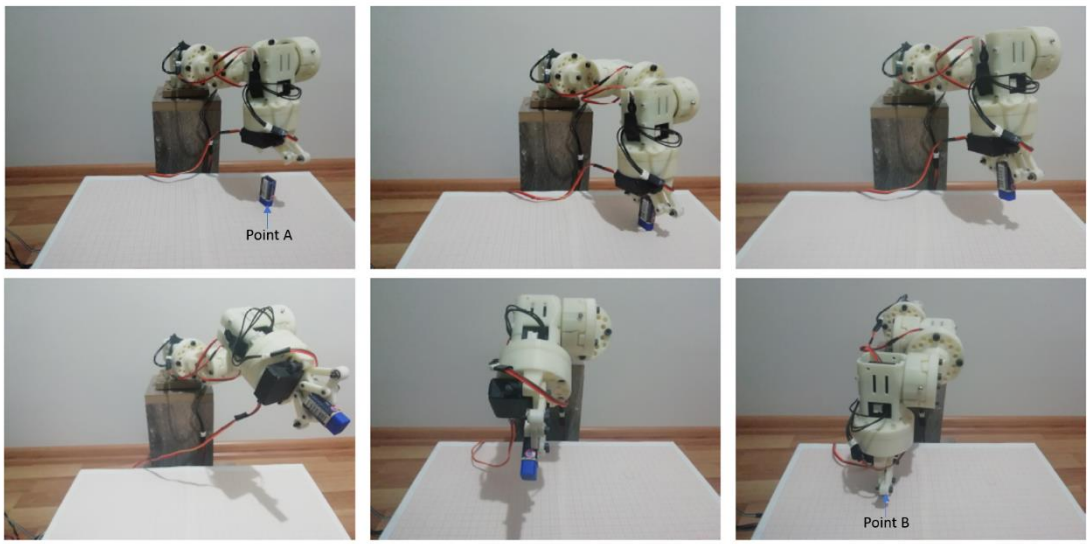


Figure 7.21 : Pick and place task.

The position deviation at the point B was calculated as Euclidian distance error and the result is given in the Figure 7.22.

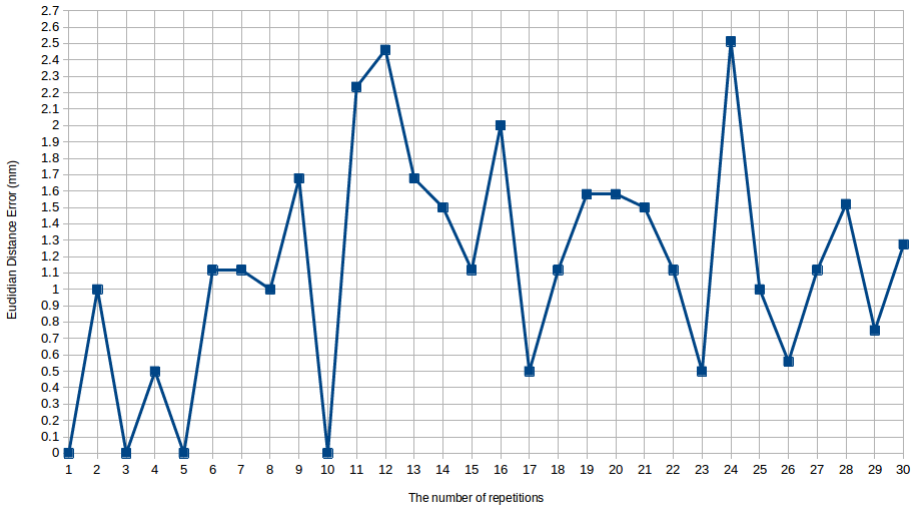


Figure 7.22 : Pick and Place task repeatitability result.

According to the Euclidian distance error data obtained during repeated pick and place task, maximum error was obtained as 2.5 mm as it is given in Figure 7.22.

8. CONCLUSION

In this thesis, new design of the modular robot manipulator was presented. Modularity of the robot manipulator was ensured with reconfigurable twist angle. Experimental setup was created with 3 active modules, 1 gripper module and 1 base module and the experiments conducted with 2 DOF and 3 DOF robot configurations as it is given in section 7.

Kinematics analyses were conducted for modular robot manipulator. In the inverse kinematics analysis section, analytical and numerical kinematics solvers were discussed and the results were given in section 7. Dynamics analyses of the modules were performed by using MATLAB and SOLIDWORKS softwares. According to the comparisons, small differences were observed between joint torque values obtained from MATLAB and SOLIDWORKS. SOLIDWORKS takes into account to friction when the material properties assigned to the parts however in the analysis conducted in MATLAB, friction was not considered. This difference was thought as it is caused by this friction effect. Maximum payloads for the 3 DOF, 4 DOF and 5 DOF modular robot manipulators were studied with static analysis.

After dynamic analyses were validated, URDF model of the modular robot manipulator was created and by using this model, simulated model of the modular robot manipulator created in GAZEBO physic engine. URDF model of the modular robot manipulator was validated by comparing joint torque values obtained from MATLAB and GAZEBO softwares when the robot manipulator tracing an example trajectory in section 7. URDF model of the modular robot manipulator enabled to investigate motion planning algorithms both in MATLAB and ROS.

Controllers for the modules of the modular robot manipulator were implemented in ROS. Joint trajectory controller was implemented on both simulation and experimental setup. It was tested with the output trajectories of the motion planning package of modular robot manipulator. Because that available controllers in the joint trajectory controller support only SISO systems, MIMO controller support was implemented with computed torque control method. Computed torque controller was created as joint

trajectory controller by using recursive inverse dynamic solver in the KDL library. Computed torque controller was tested with simulated robot model in Gazebo and the results were given in section 7. Because DYNAMIXEL MX64 servo motors does not support direct current control for DYNAMIXEL protocol 1.0, this controller could not implement on experimental setup. Direct current control was available for DYNAMIXEL MX64 with protocol 2.0. It is planned to implement current control in the inner controllers of the motors by updating software in the servo motor's inner controller to the DYNAMIXEL protocol 2.0 as a future work.

Motion planning implementation was conducted with MoveIt! package on ROS. Collision free motion planning for different modular robot manipulator configurations was realized. User interactions with the MoveIt! were eased with the developed GUI. Jog control was enabled from GUI and it was allowed to teach points to the robot and plan motions by using taught points. Database was created to store taught points. In order to test the performance of the MoveIt! a pick and place task was conducted as it is given in 7.1.7. Although point to point motion planning was returned with solutions, in the cartesian path planning, solutions were not always found.

According to the pick and place experiments conducted on the experimental setup, torque transmission problem was observed in the gripper module when high gripping forces were required. It was determined that problem was caused by mechanical connection between gripper servo motor and gripper finger. When high gripping forces are demanded, servo shaft slips, and it cannot transmit its full power to the fingers. This problem is considered to be one of the issues to be studied in the future.

REFERENCES

[1] Feczko, J., & Manka, M. (2015). Review of the modular self reconfigurable robotic systems. *10th International Workshop on Robot Motion and Control (RoMoCo)*, (pp. 182-187). Poznan. doi:10.1109/RoMoCo.2015.7219733

[2] Brunete, A., Ranganath, A., Segovia, S., Frutos, J. P., Hernando, M. & Gambao, E. (2017). Current trends in reconfigurable modular robots design. *International Journal of Advanced Robotic Systems*, 14, 1-21.

[3] Toshio, F., & Tsuyoshi, U. (1992). Concept of cellular robotic system (CEBOT) and basic strategies for its realization. *Computers & Electrical Engineering*, 18(1), 11-39.

[4] Yim, M. (1995). *Locomotion with Unit-Modular Reconfigurable Robot. Ph.D. Dissertation*. Stanford USA: Stanford University.

[5] Sanderson, G. J. (1997). TETROBOT: a modular approach to parallel robotics. *IEEE Robotics & Automation Magazine*, 4(1), 42-50.

[6] Pfotzer, L., & Ruehl, S. (2014). KAIRO 3: A modular reconfigurable robot for search and rescue field missions. *IEEE International Conference on Robotics and Biomimetics*, 205-210. Bali.

[7] Yang, B., & Han, L. (2015). A modular amphibious snake-like robot: Design, modeling and simulation. *IEEE International Conference on Robotics and Biomimetics*, (pp. 1924-1929).

[8] Tan W., & Wei, H. (2018). SambotII: A New Self-Assembly Modular Robot Platform Based on Sambot. *Applied Sciences*, 8(10).

[9] Pacheco, M., Fogh, R., Lund, H., & Christensen, D. J. (2015). Fable II: Design of a modular robot for creative learning. *IEEE International Conference on Robotics and Automation (ICRA)*, 6134-6139. Seattle. doi:10.1109/ICRA.2015.7140060

[10] Jia, X., Frenger, M., Chen, Z., Hamel, W. R., & Zhang, M. (2015). An alligator inspired modular robot. *IEEE International Conference on Robotics and Automation (ICRA)*, 1949-1954. Seattle.

[11] Romanishin, J. W., Gilpin, K. & Rus, D. (2013). M-blocks: Momentum-driven, magnetic modular robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4288-4295. Tokyo.

[12] Wright, C. et al. (2007). Design of a modular snake robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2609-2614. San Diego, CA, USA: IEEE. doi:10.1109/IROS.2007.4399617

[13] Yim, M., Roufas, K., Duff, D. et al. (2003). Modular Reconfigurable Robots in Space Applications. *Autonomous Robots*, 14(2-3), 225-237.

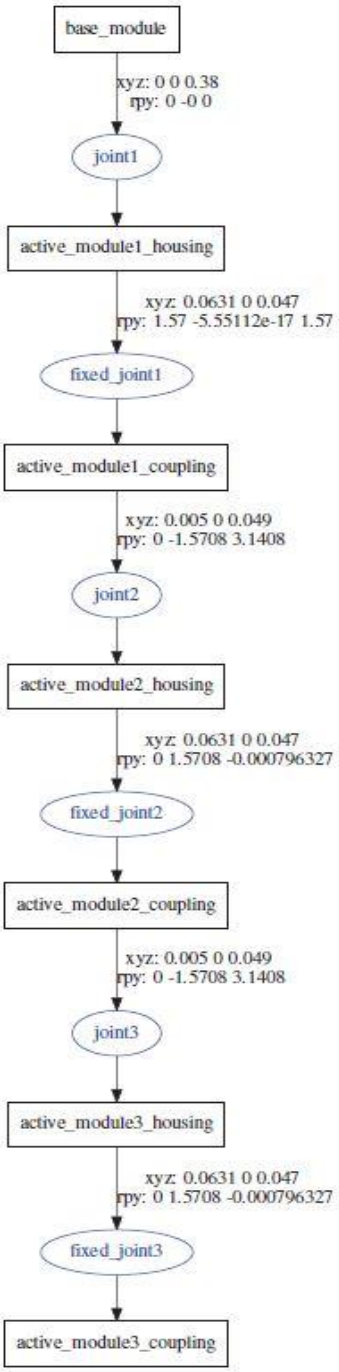
[14] Tremblay, T., & Padir, T. (2013). Modular Robot Arm Design for Physical Human-Robot Interaction. *IEEE International Conference on Systems, Man, and Cybernetics*, 4482-4487. Manchester.

[15] Yim, M. et al. (2007). Modular self-reconfigurable robot systems. *IEEE Robotics & Automation Magazine*, 14, 43-52.
[16] Acaccia, G., Bruzzone., L., & Razzoli, R. (2008). A modular robotic system for industrial applications. *Assembly Automation*, 28(2), 151-162.
[17] Pan, X., Wang, H., & Jiang, Y. (2013). Research on the Kinematic Calibration of a Modular. *International Conference on Mechatronics and Automation*. Takamatsu.
[18] Giusti, A., & Althoff, M. (2015). Automatic centralized controller design for modular and reconfigurable robot manipulators. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3268-3275. Hamburg. doi:10.1109/IROS.2015.7353831
[19] Valente, A. (2016). Reconfigurable industrial robots: A stochastic programming approach for designing and assembling robotic arms. *Robotics and Computer-Integrated Manufacturing*, 115-126.
[20] Quigley, M., Conley, K., Gerkey, B., Faust, J., Josh F., Foote, T., Leibs, J., Wheeler, R. (2009). *ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software*, 3.
[21] Bihlmaier, A. B. et al. (2015, 01). ROS-Based Cognitive Surgical Robotics. *Studies in Computational Intelligence*, 625. doi:10.1007/978-3-319-26054-9_12
[22] McKenzie, R. M., Baraclough., W. T. & Stokes, A. A. (2017). A Hybrid Pick and Place Arm. *Frontiers in Robotics and AI*, 4, 39. doi:10.3389/frobt.2017.00039
[23] Jeong, H. B., & Kang, K. (2016). Improved drone reliability using a robot operating system. *Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 151-153.
[24] Kaushik, K., Sriram, K., & Manimozhi, M. (2016). Gesture Based Control Using Windows Kinect. *Sensors & Transducers*, 196(1), 1-6.
[25] Hellmund, A., Wirges, Ö., Taş, Ş., Bandera, C. & Salscheider, N. O.. (2016). Robot operating system: A modular software framework for automated driving. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 1564-1570. doi:10.1109/ITSC.2016.7795766
[26] Cheng, H. H., & Ko, D. (2012). Programming reconfigurable modular robots. *Proceedings of 2012 IEEE/ASME 8th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, 160-165. doi:10.1109/MESA.2012.6275555
[27] Juan, H. S., & Mirats, T. J. M. (2006). Generic modular framework for robotic arm applications. In A. f. Machinery (Ed.), *International Conference on Computer Systems and Technologies*, 31-36.
[28] Craig, J. J. (2005). *Introduction to Robotics Mechanics and Control*. Pearson Education International.
[29] Aristidou, A., & Lasenby, J. (2009). Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver.
[30] http://wiki.ros.org/ROS/Concepts, access date: 06.07.2019
[31] http://wiki.ros.org/urdf/XML/model, access date: 06.07.2019
[32] https://moveit.ros.org, access date: 06.02.2018
[33] http://www.orocos.org/kdl, access date: 06.02.2018

[34] http://wiki.ros.org/rviz, access date: 06.04.2018
[35] Zahid, A. (2015), *Realization of dynamixel servo plant parameters to improve admittance control for a compliant human-robot interaction*. Theses. 263. https://digitalcommons.njit.edu/theses/263
[36] http://emanual.robotis.com/docs/en/dxl/mx/mx-64-2, access date:10.08.2019.
[37] Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Tsouroukdissian, A. T. et al.. (2017). ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, 456 - 456.
[38] http://www.robotis.us/dynamixelsdk, access date: 06.02.2018
[39] https://moveit.ros.org/documentation/planners/ acces date: 06.07.2019
[40] Moll, M., & Kavraki, L. E. (2013). The Open Motion Planning Library - OMPL. IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, Germany.
[41] Noreen, I., Khan, A., & Habib, Z. (2016). Optimal Path Planning using RRT* based Approaches: A Survey and Future Directions. *International Journal of Advanced Computer Science and Applications(IJACSA)*, 7(11).

APPENDIX

A - Graph View of the URDF Model of Modular Robot Manipulator



B - MoveIt! Configuration File for Modular Robot Manipulator

```
<?xml version="1.0" ?>
<robot name="modular_robot">
  <group name="manipulator">
    <joint name="joint1" />
    <joint name="joint3" />
    <joint name="joint5" />
    <chain base_link="base_link" tip_link="gripper_base_link" />
  </group>
  <passive_joint name="base_ll2_joint" />
  <passive_joint name="ll2_ll3_joint" />
  <passive_joint name="base_ll4_joint" />
  <passive_joint name="base_rl2_joint" />
  <passive_joint name="rl2_rl3_joint" />
  <passive_joint name="base_rl4_joint" />
  <disable_collisions link1="base_link" link2="link1" reason="Adjacent" />
  <disable_collisions link1="base_link" link2="link2" reason="Never" />
  <disable_collisions link1="gripper_base_link" link2="left_l2" reason="Adjacent" />
  <disable_collisions link1="gripper_base_link" link2="left_l3" reason="Never" />
  <disable_collisions link1="gripper_base_link" link2="left_l4" reason="Adjacent" />
  <disable_collisions link1="gripper_base_link" link2="link3" reason="Default" />
  <disable_collisions link1="gripper_base_link" link2="link4" reason="Default" />
  <disable_collisions link1="gripper_base_link" link2="link5" reason="Adjacent" />
  <disable_collisions link1="gripper_base_link" link2="right_l2" reason="Adjacent" />
  <disable_collisions link1="gripper_base_link" link2="right_l3" reason="Never" />
  <disable_collisions link1="gripper_base_link" link2="right_l4" reason="Adjacent" />
  <disable_collisions link1="left_l2" link2="left_l3" reason="Adjacent" />
  <disable_collisions link1="left_l2" link2="left_l4" reason="Never" />
  <disable_collisions link1="left_l2" link2="link3" reason="Default" />
  <disable_collisions link1="left_l2" link2="link4" reason="Never" />
  <disable_collisions link1="left_l2" link2="link5" reason="Never" />
  <disable_collisions link1="left_l2" link2="right_l2" reason="Default" />
  <disable_collisions link1="left_l2" link2="right_l3" reason="Never" />
  <disable_collisions link1="left_l2" link2="right_l4" reason="Never" />
  <disable_collisions link1="left_l3" link2="left_l4" reason="Never" />
  <disable_collisions link1="left_l3" link2="link2" reason="Default" />
  <disable_collisions link1="left_l3" link2="link3" reason="Default" />
  <disable_collisions link1="left_l3" link2="link4" reason="Never" />
  <disable_collisions link1="left_l3" link2="link5" reason="Never" />
  <disable_collisions link1="left_l3" link2="right_l2" reason="Never" />
  <disable_collisions link1="left_l3" link2="right_l3" reason="Never" />
  <disable_collisions link1="left_l3" link2="right_l4" reason="Never" />
  <disable_collisions link1="left_l4" link2="link3" reason="Never" />
  <disable_collisions link1="left_l4" link2="link4" reason="Never" />
  <disable_collisions link1="left_l4" link2="link5" reason="Never" />
  <disable_collisions link1="left_l4" link2="right_l2" reason="Never"
```

```
<disable_collisions link1="left_l3" link2="right_l3" reason="Never" />
   <disable_collisions link1="left_l3" link2="right_l4" reason="Never" />
   <disable_collisions link1="left_l4" link2="link3" reason="Never" />
   <disable_collisions link1="left_l4" link2="link4" reason="Never" />
   <disable_collisions link1="left_l4" link2="link5" reason="Never" />
   <disable_collisions link1="left_l4" link2="right_l2" reason="Never" />
   <disable_collisions link1="left_l4" link2="right_l3" reason="Never" />
   <disable_collisions link1="left_l4" link2="right_l4" reason="Never" />
   <disable_collisions link1="link1" link2="link2" reason="Adjacent" />
   <disable_collisions link1="link1" link2="link3" reason="Never" />
   <disable_collisions link1="link1" link2="link4" reason="Never" />
   <disable_collisions link1="link1" link2="link5" reason="Never" />
   <disable_collisions link1="link2" link2="link3" reason="Adjacent" />
   <disable_collisions link1="link2" link2="link4" reason="Never" />
   <disable_collisions link1="link2" link2="link5" reason="Never" />
   <disable_collisions link1="link2" link2="right_l3" reason="Never" />
   <disable_collisions link1="link2" link2="right_l4" reason="Never" />
   <disable_collisions link1="link3" link2="link4" reason="Adjacent" />
   <disable_collisions link1="link3" link2="link5" reason="Default" />
   <disable_collisions link1="link3" link2="right_l2" reason="Never" />
   <disable_collisions link1="link3" link2="right_l3" reason="Never" />
   <disable_collisions link1="link3" link2="right_l4" reason="Never" />
   <disable_collisions link1="link4" link2="link5" reason="Adjacent" />
   <disable_collisions link1="link4" link2="right_l2" reason="Never" />
   <disable_collisions link1="link4" link2="right_l3" reason="Never" />
   <disable_collisions link1="link4" link2="right_l4" reason="Never" />
   <disable_collisions link1="link5" link2="right_l2" reason="Never" />
   <disable_collisions link1="link5" link2="right_l3" reason="Never" />
   <disable_collisions link1="link5" link2="right_l4" reason="Never" />
   <disable_collisions link1="right_l2" link2="right_l3" reason="Adjacent" />
   <disable_collisions link1="right_l2" link2="right_l4" reason="Never" />
   <disable_collisions link1="right_l3" link2="right_l4" reason="Never" />
</robot>
```

C - Numerical Inverse Kinematic Test Program

```
KDL::Chain chain = urdf
fk = calculateFK()
for count < 1000:
        target = random(fk)
        ik_seed_state = getRandomJointPose()
        ik = calculateIK()
        if ik < 0 and num_try < 5:
                num_try++
                ik_seed_state = getRandomJointPose()
                ik = calculateIK()
KDL::Tree tree;
kdl_parser.treeFromUrdfModel(urdf, tree);
KDL::Chain chain
tree.getChain(base_link, end_effector, chain);
KDL::ChainFkSolverPose_recursive fk_solver(chain);
for count < 1000:
        KDL::JntArray q;
        q << randomJointsValues(n_dof);
        KDL::Frame target;
        fk_solver.JntToCart(q, target);
        ik_seed_state = getRandomJointPose();
        ik = calculateIK();
        if ik < 0 and num_try < 5:
                num_try++
                ik_seed_state = getRandomJointPose();
                ik = calculateIK();
        if ik < 0 and num_try < 5:
                num_try++
                ik_seed_state = getRandomJointPose();
                ik = calculateIK();
```

D - Determinant of the Matrix with Singular Value Decomposition Method in MATLAB

Calculating determinant of the matrix using SVD method in MATLAB :

```
[r,c]=size(A);
[u,s,v]=svd(A);
if r==1 | c==1
        s=s(1);
else
        s = diag(s);
end
        d=det(u)*prod(s)*det(v');
```

CURRICULUM VITAE

Personal Information

Name Surname: Aytaç KAHVECİ

Date of Birth: 13/10/1993

Place of Birth: Izmir (Turkey)

Education:

- 17/09/2011–13/06/2016 Bachelor's degree in Mechatronics Engineering

  Marmara University, Istanbul (Turkey)

Work Experience

- 24/07/2018–Present R&D Engineer

  MND Izolasyon LTD. Manisa (Turkey)

  Software development and control design for Automated Guided Vehicles.

- 07/07/2017–24/07/2018 Researcher

  Izmir Katip Celebi University, Izmir (Turkey)

  Programmable Working Space Untethered Electromagnetic Actuator Design and Control. Visual servoing with KUKA robot and system synchronization by using ROS.

- 10/11/2016–13/04/2017 Researcher

  Dokuz Eylül University, Izmir (Turkey)

  Thermal conductivity measurement system for liquids. Full stack web development in embedded Linux system.

Awards

Second place in Mechatronic Engineering Department in bachelor’s degree.

List of Publications

- Design and Development of a Surgical Robotic Hand with Hybrid Structure (TIPTEKNO18 - 05/12/2018)
- Electromagnet design for untethered actuation system mounted on robotic manipulator (Sensors and Actuators A: Physical - 01/12/2018)
- Guided Motion Control Methodology for Microrobots (CEIT 2018 - 25/10/2018)